NOTE TO USERS

AN ADAPTIVE M-ALGORITHM BASED CONVOLUTIONAL DECODER

Seyed Ali Gorji Zadeh

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

April 2005

Your file Votre référence
ISBN: 0-494-04371-7
Our file Notre référence
ISBN: 0-494-04371-7

ABSTRACT

AN ADAPTIVE M-ALGORITHM BASED CONVOLUTIONAL DECODER

Seyed Ali Gorji Zadeh

The Viterbi algorithm is one of the most popular convolutional decoders. This algorithm suffers from the high complexity in the decoding of the long constraint length codes. The M-algorithm is a simplified Viterbi algorithm and it is practical for the decoding of the long constraint length codes but it suffers from catastrophic error caused by the correct path loss in the algorithm. In this thesis we propose two different ways of the correct path recovery based on M-algorithm convolutional decoder.

The first method is called Ancestor Based Survivor Decision in M-algorithm Convolutional Decoder. We propose a survivor decision not only based on the path metric but also based on the path ancestor metric. This algorithm has been designed for the systems with an abrupt noise. Simulation results for the Additive White Gaussian Noise (AWGN) channel will show slightly improved error performance in some cases since the AWGN does not act as abrupt noise.

For the AWGN channels we propose another method which is called Adaptive M-algorithm Based Convolutional Decoder. In this method, we suggest using small number of survivors for most of the decoding attempts and we use higher number of survivors only in case of error decoding. The Cyclic Redundancy Check (CRC) error detection code is used to detect if the frame is an erroneous frame. Monte-Carlo simulation for the AWGN

channel shows that in most of the cases the error performance of the proposed algorithm outperforms the Viterbi algorithm or the conventional M-algorithm error performance.

Dedicated to my PARENTS, Sisters and Brother

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude and thanks to my supervisor Dr. Mohammad Reza Soleymani for his generous help and constant support. I appreciate the way Dr. Soleymani devoted his precious time to discuss with me the details of my research when it was in either up or down of its difficulties which was both encouraging and inspiring. His goal has always been in place to keep the research environment friendly and stress free.

My heartfelt thanks to my dearest father, mother, my sisters and my brother, who supported me with their love and their supports.

I would like to thank Mr. Javad Haghighat, Mr. Pouriya Sadeghi, Mr. Mohsen Ghotbi, and all of my friends at the Wireless and Communication Laboratory for their valuable advice and suggestions.

I also extend my thanks to the faculty and staff members of the Electrical and Computer Engineering Department, and to my respected defense committee Dr. Shayan, Dr. Haghighat and Dr. Hamouda.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ARQ	Automatic Repeat Request
AVA	Adaptive Viterbi Algorithm
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CCITT	Comité Consultatif International de Télégraphique et Téléphonique
CRC	Cyclic Redundancy Check
d_{free}	Free distance
DMC	Discrete Memoryless Channel
d_{min}	Minimum distance
E_b	Bit Energy
FER	Frame Error Rate
IEEE	Institute of Electrical and Electronics Engineers
LAN	Local Area Network
LDPC	Low Dencity Parity Check
MAN	Metropolitan Area Network
MAP	Maximum A Posteriory Probability
ML	Maximum Likelihood
N_0	Noise Power Spectral Density
RSSD	Reduced State Sequence Detection
SA	Search Algorithm
VA	Viterbi Algorithm

Chapter 1

INTRODUCTION

In 1948, Shannon [1] in a landmark paper showed that, using a proper channel coding for the information, the errors induced by a noisy channel can be reduced to any desired level as long as the rate of the information transmission over the channel is less than the capacity of the channel. This capacity of the channel for the Additive White Gaussian channel is defined by the formula below.

$$C = W \log_2\left(1 + \frac{S}{N}\right) \qquad (1.1)$$

where W is the bandwidth of the channel in Hertz, S is the signal power and $N = WN_0$ is the variance of the Gaussian noise of the channel.

1.1 Error Control Codes

Since Shannon's work much effort has been performed to find encoding and decoding algorithms which can control the error of the communication systems in noisy environments. The coding system designed for the purpose of error control of the information is called channel coding. In the channel encoding, redundant information is added to the original source data, to enable the decoder to estimate the original data from the noisy received data from the channel. A simplified model of the channel coded system for digital communications is shown in Figure 1-1.

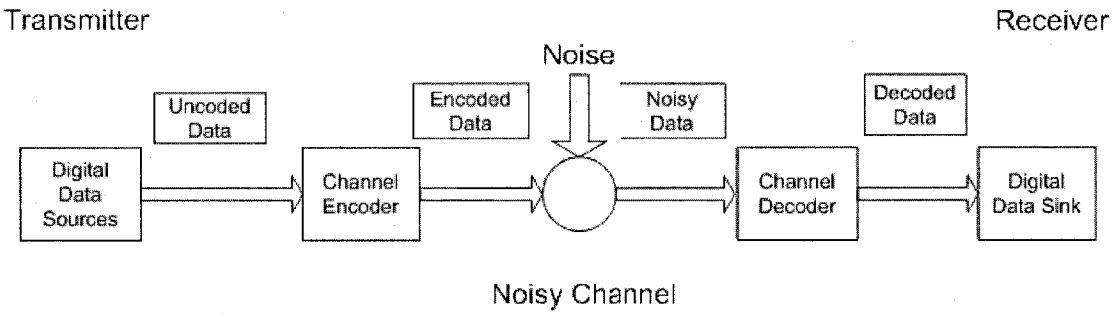

Figure 1-1 Simplified model of a channel coded system.

As it is seen in Figure 1-1 the digital data from the data source is sent to the channel encoder and the encoded data are sent to the receiver through a noisy channel. In the

receiver channel decoder estimates the original sent data from the received data using the redundant information added in the channel encoder.

1.2 Types of Channel Codes

There are two structurally different types of channel codes which are common in today's applications: *Block Codes* and *Convolutional Codes.*

In a block code the information sequence is divided into blocks of k information bits or symbols. A block of k information symbols is represented by a k-tuple $\mathbf{u}= (u_0, u_1,..., u_{k-1})$ and it is called a message. For the binary symbols, the total number of different messages will be 2^k. The block encoder transforms each message $\mathbf{u}$ independently into an n-tuple $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ of discrete symbols which is called a *codeword.* Hence, corresponding to the number of messages, there are 2^k different possible codewords at the output of the encoder. This set of the 2^k n-tuple codewords is called (n, k) *block code* and the ratio $R = k / n$ is called *code rate.* Code rate can be interpreted as the number of information symbols entering the encoder per each transmitted symbol from the encoder output. Since in the block code each n-symbol output is codeword depends only on the corresponding k-symbol input of the encoder, the encoder is memoryless and it can be implemented with a combinational logic circuit [2].

In a binary code, both the message and the codeword are binary. In a binary code, when $k < n$, $n - k$ redundant bits are added to number of the message bits to form the n-tuple codeword. These redundant bits enable the code to combat the channel noise. How to use the redundant information to provide a reliable transmission is art of code designing.

The encoder of a convolutional code transforms information sequence **u** consisting of k-symbol blocks to the encoded sequence **v** of n-symbols blocks. However, in the convolutional encoder, each n-symbol encoded block, not only depends on the corresponding k-symbol at the same time but also on K previous message blocks. Therefore, the encoder has a *memory order* of K. The set of the all possible encoded sequences forms the code. The ratio $R = k / n$ is called the *code rate* of the convolutional code. Since the encoder contains memory, it should be implemented by a sequential logic circuit.

In a binary convolutional code where input and output symbols are binary, when $k < n$ or $R < 1$, the redundant bits have been added to the code sequence for combating the channel noise. In the convolutional encoders typically n, k are small integers and more redundancy is added by increasing the memory order or K while n, k have been holding fixed and as a result R is fixed. How to use the memory to achieve a reliable transmission over the noisy channel is the art of the convolutional code designer.

As mentioned above, unlike the block coding where blocks of information bits are mapped to the blocks of code symbols, in the convolutional coding a sequence of information bits is continuously mapped to a sequence of encoded symbols. This mapping is very structural and it can be argued that convolutional coding can offer a larger coding gain than block coding with the same complexity. Therefore, we will concentrate more on the convolutional codes here.

1.3 Convolutional Code and Convolutional Decoders

One of the differences between convolutional codes and block codes is that in convolutional codes, encoder contains memory and the block of n encoder outputs at any given time unit is not only dependent on the block of k inputs of that time, but also it depends on the K other previous input blocks. Usually a convolutional code is shown with a triple of (n, k, K). In convolutional encoder, blocks of n-outputs generated from blocks of k-input in a linear sequential circuit with memory with the length of K. Typically, n, k are small integers and $k < n$, while, K is memory order of the convolutional code and for obtaining low error probabilities it should be taken a large number.

From the introduction of convolutional codes by Elias [3] in 1955 different decoding schemes have been proposed for the decoding of the convolutional codes. Sequential decoding by Wozencraft [3] and threshold decoding by Massey [5] introduced before Viterbi decoding algorithm [6]. Viterbi algorithm that is a maximum likelihood decoding scheme could be easily implemented for codes with small memory. Viterbi decoding algorithm together with improved versions of sequential decoding, led to applications of the convolutional codes in space and satellite communications in the 1970s. In the next section we will describe the encoding mechanism of convolutional codes.

Although the Viterbi algorithm is a very popular convolutional decoding algorithm and it is an Maximum Likelihood (ML) optimum decoder, it suffers from the high complexity in the decoding of the long constraint length codes. On the other hand, we know that for obtaining lower error rates in the decoding of the convolutional codes, we

need to increase the constraint length of the employed code. Therefore, we should find simpler decoding algorithms for the decoding of the longer constraint length codes.

Due to the low complexity of the simplified Viterbi algorithms or the reduced state trellis decoding algorithms, they are practical for the decoding of the long constraint length codes. The M-algorithm is one of the reduced state trellis decoder, but M-algorithm suffers from catastrophic error caused by the correct path loss in the algorithm.

In this thesis we will study the methods of the correct path recovery based on the M-algorithm convolutional decoder.

1.4 Organization of the Thesis

Chapter 2 presents introduction to the Convolutional codes and their importance in communication systems. Also some different representations of the convolutional encoders like the trellis and the state diagram is brought in this chapter. Then, the maximum likelihood decoding of the convolutional codes and the concept of the Viterbi algorithm as a Maximum Likelihood algorithm will be addressed. Stack algorithm and M-algorithm will also be briefly explained. In Chapter 3, the Cyclic Redundancy Check (CRC) Codes will be briefly introduced and it will be shown how easily they can be implemented. In Chapter 4, the procedure of decoding in the M-algorithm convolutional decoder will be explained and also the catastrophic error caused by the correct path loss, in the reduced state trellis decoders and the M-algorithm will be introduced. Then two different sources of noise causing the correct path loss will be addressed and a survivor decision scheme in the M-algorithm to avoid the correct path loss caused by the abrupt noise will be proposed. Then the proposed algorithm is simulated for the Additive White

Gaussian Noise channel. In Chapter 5, the Adaptive M-algorithm Convolutional Decoder will be proposed. In this algorithm the number of survivors of the M-algorithm is increased only when an error in the decoded frame is detected. CRC error detection codes are used for the error check of the decoded frames. Simulation results for the communication channel with AWGN, and the convolutional codes with the different lengths, combined with the different CRC code lengths, will show the error performance and the complexity of the proposed algorithm. Finally, Chapter 6 will be conclusion of the thesis.

Chapter 2

CONVOLUTIONAL CODES AND THEIR DECODING METHODS

In a communication system when information is transmitted to the receiver side, it is corrupted by the channel noise. This causes error in the received data. To correct these errors, redundant bits are added to the data bits. These redundant bits should be selected in a way that can be exploited efficiently to correct the errors. Channel coding uses this controlled redundancy to detect and correct errors. The error control coding method is chosen based on the system requirements and the nature of the channel.

Channel codes can be divided into linear and non-linear. A code is called linear when the set of code vectors is a vector space; otherwise it is non-linear [6]. A code can also be binary or non-binary. When the linear code is a block code the k bit message u is multiplied by the generator matrix G to generate the codeword v. However in convolutional codes the codeword can be simply generated using a finite state machine [6]. The *weight* of a codeword is equal to the number of its non zero elements. For example the weight of (0 0 0 1 0 1 0) is two. The *minimum distance* $d_{\min}$ of a code is the minimum number of bits that must change in any codeword to produce another codeword. The greater the minimum distance, the more powerful the code can be. The *Hamming distance* (also known as the *distance*) between two codewords is defined to be the number of elements that are different between them. For example, the *Hamming distance* between (0 1 0 0) and (1 1 1 0) is two.

Shannon proved [1] that each channel has a capacity of C, that cannot be surpassed regardless of how good the code is. He also proved that as long as the transmitted information rate, R, is less than the channel capacity, there exists some code that makes the probability of error tend to zero. Recently some codes, *e.g.*, turbo code [8] and low density parity check (LDPC) code [14] codes have been shown to approach the capacity of several channels, using techniques such as soft decision iterative decoding and code concatenation.

2.1 Convolutional Codes

A convolutional code is represented by three parameters of (n,k,K) where K is called the constraint length and is defined as

$$K = \underset{i}{Max}\ m_i + 1 \qquad (2\text{-}1)$$

where m_i is the number of shift registers (memories) in i^{th} input branch. The convolutional encoder takes k input bits and outputs n bits by modulo-2 addition of the input information bits and the contents of the shift registers. If the information sequence appears in the output, the code is called systematic; otherwise it is called a non-systematic code. The convolutional code can be with or without feedback. The encoder of a convolutional code with $k=2$ and $n=3$ is shown in Figure 2-1. The encoded data is usually divided to blocks of fixed length, before transmission. Sometimes after each block, $K-1$ zero bits are appended to reset the memory. These bits are called tailing bits [7]. If no tail biting [7] is used, the code rate is:

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

Using tail-biting slightly reduces the code rate [7].

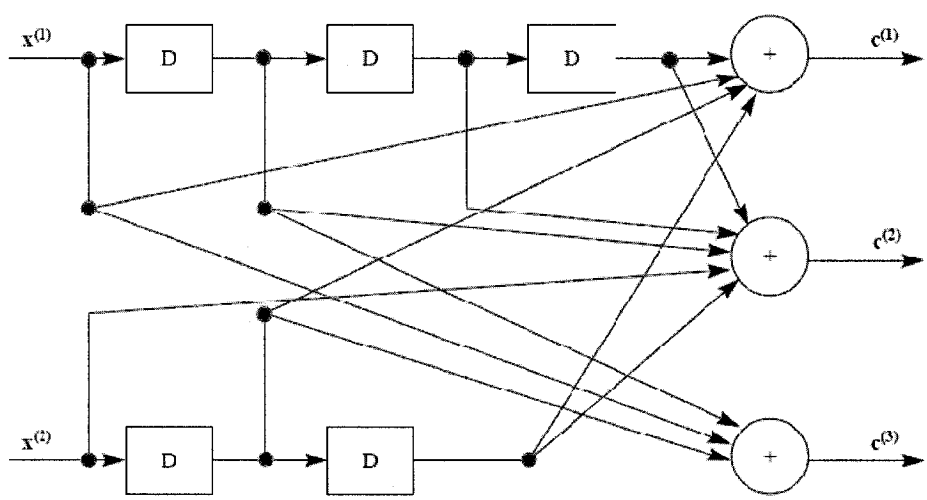

Figure 2-1 The encoder of a convolutional code with K=2 and n=3.

The convolutional encoder can be represented in different ways including: generator representation, tree diagram representation, trellis diagram representation, and state diagram representation. Since these representations will be used in the sequel, we briefly describe each of them.

For simplicity we consider the (2,1,3) convolutional encoder of Figure 2-2. The generator representation of a (n,k,K) code consists of a set of generator polynomials $\{g_{ij}(D)\}, 1 \le i \le k, 1 \le j \le n$, where $g_{ij}(D)$ represents the connection of i^{th} input to j^{th} output. For the convolutional encoder of Figure 2-2:

$$g_{11}(D) = 1 + D + D^2 \qquad (2\text{-}2)$$

$$g_{12}(D) = 1 + D^2 \qquad (2\text{-}3)$$

where operator D shows the delay. The generator polynomials can be represented as binary sequences for simplicity. For example (2-2) and (2-3) can be simply shown as $g_{11} = 111$, $g_{12} = 101$. When $k = 1$, g_{11} to g_{1n} can be shown as g_1 to g_n.

The tree diagram representation shows all possible encoded sequences on a tree. Let the solid line shows input bit 0 and a dashed line show an input bit1. The tree diagram of convolutional code of Figure 2-2 is shown in Figure 2-3. Parameter t shows time unit and the digits on each branch represent the output values corresponding to the input bit. Having the input sequence, the output sequence can be found from the tree diagram. For example let the input sequence to the encoder of Figure 2-2 be 1001, then from Figure 2-3 we see that the output sequence is 11101111.

The convolutional encoder can be also represented using different states of shift registers. For example in Figure 2-2 each shift register can take values 0 or 1, thus 4 different states exist. The state diagram shows the transitions from one state to the other as well as the corresponding outputs. The state diagram of convolutional encoder of Figure 2-2 is shown in Figure 2-4. For example from Figure 2-4 we observe that an input bit of 1 leads to a transition from state 00 to 10 and the output sequence is 11.

A trellis diagram is redrawing of the state diagram, considering the effect of time. Trellis diagram of encoder of Figure 2-2 is shown in Figure 2-5. Usually we assume that the encoding begins from state 00 as shown in Figure 2-5.

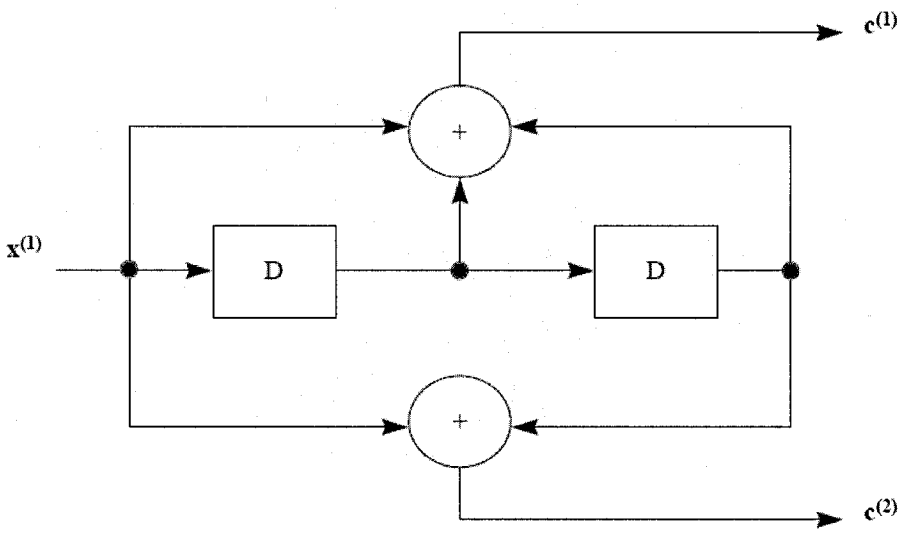

Figure 2-2 Encoder of a (2,1,3) convolutional code.

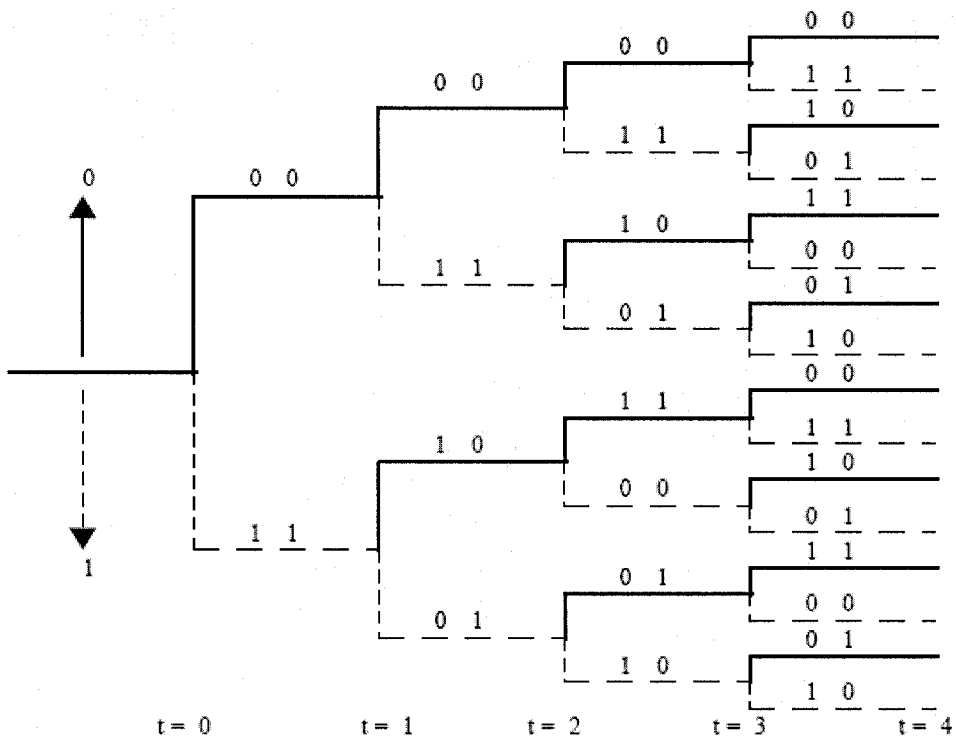

Figure 2-3 Tree diagram of convolutional code of Figure 2-2.

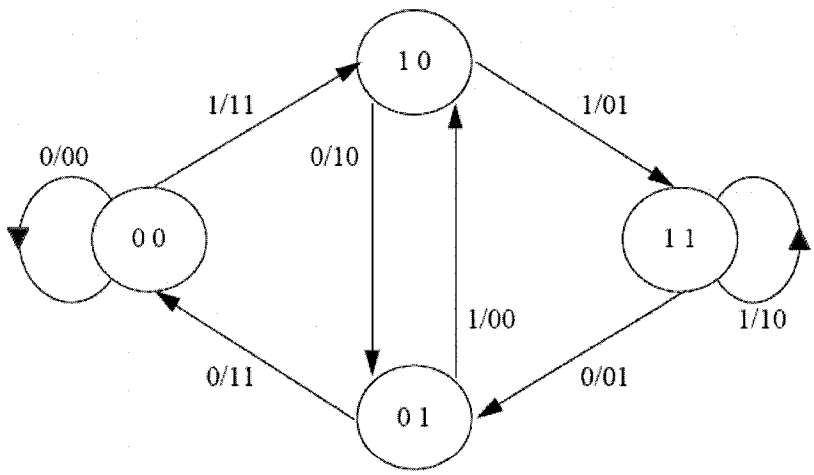

Figure 2-4 State diagram of encoder of Figure 2-2.

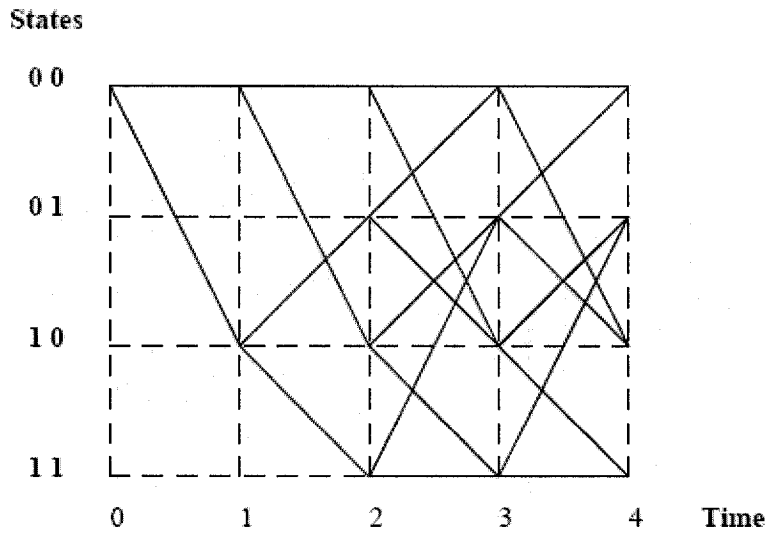

Figure 2-5 Trellis diagram related to the encoder of Figure 2-2.

2.2 Decoding of Convolutional Codes

Since each convolutionally encoded output information at time t is dependant on input information at time t - m to t, output information symbols are dependant on each other. Consequently, for achieving a good decoding algorithm of convolutionally encoded data we should consider the dependency of the sequence of output. Sequence decoding is a method that takes into account the dependency of the sequence of encoded data. Therefore we bring forth a brief overview of different sequence detection and decoding algorithms.

A sequence detection algorithm is an algorithm that tries to select the best sequence if information symbols based on some criterion. Often, the criterion is maximum likelihood

(ML) but there are some additional conditions in sequence detecting algorithm. We can classify different families of sequence decoding into three classes.

1. Breadth-first algorithms
2. Depth-first algorithms
3. List-decoders

We will discuss these algorithms separately.

2.2.1 Breadth-first algorithms

In this algorithm, the number of operations is fixed for all information symbols and each step of trellis advancing; a fixed number of operations is performed. This class of algorithms contains many algorithms. Search Algorithm or SA(B,C) [8] is one of the biggest families in this class. In SA(B,C), the total states S of a trellis is divided into C non-overlapping sets. Out of the S/C paths in each set, B paths kept for trellis advancing and in this decision, previous information of the path and path memory is used. Aulin provided that this family of convolutional decoders can be Maximum Likelihood (ML) optimum for give values of B, C [8]. As mentioned above, this family of algorithms is very big and some special cases of this family are as below:

1- The Viterbi Algorithm: SA(1,S).
2- The Decision Feedback Algorithm: SA(1,1). Where, only the best path is kept for each trellis advancing.
3- The M-Algorithm: SA(M,1). In this algorithm the best $M < S$ paths at each trellis advancing step are kept.
4- The Reduced-State Sequence Detection (RSSD) Algorithm: SA(1,C). Where all S states are grouped in C sets and the best path for each set is kept.

2.2.2 Depth-First algorithms

In depth-first algorithms the number of operations for each symbol decoding is significantly depending on the channel conditions and unlike the breadth first algorithm is not a fix number.

Depth-first algorithms also usually called Sequential Decoding Algorithms. Big advantage of these algorithms is that their complexity unlike the breadth-first algorithms does not increase exponentially with the constraint length of the convolutional code. Due to this fact, application of sequential decoding algorithms for higher constraint length codes is practically possible. Disadvantage of the depth-first algorithms is that they have time varying complexity depending on the channel condition. Another disadvantage is that these algorithms are not ML optimum. However, it can be shown that they can be asymptotically ML optimum (when signal to noise ratio goes to infinity).

Depth-first algorithms are code-tree based decoding algorithms unlike the breadth-first algorithms which are trellis based decoding algorithms. They are called depth-first since in these algorithms; we search through the depth of tree branches to find the best path. In our branch search if we reach to a state that does not fulfill out preset metric criterion (its metric is not good enough), we backtrack and start branch searching to find the best path. Consequently it is possible that we need to compare sequences of different lengths. There are two main sequential decoding algorithms.

- Fano algorithm [9]: This algorithm essentially requires no storage memory. In this algorithm the decoder examines a sequence from one staring node of the code-tree to one ending node and it never jumps between the nodes.
- Stack algorithm or ZJ algorithm: This algorithm was proposed by Zigangirov [10] and Jelinek [11] and it keeps a stack of the node sorted based on their metrics. In each step only the top node is extended and replaced with its successors in the

stack and again stack is sorted based on the metrics. Therefore decoder never revisits any node, but needs more memory and also sorting increases the complexity of the system.

2.2.3 List decoding

Indeed, list decoding is an algorithm that is similar to breadth-first algorithms from one aspect and similar to depth-first algorithms from other aspects.

In some applications like concatenated coding or Automatic Repeat Request (ARQ) system; not only the best path but also some other next paths might be necessary. For example for concatenated codes, it is usually very complex to decode the joint codes at one step. Therefore suboptimal decoding is used at the first steps and by passing the information between the encoders, we approach the optimal sequence in some steps. In this case, access to some possible sequences suggested by the inner decoder, usually enables the outer decoder and the total system to provide improved error performance. In ARQ systems usually there is a Cyclic Redundancy Check (CRC) code that checks the decoded sequence and if any error detected, it asks for retransmission. However, if we have kept the second most likely sequence of the decoder, we can check this sequence with CRC code. If it was not the correct sequence, we can check up to the next B (the number of most possible sequences kept in the decoder) sequences and continue this procedure to find the correct decoded sequence or ask for retransmission.

Viterbi list algorithm is the most frequently used List decoding algorithm. In this algorithm, in east step of decoding, B most likely paths are kept and the out put of the algorithm is a list of B sequences ordered in decreasing probability. Therefore, we ca know this algorithm one of the SA (B, S) algorithms. Different variations of Viterbi list algorithm

exist depending on that if we like to take all the B sequences as output of decoder or just one of them as the decoder output [12]. If we take only one output of the decoder as our output, the algorithm is called M-algorithm that is SA (M, 1).

2.3 Maximum Likelihood Decoding of Convolutional Codes, Viterbi Algorithm

We assume that the N bit codeword of a convolutional code, $\mathbf{v}'$, is transmitted over a discrete memoryless channel (DMC) [15] and call the channel output as vector $\mathbf{r}$. The optimal decoder is a decoder that searches among all possible codewords for a vector $\mathbf{v}$ that maximizes $\Pr(\mathbf{v} \mid \mathbf{r})$. This decoding rule is called the maximum a posterior probability (MAP) rule. From the Bays' rule [16]:

$$\Pr(\mathbf{v} \mid \mathbf{r}) = \frac{\Pr(\mathbf{r} \mid \mathbf{v}) \Pr(\mathbf{v})}{\Pr(\mathbf{r})} \qquad (2\text{-}4)$$

If all codewords are equi-probable, the MAP rule can be simplified as maximizing $\Pr(\mathbf{r} \mid \mathbf{v})$ that is called the maximum likelihood (ML) decoding rule. Since the channel is memoryless:

$$\Pr(\mathbf{r} \mid \mathbf{v}) = \prod_{i=1}^{N} \Pr(r_i \mid v_i) \qquad (2\text{-}5)$$

where v_i and r_i show the i^{th} element of $\boldsymbol{v}$ and $\boldsymbol{r}$ respectively. Now let us define a metric $M(\mathbf{r}, \mathbf{v})$ as follows:

$$M(\mathbf{r},\mathbf{v}) = \log \Pr(\mathbf{r} \mid \mathbf{v}) \qquad (2\text{-}6)$$

Substituting (2-5) in (2-6) leads to:

$$M(\mathbf{r},\mathbf{v}) = \sum_{i=1}^{N} M(r_i, v_i) \qquad (2\text{-}7)$$

It seems that the maximum likelihood decoder has to search among all 2^N available paths on the trellis, for the path that maximizes (2-7). But Viterbi showed that this is not necessary [6]. He showed that the decoding complexity can be considerably reduced by comparing the metrics in each state, rejecting the paths that has the smaller metric, and continuing the search only for surviving paths.

Let us describe the main idea of Viterbi algorithm using an example. Assume that city A is connected to city B with the roads as shown in Figure 2-6. The length of each road (in kilometers) is represented in figure. We wish to find the way from A to B which has the minimum length. Consider three different paths from A to F and their lengths as follows: ACF 80 km, AF 100 km, and ADF 70 km. Since ACF is longer than ADF, any path from A to B that begins with ACF is longer than a path that begins with ADF and continues the same as the former path from F to B.

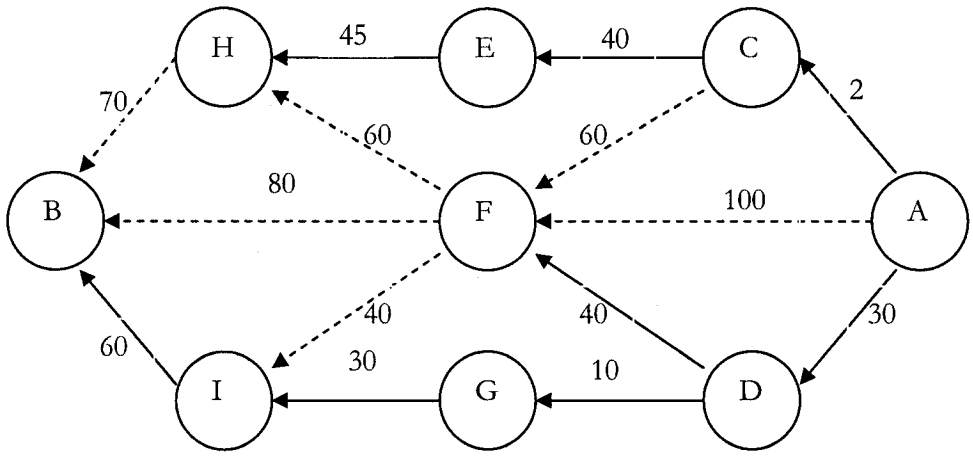

Figure 2-6 An example of Viterbi algorithm

Thus ACF cannot survive and is rejected. AF is rejected for the same reason. The same comparisons in nodes H, I and B leads to rejecting other paths and finding the minimum length path as ADGIB. The rejected paths are shown by dashed lines. Notice that if one wants to compute the distance of all 72 existing paths from A to B and then find the minimum length path, 71 comparisons are required. But using the Viterbi algorithm only 10 comparisons (3 in F, 2 in H, 2 in I, and 3 in B) and smaller number of additions are needed.

The above example shows the key concept behind the Viterbi algorithm for convolutional codes. Consider the trellis diagram of the convolutional code. The Viterbi algorithm for decoding a received vector performs as follows. In each time node t for each state S_i, compare the metrics of all paths that input this state (2 paths for binary code) by adding the metric of previous state to the metric of current branch. Then reject all paths except one surviving path that has the maximum metric and extend the search to the next

time node, only for this surviving path. Define the new metric of S_i as the metric of this surviving path. If the metrics of two paths are equal, one path can be rejected randomly.

2.3.1 Hard Decision Decoding

Assume the sequence $\mathbf{v}$ is transmitted over a binary symmetric channel, i.e. a channel which flips each bit with probability $p < 0.5$ [7]. This is equivalent to transmitting $\mathbf{v}$ over an arbitrary symmetric channel and then quantizing the channel output to one bit. If the Hamming distance between $\mathbf{v}$ and the received vector $\mathbf{r}$ is shown by d

$$\Pr(\mathbf{r} \mid \mathbf{v}) = p^d (1-p)^{N-d} \qquad (2\text{-}8)$$

Then using (2-6):

$$M(r,v) = N\log(1-p) - d\log\frac{1-p}{p} \qquad (2\text{-}9)$$

Therefore in hard decision decoding the decoded codeword is the codeword that has the minimum Hamming distance from the received vector. This vector can be found by running Viterbi algorithm on the corresponding trellis.

2.3.2 Soft Decision Decoding

Now assume the codeword $\mathbf{v}$ is transmitted over a memoryless additive white Gaussian noise (AWGN) channel, i.e.:

$$\Pr(r_i \mid v_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-(r_i - v_i)^2 / 2\sigma^2) \qquad (2\text{-}10)$$

where σ^2 is the variance of the noise. Using (2-6), (2-7), and (2-10):

$$M(\mathbf{r}, \mathbf{v}) = -\log\sqrt{2\pi\sigma^2} - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(r_i - v_i)^2 \qquad (2\text{-}11)$$

Notice that $\sum_{i=1}^{N}(r_i - v_i)^2$ is the Euclidean distance between $\mathbf{r}$ and $\mathbf{v}$. Thus in this case the decoded codeword is the codeword that has the minimum Euclidean distance from the received vector. This vector can be found by running Viterbi algorithm on the corresponding trellis.

2.4 Stack Algorithm

Viterbi algorithm suggests an exponential increase of computational complexity by increasing the code constraint length. This makes it unsuitable for the codes with a relatively large constraint length, e.g. $K = 15$. Also Viterbi decoding is independent of the level of noise, i.e. regardless of whether the channel noise is light or the channel is very noisy, a fixed amount of computations are needed to find the maximum likelihood path.

This drawback makes Viterbi decoding unsuitable for the channels with a light noise. Considering the above mentioned drawbacks, different tree based search algorithm are introduced to reduce the number of computations for large constraint length codes and/or light noise channels. These algorithms are generally called sequential decoding algorithms. Stack algorithm [10], [11], Fano algorithm [9], ... are special cases of sequential decoding algorithms.

While in Viterbi decoding, metrics of the paths of the same length are compared with each other, in sequential decoding the length of compared paths may be different. Therefore the metric must be redefined to take the effect of path length in account. The most common metric for sequential decoding is the Fano metric [7] and is defined as follows:

$$M(r_i \mid v_i) = \log \frac{\Pr(r_i \mid v_i)}{P(r_i)} - R \qquad (2\text{-}12)$$

where R is the code rate. Using (2-7) and (2-12):

$$M(\mathbf{r} \mid \mathbf{v}) = \sum_i \log \Pr(r_i \mid v_i) + \sum_i (\log \frac{1}{\Pr(r_i)} - R) \qquad (2\text{-}13)$$

The first term in (2-13) is the metric for the Viterbi algorithm and the second term changes by the sequence length. For a BSC with transition probability of p:

$$M(r_i \mid v_i) = \begin{cases} \log 2p - R & r_i \neq v_i \\ \log 2(1-p) - R & r_i = v_i \end{cases} \qquad (2\text{-}14)$$

Assume that one wishes to decode a received message of length N. Consider the tree diagram of code up to level N. The stack algorithm work as follows. A stack of different paths with maximum metric is kept in storage. The stack is ordered such that the path with maximum metric is on top. For this path, the search is extended to 2^k paths of the next levels. These paths are called the successors of the top path. The top path is deleted and its successors are inserted in the stack. The stack is rearranged and the procedure is repeated until the top path of the stack reaches the last level of the tree. Different steps of the stack algorithm can be shown as follows [7]:

Stack Algorithm:

- Initialization: Load the stack with the origin node of the tree with metric of zero.
- Repeat until the top path in the stack reaches a terminal node of the tree
  - Compute the metric of the successors of the top path
  - Delete the top path and insert its successors in the stack
  - Rearrange the stack in order of decreasing metric values.
- Output the top path and stop.

2.5 M-Algorithm

M-algorithm [17] performs the same as Viterbi algorithm with the difference that in each step only M states with the largest metrics are kept and the search is extended to the next level, only for these M states. Since each states yield 2^k (2 for the binary codes) states at the next level, at most $2^k \times M$ states are available for the next level. Among these levels,

again M levels with the maximum metrics are chosen and algorithm continues. The complexity of the M-algorithm is a function of M. Since algorithm only keeps M states, it is possible that all vestiges of the correct path are lost. In this case the algorithm cannot track the correct path any more. This event is called correct path loss and it causes catastrophic error. To obtain a good error performance of the M-algorithm the correct path recovery schemes should be provided that is the objective of the chapters 4 and 5 of the thesis. Since we need an error detection code in our correct path recovery algorithm of the chapter 5, we bring forth the basic information about the Cyclic Redundancy Check codes in the next chapter.

2.6 Summary

In this chapter we introduced the Convolutional codes and their importance in the communication systems. Also we brought forth some different representations of the convolutional encoders like the trellis and the state diagram. Then, we explained the different sequence decoding methods. Later on, the maximum likelihood decoding of the convolutional codes and the concept of the Viterbi algorithm as a Maximum Likelihood algorithm was talked about. Also the concept of hard decision and soft decision in the decoding was mentioned. As an alternative way of decoding, stack algorithm was briefly explained next and as the final step M-algorithm was introduced as a reduced state Viterbi algorithm and the correct path loss event in the M-algorithm as a catastrophic error also was introduced.

Chapter 3

CRC ERROR DETECTION CODES

One of the most popular methods of error detection for digital signals is the Cyclic Redundancy Check (CRC) coding [18], [19]. We explain the basic idea behind the CRC codes and explain how easily they can be implemented in this chapter.

3.1 CRC Codes: the Concept and the Implementation

The basic idea behind CRCs is to treat the message string as a single n bit binary word $\mathbf{v}$, and divide it by a k bit keyword $\mathbf{s}$ that is known to both the transmitter and the receiver.

The remainder **r** constitutes the check word for the given message. The transmitter sends both the message string and the check word, and the receiver can then check the data by repeating the calculation, dividing **v** by **s** and verifying that the remainder is **r**. The novel aspect of the CRC process is that it uses a simplified form of arithmetic, which we will explain by an example, in order to perform the division. This method of checking for errors is obviously not foolproof, because there are many different message strings that give a remainder of **r** when divided by **s**. In fact, about one out of every 2^k randomly selected strings will give any specific remainder. Thus, if our message string is garbled in transmission, there is a probability (about 2^{-k}, assuming the corrupted message is random) that the garbled version would agree with the check word. In such case the error would be undetected. By making k large enough, this probability can become small enough. The rest of this discussion will consist simply of refining this basic idea to optimize its effectiveness.

When discussing CRCs it is customary to present the keyword **s** in the form of a generator polynomial with coefficients equal to the corresponding bit of the binary description of **s**. For example, suppose we want our CRC to use the keyword **s**=37. This number in binary basis is 100101, and can be expressed as a polynomial $X^5 + X^2 + 1$. In order to implement a CRC based on this polynomial, the transmitter and the receiver must have agreed in advance that this is the keyword they intend to use. So, for the sake of discussion, let us say we have agreed to use the generator polynomial 100101. Note that the remainder of any word divided by a 6-bit word will contain no more than 5 bits, thus our CRC words based on the polynomial 100101 will always fit into 5 bits. A CRC system based on this polynomial would be called a 5-bit CRC. In general, a polynomial with k bits leads to a *k-1* bit CRC. Now suppose one wants to send a message consisting of the string of bits **v** = 00101100010101110100011. Using the agreed keyword 100101, he simply

divides **v** by **s** to form the remainder r, which will constitute the CRC check word. A worksheet for the entire computation is shown below:

```
          ____________________________
100101 |00101100010101110100011
         100101
         ------
         00100101
           100101
           ------
           0000000101110
                  100101
                  ------
                  00101110
                    100101
                    ------
                    00101100
                      100101
                      ------
                      00100111
                        100101
                        ------
                        000010    remainder = CRC
```

Our CRC word is simply the remainder, i.e., the result of the last 6-bit exclusive OR operation. Of course, the leading bit of this result is always 0, so we really only need the last five bits. This is why a 6-bit keyword leads to a 5-bit CRC. In this case, the CRC word for this message string is 00010, so when the message word **v** is transmitted, the

corresponding CRC word of 00010 is also sent along with it. When the receiver receives the corrupted message it repeats the above calculation on **v** with the agreed generator polynomial **s** and verifies that the resulting remainder agrees with the CRC word included in transmission.

What has just been done is a perfectly fine CRC calculation, and many actual implementations work exactly that way, but there is one potential drawback in this method. As one can see, the computation described above totally ignores any number of zeros ahead of the first non-zero (one) bit in the message. It so happens that many data strings in real applications are likely to begin with a long series of zeros, so it is a little bothersome that the algorithm is not working very hard in such cases. To avoid this problem, we can agree in advance that before computing our k-bit CRC we will always begin by exclusive OR ing the leading $m < k$ bits of the message string with a string of m ones. With this convention (which of course must be agreed by the transmitter and the receiver in advance) our previous example would be evaluated as follows

```
00101100010101110100011    <--  Original message string
11111                      <-- "Fix" the leading bits
-----------------------
11010100010101110100011    <-- "Fixed" message string
100101
------
0100000
 100101
 ------
 000101001
    100101
    ------
```

```
00110001
  100101
  ------
  0101000
   100101
   ------
   00110111
     100101
     ------
     0100101
      100101
      ------
      0000000100011
             100101
             ------
             000110    remainder = CRC
```

So with the "leading zero fix" convention, the 5-bit CRC word for this message string based on the generator polynomial 100101 is 00110. People sometimes use various table-lookup routines to speed up the divisions, but that does not alter the basic computation or change the result. In addition, people sometimes agree to various non-standard conventions, such as interpreting the bits in reverse order, but the essential computation is still the same. (Of course, it is crucial for the transmitter and receiver to agree in advance on any unusual conventions they intend to observe.)

Now that we have seen how to compute CRCs for a given key polynomial, it is natural to wonder whether some key polynomials work better (i.e., give more robust "checks") than others. From one point of view the answer is obviously yes, because the larger our keyword, the less likely it is that corrupted data will be undetected. By

appending an k-bit CRC to our message string we are increasing the total number of possible strings by a factor of 2^k, but we are not increasing the degrees of freedom, since each message string has a unique CRC word. Therefore, we have established a situation in which only *1* out of 2^k total strings (message plus CRC) is valid. Notice that if we append our CRC word to the message word, the result is a multiple of our generator polynomial [19]. Therefore, of all possible combined strings, only multiples of the generator polynomial are valid. Thus, if we assume that any corruption of our data affects our string in a completely random way, i.e., such that the corrupted string is totally uncorrelated with the original string, then the probability of a corrupted string being undetected is 2^{-k}. For example a 16-bit CRC has a probability $2^{-16} = 1.5\times10^{-5}$ of failing to detect an error in the data, and a 32-bit CRC has a probability of $2^{-32} = 2.3\times10^{-10}$ of failing to detect an error.

Since most digital systems are designed around blocks of 8-bit words (called "bytes"), it is most preferred to find keywords whose lengths are a multiple of 8 bits. The two most common lengths in practice are 16-bit and 32-bit CRCs (so the corresponding generator polynomials have 17 and 33 bits respectively). A few specific polynomials have come into widespread use. For 16-bit CRCs one of the most popular keywords is 10001000000100001, and for 32-bit CRCs one of the most popular keywords is 100000100110000010001110110110111. The 16-bit polynomial is known as the "X25 standard", and the 32-bit polynomial is the "Ethernet standard", and both are widely used in all sorts of applications. (Another common 16-bit key polynomial familiar to many modem operators is 11000000000000101, which is the basis of the "CRC-16" protocol.) These polynomials are certainly not unique in being suitable for CRC calculations, but it is probably a good idea to use one of the established standards, to take advantage of all the experience accumulated over many years of use.

Although we agree to use the standard keywords, we may still be curious to know how these particular polynomials were chosen. Notice that it is possible that one could use just about any polynomial of a certain degree and achieves most of the error detection benefits of the standard polynomials. For example, any k-bit CRC will certainly catch any single "burst" of $m<k$ consecutive "flipped bits" basically because a smaller polynomial cannot be a multiple of a larger polynomial. Also, we can ensure the detection of any odd number of erroneous bits simply by using a generator polynomial that is a multiple of the "parity polynomial", which is $X+1$. A polynomial of our simplified kind is a multiple of $X+1$ if and only if it has an even number of terms. It is interesting to note that the standard 16-bit polynomials both include this parity check, but the standard 32-bit CRC does not. It might seem that this represents a shortcoming of the 32-bit standard, but it really does not, because the inclusion of a parity check comes at the cost of some other desirable characteristics. In particular, much emphasis has been placed on the detection of two separated single-bit errors, and the standard CRC polynomials were basically chosen to be as robust as possible in detecting such double-errors. The basic error word **e** representing two erroneous bits separated by j bits is of the form X^j+1 or equivalently, X^j-1. Also, an error **e** superimposed on the message **v** will be undetectable if and only if **e** is a multiple of the key polynomial **s**. Therefore, if we choose a key that is not a divisor of any polynomial of the form X^j+1 for $j=1,2,...,m$, then we are assured of detecting any occurrence of precisely two erroneous bits that occur within m places of each other. For this purpose we can use a primitive polynomial [7]. For example, suppose that we want to ensure detection of two bits within 31 places of each other. Let us factor the error polynomial $X^{31}+1$ into its irreducible components [7]

$$X^{31}+1 = (X+1)$$
$$*(X^5+X^3+X^2+X+1)$$

$*(X^5 + X^4 + X^2 + X + 1)$

$*(X^5 + X^4 + X^3 + X + 1)$

$*(X^5 + X^2 + 1)$

$*(X^5 + X^4 + X^3 + X^2 + 1)$

$*(X^5 + X^3 + 1)$

Aside from the parity factor $X+1$, these are all primitive polynomials, representing primitive roots of $X^{31} + 1$, so they cannot be divisors of any polynomial of the form $X^j + 1$ for any j less than 31. Notice that $X^5 + X^2 + 1$ is the generator polynomial 100101 for the 5-bit CRC in our first example.

Another way of looking at CRC codes is via recurrence formulas. For example, the polynomial $X^5 + X^2 + 1$ corresponds to the recurrence relation s[n] = (s[n-2] + s[n-5]) modulo 2. Beginning with the initial values 00001 this recurrence yields

```
                    |--> cycle repeats
0000100101100111110001101110101 00001
```

Notice that the sequence repeats with a period of 31, which is another consequence of the fact that $X^5 + X^2 + 1$ is primitive. It can also be observed that the sets of five consecutive bits run through all the numbers from 1 to 31 before repeating. In contrast, the polynomial $X^5 + X^2 + 1$ corresponds to the recurrence s[n] = (s[n-4]+ s[n-5]) modulo 2, and gives the sequence

```
                |--> cycle repeats
000010001100101011111 00001
```

Notice that this recurrence has a period of 21, which implies that the polynomial $X^5 + X^2 + 1$ divides $X^{21} + 1$. On the other hand, $X^5 + X + 1$ can be factored as

$(X^2+X+1)(X^3+X^2+1)$, and both of these factors divide $X^{21}+1$. Therefore, the polynomial X^5+X+1 gives a less robust CRC than X^5+X^2+1 from the standpoint of maximizing the distance by which two erroneous bits must be separated in order to go undetected. On the other hand, there are error patterns that would be detected by X^5+X+1 but would not be detected by X^5+X^2+1.

As noted previously, any *k*-bit CRC increases the space of all strings by a factor of 2^k, so a completely arbitrary error pattern really is not less likely to be detected by a poor polynomial than by a good one. The distinction between good and bad generators is based on the premise that the most likely error patterns in real life are not entirely random, but are most likely to consist of a very small number of bits (e.g., one or two) very close together. To protect against this kind of corruption, we need a generator that maximizes the number of bits that must be "flipped" to reach from one formally valid string to another. We can certainly cover all 1-bit errors, and with a suitable choice of generators we can effectively cover virtually all 2-bit errors. Whether this particular failure mode deserves the attention it has received is debatable. If our typical data corruption event flips dozens of bits, then the fact that we can cover all 2-bit errors seems less important.

3.2 Summary

In this chapter we briefly introduced the Cyclic Redundancy Check (CRC) Codes. We explained the characteristics of the CRC codes and introduced some common CRC codes in the currently used communication systems. It is also was shown that how easy the CRC codes can be implemented just using some XOR operations.

Chapter 4

ANCESTOR BASED SURVIVOR DECISION IN THE M-ALGORITHM CONVOLUTIONAL DECODER

As mentioned in last chapters, Viterbi algorithm is one of the most widely used algorithms for decoding of the convolutional codes. In this algorithm all the possible paths in the trellis are examined and the most likely path is determined as the sent code sequence. Since the number of possible paths increases exponentially with the constraint length K of the convolutional code, this algorithm is not a practical algorithm for long constraint length codes.

4.1 M-algorithm Decoding Procedure

The procedure of decoding in M-algorithm decoder has been depicted in Figure 4-2. Using the simple example of Figure 4-2, we will show how the M-algorithm is performed.

We consider the convolutional code with constraint length K=2 and generator sequences as $g1$=7 (111) and $g2$=5 (101). Figure 4.2 shows the encoder of this code.

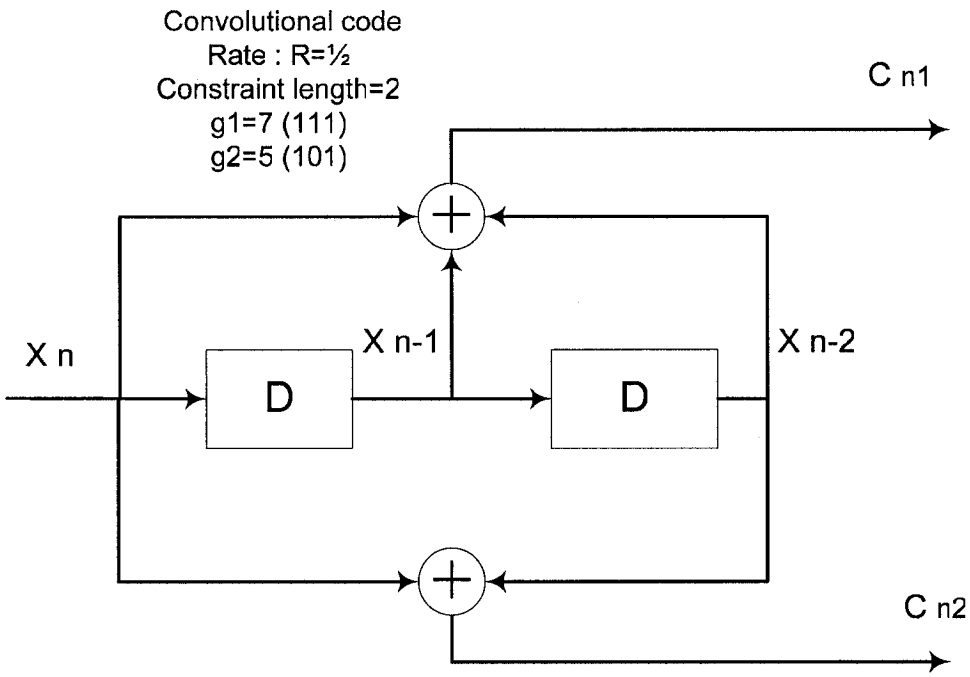

Figure 4-1 Encoder of Convolutional Code Rate 1/2 and
Constranint length = 2.

As Figure 4-2 shows, the procedure of decoding in M-algorithm can be divided in the following steps.

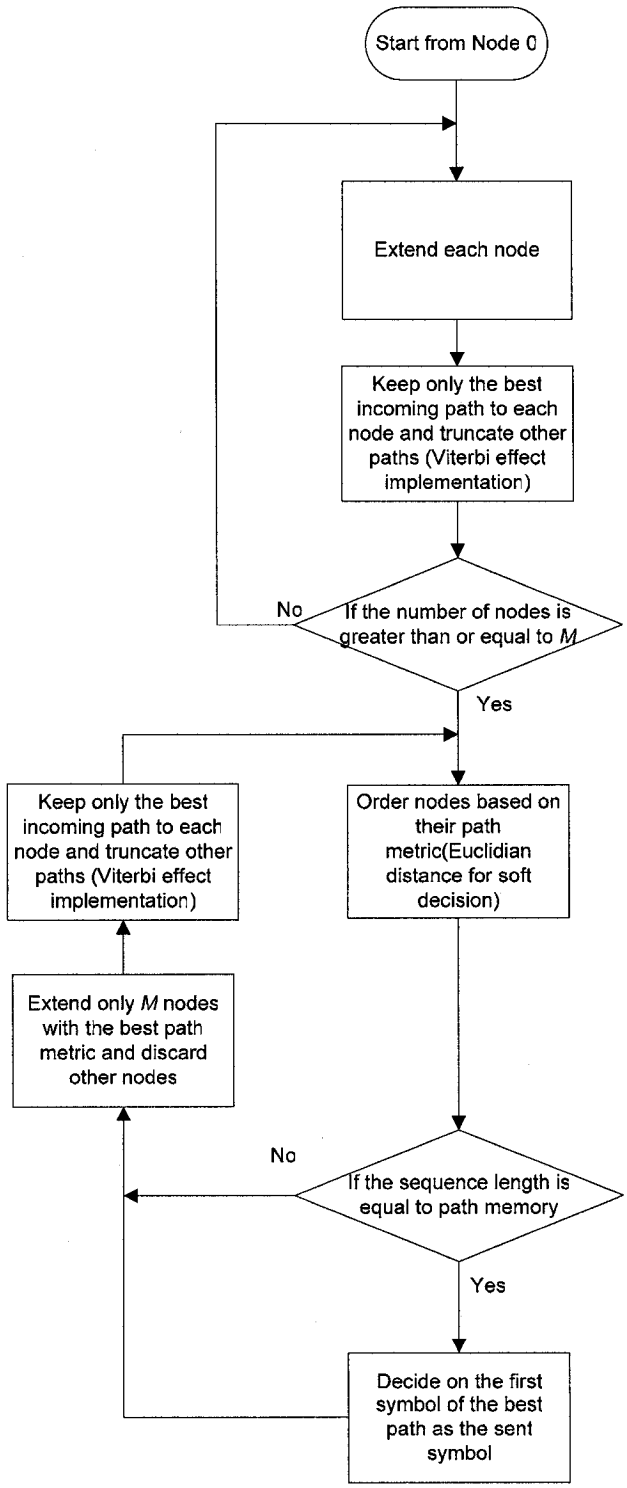

Figure 4-2 M-algorithm decoding procedure.

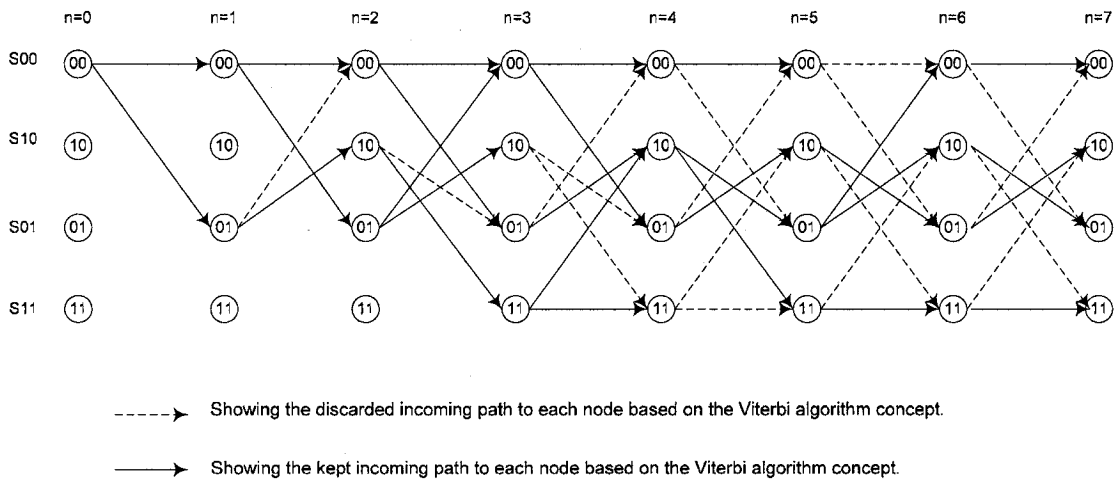

Figure 4-3 In Viterbi algorithm only one best incoming path to each node is kept.

1- Start from the first node: The algorithm starts with one single node and the number of nodes is increased while going to the next depth of decoding (each step of trellis advancing is corresponding to one more depth of the algorithm). Step $n = 0$ in Figure 4-3 shows the starting point from one single node.

2- Extend each node: The existing nodes are extended in each step of trellis advancing. If the code is binary, each node is extended to two nodes. Figure 4-3 shows an extension of each node.

3- Keep only one incoming path to each node: Based on the concept behind the Viterbi algorithm, we know if we keep only one incoming path to each node of the trellis; still we will have a Maximum

Likelihood optimum algorithm. In Figure 4-3 the dotted lines are the branches of those paths which should be discarded in this stage.

4- Repeat step 2, 3 until we have at least M nodes: Before we have M nodes or paths, we can extend all nodes without any limitation. In Figure 4-4 we see that until depth n=1 we don't need to discard any node or path until this step, the existing paths are not higher than M=2.

5- Order all nodes and paths (each node has just one incoming path): Nodes are ordered based on the whole sequence, Euclidian (Soft decision) or Hamming (Hard decision) distance of their paths from the received sequence. This step provides us with the necessary measurement for node and path survivor decision in the next steps.

6- If the decoder path memory is full, decide on the first symbol of the best path as the received symbol: Like Viterbi algorithm, here we have a maximum path memory L which is usually at least 5 times as long as the constraint length of the code. We store the received sequence as well as the survived paths of length L. If the path memory is full, and we want to add one more symbol to the sequence, we have to remove one of the oldest symbols from the memory. This oldest symbol is our decided symbol for this step of trellis decoding. For example, if our path memory length is L=10, at time n=10 we should decide the first received symbol and at the time n=11 we will decide for the second symbol and so on for the next symbols. In this example we have M=2

and L=10 at time n=10 there are two survived paths. We take the fist symbol of the path with the better distance metric as the decoded symbol.

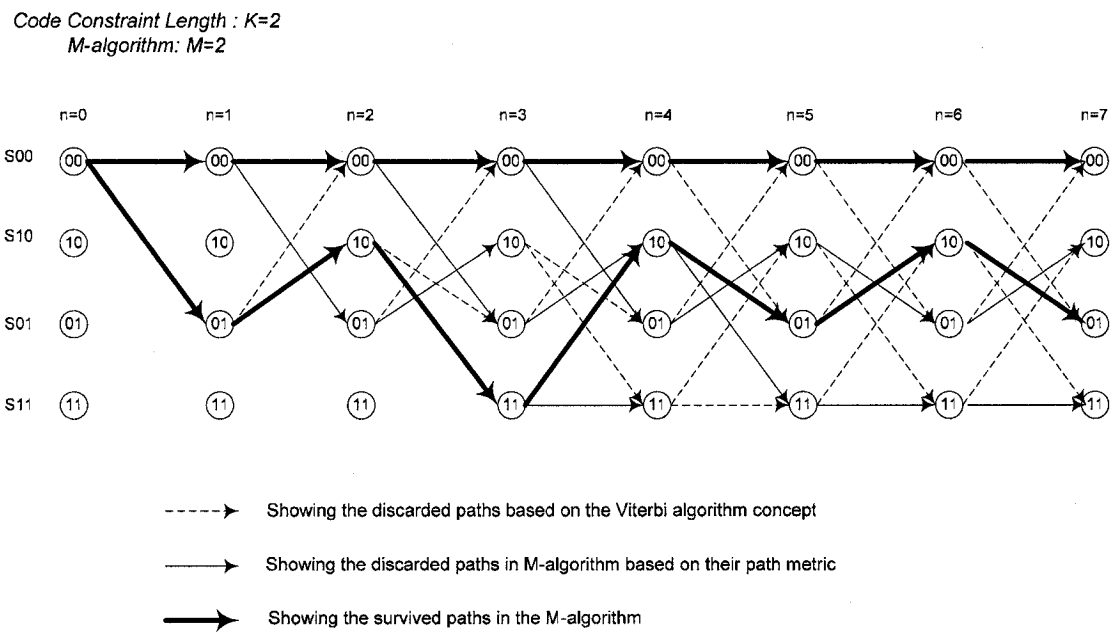

Figure 4-4 M-algorithm convolutional decoder does not keep all the paths in each step of trellis advancing

7- Extend only M nodes with the best path metric and discard other nodes: Since in the M-algorithm only M nodes can survive in each step, we select the most likely M paths from the existing paths and extend their corresponding nodes. Given that our nodes are already ordered, we can select the first (best) M nodes and discard other nodes. In Figure 4-5 it

is seen that only 2 nodes survive in each step and other nodes are discarded in this example. .

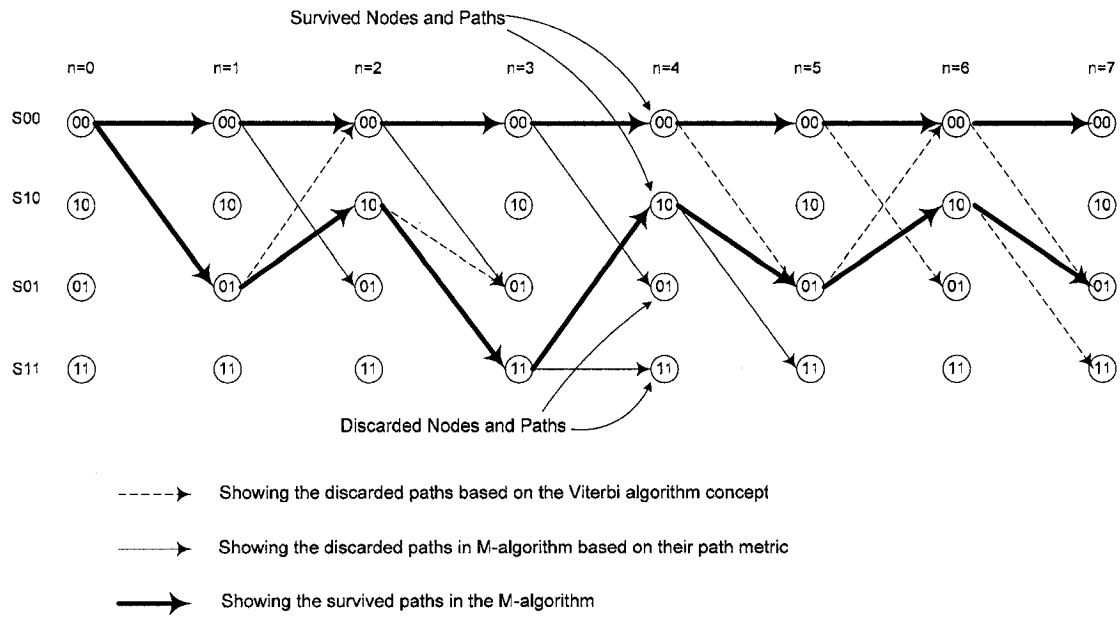

Figure 4-5 In the M-algorithm the nodes with the worst metrics are discarded.

8- Keep only the best incoming path and (Viterbi effect implementation): This step is like step 3 and is performed in every trellis advancing.

9- Go back to step 5, and continue the procedure from step 5 for every step of decoding.

4.2 The Error Performance of the M-algorithm

Unlike the Viterbi algorithm, the application of the M-algorithm convolutional decoders for higher constraint length codes is practical since the decoding complexity does not depend on the code constraint length. In the M-algorithm, M numbers of the paths in each step of the trellis advancing survive. M or the number of survivors of the M-algorithm does on one hand have a direct effect on the probability of error in the system and on the other hand it determines the complexity of the decoder. If M increases the algorithm will be more like Viterbi algorithm and it can have a better performance with a maximum likelihood algorithms. Meanwhile, increasing M will increase the number of checked nodes for each symbol decoding and then the complexity of the system will be increased. Consequently a trade off between the complexity of the system, and the error performance of the system will decide the value of M.

4.2.1 Correct Path Loss in M-algorithm

Since in the M-algorithm we don't keep all possible paths in each step of trellis advancing; it is possible that the correct path does not have a good metric in one step and it is discarded in that step. In this case correct path loss happens.

If the correct path loss happens; as long as the correct path has not been recovered, the decoder is in error decoding state and its decoded stream is in erroneous state. This event dramatically increases the bit error probability of the system, if the correct path is not recovered shortly.

The probability of the correct path recovery for convolutional codes decreases while d_{free} or constraint length of the code increases. It is due to this fact that the ratio of the number of survivors to the total number of possible paths is less in this case and therefore, the probability that the correct path is again located in the group of survived paths is lower. Then again, we know that the codes with higher constraint length have better error performance. Considering the previously mentioned facts, a good coding system uses long constraint length codes while simultaneously being equipped with the techniques of correct path recovery and prevention.

4.2.2 Correct Path Recovery Methods

As previously mentioned, correct path loss has a dramatic effect on the bit error probability of the M-algorithm convolutional decoder. It is preferable to ensure correct path loss is prevented. If correct path loss prevention was not possible, that it is recovered quickly..

The error event and the error propagation due to correct path loss can be controlled by organizing the data in frames and blocks of data with known starting or tail states and bit structure by using special recovery techniques [20]. However, in these techniques having the starting state know itself is an extra complication and although these algorithms might achieve a better error event probability, their bit error probability without a very good recovery scheme is still poor.

Another recovery method is the so called Adaptive Viterbi Algorithm or AVA which was proposed by F. Chan and David Haccoun [21]. This algorithm can recover the correct path after a few trellis levels. However, in this algorithm for obtaining an error performance like the Viterbi algorithm the value of M is increased to half of the total number of paths in

the Viterbi algorithm which is a high value and makes the decoding procedure very complex.

The simple method of data framing is a very common method for correct path recovery. When the data framing technique is used, data bits are sent fame by frame. In this case if the correct path loss happens, the stream of errors will not go beyond the end of each frame and the error event finishes at the end of the frame. Although data framing is very useful in many applications, in many other applications it is not sufficient. It is clear that even if data framing is used, still each error event results in many erroneous bits mostly to the end of the frame. This event degrades the overall bit error probability of the communication system. Therefore, the probability of an error event should be kept as low as possible.

In the M-algorithm, the number of survivors is increased to keep the error event probability as low as possible. However, increasing the number of survivors leads to higher complexity of the system and should be avoided if it is possible. Therefore, algorithms which avoid correct path loss or recover the lost path without increasing the complexity of the system are desirable.

4.3 Sources of the Correct Path Loss

For avoiding correct path loss, we should know what causes it. Here we explain two different causes of the correct path loss. Then we will be able to design correct path loss avoidance schemes for different sources of correct path loss.

In M-algorithm, that is a trellis based convolutional decoder, in every step of trellis advancing only M paths survive and all other paths are discarded. In the conventional M-algorithm, Hamming (hard decision) or Euclidian distance (soft decision) of each path from the received sequence of data is used to determine which path should survive and which one should be discarded.

If we assume that the correct path is lost at the step n of the trellis, it means that at step n-1 and other steps before it, the correct path has been among the survivor paths and if we had kept all the successors of the correct path in the trellis, we would have not lost it. Here we consider the possible reasons of discarding the correct path in going from step n-1 to n.

In the M-algorithm, the M paths with better distance metric are kept and the others are discarded. Therefore correct path loss at step n means that the correct path metric from step n-1 to n has been affected by such deterioration that it has left the group of survivor paths. There can be two types of system noise that cause this event.

4.3.1 Abrupt Noise

The first type of noise that can cause correct path loss is abrupt noise. This kind of noise is induced to the system in short period. However, in that short period it has high values and consequently causes severe impact on the system. If such a noise happens, even if the correct path has an excellent metric in step n-1, it might become severely affected by this noise and in step n, it has such a deteriorated metric that even it is not among the M best surviving paths.

Different short term interferences in the communications like the noise in fading channels or shot noise can be of examples of this kind of noise.

4.3.2 Accumulated Noise

The second type of noise that might cause correct path loss is accumulated noise. This noise can deteriorate the correct path metric gradually. The accumulated effect of noise can be so high that in the n^{th} step, correct path metric can be worse than at least M other survivor paths and it is discarded. In this case, the correct path has been vulnerable to be lost from steps before step n and it can be lost in step n even with a rather low value noise induced in the transition from step n-1 to n.

4.4 Ancestor Based Survivor Decision in M-algorithm Convolutional Decoders

Here we propose a scheme for correct path recovery in the case of abrupt noise. We call this scheme "Ancestor Based Survivor Decision in M-algorithm Convolutional Decoders".

Before explaining the proposed scheme, we will use an example to briefly show how abruptly induces noise causes correct path loss.

Figure 4-6 shows an example of an ordered graph based on the path metrics. The paths and the nodes with better metrics have been located in the upper part of the graph. The M-algorithm decoder used here has a number of survivor of $M = 4$. In this example, an abrupt noise affects the sequence of data between time n-1 and n. Affected by that, the

order of the correct path metric at time n turns to the 6th metric from the fist one at time n-1. In this example the survivor decision is made only on the basis of the current-metric of each path. Since M=4, only 4 paths with the best metric survive in each step and consequently the correct path that has the 6th place at time n will be discarded leading to the correct path loss event.

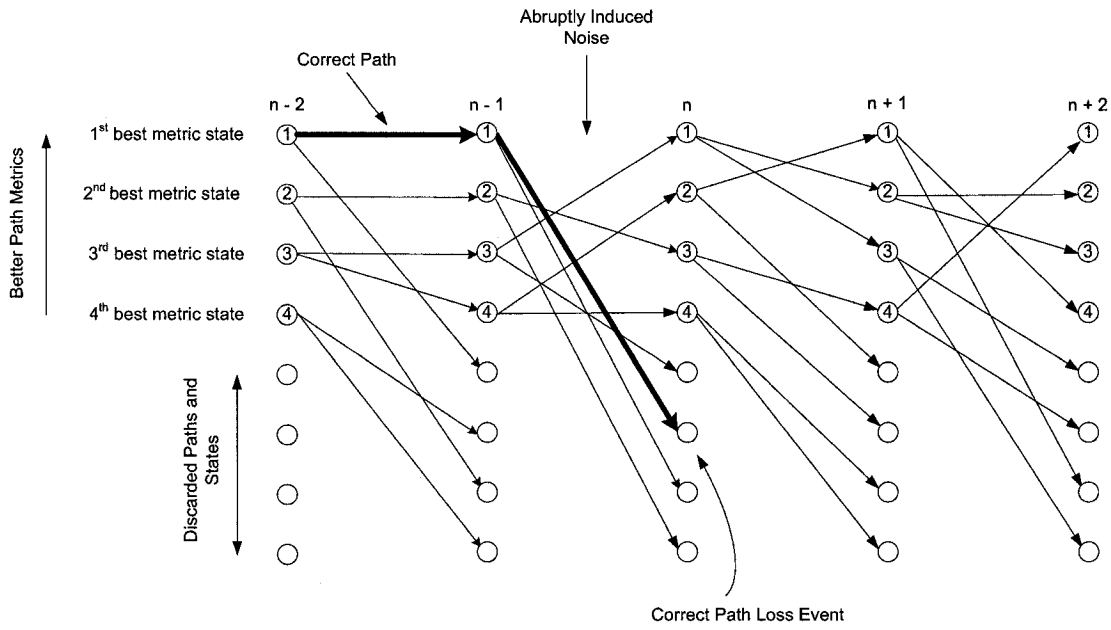

Figure 4-6 Correct path loss event in the M-algorithm in the presence of abrupt noise.

Our proposed scheme of survivor decision tries to reduce the correct path loss caused by this abrupt noise.

Let's consider the example of Figure 4-6 and assume that the correct path has had the best path metric at step n-1. Also, we know that the correct path in step n should also be a successor of the correct path in step n-1. Now if in step n, we keep all successors of the best metric path of the step n-1, surely we have kept the correct path regardless of the abrupt

noise between steps *n-1* and *n*. In this case, abrupt noise has not been able to affect our system. We take advantage of this fact, and proposed our survivor decision scheme based on it.

We explain the proposed scheme by an example. Figure 4-7 shows an example like example of Figure 4-6. In this example two groups of paths survive in each step. Not only a group of paths with the best metric survive, but also another group of paths which does not have very good metrics but have fathers with very good path metrics survive. This group of survivors has been selected as survivor to avoid the correct path loss caused by abrupt noise. Hence, we can say that in this algorithm the total number of survivors is divided into two groups. The first group includes the nodes with good ancestor metrics and the remaining number of the survivors out of the total *M* number is selected from the paths with the currently best metrics.

In this example, as it is shown in Figure 4-7, correct path in step *n* is not discarded and it will recover its good metric order and correct path loss that would happen in the conventional M-algorithm that is not happening now.

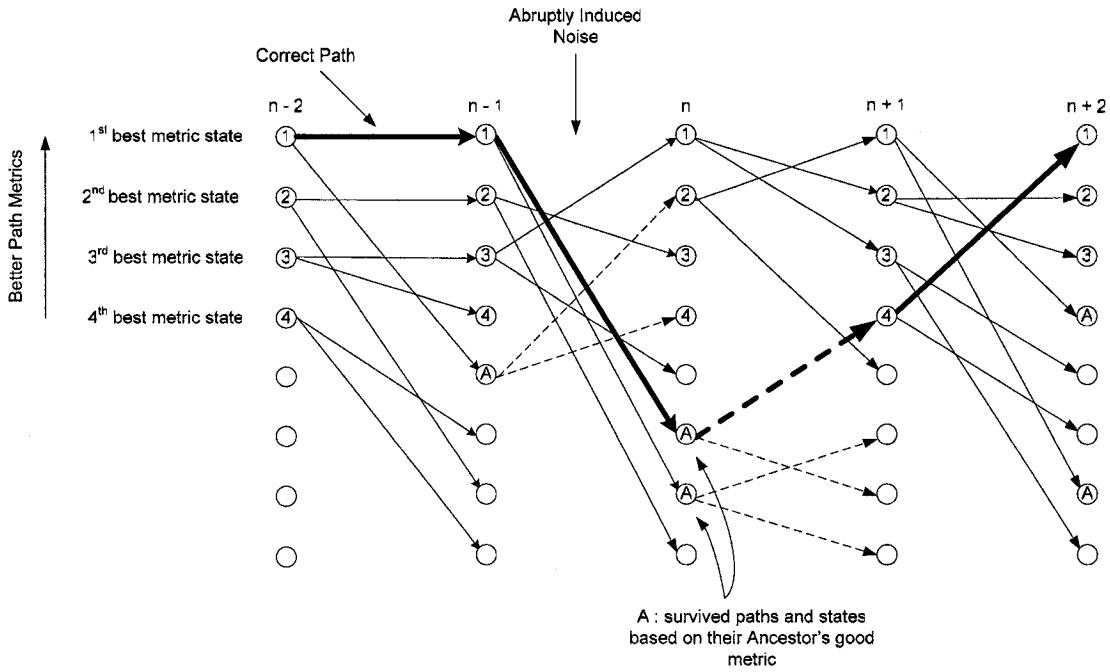

Figure 4-7 Correct path loss event in the Ancestor Based Survivor Decision M-algorithm in the presence of abrupt noise

In this example, we made two assumptions that should be taken into account for generalization of the algorithm.

The first assumption is that the correct path is the first rank metric or the best metric in step *n-1*. This does not always happen in real systems. Therefore, if we keep not only all the successors of the best path in step *n-1*, but also keep all successors of the next order paths in step *n-1*, the correct path will survive in step *n* if it is a successor of the next order nodes in step *n-1*. However, we should keep in mind that the number of survivors is limited and we cannot allocate all spaces to the ancestor-best paths (the survivor paths which their ancestors have been the best in their own time).

Another assumption in our example is that that the effect of abrupt noise will be diminished after one step and the correct path metric can recover its metric in one step and join back the group of high rank paths. Again, this is not always the case. In this case, if we keep all successors of the further ancestors like grand father or higher level ancestors, we can give enough time to the abrupt noise affected correct path to recover its position among the high rank paths. No need to re-mention that still we should consider the limitation of the number of survivors in this case.

To briefly summarize this algorithm we can say that in this algorithm, two different categories of paths survive. One group is the group of paths with the best ancestor metrics and the other group is the group of the best metric paths. The higher number of ancestor-best metric group members helps to avoid correct path loss due to the abrupt noise and higher number of the best-metric paths helps to keep far the system from the path loss caused by the accumulated noise. However, jointly increasing these numbers increases the complexity of the system and a trade-off between these two numbers and also the complexity of the system is needed.

In the next section we will show an implementing procedure of this algorithm and explain the complexity of the proposed algorithm.

4.4.1 Implementing Procedure of the Ancestor Based Survivor Decision M-algorithm

Before starting the algorithm we should indicate the following numbers:

1- *M*: The total number of survivor paths in M-algorithm.

2- *P:* The number of the ancestor based survivor paths.

3- *L*: The generation level of the ancestor from the current step in which all of its successors should be kept at the current step.

For every survivor decision, we keep *M* paths. Out of this *M* numbers *P* paths are selected based on their L^{th} generation ancestor metric and *M-P* other paths selected from the paths with currently best metrics. Therefore, in each step of the trellis advancing we need to know two parameters. The first one is the path metric of each node and second is the order of its L^{th} generation ancestor.

If we store the metric rank of the nodes in each step and keep it for *L* other steps, we can use it when it is necessary, in the time that survivor decision for its L^{th} successor is performed. Therefore we can say that this algorithm can be implemented like the conventional M-algorithm but in this algorithm we need a storage memory for each node to store the rank of the 1^{st} to L^{th} generation ancestors of the node. Calculation of the rank is performed in the conventional M-algorithm also, and only it should be stored for the proposed algorithm. Hence, there is no need to process more than the conventional M-algorithm since in the conventional M-algorithm for every survivor decision, nodes are ordered and we can store the rank of every node and use it while deciding the survivors of its L^{th} generation successors. Therefore, we can say that the proposed algorithm needs only more memory to be implemented.

The amount of memory is dependent on the effective level *L* of the ancestors on the survivor decision. In the next section we will illustrate the simulation result of some special cases and explain how the proposed algorithm can be implemented.

4.4.2 Simulation and Numerical Results

We consider M-algorithm convolutional decoder in the presence of Additive White Gaussian Noise or AWGN. As previously explained the proposed algorithm is expected to offer better performance in the presence of abrupt noise and the performance of the algorithm in the presence of this kind of noise can be studied in future works on the algorithm.

Monte Carlo simulation is performed to check the error performance of the system. Different combinations of the following parameters are possible.

- *M*: Survivor number of M-algorithm.
- *K*: Constraint length of the convolutional code.
- *R*: Code rate of the convolutional code.
- *F*: Number of bits in each frame.
- *P*: Number of survivor nodes based on their ancestors metric.
- *L*: Shows that the L^{th} ancestor metric should be taken into account for ancestor-best metric survivor decision.

In our simulation we use the following parameters.

- $M = 64$: Meaning we keep 64 paths in each step of trellis advancing.
- *K1=9, K2=15:* Two convolutional codes of the rate of $R=1/2$ and the constraint lengths of $K1=9$, $K2=15$ from codes with maximum free distance introduced in [22] are employed.

  $K1=9$, $d_{free}=12$, Encoder Polynomials = [561,753].

$K2$=15, d_{free} =18, Encoder Polynomials = [63057, 44735].

- F=512: Frame length of F = 512 bits is selected for all of our simulations.
- $P1$=0, $P2$=2, $P3$=8, $P4$=16 are the different number of ancestor-best survivor paths that we examine in our simulations.
- $L1$ = 0, $L2$ =1 are the values we select for simulation. L = 0 means that we use the conventional M-algorithm without considering ancestor metrics and L = 1 meaning that the only ancestor considered for survivor decision is the ancestor level 1, that is the father node.

For Survivor decision in each step of the trellis advancing the following procedure is followed:

1- P paths with the best L^{th} generation ancestor metric are flagged as the survivor paths.

2- All paths are ordered based on their path metric and their rank is stored, to be used by their successors.

3- M-P other paths which have the best metric among the current paths are also flagged as survivor by flagging from the start of the ordered paths.

4- The flagged paths are extended to the next step.

Bit Error Probability and Frame Error Probability for the aforementioned values of L, P and K are shown respectively in Figure 4-8 and Figure 4-9.

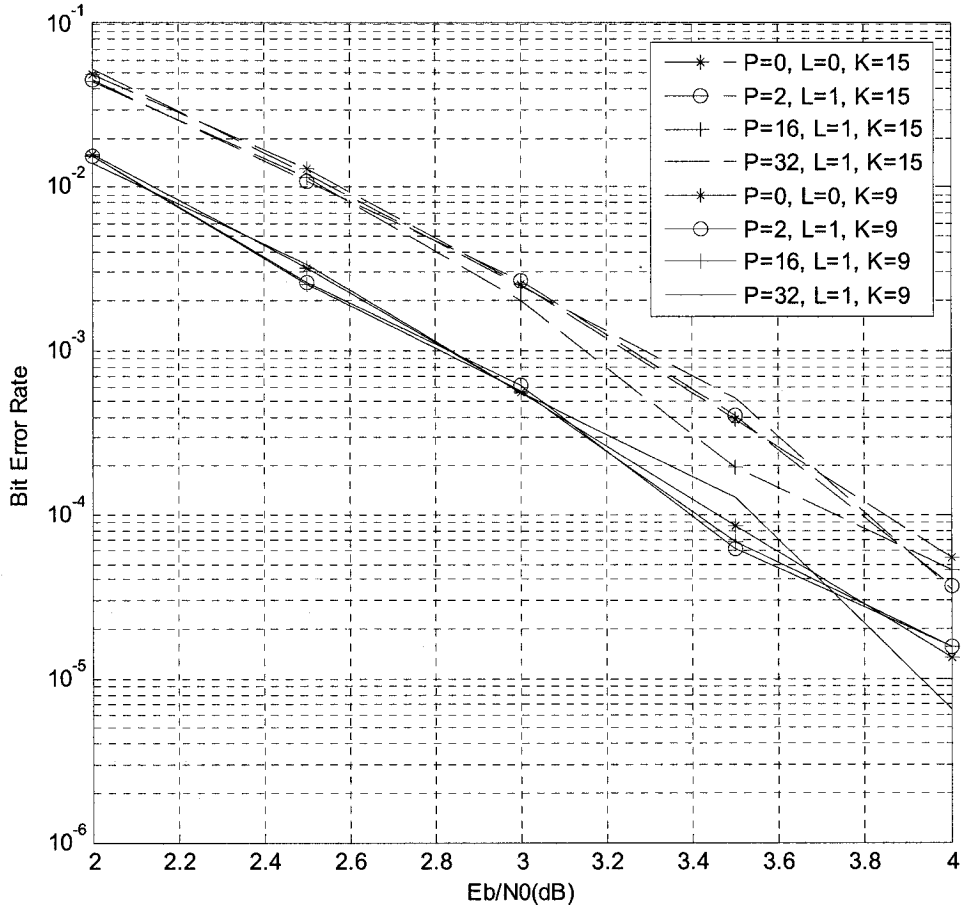

Figure 4-8 Bit Error Probability, Frame Length = 512 bits, Rate: R=1/2, Constraint Length: K = 15, K = 9.

In both Figure 4-8 and Figure 4-9 we see that there are some cases where the error performance in the proposed algorithm, is slightly better than the conventional M-algorithm scheme. For example, both Bit error performance and Frame error performance of the system at E_b/N_0= 3.5dB for the case of P=16, L=1 is slightly better than other cases.

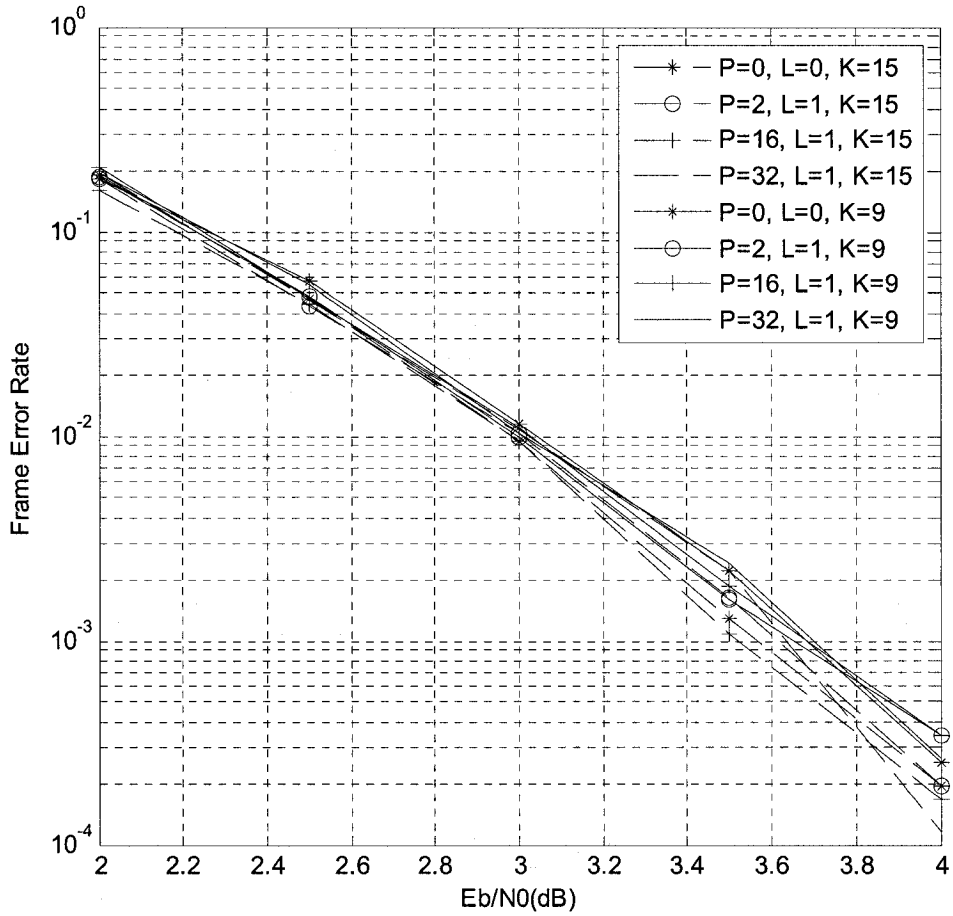

Figure 4-9 Frame Error Probability, Frame Length equal to 512 bits, R=1/2, Constraint Lengths 15, 9.

If we observe Figure 4-8, we see that the code with K=9 offers better error performance than the code of K=15. It is due to this fact that error event in longer constraint

length codes is longer because they have higher minimum distance and correct path recovery takes more time for them. Nevertheless, if we consider Figure 4-9, we will see that the code with k=15 has better Frame error performance than the code with K=9. It is because the longer code has a better minimum distance.

However, it is clearly shown that all the error performances of different P, L are very close to each other. It can be due to this fact that AWGN does not have effect of abrupt noise. Therefore, if we want to have a good error performance for the AWGN channel, we need to look for other ways of correct path loss recovery.

In the next chapter we introduce another reduced complexity trellis based convolutional decoder that uses the M-algorithm iteratively and changes the number of survivor adaptively based on the coding system error.

4.5 Summary

In this chapter, we explained the procedure of decoding in the M-algorithm convolutional decoder and also introduced the catastrophic error caused by the correct path loss, in the reduced state trellis decoders and the M-algorithm. Then we introduced abruptly induced and accumulated noise and their different effects as the sources of correct path loss. To avoid the correct path loss caused by the abrupt noise, we proposed an algorithm for survivor decision in the M-algorithm. Simulation results illustrated the proposed algorithm has slightly better performance in some cases, than the conventional M-algorithm in the presence of AWGN. However, since the algorithm has been designed for the systems with the abrupt noise, we didn't see a high increase in the error performance for AWGN channel. Therefore, we will introduce another reduced state trellis decoding algorithm that can have better performance in the presence of AWGN in the next chapter.

Chapter 5

ADAPTIVE M-ALGORITHM CONVOLUTIONAL DECODER

In the previous chapter, we noticed that although M-algorithm can be employed in the decoding of long constraint length convolutional codes with a low complexity, it also can lead to a poor bit error performance if the correct path loss happens. Thus, we can say that instead of decreasing the probability of error, the high minimum distance of the long constraint length codes, increases error sequence length and as a result, the bit error rate of the system.

Therefore, if we want to use M-algorithm, we should prevent correct path loss or provide algorithms of correct path recovery. Increasing the number of survivors or M in the M-algorithm can decrease the probability of the correct path loss.

In the currently used M-algorithm for decoding the convolutional codes, the number of survivors is fixed for the whole process of decoding. Even if data framing is used to decrease the length of the error event caused by the correct path loss, a fixed number of survivors is used for the M-algorithm for all frames. In this scheme, if we increase the number of survivors to reduce the probability of correct path loss, we have increased the complexity of the system for all the frames.

On the other hand we know that in most of the applications, even with a low value of M, the probability of correct pass loss is very small. This characteristic of the system can be employed to reduce its complexity. Hence, if we can provide a scheme that keeps small number of survivors in most of the cases, and in the case of correct path event it uses a higher number of survivors, we can tremendously decrease the complexity of the decoding system.

Based on this idea we propose Adaptive M-algorithm Convolutional Decoder.

5.1 Adaptive M-algorithm Convolutional Decoder

In the conventional M-algorithm, each frame is decoded once with a fixed value of M. Due to the fact that increasing the value of M decreases the probability of correct path loss, M is taken as high as possible and it increases the complexity of the decoder. On the other hand, it is known that in most of the applications, the correct path is among the best metric paths and even with lower values of M, most of the time correct path loss does not happen.

Taking advantage of this fact, if we can mostly decode with a small value of M and only use big values of M when it is necessary, we can decrease the complexity of the decoding system [23].

We propose a scheme that is based on this idea. In this scheme, we decode each frame with an initial small value of M. If the M-algorithm with that small value of M can decode the frame correctly, we continue decoding of the next frames using the same value of M. Otherwise, if M-algorithm cannot decode the frame correctly, other iterations of decoding with higher values of M is performed for the same frame. This process can be repeated every time that error happens.

Therefore, in this method, higher values of M and a more complex decoding process is used only if it is necessary. As a result the average complexity of the system is low.

Further iterations of the decoding with a higher number of survivors are performed if the system is informed of erroneous decoding in the last round of decoding. Therefore, an error detection mechanism should be included in the decoding system. Since all of our efforts are to reduce the complexity of the decoder, the additional error detection process should not add a significant complexity to the system.

5.1.1 Error Detection Algorithm

Considering the fact that the error detection process should be a powerful and meanwhile low complexity mechanism, Cyclic Redundancy Check Codes (CRC) which was explained in Chapter 3 is selected for error detection purpose.

CRC bits calculated for each frame of data and are added to the end of the frame, then, encoding in a convolutional encoder is taken place and after transmission over the channel. In the receiver the frame is convolutionally decoded and then the decoded frame is checked by its CRC bits to make sure if an error free frame has been received.

Indeed, CRC bits are the remainder r of the division of the frame of data of W by a known number of d. This remainder of r along with the frame data of W is sent to the receiver. In the receiver the received frame of data is divided by d and the remainder is compared with the received CRC bits. If no error has been occurred in the received bits, the remainders will be the same and otherwise they will be different.

Obviously CRC error detection is not foolproof, because there are many different message strings that give a remainder of r when divided by d. In fact, about one out of every d randomly selected string will give any specific remainder. Thus, if our message string is distorted in transmission, there is a chance (about $1/d$, assuming the corrupted message is random) that the CRC bits agree with the distorted frame of data. In such a case the error will remain undetected. Nevertheless, by making d large enough, the chances of a random error going undetected can be made extremely small. However, we should consider this fact that longer CRC bits decreases signal to noise ratio of the system and slightly increases the complexity of the system and can lead to increase in the probability of error.

In the next section, we introduce our proposed coding system.

5.2 The Proposed System Model

The proposed system model is depicted as Figure 5-1. This figure shows the encoding and decoding algorithm of the Transmitter and the Receiver.

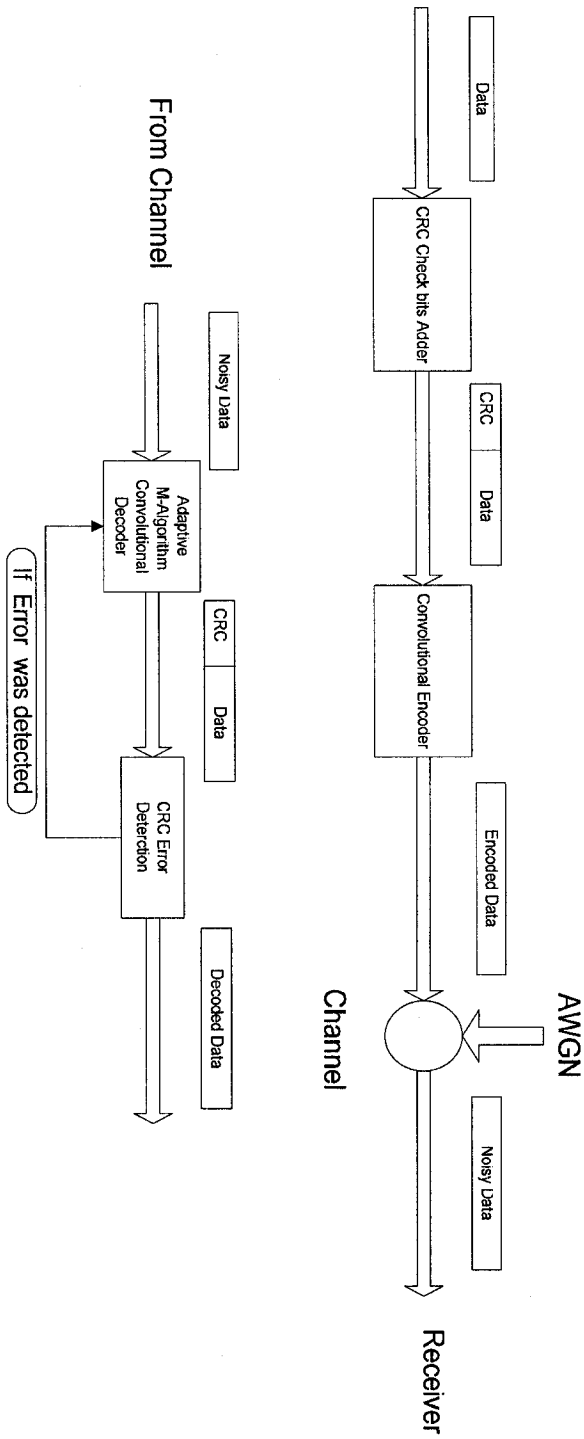

Figure 5-1 The Proposed System Model.

5.2.1 Transmitter Side

To be able to detect the error, CRC bits are added to each frame of sent data. Therefore, in the transmitter at the first step the Data frame passes through a CRC encoder and this encoder generates CRC bits for frame of data and adds them to the frame. The new frame consists of CRC and information bits. This CRC added frame, in this stage is encoded by a convolutional encoder. The convolutionally encoded data is sent to the receiver through the communications channel. In our discussions, we consider this channel an Additive White Gaussian Channel or AWGN.

5.2.2 Receiver Side

At the receiver side, the noisy frame is received from the channel. This frame at the first step is decoded by an M-Algorithm convolutional decoder. The M number of survivors of the M-Algorithm is set to a predefined small initial value. Then the convolutionally decoded frame is passes through the CRC error detection module to check for any possible error. If no error is detected, the decoding process on the frame is accomplished and the process of decoding for the next frame is started. However, if any error is detected in the frame in this stage, the number of survivors of the M-algorithm is increased and another iteration of decoding is performed on the frame with the new M of the M-Algorithm convolutional decoder.

This process can be repeated if the error happens again. Since the value of M is increased every time that the decoding is repeated for a frame, the number of survivors might become too high for very noisy frames and as a result decoding complexity increased unexpectedly. To avoid too high values of M a predefined maximum value of M as M_{max} is

selected. If M reaches M_{max} and still error in the decoded frame is detected, the frame is left and labeled as error frame and no more effort for its decoding is performed. Although this limit on M can introduce some errors to our system, it prevents the system going through long time consuming loops of decoding with high values of M. Hence, a trade-off between the probability of error from one side, and the complexity and time consumption of the decoding system from the other side, leads to the appropriate value of the M_{max}.

5.3 CRC Code Selection

As mentioned before, CRC bit are added to the frame of data in the transmitter to provide the necessary information for error detection in the receiver side. Hence, error detection capability of the applied CRC code should be powerful enough for the desired probability of error.

It has been shown in [24] and [25] that if all the error patterns are considered equally likely, a CRC code of length r can detect the below types of errors in the frame.

1- All single-bit errors.

2- All double-bit errors as long as the generator $G(x)$ of the CRC code has a factor of x.

3- Any odd number of errors, as long as the generator G of the CRC code has a factor of $x+1$.

4- Any burst error of length less than r.

5- With probability of greater than1-1/2^r, all burst errors with the length of greater than $r+1$.

Therefore we can say that the upper bound of the probability of undetected error of a CRC code with length of r is $1/2^r$ or 2^{-r}.

Table 5-1 Some Common CRC Codes, Their Generator Polynomials and Their Undetected Error Upper Bound.

CRC Name	CRC Length (r)	Generator	
		Polynomial	Undetected Error Upper Bound($1/2^r$)
CRC-12	12	$x^{12}+x^{11}+x^3+x^2+x+1$	2.4 E-4
CRC-16	16	$x^{16}+x^{15}+x^2+1$	1.5 E-5
CRC-CCITT	16	$x^{16}+x^{12}+x^5+1$	1.5 E-5
CRC-32	32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	2.3 E-10

Based on the above mentioned facts, for different desirable probability of errors, we should select the appropriate CRC code. As it is shown in Table 5-1 CRC-16 and CRC-CCITT which both have length of r =16 have less than 1.5 E-5 probability of undetected error that is a good value for many of applications. However, if we want to use convolutional codes with longer constraint length and as a result less probability of errors, we have to use longer and stronger CRC codes. A common long CRC code that is used in most of the very low error applications like in PKZip, Ethernet, AAL5 (ATM Adaptation

Layer 5), FDDI (Fiber Distributed Data Interface), the IEEE-802 LAN/MAN standard is CRC-32 which can provide probability of undetected error of less than 2.3 E-10.

Now by performing Monte-Carlo simulation we show the error performance of the system and also its complexity.

5.4 Simulation and Numerical Results

In this part we perform Monte-Carlo simulation and compare the effect of different parameters of the system like on the performance of the system.

We compare different convolutional codes which all have been selected from the optimum minimum distance codes introduced in [22]. Table 5-2 is showing some of the codes we use in our simulations.

Table 5-2 Rate 1/2 Optimum Distance Codes introduced in [22].

K	Code Rate (R)	Generators	Minimum Distance (d_{free})
5	½	23, 35	7
7	½	133, 171	10
9	½	561, 753	12
11	½	3345, 3613	14
15	½	63057, 44735	18

The starting value of M is set to 2 for all different constraint length codes and every time that we encounter an error, we double the value of M. The maximum possible M_{max} for the code with constraint length K can be 2^{K-1} that is all number of paths. When $M = M_{max}$

$= 2^{K-1}$ the algorithm is acting like Viterbi algorithm and all the nodes of the trellis are survive and used for decoding.

In general we can say that if $M_{max} = 2^{K-1}$,our algorithm tries to use reduced state Viterbi algorithm at the first rounds of decoding but if it could not decode the frame correctly with the small values of M, it finally uses the Viterbi algorithm and keeps all possible paths in each step of trellis advancing.

For all of the codes used in our simulations we use $M_{max} = 2^{K-1}$ except the K=15 code. In this case since $M_{max} = 2^{K-1} = 2^{14}$ =16384 is a very high value and this number of survivors dramatically increases the complexity of the M-algorithm decoder, we use a smaller value of M_{max} =1024 and some simulation also performed for the M_{max} =16384. For the case that M_{max} =1024 we never reach to the Viterbi performance of the code K =15 and if it cannot decode the frame correctly, it just leaves the frame as error for the sake of lower complexity.

Two different frame lengths of L=512, L =102 are examined in our simulations.

For error detection purpose, CRC codes with length 16 and 32 are selected. CRC-CCITT-16-bit and CRC-32 which have been introduced in [19] are very common in communication purposes and we employ them in our simulations. As it has been shown in Table 5-1 each of these CRC codes have different error correction capabilities and each of them can be useful for different Error performance areas and different frame lengths.

For the error performance study of the system we compare the Frame Error Rate and also the Bit Error Rate of the coding system over Additive White Gaussian Noise (AWGN) Channel. In our simulations the code length of the used convolutional code is one of the

codes of the Table 5-2. In each following section we will bring forth the simulation results for specific parameters. Also in all of our simulations we use soft decision decoding.

5.4.1 Frame Length: 102 Bits, CRC Code: CCITT-16

Figure 5-2 shows the Frame Error Rate for convolutional codes with K=5, 7, 9, 11, 15. Iterative M-algorithm decoding scheme is used with initial M=2 and for different codes different M_{max} are selected. CRC- CCITT-16-bit has been used for error detection of all of the codes of Figure 5-2. We know that this code provides an undetected error probability of equal or less than 1.5E-5.

We also have shown in the Figure 5-3 the complexity of the algorithm based on the average number of survivors for each frame. For calculation of the average number of survivors, we add all the number of survivors of different decoding tries and it will be the total number of survivors for that frame. Then, we take the average of this number on the all decoded frames in our simulation. For example, if for a frame we start the decoding process with M=2 and the frame is not decoded correctly and we use the M-algorithm again with M=4 and this time the frame is decoded correctly, the total number of survivor for this frame is M_{tot}=2+4=6.

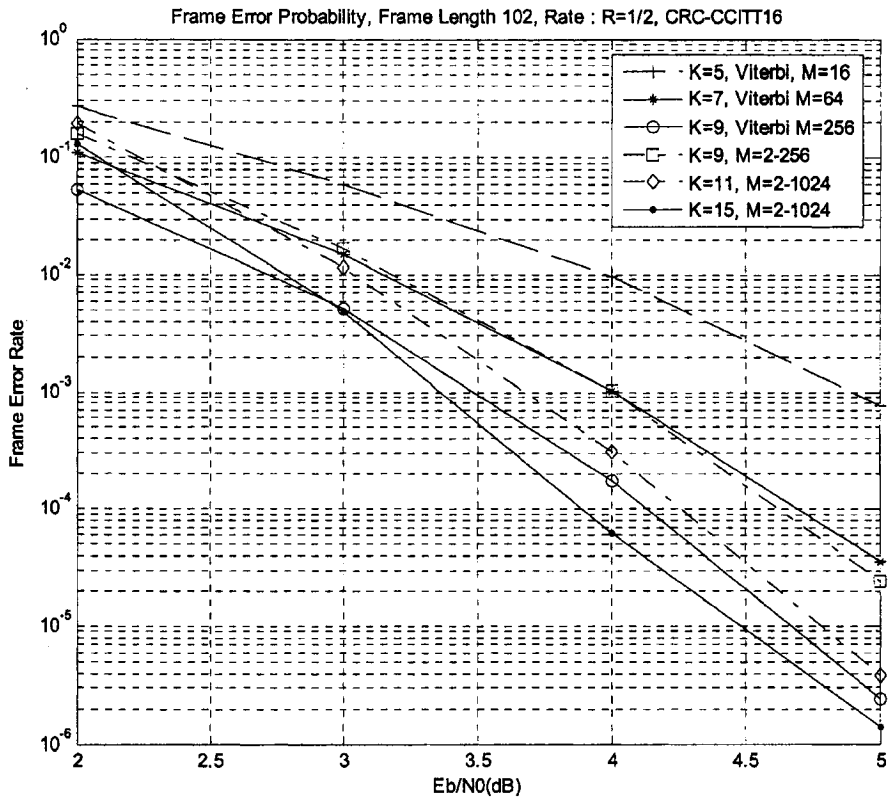

Figure 5-2 Frame Error Rate for the convolutional code with rate R=1/2 and CRC-CCITT 16 Bit and the frame length of 102 bits for different constraint lengths and number of survivors.

In Figure 5-2 we notice that the probability of error of the Adaptive M-algorithm of the codes with K=9, 11, 15 is much better than the Viterbi algorithm decoding of the codes with K=5, 7, 9. This performance improvement happens while the number of the survivors in the adaptive algorithm is less than the number of survivors of Viterbi algorithm with

K=9, 7 in E_b/N_0 of 3dB and higher. Also the number of survivors for the adaptive M-algorithm is even less than the Viterbi algorithm with K=5 for E_b/N_0 of 3.5dB and greater.

The number of survivors for the adaptive M-algorithm is also much less than the conventional M-algorithm. In the conventional M-algorithm the number of survivors is fixed and for having the same error performance as the adaptive M-algorithm, M should be equal to the M_{max} for the conventional M-algorithm and hence the average number of survivors is also equal to M_{max}. In Figure 5-3 we see that for the adaptive M-algorithm the average number of survivors especially for higher E_b/N_0 is much less than M_{max} or the average number of survivors in the conventional M-algorithm.

It is also shown that although the M_{max} for the code with K=9 is 256 (which is equal to the Viterbi algorithm for this K), the frame error rate of the adaptive algorithm is worse that the Viterbi algorithm. The reason is that in the adaptive algorithm, we add the CRC bits t the frame and consequently the energy assigned to each coded Bit is decreased. For the frame with length 102 and CRC with length 16, the actual E_b/N_0 is decrease corresponding to the ratio of the Frame length before CRC to the Frame length after CRC that here is 102/ (102+16). This value is equal to *"10*log (102/118) dB"* that is about 0.65dB.

If we look at our simulation results we see that the difference between the adaptive M-algorithm and the Viterbi algorithm for K=9 in Figure 5-2 is about 0.65dB that we are expecting.

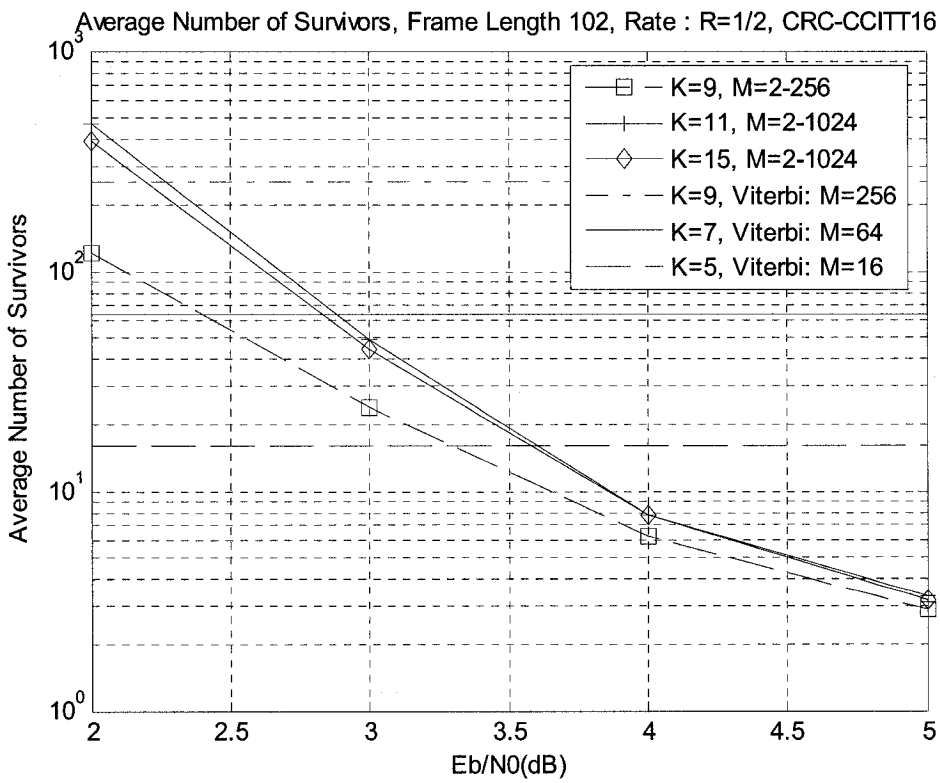

Figure 5-3 The Average Number of Survivor for the frame length of 102 bits and CRC CCITT- 16 Bit.

Although in the communication systems that data is sent and received in the frame format, the frame Error Rate is the main assessment for the error performance of the system, knowing the Bit Error Rate of still contains useful information. Figure 5-4 shows the Bit Error Rate of the codes with the length of K=5, 7, 9, 11, 15. Still we see that the adaptive algorithms outperform Viterbi algorithm and conventional M-algorithm in higher values of E_b/N_0. For lower E_b/N_0 the codes with longer constraint length don't offer better bit error performance this is because in this area, Signal to Noise ratio is very low and none of the codes can prevent correct path loss and when correct path loss happens its recovery is more difficult in longer constraint length codes since they have higher d_{free} and it leads to higher Bit Error Probability in for longer constraint length codes.

Also another reason for the poor performance of the adaptive algorithm in the lower E_b/N_0 area is that since we add CRC bits to the frames of data in the adaptive algorithm, we decrease the energy assigned to the each sent Bit and in lower areas of the E_b/N_0 energy is a critical parameter for good performance and then this decrease in the energy caused by CRC bits degrades the error performance of the adaptive M-algorithm in very low areas of the E_b/N_0.

It is also seen that for $E_b/N_0 = 5dB$, the frame error rate of the code with K=15 is approaching the other codes. This can be because of this fact that in this area Frame error rate is very low and the applied CRC- CCITT-16-bit does not have the capability of detecting some of the happened errors in this area.

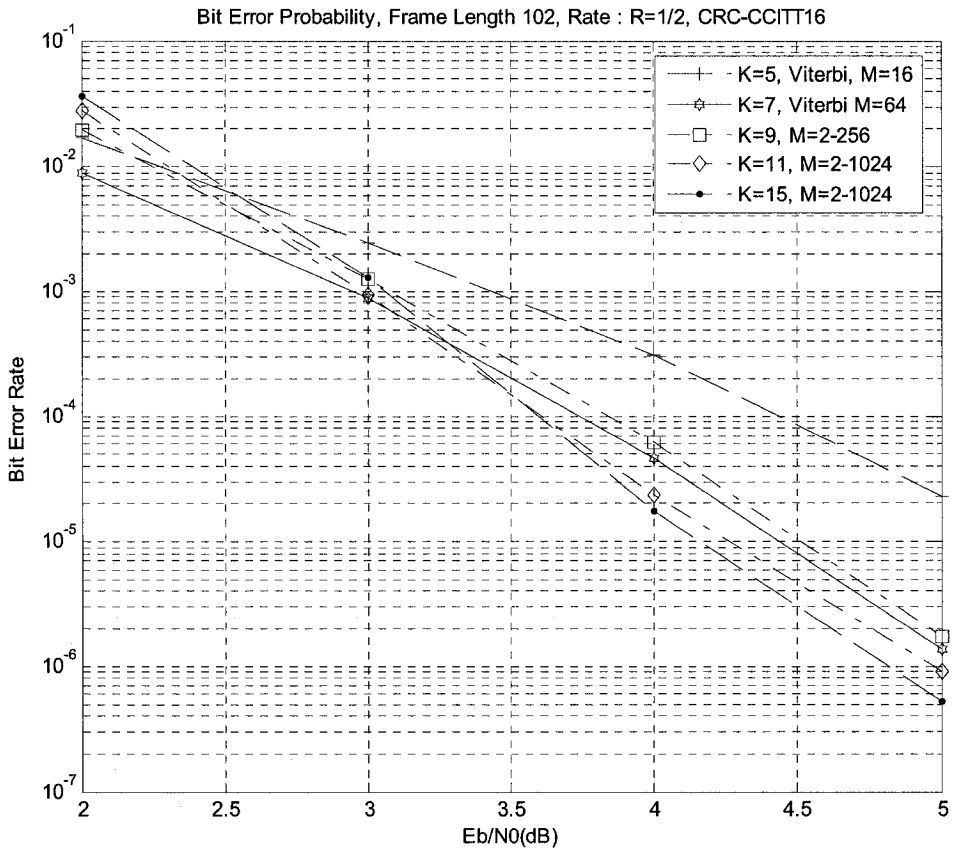

Figure 5-4 Bit Error Rate for the convolutional code with rate R=1/2 and CRC-CCITT 16 Bit and the frame length of 102 bits for different constraint lengths and number of survivors.

5.4.2 Frame Length: 512 Bits, CRC Code: CCITT-16

We have also studied the performance of our proposed adaptive M-algorithm convolutional decoder for decoding of the longer frame lengths.

The frame length that we have used for this purpose is L=512. In this case the energy decrease in the data bits due to CRC bits is smaller that the Frame with Length L=102 and then we expect that the performance improvement of the adaptive M-algorithm even be better than the shorter frame case.

Figure 5-5 shows the Frame Error Probability of the coding system when the frame length is 512 bits. It is seen that codes with longer constraint lengths outperform the shorter codes. However, we see that the probability of frame error even for the code with K= 15 is approaching the other codes for E_b/N_0 = 5 and expectedly for higher Signal to Noise Ratios. This limitation is imposed by the CRC error detection code. Since we use CRC-CCITT-16 for the purpose of error detection and its ability of error detection is limited, we see that for very low error areas, even longer convolutional codes cannot be helpful unless we use a more powerful CRC error detection code.

Figure 5-6 shows the Average Number of Survivors of the employed cases. It is seen that all of the codes have almost the same number of survivors for the E_b/N_0 of 3.5dB or greater than that. Therefore we notice that with the fixed complexity (as a function of number of survivors) we can obtain much better performance from our Adaptive M-algorithms than the conventional M-algorithm or Viterbi algorithm.

Figure 5-7 sows the simulation results of the Bit Error Rate of the coding system. It is seen that the bit error rate of the longer codes is not always superior to the shorter codes. It is because the frame length is longer than the previous simulations and in the shorter

convolutional codes the correct path might be recovered before the end of the frame but it is not the case for the longer codes therefore each error event or correct path loss causes more number of erroneous bits for longer convolutional codes and the bit error performance of the longer codes is degraded for longer constraint length codes.

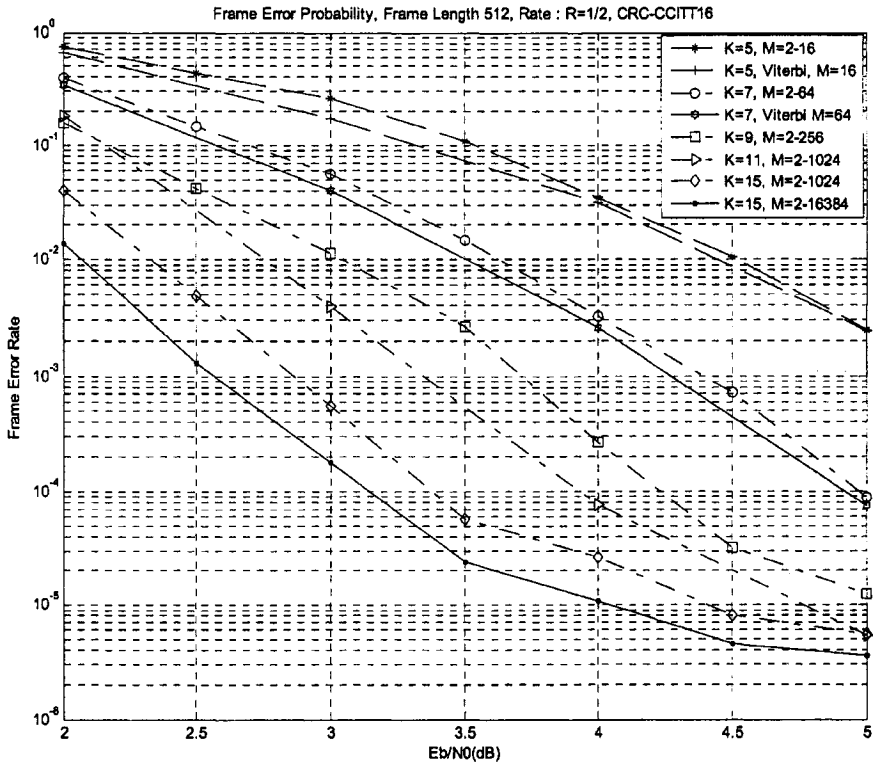

Figure 5-5 Frame Error Rate for the convolutional code with rate R=1/2 and CRC-CCITT 16 Bit and the frame length of 512 bits for different constraint lengths and number of survivors.

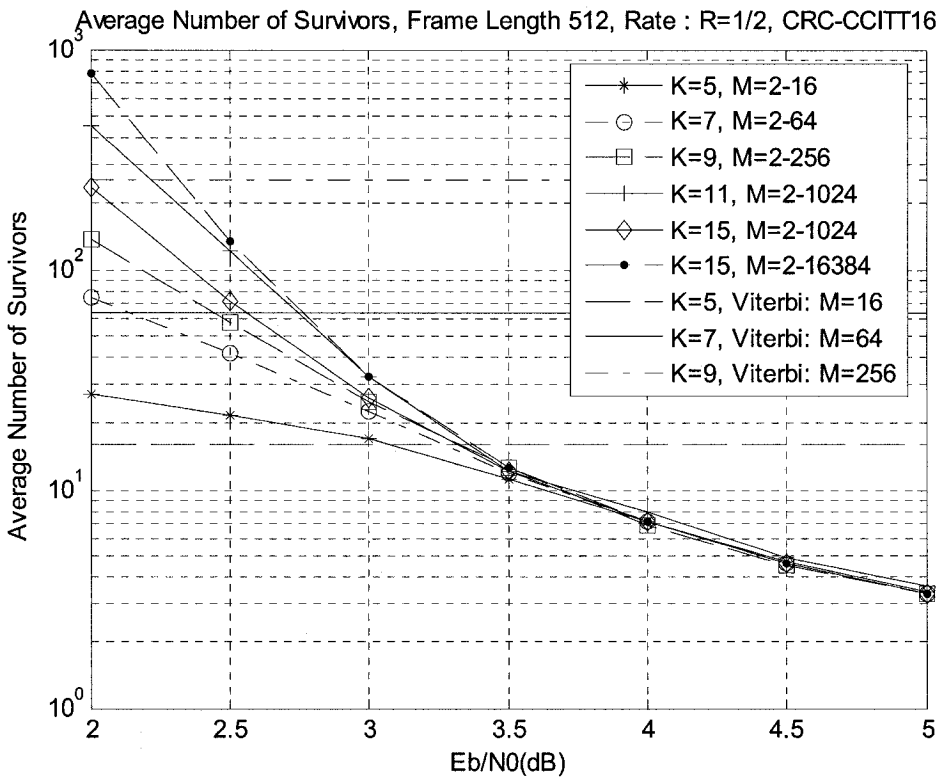

Figure 5-6 The Average Number of Survivors for the convolutional code with rate R=1/2 and CRC-CCITT 16 Bit and the frame length of 512.

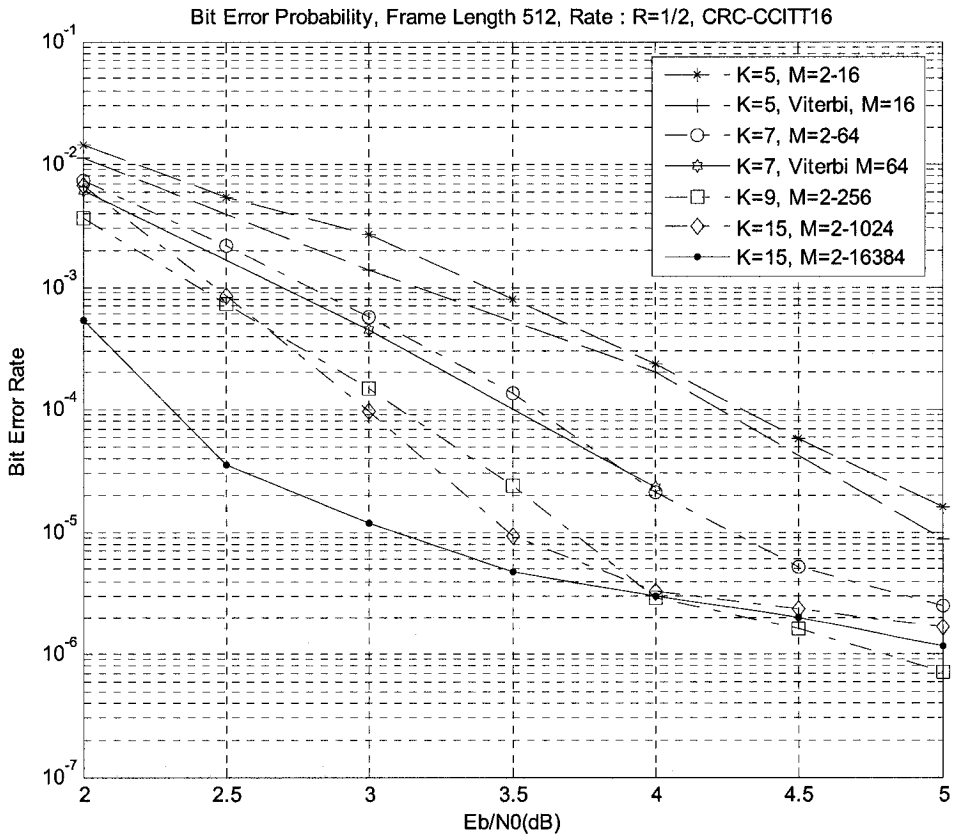

Figure 5-7 Bit Error Rate for the convolutional code with rate R=1/2 and CRC-CCITT 16 Bit and the frame length of 512 for different constraint lengths and number of survivors.

5.4.3 Frame Length: 102 Bits, CRC Code: 32 Bits Length

In this section we show the simulation results when we employ the previously mentioned CRC 32 Bits. Figure 5-8 shows the Frame Error Rate of the Adaptive M-algorithm for K=9, 11 and we compare them with the Viterbi algorithm for K= 5, 7, 9.

As it is seen, in this case since length of the CRC code is rather long in comparison with the length of the frame, the bits of the CRC added frame have twice the CRC-16 Bits case lower energy than the frame without CRC bits added. This is almost equal $1.2dB$ and as a result this CRC code does not let the adaptive algorithm outperforms the Viterbi algorithm for the code with K=9.

However, still the adaptive M-algorithm scheme with a lower number of survivors has better frame error rate than Viterbi algorithm or the conventional M-algorithm schemes for the convolutional code of the constraint length of K=5 in the region of E_b/N_0 equal or greater than $4dB$. As we know the number of survivors for the Viterbi algorithm or the conventional M-algorithm for K=5 is 24 =16 and

Figure 5-9 shows that the adaptive M-algorithm with K=9 and K=11 have less average number of survivors than 16 when E_b/N_0 is respectively greater 3.8dB and 4.25dB and meanwhile they have better frame error performance than the Viterbi algorithm decoding of the code with the constraint length of K=5.

For the Bit Error Performance of the system as shown in Figure 5-10 the adaptive M-algorithm cannot improve the performance of the system even compared to the Viterbi algorithm decoding of the code with the length of K=5 and that is due to the rather long length of the CRC 32 Bits code to the Frame Length.

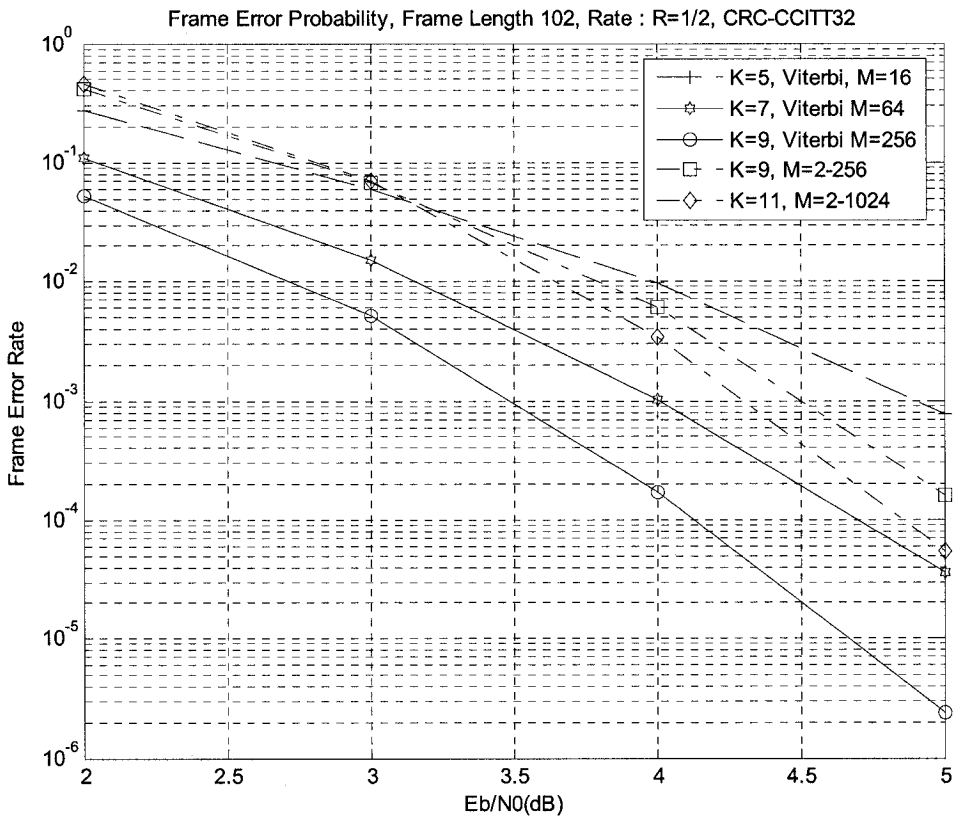

Figure 5-8 Frame Error Rate for the convolutional code with rate R=1/2 and CRC 32 Bit and the frame length of 102 bits for different constraint lengths and number of survivors.

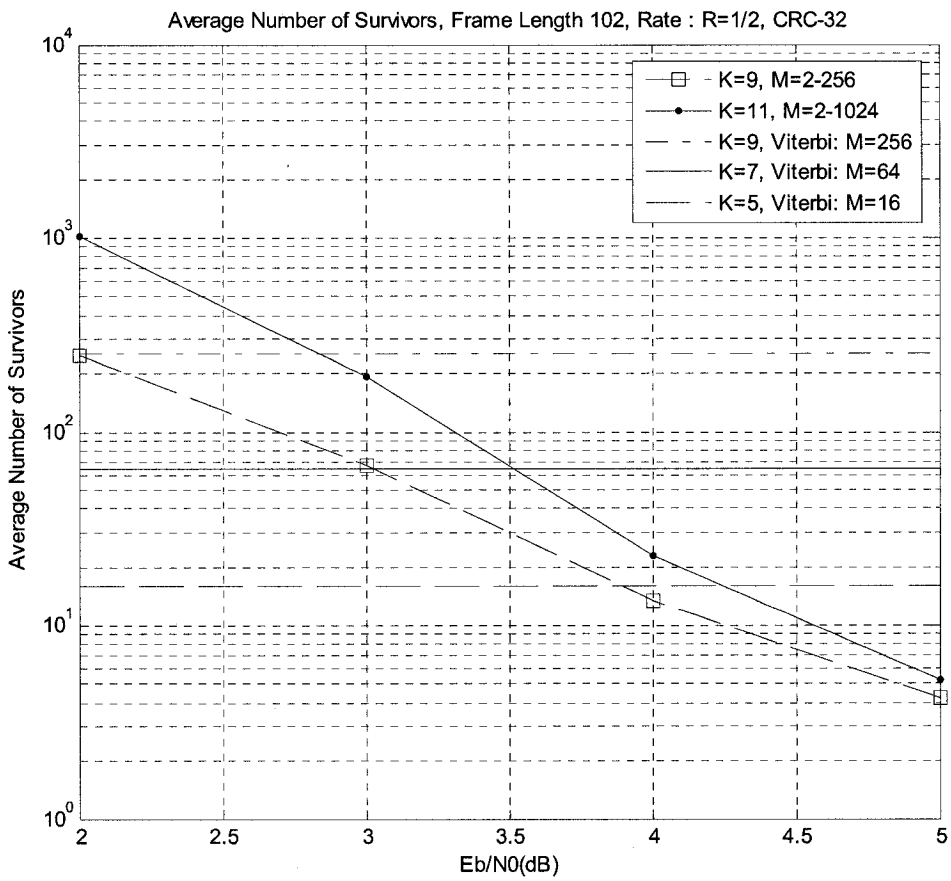

Figure 5-9 Average Number of Survivors for the convolutional code with rate R=1/2 and CRC 32 Bit and the frame length of 102 bits for different constraint lengths and number of survivors.

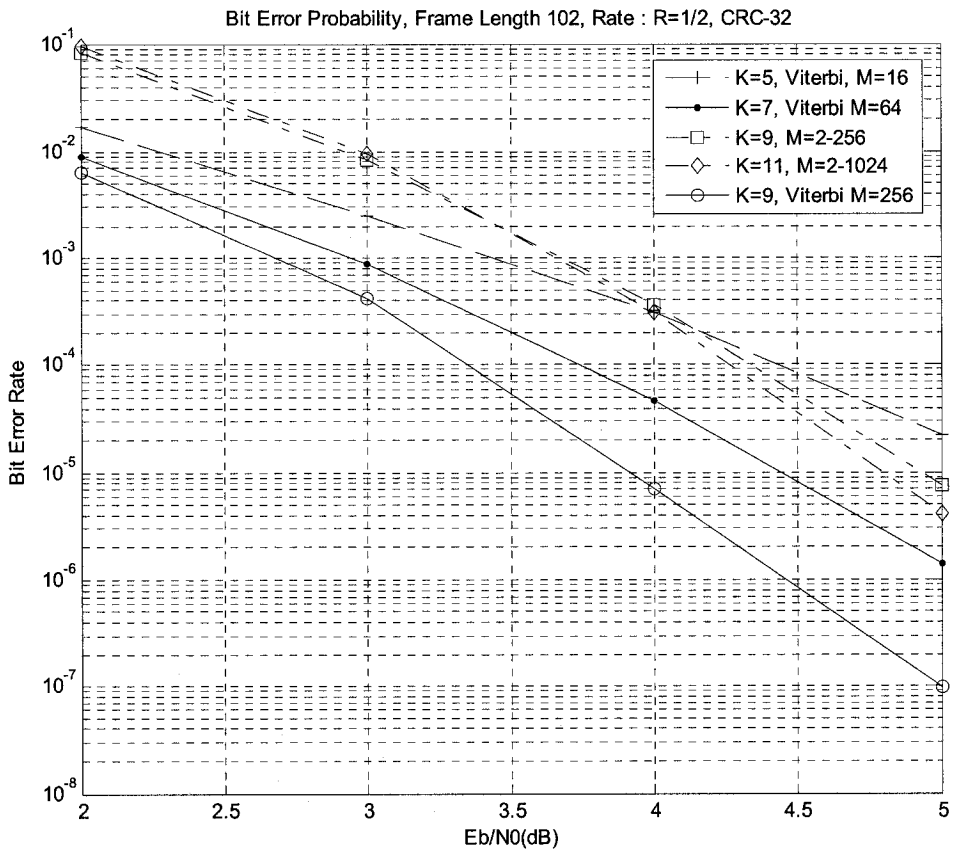

Figure 5-10 Bit Error Rate for the convolutional code with rate R=1/2 and CRC 32 Bit and the frame length of 102 bits for different constraint lengths and number of survivors.

5.4.4 Frame Length: 512 Bits, CRC Code: 32 Bits Length

As shown in the previous section, CRC-32 Bits is too long for the frame of data with shorter length. We have also seen that for very low probability of error areas, CRC-16 Bits is not powerful enough and if we want to obtain lower probability of errors, we have to use longer CRC codes. In this section we show the simulation results of the decoding of the frames of 512 bits and CRC-32Bits.

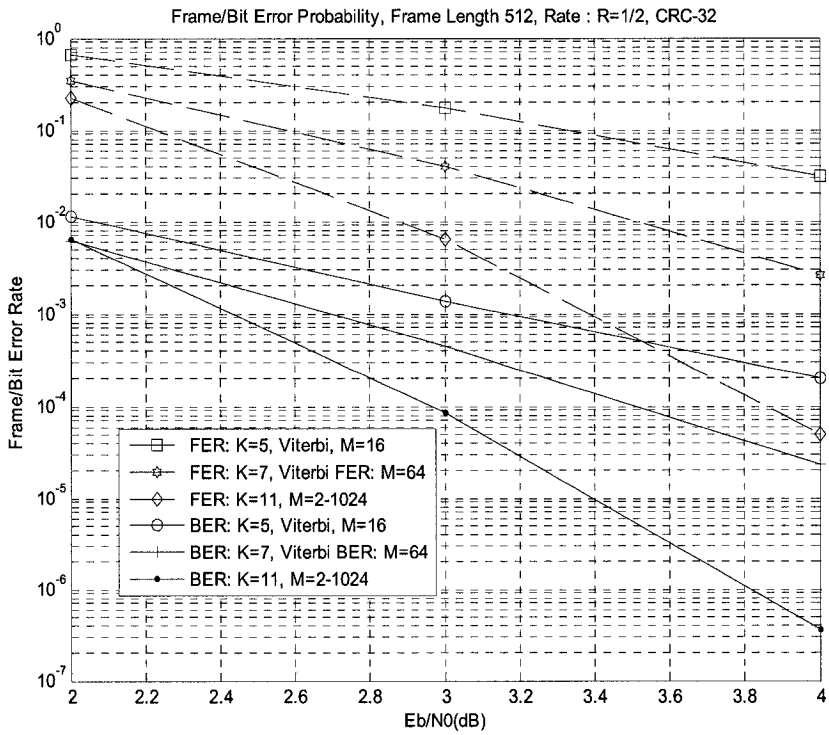

Figure 5-11 Frame/Bit Error Rate for the convolutional code with rate R=1/2 and CRC 32 Bit and the frame length of 512 bits for different constraint lengths and number of survivors.

Figure 5-11 shows the Frame Error Rate and the Bit Error Rate of the decoding of the frames of 512 bits. It is obviously seen that the codes with longer constraint lengths offer better probability of errors. For both of the Bit Error Rate and Frame Error Rate it is the case.

Figure 5-12 also shows the number of survivors of adaptive M-algorithm for the decoding of the convolutional code with the constraint length of K=11. It is seen that the employed algorithm uses even fewer number of survivors than the Viterbi algorithm with code of length K=5 for E_b/N_0 higher than $4dB$.

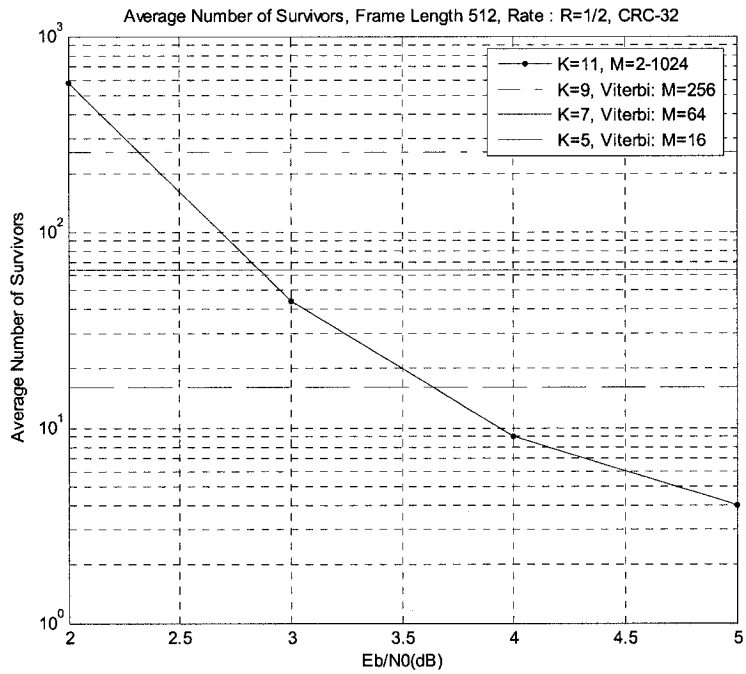

Figure 5-12 Average Number of Survivors for the convolutional code with rate R=1/2 and CRC 32 Bit and the frame length of 512 bits for convolutional code of length 11.

5.4.5 Comparison between Effect of the CRC-32 and the CRC-16 Codes on the Frame Lengths of 512 Bits

It is worth noting to compare CRC-32 and CRC-16 in one figure. Figure 5-13 shows the Frame error Rate of the system when CRC-16 and CRC-32 is used for error detection purpose. As it is shown, the error performance of the longer CRC code is superior to the shorter CRC when long constraint length codes have been employed as the convolutional code. The code with $K = 11$ has the best frame error performance in the area of higher than $3.5dB$ of E_b/N_0.

Figure 5-14 shows the number of survivors for the different used codes. The code with longer constraint length has higher average number of survivors than the other codes and it is because in that code the probability of error is very small and since we use a very powerful error detection code, most of the errors are detected and the decoding is repeated for the frame containing the error. Although this repetition can decrease the probability of error, it increases the average number of survivors. However, in higher E_b/N_0 areas, still the average number of survivors for the adaptive M-algorithm is even fewer than Viterbi algorithm or the conventional M-algorithm offering the worse probability of error.

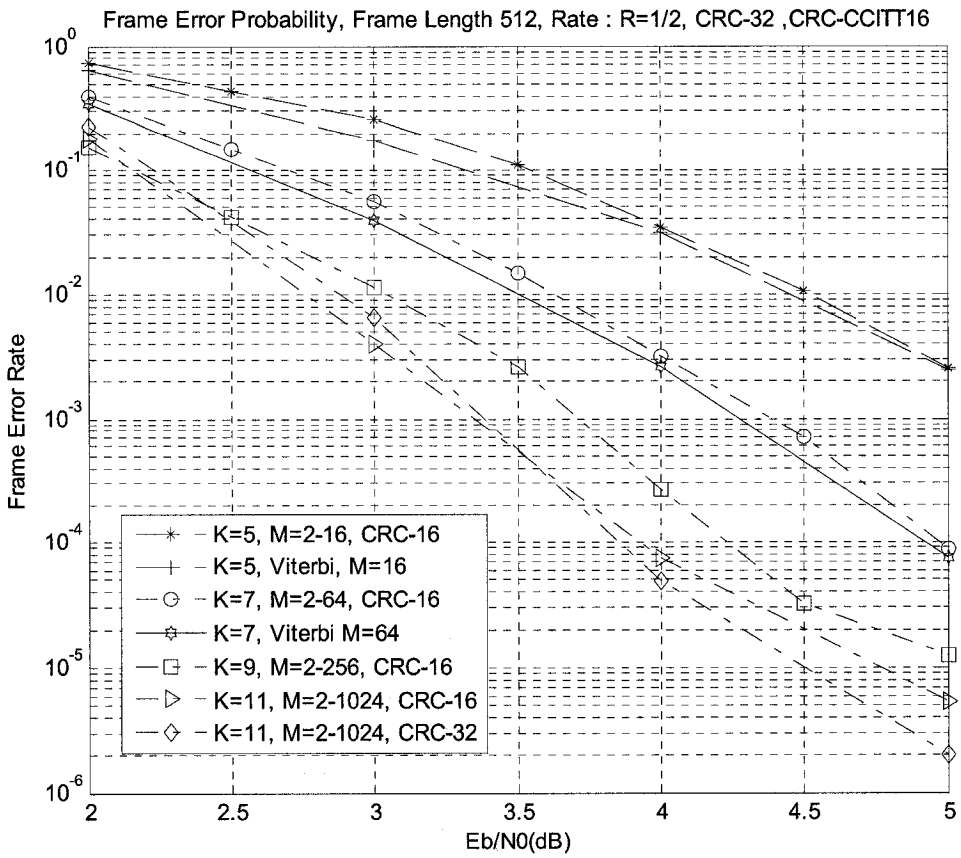

Figure 5-13 The Frame Error Rate for the convolutional code with rate R=1/2 and CRC 32/16 Bit and the frame length of 512 bits.

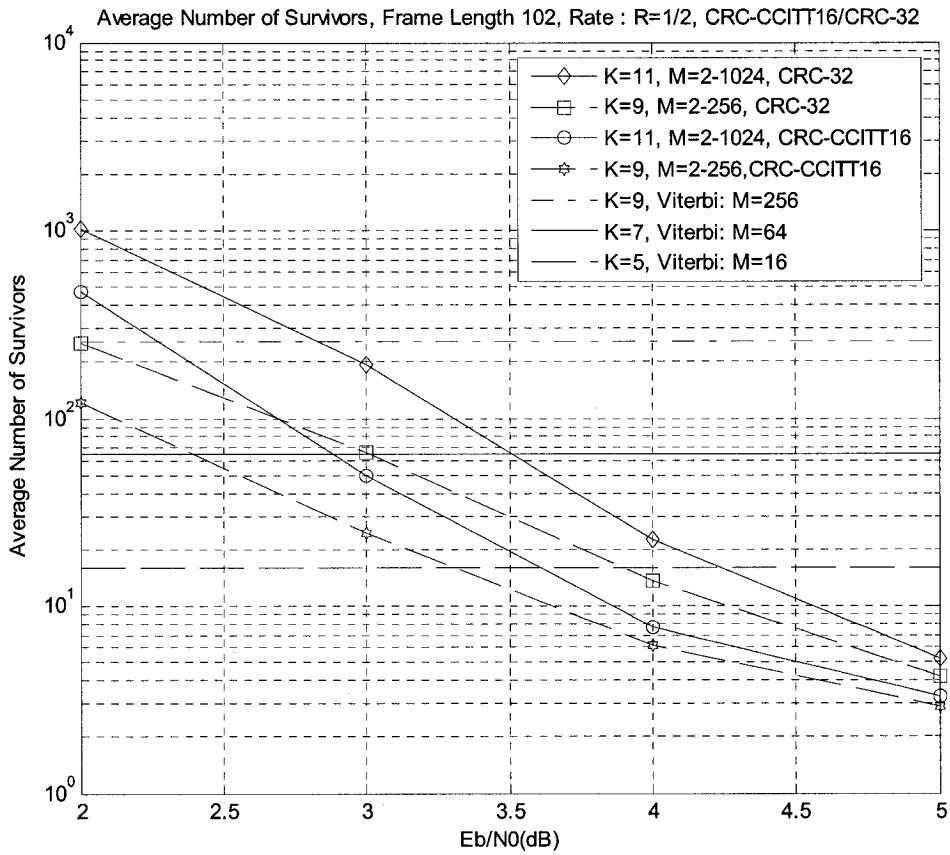

Figure 5-14 Average Number of Survivors for the convolutional code with rate R=1/2 and CRC 32/16 Bit and the frame length of 512 bits.

5.5 Summary

In this chapter we explained that for decreasing the probability of the correct path loss we can increase the number of survivors in the M-algorithm. Based on this characteristic of the M-algorithm that in most of the times even with smaller number of survivors, the correct path loss does not happen; we introduced the adaptive M-algorithm Convolutional decoder. In this algorithm we increase the number of survivors only when an error in the decoded frame is detected. CRC error detection codes are used for the error check of the decoded frames.

We proposed our coding system model and then presented the simulation results. Simulation was performed assuming that the communication channel is AWGN channel and convolutional codes with different lengths combined with different CRC code length were compared to each other. It was shown that in most of the cases the proposed M-algorithm Based Convolutional Decoder outperforms the Viterbi algorithm or the conventional M-algorithm. The complexity of the system in terms of the average number of survivors also was compared for the different cases. It showed that the new algorithm can be implemented with less complexity in comparison with the Viterbi algorithm or the conventional M-algorithm which have the same error performance.

Chapter 6

CONCLUSION

We conclude our thesis with a brief summary of what was brought forth in the previous chapters and then very short suggestions wrap up the thesis.

6.1 Summary

There are different algorithms for decoding of the encoded sequences. The Viterbi algorithm is one of the most popular convolutional decoders. Although this algorithm is an ML optimum decoder, it suffers from the high complexity in the decoding of long constraint length codes. On the other hand, we know that for obtaining lower error rates in

the convolutional codes, we need to increase the constraint length of the employed code. Therefore, we should find simpler decoding algorithms for the decoding of the longer constraint length codes.

Due to the low complexity of the simplified Viterbi algorithms or the reduced state trellis decoding algorithms, they are practical for the decoding of long constraint length codes. M-algorithm is one of the reduced state trellis decoders, but it suffers from catastrophic error caused by the correct path loss in the algorithm. In this thesis we proposed two different ways of correct path recovery based on M-algorithm convolutional decoder.

The first method is called Ancestor Based Survivor Decision in M-algorithm Convolutional Decoder. This method has been designed for the cases that an abrupt noise degrades the metric of the correct path suddenly. In this case, knowing that the ancestor of the path has been the correct path in its time, if we keep all the successors of that ancestor, we certainly have kept the correct path. We explained the algorithm of decoding for the proposed method and also simulated it for the AWGN channel. Simulation results showed only slightly improved error performance in some cases. It was mostly for the reason that the AWGN does not act as abrupt noise. As a result we decided to find a method applicable for the AWGN channel that is our second proposed method.

The second proposed method is called Adaptive M-algorithm Based Convolutional Decoder. This method takes advantage of the low probability of correct path loss. Even using a small number of survivors, the probability of correct path loss is very small; therefore, we suggest using a small number of survivors for most of the decoding attempts, and we use a higher number of survivors only in case of error decoding. The CRC error detection code is used to detect whether the frame is an erroneous one or not.

A coding system based on the combination of a convolutional code and a CRC code was proposed and simulation results also were presented. Simulation was performed assuming that the communication channel is AWGN channel. Convolutional codes of different constraint lengths combined with CRC codes of different lengths were compared to each other. It was shown that in most of the cases the proposed M-algorithm Based Convolutional Decoder outperforms the Viterbi algorithm or the conventional M-algorithm. The complexity of the system in terms of the average number of survivors also was compared for the different cases, and it was shown that for the same error performance, the new algorithm can be implemented with less complexity in comparison with the Viterbi algorithm or the conventional M-algorithm.

6.2 Suggested Future Research

The future research based on this thesis can be performed in two parts. The first part is the analysis and study of the application of the Ancestor-Based Survivor Decision in the M-algorithm Convolutional Decoder for different abrupt noise environments, like the fading channel or the shot noise. Since the design of this algorithm is for this kind of noise, it is expected that in these kinds of environments we obtain better error performance from the proposed method.

Also other research can be done on the Adaptive M-algorithm Based Convolutional Decoder. One of these research can be on the analysis and study of the number of attempts for the decoding of each frame that can lead to the comparison of the algorithm with the sequential decoding algorithms. Also other works on the complexity reduction of the algorithm in the sorting can be useful, since sorting in M-algorithm can increase its

complexity. As a result if its complexity is reduced, then the complexity of the whole coding system is reduced.

BIBLIOGRAPHY

[1] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell Labs Technical Journal,* vol. 27, pp. 379-457, 623, 656, July-October, 1948.

[2] S.Lin and D.Costello, *Error Control Coding: Fundamentals and Applications,* Upper Saddle River, NJ: Pearson Prentice-Hall, 2004.

[3] P, Elias, " Coding for Noisy Channels," *IRE Conv. Rec.,* Part 4, pp. 33-47, 1955.

[4] J. M. Wozencraft and B. Reiffen, *Sequential Decoding,* MIT Press, Cambridge, Mass., 1961.

[5] J. L. Massey, *threshold Decoding,* MIT Press, Cambridge, Mass., 1963.

[6] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, no. 2, pp. 260-269, Apr. 1967.

[7] S.Lin and D.Costello, *Error Control Coding: Fundamentals and Applications,* Englewood Cliffs, NJ: Prentice-Hall, 1983.

[8] T.M. Aulin, "Breadth-first maximum likelihood sequence detection: basics" *IEEE Trans. Commun.*, Volume 47, Issue 2 pp. 208 - 216, Feb. 1999.

[9] R. M. Fano, "A Heuristic Discussion of Probabilistic Decoding," *IEEE Trans. Inform. Theory*, IT-9, pp. 64-74, April 1963.

[10] K. Zigangirov, *"Some sequential Decoding Procedures,"* Probl. Peredachi Inf., 2, pp. 13-25, 1966.

[11] F. Jelinek, "A Fast Sequential Decoding Algorithm Using a Stack," IBM J. Res. And Dev., 13, pp. 675-685, Nov. 1969.

[12] N. Seshadri, C E. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Transactions on Communications,* pp. 313-323, Vol. 42, No. 21314, February/March/April 1994.

[13] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in Proc. *Int. Conf. Commun., ICC 93,* pp. 1064-1070, May 1993.

[14] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory,* vol. 47, Issue: 2, pp. 619 – 637, Feb 2001.

[15] J. Proakis, *Digital Communications,* McGraw-Hill, 2001.

[16] A. Populis, *Probability, Random Variable and Stochastic Process,* Mc. Graw-Hill, 1991.

[17] J. B. Anderson, and S. Mohan, *Source and Channel Coding: An Algorithmic Approach* , Kluwer Academic Publishers:London,pp.297-306, 1991.

[18] A. S. Tannenbaum. *Computer Networks*. Prentice Hall, Englewood Cliffs, NJ USA, 3rd edition, 1996.

[19] William Stallings, *Data and Computer Communications*, 6/e, Prentice Hall, 1999; 7/e, Pearson Prentice Hall, 2004.

[20] C. F. Lin, "A truncated Viterbi algorithm approach to trellis codes," Ph.D. dissertation, Dep. Elect. Eng., Rensselaer Polytechnic Inst., Troy, NY, Sept. 1986.

[21] F. Chan and D. Haccoun, "Adaptive Viterbi decoding of convolutional codes over memoryless channels," *IEEE Trans. Commun.*, vol. 45, pp. 1389-1400, Nov. 1997.

[22] P. Frenger, P. Orten, and T. Ottosson, "Convolutional codes with optimum distance spectrum," *IEEE Commun. Lett.*, vol. 3, pp. 317-319, Nov. 1999.

[23] Seyed Ali Gorji Zadeh, Mohammad Reza Soleymani, "An Adaptive M-Algorithm Convolutional Decoder," in Proceedings of the *IEEE Vehicular Technology Conference*, Fall 2005, Dallas, Texas.

[24] Peterson, W.W. and Brown, D.T. "Cyclic Codes for Error Detection." *Proceedings of the IRE*, January 1961, pp. 228–235.

[25] Tanenbaum, Andrew S. *Computer Networks, Second Edition.* Prentice Hall, 1988.