Near Optimal Decoding for Trellis Coded Modulation

Qi Wang

B.E. NUPT (P.R.China)

M.E. NUPT (P.R.China)

June 2000

A thesis submitted for the degree of Doctor of Philosophy

of The Australian National University

Department of Engineering
Faculty of Engineering and Information Technology
The Australian National University

Errata and Addendum

1. On page 36, we mention that most of the good linear trellis codes in the literature are feedback codes. However, the feedfoward codes are used in all the simulations presented in chapter 6. The reason for using the feedfoward codes is given as follows. Unlike the convolutional-convolutional concatenated codes, for block-convolutional concatenated codes we find that either feedfoward or feedback codes yield the similar performance. But it is difficult to find tail-biting cycles for the feedback codes. In fact, many feedback codes cannot form a tail-biting cycle.

2. On page 70, for equivalent TWL graph shown in Fig. 5-4 (b), the similar illustration can also be found in Wiberg's dissertation [77] and Forney's paper [38].

3. In section 6.3, we can see that, comparing with the modified min-sum algorithm with 10 iterations, a better performance is achieved by using the MLD rule in proposition 1 with proper d and only two iterations at high SNRs.

4. On page 26, replace "$\cdots$ near-ML decoding algorithms $\cdots$" with "$\cdots$ near-ML decoding algorithm $\cdots$".

5. On page 33, replace "$\cdots$ in many cases we have $d^2_{parallel} \leq d^2_{free,c} \cdots$" with "$\cdots$ in many cases we have $d^2_{parallel} \leq 2d^2_{free,c} \cdots$".

6. On page 50, replace "$\cdots$ the i^{th} codeword $\cdots$" with "$\cdots$ the i^{th} symbol $\cdots$".

7. On page 66, replace "there are many the smallest cycles" with "there are many small cycles".

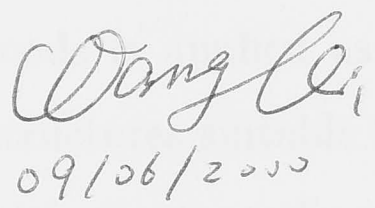
8. On page 109, we find that the parity-concatenated 4-D 16-state Wei trellis codes with the IVA can achieve about more 0.47 dB gain than $\nu = 18$ linear trellis codes with Wei 4-D constellation at a BER of 3.7×10^{-6}.

9. On page 140, reference [7] has been published as
   B. Vucetic and J. H. Yuan, *Turbo Codes: Principles and Applications.* Kluwer Academic, 2000.

Declaration

This thesis contains no material which has been previously accepted for the award of any other degree or diploma at any university, institute or college, and contains no material previously published or written by another person, except where due reference is made.

Canberra, May 2000.



Qi Wang

Department of Engineering

Faculty of Engineering and Information Technology

The Australian National University

Canberra ACT 0200, AUSTRALIA.

Abstract

The objective of this thesis is to develop a novel coding/decoding scheme suitable for current communications systems for providing superior bit error rate performance over existing systems at a high bandwidth efficiency with low complexity and without significant system modifications.

In this thesis, we construct parity-concatenated two-dimensional and multi-dimensional trellis coded modulation (TCM) schemes in which a trellis code is used as the inner code and a simple even parity code is applied as the outer code. Single-parity-check and double-parity-check structures suitable for continuous transmission as well as packet transmission with short, medium and long block lengths are designed. In addition, shaping techniques are combined with parity-concatenated trellis codes to achieve a better bit error rate (BER) performance.

In the receiver, firstly, the iterative Viterbi algorithm (IVA) is developed and new metric functions which take into account *extrinsic* information are derived. Then, based on the Tanner-Wiberg-Loeliger (TWL) graph representations, the iterative two-way algorithms (ITWAs) are derived. To deal with small cycles in parity-concatenated trellis codes, we present modified min-sum/sum-product algorithms in which a normalization function is applied. From the perspective of the TWL graph, the IVA actually is shown to be a simplified version of the ITWAs. Compared with the standard VA, the ITWAs and IVA can achieve significant performance improvement for the parity-concatenated trellis codes.

Based on the ITWAs and IVA, five iterative decoding algorithms (IDAs) are developed. We compare their performance and complexity through the simu-

lations. For the continuous transmission or packet transmission with long (say ten thousands of symbols) or medium (say thousands of symbols) block length, the ITWAs can get slightly better performance than the IVA. However, for short (say hundreds of symbols) packet transmission, the IVA often outperforms the ITWAs. In addition, no matter whether packet or continuous transmission is used, the complexity of ITWAs is much higher than that of the IVA, especially for multi-dimensional trellis codes. Therefore, considering the tradeoff between the computational complexity and BER performance, the IVA is preferable. Due to its low complexity, the IVA can be applied to many current standard systems, such as in high-rate voice band modems or ADSL modems, without or with very little modification.

In simulations, with the trellis shaping, the performance of 1.25 dB away from the Shannon limit at a $BER = 3.0 \times 10^{-5}$ can be achieved by the IVA for parity-concatenated Ungerboeck 256-state trellis codes, but the error floor occurs. Further, using a simple binary BCH code, the error floor can be reduced to 10^{-9} with very little additional cost. For the popular 4-D 16-state Wei code, the numerical results show that about 2.2 dB net gain can also be obtained using the IVA. All these designs have been achieved with low complexity and computation.

Finally, a robust Viterbi algorithm and robust iterative decoding algorithm are studied for conventional trellis codes and parity-concatenated trellis codes under an uncertain noisy environment. Using the *minimax* technique, a simple and effective robust decoder is devised. The numerical simulations show that these robust VA and IVA decoders always outperform the worst mismatched standard decoders and generally perform close to the optimal matched decoder in various mismatched noise environments.

Acknowledgements

I would like to express my gratitude to Dr. Lei Wei and Dr. Rodney A. Kennedy, for guiding and inspiring my research and also for reviewing the draft versions of this thesis.

I am grateful to all of the people in the Department of Engineering and Research School of Information Sciences and Engineering (RSISE) who helped me and made it possible for me to complete my thesis. Here many thanks to Dr. Bob Williamson for his great support during my mid-term review, and Dr. Brian Hart for his valuable suggestion.

Special thanks also go to my colleagues, Mr. Zheng (Bruce) Li, Mr. Yung Chung (Michael) Wei and Mr. Lei Qiu, for their contribution to this thesis by way of participation in numerous discussions over the years.

I would also like to thank both the Australian Government and the Australian National University for providing the scholarships which made this study possible.

Finally, I want to thank my wife and our parents, who gave me incredible moral support all through the years, which make all the work bearable and meaningful.

Contents

List of Figures

List of Tables

Notation

Λ	Signal constellation
d	Number of time units in which the symbol in the error path is different from the symbol of the correct path
$d_{min,(\Lambda)}$	Minimum Euclidean distance between pairs of signal points in the constellation Λ
d_{free}	Minimum free distance of trellis codes
$d_{parallel}$	Parallel transition distance of trellis codes
$d_{free,c}$	Minimum Euclidean distance of coded bits in trellis codes
$B_0,\ B_1, \cdots$	Subsets of trellis codes
U	2D symbol before trellis encoding
u	Bit of symbol U
V	2D symbol after trellis encoding (trellis code)
v	Bit of symbol V
I	4D symbol before trellis encoding
Z	4D symbol after trellis encoding
R	Received signal of trellis decoder (channel output)
S	Number of states of trellis encoder
k	Number of bits in U
$\tilde{k}$	Number of coded bits in U
n	Number of bits in V
2ι	Number of bits in a 4D symbol I
$\tilde{\iota}$	Number of trellis-encoded bits in a 4D symbol I
l	Number of 2D symbols in one packet

m	Number of packets in one block
q	Number of super packets in one super block
ν	Number of shift registers in trellis encoder
E_b	Energy per bit
N_0	One-sided power spectral density of Gaussian noise
t	Time unit

Abbreviations

ACS	Add-Compare-Select
ADSL	Asymmetrical Digital Subscribe Line
AWGN	Additive White Gaussian Noise
APP	*A Posteriori* Probability
BCJR	Bahl-Cocke-Jelinek-Raviv algorithm
BER	Bit Error Rate
BFA	Backward-Forward Algorithm
BIVA	Bootstrap Iterative Viterbi Algorithm
BM	Branch Metric
BP	Belief Propagation
BSC	Binary Symmetric Channel
DMT	Discrete Multi-Tone
IDA	Iterative Decoding Algorithm
LSM	Likelihood Separation Metric
ITWA	Iterative Two-Way Algorithm
IVA	Iterative Viterbi Algorithm
LDPC	Low-Density Parity-Check
MAP	Maximum *A Posteriori* probability
M-D	Multi-Dimensional
ML	Maximum-Likelihood
MLD	Maximum-Likelihood Decision
PC	Parity-Check
PDF	Probability Density Function

PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
RS	Reed-Solomon
SNR	Signal-to-Noise Ratio
SOVA	Soft-Output Viterbi Algorithm
TCM	Trellis Coded Modulation
TTCM	Turbo Trellis Coded Modulation
TWA	Two-Way Algorithm
TWL	Tanner-Wiberg-Loeliger
VA	Viterbi Algorithm

Chapter 1

Introduction

1.1 Background and Motivation

In today's telecommunications market there are dramatically increasing demands for capacity, high date rates, and service quality, which have to be achieved with spectrum utilization efficiency and low complexity of technologies. Meanwhile, error control coding plays a key role in the design of digital communication systems. The aim of error control is to ensure that the received information is as close as possible to the transmitted information, with as low as possible complexity. A well known result from Information Theory is that a randomly chosen code of sufficiently large block length is capable of approaching channel capacity [Shannon, 1948]. However, the optimal decoding complexity increases exponentially with block length up to a point where decoding becomes physically unrealisable. During the past half century, the problem of efficient data communications over a transmission medium impaired by noise and interference has driven much of communication and coding research.

As the landmark developments in coding area, the invention of turbo error control codes [15] and the rediscovery of low-density parity-check (LDPC) codes [87][16] have created tremendous excitement since the gap between the Shannon capacity limit and practically feasible channel utilization is essentially closed. Since then, much attention has being drawn to theoretically understand

the essence of turbo codes and LDPC codes. Motivated by the principle of turbo codes, people have developed a wide range of codes, so called compound codes, which are composed of a collection of interacting constituent codes. Examples of such compound codes include turbo codes, LDPC codes, serially concatenated codes [96] and various product codes [57].

Along with the great success of binary turbo codes, a similar idea to turbo codes can be naturally extended to the bandwidth-limited regime. Some powerful schemes, such as turbo trellis coded modulation [80][81][95] and multilevel coding with binary turbo codes [107], were proposed, and the superior performance close to the Shannon limit have also been achieved.

However, after investigating the existing turbo-style coding/decoding schemes, we find that the popular Forney's conventional concatenated code [35] (i.e, using a simple trellis inner code, a powerful block outer code and a block table-like interleaver) and the famous Viterbi algorithm (VA) [1], which have been widely used over last 40 years, are not considered. Therefore, the main motivation of this thesis is to develop a novel coding/decoding scheme based on current communications systems for providing superior BER performance over existing systems at a high bandwidth efficiency with low complexity. Without or with very little modification to these existing systems, this coding/decoding scheme may be employed in many current applications, such as telephone, satellite and microwave digital radio channels.

1.2 Thesis Outline

The thesis is organized as follows. In chapter 2, previous work on various TCM schemes, such as Ungerboeck's TCM [31][32], multi-dimensional TCM [42][67] and large constraint length TCM [26][28], are first briefly presented. A relevant technique called shaping [4][5][36] is also reviewed. We then introduce LDPC codes and turbo codes which can achieve the near-Shannon limit performance and, therefore, have drawn much attention from coding theory researchers re-

cently. As a natural extension of binary turbo codes, several turbo trellis coded modulation (TTCM) schemes have been developed for bandwidth-limited communication systems, and the remarkable error performance close to the Shannon capacity limit has been able to be achieved. From a different perspective, the iterative Viterbi algorithm (IVA) is developed for concatenated convolutional codes, and competitive performance to turbo codes is obtained [62][63][64]. This motivated the issue about how to apply the IVA for concatenated TCM addressed later in this thesis. The possible applications of concatenated TCM are also suggested. Finally, robust decoding algorithms for convolutional codes and turbo codes are reviewed.

In chapter 3, the encoding methods for the Ungerboeck 2-D TCM and Wei multi-dimensional TCM schemes are described first, and then we present the parity-concatenated trellis codes in which a simple even parity check code is serially concatenated with conventional 2-D or M-D trellis codes. For both packet and continuous transmissions, the corresponding single-parity-check and double-parity-check structures are designed, respectively. Packet transmission with long, medium and short block lengths are also considered. In addition, shaping techniques and multilevel codes are combined with the parity-concatenated trellis codes for further performance improvement. As an effective tool for code interpretation, graph representations of the parity-concatenated trellis codes are presented. In this thesis, we focus on the Tanner-Wiberg-Loeliger (TWL) graph. Based on the graph representations, the iterative decoding algorithms for parity-concatenated trellis codes can be derived.

In chapter 4, we present the iterative Viterbi algorithm for decoding the parity-concatenated 2-D trellis codes as well as M-D trellis codes. The new branch metric functions are derived, respectively. It can be seen that the only difference between the IVA and the standard VA is the calculation of branch metrics. Due to a nonlinear operation in the encoding procedure of M-D trellis codes, some extra efforts with negligible additional computation is made in the IVA for decoding the parity-concatenated M-D trellis codes. Some relevant issues on the IVA, such as

packet size selection, the manner of incorporating *extrinsic* information feedback, the value setting of scaling parameter used in the IVA, stop criteria in the IVA and a "pre-decision" method used in the IVA, are also discussed.

In chapter 5, firstly, the conventional iterative two-way algorithms (ITWAs) based on graph representations are presented for parity-concatenated trellis codes. Then we show the modified iterative two-way algorithms in which a normalization function is applied. We can see that the modified version is an effective solution for graphs with many small cycles. From the perspective of graph interpretation, the IVA discussed in chapter 4 can be viewed as a simplified version of the modified iterative min-sum algorithm. Based on the graph representations, five iterative decoding algorithms are then developed, and their computational complexity are compared. For the parity-concatenated trellis codes at a high spectral efficiency, error floors would be appeared at the region close to the Shannon capacity limit due to the parallel transition errors in trellis codes. The upper bound on the error floor is analytically determined, and an appropriate multilevel BCH code is then selected to dramatically bring down the error floor to a very low level.

In chapter 6, extensive simulation results for parity-concatenated trellis codes are reported. Firstly, the scaling parameter used in the IVA is determined for a specific example. Then we present and compare the performance of the IVA for two types of parity-concatenated 2-D trellis codes with partial and full protection, respectively. The performance of the trellis codes with trellis shaping and multilevel codes are also given. We then compare the conventional ITWA and modified ITWA. Finally, several iterative decoding algorithms are compared according to the error performance and computational complexity for parity-concatenated 2-D trellis codes and 4-D trellis codes. Both continuous transmission and packet transmission with long, medium and short block lengths are considered.

In chapter 7, a robust Viterbi algorithm and robust iterative decoding algorithms emphasizing the IVA are studied for conventional trellis codes and parity-concatenated trellis codes respectively, under an uncertain noisy environment. Using the *minimax* technique, a simple but effective robust decoder is devised.

The problem of selecting the scaling parameter is also studied for the robust IVA.

Finally, in chapter 8, a brief summary of the accomplished work, with an emphasis on the contributions to the area of the iterative decoding algorithms for parity-concatenated trellis codes, is presented. Also, promising areas for further research are discussed.

1.3 Contributions

Here we summarize the key contributions of this thesis as follows:

(1) Parity-concatenated TCM schemes with single-parity-check and double-parity-check structures are investigated for various packet length(s), and superior performance can be achieved with both structures; (chapter 3)

(2) The TWL graph representations for parity-concatenated trellis codes are given. They provide an effective way for investigating the graph-based decoding algorithms; (chapter 3)

(3) The IVA is extended for decoding the parity-concatenated trellis codes, which thus can be applied to many current systems without or with very little modification; (chapter 4)

(4) Modified min-sum and sum-product algorithms are studied for the TWL graphs with many small cycles. It can be seen that significant performance improvement is made over conventional min-sum and sum-product algorithms; (chapter 5)

(5) Five iterative decoding algorithms (IDAs) are developed for the parity-concatenated TCM, and their complexity and performance are compared; (chapter 5)

(6) The IDAs are extended for decoding the parity-concatenated multi-dimensional TCM schemes, and the significant performance is achieved; (chapters 4 and 5)

(7) It is shown how to apply the trellis shaping technique to the parity-concatenated TCM system; (chapter 3)

(8) Multilevel coding and multistage decoding are introduced for parity-concatenated TCM to reduce the error floor level; (chapters 3 and 4)

(9) The upper bound on the error floor is analytically determined; (chapter 5)

(10) The parity-concatenated TCM schemes are extended for parallel continuous transmission; (chapter 3)

(11) Robust IDAs are studied for parity-concatenated trellis codes under an uncertain noise environment; (chapter 6)

(12) The possible applications of parity-concatenated TCM on current standard systems are suggested. (chapter 2)

1.4 Paper List

(1) Qi Wang, Lei Wei and Rodney A. Kennedy, "Iterative Viterbi Decoding, Trellis Shaping and Multilevel Structure for High-Rate Concatenated TCM", submitted to *IEEE Trans. on Commun.*.

(2) Qi Wang and Lei Wei, "Graph-Based Iterative Decoding Algorithms for Parity-Concatenated Trellis Codes", submitted to *IEEE Trans. on Inform. Theory.*

(3) Qi Wang, Lei Wei and Rodney A. Kennedy, "Near Optimal Decoding for Trellis-Coded Modulation Using the BIVA and Trellis Shaping", *Applied Algebra, Algebraic Algorithms and Error-correcting codes - 13th International Symposium, AAECC-13*, pp. 191-200, Honolulu, Hawaii, U.S.A., Nov. 1999.

(4) Qi Wang and Lei Wei, "Iterative Viterbi Algorithm for Concatenated Multi-dimensional TCM", accepted by *IEEE Symposium on Information Theory (ISIT)'2000*, Italy.

(5) Qi Wang and Lei Wei, "Iterative Decoding Algorithms for Parity Concatenated TCM", accepted by *International Symposium on Information Theory and Its Applications (ISITA'2000)*, Honolulu, Hawaii, U.S.A., Nov. 2000.

(6) Qi Wang and Lei Wei, "Robust Decoding Algorithm for Trellis Coded Modulation", in preparation for *IEEE Trans. on Commun.*.

Chapter 2

Literature Review

2.1 Trellis Coded Modulation and Shaping Techniques

Trellis coded modulation (TCM) has been widely used as a combined coding and modulation technique for digital transmission over band-limited channel. In [31][32], Ungerboeck has shown that significant coding gains can be achieved using trellis coded modulation with the Viterbi decoding over uncoded modulation without increasing the transmitted power or sacrificing bandwidth efficiency on a band-limited channel. This makes TCM a popular choice for digital transmission over band-limited channels.

Since the 1980's, more powerful multi-dimensional (M-D) trellis codes have been discovered and introduced into the telecommunications industry due to a number of potential advantages, such as more coding gain, more robustness to signal-dependent impairment, smaller constellation expansion, etc. As the most attractive selection, 4-D Wei TCM schemes [67] have been widely accepted due to the modest tradeoff between complexity and coding gain improvement. Nowadays they have been used in many applications such as the high-rate voice band modem [115] and the ADSL modem [117].

From a different perspective for achieving more coding gains, Wang and

Costello constructed and searched out large constraint length two-dimensional and multi-dimensional trellis codes [26][27][28]. With the sequential decoding algorithm, the cutoff rate bound at a bit error rate (BER) of 10^{-5}-10^{-6} could be achieved with the reasonable computational complexity.

In the late 1980's, constellation shaping was recognized as a separable part of the high-SNR coding problem [36][40]. In [36][5][4], it has been shown that shaping gain can be achieved by using nonuniform, Gaussian-like signalling. An efficient shaping algorithm called shell mapping has been proposed by Khandani and Kabal [5] and several others, and it has been used in V.34 modems. It is also possible to use block shaping codes to achieve non-equiprobable signalling [4]. Another approach, called trellis shaping, was proposed by Forney [36]. It was shown that a simple 4-state shaping code can achieve about 1.0 dB shaping gain, which is about 2/3 of the full 1.53 dB ultimate shaping gain. This leads to fundamental questions: (1) If a shaping scheme is used along with parity-concatenated trellis codes, will the full shaping gain still be achieved? (2) Will iterative decoding algorithms with shaping decoding operate reliably at signal-to-noise ratio (SNR) close to the Shannon limit? In this thesis, we try to answer these questions.

2.2 Low-Density Parity-Check Codes

Low-density parity-check (LDPC) codes, proposed by Gallager [86][87], are now seen as the grandfather of all the graph codes and decoding algorithms [41]. Due to the limitation of computational power at the time, Gallager's excellent work was neglected until being re-discovered by MacKay in 1997 [18][16].

The major attractiveness of the LDPC is its excellent performance built upon a simple structure. The original Gallager's LDPC codes are defined in terms of a very sparse random parity check matrix with uniform weight per column and per row [86][87]. These codes are asymptotically good, and can be practically decoded. Gallager in his thesis provided two decoding algorithms, one simpler ver-

sion which flip-flops bits until all the parity checks can be satisfied, the other one is flooding version which is a certain variation of the sum-product algorithm [25]. Recently, it was shown that the belief propagation (BP) algorithm [51] provides a powerful tool for iterative decoding of LDPC codes, by noting that the original Gallager's iterative probabilistic decoding of LDPC codes is a particular BP-based decoding approach [24][18][16][91]. Results presented in [18] showed LDPC codes have near-Shannon limit performance when decoded using the BP algorithm.

To further enhance the performance of Gallager's binary LDPC codes, some variations have been proposed, such as LDPC codes over finite fields $GF(q)(q > 2)$ [73] and irregular Gallager codes [19]. These codes can yield very good performance on the binary symmetric channel (BSC) as well as on the additive white Gaussian noise (AWGN) channel.

2.3 Turbo Codes

Turbo codes, first presented to the coding community in 1993 by Berrou, Glavieux, and Thitimajshima [15], represent the most important breakthrough in coding theory since Ungerboeck introduced trellis codes in 1982 [31]. Whereas Ungerboeck's work eventually led to coded modulation schemes capable of operation near capacity on band-limited channels [74], the original turbo codes offer near-capacity performance for deep space and satellite channels. Many of the structural properties of turbo codes have now been put on a firm theoretical footing [97][20][57][102][69][91], and several innovative variations on the turbo theme have appeared in [6][96][102][69][89].

A turbo encoder is showed in Fig. 2-1, which is formed by parallel concatenation of two recursive systematic convolutional (RSC) encoders separated by a pseudo-random interleaver [90]. Unlike the classical interleaver (e.g., block or convolutional interleaver), which rearranges the bits in some systematic fashion, the interleaver in turbo coding is a pseudo-random block scrambler defined by a permutation of N elements with no repetitions. The first reason for using an

interleaver in turbo coding is to generate a concatenated code with large block length which leads to a large coding gain [7]. Secondly, it de-correlates the inputs to the two decoders so that an iterative suboptimum decoding algorithm based on information exchange between the two component decoders can be applied.

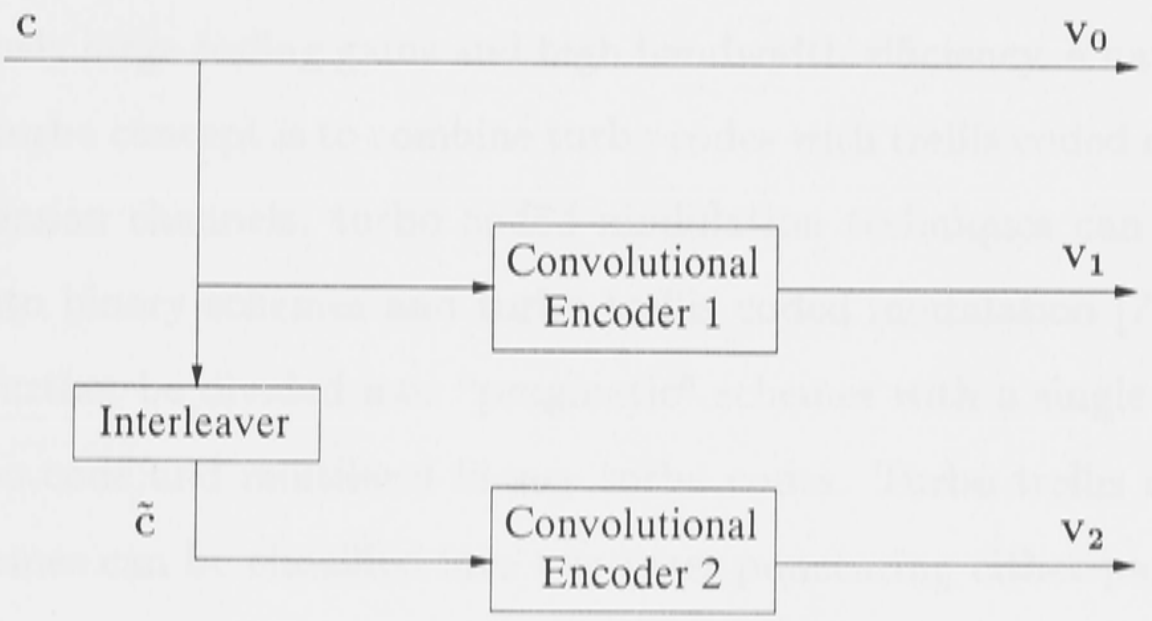


Figure 2-1: Turbo encoder

For a turbo code, a maximum-likelihood (ML) sequence decoder would be far too complex due to the presence of the interleaver. However, the suboptimum iterative decoding algorithm offers near-ML performance. The turbo decoder consists of two concatenated decoders of the component codes separated by the same interleaver. The component decoders are based on a *maximum a posteriori* (MAP) probability algorithm or a *soft output Viterbi algorithm* (SOVA) [56] generating a weighted soft estimate of the input sequence. The iterative process performs the information exchange between the two component decoders. By increasing the number of iterations in the turbo decoding, the bit error probability as low as 10^{-5}-10^{-7} can be achieved at a SNR close to the fundamental limits established by Shannon.

2.4 Turbo Trellis Coded Modulation

Turbo codes can achieve remarkable error performance at a low SNR close to the Shannon capacity limit. However, the powerful binary coding schemes are not suitable for bandwidth limited communication systems [40]. In order to achieve simultaneously large coding gains and high bandwidth efficiency, a natural extension of the turbo concept is to combine turbo codes with trellis coded modulation.

For Gaussian channels, turbo coded modulation techniques can be broadly classified into binary schemes and turbo trellis coded modulation [7]. The first group can further be divided into "pragmatic" schemes with a single component binary turbo code and multilevel binary turbo codes. Turbo trellis coded modulation schemes can be classified into two cases puncturing either parity symbol or information symbol.

A. Binary turbo coded modulation

In pragmatic turbo coded modulation design [95] a single binary turbo code of rate $1/n$ is used as the component code. The output of the turbo encoder is then simply mapped onto an M-ary modulator. Decoding is done by calculating the log-likelihood function for each encoded binary digit based on the received noisy symbol and the signal subsets in the signal constellation specified by each binary digit. The stream of the bit likelihood values is then passed to the binary turbo decoder which can be based either on MAP or soft output Viterbi algorithms (SOVA).

The pragmatic approach is simple, as only one turbo encoder and one turbo decoder are used. By modifying the puncturing function and modulation signal constellation, it is possible to obtain a large family of turbo coded modulation schemes. However, although this system utilizes a bandwidth efficient modulation scheme, the encoder and modulator are not designed cooperatively as in TCM systems.

In [107][108], multilevel turbo codes are constructed by using binary turbo codes as the component codes. The transmitter for an M-ary signal constellation

consists of $l = log_2^M$ parallel binary encoders as shown in Fig. 2-2.

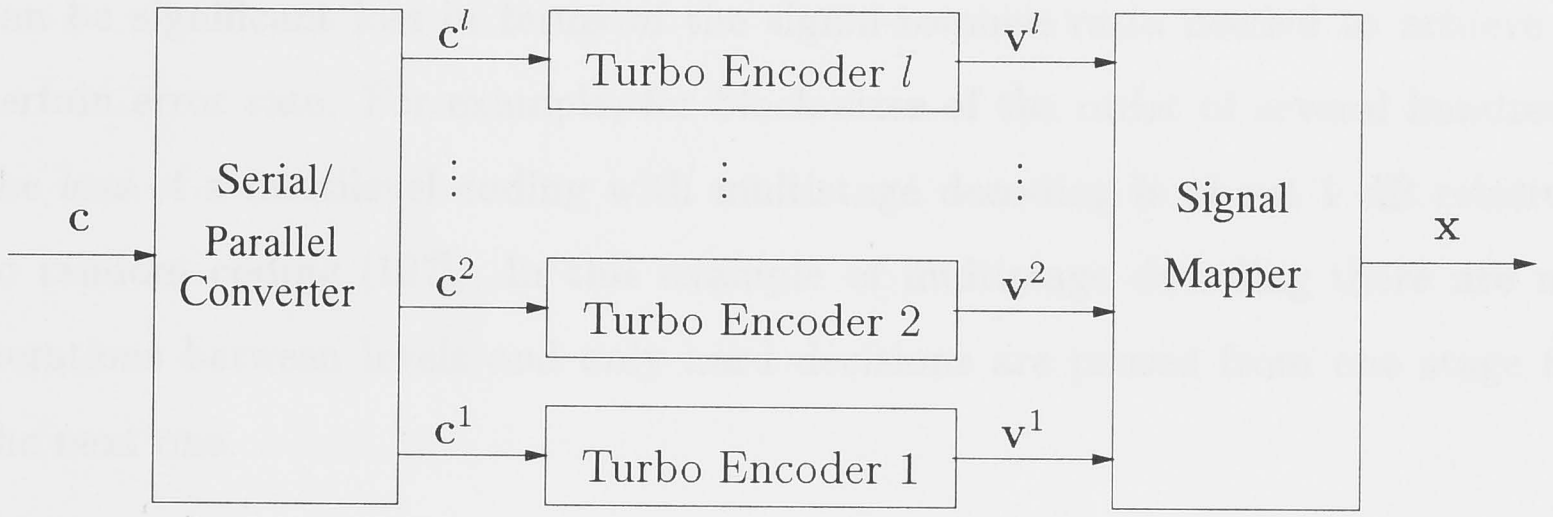


Figure 2-2: Multilevel turbo encoder

A message sequence is split into l blocks. Each message block $\mathbf{c}_i$ is then encoded by an individual binary turbo encoder. The output digits of the encoders form a binary label $(\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_l)$, which is mapped onto an M-ary signal constellation.

The maximum likelihood decoder operates on the overall code trellis. In general, however, this decoder is too complicated to implement. Alternatively, a suboptimum technique, called multistage decoding [47], can be used, resulting in the same asymptotic error performance as the maximum likelihood decoding.

An important issue in the code design is the choice of component codes and their code rates. Wachsmann and Huber [107][108] proposed a technique for selecting the component code rates. In this design, the component code rate at a particular modulation level, is chosen to be equal to the capacity of the equivalent binary input channel associated with that level. For infinite code lengths, in theory, as the overall channel capacity is equal to the sum of the channel capacities for all levels, this design results in error free coding. Turbo codes come close to the Shannon capacity limit and provide almost error free coding. Therefore, they are suitable candidates for components codes in a multilevel scheme. Another good property is that due to their good performance it can be assumed that there is negligible error propagation between the modulation levels [108]. This is an important conclusion which enables the use of multistage decoding, as it

asymptotically leads to the optimum results. However, for small block sizes, there can be significant loss in terms of the signal-to-noise ratio needed to achieve a certain error rate. For example, for block sizes of the order of several hundred, the loss of a multilevel coding with multistage decoding is about 1 dB relative to random coding [107]. In this example of multistage decoding there are no iterations between levels and only hard decisions are passed from one stage to the next one.

B. Turbo trellis coded modulation

In [80][81], a turbo trellis coded modulation (TTCM) system was presented in which two recursive Ungerboeck type trellis codes with rate $k/(k+1)$ are concatenated in parallel. Fig. 2-3 shows the encoder structure comprising of two recursive convolutional encoders linked by a symbol interleaver and followed by a signal mapper.

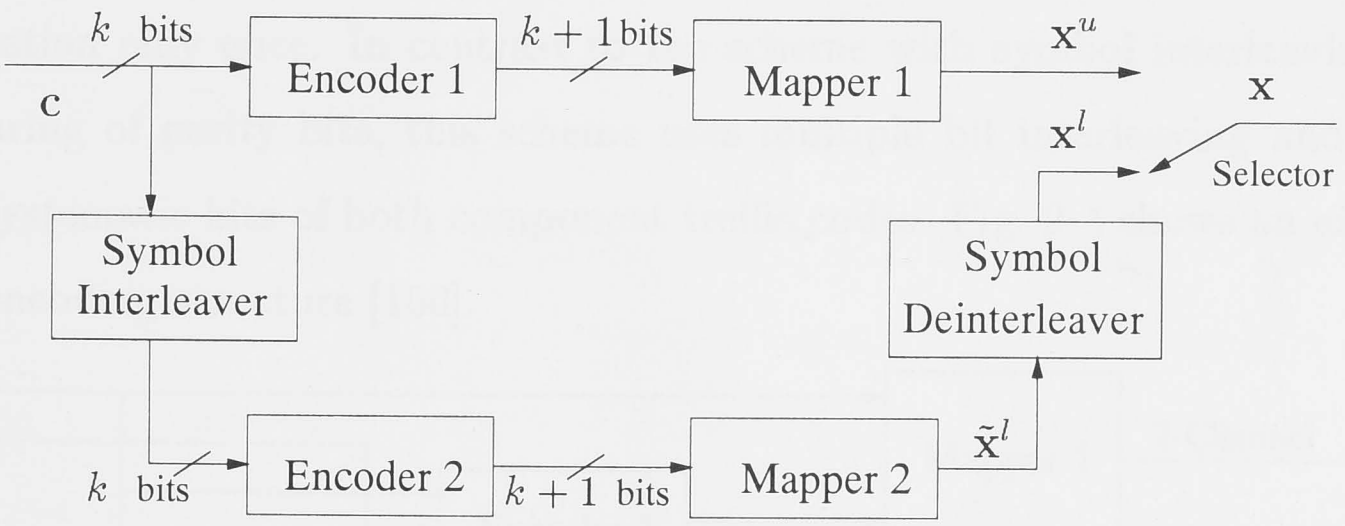


Figure 2-3: Turbo TCM encoder with parity symbol puncturing

It is noted that the interleaver is constrained to interleave symbols. That is, the ordering of k information bits arriving at the interleaver at a particular instant remains unchanged. For the component trellis code, some of the input bits may not be encoded. In practical implementations these inputs do not needed to be interleaved, but are directly used to select the final point in a signal subset. At the receiver the values of these bits are estimated by subset decoding [31].

The output of the second encoder is de-interleaved. This ensures that the k

information bits which determine the encoded $(k+1)$ binary digits of both the upper and lower encoder at a given time instant are identical. The selector then alternately connects the upper and lower encoder to the channel. Thus, the parity symbol is alternately chosen from the upper and lower encoder. Each information group appears in the transmitted sequence only once.

In the receiver, the log-MAP decoding or SOVA decoding algorithms are used to decode the turbo trellis codes [7][8][58]. The decoding process is very similar to the binary turbo codes except that the symbol probability is used as the extrinsic information rather that the bit probability.

As a different type of turbo TCM scheme, parallel concatenation of two recursive trellis codes with puncturing of systematic bits was proposed by Benedetto, Divsalar, Montorsi and Pollara [100]. The basic idea of the scheme is to puncture the output symbols of each trellis encoder and select the puncturing pattern such that the output symbols of the parallel concatenated code contains the input information only once. In contrast to the scheme with symbol interleaving and puncturing of parity bits, this scheme uses multiple bit interleaving and punctures systematic bits of both component trellis codes. Fig. 2-4 shows an example of its encoding structure [100].

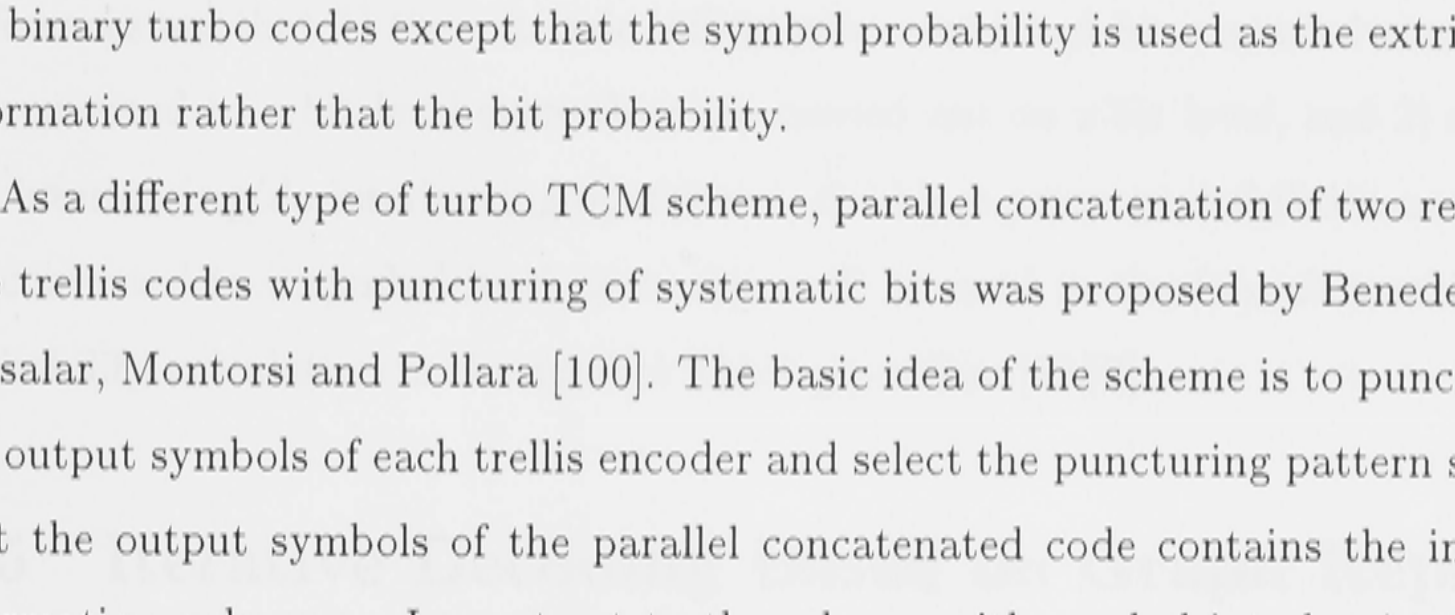


Figure 2-4: Turbo TCM encoder with systematic symbol puncturing

Here each encoder forms a modulation symbol from the parity stream produced by the encoder and a subset of the systematic information. These symbols

are then sent, as depicted in Fig. 2-4, using in-phase and quadrature modulation to form a composite symbol or by serializing the symbols for transmission, e.g. sequentially sending two trellis symbols. Since two modulated signals are generated in every encoding interval, this scheme is equivalent to a multi-dimensional M-PSK or M-QAM trellis code.

Decoding this type of turbo TCM is a straightforward application of the iterative symbol-by-symbol MAP algorithm for the binary turbo codes. The only differences are that 1) the extrinsic information computed for a symbol needs to be converted to a bit level since they are carried out on a bit level, and 2) after the interleaving/de-interleaving operations, the bit *a priori* probabilities need to be converted to a symbol level since they will be used in the branch transition probability calculation in the symbol MAP algorithm [58][7].

2.5 Iterative Decoding Based on Graph Representations

One of the key techniques in turbo decoding, also a main reason for the success of turbo codes, is the adoption of iterative decoding, which actually has been invented long before the discovery of the turbo codes but unfortunately ignored for 40 years. Nowadays, it has been seen that the iterative decoding techniques have widespread applications not only in error control, but also in detection, interference suppression and equalisation [7].

On the other hand, some leading researchers have made significant progress in generalising the iterative decoding algorithms. It is now well-known as the iterative decoding on graphs. In [88], Tanner generalized Gallager's LDPC codes to codes on general bipartite graphs, known as *Tanner graphs*, with two types of nodes representing symbols and parity checks, respectively. In a Tanner graph, each symbol is represented by a symbol node, and each parity check is represented by a check node. Each check node is connected by an edge to the symbols that it checks. A Tanner graph is thus a bipartite graph in which symbol nodes are

connected only to check nodes and vice versa.

In [77][78], Wiberg *et al.* made a key extension of Tanner graphs to include a third type of nodes, representing invisible "state" nodes, thus establishing a bridge to the trellis diagrams. Such graphs are known now as *Tanner-Wiberg-Loeliger (TWL) graphs* [38]. Using Tanner graphs and/or TWL graphs, all the iterative decoding algorithms can be classified as one of two types of generic algorithms: *min-sum* and *sum-product* algorithms. From this point of view, the famous Viterbi algorithm and the backward-forward algorithm (BFA) can be seen as special cases of the min-sum and sum-product algorithms that arise when the underlying graph is trellis. In [38] Forney gave a description of the history of various "two-way" algorithms (TWA) and their connections with coding theory.

In [16][18], MacKay and Neal firstly connect Gallager's LDPC codes with Pearl's "Belief Propagation" algorithm in a Bayesian network [51], which has been developed in the past decade in the fields of expert systems and artificial intelligence. Extensive simulation results of MacKay and Neal show that LDPC codes are competitive with turbo codes, and that they can reach channel capacity [16]. In [91] McEliece *et al.* have also independently observed that turbo decoding is an instance of "belief" propagation. They provide a description of Pearl's algorithm, and make explicit the connection to the basic turbo decoding algorithm described in [15].

After reviewing a variety of graphical models, Kschischang and Frey presented a unified graphical model, known as *factor graph*, for describing compound codes and deriving iterative decoding algorithms [24]. A factor graph is a bipartite graph that expresses how a "global" function of many variables factors into a product of "local" functions [25]. Factor graphs subsume many other graphical models including Bayesian networks, Markov random fields, and Tanner graphs. From this general framework, a wide variety of iterative decoders can be developed for parallelly and serially concatenated coding systems [15][102], product codes [59][57], and LDPC codes.

Tanner has proved that the min-sum and sum-product algorithms can con-

verge on finite cycle-free graphs [88], and Pearl's BP (or probability propagation) algorithm also only gives exact answers when there are no loops in *Bayesian networks* [51]. However, it has been discovered that the most powerful codes known, namely turbo codes and LDPC codes, are naturally represented by graph with cycles. Empirically, iterative decoding algorithms using probability propagation often work very well in graphs with cycles, provided that the cycles are long enough that cyclical dependencies die out as they propagate around a cycle [38]. Theoretically, however, it is still not well understood why these decoders work so well. Frey, Mackay [11] and Weiss [113] have made efforts toward this direction.

In this thesis, we will investigate the parity-concatenated trellis codes in which lots of small cycles exist. As the effective methods dealing with small cycles, the iterative Viterbi algorithm (IVA) and modified iterative two-way algorithm (ITWA) are developed.

2.6 Iterative Viterbi Algorithm for Forney's Concatenated Systems

During the last 7 years, much work has been done in the area of iterative decoding for concatenated compound codes, such as turbo codes and LDPC codes. Until now, there are two key results on iterative decoding of those compound codes:

(1). The sum-product type of algorithms (i.e., soft-in/soft-out decoders including the Gallager's APP [87], BCJR [70] and SOVA [56]) are favoured over the min-sum type of algorithms (including bi-directional VA and the standard VA).

(2). Random interleavers with large minimum girth are favoured over block interleavers, except block-block component codes in [57] in which both random and block interleavers result in a similar error performance. Here the minimum girth is defined as the length of minimum cycle in the TWL graph.

Obviously, these results rule out the popular Forney's conventional concatenated code [35] and the well known Viterbi algorithm. Inspired by Jelinek and Cocke's work [22][23] in 1970s, Cabral, Costello and Chevillat [45] noted significant similarity between the turbo decoding method and the bootstrap iterative decoding method. Bootstrap decoding [23][62] is a method which imposes algebraic constraints on streams of convolutionally encoded information sequences so as to gather *extrinsic* information from other streams when one stream is decoded. In [62] Wei extended the results of [23][45] to near optimally decode the bootstrap case using long convolutional codes. One of the simplified bootstrap algorithms, which only uses the Viterbi algorithm, was given in [62] and named as BIVA in [63], now called iterative Viterbi algorithm (IVA) [64]. It is showed that the IVA can achieve near the Shannon limit performance for a modified Forney's concatenated convolutional code in which the powerful block code is replaced by a simple parity check code [62][63][64][61]. How to apply the IVA in concatenated TCM is the open problem which will be addressed in this thesis.

2.7 Robust Decoding Algorithms

It has been known that turbo codes can achieve the highest performance for memoryless channels, provided an iterative decoding algorithm, such as APP, MAP or BCJR algorithm, is applied. However, unlike traditional minimum distance decoders which typically implement the Viterbi algorithm, turbo decoders require knowledge of the channel SNR and the noise distribution (probability density function (PDF)). In [75][106][112], the authors investigated the sensitivity of the turbo decoder to SNR mismatch, and proposed several methods of estimating the variance which is necessary for turbo decoding.

Different from those variance estimation techniques, Wei *et al.* presented a probabilistic minimax decoding algorithm [66][114] in which the minimax concept is used to minimize the worst possible error performance over a family of possible channel noise PDFs. It is shown that the minimax decoder is robust to the

mismatch between the channel noise model and the noise model used for receiver design. The robust VA and robust two-way APP decoders are devised in [66] and always outperform the worst mismatched standard decoders and generally perform very close to the optimal matched decoder in various mismatched noise environment.

In chapter 7 of this thesis, the minimax concept is extent for trellis codes and parity-concatenated trellis codes. We will see some interesting results when the minimax decoders are applied in decoding the parity-concatenated trellis codes.

2.8 Possible Applications of Parity-Concatenated TCM

Parity-concatenated TCM can be potentially applied in the same application domains where TCM is used. TCM has been widely used in the areas of digital communications. Some typical applications are as follows:

(1) V.34 high-rate voice-band modem [115];

V.34 modem employs the 4-D 16-/32-/64-state Wei trellis codes for different rate;

(2) Cable modem [116];

A concatenated coding method with outer Reed-Solomon (RS) coding and inner trellis coded modulation is applied in the cable modem standard.

(3) ADSL modem [117].

As a new high-speed digital access line technology, ADSL modem has become a main solution for the high rate Internet access in the world. In the ADSL system, in terms of the characteristic, the channel is divided as the N (generally $N = 256$) sub-channels. Various information symbols are allocated into each sub-channel to maximize the transmission capability. In each sub-channel, the 4-D Wei trellis codes can probably be applied with

the DMT (Discrete Multi-Tone) modulation, depending on the channel situation.

Using the iterative Viterbi decoding algorithm, we can employ the parity-concatenated trellis codes in some applications, such as ADSL modem and V.34 modem, with very few changes in current systems. To cable modem, the only modification is to replace the powerful outer RS code by a simple parity code, and then the IVA can be applied.

2.9 Discussion

In this chapter, previous work on various TCM schemes and relevant shaping techniques has been briefly presented. Compound codes such as LDPC codes and turbo codes were introduced. As combinations of turbo codes with TCM, several turbo trellis coded modulation schemes have been described. Then we mentioned the IVA which will be presented in this thesis for decoding the parity-concatenated trellis codes. Finally, robust decoding algorithms for convolutional codes and turbo codes were reviewed.

Chapter 3

Encoder Structures for Parity-Concatenated Trellis Codes

3.1 Introduction

Concatenation is a specific method of constructing long codes from shorter codes [35]. In this chapter, we will present the encoder structures for parity-concatenated trellis codes in which a simple even parity check code is serially concatenated with trellis codes including the 2-D trellis codes and multi-dimensional trellis codes. To be suitable in packet transmission with short, medium and long block lengths, the corresponding single- and double-parity-check encoding structures are built up respectively, and their graph representations based on TWL graph are presented. In addition, since shaping techniques have been used with the conventional TCM schemes, in this chapter we also consider shaping with the parity-concatenated trellis codes. First, let us introduce conventional trellis coded modulation.

3.2 Trellis Coded Modulation

3.2.1 Ungerboeck's trellis coded modulation

Trellis coded modulation has been widely used in the industry since Ungerboeck's papers published in the 1980's [31][32]. A general TCM scheme is shown in Fig. 3-1.

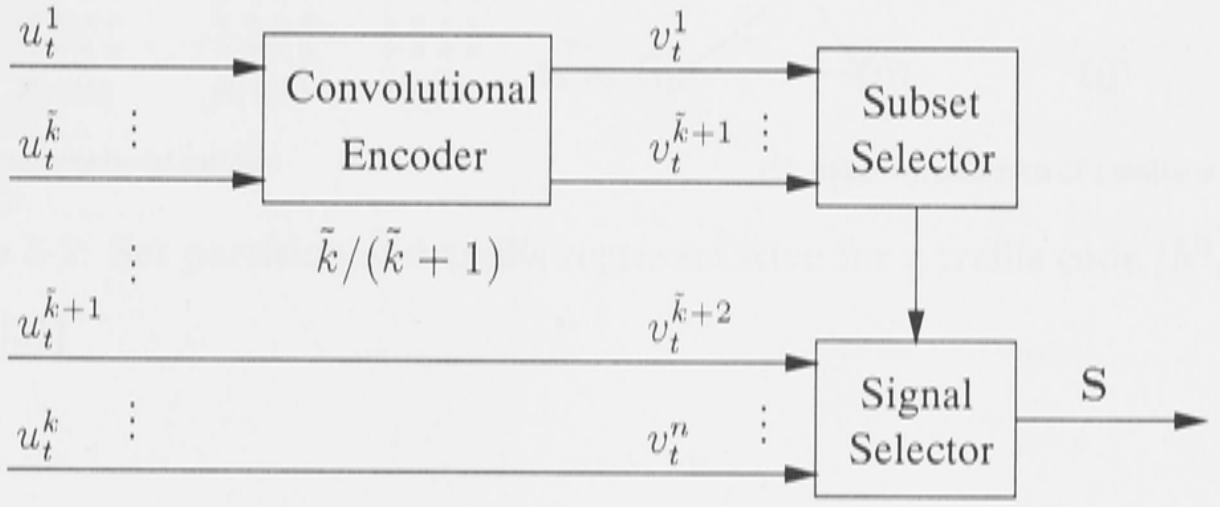
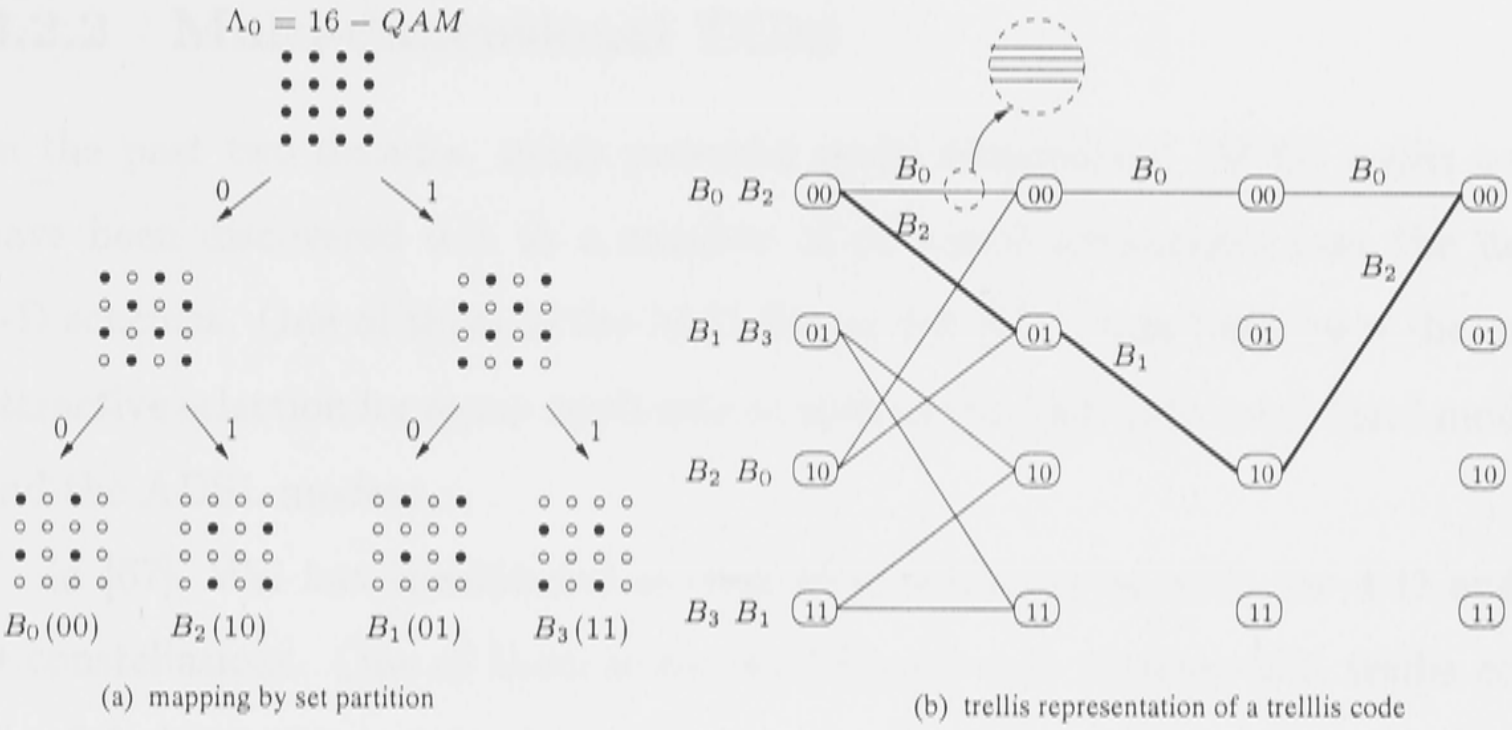


Figure 3-1: Schematic representation of a typical trellis code

In the trellis encoder, the k information bits $u_t^1 u_t^2 \cdots u_t^k$ of each symbol U_t are divided into two parts at time t. One part $u_t^1 \cdots u_t^{\tilde{k}}$ is encoded by a rate $\tilde{k}/(\tilde{k}+1)$ convolutional encoder whose output is used to select one subset from the whole constellation; the other part $u_t^{\tilde{k}+1} \cdots u_t^k$ comprises uncoded bits which are used to determine one signal point within that selected subset. It is noted that the convolutional codes can be viewed as a special case of trellis codes in which $\tilde{k} = k$, namely, there are no uncoded bits.

In TCM, the key concept is *mapping by set partitioning* [31][32] which conveys the idea of combined design of coding and modulation. Let the minimum Euclidean distance between pairs of signal points in the constellation Λ_0 be denoted by $d_{min,(\Lambda_0)}$. Through the *set partitioning*, a M-ary constellation Λ_0 is successively partitioned into 2, 4, 8, $\cdots$ subsets with size $M/2$, $M/4$, $M/8$, $\cdots$, having progressively larger minimum distances $d_{min,(\Lambda_1)}$, $d_{min,(\Lambda_2)}$, $d_{min,(\Lambda_3)}$, $\cdots$. Here a *partition* Λ_i/Λ_{i+1} of Λ_i is defined as a collection of Λ_{i+1} and its non-overlapping cosubsets. Fig. 3-2 (a) displays a set partition of a 16-QAM constellation, in



(a) mapping by set partition

(b) trellis representation of a trelllis code

Figure 3-2: Set partition and trellis representation for a trellis code $(h^0, h^1) = (2, 5)$ [31]

which four subsets B_0, B_1, B_2 and B_3 are obtained. The encoding process of trellis codes can be represented by the trellis diagram. Fig. 3-2 (b) shows the trellis representation of a 4-state Ungerboeck code $(h^0, h^1) = (2, 5)$ [31] with 16-QAM. The thicker line in Fig. 3-2 (b) represents an error event. Note that each transition, from the current state to the next state, actually comprises four parallel transitions resulting from two uncoded bits.

To decode the trellis codes, the Viterbi algorithm is normally applied. This is done by calculating the "branch metric" of each branch, and then looking for the path whose total metric, called "state metric", is minimum. This path often corresponds to the transmitted signal sequence.

When the TCM is employed for transmission over the AWGN channel, and the SNR is large enough, the BER performance of TCM is mainly determined by the minimum squared Euclidean distance d^2_{free} which is the minimum value of the squared parallel transition distance $d^2_{parallel}$ and the coded minimum squared Euclidean distance $d^2_{free,c}$, i.e., $d^2_{free} = \min(d^2_{parallel}, d^2_{free,c})$. If $d^2_{free,c} \leq d^2_{parallel}$, we can say that the bit errors caused by parallel transition errors can be ignored at high SNRs, and hence the BER is dominated by $d^2_{free,c}$ [21]. It can be seen that most of the Ungerboeck codes belong to this category.

3.2.2 Multi-dimensional TCM

In the past two decades, many powerful multi-dimensional (M-D) trellis codes have been discovered due to a number of potential advantages over the usual 2-D schemes. One of them is the M-D Wei codes [67] which have been the most attractive selection for many applications such as the high-rate voice band modem and the ADSL modem.

In [67], Wei has constructed several M-D trellis codes with the 4-D and 8-D constellations. One of them is the well known 4-D 16-state Wei trellis code. Fig. 3-3 shows the encoder structure for the 4-D 16-state Wei trellis code [67].

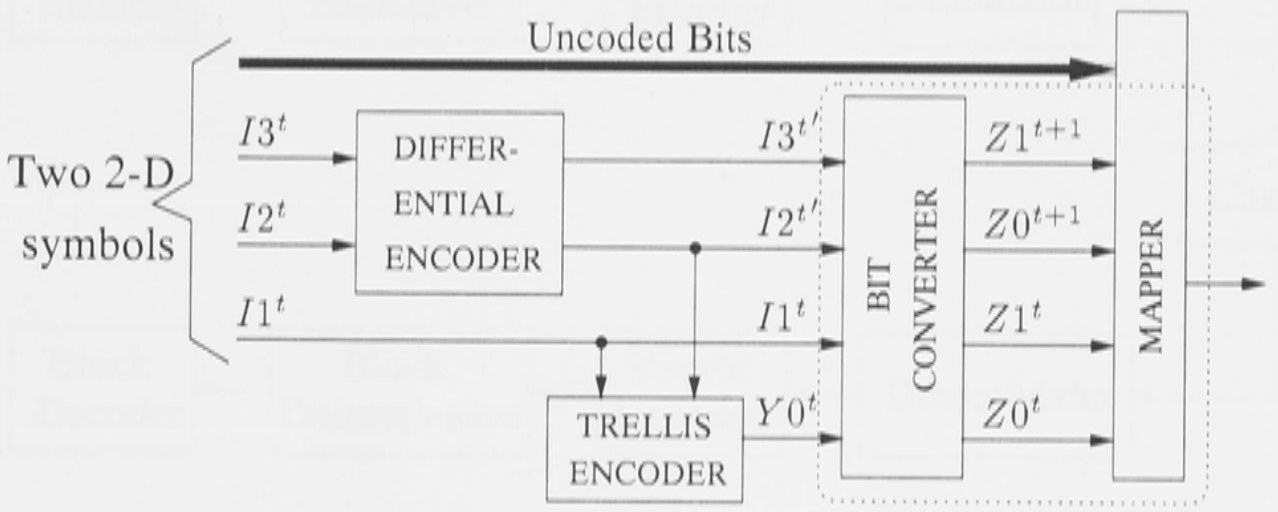


Figure 3-3: Encoder structure for the 4-D 16-state Wei trellis code

During the encoding process, two 2-D symbols are simultaneously applied to a 4-D trellis encoder at two successive time intervals t and $t+1$. Three bits are encoded via a trellis and differential encoder, while the rest of the bits from the two 2-D symbols remain uncoded. (In the Wei code design, three of those uncoded bits are encoded via a 4-D block encoder which actually implements the shell mapping presented in [5].) Four output bits $Y0^t$, $I1^t$, $I2^{t'}$ and $I3^{t'}$ are then *converted* by a bit converter to produce two groups of coded bits which correspond two 2-D sub-constellations in one 4-D constellation. Finally, combining these with the uncoded bits, we can obtain a 4-D trellis code comprising two 2-D trellis codes. In the receiver, the VA is used to decode the received 4-D signals. The only difference with decoding the 2-D trellis codes is the calculation of branch

metrics for 4-D subsets. The detailed decoding procedures can be found in [67].

3.2.3 Forney's concatenated code

As a popular choice in digital communications, Forney's concatenated code consists of two separate codes which are combined to form a large code [35]. Generally, Forney's concatenated coding system includes a moderate-strength trellis inner encoder, a powerful algebraic block outer encoder and a conventional block table-like interleaver, as illustrated in Fig. 3-4.

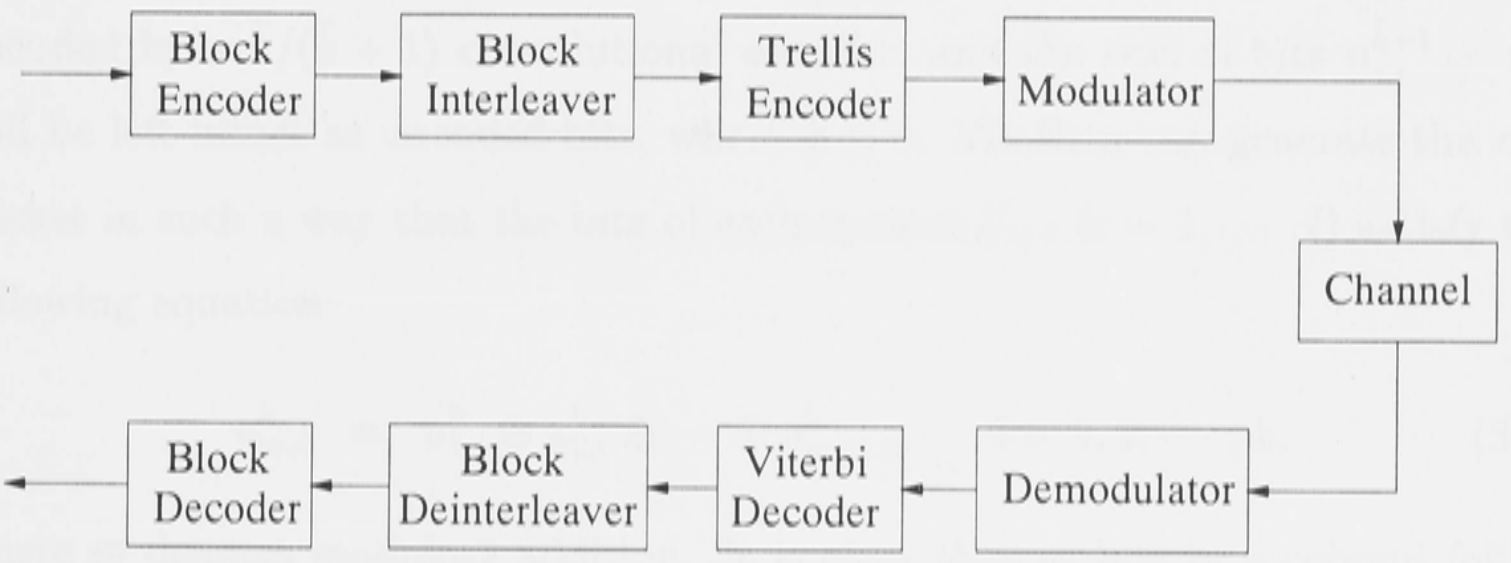


Figure 3-4: Forney's Concatenated Coding System

In the decoder, firstly a maximum-likelihood (ML) or near-ML decoding algorithms is used to achieve a moderate error rate like 10^{-2}-10^{-3} at a code rate as close to capacity as possible, and then a block decoder is applied to drive the error rate down to as low an error rate as may be desired [40]. With such a "separated" decoding scheme, it was shown in [35] that the error rate could be made to decrease exponentially with block length at any rate less than capacity, while the decoding complexity increases only polynomially.

Next we will present a novel serial concatenation scheme involving a simple parity check outer code, block interleaving, and a TCM inner code. We will see that such coding scheme can be viewed as an example of Forney's concatenated coding system.

3.3 Encoder Structures for Parity-Concatenated TCM

3.3.1 Encoding parity-concatenated 2-D trellis codes

In the encoding process of parity-concatenated TCM, a block of packets can be formed by organising $(m-1)\times l$ information symbols $U_{j,i}$ $(j = 1, 2, \cdots, m-1;\ i = 1, 2, \cdots, l)$ into a block of $(m-1)$ rows and l columns, as illustrated in Fig. 3-5. Each symbol $U_{j,i}$ includes k bits $u^1_{j,i}u^2_{j,i}\cdots u^k_{j,i}$ in which $\tilde{k}$ bits $u^1_{j,i}\cdots u^{\tilde{k}}_{j,i}$ will be encoded by a $\tilde{k}/(\tilde{k}+1)$ convolutional encoder, and the rest of bits $u^{\tilde{k}+1}_{j,i}\cdots u^k_{j,i}$ will be left intact as uncoded bits, where $\tilde{k} \leq k$. We then can generate the m^{th} packet in such a way that the bits of each symbol $U_{m,i}$ $(i = 1, \cdots, l)$ satisfy the following equation:

$$u^b_{m,i} = u^b_{1,i} \oplus u^b_{2,i} \oplus \cdots \oplus u^b_{m-1,i}, \qquad b = 1, 2, \cdots, k, \tag{3.1}$$

where $\oplus$ denotes modulo-2 addition. It is clear that m bits in a column follow the parity-check (PC) constraint, and hence m symbols in a column also follow the parity-check constraint (and therefore the symbols $U_{m,i}$ are referred to as the *parity symbols*), as shown in Fig. 3-5. The m^{th} packet is then called the *parity packet*.

Since each bit in a symbol is related to a parity-check constraint, we refer to this kind of code as a *full-parity protected* code. However, if only coded bits of m symbols in a column satisfy the parity-check constraint, i.e,

$$u^b_{1,i} \oplus u^b_{2,i} \oplus \cdots \oplus u^b_{m-1,i} \oplus u^b_{m,i} = 0, \qquad b = 1, 2, \cdots, \tilde{k}, \tag{3.2}$$

then we refer to this kind of code as a *partial-parity protected* code, since only part of symbols are protected by the parity bits.

Next, as usual, we will encode each packet with l symbols into codewords of length l through the same S-state trellis encoder. It was shown in [62] that the free distance of such a block code is doubled relative to the free distance of the original code. Let $V_{j,i}$ denote the output symbol corresponding to the input

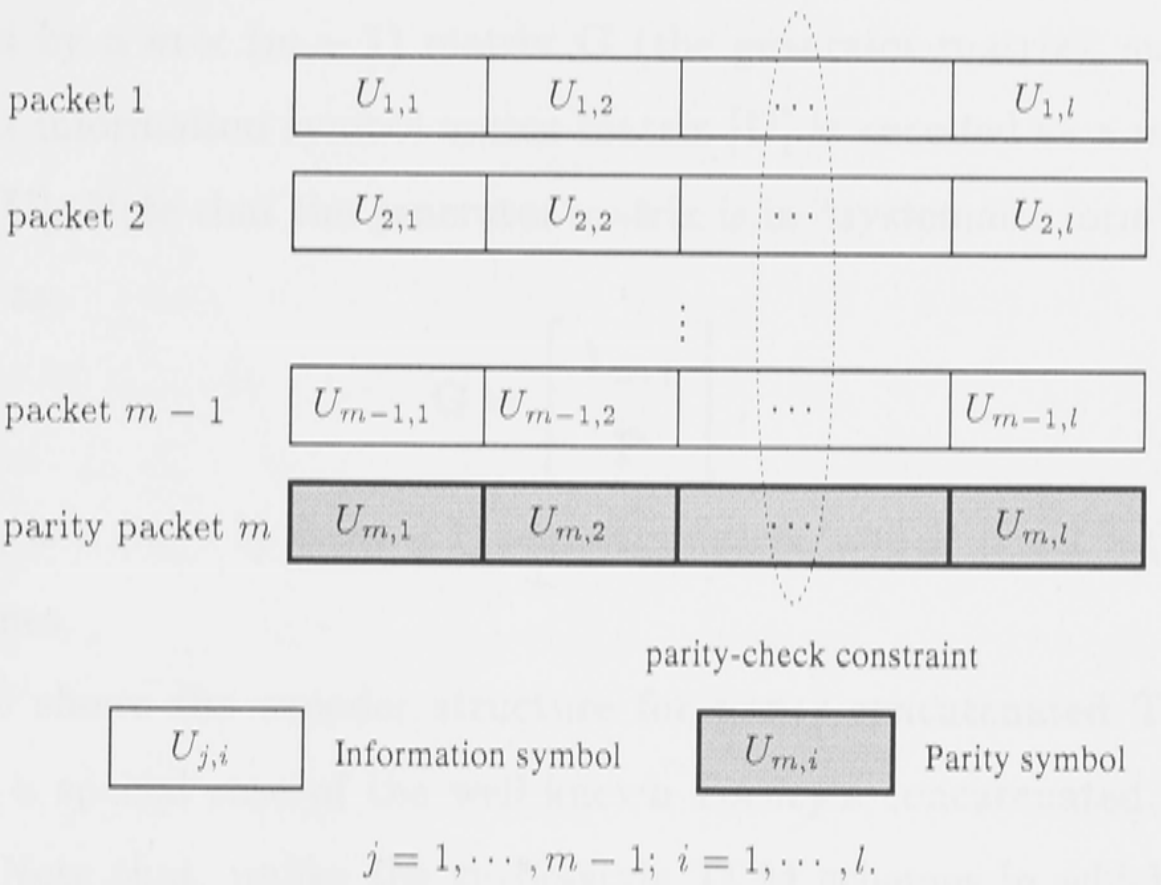


Figure 3-5: Parity-check structure for one block symbols

symbol $U_{j,i}$, and $V_{j,i}$ consist of n bits $v_{j,i}^1 \cdots v_{j,i}^{\tilde{k}+1} v_{j,i}^{\tilde{k}+2} \cdots v_{j,i}^n$, in which the $(\tilde{k}+1)$ bits $v_{j,i}^1 \cdots v_{j,i}^{\tilde{k}+1}$ are the output from the convolutional encoder with rate $\tilde{k}/(\tilde{k}+1)$, and the remaining bits $v_{j,i}^{\tilde{k}+2} \cdots v_{j,i}^n$ are uncoded. Lastly, the output symbol $V_{j,i}$ is mapped (according to Ungerboeck's set-partition rule) into a signal constellation to produce a modulated signal.

Due to the linearity of convolutional encoding[1], for the *full-parity protected* code, we then have

$$v_{1,i}^b \oplus v_{2,i}^b \oplus \cdots \oplus v_{m,i}^b = 0, \qquad b = 1, 2, \cdots, n. \tag{3.3}$$

For the *partial-parity protected* code, we have

$$v_{1,i}^b \oplus v_{2,i}^b \oplus \cdots \oplus v_{m,i}^b = 0, \qquad b = 1, 2, \cdots, \tilde{k}+1. \tag{3.4}$$

Hence, all m packets are in principle decodable by applying the VA to each packet. Clearly, the encoding process for parity-concatenated trellis codes can be

[1] All the convolutional (or trellis) encoders considered in this thesis are assumed to be linear which guarantee that the parity-check constraints among the symbols of packets still exist for coded symbols after the encoding process.

represented by a $m \times (m-1)$ matrix $\mathbf{G}$ (the generator matrix), such that the $(m-1) \times l$ information symbol vector matrix $[\mathbf{U}]$ is encoded as a $m \times l$ matrix $[\mathbf{V}] = \mathbf{G} \cdot [\mathbf{U}]$. Note that the generator matrix is in "systematic form" and it can be written as

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{m-1} \\ \mathbf{P} \end{bmatrix},$$

where $\mathbf{I}_{m-1}$ is a $(m-1) \times (m-1)$ identity matrix, and $\mathbf{P}$ is a $1 \times (m-1)$ row vector of ones.

Fig. 3-6 shows the encoder structure for parity-concatenated TCM, which actually is a special case of the well-known Forney's concatenated coding system [35]. Note that, unlike the turbo-style TCM schemes in which a pseudo-random interleaver is applied, here only a conventional block interleaver is employed.

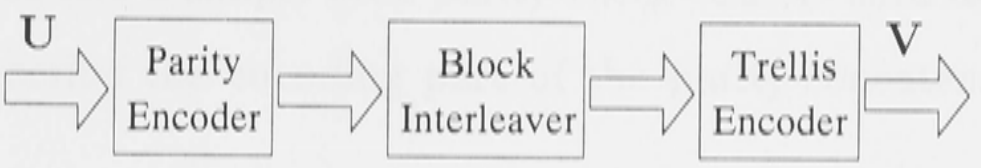


Figure 3-6: Representation of Forney's coding structure for parity-concatenated trellis codes

Now we know that the key idea of parity-concatenated trellis codes is to produce one packet which follows the PC constraint using the $(m-1)$ information packets. When used with a convolutional code, the parity packet is produced based on all the information bits in $(m-1)$ packets [63]. But for trellis codes, if same strategy is applied, potential gains will be lost due to the high degree of redundancy involved.

In section 3.2.1, we have mentioned that the error performance of TCM at high SNRs is mainly determined by $d^2_{free} = \min(d^2_{parallel}, d^2_{free,c})$. Therefore, for TCM systems in which $d^2_{free,c} \leq d^2_{parallel}$, the performance of parity-concatenated trellis codes should not be dramatically affected if we only introduce the parity-check property to the coded bits $u^1_{j,i} \cdots u^{\tilde{k}}_{j,i}$ and the uncoded bits $u^{\tilde{k}+1}_{j,i} \cdots u^{k}_{j,i}$ are left intact, i.e, with the *partial protection* format. It can be seen that most of

the Ungerboeck codes belong to this category. The benefit of such an operation is obvious — the reduction of information transmission rate due to the parity-check bits is minimised. However, for trellis codes in which $d^2_{free,c} > d^2_{parallel}$, *full protection* should be employed, namely, the parity-check constraints are imposed on all the bits of trellis codes. In this case, if only coded bits are protected, performance can be expected to be much worse than the one when the whole symbol is protected because parallel transition errors will dominate the error performance. In section 6.2, the simulation results will show this phenomenon.

3.3.2 Encoding parity-concatenated M-D trellis codes

In this section, we extend the idea of parity-concatenated 2-D TCM to construct a parity-concatenated multi-dimensional TCM in which a M-D trellis code is used as the inner code and a simple even parity check code is used as the outer code. First, let us describe the encoding part of the parity-concatenated M-D TCM schemes.

In this thesis, we will use the popular 4-D 16-state Wei trellis code [67] as an example to illustrate how to construct the parity-concatenated M-D TCM and how to derive the corresponding decoding algorithms. In addition, only a continuous transmission format is considered when building up the parity-concatenated Wei M-D TCM schemes.

In Fig. 3-7, there are $(m-1)$ information streams organized into a block of $(m-1)$ rows. Similar to the case in the previous section, the m^{th} stream, called the *parity stream*, is generated in such a way that the trellis-encoded bits in the m^{th} stream will be the parity bits of the trellis-encoded bits of the $(m-1)$ information streams, i.e., $I^t_{m,c} = \sum_{j=1}^{m-1} \oplus I^t_{j,c}$. The non-trellis-encoded bits in the m^{th} stream are intact.

As shown in section 3.3.1, for parity-concatenated 2-D trellis codes the PC property remains after the trellis encoding procedure. However, this is not always true in constructing the parity-concatenated M-D trellis codes. In the M-D Wei trellis code design, to remove the phase ambiguities of the constellation, a

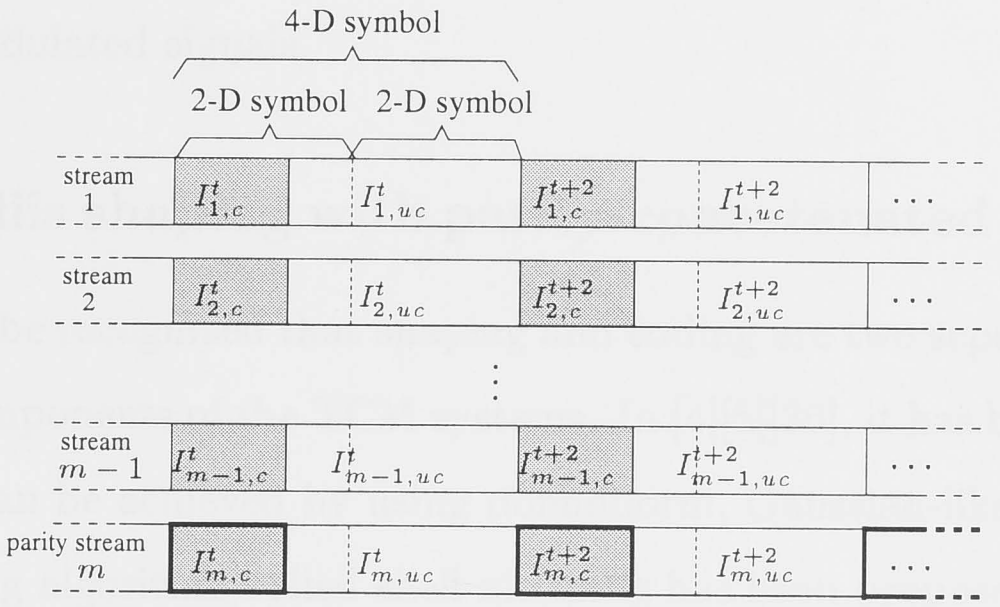


Figure 3-7: Parity-concatenated 4-D trellis codes on a single-parity-check structure consisting of m streams

differential encoder is necessary in the transmitter (see Fig. 3-3). We note that the differential encoder is a nonlinear operator (i.e., a modulo four addition; refer to [67]). Therefore, to prevent the PC property being violated after the differential encoding, we impose the PC property on the trellis-encoded bits of all m streams only after the differential encoding, i.e., $I^t_{j,c}$ $(j = 1, \cdots, m)$ consists of three bits $I1^t_j$, $I2^{t'}_j$ and $I3^{t'}_j$ rather than $I1^t_j$, $I2^t_j$ and $I3^t_j$, (see Fig. 3-3). Then we have

$$\sum_{j=1}^{m} \oplus I1^t_j = \sum_{j=1}^{m} \oplus I2^{t'}_j = \sum_{j=1}^{m} \oplus I3^{t'}_j = 0. \tag{3.5}$$

Next all m streams are then encoded by the same 4-D trellis encoder independently. Due to the linearity of the trellis encoder, the output bits of the trellis encoder also follow the PC property at time t, i.e., $\sum_{j=1}^{m} \oplus I^t_j = 0$, where I^t_j includes the four bits $Y0^t$, $I1^t_j$, $I2^{t'}_j$ and $I3^{t'}_j$. Then, as for the operations in the conventional 4-D Wei encoder, $Y0^t$, $I1^t_j$, $I2^{t'}_j$ and $I3^{t'}_j$ are converted by a bit converter, and then (with those uncoded bits) mapped into a 4-D signal constellation

to produce modulated signals.

3.3.3 Trellis shaping with parity-concatenated trellis codes

It has come to be recognized that shaping and coding are two separable and complementary components of the TCM systems. In [4][5][36], it has been shown that shaping gain can be achieved by using nonuniform, Gaussian-like signalling. An efficient shaping algorithm called shell mapping has been proposed by Khandani and Kabal [5] and several others, and it has been used in the V.34 modem. It acts on N consecutive 2-D constituent constellations (Λ) and selects a subconstellation of Λ_N that approximates the $2N$-D spherical constellation. It is also possible to use block shaping codes to achieve non-equiprobable signalling [4]. In this approach, the 2-D constellation is divided into subregions that have different average signal energies and a block code is used to specify the subregions such that the regions with small average energy are used more often. Another approach, called trellis shaping, was proposed by Forney [36]. Trellis shaping is a method of selecting a minimum-weight sequence from an equivalence class of possible transmitted sequences by searching through the trellis diagram of a shaping convolutional code and it results in "spherical constellations" that can be described only in infinite-dimensional sequence space [36]. It was shown that a simple 4-state shaping code can achieve about 1.0 dB shaping gain, which is about 2/3 of the full 1.53 dB ultimate shaping gain [36]. In this section, we concentrate on trellis shaping combined with the parity-concatenated 2-D TCM systems (Shell mapping is self-contained in the M-D Wei TCM schemes and hence also self-contained in the parity-concatenated M-D TCM). It is expected that similar results can be obtained when the two other approaches to shaping are combined with parity-concatenated TCM schemes.

A general schematic diagram of a parity-concatenated 2-D TCM system with shaping is shown in Fig. 3-8. A 2^n-point 2-D constellation is used to transmit $k = k_c + n_u + r_s$ information bits/T, where k_c, n_u and r_s are the number of channel coded bits x_t, uncoded bits w_t and shaping coded bits s_t, respectively. The 2-D

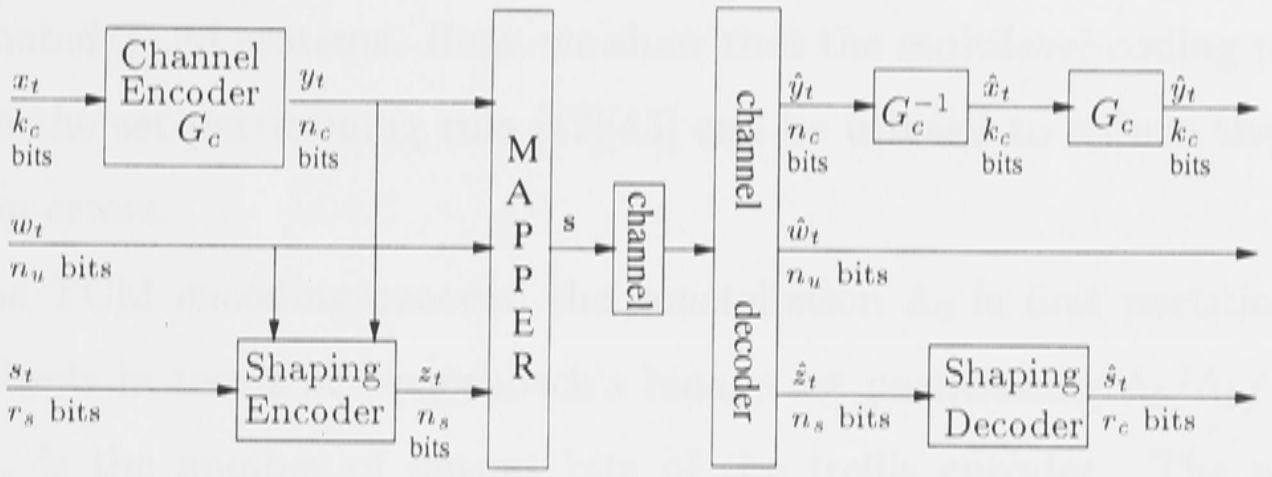


Figure 3-8: Trellis coded modulation system with trellis shaping

constellation is first partitioned into 2^{n_c} subsets using *set partitioning* [31]. The n_c coded bits y_t produced by the channel encoder G_c at time t specify one of the 2^{n_c} subsets. The 2-D constellation is also divided into 2^{n_s} subregions. The n_s shaping bits z_t produced by the shaping encoder at time unit t specify one of the 2^{n_s} subregions. The n_u uncoded bits w_t at time t specify a point $\mathbf{a} = M(y_t, w_t, z_t)$ in the constellation for a given subregion and a given subset.

It was shown in [62] that the free distance of the parity concatenated code could be twice of the free distance of the inner code. We note that the free distance is just determined by the coded bits x_t for such TCM schemes in which $d^2_{parallel}$ is larger than $d^2_{free,c}$. Therefore, the shaping codes s_t should not affect the distance property and the full shaping gains should be achieved when shaping techniques are combined with the parity-concatenated TCM systems. This has been confirmed by simulations presented in Chapter 6.

3.3.4 Multilevel code with parity-concatenated trellis codes

We have known that in the TCM scheme the errors caused by parallel transitions can be ignored at high SNRs, if $d^2_{free,c} \leq d^2_{parallel}$. However, when we apply partial parity checking to protect the coded bits, the $d^2_{free,c}$ for a parity-concatenated code could be doubled. Thus, we may find that in many cases we have $d^2_{parallel} \leq d^2_{free,c}$ and errors among the uncoded bits become dominant, which results in an error floor. The question now is how to reduce the parallel transition errors in parity-

concatenated TCM systems. Here, we show that the multilevel coding technique based on the set partitioning rule [47][43] can be utilised to reduce the parallel transition errors.

In the TCM encoding process, the constellation Λ_0 is first partitioned into 2^{n_c} cosubsets in terms of Ungerboeck's binary set partitioning $\Lambda_0/\Lambda_1/\cdots/\Lambda_{n_c}$, where n_c is the number of output bits of the trellis encoder. The minimum squared Euclidean distance $d^2_{min,(\Lambda_{n_c})}$ between two points within one of the cosubsets of Λ_{n_c} is therefore increased to 2^{n_c} times the minimum squared Euclidean distance $d^2_{min,(\Lambda_0)}$. For instance, if $n_c = 3$, then $d^2_{min,(\Lambda_{n_c})}/d^2_{min,(\Lambda_0)} = 8$, namely, about 9.0 dB gain is obtained for parallel transition bits (i.e., uncoded bits) after set partitioning. However, such a gain is still not big enough to protect those uncoded bits in the low SNR regions. Therefore, an error floor dominated by the uncoded bits will be formed even though errors in the coded bits have been totally eliminated.

Following the concept of the multilevel scheme proposed by Imai and Hirakawa [47], the cosubsets formed by Ungerboeck's set partitioning rule can be further binary partitioned, with one bit associated with such a partition. Obviously, another 3.0 dB gain is achieved for uncoded bits after the partitioning. If we treat trellis codes as the first-level code, then a binary BCH block code (n_b, k_b, q_b) can be selected as the second-level component code, where n_b is the BCH codeword length, k_b is the number of information bits and q_b is the number of error bits that the code can correct. (In [49], a similar scheme was proposed for turbo codes. By extending them with an outer BCH code which would correct a few bit errors, the error floor could then be lowered significantly.)

Here, two basic requirements should be considered for the selection of the second-level code. First, the redundancy introduced by the proposed code should be as small as possible, since we have noted that even a small redundancy can cause a dramatical loss in coding gain, which may offset the gain achieved by introducing the parity check code. Second, the proposed code need not to be a very powerful code. The encoder structure combining a concatenated TCM with

a binary BCH block code is illustrated in Fig. 3-9, in which the convolutional encoder is associated with the partitioning $\Lambda_0/\Lambda_1/\cdots/\Lambda_{n_c}$ and the BCH encoder is associated with the partitioning $\Lambda_{n_c}/\Lambda_{n_c+1}$. The shaping bits and the uncoded bits can then be viewed as a third-level code.

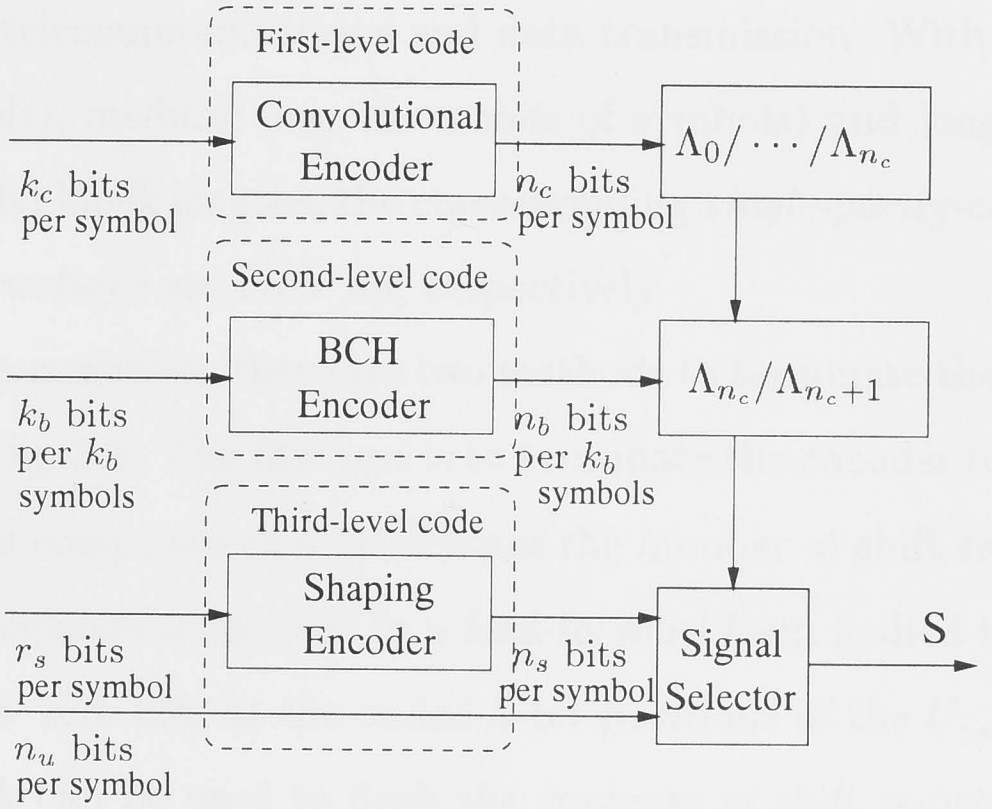


Figure 3-9: Encoder structure of multilevel concatenated code

Due to the introduction of a BCH code, it is clear that $(n_b - k_b)$ redundant bits every k_b trellis symbols are produced. Then the question is how to deal with those redundant bits. In the TCM scheme, the normal method is to expand the constellation size to settle in the redundant bits. But in our case, the average number of redundant bits for each symbol is $(n_b/k_b - 1)$. If the error-correct capability t_b is small, then n_b/k_b will be very close to one for long BCH codes. Therefore, expanding the constellation size is not appropriate here. Instead, we just simply replace $(n_b - k_b)$ bits in parity-concatenated trellis codes with $(n_b - k_b)$ redundant bits. Moreover, such bits should belong to the trellis coded bits because the coded bit errors on the error floor are pretty infrequent and the uncoded bit errors dominate the error floor.

3.4 Single-Parity-Check and Double-Parity-Check Structures

In this section, we will focus on packet transmission formats which have been widely used in telecommunications and data transmission. With short (say hundreds of symbols), medium (say thousands of symbols) and long (say ten thousands of symbols) block lengths, the corresponding *single-parity-check* and *double-parity-check structures* are built up, respectively.

In packet transmission, there are two methods to terminate the encoder for the block given in Fig. 3-5. The first one is to terminate the encoder to a known state. If the encoder is comprised of ν (ν denotes the number of shift registers in trellis encoder) shift registers connected in a feed-forward form (called the feed-forward encoder), then ν zero bits at the coded $\tilde{k}$-bit positions of the $U_{j,i}$ symbols at the end of the block can be used to flush the contents of shift registers to zero, i.e., push the encoder into all zero state. One of the advantages is that the decoder knows the terminal state, and thus it can achieve a better performance [65]. The significant disadvantage is the rate loss due to the tail bits.

The second method is to terminate the encoder at its starting state. This technique is known as *tail-biting* [44] which can make the decoding of the last several (depending on the decoding depth) coded symbols reliable. With tail-biting, the encoder is first initialised by inputing the last ν bits into the encoder and ignoring the output. Therefore, the start and end encoder states are constrained to be identical; that is, a trellis codeword starts from the state at which it will eventually end. The encoding path can be viewed as a trace around a circular trellis.

We know that the trellis codes can be described either in feedback (FB) form or feed-forward (FF) form although almost all good linear trellis codes proposed in the literature are FB codes. It has shown that for convolutional codes every FF code has an equivalent FB code and vice versa, i.e., they generate the same set of codewords [48]. Fig. 3-10 displays the structure of a FF encoder with two

encoded bits. However, we note that in TCM schemes tail-biting technique can always be used with the FF form and any packet length, but not for FB form and all packet lengths.

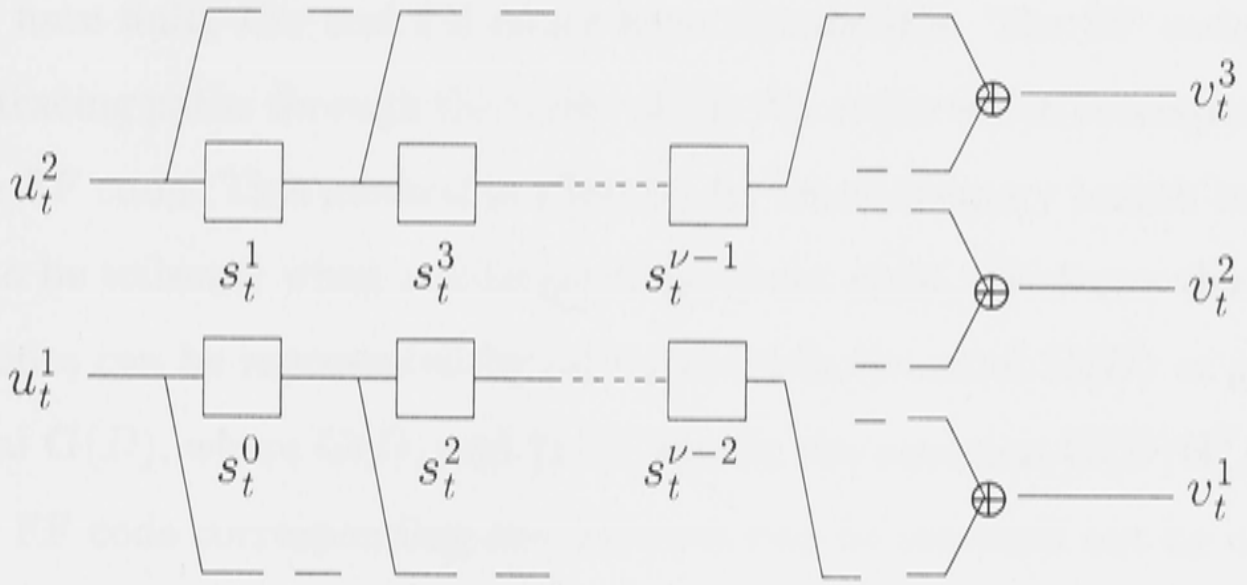


Figure 3-10: Structure of the feed-forward trellis encoder

Suppose there are l symbols $U_1, \cdots, U_l$ in one packet. From Fig. 3-10, the states of shift registers at time $(t+1)$ are updated by the previous states and the input symbol $U_t = (u_t^1, u_t^2)$ at time t, namely,

$$\begin{cases} s_{t+1}^0 &= u_t^1, \\ s_{t+1}^1 &= u_t^2, \\ s_{t+1}^i &= s_t^{i-2}, \qquad i = 2, \cdots, \nu - 1. \end{cases} \tag{3.6}$$

It is obvious that the ending states $(s_l^0, \cdots, s_l^{\nu-1})$ of the packet just depend on the last $\nu/2$ input symbols $U_t, \cdots, U_{t-\nu/2+1}$ and are not affected by earlier input symbols. Therefore, to implement tail-biting, we select the last $\nu/2$ symbols as the initial states and the ending states will always "bite" the beginning states. As to the FB form of codes, however, the ending states are decided by all symbols of the packet as well as the initial states, namely, the ending states satisfy $(s_l^0, \cdots, s_l^{\nu-1}) = \mathbf{F}(U_1, \cdots, U_l; s_0^0, \cdots, s_0^{\nu-1})$, where $\mathbf{F}$ is the appropriate function defined by the encoder structure. Hence, tail-biting can be implemented only if all the symbols $U_1, \cdots, U_l$ in the packet satisfy the given condition determined by the encoder structure. Otherwise, tail-biting will fail for some cases.

Now the problem is how to convert a FB code into a FF code? In [48], an algorithm for converting a FB code to a FF code was proposed based on the definition that the impulse responses (IRs) of a code are the coder outputs, and FF codes have finite IRs and FB codes have infinite IRs. The FF codes can be found by tracing paths through the trellis of the FB codes which correspond to the IRs of the FF code. This method is effective for small memory length codes, but it seems to be toilsome when ν is larger than about eight. We know that the FB and FF codes can be represented by parity check polynomial $\mathbf{H}(D)$ or generator polynomial $\mathbf{G}(D)$, where $\mathbf{G}(D)$ and $\mathbf{H}(D)$ satisfy the equation $\mathbf{G}(D)\mathbf{H}^T(D) = 0$. Thus, the FF code corresponding one FB code can be searched out by computer depending on such a relation. It is noted that generally one FB code has more than one equivalent FF code.

3.4.1 Single-parity-check structure with tail-biting

With the FF trellis encoder, we can use tail-biting technique to encode all the symbols in each packet, as illustrated in Fig. 3-11. We refer to the codes as parity-concatenated trellis codes based on single-parity-check structure, since each symbol in one packet is just protected by single parity-check constraint. In the receiver, the VA can be applied to decode each "unwrapped" packet (see Fig. 3-12). In Fig. 3-12, $R_{j,i}$ $(j = 1, 2, \cdots, m;\ i = 1, 2, \cdots, l)$ denotes the received signal corresponding to the coded symbol $V_{j,i}$, and ρ means the decoding depth. Generally, ρ is $5 \sim 7$ times ν.

We note that, in order to achieve near optimal decoding for the VA on the "unwrapped" trellis, the length of each packet often needs be no less than about 5 times ν. Thus the block length could be more than $5\nu \times m$. In this subsection, we will present a modification which will cut down the block length without significant affecting its error performance [63]. We refer to such a modification as the modified single-parity-check structure.

The key concept of the modified single-parity-check structure is to connect all m packets into one super packet. In other words, instead of terminating each

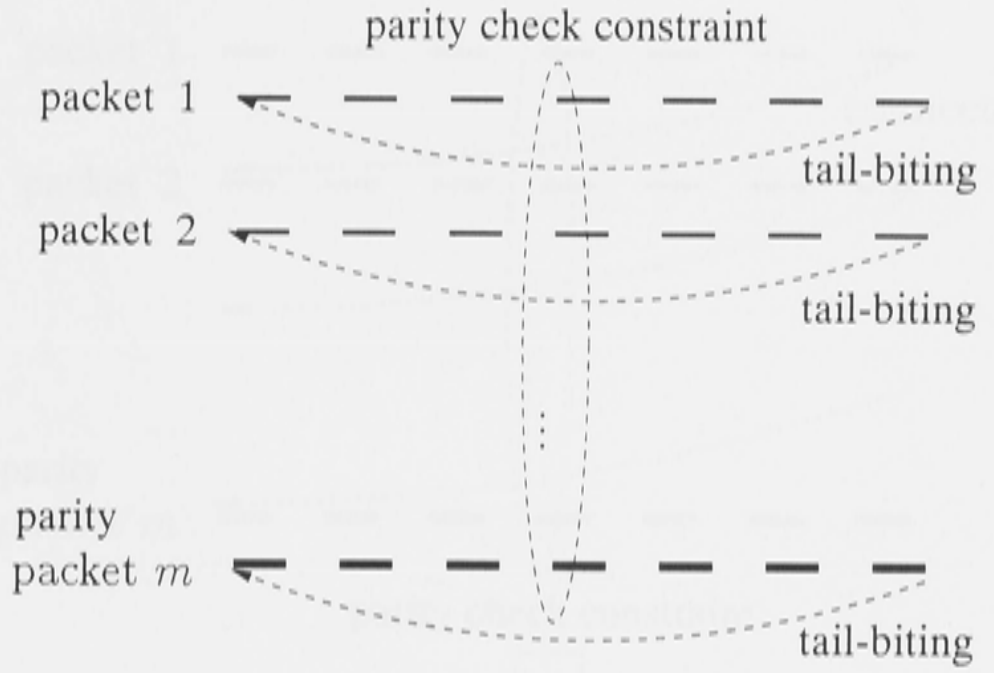


Figure 3-11: Single-parity-check structure with tail-biting for each packet

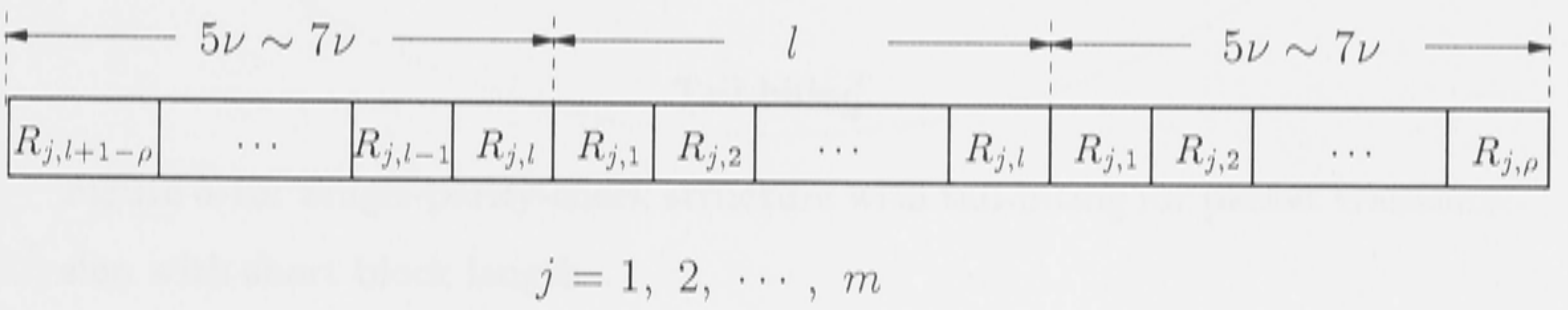


Figure 3-12: Illustration of "unwrapped" packet for the VA decoding

packet, the symbols of all m packets will continue to be fed into the encoder until the last symbol. Figure 3-13 illustrates this modification.

We know that the PC constraint is essential in the encoding structure of the parity-concatenated trellis codes. After connecting all m packets into one super packet, we found that the PC constraint in this super packet still holds. (The detailed proof can be found in [64]). The modified single-parity-check structure will be applied in the rest of the thesis unless otherwise mentioned.

3.4.2 Double-parity-check structure

In the last subsection, parity-concatenated trellis codes have been constructed based on a single-parity-check structure, namely, a symbol of each packet is just parity checked by a group of m symbols in column. We can re-apply the parity-

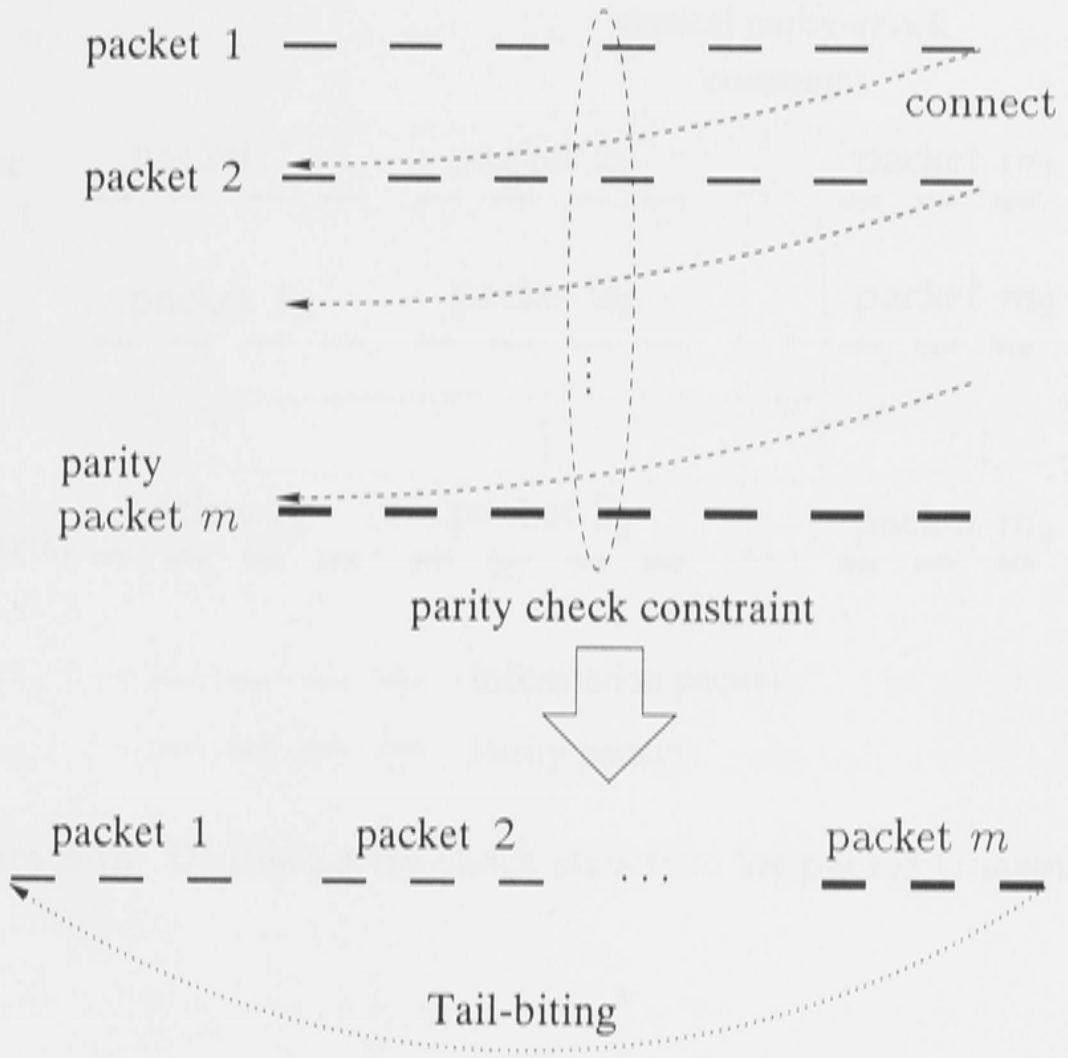


Figure 3-13: Single-parity-check structure with tail-biting for packet transmission with short block length

check constraint on several single-parity-check structures to build up a double-parity-check structure, which is applicable to packet transmission with medium or long block lengths. The main motivation is to build up a block code with a larger minimum free distance, but remain decodable with low complexity. A double-parity-check structure is illustrated in Fig. 3-14. It is shown that there are q super packets, and each of them comprises m packets. Therefore a total of $(m + q - 1)$ parity packets are included in this double-parity-check structure. From Fig. 3-14, we can see that each symbol of each packet now is parity checked by both a group of q symbols in the symbol's column and a group of m symbols in the symbol's row, which correspond to the horizontal and vertical parity-check constraints, respectively. In general, m is set to q since equal parity protection can then be provided by both types of constraints. In addition, it is worth mentioning here that the strategy of either partial protection or full protection discussed for the single-parity-check structure is also suitable for the double-parity-check structure.

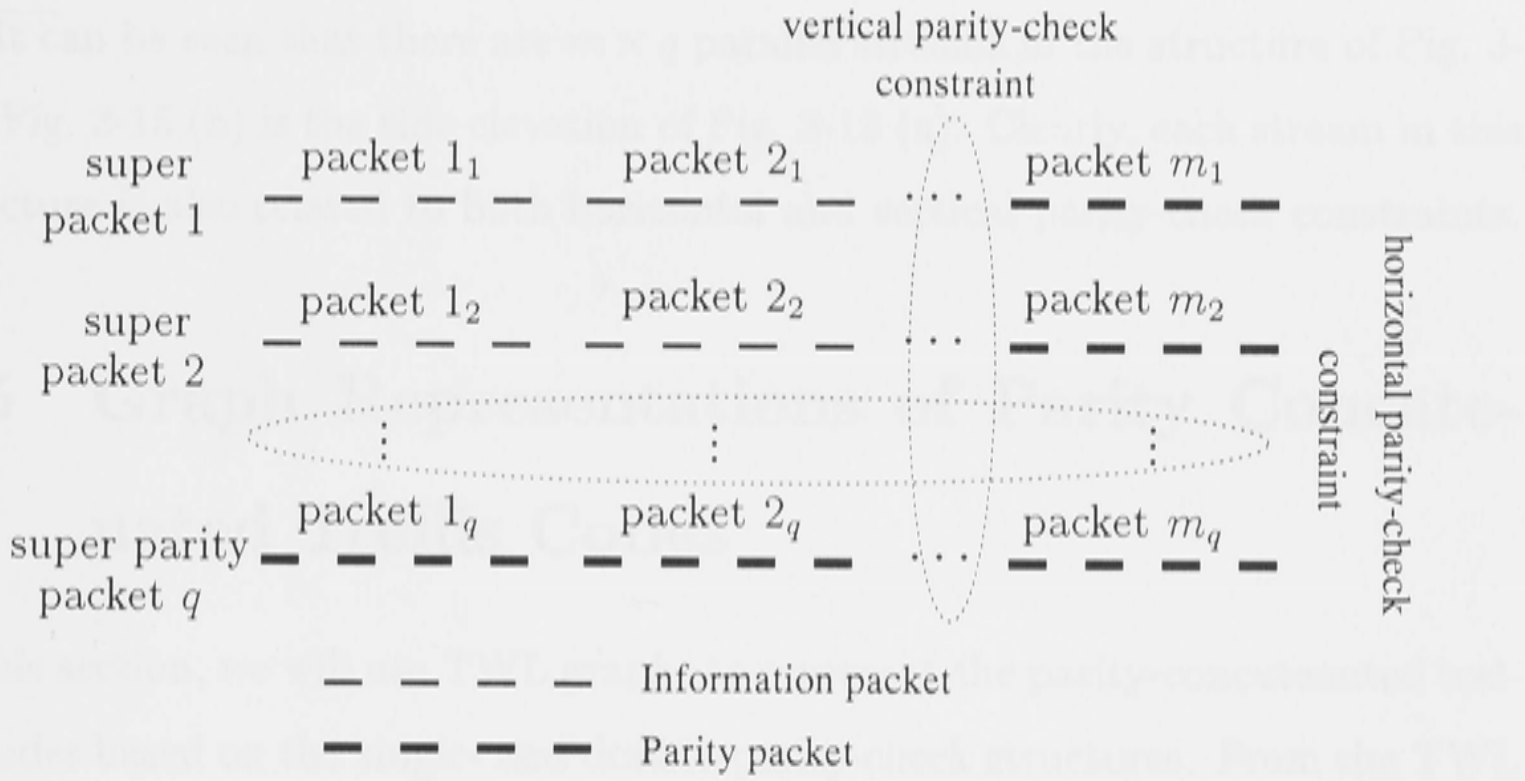


Figure 3-14: Double-parity-check structure for packet transmission

So far, we have got two types of parity-check structures for different block sizes of codes. It is clear that connecting all packets into one super packet in packet transmission can reduce the block size. Obviously, such an operation is not necessary in continuous transmission. Fig. 3-15 shows how the double-parity-check structure can be employed for the continuous transmission format.

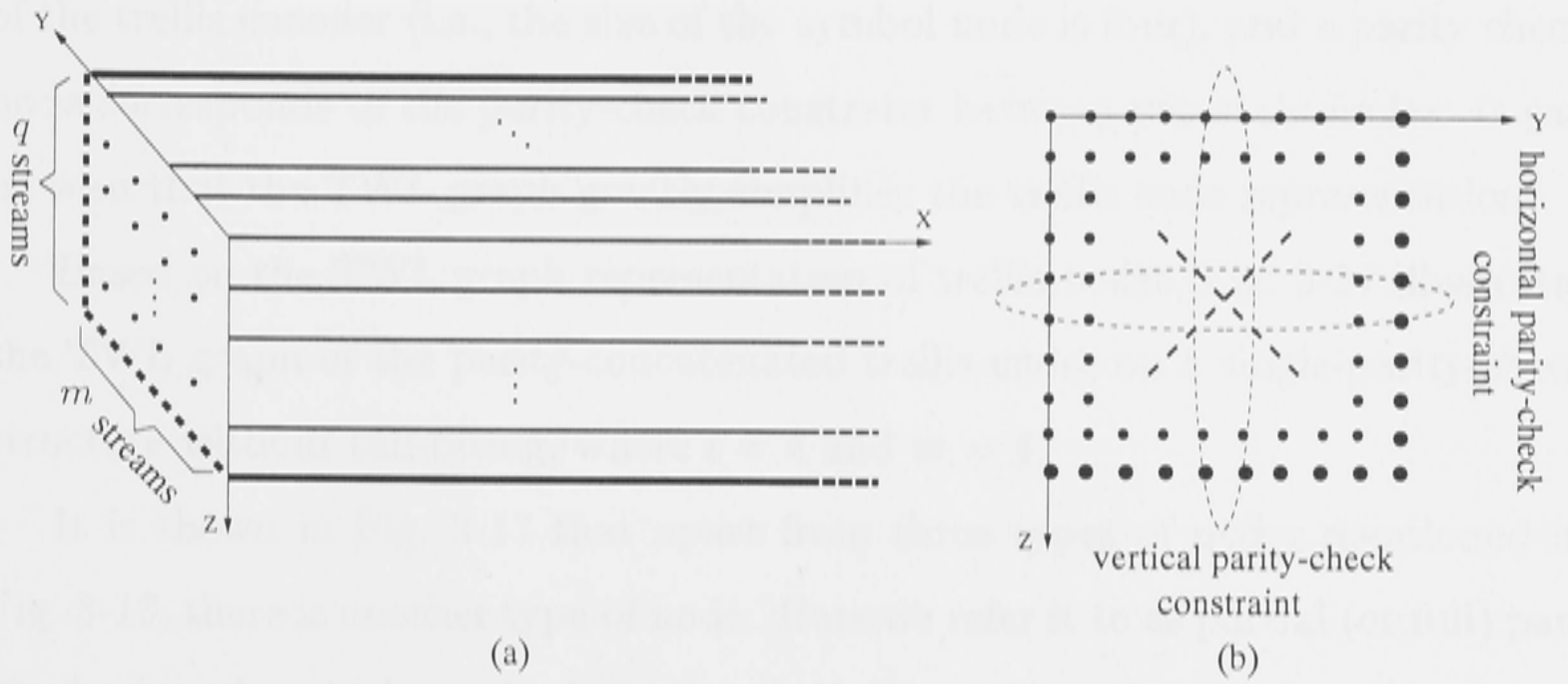


Figure 3-15: Double-parity-check structure for continuous transmission

It can be seen that there are $m \times q$ parallel streams in the structure of Fig. 3-15. Fig. 3-15 (b) is the side elevation of Fig. 3-15 (a). Clearly, each stream in this structure is also related to both horizontal and vertical parity-check constraints.

3.5 Graph Representations of Parity Concatenated Trellis Codes

In this section, we will use TWL graphs to represent the parity-concatenated trellis codes based on the single- and double-parity-check structures. From the TWL graph representations, the corresponding decoding algorithms will be derived in the following chapters.

First we will explain how a trellis code can be represented by a TWL graph. Here a simple example is given to establish the connection between trellis codes and TWL graph representations. Fig. 3-16 illustrates both the trellis representation and the TWL graph representation for a simple 4-state Ungerboeck trellis code. In Fig. 3-16 (b), a state node denotes a 4-state space (i.e., 00, 01, 10 and 11), a coded symbol node represents four types of outputs (B_0, B_1, B_2 and B_3) of the trellis encoder (i.e., the size of the symbol node is four), and a parity check node corresponds to the parity-check constraint between two state nodes. It can be seen that the TWL graph greatly simplifies the trellis code representation.

Based on the TWL graph representation of trellis codes, Fig. 3-17 illustrates the TWL graph of the parity-concatenated trellis codes on a single-parity-check structure without tail-biting, where $l = 4$ and $m = 4$.

It is shown in Fig. 3-17 that apart from three types of nodes mentioned in Fig. 3-16, there is another type of node. Here we refer it to as partial (or full) parity check node, which results from the partial (or full) parity-check constraints in parity-concatenated trellis codes. Since many trellis codes have $d^2_{free,c} \leq d^2_{parallel}$, and for these codes a partial protection can achieve a better performance in terms of approaching the Shannon limit, in this thesis we will focus on partial parity check node unless otherwise mentioned.

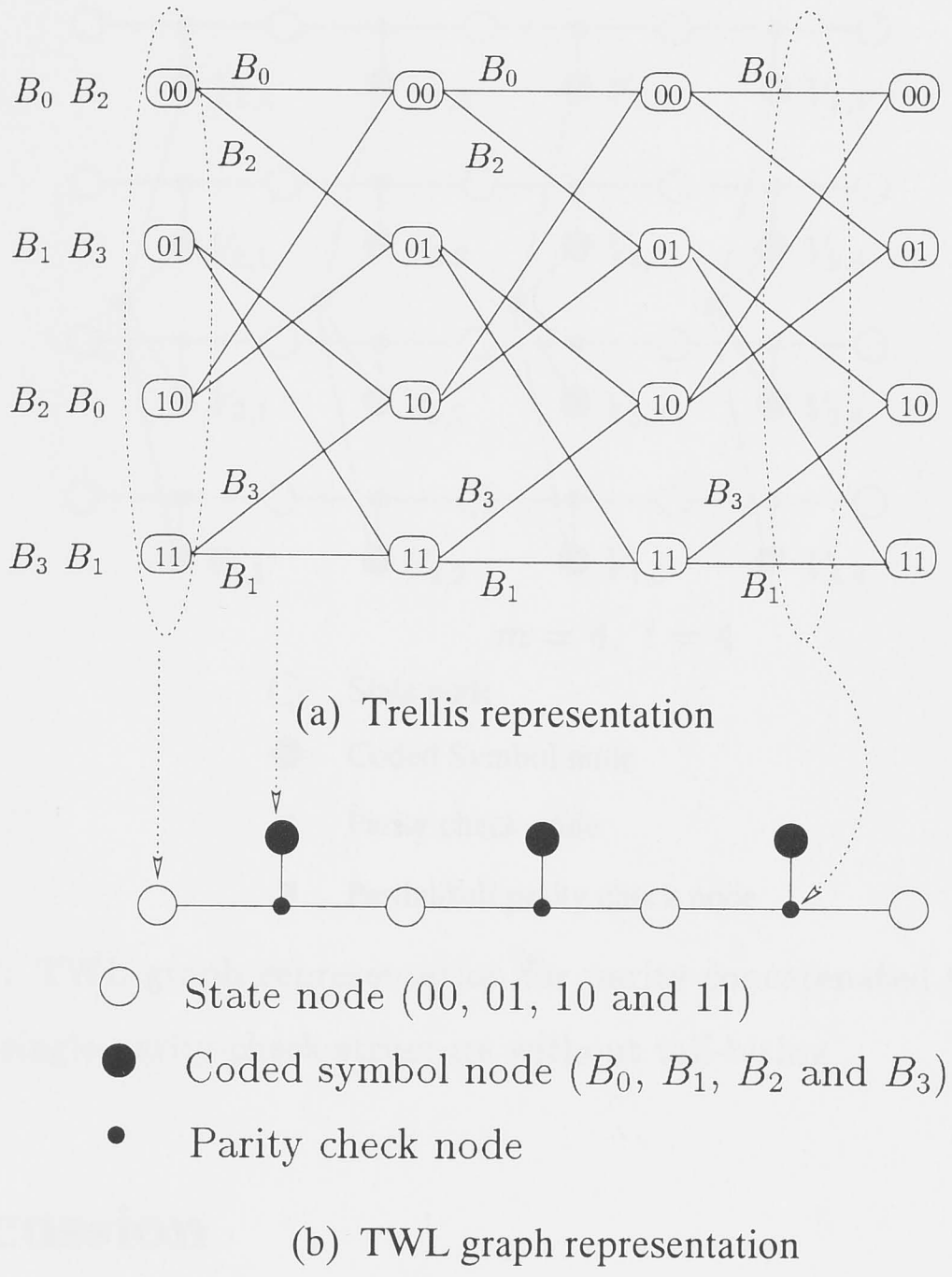


Figure 3-16: Trellis representation and TWL graph representation for a 4-state Ungerboeck trellis code

Figure 3-18 illustrates the TWL graph of the parity-concatenated trellis codes with a modified single-parity-check structure (i.e., with tail-biting), in which $m = 4$ and $l = 4$. It is clear that Fig. 3-18 (b) is another form of Fig. 3-18 (a).

Figure 3-19 illustrates the TWL graph of the parity-concatenated trellis codes with tail-biting on a double-parity-check structure, in which $m = 2$, $q = 3$ and $l = 2$. From Fig. 3-19, it is clearly shown that a symbol node is parity checked by two partial parity check nodes.

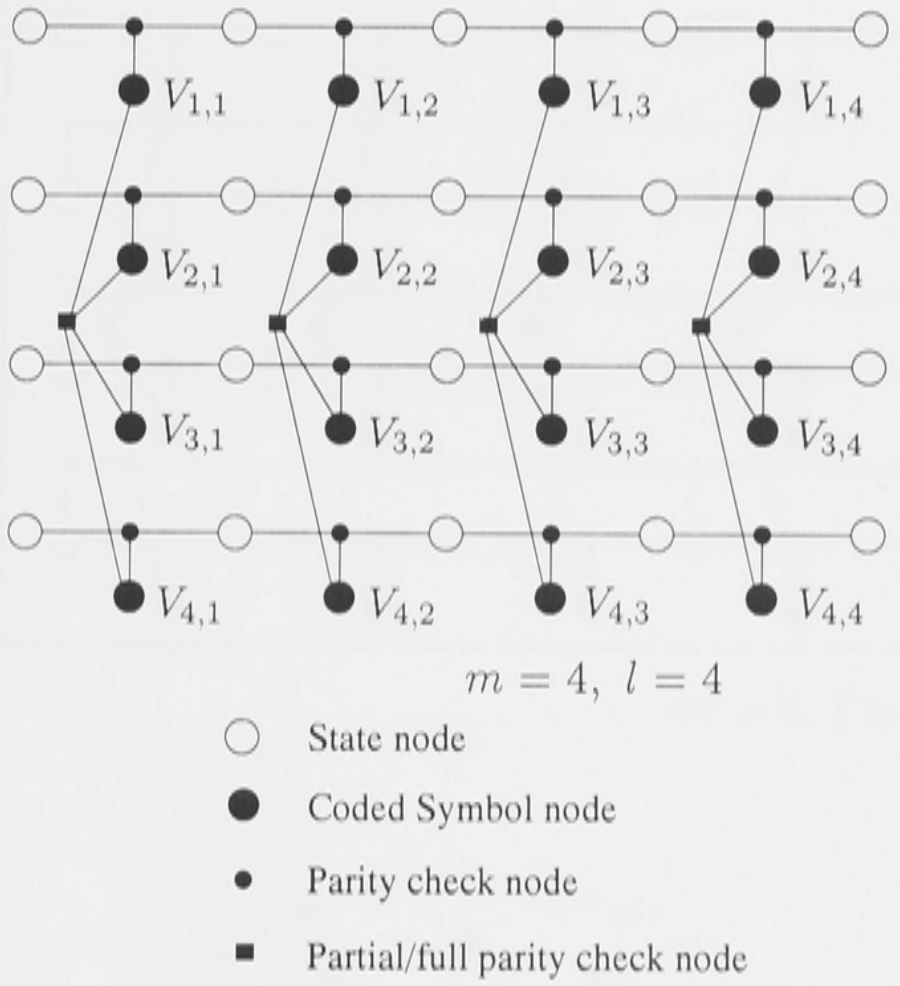
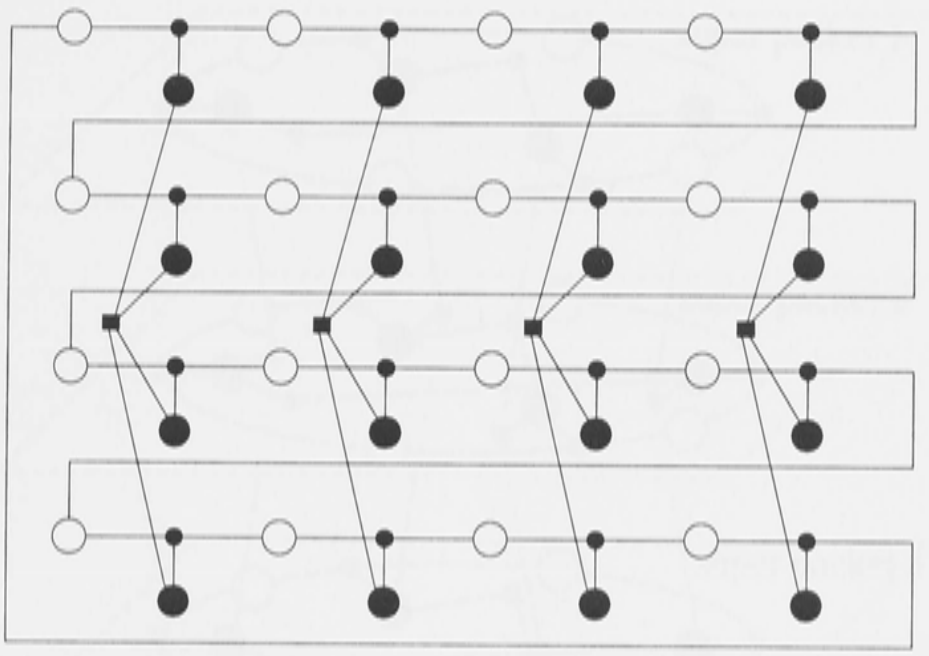


Figure 3-17: TWL graph representation for parity-concatenated trellis codes based on a single-parity-check structure without tail-biting

3.6 Discussion

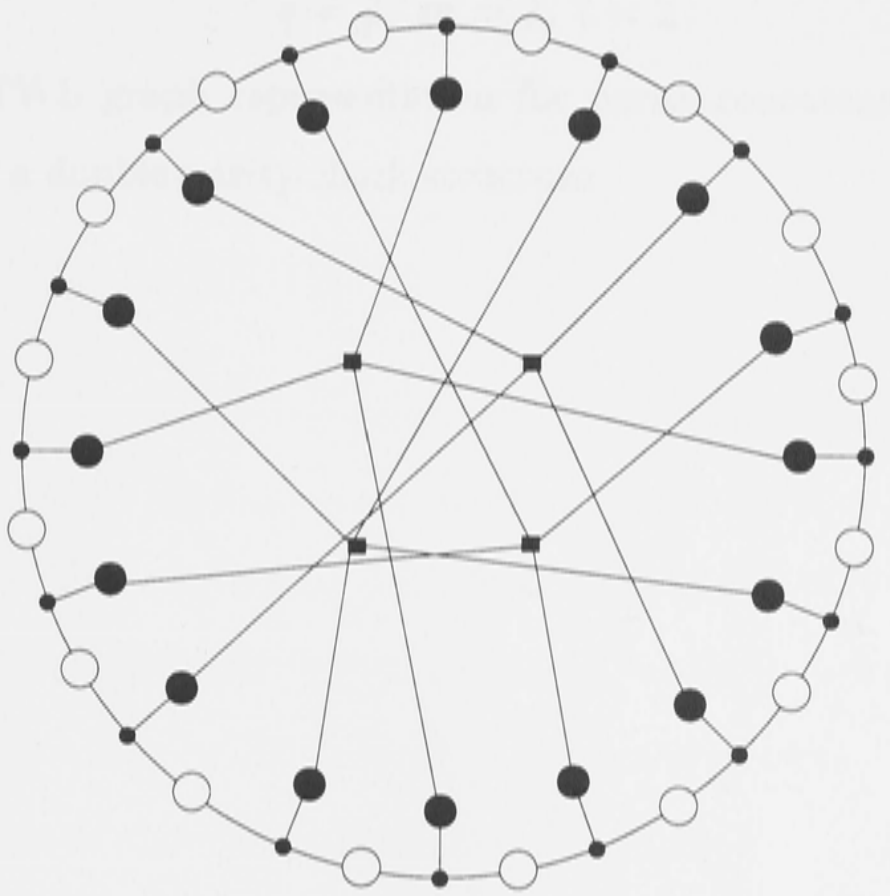
In this chapter, the encoding principles for the Ungerboeck 2-D TCM and Wei multi-dimensional TCM schemes were described first, and then we presented the parity-concatenated trellis code in which a simple even parity check code was serially concatenated with a conventional 2-D or M-D trellis code. We saw that a parity-concatenated trellis code could be viewed as a special case of a Forney's concatenated code. The only difference is that a powerful block outer code is replaced by a simple parity code. Compared with turbo-style TCM schemes, one significant difference is that a conventional block interleaver rather than a pseudo-random one is employed in the encoder.

For both packet and continuous transmissions, the corresponding single- and double-parity-check structures were designed, respectively. Packet transmission with long, medium and short block lengths were considered. In addition, shaping techniques and multilevel codes were combined with the parity-concatenated



$m = 4,\ l = 4$

(a)

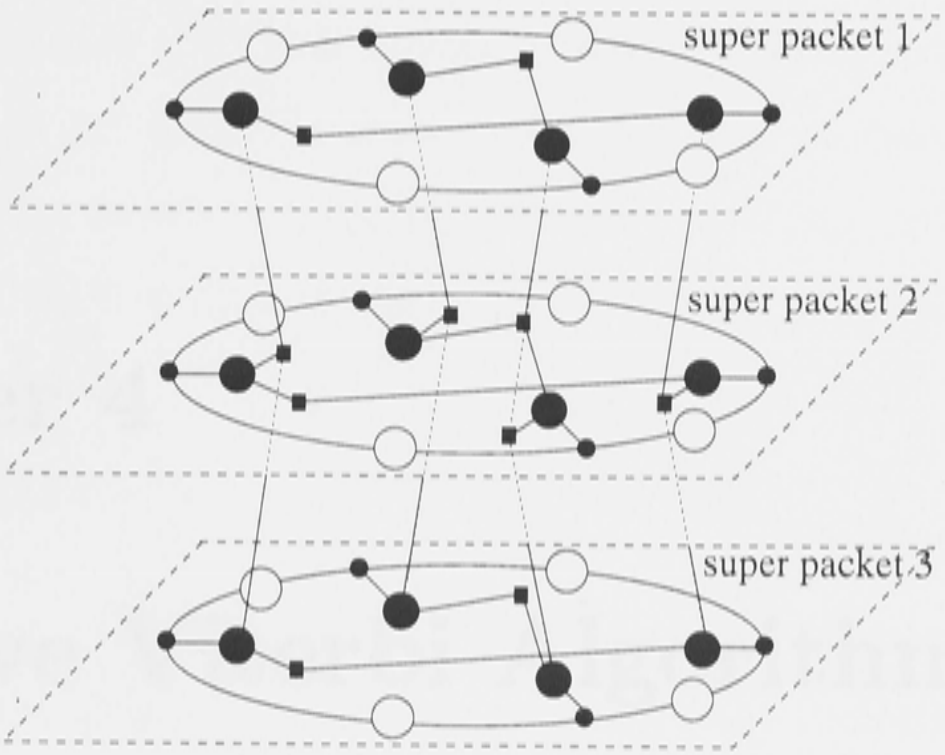


$m = 4,\ l = 4$

(b)

Figure 3-18: TWL graph representation for parity-concatenated trellis codes on a single-parity-check structure with tail-biting

trellis codes for further performance improvement. As an effective tool for code interpretation, TWL graph representations for the parity-concatenated trellis codes were presented. Based on the TWL graph representations, the iterative decoding algorithms for the parity-concatenated trellis codes will be presented in the



$$q = 3,\ m = 2,\ l = 2$$

Figure 3-19: TWL graph representation for parity-concatenated tail-biting trellis codes on a double-parity-check structure

following chapters.

Chapter 4

Iterative Viterbi Algorithm for Parity-Concatenated TCM

4.1 Iterative Decoding Algorithms

Iterative decoding is a generic term for decoding algorithms whose basic operation is to modify some internal state in small steps until a valid codeword is reached [77]. In our framework, the decoder utilises the parity-check constraints among the parity-concatenated trellis codes to "collect" *extrinsic* information which will be fed back into the decoder for the next iteration. Fig. 4-1 illustrates the fundamental idea of this type of iterative decoder.

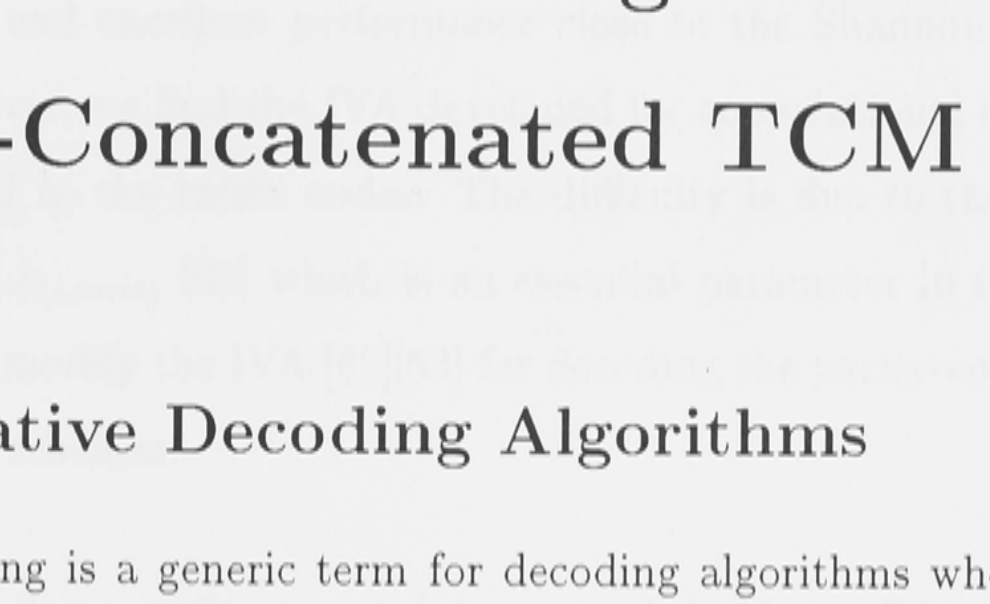


Figure 4-1: Block diagram of the iterative decoder

In this chapter and the following chapters we will study several iterative decoding algorithms for decoding the trellis codes and calculating the *extrinsic* information for the iterative decoding. First let us study the iterative Viterbi algorithm (IVA) for decoding the parity-concatenated trellis codes.

4.2 Iterative Viterbi Algorithm

In [62][63][64], Wei has presented the IVA for decoding the concatenated convolutional codes, and excellent performance close to the Shannon limit has been achieved. However, we find the IVA developed for convolutional codes cannot be simply extended to the trellis codes. The difficulty is due to the complexity of computation of $\lambda_{ij,parity}$ [62] which is an essential parameter in the IVA. In this section, we will modify the IVA [62][63] for decoding the parity-concatenated 2-D and M-D TCM schemes.

4.2.1 IVA for parity-concatenated 2-D trellis codes

First let us consider the IVA based on a single-parity-check structure without tail-biting. As shown later, the only difference between the IVA and the standard VA is the calculation of branch metric. Therefore we will derive the new branch metric function which takes into account the parity-check constraint for the IVA. We will focus on a code in which only the coded bits are protected by the parity-check constraint (i.e., partial protection). The IVA for the code with full protection can be obtained in a similar way.

Let $R_{j,i}$ $(j = 1, 2 \cdots, m;\ i = 1, 2, \cdots, l)$ denote the i^{th} received signal of the j^{th} packet in the receiver, and $V_{j,i} \equiv (V_{j,i}^{(p)}; V_{j,i}^{(np)}) = (v_{j,i}^{1} \cdots v_{j,i}^{\tilde{k}+1}; v_{j,i}^{\tilde{k}+2} \cdots v_{j,i}^{n})$ denotes the i^{th} coded symbol of the j^{th} packet, where $V_{j,i}^{(p)} = (v_{j,i}^{1} \cdots v_{j,i}^{\tilde{k}+1})$ is the output from the convolutional encoder with rate $\tilde{k}/(\tilde{k}+1)$, and $V_{j,i}^{(np)} = (v_{j,i}^{\tilde{k}+2} \cdots v_{j,i}^{n})$ is the uncoded part of the input symbol $U_{j,i}$. If only the coded bits are protected, then only $V_{j,i}^{(p)}$ satisfies the parity-check constraint in the coded symbol. Suppose that the $1^{st}, 2^{nd}, \cdots, (m-2)^{th}$ packets have been successfully decoded by the

standard VA in the first iteration, and now we are going to decode the $(m-1)^{th}$ packet using the updated (new) branch metrics. The original PC constraint on the i^{th} symbols in one block (m packets) is

$$W_{m,i} = V_{1,i}^{(p)} \oplus V_{2,i}^{(p)} \oplus \cdots \oplus V_{m,i}^{(p)} = 0, \qquad i = 1, 2, \cdots, l. \tag{4.1}$$

Let $W_{m-2,i} = V_{1,i}^{(p)} \oplus \cdots \oplus V_{m-2,i}^{(p)}$. The receiver replaces the j^{th} ($j = 1, 2, \ldots, m-2$) received packets by the estimated j^{th} transmitted packets. So we can get the estimated PC constraint

$$\widehat{W}_{m-2,i} = \widehat{V}_{1,i}^{(p)} \oplus \cdots \oplus \widehat{V}_{m-2,i}^{(p)}, \tag{4.2}$$

where $\widehat{V}_{j,i}^{(p)}$ is the decision of $V_{j,i}^{(p)}$. Now if considering the parity-check constraint among the received signals $R_{1,i}, \cdots, R_{m-1,i}$ and $R_{m,i}$, we can get the likelihood function for decoding the $(m-1)^{th}$ packet as follows,

$$\lambda'_{m-1,i} = -\log\left[P(R_{1,i}, \cdots, R_{m-1,i}, R_{m,i} | V_{m-1,i}^{(p)}; V_{m-1,i}^{(np)})\right], \tag{4.3}$$

where $V_{m-1,i}^{(p)}$ and $V_{m-1,i}^{(np)}$ are coded part and uncoded part of symbol $V_{m-1,i}$, respectively. Assuming $R_{1,i}, \cdots, R_{m-1,i}$ and $R_{m,i}$ are independent of each other[1], we then have

$$\begin{aligned} \lambda'_{m-1,i} &\approx -\log\left[P(R_{m-1,i} | V_{m-1,i}^{(p)}; V_{m-1,i}^{(np)})\right] \\ &\quad -\log\left[P(R_{1,i}, \cdots, R_{m-2,i}, R_{m,i} | V_{m-1,i}^{(p)}; V_{m-1,i}^{(np)})\right], \\ &= \lambda_{m-1,i}^{(v)} + \lambda_{m-1,i}^{(p)}. \end{aligned} \tag{4.4}$$

where $\lambda_{m-1,i}^{(v)}$ denotes the branch metric value which is identical to the metric used in the VA, and $\lambda_{m-1,i}^{(p)}$ denotes the *extrinsic* metric value introduced by the PC constraint from the other packets. With the IVA, computing $\lambda_{m-1,i}^{(p)}$ for high-rate TCM can be significantly simplified.

If $\widehat{W}_{m-2,i} = W_{m-2,i}$, where $W_{m-2,i} = V_{1,i}^{(p)} \oplus \cdots \oplus V_{m-2,i}^{(p)}$, then we have

$$\widehat{W}_{m-2,i} \oplus V_{m-1,i}^{(p)} \oplus V_{m,i}^{(p)} = 0, \tag{4.5}$$

[1] Actually, $R_{1,i}, \cdots, R_{m,i}$ are weakly dependent due to the PC constraint among them.

or

$$V_{m-1,i}^{(p)} = V_{m,i}^{(p)} \oplus \widehat{W}_{m-2,i}. \tag{4.6}$$

Therefore, $V_{m,i}^{(p)}$, the coded part of the i^{th} codeword in the m^{th} packet, can be obtained through the PC constraint and $(m-2)$ decoded received packets as well. However, there is no PC property among the uncoded part $V_{j,i}^{(np)}$ of symbols, so the question is how to determine the uncoded part $V_{m,i}^{(np)}$ of the i^{th} codeword for the m^{th} packet.

We know that the coded part $V_{m,i}^{(p)}$ decides which subset will be selected in the constellation, and the parallel transition error in the subset can be ignored at high SNRs. Therefore, after $V_{m,i}^{(p)}$ is determined, $V_{m,i}^{(np)}$ can also be decided by selecting one point $(V_{m,i}^{(p)}; V_{m,i}^{(np)})$ which is the closest to the received signal $R_{m,i}$ in this subset. Therefore, we have

$$\begin{aligned} \lambda_{m-1,i}^{(p)} &= -\log\left[P(R_{1,i}, \cdots, R_{m-2,i}, R_{m,i}|V_{m-1,i}^{(p)}; V_{m-1,i}^{(np)})\right], \\ &= -\log\left[P(R_{1,i}, \cdots, R_{m-2,i}, R_{m,i}|V_{m,i}^{(p)} \oplus \widehat{W}_{m-2,i}; V_{m,i}^{(np)})\right]. \end{aligned} \tag{4.7}$$

Here $V_{m-1,i}^{(np)}$ has been replaced by $V_{m,i}^{(np)}$. Exact calculation of (4.7) is very computation expensive and almost practically impossible when m is large. Also, considering $R_{1,i}, \cdots, R_{m-2,i}$ are just weakly dependent on $R_{m,i}$, therefore, we approximate (4.7) as

$$\lambda_{m-1,i}^{(p)} \approx -\log[P(R_{m,i}|V_{m,i}^{(p)} \oplus \widehat{W}_{m-2,i}; V_{m,i}^{(np)})]. \tag{4.8}$$

Now the metric function $\lambda_{m-1,i}^{(p)}$ is approximately equal to the branch metric used in the VA for the m^{th} packet. Thus the computation of (4.4) is simply the sum of the VA branch metric functions of the $(m-1)^{th}$ and m^{th} packets. It is worth mentioning here that $\lambda_{m-1,i}^{(p)}$ will not be the correct metric and the error propagation will result if $\widehat{W}_{m-2,i} \neq W_{m-2,i}$. The effect of error propagation can be reduced by scaling down the value of $\lambda_{m-1,i}^{(p)}$ with a scaling factor α, i.e,

$$\lambda'_{m-1,i} = \lambda_{m-1,i}^{(v)} + \alpha \cdot \lambda_{m-1,i}^{(p)}. \tag{4.9}$$

We will discuss α in more detail in section 4.2.3.

So far, we have considered the branch metric function on a single-parity-check structure. In the double-parity-check structure, there are two types of PC constraints which can produce two sorts of *extrinsic* metrics[2] (see Figs. 3-14 and 3-15). The horizontal PC constraint provides the *extrinsic* metric given in (4.8) for each super packet, and the vertical PC constraint provides the *extrinsic* metric for each column in the structure given in Fig. 3-14. These two *extrinsic* metrics are almost independent each other. Therefore, for a double-parity-check structure, the updated branch metric function of one symbol will be the sum of its original VA branch metric value and two *extrinsic* metric values. For example, for the m^{th} packet of the q^{th} super packet, the updated branch metric functions $\lambda'_{q-1,m-1,i}$ will be

$$\lambda'_{q-1,m-1,i} = \lambda^{(v)}_{q-1,m-1,i} + (\lambda^{(p)(h)}_{q-1,m-1,i} + \lambda^{(p)(v)}_{q-1,m-1,i}) \cdot \alpha, \qquad i = 1, 2, \cdots, l, \quad (4.10)$$

where α is a scaling parameter, $\lambda^{(v)}_{q-1,m-1,i}$ is the original VA branch metric of the symbol, $\lambda^{(p)(h)}_{q-1,m-1,i}$ and $\lambda^{(p)(v)}_{q-1,m-1,i}$ are the *extrinsic* metric values introduced by the horizontal and vertical PC constraints, respectively. It is clear that the calculations of $\lambda^{(p)(v)}_{q-1,m-1,i}$ and $\lambda^{(p)(h)}_{q-1,m-1,i}$ are exactly the same.

4.2.2 IVA for parity-concatenated M-D trellis codes

In this section, we will extend the IVA discussed above for decoding the parity-concatenated 4-D Wei trellis codes. The cases of decoding the 8-D and higher dimensional concatenated trellis codes can be obtained through the similar way. Similarly, first let us derive the updated branch metric used in the IVA for decoding the parity-concatenated 4-D trellis codes based on a single-parity-check structure (i.e., m streams).

As mentioned in section 3.3.2, the outputs (four bits) from the 4-D trellis encoder follow the PC property due to the linearity of the 4-D Wei trellis encoder.

[2] A third type of parity-check constraint can be contemplated for the double-parity-check structure which is a combination of row and column check sums. However, it is found that this posed computational and technical difficulties which did not significantly contribute to the performance.

In the 4-D Wei TCM schemes, to make the schemes transparent to all phase ambiguities of the constellation, a bit converter converts the output bits $Y0^t$, $I1_i^t$, $I2_i^{t'}$ and $I3_i^{t'}$ into two pairs of selection bits $Z0^tZ1^t$ and $Z0^{t+1}Z1^{t+1}$, see Fig. 3-3. Unfortunately, we found that such a conversion destroys the PC property of some output bits of bit converter. For the 4-D Wei code, we can see $\sum_{i=1}^{m} \oplus Z1_i^{t+1} \neq 0$ (The conversion table is in [67].) Fortunately, this PC property can be preserved if bit conversion is viewed as a part of the 4-D constellation mapping.

Let R_j^t $(j = 1, 2, \cdots, m)$ denote the received 4-D signals of m streams at time t, where R_j^t actually consists of two 2-D component signals: r_j^t and r_j^{t+1} which are resulted from two 2-D sub-constellations, respectively. Let $Z_j^t \equiv (Z_j^{t(p)}; Z_j^{t(np)}) = (Z0_j^t, Z1_j^t; Z2_j^t, \cdots, Z\iota_j^t)$ denote a 2-D encoded symbol of the j^{th} stream in the first 2-D subconstellation at time t, where $Z_j^{t(p)}$ is the output from the bit converter to select a 2-D subset in the first 2-D subconstellation, $Z_j^{t(np)}$ is the non-trellis-encoded part of the input symbol, and ι is the number of the bits in a 2-D input symbol. Similarly, let $Z_j^{t+1} \equiv (Z_j^{t+1(p)}; Z_j^{t+1(np)}) = (Z0_j^{t+1}, Z1_j^{t+1}; Z2_j^{t+1}, \cdots, Z\iota_j^{t+1})$ denote a 2-D encoded symbol of the j^{th} stream in the second 2-D subconstellation at time t.

Suppose that the j^{th} $(j = 1, 2, \cdots, m-2)$ stream at time t has been successfully decoded using the standard VA and now we are going to decode the $(m-1)^{th}$ stream at time t. Obviously, the original PC constraint on the m streams at time t is

$$W_m^t = \sum_{j=1}^{m} \oplus I_j^t = 0, \tag{4.11}$$

where I_j^t comprises four encoded bits $Y0_j^t$, $I1_j^t$, $I2_j^{t'}$ and $I3_j^{t'}$. Let $W_{m-2}^t = \sum_{j=1}^{m-2} \oplus I_j^t$. Then the receiver replaces the j^{th} $(j = 1, 2, \cdots, m-2)$ stream by the estimated j^{th} transmitted stream at time t. So we can get the estimated PC constraint

$$\widehat{W}_{m-2}^t = \sum_{j=1}^{m-2} \oplus \hat{I}_j^t, \tag{4.12}$$

where $\hat{I}_j^t$ is the decision of I_j^t. In generalizing, let $(Z_j^{t(p)}; Z_j^{t+1(p)}) = \Psi(I_j^t)$, where

$\Psi()$ is the bit conversion function determined by the bit converter. Now the updated likelihood function of the 4-D type[3] in the $(m-1)^{th}$ stream at time t is

$$\lambda_{m-1}^{t} = -\log\left[P(R_1^t, R_2^t, \cdots, R_m^t | Z_{m-1}^t, Z_{m-1}^{t+1})\right], \tag{4.13}$$

where Z_{m-1}^t and Z_{m-1}^{t+1} are the first and second 2-D symbols in the $(m-1)^{th}$ stream at time t, respectively. Assuming R_1^t, R_2^t, $\cdots$ and R_m^t are independent of each other[4], we then have

$$\begin{aligned}\lambda_{m-1}^{t} &\approx -\log\left[P(R_{m-1}^t | Z_{m-1}^t, Z_{m-1}^{t+1})\right] \\ &\quad -\log\left[P(R_1^t, \cdots, R_{m-2}^t, R_m^t | Z_{m-1}^t, Z_{m-1}^{t+1})\right] \\ &= \lambda_{m-1}^{t(v)} + \lambda_{m-1}^{t(p)},\end{aligned} \tag{4.14}$$

where $\lambda_{m-1}^{t(v)}$ denotes the branch metric value (of the 4-D type) which is identical to the branch metric (of the 4-D type) used in the VA, and $\lambda_{m-1}^{t(p)}$ denotes the *extrinsic* metric value introduced by the PC property from the other streams.

If $\widehat{W}_{m-2}^t = W_{m-2}^t$, where $W_{m-2}^t = \sum_{j=1}^{m-2} \oplus I_j^t$, then we have

$$\widehat{W}_{m-2}^t \oplus I_{m-1}^t \oplus I_m^t = 0. \tag{4.15}$$

Therefore, using the relationship between $(Z_j^{t(p)}; Z_j^{t+1(p)})$ and I_j^t, we can get

$$\begin{aligned}\lambda_{m-1}^{t(p)} &= -\log\left[P(R_1^t, \cdots, R_{m-2}^t, R_m^t | \Psi(I_{m-1}^t); Z_{m-1}^{t(np)}, Z_{m-1}^{t+1(np)})\right] \\ &= -\log\left[P(R_1^t, \cdots, R_{m-2}^t, R_m^t | Z_m^{t(p)'}; Z_{m-1}^{t(np)}, \right. \\ &\quad \left. Z_m^{t+1(p)'}; Z_{m-1}^{t+1(np)})\right],\end{aligned} \tag{4.16}$$

where $(Z_m^{t(p)'}; Z_m^{t+1(p)'}) = \Psi(I_m^{t\,\prime})$ and $I_m^{t\,\prime} = I_m^t \oplus \widehat{W}_{m-2}^t$.

Similar with the IVA for 2-D trellis codes, the uncoded parts $Z_m^{t(np)}$ and $Z_m^{t+1(np)}$ can be determined by selecting two points $(Z_m^{t(p)'}; Z_m^{t(np)})$ and $(Z_m^{t+1(p)'}; Z_m^{t+1(np)})$ which are the closest to the received 2-D component signals r_m^t and r_m^{t+1} in the subset, respectively. In addition, exactly calculation of (4.16) is also computationally expensive and almost practically impossible when m is large. Hence, we

[3]The definition of the 4-D type can be found in Wei's paper [67].

[4]Actually, $R_1^t, \cdots, R_m^t$ are weakly dependent due to the PC property among them.

approximate (4.16) as

$$\lambda_{m-1}^{t(p)} \approx -\log\left[P(R_m^t | Z_m^{t(p)'}; Z_m^{t(np)}, Z_m^{t+1(p)'}; Z_m^{t+1(np)})\right]. \tag{4.17}$$

Here in (4.17) $Z_{m-1}^{t(np)}$ and $Z_{m-1}^{t+1(np)}$ have been replaced by $Z_m^{t(np)}$ and $Z_m^{t+1(np)}$, respectively.

Now the metric function $\lambda_{m-1}^{t(p)}$ is equal to the branch metric of the 4-D type used in the VA for the m^{th} stream. The computation of (4.14) is simply the sum of the VA branch metrics of the $(m-1)^{th}$ and m^{th} streams. The effect of error propagation can be reduced by scaling down the value of $\lambda_{m-1}^{t(p)}$ with a scaling factor α. After λ_{m-1}^{t} is obtained, the rest of the steps for decoding the 4-D symbol are exactly the same as the procedures in [67].

For the codes based on a double-parity-check structure, calculation of branch metric function for the 4-D parity-concatenated trellis codes is the same as that for the 2-D trellis codes, namely, the *extrinsic* metric values in a double-parity-check structure is the sum of component metrics introduced by horizontal and vertical PC properties, respectively.

Comparing the IVA for decoding the parity-concatenated 2-D and 4-D trellis codes, we can see that extra effort has to be made for updating the PC constraint in the latter case due to a nonlinear operation (bit converter). Since the conversions between $(Y0_j^t,\ I1_j^t,\ I2_j^{t'},\ I3_j^{t'})$ and $(Z0_j^t,\ Z1_j^t,\ Z0_j^{t+1},\ Z1_j^{t+1})$ $(j = 1, \cdots, m)$ are able to be implemented through a table-looking, no extra computation actually is consumed by this part.

4.2.3 Some relevant issues on the IVA

A. Block size of the parity check structures

We have noted that the key concept in the IVA is the introduction of *extrinsic* metric value $\lambda_{j,i}^{(p)}$ $(j = 1, 2, \cdots, m)$ based on the $(m-1)$ packets when one packet is being decoded (here supposing the single-parity-check structure is used). Since the m^{th} packet (parity packet) actually represents redundancy, then how to choose the value of m is an important issue. If we extend to the double-parity-check

structure, then the analogous question is how to select m as well as q. Now suppose the partial protection is employed in the parity-concatenated 2-D trellis codes, and the spectral efficiency is k bits/T. After deduction of redundant bits, the real rate k_{real} in a single-parity-check structure (m packets) is

$$k_{real} = k - \tilde{k}/m, \tag{4.18}$$

where $\tilde{k}$ is the number of bits involved in the partial protection. Similarly, the real rate k_{real} in a double-parity-check structure ($m \times q$ packets) is

$$k_{real} = k - \frac{(m+q-1)}{mq}\tilde{k}. \tag{4.19}$$

For the parity-concatenated 4-D Wei trellis codes, suppose each 2-D symbol includes ι bits, and each 4-D symbol has $\tilde{\iota}$ trellis-encoded bits which are involved in the PC constraint, then the real rate ι_{real} in a single-parity-check structure (m streams) is

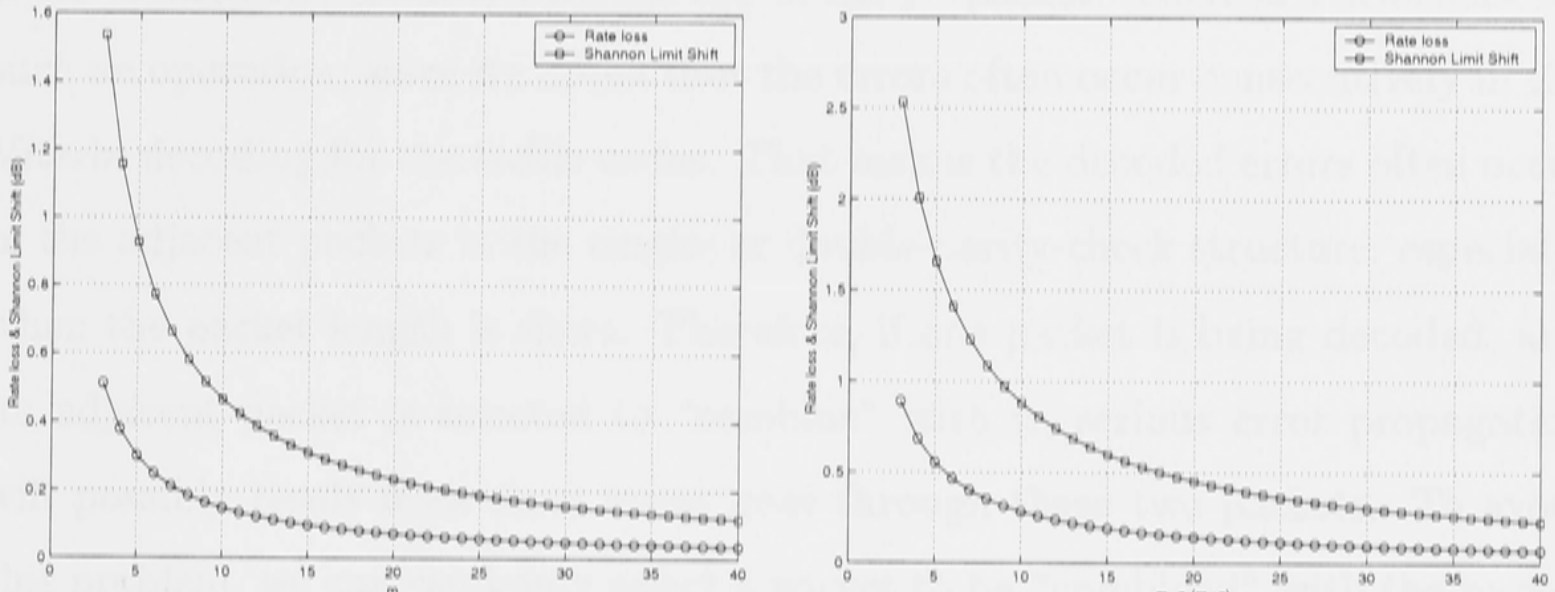
$$\iota_{real} = \iota - \frac{\tilde{\iota}}{2m}. \tag{4.20}$$

And the real rate in a double-parity-check structure ($m \times q$ streams) is

$$\iota_{real} = \iota - \frac{(m+q-1)}{2mq}\tilde{\iota}. \tag{4.21}$$

Figure 4-2 shows the rate loss and the Shannon limit left-shift due to the redundancy at a spectral efficiency of 6 bits/T (i.e., $k = 6$) for a parity-concatenated 2-D trellis code. Here we suppose m is equal to q in the double-parity-check structure, and two coded bits (i.e., $\tilde{k} = 2$) of each symbol are protected.

From Fig. 4-2, we can see that the rate loss and the Shannon limit left-shift are substantially increased when m and/or q is decreased. In a single-parity-check structure, if m is small, for example $m = 5$, then the rate reduction is large (20%). If m is large, say $m = 40$, then the rate reduction is small (2.5%). However, we note that the accuracy of *extrinsic* metric $\lambda_{j,i}^{(p)}$ is determined by the PC constraint $\widehat{W}$ which is updated by the decoded values of $(m-2)$ packets. If m is large, generally $\widehat{W}$ is more likely to be erroneous and consequently results in



(a) Single-parity-check structure (b) Double-parity-check structure

Figure 4-2: Rate loss and the Shannon limit left-shift for the single- and double parity-check structures at a spectral efficiency of 6 bits/T

error propagation. Therefore, the values of m and q should be properly selected to achieve a best BER performance in terms of approaching the Shannon limit.

Next we consider how to update the PC constraint $\widehat{W}$ in the IVA. Also the graphic interpretation of the IVA can be found in section 5.3.

B. Updating the parity-check constraint $\widehat{W}$

(a). In the IVA, the Viterbi decoder progresses over the trellis for a certain depth (i.e., truncation length or decoding depth), it then produces its decision results. Ideally, updating for one packet/stream should be based on the decoding history of the other packets/streams, namely, $\widehat{W}$ should be updated based on the newest decoded values of the other packets/streams. However, in the IVA for packet transmission with tail-biting techniques on parity-check structures, the decision results of any packet cannot be determined until the whole super packet is finished. Thus, for all packets, $\widehat{W}$ is updated only once in every iteration. We can see that the way $\widehat{W}$ is updated has little effect on the performance of the IVA.

(b). In the derivation of updated metric function used in the IVA, when the j^{th} packet is decoded, the metric function $\lambda^{(p)}_{j+1,i}$ of the $(j+1)^{th}$ packet is supposed

to be "combined" (added) with the $\lambda_{j,i}^{(v)}$ of the j^{th} packet. There is a drawback for such an operation, since we noted that the errors often occur consecutively in the Viterbi decoding for the trellis codes. That means the decoded errors often occur in the adjacent packets in the single- or double-parity-check structure, especially when the packet length is short. Therefore, if one packet is being decoded, and its adjacent packet is selected to "combine" with it, serious error propagation will possibly result if an error event goes through these two packets. To avoid this problem, we can randomly select a packet to be "combined" with the packet which is being decoded. Fig. 4-3 illustrates this random selection idea for the IVA on a single-parity-check structure (suppose the first packet is being decoded here). Through the simulations, we find that using the random selection can achieve more 0.1 dB–0.2 dB gain, and also make the convergence speed of the IVA faster as well.

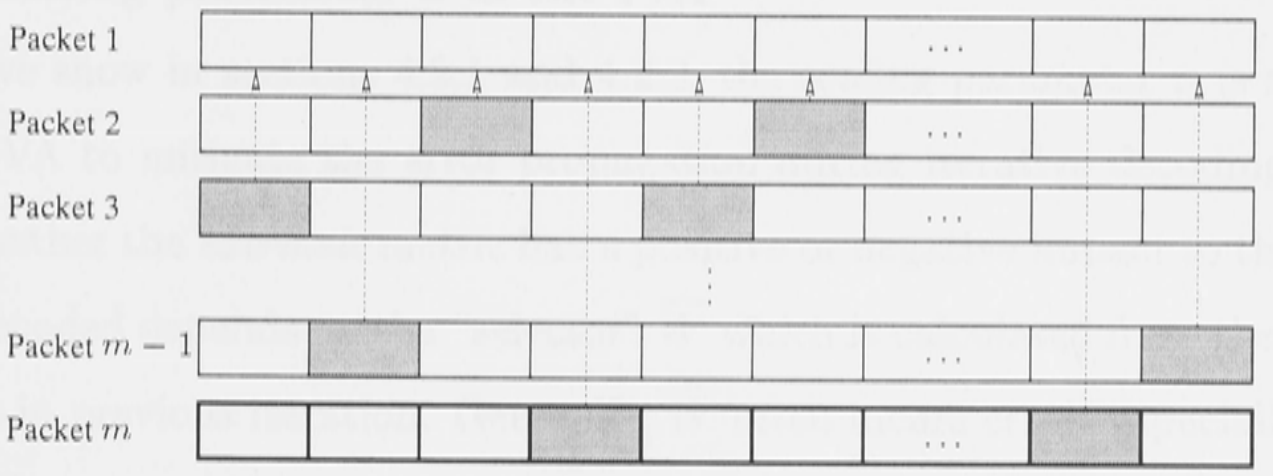


Figure 4-3: A simple case of random selection idea in the IVA on a single-parity-check structure

Therefore, based on the updated metric functions for parity-concatenated trellis codes, the corresponding IVA can be summarized as follows.

(The IVA for decoding the parity-concatenated trellis codes:)

step (a) In the first iteration, decode all the packets/streams in the single- or double-parity-check structures using the standard VA without consideration of the PC constraint among those packets/streams;

step (b) In the next iteration, first, update the PC constraint $\widehat{W}$ based on the

decoded symbol values. Here the random selection mentioned in Fig. 4-3 is employed;

step (c) Then, update the branch metric for each symbol which is the sum of original VA branch metric and *extrinsic* branch metric(s) "selected" by $\widehat{W}$;

step (d) Finally, decode all the packets using the Viterbi algorithm except that the updated branch metrics given in *step (c)* rather than original ones are employed;

step (e) Repeat *steps (b)*, *(c)* and *(d)* for several iterations until a stop criterion (we will discuss it in part **E**) is satisfied or a pre-set maximum number of iterations is reached.

C. Scaling parameter α in the IVA

As we show in sections 4.2.1 and 4.2.2, the scaling parameter α is necessary in the IVA to mitigate the error propagation during iterative decoding. In the IVA, whether the *extrinsic* metric has a positive or negative impact to the symbol being decoded depends on the "selector" $\widehat{W}$ which is calculated from the decoded symbols in previous iteration. Generally, $\widehat{W}$ often incurs errors especially in first several iterations. Therefore, scaling down the fed-back information is essential to mitigate the negative impact. Otherwise serious error propagation will result in. However, we noticed that the positive impact is also reduced by the operation of scaling down at the same time. If α is too small, little positive information is fed back to assist the decoder to improve the performance. Therefore, the scaling parameter α should be elaborately selected. It is a hard task to determine α via mathematical derivation. In section 6.1, we will use an experimental method to roughly select α for obtaining the good performance of the IVA.

D. Multistage decoding for multilevel codes

As showed in section 3.3.4, a BCH block code is combined with the parity-concatenated trellis codes to deal with the parallel transition errors (i.e., the errors

caused by uncoded bits). In the receiver, the received signals are correspondingly decoded using multistage decoding [47]. A block diagram of the decoder structure is given in Fig. 4-4. The iterative decoder in each symbol interval decides upon the correct cosubsets of Λ_{n_c}. Using this decision, the BCH decoder in every k_b symbol intervals finds the correct cosubsets of Λ_{n_c+1}. Finally, the shaping decoder and uncoded bits decoder decide on the parallel transition bits based on the result of BCH decoder.

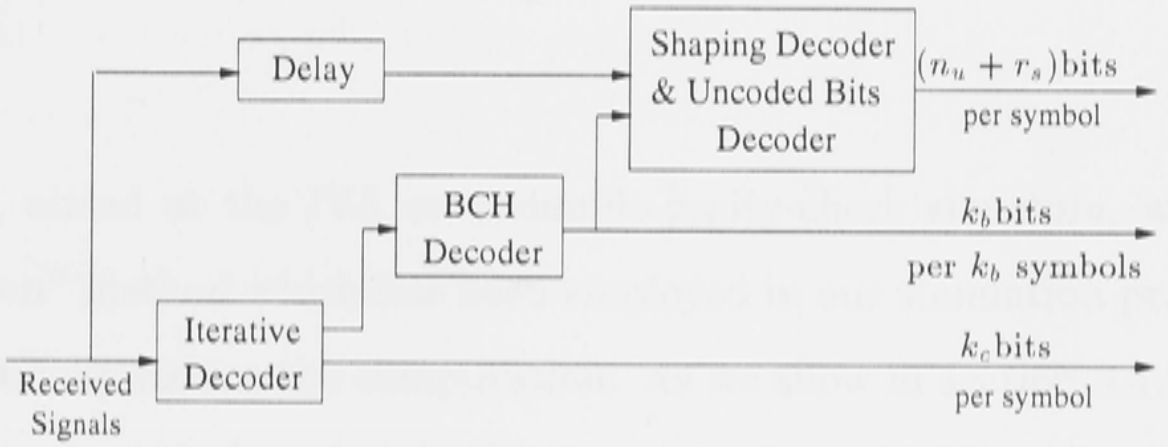


Figure 4-4: Multistage decoder of multilevel concatenated codes

E. Stop criteria in the IVA

The stop criteria in the IVA is quite straightforward. If the updated PC constraint $\widehat{W}$ satisfies $\widehat{W}_{m,i} = \sum_{j=1}^{m} \widehat{V}_{j,i}^{(p)} = 0$ $(i = 1, \cdots, l)$ (for the single-parity-check structure) or $\widehat{W}_{m,q,i} = \sum_{y=1}^{q} \sum_{j=1}^{m} \widehat{V}_{y,j,i}^{(p)} = 0$ (for the double-parity-check structure), then the iterative decoding process will be terminated. However, in some cases, we can see that the decoding process is stopped with coded bit errors. This phenomenon is due to the same error events appearing at the same place in the even-number packets (see Fig. 4-5), especially for the small constraint length trellis codes. Obviously, this kind of error is more prone to occur in the single-parity-check structure than double-parity-check structure. Therefore, from this point of view, the codes on the double-parity-check structure is more robust than the codes on the single-parity-check structure.

F. "Pre-decision" method used in the IVA on double-parity-check structure

$$
\begin{array}{ccccccc}
0 & 0 & & \cdots & & 0 & 0 \\
0 & & & & & & 0 \\
 & & 1 & \cdots & 1 & & \\
\vdots & & \vdots & & \vdots & & \vdots \\
 & & 1 & \cdots & 1 & & \\
0 & & & & & & 0 \\
0 & 0 & & \cdots & & 0 & 0
\end{array}
$$

Figure 4-5: An error pattern of block interleaver in double-parity-check structure

Finally, aimed at the IVA on a double-parity-check structure, we discuss a "pre-decision" method which has been employed in our simulation programs and can dramatically reduce the computation. As we show in section 3.4.2, there are horizontal and vertical parity-check constraints in a double-parity-check structure. During the iterative decoding process, if a super packet (including m packets for packet transmission) or a horizontal (or vertical) plane of streams (including m(or q) streams for continuous transmission) satisfy its own PC constraint, this super packet or horizontal (vertical) plane of streams will be excluded from the decoding during next iteration because the decoder "thinks" the errors in those packets or streams have been successfully wiped out with high possibility. Obviously, such a "pre-decision" can make the computation of iterative decoding cut down significantly. The other benefit is to avoid those packets/streams in which the errors have already been removed being involved in errors again due to occasional error propagation.

The "pre-decision" will be continued until all the super packets or horizontal (or vertical) planes of streams have been excluded from the decoding process. At this stage, if all the errors in a double-parity-check structure have really been corrected, the iterative decoding will be terminated in terms of the stop criteria $\widehat{W}_{m,q,i} = \sum_{y=1}^{q} \sum_{j=1}^{m} \widehat{V}_{y,j,i}^{(p)} = 0$. But sometimes, the decoding process will be continued because the PC constraint is satisfied in only one direction (horizontal or vertical) and failed in other direction. Since the errors cannot be located from

the other direction, we have to re-decode all the packets no matter whether the errors in them have been really corrected or not. Generally the remaining errors can be corrected quickly since the positive information is dominated in feedback at this stage.

4.3 Discussion

In this chapter, the iterative Viterbi algorithm was presented for decoding the parity-concatenated 2-D or M-D Wei trellis codes. The new branch metric functions used in the IVA were derived using some simplifying assumptions. We can see that the IVA decoder is identical to the standard VA except that the branch metric functions are updated according to the decisions of the Viterbi decoder in previous iterations, and the remaining parts are exactly same. If all the branch metrics of symbols in a single- or double-parity-check structure are able to be saved in the memory during the first iteration, then only few calculations are needed for updating new branch metrics used in next iteration. In addition, the computations of the IVA on a double-parity-check structure can be further reduced through a "pre-decision" method which excludes some packets satisfying parity-check constraint being involved in the next iterative decoding process. For the parity-concatenated M-D trellis codes, due to a nonlinear operation in the encoding process, some extra effort, but with negligible additional computation has to be made in the IVA.

Some relevant issues on the IVA have been discussed. Here we emphasised following three points: *(a)* the number of packet(s)/stream(s) (i.e., m and/or q) should be properly selected to minimize the gap between the decoding performance and the Shannon limit; *(b)* during the process of updating branch metric, we should randomly choose one symbol to feedback its *extrinsic* information; *(c)* the scaling parameter α used in the IVA should be carefully determined. Both over-setting and under-setting scaling factor could degrade the error performance as well as convergence speed of the IVA.

Chapter 5

Graph-Based Iterative Decoding Algorithms

In this chapter, the conventional iterative two-way decoding algorithms (ITWA) [38] will be first reviewed. Then we will show that for parity-concatenated trellis codes the two-way algorithms need to be modified in order to achieve a better performance. After that, we will give a graph interpretation of the IVA. Lastly, five iterative decoding algorithms will be proposed and their computational complexity are compared.

5.1 Conventional Two-Way Algorithms

It is convenient to represent the error control codes and describe the decoding algorithms using the language of graphs. In [38], Forney summarized the two-way algorithms (TWAs) which were applicable to decode the codes defined on general TWL graphs. It is shown that, with the TWL graph, all the standard decoding algorithms for turbo codes, LDPC codes, and other compound codes can be classified as one of two types of algorithms: *min-sum* and *sum-product* algorithms. In [38][77], these two algorithms have been explicitly described through a simple binary code case. In this section, we will first introduce the min-sum and sum-product algorithms for trellis codes, and then show how the iterative two-way

algorithms (include iterative min-sum and sum-product algorithms) work on the parity-concatenated TCM scheme based on its TWL graph representation.

5.1.1 The min-sum algorithm

In the TWL representation of trellis codes, each possible value x_i of each coded symbol node V_i can be assigned a weight $\omega_i(x_i)$; e.g., a log likelihood weight

$$\omega_i(x_i) = -\log P(R_i|V_i = x_i), \tag{5.1}$$

where R_i is the received signal corresponding to the coded symbol V_i, and $P(\cdot)$ denotes the probability density function. The total weight $\omega(\mathbf{V})$ of a valid configuration (codeword) $\mathbf{V}$ is the sum of its symbol weights:

$$\omega(\mathbf{V}) = \sum_i \omega_i(V_i). \tag{5.2}$$

The min-sum algorithm is then used to find the most likelihood decision on the codeword $\mathbf{V}$, i.e.,

$$\widehat{\mathbf{V}} = \min\{\omega(\mathbf{V})|\mathbf{V} \in \mathbf{C}\}, \tag{5.3}$$

where $\mathbf{C}$ represents the overall possible codewords, $\omega(\mathbf{V})$ are the log likelihood weights $-\log P(\mathbf{R}/\mathbf{V})$, $\mathbf{R}$ denotes the received codeword corresponding to the codeword $\mathbf{V}$. For a cycle-free graph, if we cut through any edge, the graph becomes two disjoint parts, say "upstream" and "downstream". The weigh $\omega(\mathbf{V})$ of any codeword $\mathbf{V}$ may correspondingly be expressed as the sum $\omega(\mathbf{V}) = \omega(\mathbf{V}^+) + \omega(\mathbf{V}^-)$ of the upstream and downstream weights $\omega(\mathbf{V}^+)$ and $\omega(\mathbf{V}^-)$, respectively. The upstream and downstream weights can be computed separately by the min-sum algorithm (whence the name "two-way algorithm" [38]).

There is an important issue that should be mentioned here. It is about updating order of the graph. For a cycle-free graph, the flooding schedule and sequential updating schedule [38] are the two alternative methods. In the flooding schedule, each symbol node and check node are allocated one processing unit, and then all computations are simultaneously performed for all values of all edges at all

times, based on the currently available inputs. This approach may be attractive in hardware. But for software implementation, the sequential updating schedule is possibly a more natural choice. With the sequential updating order, the min-sum algorithm for a finite cycle-free graph can be summarized as follows [38].

(**The min-sum algorithm for a finite cycle-free graph:**)

step (a) Start at the "leaf" nodes, which are connected to the graph via a single edge; the upstream values of the edge connected to the leaf node are simply the values of that leaf node.

step (b) At each interior node, update the weights of the values of each outgoing edge as soon as all incoming weights are known. The *update rules* are as follows.

(1) For an edge downstream of a symbol or state node, for each possible value x_i, the outgoing weight is simply the sum of the incoming weights into the node plus the local weight of the node itself.

(2) For an edge downstream of a parity check node, for each possible value x_i, the outgoing weight is the minimum of the sum of the incoming weights over the set of all incoming configurations consistent with x_i.

step (c) Repeat *step (b)* until reaching every leaf node.

step (d) The sum of the upstream and downstream weights gives the final weights of each value of each edge, and thus of its associated symbol node.

When the min-sum algorithm is applied to a graph corresponding to a finite-length trellis (see Fig. 3-16 (b)), the natural updating schedule is started first from left to right and then from right to left, which yields the *backward-forward algorithm* (BFA) [38]. Furthermore, if the computation in one direction is replaced by a simpler backtracking procedure, the standard Viterbi algorithm will result. It is clear that the complexity of the BFA is approximately twice the VA decoding complexity for the same trellis, but the BFA can deliver the soft-output information for each symbol.

In practical implementation, it is important to handle numerical issues properly. Typically, the incoming or outgoing values of a node may grow out of range quickly. To overcome this, it is necessary to include an arbitrary normalization factor in the updating process. Since the min-sum algorithm only involves addition and minimization, the normalization factor does not influence the final performance.

5.1.2 The sum-product algorithm

The sum-product algorithm is exactly the same as the min-sum algorithm, except that the "min" is replaced by "sum", and the "sum" is replaced by "product" and the weights are replaced by $P(\mathbf{R}/\mathbf{V})$. In particular, the update rules are as follows.

(1) For an edge downstream of a symbol or state node, for each possible value x_i, the outgoing weight is simply the product of the incoming weights into the node plus the local weight of the node itself.

(2) For an edge downstream of a parity check node, for each possible value x_i, the outgoing weight is the sum of the product of all incoming weights over the set of all incoming configurations consistent with x_i.

When the weights are proportional to likelihoods, the sum-product algorithm is alternatively called the APP (a *posteriori* probability) decoding algorithm. When and only when the decoding algorithm goes on to make symbol decisions based on the maximum APP, the MAP (maximum a *posteriori*) decoding algorithm results [38]. Compared with the VA which provides optimum performance in the sense of minimizing the sequence error probability, the MAP algorithm is optimal in the sense of minimizing the probability of a symbol error.

The properties of the sum-product algorithm are just the same as the properties of the min-sum algorithm. In particular, it converges in the same time on a finite cycle-free graph, and the number of computations is the same [38]. But the complexity of sum-product is a bit of higher due to the requirement for multiplications.

The problem of numerical overflow also exists in the sum-product algorithm. Therefore, an arbitrary normalization factor should be included in the decoding process.

5.1.3 Iterative two-way algorithms (ITWAs) for parity concatenated trellis codes

Based on the TWL graph representation of the parity-concatenated trellis codes, the ITWA is a straightforward solution. Fig. 5-1 shows a TWL graph corresponding to the codes based on a single-parity-check structure without tail-biting, in which $m = 3$ and $l = 4$. The definitions of four types of node appearing in Fig. 5-1 have been explained in section 3.5.

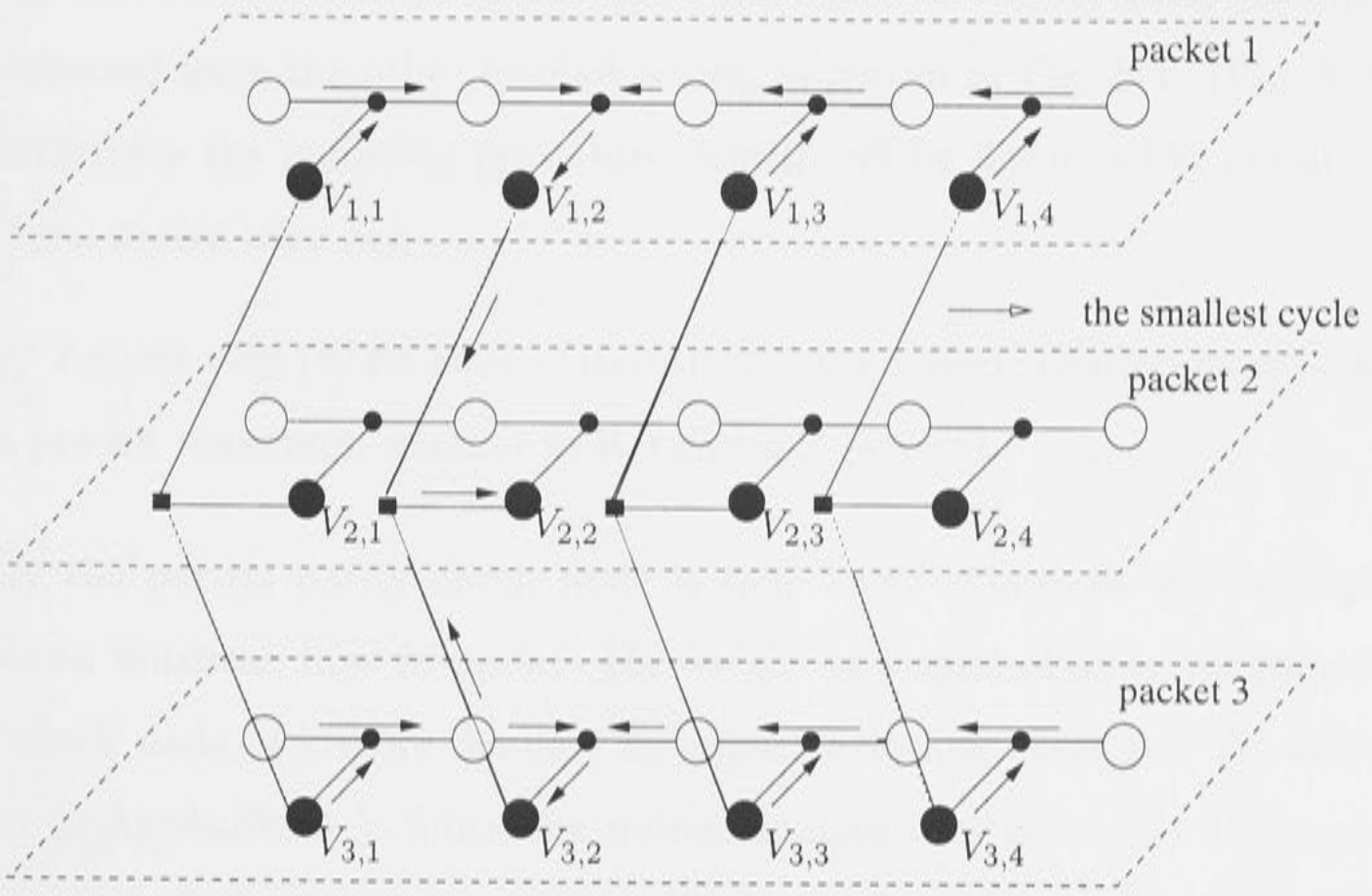


Figure 5-1: TWL graph representation for the parity-concatenated trellis codes on a single-parity-check structure with $m = 3$ and $l = 4$

Obviously, the TWL graphs corresponding to each trellis diagram are cycle-free. But after concatenating them via partial parity check nodes, there are many the smallest cycles in the TWL graph (see Fig. 5-1). Suppose the coded symbol $V_{2,2}$ will be decoded, then all the *extrinsic* information about $V_{2,2}$, spread

throughout the symbol nodes in other packets, is "collected" through the two-way algorithm. The arrows shown in Fig. 5-1 illustrate how the *extrinsic* information flows to the coded symbol $V_{2,2}$. The conventional ITWAs for decoding the parity-concatenated trellis codes are summarized as follows.

(The conventional ITWAs for parity-concatenated trellis codes:)

step (a) In the first iteration, cut all partial parity check nodes and apply the min-sum (or sum-product) algorithm to an "unwrapped" trellis[1] [41][60] (an example is shown in Fig. 5-2). The weight of each possible value of symbol node is then obtained;

step (b) In the next iteration, first update the weight of each symbol node using the min-sum (or sum-product) algorithm based on the *extrinsic* information collected from the other symbol nodes, as shown in Fig. 5-1. (Fig. 5-3 (b) illustrates the updating procedure, which will be discussed in detail late.) Then repeat *step (a)*;

step (c) Repeat *step (b)* for several iterations until a stop criterion is satisfied or a pre-set maximum number of iterations is reached.

Only the partial parity check node is new here. Thus, we go through an example to illustrate how to update the weight of a symbol node via its partial parity check node. (A more detailed example, including *steps (a)*, *(b)* and *(c)*, is given in Appendix A.) Since the min-sum algorithm is exactly the same as the sum-product algorithm, except that the "min-of-sum" is replaced by a "sum-of-product" operation, here we just use the min-sum algorithm to illustrate the updating procedure.

Consider a parity-concatenated trellis code defined in Fig. 5-1, i.e., $m = 3$. Assume that 2 bits of each symbol are inputted into a linear rate 1/2, 4-state

[1]This is for the parity-concatenated trellis codes based on a single-parity-check structure with tail-biting (see Fig. 3-13). For the codes defined in Fig. 3-11, the "unwrapped" packet, as shown in Fig. 3-12, is employed.

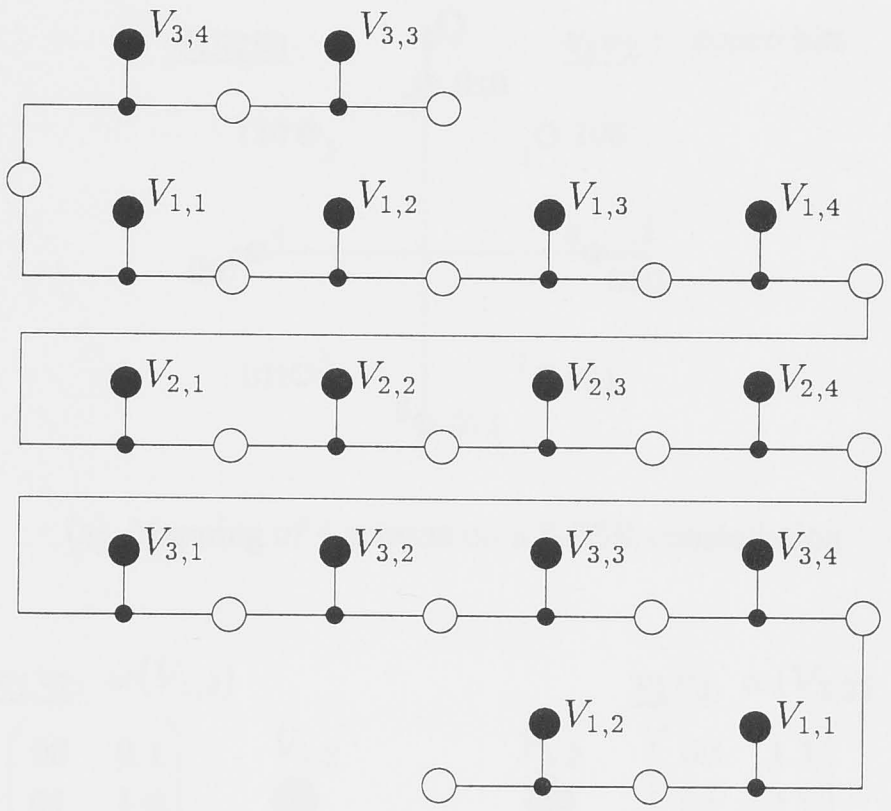


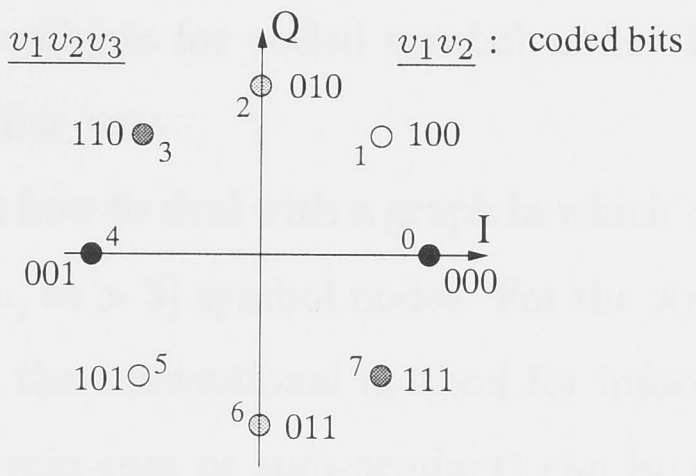
Figure 5-2: An "unwrapped" trellis for a single-parity-check structure with tail-biting

trellis encoder (i.e., $k = 2$, $\tilde{k} = 1$ and $\nu = 2$), and the output symbols are mapped into an 8-PSK constellation (see Fig. 5-3 (a)). Three coded symbol nodes $V_{1,2}$, $V_{2,2}$ and $V_{3,2}$ are connected via a partial parity check node. Therefore, each coded symbol node V consists of four possible values 00, 01, 10 and 11. In the receiver, the local weights $\omega(V)$ of symbol V, i.e., $-\log P(R/V)$ can be obtained by selecting the smallest weight in each of four subsets. Suppose that $\omega(V_{1,2}) =$ (0.1, 1.0, 3.0, 1.3), $\omega(V_{2,2}) = (1.3,\ 0.1,\ 1.0,\ 3.0)$ and $\omega(V_{3,2}) = (1.3,\ 0.1,\ 1.0,\ 3.0)$ for four subsets 00, 01, 10 and 11, respectively, as shown in Fig. 5-3 (b). Now let us focus on updating the weights for symbol node $V_{2,2}$.

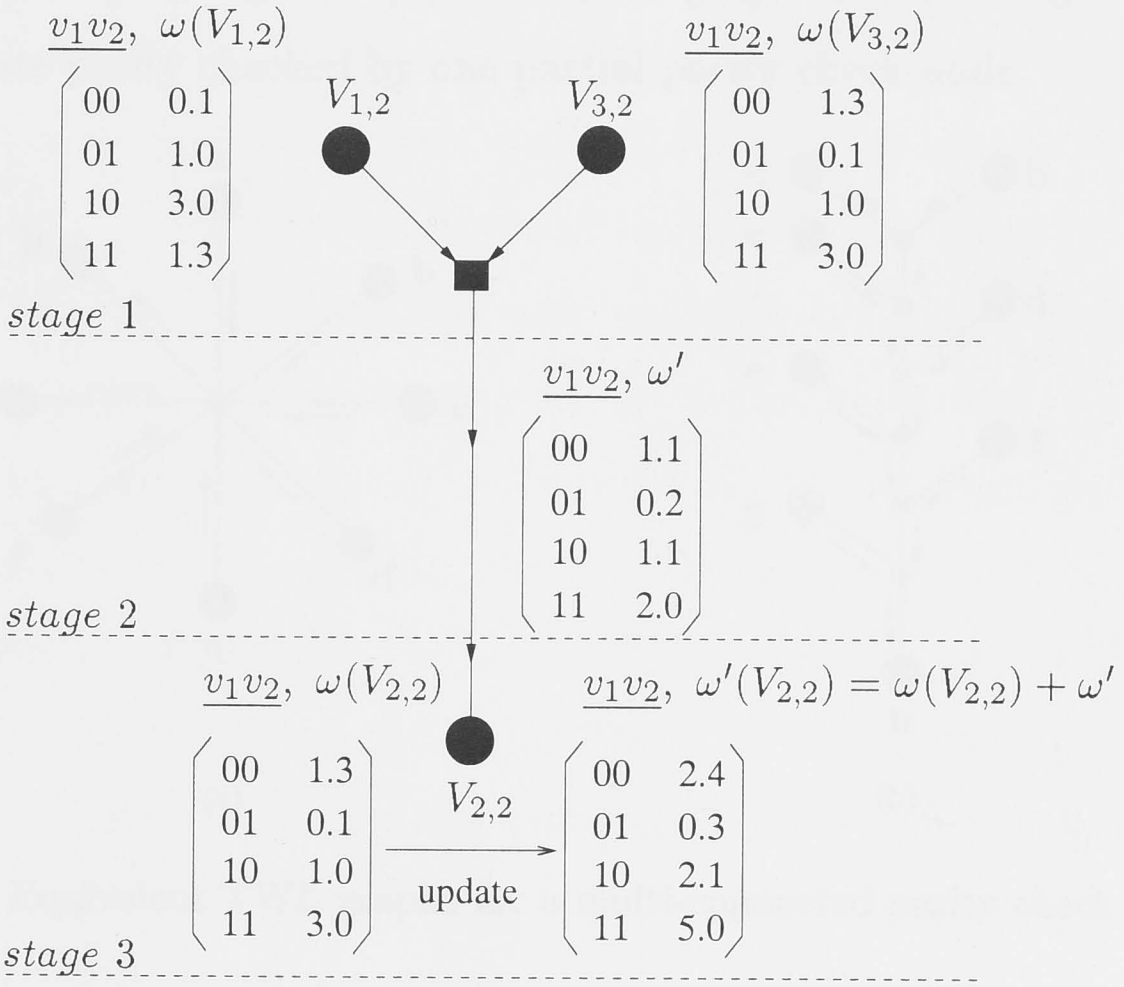
According to Fig. 5-3 (b), the updating procedure can be divided into three stages.

stage 1 List all subsets and their corresponding weights for all starting symbol nodes, i.e., $V_{1,2}$ and $V_{3,2}$. Then simply transfer their weights to the neighbouring edge;

stage 2 The weight for every subset after the partial parity check node is computed via the min-sum algorithm. For example, for subset $v_1 v_2 = 01$, we



(a) Mapping of 4 subsets on a 8-PSK constellation



(b) Update the weight for $V_{2,2}$

Figure 5-3: Illustration of the updating procedure for the min-sum algorithm on a finite cycle-free graph

have $\omega' = \min\{0.1 + 0.1,\ 1.0 + 1.3,\ 3.0 + 3.0,\ 1.3 + 1.0\} = 0.2$;

stage 3 List all subsets and their corresponding weights for updating symbol node, i.e, $V_{2,2}$. Then, the updated weight $\omega'(V_{2,2})$ for each subset is equal to the sum of its local weight $\omega(V_{2,2})$ and the *extrinsic* weight ω'. For example, for subset $v_1v_2 = 00$, $\omega'(V_{2,2}) = 1.3 + 1.1 = 2.4$.

Clearly, the updated weights for coded symbol nodes $V_{1,2}$ and $V_{3,2}$ can be obtained through the similar way.

There is an issue about how to deal with a graph in which a partial parity check node combines several (i.e., $m > 3$) symbol nodes. For the structure illustrated in Fig. 5-1, in which $m = 3$, the conventional method for information transmission discussed in [38][77] (i.e., min-sum or sum-product) can be employed. However, if $m > 3$, then the conventional way will not be suitable due to high complexity involved. For example, Fig. 5-4 (a) displays a graph in which eight $(m = 8)$ symbol nodes are parity checked by one partial parity check node.

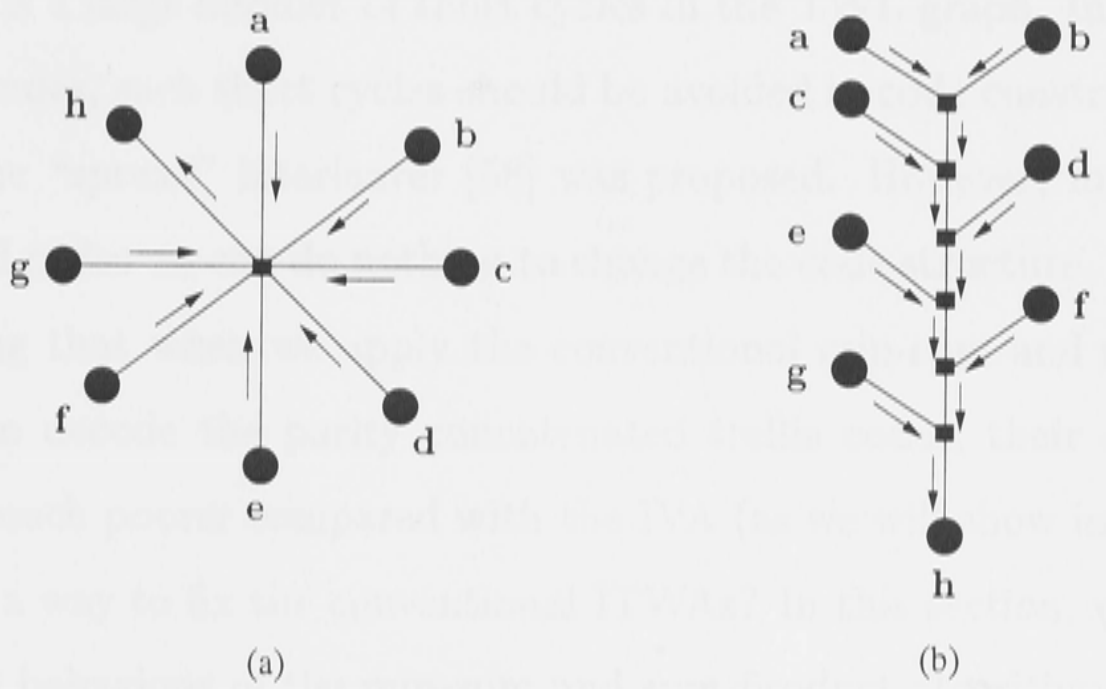


Figure 5-4: Equivalent TWL graphs for a multi-connected parity check node

Suppose the information on symbol nodes **a**, **b**, $\cdots$, **g** is passed to symbol node **h** simultaneously. Directly calculating the weights of each symbol node is quite computation expensive if based on the structure of Fig. 5-4 (a). Note that the local weights of parity check nodes are assumed to be zero. Therefore, a variant of the graph in Fig. 5-4 (a) is shown in Fig. 5-4 (b) which does the same work as Fig. 5-4 (a) but has much lower complexity. (It is clear that the complexity of Fig. 5-4 (b) is proportional to the number and size of symbol nodes. With regard to Fig. 5-4 (a), however, the complexity is exponentially increased with the number and size of symbol nodes.) The min-sum algorithm is then be able to be conducted in each parity check node. For this case, if each symbol

node represents a four-symbol space (00,01,10 and 11), we can see that 6×16 addition operations and 6×12 comparison-selection operations are required if the information on the symbol nodes **a**, **b**,$\cdots$,**g** is passed to the symbol node **h**.

5.2 Modified Iterative Two-Way Algorithms

The two-way algorithms in section 5.1.3 are sub-optimal. But for many codes such as turbo codes and LDPC codes without small cycles, their performance can closely approach the Shannon limit. The key problem for parity-concatenated trellis codes is a large number of short cycles in the TWL graph. In turbo codes and LDPC codes, such short cycles should be avoided in code construction. Consequently, the "spread" interleaver [58] was proposed. However, for the parity-concatenated codes we can do nothing to change the code structure. Thus, it was not surprising that when we apply the conventional min-sum and sum-product algorithms to decode the parity concatenated trellis codes, their error performances are much poorer compared with the IVA (as we will show in Chapter 6). Can we find a way to fix the conventional ITWAs? In this section, we will study asymptotical behaviour of the min-sum and sum-product algorithms, which will indicate how to fix the algorithms. Similar analysis for the parity-concatenated convolutional codes can be found in [62].

When the SNR is large, error events with minimum free distance will commonly dominate the error performance. Here, the error event is defined in the traditional way such that it starts when the error path differs from the correct path and ends when the two paths agree again for the first time. Now let us study the ML and APP decoders for a simple parity-concatenated code which involves a single error event.

Proposition 1 Considering a simple parity-concatenated code given in Fig. 5-5, where all m packets transmit either paths 1 or 2 of an identical error event and the number of packets selecting path 2 is even, we can then construct a ML or APP decoder as follows.

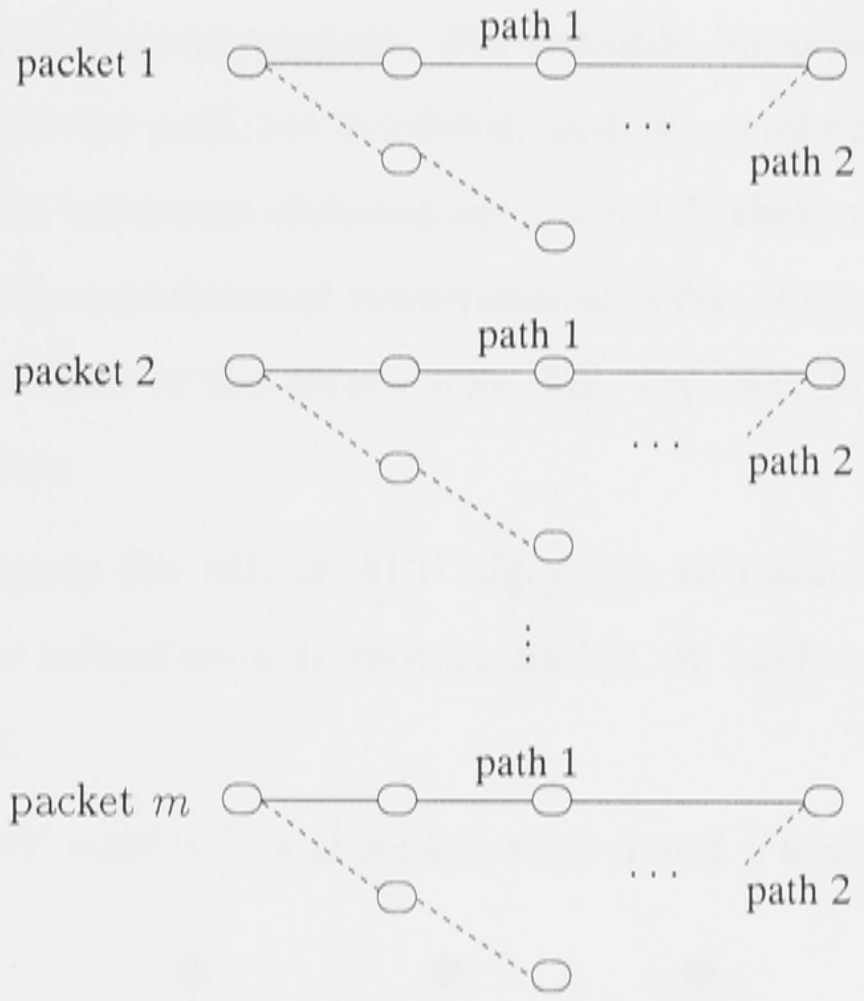


Figure 5-5: A simple parity-concatenated code

Step 1 In the first iteration, apply the ML or APP algorithm to compute the metric (which is related to the ML value) of two paths in each packet;

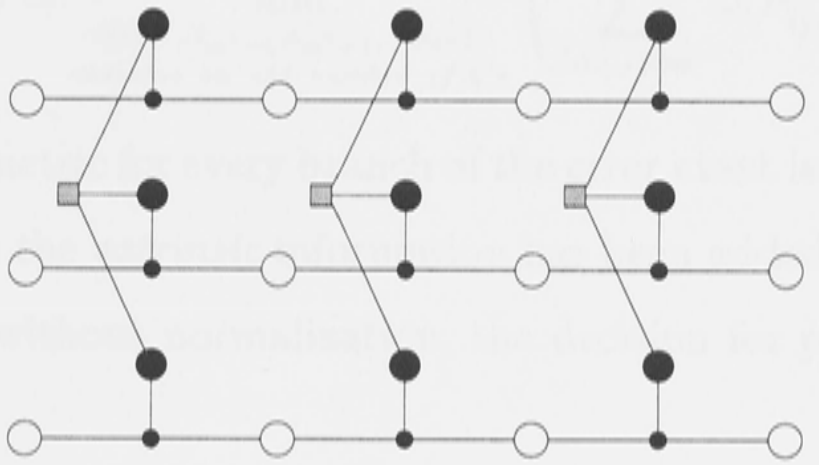
Step 2 Set $m' = 1$;

Step 3 All packets other than packet m' will contribute their *extrinsic* information - the ML metrics or APP values obtained in *step 1* - to packet m'. The *extrinsic* information is computed using the same procedure as in the conventional min-sum or sum-product algorithm, except that the *extrinsic* ML metrics have to be scaled down by a factor of $1/d$ or the *extrinsic* APP values have to pass through a d-th root device before being sent to packet m', where d is the number of time units in which the symbol in the error path is different from the symbol of the correct path. In other words, d is the length of the error event with minimum distance minus the number of symbols in the error path which

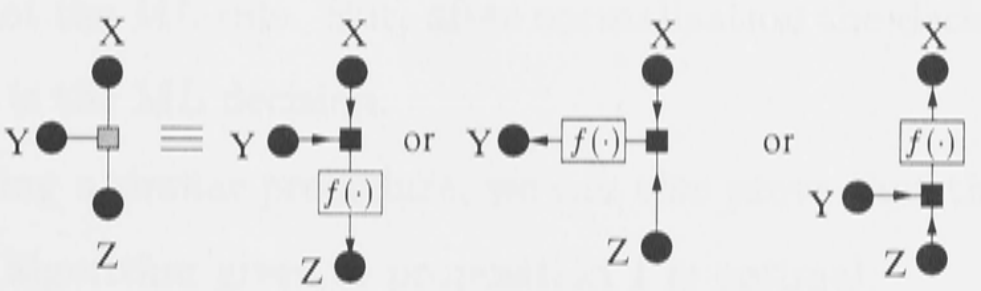
agree with the correct path. For example, suppose the symbols in the correct path are 0-0-0-0-0, and the symbols in the error path with minimum distance are 1-0-4-1-2, then $d = 5 - 1 = 4$. For parity-concatenated convolutional codes, d equals the Hamming distance of the error event [62]. Fig. 5-6 illustrates these operations;

Step 4 Re-apply the ML or APP algorithm to packet m' after the *extrinsic* information is sent to packet m' and make the final decision;

Step 5 Set $m' = m' + 1$, and repeat *steps 3* and *4* until $m' = m$.



special parity check node with normalization



$$f(x) = (x)^{1/d} \qquad \text{for APP}$$
$$f(x) = x/d \qquad \text{for ML}$$

Figure 5-6: A TWL graph with information normalization

Proof 1 Delete the time units in which two paths have the same symbols, since they have no contribution to the decision. Let $\omega(P_i^j)$

denote the metric of path i in packet j, where $\omega(P_i^j) = \sum_{t=1}^{l} \left(R_t^j - s_{t,i}^j\right)^2$, $\mathbf{R}^j = [R_1^j, ..., R_l^j]$ denotes the received signal sequence of packet j, $\mathbf{s}_i^j = [s_{1,i}^j, ..., s_{l,i}^j]$ denotes the transmitted signal sequence of path i in packet j. Then the ML decoder for packet m' is

$$\widehat{i}_{m'} = \arg\min_{i\in 1,2} \left(\omega(P_i^{m'}) + A_{i,m'}\right), \tag{5.4}$$

where

$$A_{1,m'} = \min_{\substack{\{i_1,...,i_{m'-1},i_{m'+1},...,i_m\} \\ contains\ an\ even\ number\ of\ 2's}} \left(\sum_{j=1,j\neq m'}^{m} \omega(P_{i_j}^j)\right), \tag{5.5}$$

$$A_{2,m'} = \min_{\substack{\{i_1,...,i_{m'-1},i_{m'+1},...,i_m\} \\ contains\ an\ odd\ number\ of\ 2's}} \left(\sum_{j=1,j\neq m'}^{m} \omega(P_{i_j}^j)\right). \tag{5.6}$$

The extra metric for every branch of the error event is $\{A_{1,m'}, A_{2,m'}\}$. For d branches the *extrinsic* information has been added to packet m' d times; thus without normalization, the decision for packet m' will be

$$\hat{i}_{m'} = \arg\min_{i\in 1,2} \left(\omega(P_i^{m'}) + dA_{i,m'}\right), \tag{5.7}$$

which is not the ML rule. But, after normalization the decision of the algorithm is the ML decision.

Following a similar procedure, we can also prove that the normalized APP algorithm given in proposition 1 is optimal.

There are three key differences between the conventional min-sum or sum-product algorithm and those given in proposition 1: (1) using a normalization function; (2) only two iterations are required; (3) *extrinsic* information is based on the first iteration decoding, i.e., in *step 4*, the *extrinsic* information is not updated. In section 6.3 we will show that the first modification (i.e., normalization) is very important for near optimal iterative decoding. Thus, for a TWL graph containing many short cycles, we can also achieve near optimal iterative decoding using the min-sum/sum-product algorithms, except that the *extrinsic* information in the

short cycles has to be normalised. We will also show that asymptotically the modified min-sum/sum-product algorithms only need two iterations, which is a very surprising result. Consequently, we consider the iterative min-sum and sum-product algorithms with the information normalization in the rest of the thesis.

If the ITWAs are applied to decode the parity-concatenated 4-D trellis codes, the procedure is similar with that of decoding the parity-concatenated 2-D trellis codes. The only difference is that the symbol nodes in TWL graph are treated as 4-D symbols which consists of two 2-D symbols, and hence the computational complexity will be dramatically increased due to the high diversity of the coded symbol. (This is because the PC constraint should be considered only based on the 4-D symbols, and cannot be separately considered based on two 2-D symbols.) For the 4-D 16-state Wei code, each symbol node contains 16 possible combinations (values) of $Y0^t$, $I1_i^t$, $I2_i^{t'}$ and $I3_i^{t'}$. For the 4-D 64-state Wei code, 64 possible combinations are included in each symbol node.

5.3 A Special Case of ITWA — IVA

For a TWL graph with cycles, the conventional two-way algorithms are not optimal. In order to obtain the optimal min-sum and sum-product algorithms, we have to convert graphs with cycles to graphs without cycles. For example, a single cycle graph, say a tail-biting trellis, can be decomposed into 2^ν cycle-free trellises. the optimal decoding algorithms are then obtained by applying the two-way algorithms to each of 2^ν cycle-free trellis [92]. Thus, the complexity of the optimal algorithms for a tail-biting trellis is 2^ν times the complexity for the cycle-free trellises. If a graph contains two cycles (say an ∞ length trellis), then it is easy to show that the complexity of the optimal decoders is about $(2^\nu)^2$ times of that for the cycle-free trellises [64]. A powerful code like Turbo codes or LDPC codes often contains a large number of cycles, thus the optimal decoder has a prohibitive large complexity. Consequently, the ITWA is often used as an alternative suboptimal decoding algorithm. If the girth of the cycle is large

enough (say larger than 5ν), then the error performance of ITWA is very close to the optimal decoders [53][60]. However, when a TWL graph has many small cycles, the ITWA often performs poorly. By modifying the ITWA to avoid the effect of small cycles, we might improve its performance. The IVA is one such a modification.

Actually, the IVA can be viewed as a simplification of the iterative min-sum algorithm. It is clear that the IVA follows the same procedure as the iterative min-sum algorithm, except that the VA during trellis decoding process is replaced by the min-sum algorithm, and the updating procedure (i.e., calculation of *extrinsic* information) is simplified significantly. Therefore, the IVA can also be interpreted through a TWL graph.

Fig. 5-7 shows the procedures in the IVA using the TWL representation. For the sake of simplicity, in this case $m = 3$. First, in Fig. 5-7 (a), the VA is applied in the trellis to get the estimated value $\widehat{V}_{j,2}$ $(j = 1, \cdots, 3)$ of each symbol node. In the next iteration, the PC constraint within the symbol nodes is considered, see Fig. 5-7 (b). In Fig. 5-7 (c), when decoding one symbol node (here suppose $V_{1,2}$), we cut down the connections between all other symbol nodes ($V_{2,2}$ and $V_{3,2}$) which are connected by their parity check nodes with the graph. In Fig. 5-7 (d), one of the symbol nodes is randomly selected (here suppose $V_{3,2}$), and then the PC constraint $\widehat{W}_{1,2}$ can be calculated based on the estimated values of the remaining symbol node $V_{2,2}$, i.e., $\widehat{W}_{1,2} = \widehat{V}_{2,2}$. Here $\widehat{W}_{1,2}$ can be viewed as a "selector" which chooses the appropriate local weights (branch metrics) of symbol node $V_{3,2}$ to pass through the partial parity check node to update the branch metrics of symbol node $V_{1,2}$.

To compare the IVA with the iterative min-sum algorithm, we go through the example given in Fig. 5-3 to illustrate how to update *extrinsic* information (i.e., branch metrics) in the IVA. Again, as shown in Fig. 5-8, we focus on updating the branch metrics for the symbol node $V_{2,2}$.

stage 1 Randomly select one symbol (say $V_{1,2}$) from the set comprised of all starting symbols, i.e., $V_{1,2}$ and $V_{3,2}$, in this example. List all subsets and

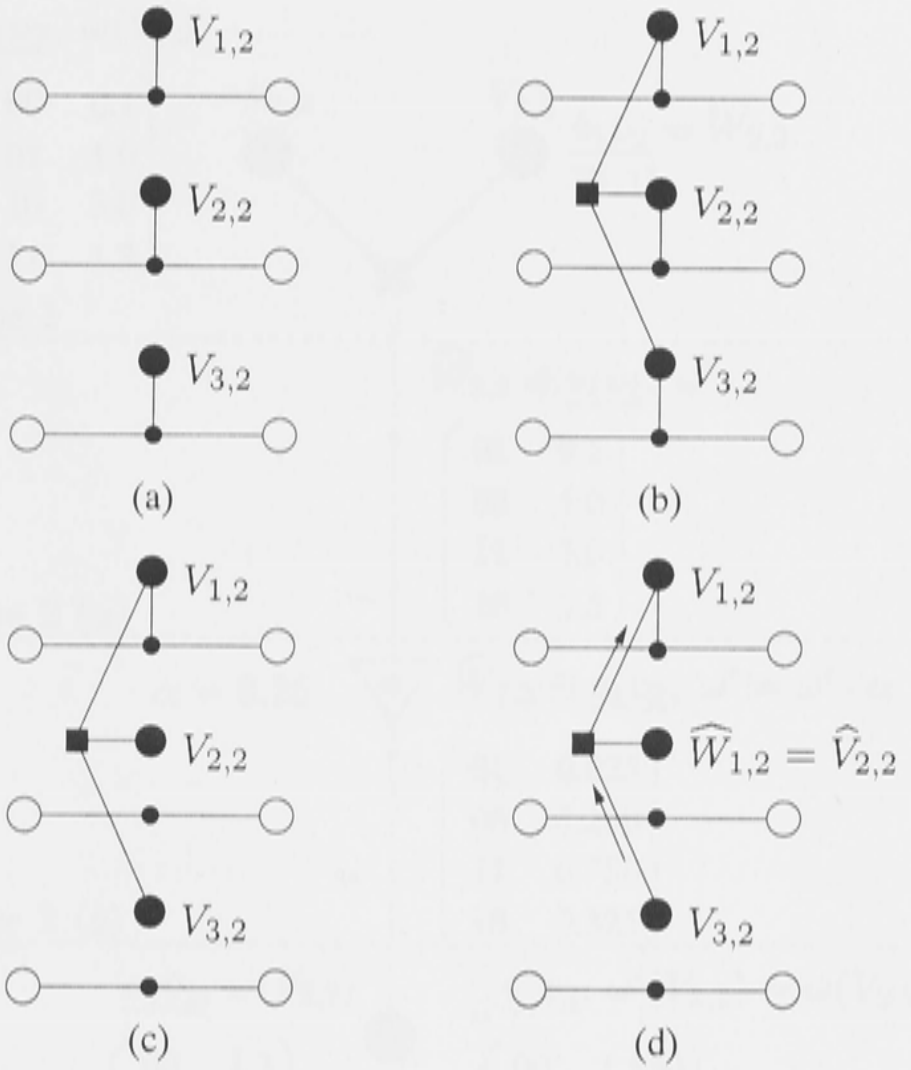


Figure 5-7: TWL graph interpretation for the IVA on a single-parity-check structure

their corresponding local weights (branch metrics) for this selected symbol. Computer the "selector" $\widehat{W}_{2,2}$ based on the VA decision of the rest of starting symbols. In this example, the decision for $V_{3,2}$ is $\hat{v}_1\hat{v}_2 = 01$ after the VA in previous iteration, and hence $\widehat{W}_{2,2} = 01$;

stage 2 There are two parts in *stage 2*. First, in *stage 2 (a)*, we calculate $\widehat{W}_{2,2} \oplus v_1v_2$ (of symbol $V_{1,2}$), and then copy the weights from the selected symbol $V_{1,2}$. For example, we have $\widehat{W}_{2,2} \oplus v_1v_2 = 01 \oplus 00 = 01$ and weight $\omega' = 0.1$ for the first row of the weight table in *stage 2 (a)*. In *stage 2 (b)*, the weights ω' are scaled down by a factor α (here suppose $\alpha = 0.25$), which is introduced to control error propagation;

stage 3 This stage is identical to *stage 3* in the min-sum algorithm (see section 5.1.3).

From Figures 5-7 and 5-8, we can see that in the IVA, only the branch metrics

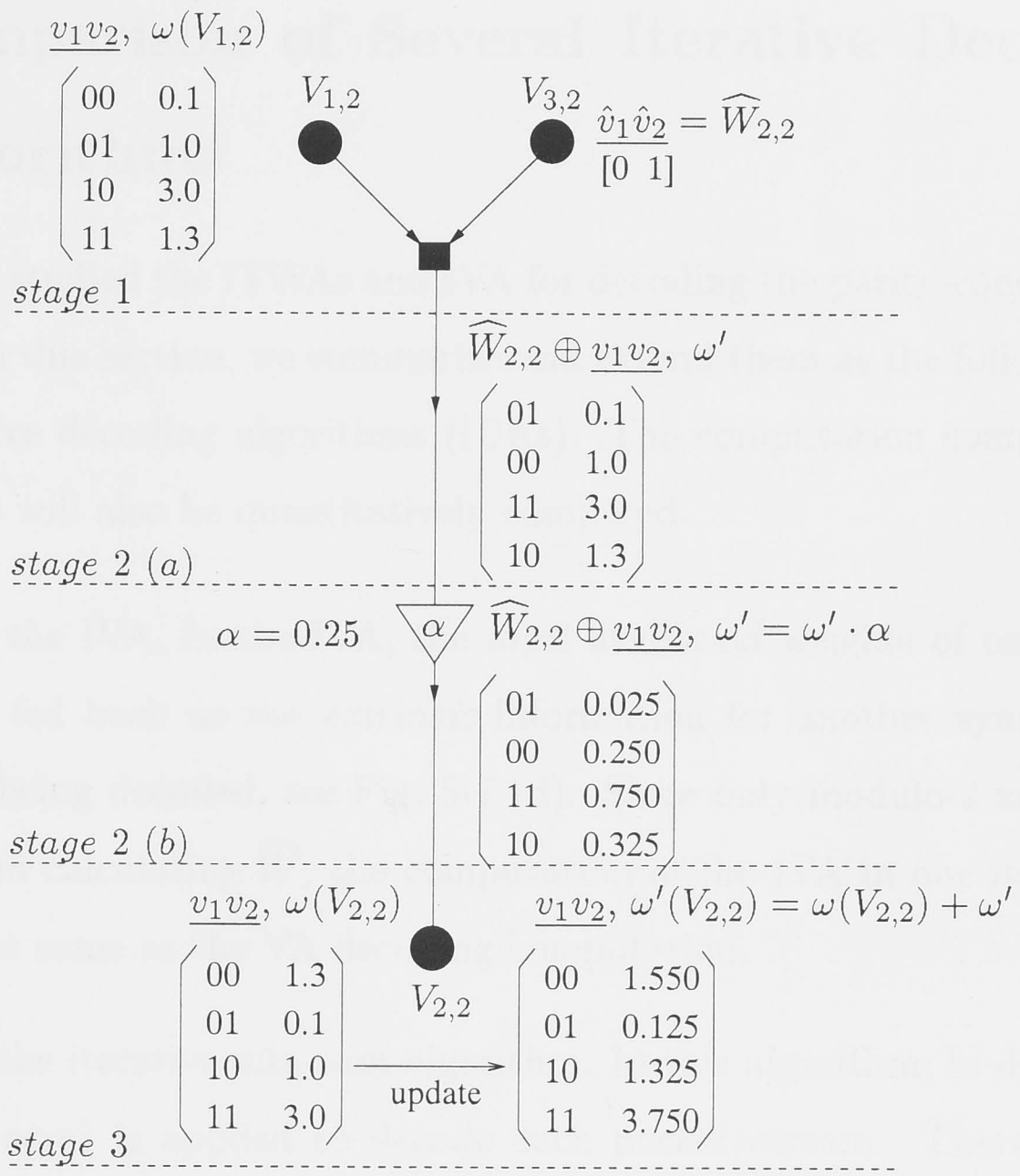


Figure 5-8: Illustration of the updating procedure for the IVA

(local weights) of one symbol controlled by the decisions of the other symbol values are transmitted to the symbol node which is being decoded. Unlike the iterative min-sum algorithm, the sum of upstream and downstream weights of one symbol are passed.

Finally, it is worth mentioning here that the issues **A**, **B**, **D**, **E** and **F** discussed in section 4.2.3 for the IVA are also applicable to the ITWAs.

5.4 Comparison of Several Iterative Decoding Algorithms

So far, we have studied the ITWAs and IVA for decoding the parity-concatenated trellis codes. In this section, we summarize and extend them as the following five types of iterative decoding algorithms (IDAs). The computation complexity of these five IDAs will also be quantitatively compared.

- *IDA 1* is the IVA. In the IVA, the local likelihood weights of one symbol node are fed back as the *extrinsic* information for another symbol node which is being decoded, see Fig. 5-7 (d). Since only modulo-2 addition is involved in calculating $\widehat{W}$, the computation of the IVA in one iteration is almost the same as the VA decoding computation.

- *IDA 2* is the iterative min-sum algorithm. In this algorithm, bi-directional VA (min-sum) is applied to decode each packet/stream. Therefore, the complexity of this part is approximately twice the VA decoding complexity. To obtain the *extrinsic* information, the weight ω (sum of the upstream weight ω^+ and downstream weight ω^-) of all the symbol nodes except one which is being decoded are concentrated on the parity check node using the min-sum algorithm, see Fig. 5-1. As we mentioned in section 5.1, the complexity of calculating ω is proportional to the number and size of the symbol nodes.

- *IDA 3* is the iterative sum-product algorithm, which is exactly the same as *IDA 2*, except the "min-sum" is replaced by the "sum-product". The number of computation of *IDA 3* is the same as that of *IDA 2*. But the complexity of *IDA 3* is higher due to the requirement for multiplication.

- *IDA 4* is the mixture of *IDA 1* with *IDA 2*. In *IDA 4*, the bi-directional VA is still used to decode each packet/stream. But the "selection" idea in the IVA is employed to obtain the *extrinsic* information which is the weight

ω rather that the local weight of one symbol node. It is clear that the computation complexity of *IDA 4* is approximately twice that of *IDA 1*.

- *IDA 5* is the mixture of *IDA 1* with *IDA 3*. We can see that *IDA 5* is the same as *IDA 4*, except the "sum-product" is applied on each packet/stream. Therefore, its complexity is a bit of higher than that of *IDA 4*.

We have described in section 4.1 that the procedure in the IDAs includes two parts: decoding the trellis codes and calculating the *extrinsic* information. In accordance with the single-parity-check structure (m packets $\times$ l symbols) defined in Fig. 3-5 (i.e., without tail-biting) and 2-D trellis codes described in section 3.3.1, in Table 5.1 we list the types of operation and their numbers involved in the trellis code decoding and *extrinsic* information calculation for one packet in five IDAs. The parameter N in Table 5.1 is equal to $2^{\tilde{k}+1}$ which means each symbol node in TWL graph comprises N possible values.

For the codes on a double-parity-check structure with $m \times m$ packets (i.e., $q = m$), we can see that the level of calculation in decoding the trellis codes is exactly same as the one in Table 5.1. But the numbers of calculating the *extrinsic* information will be doubled due to the horizontal and vertical parity-check constraints involved. Roughly, for the same trellis codes and same m (and q), the computation complexity of these five IDAs can be ordered as follows: *IDA 3* > *IDA 2* > *IDA 5* > *IDA 4* > *IDA 1*, i.e., the iterative sum-product algorithm has the highest complexity, and the IVA has the lowest complexity.

It is worth mentioning here that the re-encoding of the decoded symbols may be necessary to produce the "selector" $\widehat{W}$ in *IDA 1*, *IDA 4* and *IDA 5*. However, in *IDA 2* and *IDA 3*, the re-encoding procedure can be avoided since they are totally "soft-in/soft-out" algorithms.

5.5 Performance Analysis

Performance analysis for iterative min-sum and sum-product algorithms is a difficult task, even for a simple single cycle trellis. Many leading researchers are

IDA	Decoding the trellis code	Calculating the *extrinsic* information
IDA 1	Addition: $NSl/2$ Comparison-Selection: $NSl/2 - Sl$	Addition[a]: $[(m-3)+N]l$ Multiplication[b]: Nl
IDA 2	Addition: $3NSl/2 - NS + Nl$ Comparison-Selection: $(N-2)S(l-1)$+$[(N/2)S-1]l$	Addition: $[N^2(m-2)+N]l$ Comparison-Selection: $(N-1)N(m-2)l$ Multiplication[c]: Nl
IDA 3	Addition: $(N-2)S(l-1)$ Multiplication: $NS(l-1)$+$[(N/2)S-1]l$+Nl Comparison-Selection: $[(N/2)S-1]l$	Multiplication: $[N^2(m-2)+N]l$ Addition: $(N-1)N(m-2)l$ Root-d[d]: Nl
IDA 4	same as that of *IDA 2*	same as that of *IDA 1*
IDA 5	same as that of *IDA 3*	same as that of *IDA 1*

[a]Including $(m-3)l$ modulo-2 addition
[b]Due to the scaling down operation
[c]Due to the normalization operation
[d]Due to the normalization operation

Table 5.1: Comparison of the computational complexity of five IDAs on a single-parity-check structure with m packets $\times$ l symbols

currently focussing on this problem [11][41][54][105].

In this section, we will calculate the upper bounds for the error floors of the parity-concatenated trellis codes with and without trellis shaping. The premise is to suppose that the coded bits of trellis codes have been successfully decoded. Based on the calculated upper bound, the proper BCH code can then be selected to lower error floor level.

The upper bound for the error floor of the parity-concatenated trellis codes

without shaping can be obtained through the union bound technique since the cosubsets of constellation Λ_0 generally are non-rectangular constellations. Assuming a signal point $\mathbf{a}$ in the constellation Λ_0 is transmitted and decoded as $\mathbf{a}'$ in the same cosubset with $\mathbf{a}$. Suppose $\mathbf{a}'$ is one of the closest points with $\mathbf{a}$ and the squared distance between $\mathbf{a}$ and $\mathbf{a}'$ is $d^2_{\mathbf{a},\mathbf{a}'}$. Then the probability of the correct decision in $\mathbf{x}$ or $\mathbf{y}$ coordinate is [50]

$$P_{cx} = P_{cy} \geq 1 - 2Q\left(\sqrt{\frac{d^2_{\mathbf{a},\mathbf{a}'}}{E_{av}}\frac{n \cdot E_b}{2N_0}}\right) \tag{5.8}$$

where E_{av} is the average energy of the constellation, n is the number of bits in one symbol, and E_b/N_0 is the average SNR per bit, where E_b is the energy per bit, and N_0 denotes the (one-sided) noise power spectral density. Thus, the probability of the correct decision is

$$P_c = P_{cx} \cdot P_{cy} \geq \left[1 - 2Q\left(\sqrt{\frac{d^2_{\mathbf{a},\mathbf{a}'}}{E_{av}}\frac{n \cdot E_b}{2N_0}}\right)\right]^2 \tag{5.9}$$

Therefore, the probability of a symbol error is upper-bounded as

$$\begin{aligned} P_M &= 1 - P_c \\ &\leq 1 - \left[1 - 2Q\left(\sqrt{\frac{d^2_{\mathbf{a},\mathbf{a}'}}{E_{av}}\frac{n \cdot E_b}{2N_0}}\right)\right]^2 \\ &= 4Q\left(\sqrt{\frac{d^2_{\mathbf{a},\mathbf{a}'}}{E_{av}}\frac{n \cdot E_b}{2N_0}}\right) - 4Q^2\left(\sqrt{\frac{d^2_{\mathbf{a},\mathbf{a}'}}{E_{av}}\frac{n \cdot E_b}{2N_0}}\right) \end{aligned} \tag{5.10}$$

The average bit error probability is upper-bounded as

$$P_b(e) \leq \frac{N_{av}}{n} \cdot 4Q\left(\sqrt{\frac{d^2_{\mathbf{a},\mathbf{a}'}}{E_{av}}\frac{n \cdot E_b}{2N_0}}\right) \tag{5.11}$$

where N_{av} is the average number of error bits per symbol over all cosubsets.

If trellis shaping is combined with the parity-concatenated trellis codes, then the bit error rate will be slightly effected by the trellis shaping decoder, which can be chosen to be feedback free syndrome-former [36] and therefore only limited

error propagation will be caused for shaping coded bits. So after the trellis shaping decoding, the error bit rate is rectified as

$$P_b'(e) = P_b(e) \cdot L_{av} \tag{5.12}$$

where L_{av} is the average length caused by the error propagation. It is also worth mentioning here that the effect of trellis shaping should be considered when calculating the E_{av} value in (5.12).

According to the calculated upper bound, we can then derive the error probability when a BCH code (n_b, k_b, q_b) is combined with the parity-concatenated trellis codes to lower the error floor level. For a BCH code with hard-decision decoding, the probability of a code word error is upper-bounded by the expression [50]

$$P_M \leq P_M^{(max)} = \sum_{i=q+1}^{n_b} \binom{n_b}{i} p^i (1-p)^{n_b - i} \tag{5.13}$$

and lower-bounded by [50]

$$\begin{aligned} P_M &\geq P_M^{(min)} \\ &= \sum_{i=[\frac{d_{min,b}}{2}]+1}^{d_{min,b}} \binom{d_{min,b}}{i} p^i (1-p)^{d_{min,b} - i} \end{aligned} \tag{5.14}$$

where $d_{min,b}$ is the minimum distance of the code words, and p is the probability of binary digit error. In our case, p is the probability of binary digit protected by the BCH code. It is clear that a more powerful code needs more redundancy. Therefore, if the specified error level is given, according to the formulae (5.12)-(5.14), we can then select the appropriate BCH code to significantly reduce the error probability, with the proper tradeoff between redundancy and performance.

5.6 Discussion

In this chapter, firstly, the conventional iterative two-way algorithms based on the TWL graph representations were described for decoding the parity-concatenated

trellis codes. A simple example was given accordingly. Then we showed the modified iterative two-way algorithms in which a normalization function was applied, and the normalization factor d was determined by the number of time units in which the symbol in the error path is different from the symbol of the correct path. It can be seen that the modified two-way algorithm is an effective solution for graphs with many small cycles. From the perspective of graph interpretation, the IVA discussed in Chapter 4 can be viewed as a simplified case of the iterative min-sum algorithm in which the local weights (branch metrics) of one symbol node rather than sum of upstream and downstream weights are fed back as the *extrinsic* information.

Based on the graph representations, five iterative decoding algorithms, including the IVA and modified iterative min-sum/sum-product algorithms, were developed, and their computational complexity were compared. We can see that the iterative sum-product algorithm has the highest computational complexity, and the IVA has the lowest one.

For the parity-concatenated trellis codes at a high spectral efficiency, error floors would be appeared at the region close to the Shannon capacity limit due to the parallel transition errors in trellis codes. The upper bound on the error floor was analytically determined, and an appropriate multilevel code (BCH code) could be selected to dramatically bring down the error floor to a very low level.

Chapter 6

Numerical Results for Parity-Concatenated TCM

In this chapter, we present extensive simulation results for several parity-concatenated trellis codes. The 2-D trellis codes used for simulations are typical Ungerboeck codes and the notation is in octal form following that of [31]. For all cases the rate loss due to the redundancy has been deducted from the E_b/N_0 computation. Each simulation trial was terminated if 100 block errors were obtained, or if the total number of bits processed reached 6×10^8. In addition, the only disturbance in the channel is AWGN.

6.1 Scaling Parameter α in the IVA

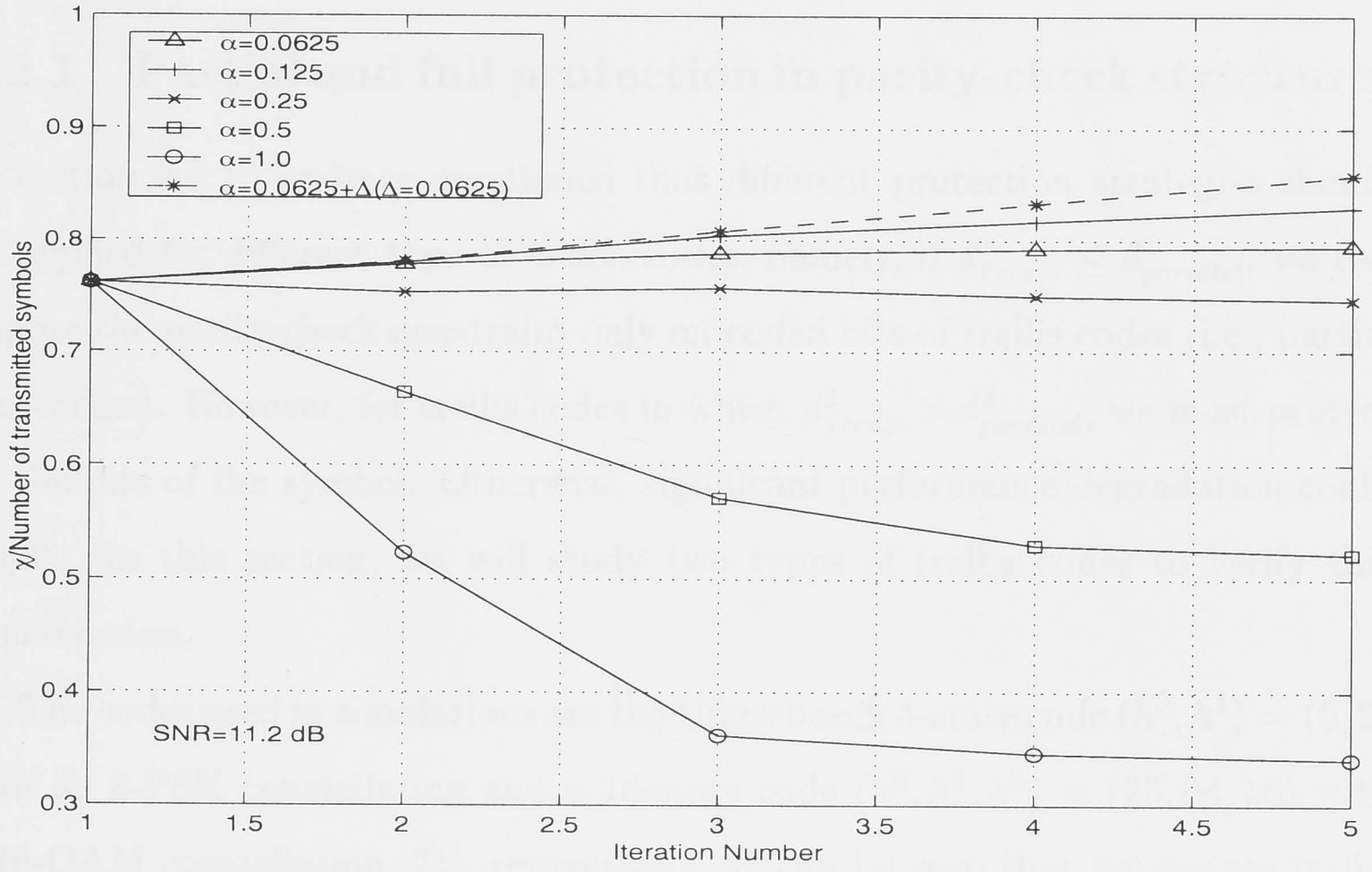
First, let us investigate the scaling parameter α used in the IVA. As we discussed in section 4.2.3, α is an important parameter in mitigating error propagation during iterative decoding. Different values for α could affect the bit error performance and convergence speed significantly. Since it is a hard task to determine α mathematically, we have to apply the experimental method to tune α.

Here we define χ as the number of cases in which the received signal R is mostly close to the subset which includes a transmitted symbol corresponding to R. Generally if χ is larger, then better performance can be achieved. If all the

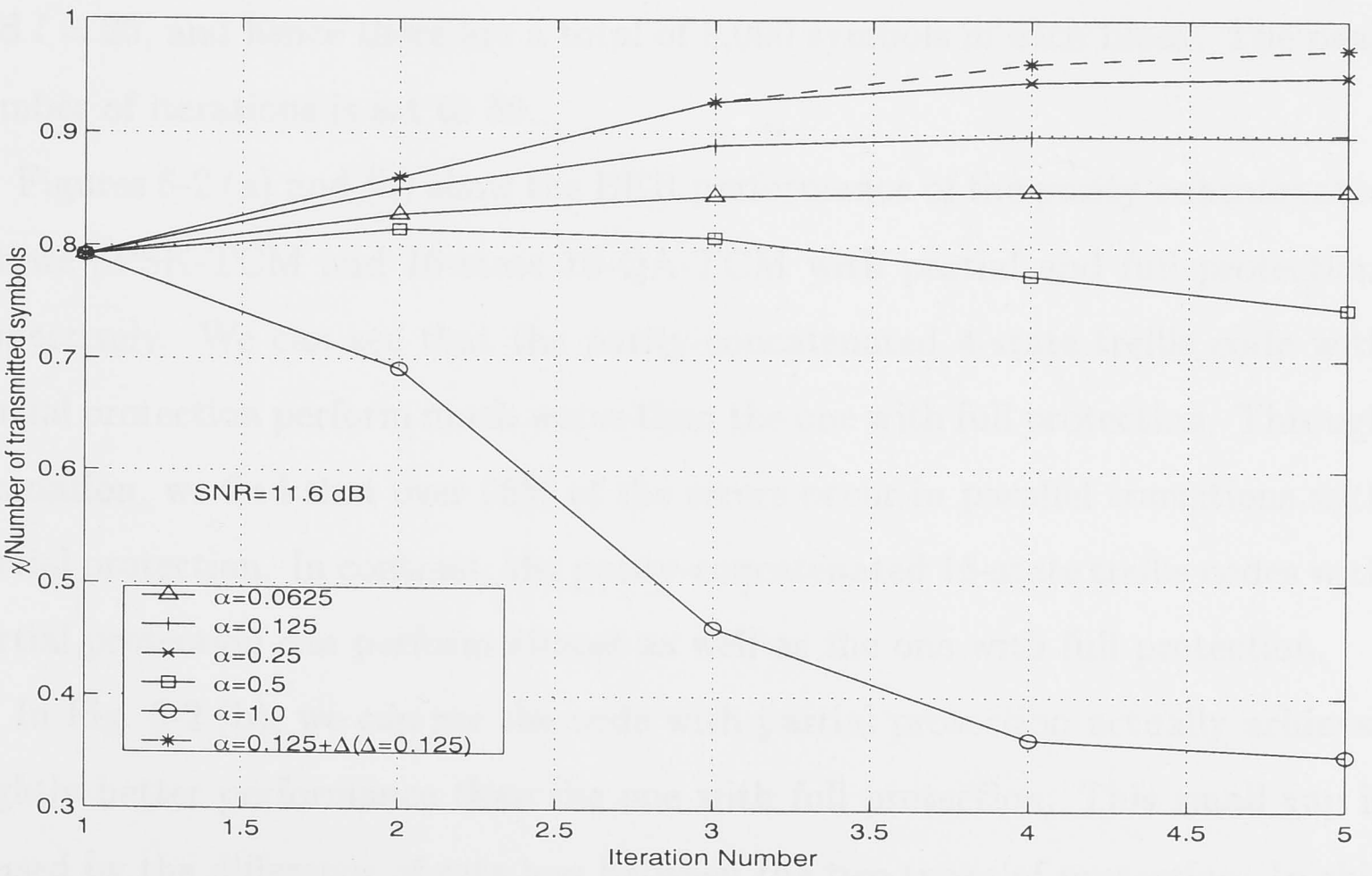
transmitted signals are mostly close to the constellation points of transmitted symbols compared with other points in the constellation, then there will be no errors in the output of the decoder. In the IVA, if α can be properly selected, χ should increase when the number of iterations increases. Therefore, we may use χ as an index to roughly compare the error performance and convergence speed of the IVA when different α values are used. With a 256-QAM constellation, a 16-state trellis code $(h^0, h^1, h^2) = (23, 04, 16)$ [31] and 16-state trellis shaping are employed on a double-parity-check structure in which $m = q = 20$ and $l = 50$. Figures 6-1 (a) and 6-1 (b) show the relationship between χ and the number of iterations when SNR is 11.2 dB and 11.6 dB, respectively.

In Fig. 6-1, it is shown that due to error propagation χ dramatically decreases with an increasing number of iterations when $\alpha = 1.0$. If α is fixed during all iterations, we can see that $\alpha = 0.125$ and $\alpha = 0.25$ are the best values for SNR=11.2 dB and 11.6 dB, respectively. However, if α is below those values, χ will increase but too slowly to be practically useful. This is because the positive feedback information is becoming weak. Generally, if α is selected properly, more positive information can be fed to the next iteration, and then less error propagation occurs. Therefore, we may increase α with the number of iterations. In Fig. 6-1, we can see that if we gradually increase α by a small value after each iteration, better performance than using a fixed α can be obtained. In the following simulations, the gradually increased scale will be applied in the IVA.

It is worth mentioning here that the scales suitable for the above codes probably are not exactly appropriate for other codes. According to our extensive simulations, we find that the scale is related to many factors such as the particular trellis code, the size of the block (i.e., the values of m, q and l) and the SNR value. However, although it is very difficult to determine an optimal scale for different trellis codes with different packet sizes at various SNR, we find that near optimal scales are generally good enough for achieving satisfactory performance.



(a)



(b)

Figure 6-1: Comparison of χ with different α values in the IVA

6.2 Performance of the IVA and error floor

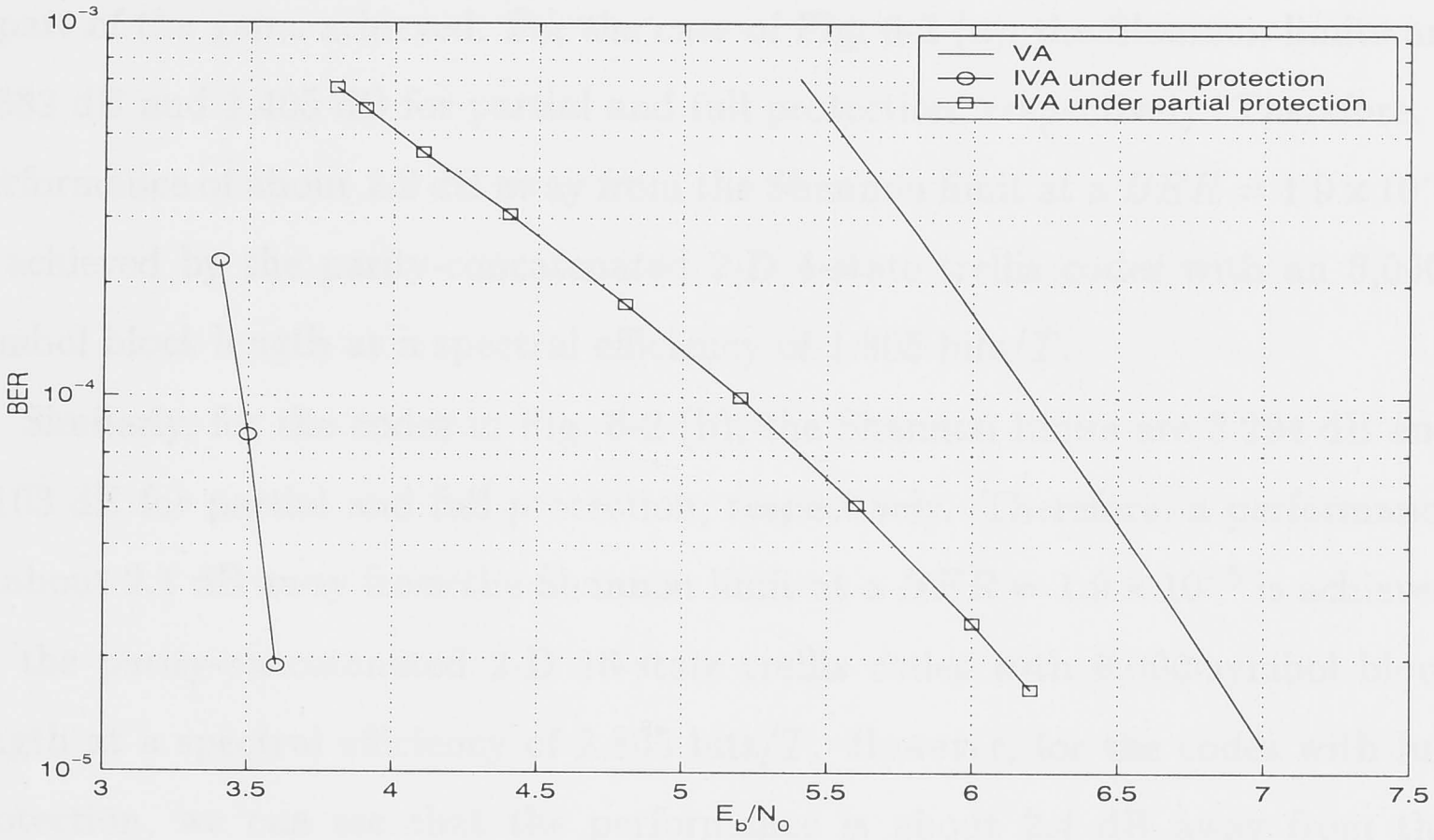
6.2.1 Partial and full protection in parity-check structures

In section 3.3.1, we have mentioned that different protection strategies should be applied for different type of trellis codes, namely, if $d^2_{free,c} \leq d^2_{parallel}$, we can impose the parity-check constraint only on coded bits of trellis codes (i.e., partial protection). However, for trellis codes in which $d^2_{free,c} > d^2_{parallel}$, we must protect all the bits of the symbol. Otherwise, significant performance degradation could result. In this section, we will study two types of trellis codes to verify this phenomenon.

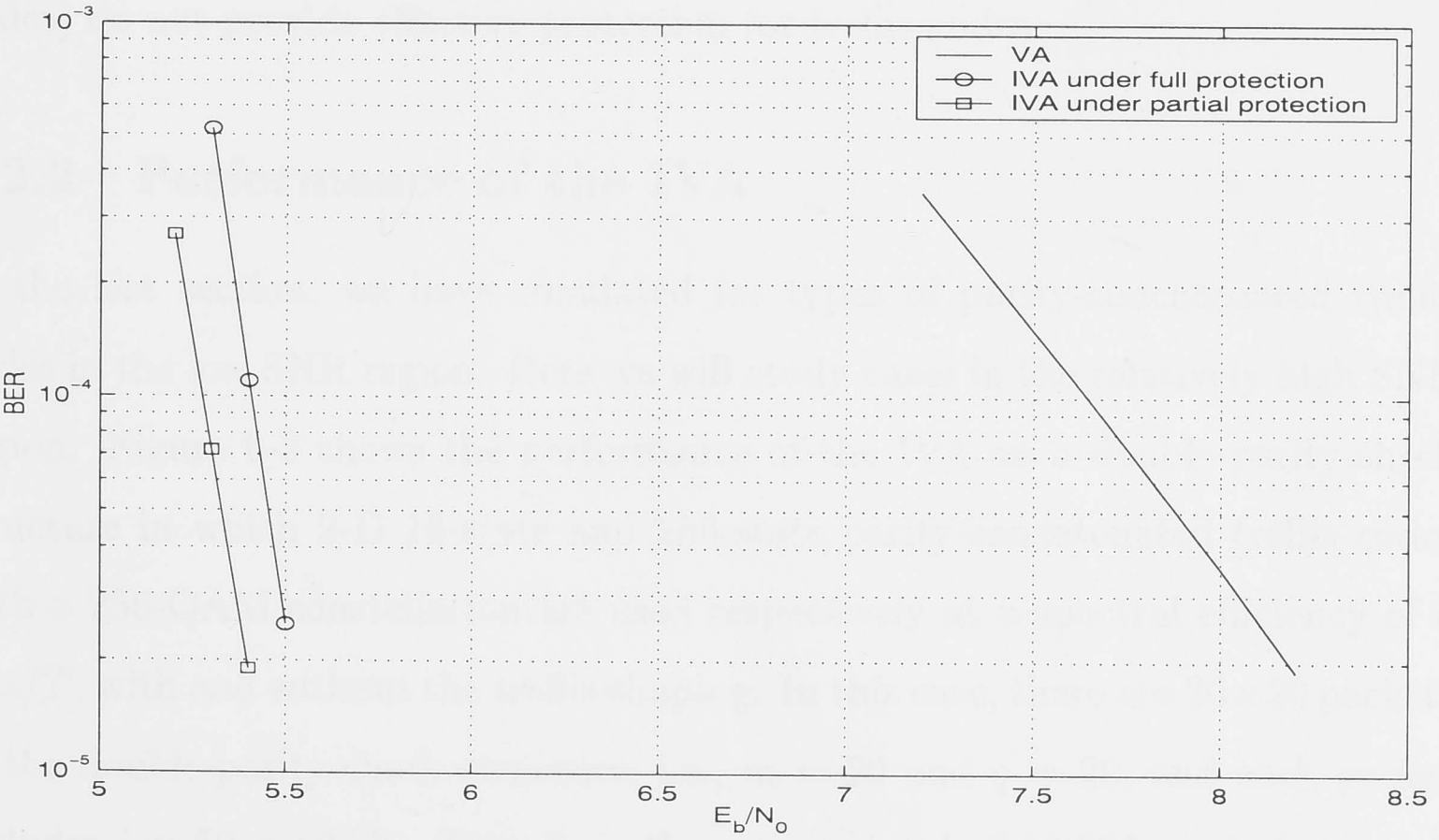
The codes used in simulations are the Ungerboeck 4-state code $(h^0, h^1) = (5, 2)$ with an 8-PSK constellation and a 16-state code $(h^0, h^1, h^2) = (23, 04, 16)$ with a 16-QAM constellation [21], respectively. It can be seen that for 4-state trellis codes $d^2_{free,c}$ is larger than $d^2_{parallel}$ while for 16-state trellis codes $d^2_{free,c}$ is smaller than $d^2_{parallel}$. We use a double-parity-check structure in which $m = 20$, $q = 20$ and $l = 20$, and hence there are a total of 8,000 symbols in each block. The peak number of iterations is set to 50.

Figures 6-2 (a) and (b) show the BER performance of the parity-concatenated 4-state 8PSK-TCM and 16-state 16-QA-TCM with partial and full protection, respectively. We can see that the parity-concatenated 4-state trellis code with partial protection perform much worse than the one with full protection. Through simulation, we find that over 95% of the errors occur in parallel transitions with partial protection. In contrast, the parity-concatenated 16-state trellis codes with partial protection can perform almost as well as the one with full protection.

In Fig. 6-2 (b), we can see the code with partial protection actually achieves slightly better performance than the one with full protection. This small gap is caused by the difference of rate loss between the two types of protection. In this example, after reduction for the redundancy bits, the real rates are 2.805 bits/T for partial protection and 2.7075 bits/T for full protection, respectively. Therefore the difference of rate loss is about 0.154 dB. Apart from the rate loss, the Shannon



(a) Bit error rate of parity-concatenated 4-state 8PSK-TCM



(b) Bit error rate of parity-concatenated 16-state 16-QA-TCM

Figure 6-2: Comparison of BER performance for parity-concatenated 4-state 8PSK-TCM and 16-state 16-QA-TCM with partial and full protection, respectively ($m = 20$, $q = 20$ and $l = 20$)

capacity limit is also left-shifted due to the redundant bits which actually offsets a part of the gains achieved. For the case of Fig. 6-2 (a), the Shannon limits are 1.582 dB and 1.405 dB for partial and full protection, respectively. Therefore, a performance of about 2.2 dB away from the Shannon limit at a $BER = 1.9 \times 10^{-5}$ is achieved by the parity-concatenated 2-D 4-state trellis codes with an 8,000-symbol block length at a spectral efficiency of 1.805 bits/T.

Similarly, for the codes in Fig. 6-2 (b), the Shannon limits are 3.294 dB and 3.103 dB for partial and full protection, respectively. Therefore, a performance of about 2.1 dB away from the Shannon limit at a $BER = 1.9 \times 10^{-5}$ is achieved by the parity-concatenated 2-D 16-state trellis codes with 8,000-symbol block length at a spectral efficiency of 2.805 bits/T. However, for the codes with full protection, we can see that the performance is about 2.4 dB away from the Shannon limit. That means that protecting the full symbols is not suitable in this case because the additional non-information bits (i.e, uncoded bits of trellis codes) do not provide effective protection for trellis codes.

6.2.2 Performance of the IVA

In the last section, we have simulated for types of parity-concatenated trellis codes in the low SNR region. Here we will study cases in the relatively high SNR region. Figure 6-3 shows the performance of the IVA on a double-parity-check structure in which 2-D 16-state and 256-state parity-concatenated trellis codes with a 256-QAM constellation are used respectively at a spectral efficiency of 6 bits/T, with and without the trellis shaping. In this case, there are 20×20 packets in the double-parity-check structure, i.e., $m = 20$ and $q = 20$, and each packet includes $l = 50$ symbols. Therefore, there are a total of 20,000 symbols in each block. A 16-state shaping code is applied in the simulation. The peak iteration number is set to 50. The 16-state and 256-state trellis codes are the FF form of the Ungerboeck codes $(h^0, h^1, h^2) = (23, 04, 16)$ and $(h^0, h^1, h^2) = (401, 056, 304)$ [21]. Their d^2_{free} are $6d^2_{min,(\Lambda_0)}$ and $8d^2_{min,(\Lambda_0)}$, respectively. After set partitioning, $d^2_{parallel}$ $(d^2_{min,(\Lambda_3)})$ is $8d^2_{min,(\Lambda_0)}$ in one of cosubsets of Λ_3. Therefore, protecting

the trellis-coded bits is enough to achieve almost all of the expected gains. For comparison, the performance of a 256-state TCM scheme using the standard VA is also reported in Fig. 6-3. The results show that for the $\nu = 8$ codes about 2.0 dB gross gain can be achieved by the IVA beyond the VA without shaping and about 2.7 dB gross gain with shaping. However, note that the shaping gain is smaller at low SNRs and only about 0.7 dB shaping gain was obtained in this case (a similar example can be found in [104]) if 16-state trellis shaping was used (assuming that the baseline constellation is a 128-point cross constellation). In Fig. 6-3, we also noted that the performances of the $\nu = 4$ and $\nu = 8$ codes are pretty close. This may be explained by the same reason that the performance of the $\nu = 8$ codes is worse than the $\nu = 4$ codes at low SNRs if only the VA is used as the decoding algorithm.

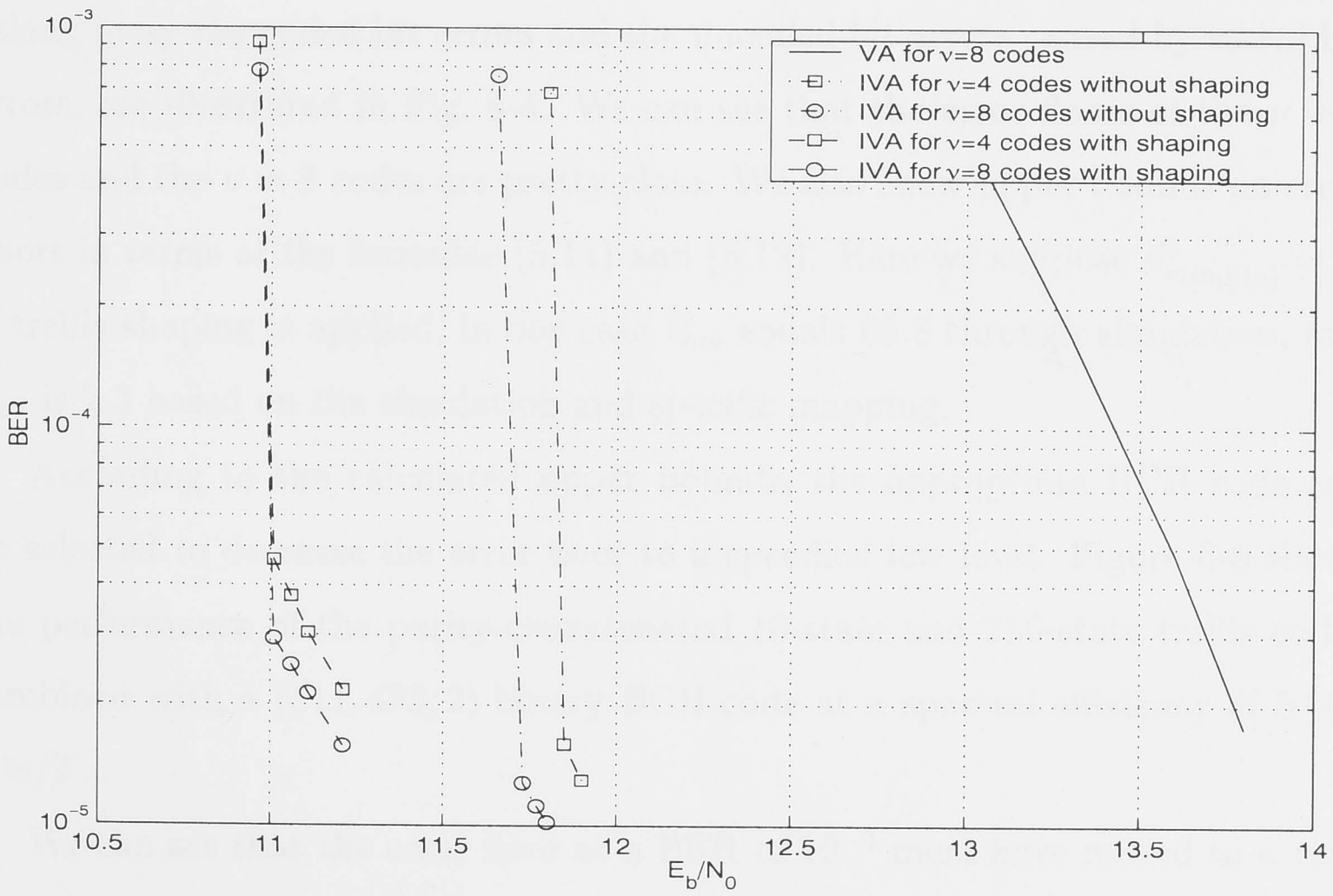


Figure 6-3: BER performance of the 16-state and 256-state trellis codes using the IVA based on a double-parity-check structure at a spectral efficiency of 5.805 bits/T with partial protection

Similar to the two codes in Fig. 6-2, the results shown in Fig. 6-3 are not the ultimate gains because each block includes some non-information bits which cause

the real transmission rate to be lower than the nominal rate. In this example, the real rate now is 5.805 bits/T rather than 6 bits/T. The Shannon capacity limit for this rate is 9.76 dB. Therefore, a performance of about 1.25 dB away from the Shannon limit at a $BER = 3.0 \times 10^{-5}$ is achieved by the parity-concatenated 2-D 256-state trellis codes using the IVA on a double-parity-check structure at a spectral efficiency of 5.805 bits/T.

6.2.3 Performance of the error floor and multilevel code

We can see the error floors in Fig. 6-3. These error floors are mainly dominated by the parallel transition errors at relatively low SNRs. In our simulation, we find that over 95% of the errors belong to the parallel transition bits (i.e., uncoded bits) in the $\nu = 8$ codes when the SNR equals 11.1 dB. The error floors, after taking away the coded bit errors and the uncoded bit errors caused by coded bit errors, are illustrated in Fig. 6-4. We can see that the error floors of the $\nu = 4$ codes and the $\nu = 8$ codes are pretty close. We also show upper bounds on error floors in terms of the formulae (5.11) and (5.12). Here we suppose $d^2_{min,(\Lambda_0)} = 4$. If trellis shaping is applied, in our case E_{av} equals 65.8 through simulation, and L_{av} is 2.3 based on the simulation and specific mapping.

According to the calculated upper bounds, the appropriate BCH code can be selected to decrease the error floor to a specified low level. Figure 6-5 shows the performance of the parity-concatenated 16-state and 256-state trellis codes combined with a $(511, 493, 2)$ binary BCH code at a spectral efficiency of 5.805 bits/T.

We can see that the error floor at a BER of 10^{-5} must have moved to a very low level. Due to the prohibitive length of the simulation involved, we could not determine the next error floor level through simulation. However, we can compute the next error floors in terms of the formulae (5.11) and (5.12) with a new $d^2_{min,(\Lambda_i)}$. It is shown in Fig. 6-6 that the error floors have been reduced to the level of $BER = 10^{-9}$. It is obvious that $d^2_{min,(\Lambda_i)}$ has doubled after the set partitioning corresponding to the BCH code.

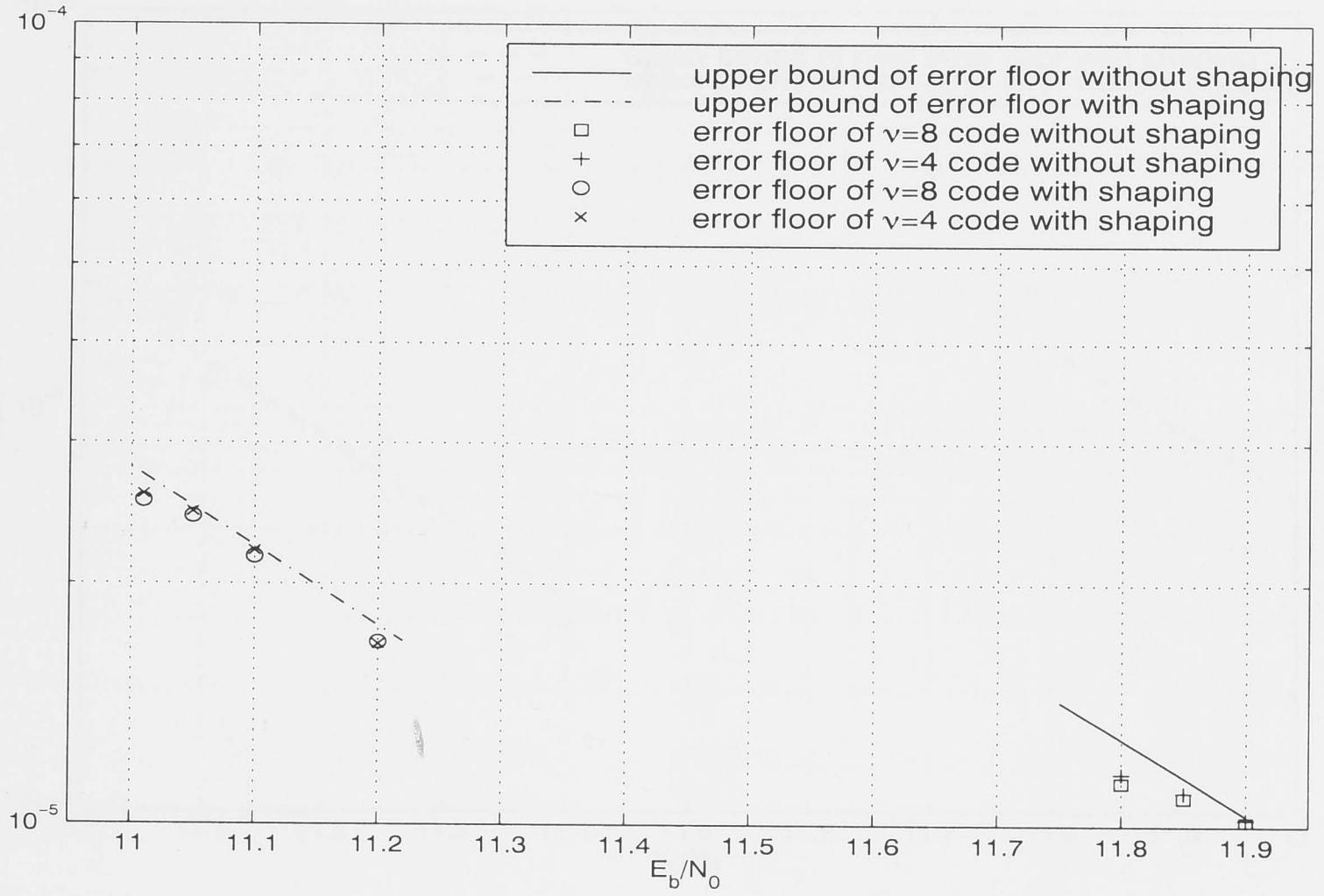


Figure 6-4: Error floors and their upper bounds corresponded to the codes in Fig. 6-3

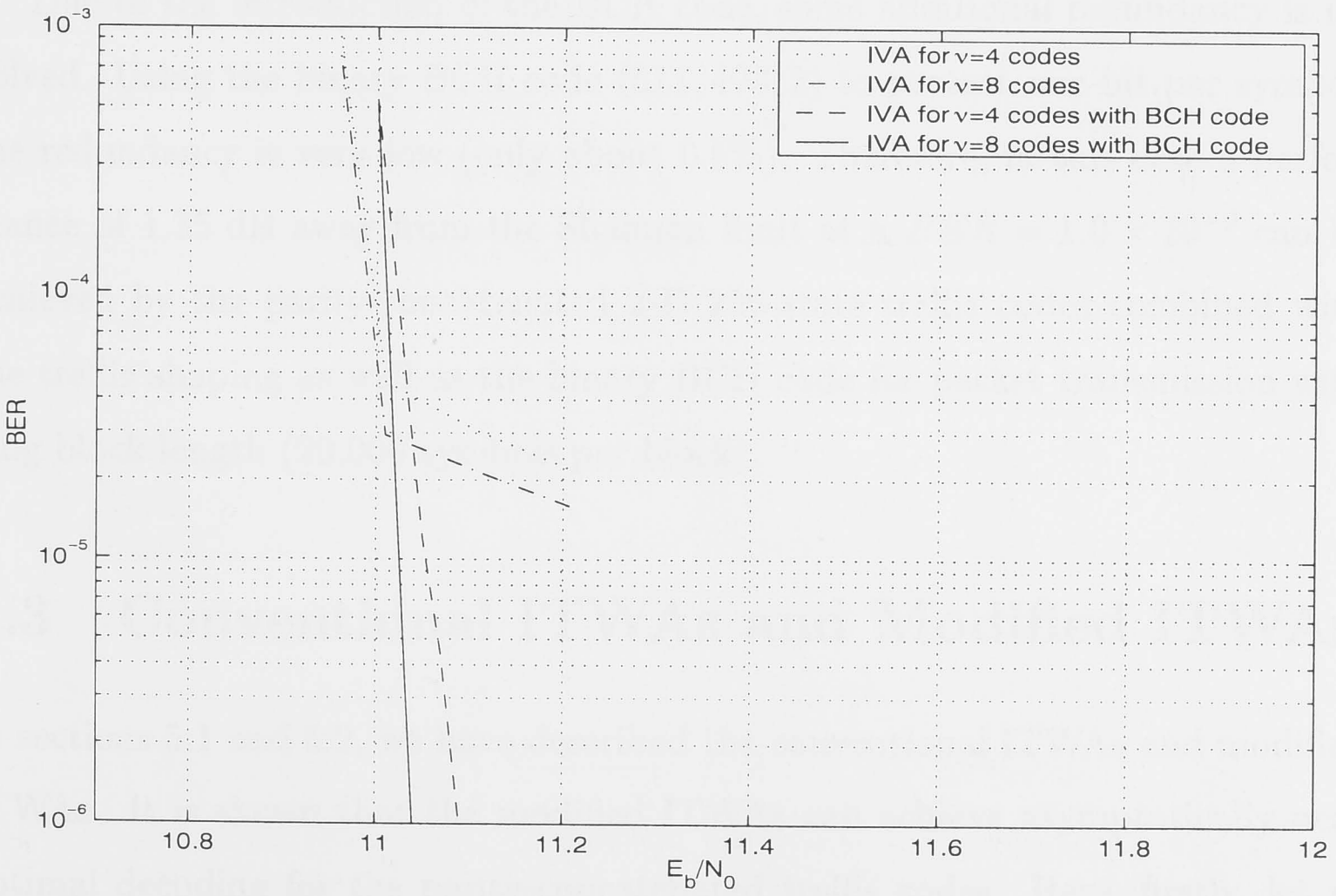


Figure 6-5: BER performance of the parity-concatenated 16-state and 256-state trellis codes (the case of Fig. 6-3) combined with a (511,493,2) BCH code

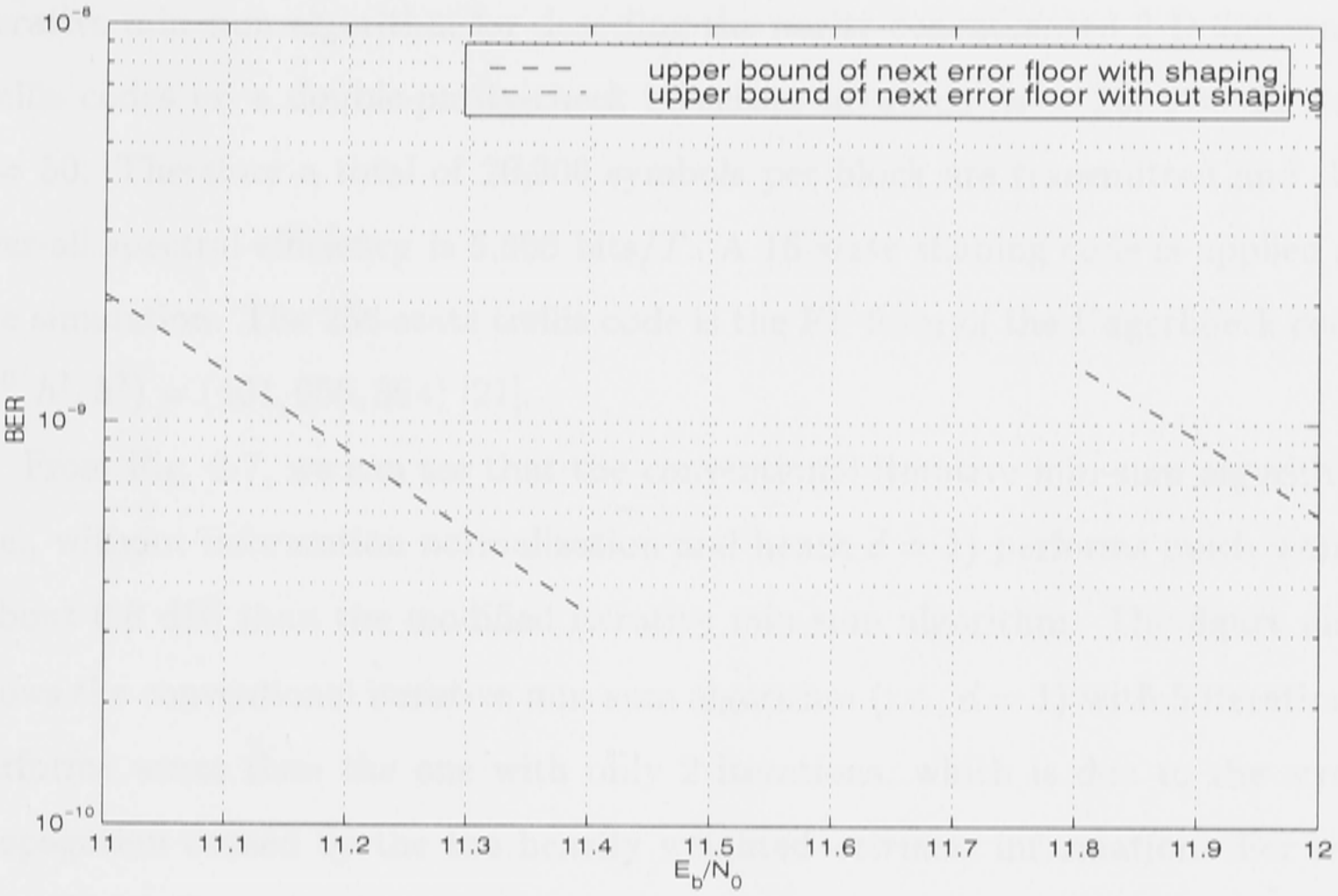


Figure 6-6: Upper bounds of the new error floors after introduction of a BCH code

Due to the introduction of the BCH code, some additional redundancy is involved. Using the binary BCH code $(511, 493, 2)$ to protect one bit per symbol, the redundancy is very low (only about 0.5%). Therefore, in this case a performance of 1.35 dB away from the Shannon limit at a $BER = 1.0 \times 10^{-9}$ can be achieved by the parity-concatenated 2-D 256-state trellis codes combined with the trellis shaping as well as the binary BCH code for packet transmission with long block length (20,000 symbols per block).

6.3 Conventional ITWAs and Modified ITWAs

In sections 5.1 and 5.2, we have described the conventional ITWAs and modified ITWAs. It is shown that the modified ITWAs can achieve asymptotically near optimal decoding for the parity-concatenated trellis codes. Here, firstly, let us verify the results of proposition 1 in section 5.2. In Fig. 6-7 we compare the bit error rate performances of conventional iterative min-sum algorithm and modified

iterative min-sum algorithm for decoding the parity-concatenated 2-D 256-state trellis codes on a double-parity-check structure in which $m = 20$, $q = 20$ and $l = 50$. Therefore a total of 20,000 symbols per block are transmitted and the over-all spectral efficiency is 5.805 bits/T. A 16-state shaping code is applied in the simulation. The 256-state trellis code is the FF form of the Ungerboeck code $(h^0, h^1, h^2) = (401, 056, 304)$ [21].

From Fig. 6-7, we can see that the conventional iterative min-sum algorithm (i.e., without information normalization and hence $d = 1$) performs much worse (about 0.8 dB) than the modified iterative min-sum algorithm. The figure also shows the conventional iterative min-sum algorithm (i.e., $d = 1$) with 5 iterations performs worse than the one with only 2 iterations, which is due to the error propagation caused by the too heavily weighted *extrinsic* information. For the 256-state trellis code, the length of the error events with minimum distance is 5, but some of the error events with minimum distance have the same symbols at one time unit. For these error events, we have $d = 5 - 1 = 4$. In the other error events with minimum distance, all symbols of the error paths are different from the symbols of the correct path. Therefore, we have $d = 5$ for these error events. Thus, we expect that the modified iterative min-sum algorithm with $d = 4$ or 5 will achieve the best asymptotic performance. This figure shows that. When $d = 2$, error propagation is still severe. As d approaches 4, error propagation becomes less a problem. When d reaches 7, the performance becomes worse again due to the under-fed *extrinsic* information.

In Fig. 6-8, we compare the modified iterative min-sum algorithm and the iterative ML algorithm given in proposition 1. We use the parity-concatenated 2-D 16-state trellis codes with $(h^0, h^1, h^2) = (23, 04, 16)$ [31] based on a single-parity-check structure, in which $m = 10$ and $l = 40$. Therefore a total of 400 symbols are transmitted in each block and the over-all spectral efficiency is 5.800 bits/T. A 256-QAM constellation and 16-state trellis shaping is employed. Since the length of the error event with minimum distance is 3 in this case, and the symbols of two paths at all time units are different, we set $d = 3$. In the modified

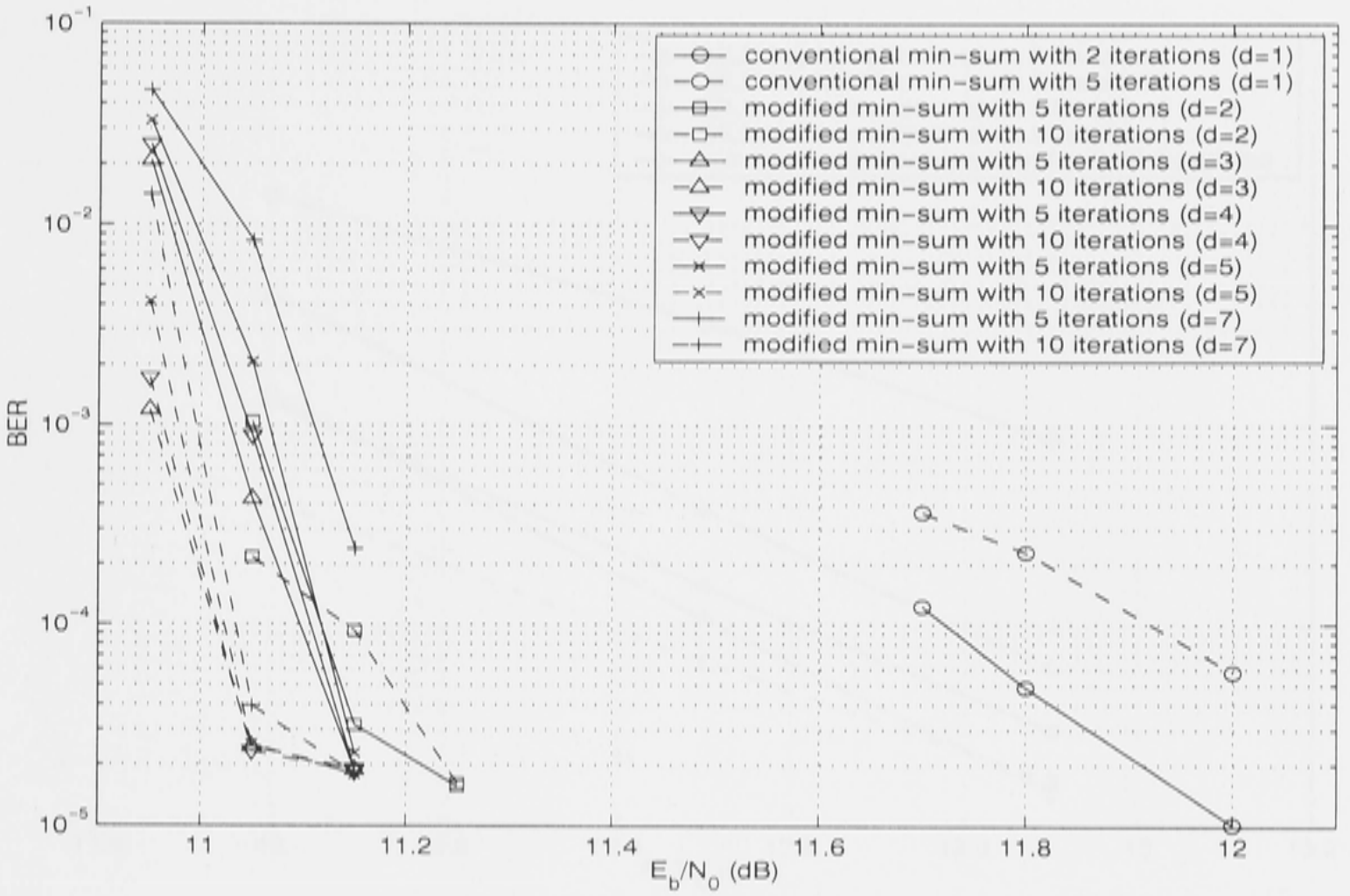


Figure 6-7: Comparison of BER performance between conventional and modified iterative min-sum algorithms with different values of d

iterative min-sum algorithm the *extrinsic* information is updated following every application of the bi-directional VA algorithm to each packet.

Figure 6-8 shows that the iterative ML algorithm given in proposition 1 is very close to the modified iterative min-sum algorithm at high SNRs and the number of iterations required is only 2. At high SNRs the performance of the ML algorithm with $d = 3$ and 2 iterations is close to the modified iterative min-sum algorithm. For low SNRs, a larger number of iterations is needed for near-optimal decoding.

6.4 Performance and Complexity Comparison of Several Iterative Decoding Algorithms

We have discussed five IDAs in section 5.4 for the parity-concatenated trellis codes. Here we simulate and compare their performance and computational complexity for these IDAs under various circumstances.

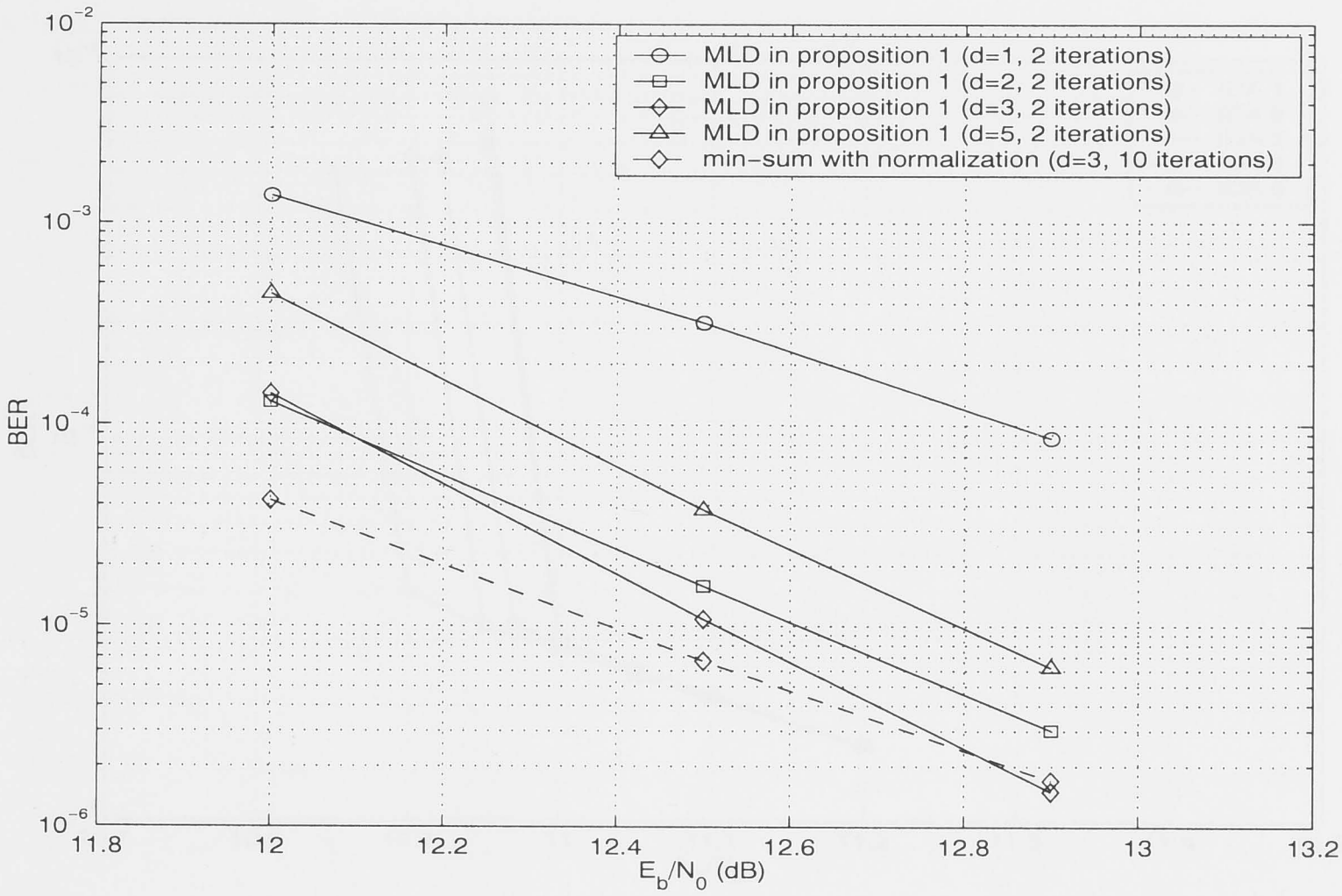
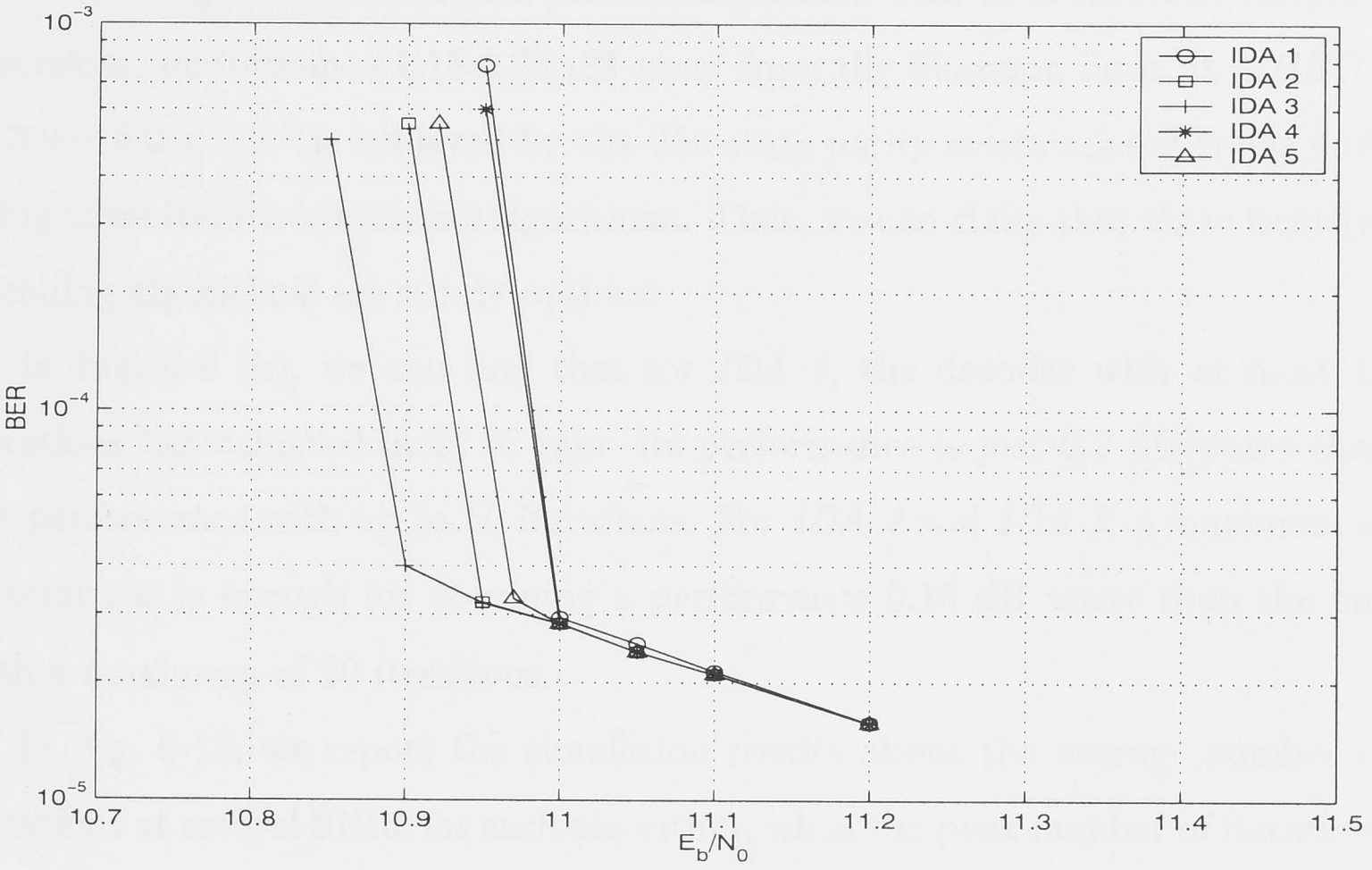


Figure 6-8: Comparison of BER performance between the modified iterative min-sum algorithm and the iterative ML algorithm at high SNRs

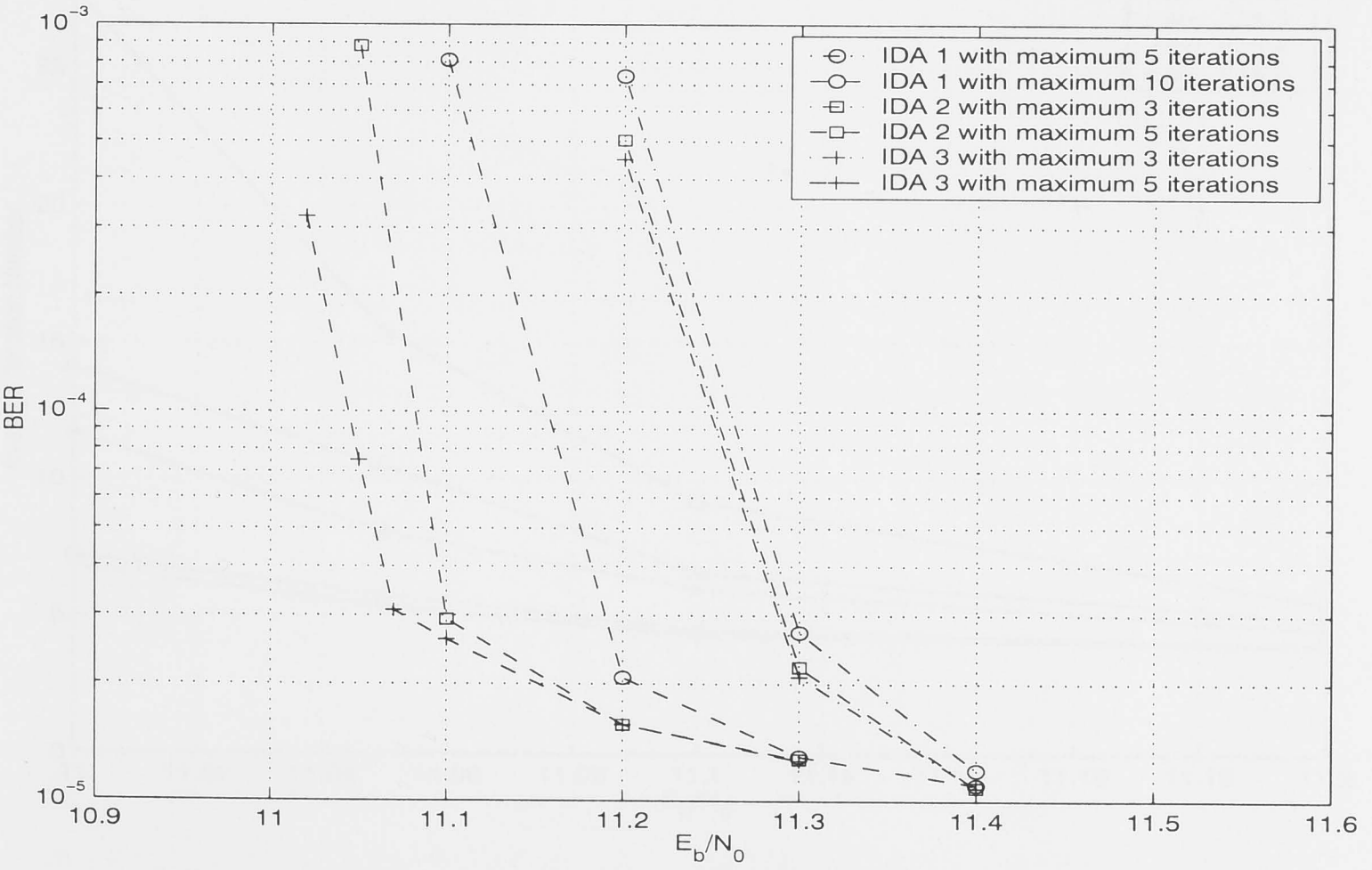
6.4.1 Packet transmission with long block length

In Figures 6-9 (a) and (b), with various maximum iterations, we compare the performances of five IDAs for the same 256-state trellis codes used in Fig. 6-3 with long packet transmission. Here still $m = 20$, $q = 20$ and $l = 50$. The peak iteration number was set to 20 for the modified iterative min-sum and sum-product algorithms ($d = 4$) and 50 for *IDA 4*, *IDA 5* and the IVA, since the IVA often converges much more slowly than the modified iterative min-sum or sum-product algorithm. 16-state trellis shaping is combined with all the five IDAs.

In Fig. 6-9 (a), we can see that *IDA 3* (iterative sum-product algorithm) achieves the best performance and *IDA 1* (IVA) gives the worst performance, but the difference in performance achieved by these five IDAs is not substantial (less than 0.1 dB). Compared with the performance of a 256-state TCM scheme using the VA reported in Fig. 6-3, the results show that at a $BER = 4.0 \times 10^{-5}$, about 2.8 dB coding gain can be achieved by the parity-concatenated code



(a) Bit error rate of five IDAs with a maximum of 20/50 iterations



(b) Bit error rate of five IDAs with various maximum iterations

Figure 6-9: Comparison of BER performance of the $\nu = 8$ trellis codes at a spectral efficiency of 5.805 bits/T using the IDAs for packet transmission with long block length ($m = 20$, $q = 20$ and $l = 50$, 20,000-symbol in each block)

decoded using the modified sum-product algorithm with 16-state trellis shaping. Therefore, performance 1.15-1.25 dB away from the Shannon limit at a BER's of $3.0 - 4.0 \times 10^{-5}$ is achieved by the 256-state parity-concatenated trellis code using these iterative decoding algorithms. Thus, we can claim that these iterative decoding algorithms are nearly optimal.

In Fig. 6-9 (b), we can find that for *IDA 1*, the decoder with at most 10 iterations has achieved most of gain: its performance is just 0.2 dB worse than the performance with up to 50 iterations. For *IDA 2* and *IDA 3*, a maximum of 5 iterations is enough for obtaining a performance 0.15 dB worse than the one with a maximum of 20 iterations.

In Fig. 6-10, we report the simulation results about the average number of iterations at several SNRs for each algorithm, when the peak number of iterations is set to 20/50.

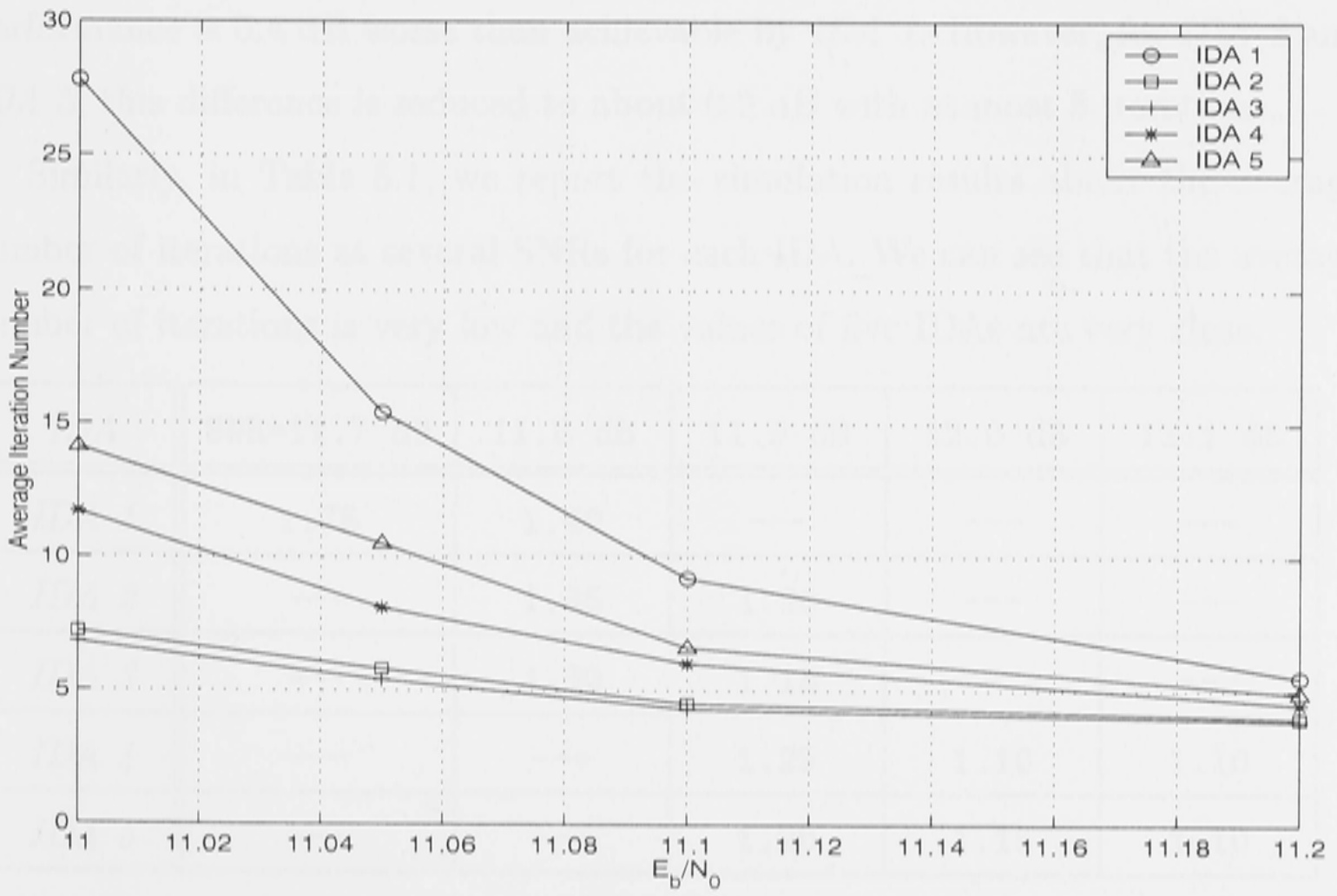


Figure 6-10: Average number of iterations of five IDAs for the case of Fig. 6-9

From Fig. 6-10, we note that *IDA 3* uses the lowest number of iterations, and *IDA 1* uses the largest number of iterations. For *IDA 1*, when the SNR increases from 11.0 dB to 11.2 dB, the average number of iterations drops dramatically

from 27.8 to 5.5. However, if considering the computational complexity of these five IDAs and taking into account the additional "sum" and "product" operations needed for updating each symbol node, we found that the computational complexity of the modified iterative sum-product algorithm is still much higher than that of the IVA, especially at high SNRs.

6.4.2 Packet transmission with short block length

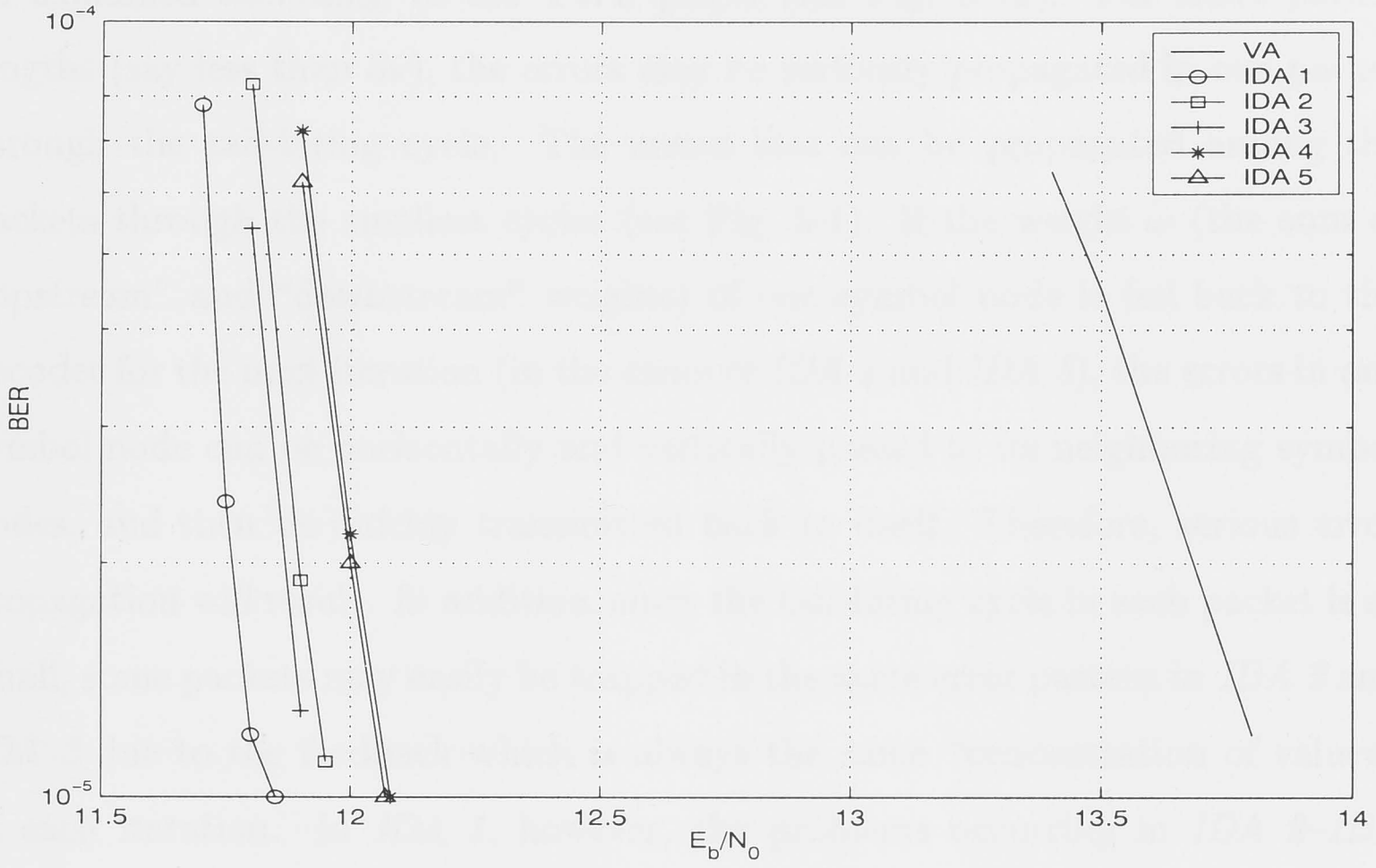
In Fig. 6-11, we present the performances of five IDAs with various maximum iterations for the $\nu = 8$ trellis codes with short packet transmission. Here we use a single-parity-check structure in which $m = 10$ and $l = 20$ (i.e., 200 symbols in each block, hence the real rate is 5.800 bits/T). The peak number of iterations is set to 50. We can see from Fig. 6-11 (a) that a performance of about 2.1 dB away from the Shannon limit is achieved by *IDA 1*. With a maximum of 10 iterations, performance is 0.4 dB worse than achievable by *IDA 1*. However, for *IDA 2* and *IDA 3*, this difference is reduced to about 0.2 dB with at most 5 iterations.

Similarly, in Table 6.1, we report the simulation results about the average number of iterations at several SNRs for each IDA. We can see that the average number of iterations is very low and the values of five IDAs are very close.

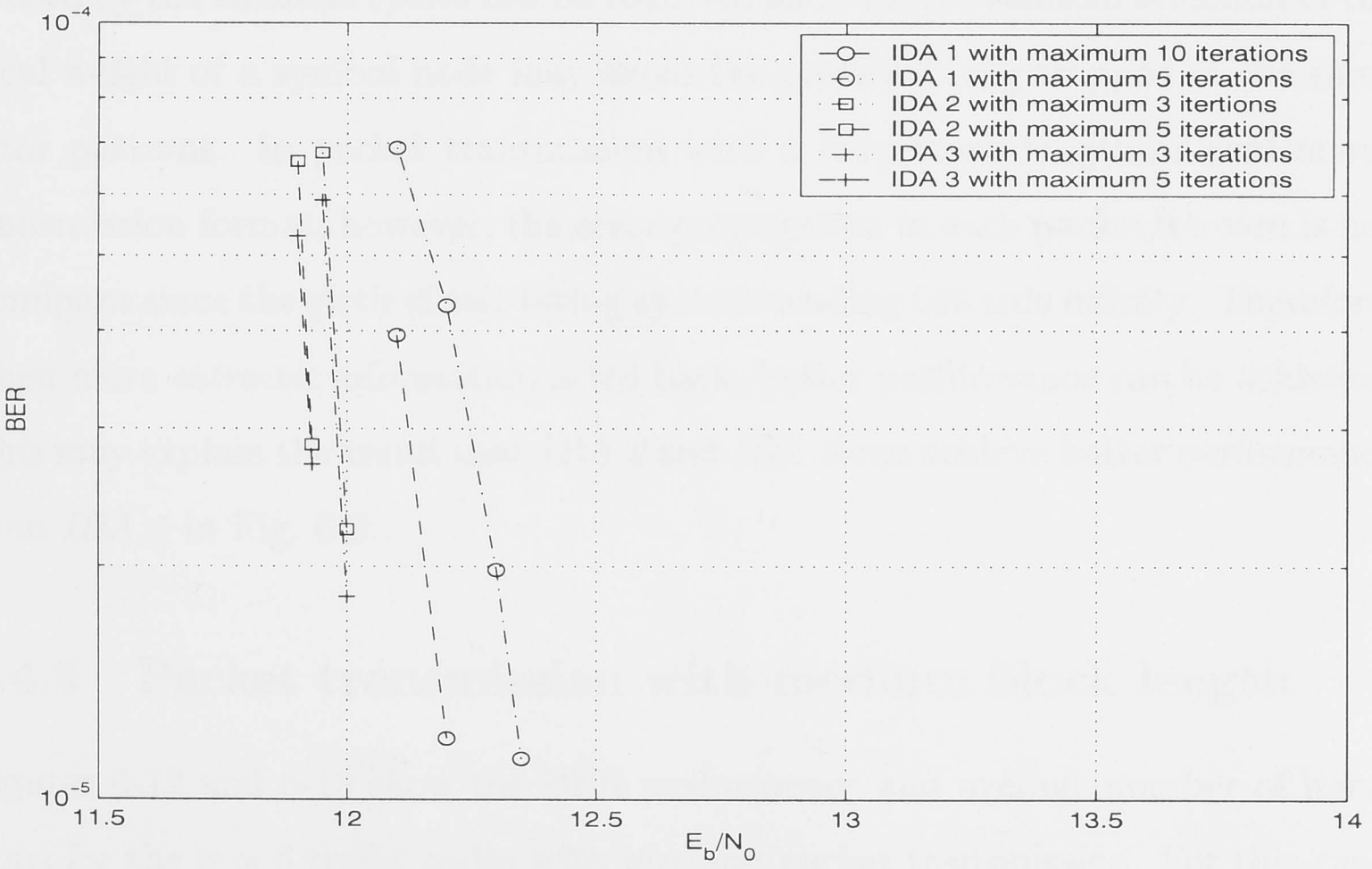
IDA	SNR=11.7 dB	11.8 dB	11.9 dB	12.0 dB	12.1 dB
IDA 1	1.75	1.40	---	---	---
IDA 2	---	1.35	1.20	---	---
IDA 3	---	1.30	1.18	---	---
IDA 4	---	---	1.25	1.16	1.10
IDA 5	---	---	1.20	1.15	1.10

Table 6.1: Average number of iterations for five IDAs in the case of Fig. 6-11

Different from the case of Fig. 6-9 (a), in Fig. 6-11 (a) we notice that *IDA 1* performs slightly better than *IDA 2* and *IDA 3*. The performance difference is within 0.15 dB. *IDA 4* and *IDA 5* achieve the worst performance. This may



(a) Bit error rate of the IDAs with a maximum of 50 iterations



(b) Bit error rate of the IDAs with various maximum iterations

Figure 6-11: Comparison of BER performance of the $\nu = 8$ trellis codes at a spectral efficiency of 5.800 bits/T using the IDAs for packet transmission with short block length($m = 10$ and $l = 20$, 200-symbol in each block)

be explained according to the TWL graph (see Fig. 3-19). For short packet lengths (say less than 5ν), the errors may be seriously propagated in one packet through the tail-biting cycle. The errors also can be propagated among the packets through the smallest cycles (see Fig. 5-1). If the weight ω (the sum of "upstream" and "downstream" weights) of one symbol node is fed back to the decoder for the next iteration (in the cases of *IDA 4* and *IDA 5*), the errors in one symbol node can be horizontally and vertically passed to its neighboring symbol nodes, and then be quickly transmitted back to itself. Therefore, serious error propagation will result. In addition, since the tail-biting cycle in each packet is so small, some packets may easily be trapped in the same error pattern in *IDA 2* and *IDA 3* due to the feedback which is always the same "concentration of values" at each iteration. In *IDA 1*, however, the problems occurring in *IDA 2*–*IDA 5* are partially avoided. Namely, since the local weight rather than the "sum" weight ω of one symbol node is fed back to the decoder, the error propagation caused by the smallest cycles can be reduced, and also the random selection of the local weight of a symbol node may avoid the packets being trapped in the same error patterns. In packet transmission with a long block length or continuous transmission format, however, the error propagation in each packet/stream is not dominant since the girth of tail-biting cycle is tending towards infinity. Therefore, when more *extrinsic* information is fed back, better performance can be achieved. This may explain the result that *IDA 2* and *IDA 3* can achieve better performance than *IDA 1* in Fig. 6-9.

6.4.3 Packet transmission with medium block length

Figures 6-12 and 6-13 show the BER performance and average number of iterations for the $\nu = 8$ trellis codes with medium packet transmission. For this case, there are two possibly alternative encoding structures. One is a single-parity-check structure in which $m = 10$ and $l = 200$ (the real rate is 5.800 bits/T), and the other one is a double-parity-check structure in which $m = 10$, $q = 10$ and $l = 20$ (the real rate is 5.620 bits/T). The peak number of iterations is set to

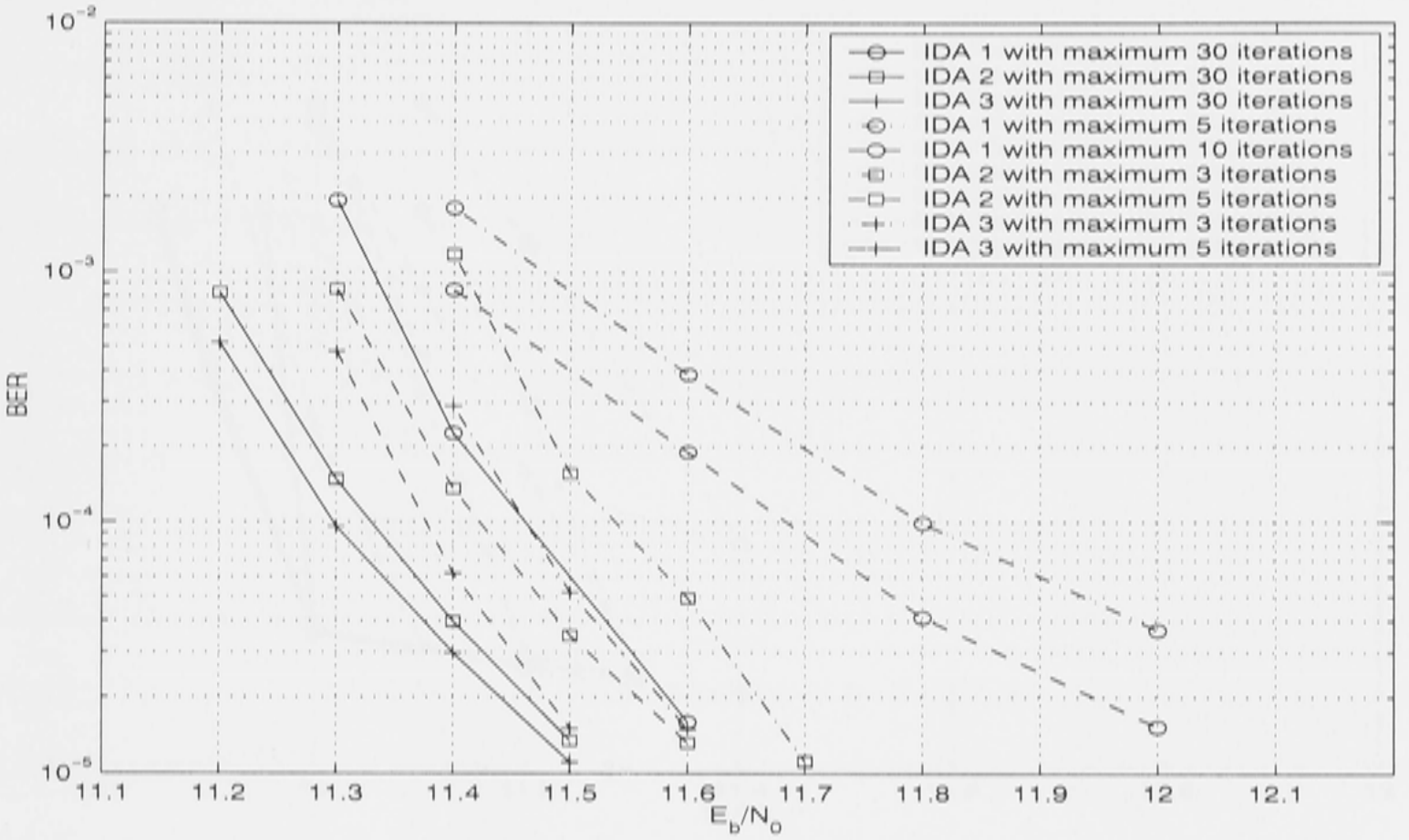
30 for both cases. We present and compare their performance in Figures 6-12 and 6-13, respectively.

From Fig. 6-12 (a), we can see that, based on the single-parity-check structure, a performance of about 1.85 dB and 1.75 dB away from the Shannon limit at a $BER = 1.5 \times 10^{-5}$ is achieved by the IVA and iterative min-sum algorithm, respectively, with a maximum of 30 iterations. With a maximum of 10 iterations, the gain achieved by the IVA is reduced by about 0.4 dB. However, for the iterative min-sum and sum-product algorithms with a maximum of only 5 iterations, the gain losses are just 0.1 dB and 0.025 dB, respectively.

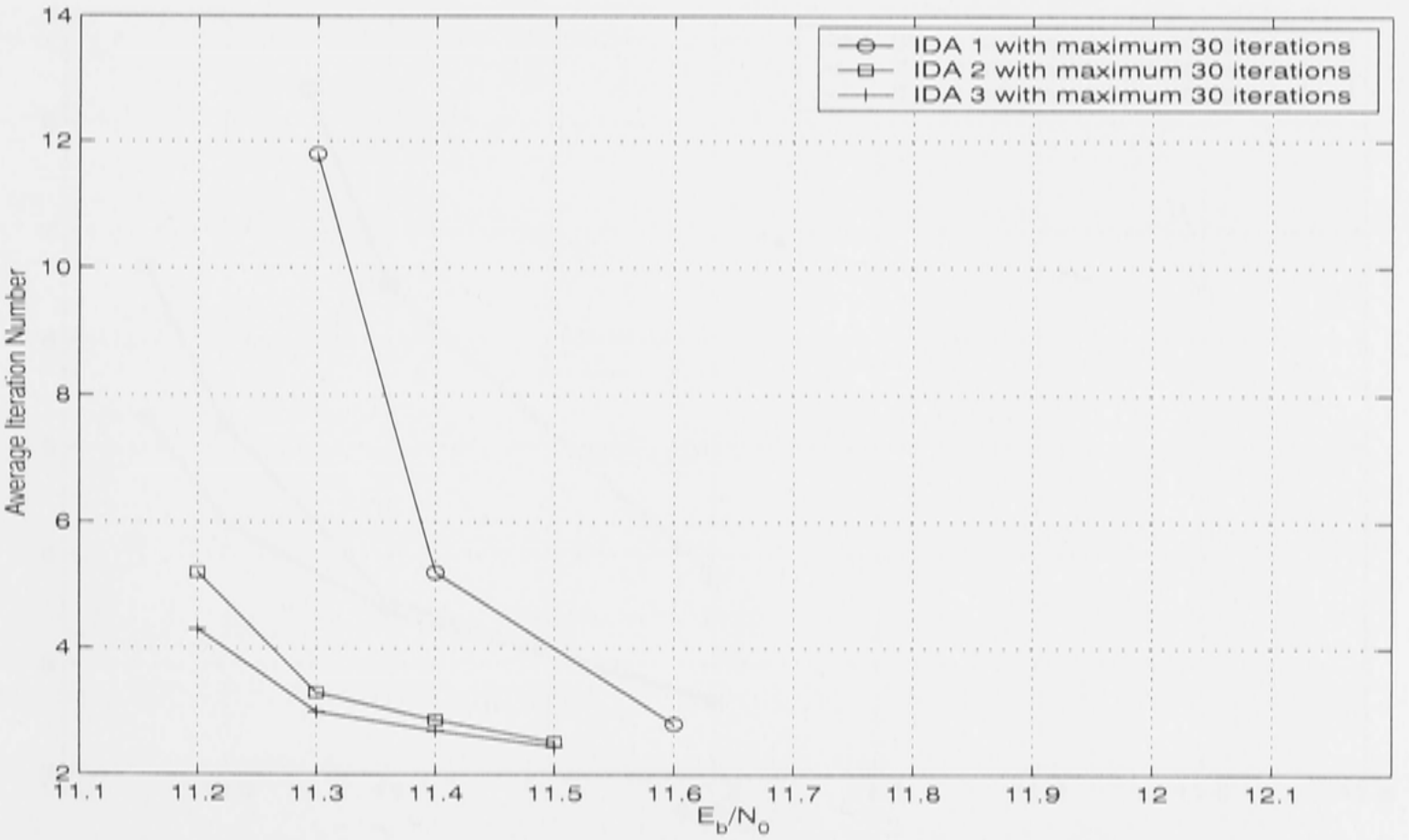
As a comparison, in Fig. 6-13 (a), we can see that, based on the double-parity-check structure, a performance of about 1.67 dB away from the Shannon limit at a $BER = 3.5 \times 10^{-5}$ is achieved by both the IVA and iterative min-sum/sum-product algorithms with a maximum of 30 iterations. With a maximum of 10 iterations, the gains achieved by the IVA are reduced by about 0.2 dB. For the iterative min-sum and sum-product algorithms with a maximum of 5 iterations, the gain losses are 0.2 dB and 0.1 dB, respectively.

Comparing the gains obtained in Fig. 6-12 (a) and Fig. 6-13 (a), we find that similar BER performance has been achieved by both single-parity-check and double-parity-check structures, in this example. In [62], a similar conclusion has been made for parity-concatenated convolutional codes, namely, a better performance can be achieved by using the double-parity-check structure than single-parity-check structure provided packet transmission with long block length is applied. However, as the packet length becomes short, the gains achieved by using the double-parity-check structure will be consequently reduced to one "point" which can be obtained by using the single-parity-check structure as well.

However, even though the same gains could be achieved by both structures, there are still some other aspects worthy of consideration. The first one is the scaling parameter in the IVA. According to the simulations, we find the performance of the IVA is more sensitive to the scale in the single-parity-check structure than in double-parity-check structure. That means it is more difficult to select an

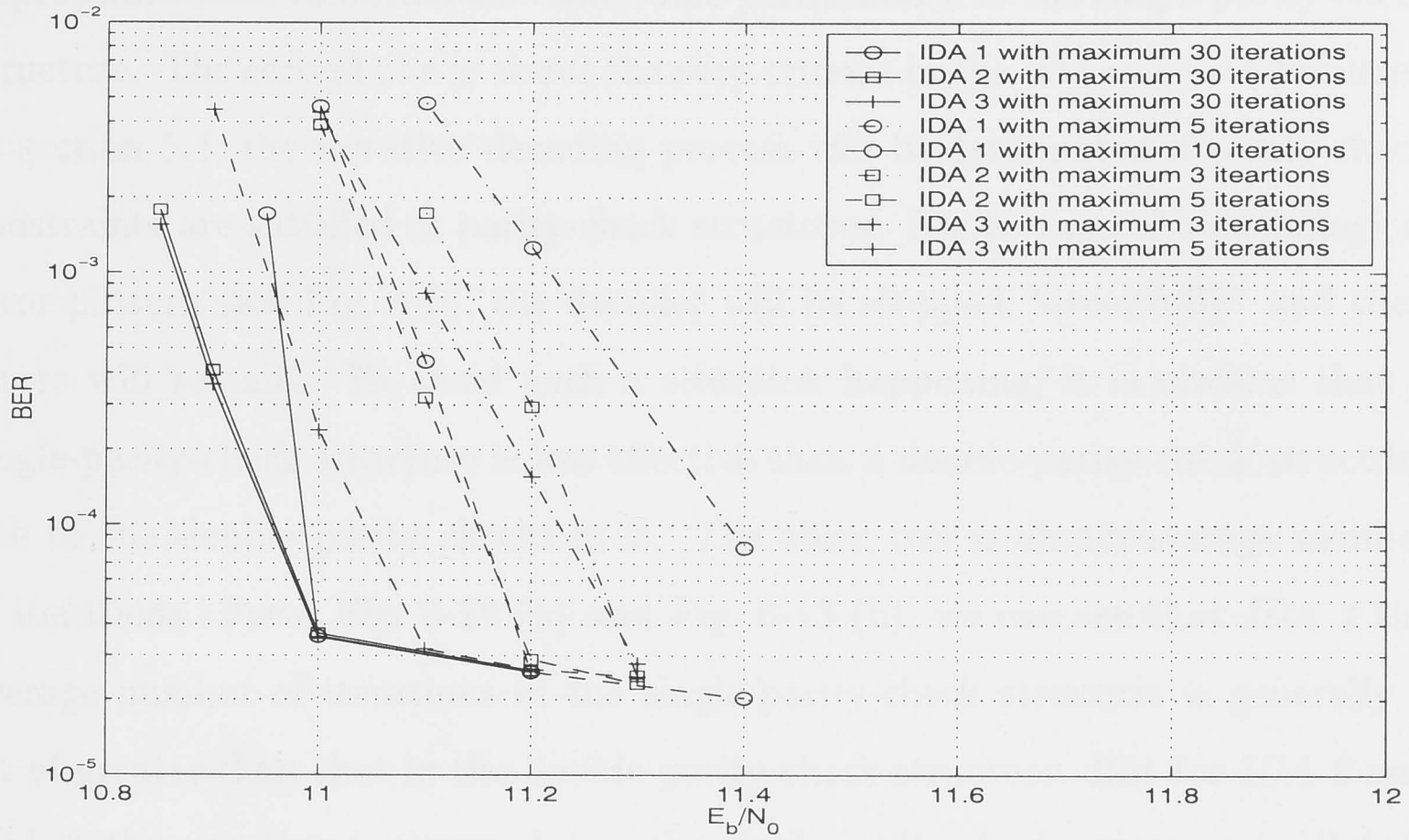


(a) Bit error rate with various maximum iterations

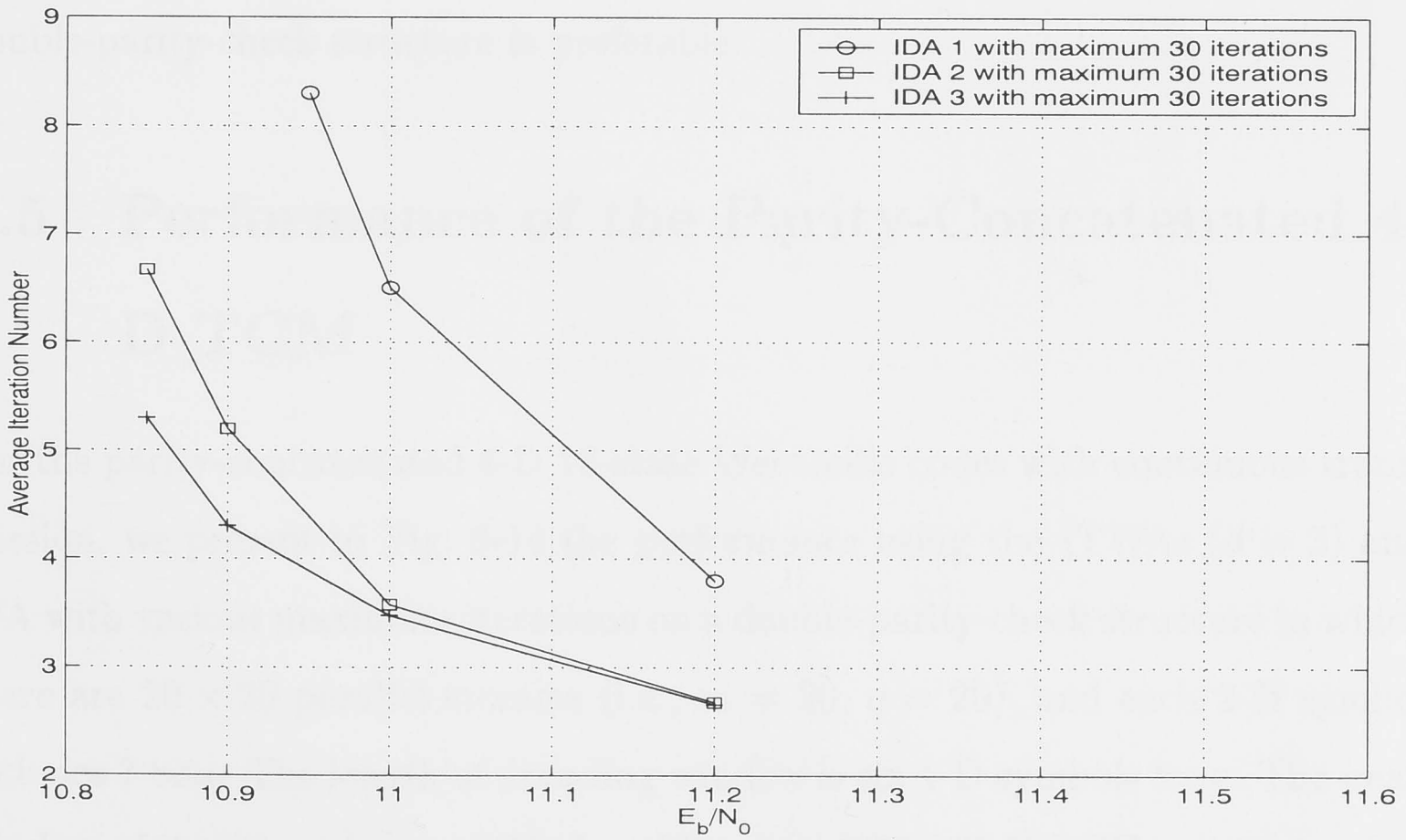


(b) Average number of iterations under a maximum of 30 iterations

Figure 6-12: Comparison of BER performance of the $\nu = 8$ trellis codes at a spectral efficiency of 5.800 bits/T using the IDAs for packet transmission with medium block length (single-parity-check structure, $m = 10$, $l = 200$, 2,000-symbol in each block)



(a) Bit error rate with various maximum iterations



(b) Average number of iterations under a maximum of 30 iterations

Figure 6-13: Comparison of BER performance of the $\nu = 8$ trellis codes at a spectral efficiency of 5.620 bits/T using the IDAs for packet transmission with medium block length (double-parity-check structure, $m = 10$, $q = 10$ and $l = 20$, 2,000-symbol in each block)

appropriate scale to obtain the achievable performance in the single-parity-check structure. The second one is about the stop criteria in the IDAs. As we discussed in section 5.4, the iterative decoding process will be terminated if parity-check constraints are satisfied in parity-check structures. But in case of a pathological error pattern (see Fig. 4-5), the decoder will be stopped "deceitfully" and then errors will remain. To avoid such a situation happening, it is obvious that a single-parity-check structure is less effective than a double-parity-check structure due to the double parity checks in it. The third one is about average number of iterations. From Fig. 6-12 (b) and Fig. 6-13 (b), we can see that *IDA 1* the average number of iterations in the single-parity-check structure is generally a bit of greater than that in the double-parity-check structure. But for *IDA 2* and *IDA 3*, the situation is reversed, i.e., the single-parity-check structure is slightly better than the double-parity-check structure in terms of the average number of iterations. Therefore, according to these three aspects discussed above, the double-parity-check structure is preferable.

6.5 Performance of the Parity-Concatenated 4-D TCM

For the parity-concatenated 4-D 16-state Wei trellis codes with continuous transmission, we present in Fig. 6-14 the performance using the ITWAs ($d = 3$) and IVA with various maximum iterations on a double-parity-check structure in which there are 20×20 parallel streams (i.e., $m = 20$, $q = 20$), and each 2-D symbol includes 7 bits. The length of decoding window is six 4-D symbols long. The peak number of iterations is set to 30. In addition, a (255,239,2) BCH code is used to lower the error floor dominated by the parallel transition errors. Therefore, the real rate now is 6.7871 bits/T.

In Fig. 6-14 (a), it is shown that at a $BER = 3.7 \times 10^{-6}$ about 2.7 dB gross gain or 2.2 dB net gain (2.4 dB away from the Shannon limit) is obtained. (If the 16-D shell mapping is applied, an additional 0.3–0.4 dB gain is expected.) It

is noted that both the ITWAs and IVA can achieve quite similar performance in this case. In Fig. 6-14 (b), we can see *IDA 1* with a maximum of 10 iterations outperforms by about 0.2 dB at a $BER = 3.5 \times 10^{-6}$ *IDA 2* and *IDA 3* with a maximum of 5 iterations. This is opposite to the cases in Figures 6-9, 6-11, 6-12 and 6-13.

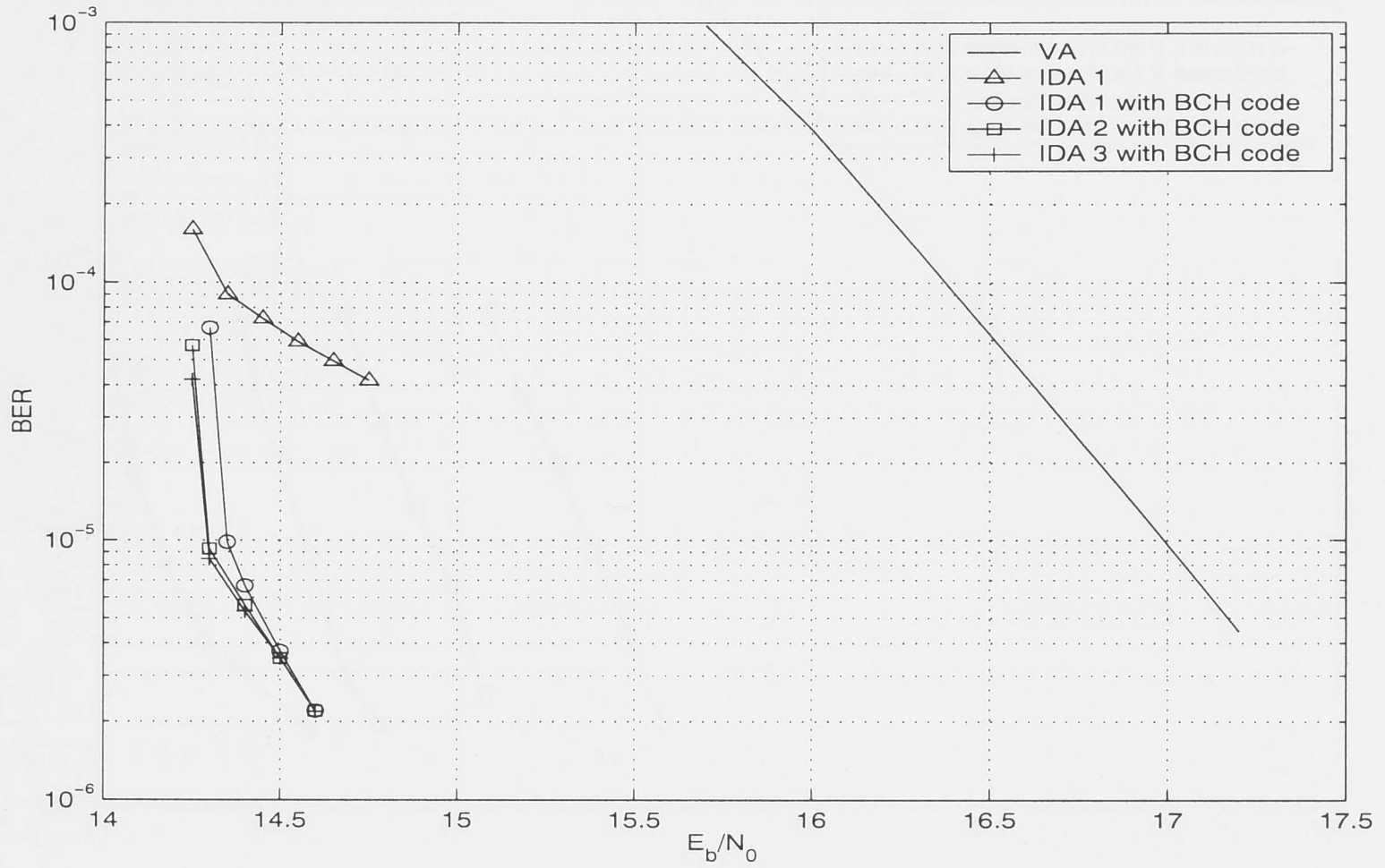
Similarly, in Table 6.2, we report the simulation results about the average number of iterations at several SNRs. Due to the high diversity (16 possible values) of a 4-D coded symbol node, the computational complexity of *IDA 2* and *IDA 3* is much higher than that of the IVA. Such a difference will be more significant for higher dimensional parity-concatenated trellis codes. Therefore, according to Fig. 6-14 and Table 6.2, we can see that the IVA is the best choice for decoding the parity-concatenated M-D TCM systems in terms of the tradeoff between BER performance and complexity.

IDA	SNR=14.4 dB	14.5 dB	14.6 dB
IDA 1	9.1	6.0	4.3
IDA 2	6.5	4.8	3.6
IDA 3	5.8	4.4	3.4

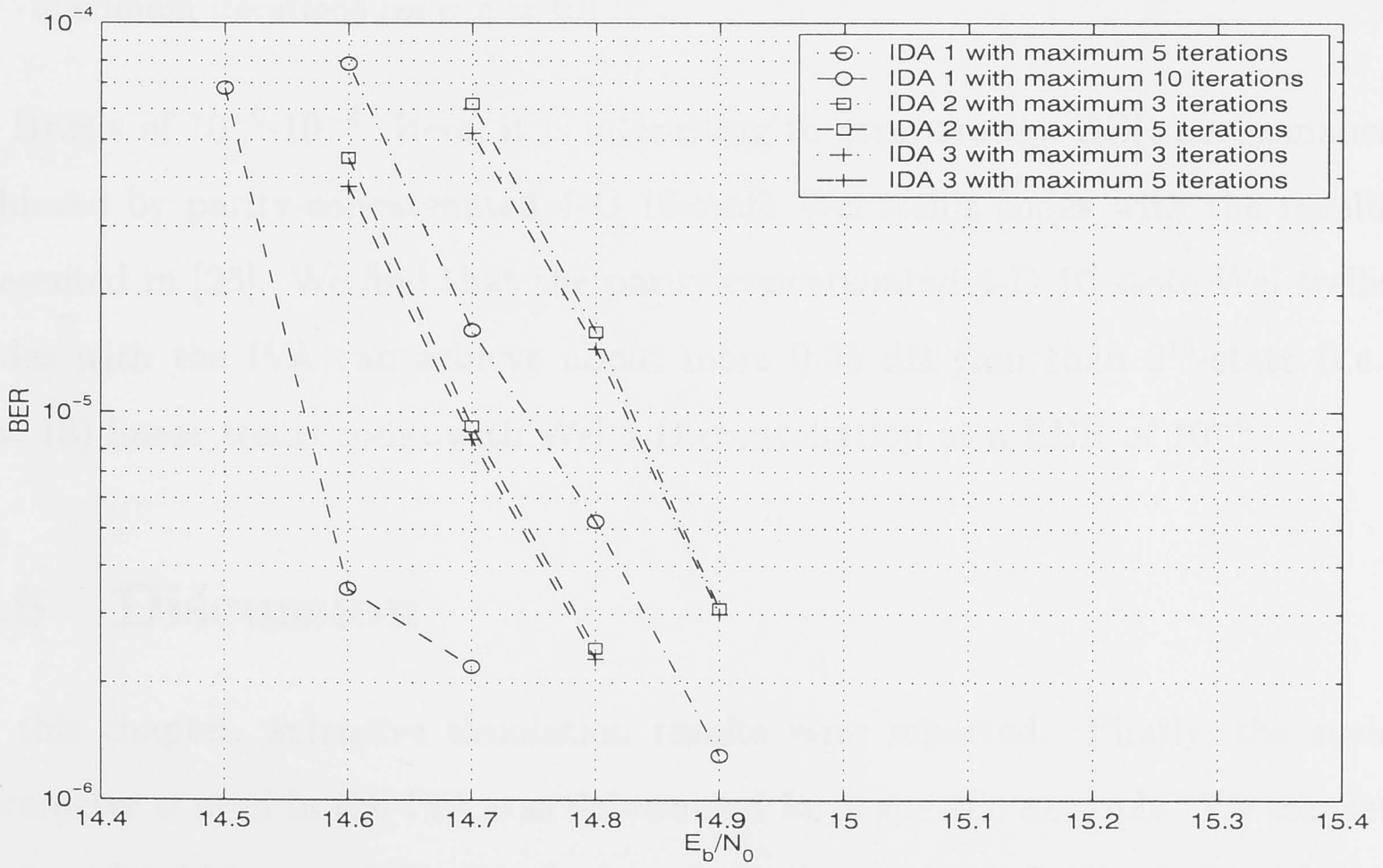
Table 6.2: Average number of iterations for the case of Fig. 6-14

In Fig. 6-15, we present the performance of the parity-concatenated 4-D 16-state Wei trellis codes using the IVA with a smaller block size, in which $m = q = 10$. It is shown that at a $BER = 4.0 \times 10^{-6}$ about 2.9 dB gross gain or 2.1 dB net gain has been obtained, which is similar with the case of Fig. 6-14. In Fig. 6-15, we also give the performances of the IVA with a maximum of 2, 3 and 5 iterations. It is shown that most of the gains can be achieved with 5 iterations.

In [26], Wang and Costello studied linear and nonlinear trellis codes for 4-D constellations at high spectral efficiencies with constraint lengths up to 19. Simulation results were presented showing that the "channel cutoff rate bound" [26] can be achieved using these new trellis codes with sequential decoding algorithm



(a) Bit error rate with a maximum of 30 iterations



(b) Bit error rate with various maximum iterations

Figure 6-14: Comparison of BER Performance of the parity-concatenated 4-D 16-state Wei trellis codes at a spectral efficiency of 6.7871 bits/T using the IVA and ITWAs ($m = q = 20$)

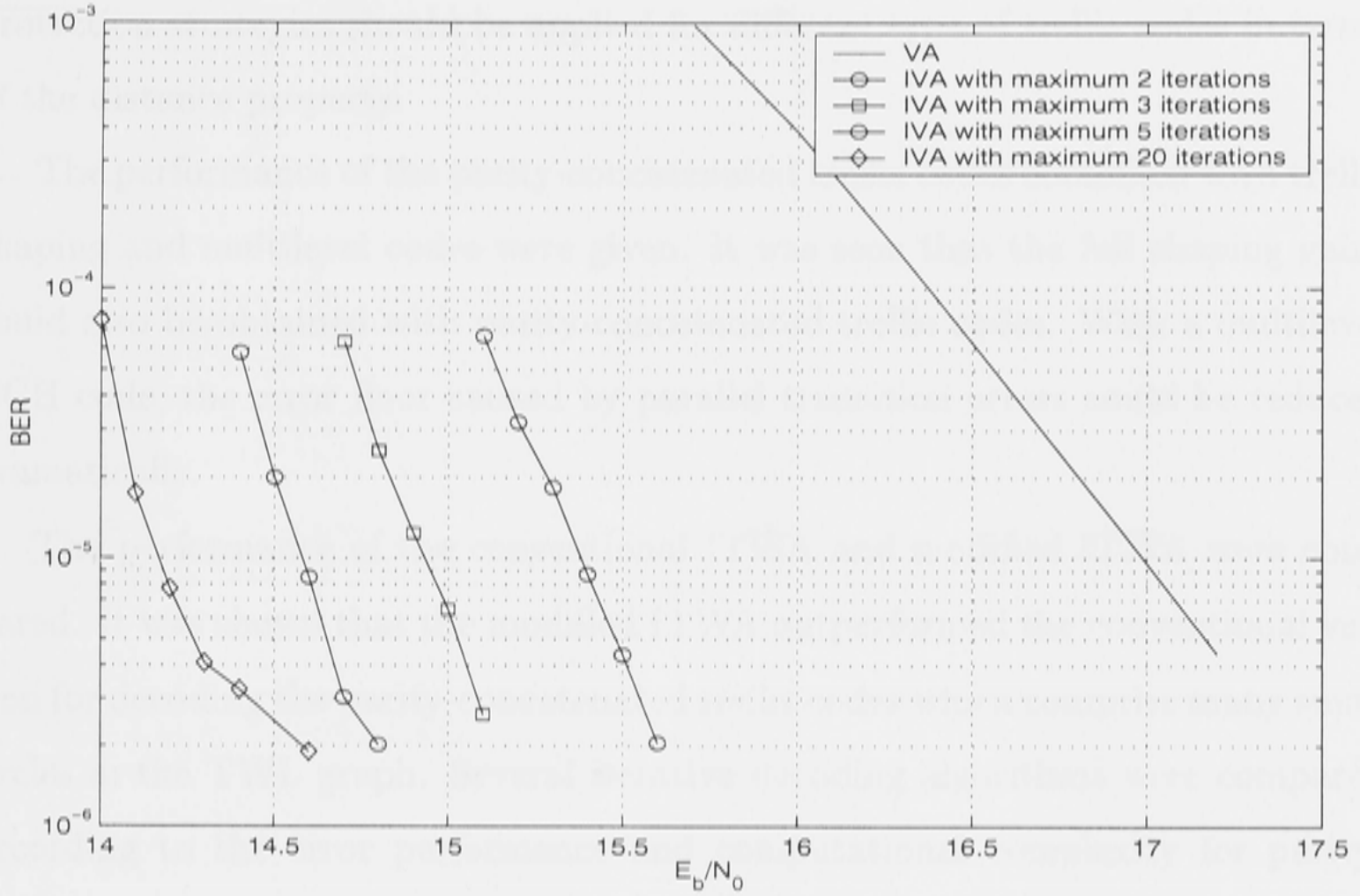


Figure 6-15: Performance of the parity-concatenated 4-D 16-state Wei trellis codes at a spectral efficiency of 6.6483 bits/T using the IVA with various maximum iterations ($m = q = 10$)

at BER's of 10^{-5}-10^{-6}. Here, it is interesting to compare the BER performance achieved by parity-concatenated 4-D 16-state Wei trellis codes with the results presented in [26]. We find that the parity-concatenated 4-D 16-state Wei trellis codes with the IVA can achieve about more 0.35 dB gain than 2^{18}-state (i.e., $\nu = 18$) linear trellis codes with Wei 4-D constellation at a BER of 10^{-6}.

6.6 Discussion

In this chapter, extensive simulation results were reported. Firstly, the scale parameter α used in the IVA was determined for a specific example. We can see that α should be carefully tuned to prevent the negative feedback from being enhanced and positive feedback from being weakened. We then presented and compared the performance of the IVA for the parity-concatenated 2-D trellis codes with partial and full protection, respectively. It is shown that different

protection strategies should be applied for different type of trellis codes in terms of the distance property.

The performance of the parity-concatenated trellis codes combined with trellis shaping and multilevel codes were given. It was seen that the full shaping gains could also be obtained with parity-concatenated trellis codes. With a multilevel BCH code, the error floor caused by parallel transition errors could be reduced dramatically.

The performance of the conventional ITWA and modified ITWA were compared. It was shown that the modified ITWA outperformed the conventional version for decoding the parity-concatenated trellis codes which comprise many small cycles in the TWL graph. Several iterative decoding algorithms were compared according to the error performance and computational complexity for parity-concatenated 2-D trellis codes as well as 4-D Wei trellis codes. Both continuous transmission and packet transmission with long, medium and short block lengths were considered. Through simulations, we found that generally the IVA is the best decoding algorithm for the parity-concatenated trellis codes in terms of the tradeoff between error performance and complexity.

Chapter 7

Robust Decoding Algorithms for Parity-Concatenated TCM

7.1 Introduction

In previous chapters, five iterative decoding algorithms have been studied for decoding the parity-concatenated 2-D and M-D TCM schemes. Using the iterative Viterbi decoding algorithm, we can apply parity-concatenated trellis codes in many current applications in which the VA is used.

When we studied the iterative decoding algorithms in previous chapters, the channel was disturbed by additive white Gaussian noise (AWGN) only. In reality, however, some other types of noise also exist simultaneously or during some time periods. In practice, the probability density function (PDF) of channel noises could change within a short time frame in an uncertain manner. This could be caused by either natural phenomenon such as lighting or man-made noises such as automotive noise and power-line noise.

If the receiver knows the noise PDF at each time slot or its *a priori* probability, the standard Viterbi algorithm or the *a posteriori* probability algorithm (such as sum-product algorithm) can achieve optimal performance. However, if the actual channel noise distribution differs from the noise model used to design the receiver, there can be significant performance degradation due to the model mismatch.

Several recent works have been developed for dealing with uncertainty in channel noises. In [75][106][112], some estimation methods are investigated for turbo codes with unknown channel noise. From a different perspective, in [66] Wei *et al.* developed the *minimax* concept and proposed a robust decoder for convolutional and turbo codes. The major difference between the methods in [66] and [75][106][112] is that in [66] inherently robust decoders are designed while the works in [75][106][112] focus on the algorithmic study of estimating the channel noise parameters. Therefore, the works in [66] and [75][106][112] are complementary. The noise model estimation can be used for the whole block, and then the *minimax* robust decoder is applied to fight further uncertainties within the block.

In this chapter, we aim to extend the *minimax* concept of [66] to deal with the robust decoding problem for trellis codes and parity-concatenated trellis codes in an uncertain noise environment. The situations where there is mixed noise within one symbol block as well as across several blocks are taken into account.

When the VA is used to decode trellis codes, it is clear that no knowledge about noise variance (channel SNR) is needed. Thus we just consider the robust decoding for various types of noises. Since the IVA has been developed for the parity-concatenated trellis codes, the procedure of estimating the noise variance can be omitted in the IVA as well.

7.2 System Model

We consider an additive noise channel in which the noise distribution is unknown. Fig. 7-1 shows the system block diagram with TCM.

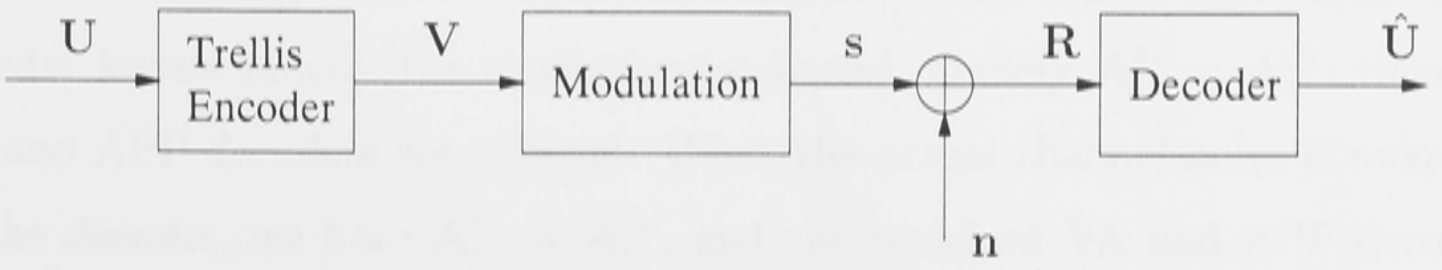


Figure 7-1: System block diagram for TCM

Let the input and output symbols of the trellis encoder be U_i and V_i respectively at time i. After modulation, we have the transmitted symbol s_i. The received symbol R_i at time i can then be expressed as

$$R_i = s_i + n_i. \tag{7.1}$$

where n_i is the channel noise whose PDF can change. In this thesis, we assume that the noise variables n_i are mutually independent for all time.

Let the noise model at time slot i depend on a parameter vector $\mathbf{A}_i$, which indicates the noise type and variance. The noise PDF of the channel (characterised by β_c) and the noise PDF used for decoder design (β_d) are then able to written as $p(n_i; \mathbf{A}_i^{\beta_c})$ and $p(n_i; \mathbf{A}_i^{\beta_d})$, respectively. For purpose of simulation, we will use a class of generalised noise density functions, namely,

$$\begin{aligned} p(n_i; \mathbf{A}_i) &= p(n_i; \sigma_i^2, \gamma_i) \\ &= \frac{\gamma_i}{2\sigma_i \sqrt{\alpha(\gamma_i)}\Gamma(1/\gamma_i)} \exp\left\{ -\frac{|n_i|^{\gamma_i}}{[\alpha(\gamma_i)]^{\gamma_i/2}\sigma_i^{\gamma_i}} \right\} \end{aligned} \tag{7.2}$$

where the parameter vector $\mathbf{A}_i$ equals (σ_i^2, γ_i), $\Gamma(\cdot)$ denotes the Gamma function, σ_i^2 is the variance of the noise and $\alpha(\gamma_i) = \frac{\Gamma(1/\gamma_i)}{\Gamma(3/\gamma_i)}$, at time i. This class of distributions is chosen because it is a large class which contains several commonly used distributions as special cases. When $\gamma = 2$, $p(n)$ is the Gaussian distribution function; when $\gamma = 1$, $p(n)$ given in equation (7.2) is the Laplace distribution function; when $\gamma = 0.5$, $p(n)$ denotes the square-root noise; when $\gamma = 4$, $p(n)$ represents the generalised Gaussian noise.

The PDF $p(n_i; \mathbf{A}_i^{\beta_d})$ is used to derive various branch metrics for optimal trellis-based decoding algorithms such as the standard VA and APP algorithms. If the decoder knows exactly the channel noise model, namely $\mathbf{A}_i^{\beta_c} = \mathbf{A}_i^{\beta_d}$, then the VA and APP decoders are optimal. When the actual channel noise is unknown to the decoder, we have $\mathbf{A}_i^{\beta_c} \neq \mathbf{A}_i^{\beta_d}$, and the standard VA and APP decoders are no longer optimal due to the mismatch of noise parameters between channel and decoder. Therefore, to achieve optimal performance for the decoder, we need to know the exact noise PDF, in our case, the parameter vector β_c. In reality,

however, this is often difficult because of the existence of impulsive environmental noise such as lighting and man-made noise such as power-line noise, automotive noise, etc. The impulsive noise often lasts a short period of time, which makes noise estimation a very difficult task.

7.3 Robust Decoder Design for Trellis Codes

The decoder design clearly depends on how much information the receiver has about the channel noise. Before we start to discuss the robust decoding algorithm for trellis codes, let us first review the following two cases: (a) the decoder knows the exact noise type for each time slot, and (b) the decoder just knows the *a priori* probability of each noise type. Similar analyses for convolutional and turbo codes have been reported in [66].

7.3.1 Optimal decoder with perfect knowledge of channel noise

If we have perfect knowledge of the channel noise, i.e., the exact noise PDF parameter vector $\mathbf{A}_i^{\beta_d} = \mathbf{A}_i^{\beta_c}$ at time slot i is known, then the optimal maximum likelihood decision (MLD) rule is to select the information symbol sequence $\hat{\mathbf{U}}_{1,l} = (\hat{U}_1, \hat{U}_2, \cdots, \hat{U}_l)$, which minimises the state metric:

$$\hat{\mathbf{U}}_{1,l} = \arg\min_{\mathbf{U}_{1,l}} \left\{ \sum_{i=1}^{l} -\log\left[p(R_i - \hat{s}_i; \mathbf{A}_i^{\beta_d})\right] \right\}. \tag{7.3}$$

The conventional Viterbi algorithm can be used to find the best path. Suppose $R_i = R_{i,I} + jR_{i,Q}$ and $\hat{s}_i = \hat{s}_{i,I} + j\hat{s}_{i,Q}$, where $R_{i,I}$, $\hat{s}_{i,I}$ and $R_{i,Q}$, $\hat{s}_{i,Q}$ denote the real and imaginary parts of the complex-valued R_i and $\hat{s}_i$ respectively, then the only modification of (7.3) will be replacing $-\log\left[p(R_i - \hat{s}_i; \mathbf{A}_i^{\beta_d})\right]$ by $|R_{i,I} - \hat{s}_{i,I}|^{\gamma_i} + |R_{i,Q} - \hat{s}_{i,Q}|^{\gamma_i}$, where γ_i denotes the noise type. Obviously, if $\gamma = 2$ (Gaussian noise), the branch metric will be $|R_i - \hat{s}_i|^2$. To obtain this optimal decoder, we need to estimate the PDF of the noise from symbol duration to symbol duration, which could be very difficult in practice.

7.3.2 Optimal decoder with knowledge of channel noise *a priori* probability

If we know the *a priori* probability of the noise PDF parameter vector $\mathbf{A}_i^{\beta_c}$, denoted as $P(\mathbf{A}_i^{\beta_c})$, but we do not know the exact noise PDF at each time slot i, then the optimal decoder can be obtained as follows.

The optimal maximum likelihood decision rule is to select the information symbol sequence $\hat{\mathbf{U}}_{1,l}$ which minimises the state metric:

$$\hat{\mathbf{U}}_{1,l} = \arg\min_{\mathbf{U}_{1,l}} \left\{ \sum_{i=1}^{l} -\log\left[P_{ave}(R_i - \hat{s}_i)\right] \right\}, \tag{7.4}$$

where

$$P_{ave}(n_i) = \sum_{\mathbf{A}_i^{\beta_c}} P(\mathbf{A}_i^{\beta_c}) p(n_i; \mathbf{A}_i^{\beta_c}). \tag{7.5}$$

The VA can also be used to select the best path, with the branch metric replacing $-\log\left[P_{ave}(R_i - \hat{s}_i)\right]$ by $|R_{i,I} - \hat{s}_{i,I}|^{\gamma_i} + |R_{i,Q} - \hat{s}_{i,Q}|^{\gamma_i}$.

If we do not know the above *a priori* probability $P(\mathbf{A}_i^{\beta_c})$, and use a fixed noise PDF parameter vector $\mathbf{A}_i^{\beta_d}$ for all time slots, we end up with a mismatched decoder which could perform much worse that the matched one, as will be shown in the numerical results. Therefore, the key task of this chapter is to design a robust decoder to prevent significant performance loss due to the noise model mismatch. To achieve this, the *minimax* concept [66][94] is extended from binary cases (convolutional codes, turbo codes) to non-binary cases (trellis codes, parity-concatenated trellis codes).

7.3.3 Minimax robust decoder

The *minimax* criterion is to minimise the worst possible error performance over a family of possible noise PDFs [94]. In this section, we will present the minimax robust decoder design based on two types of criterion: path likelihood ratio and branch likelihood ratio [66].

A. Minimax robust decoder based on path likelihood ratio

First let us get through a simple example to see how the *minimax* concept is applied based on the path likelihood ratio. Fig. 7-2 shows a 4-state trellis code with a single error event.

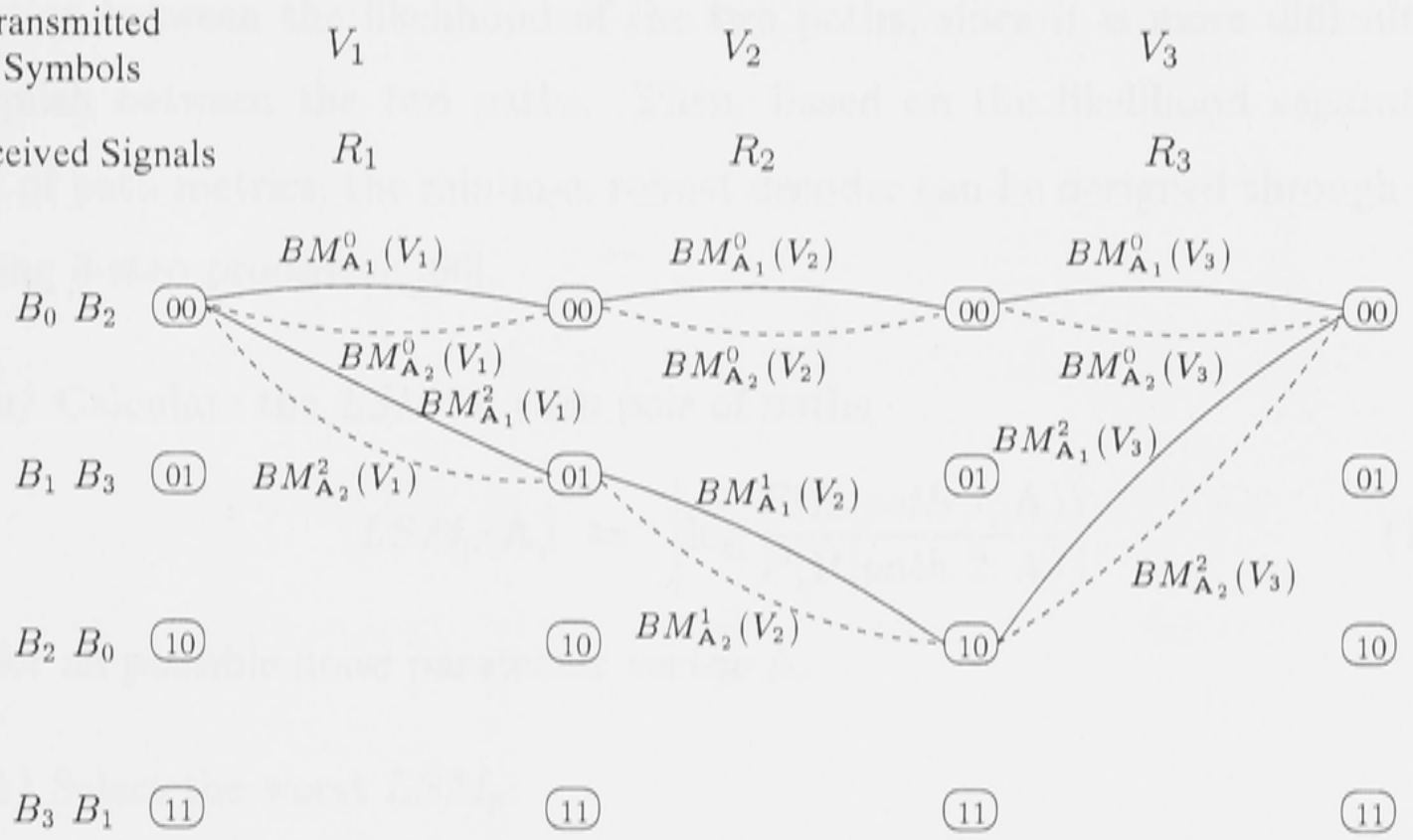


Figure 7-2: Single error event with two types of noise in a 4-state trellis code

Suppose that there is a transmitted symbol vector $\mathbf{V} = (V_1, V_2, V_3)$ and received signal vector $\mathbf{R} = (R_1, R_2, R_3)$, as shown in Fig. 7-2. For this single error event case, there are two possible symbol vectors $\mathbf{V}^1$ and $\mathbf{V}^2$ corresponding to *path 1* and *path 2*, respectively. At each transmitted symbol, the channel noise $\mathbf{A}_i^{\beta_c}$ could be either $\mathbf{A}^1$ or $\mathbf{A}^2$. We can calculate two branch metrics (the logarithm of the likelihood function) based on both noise parameter vectors. Thus, we have branch metric 1 (BM_1) and branch metric 2 (BM_2) for each branch, as shown as the dashed and solid lines in Fig. 7-2.

Similar to [66], we define the *Likelihood Separation Metric (LSM)* for path metrics of *path 1* and *path 2*.

$$LSM_p = \left| \log \frac{P(\mathbf{R}|\mathbf{V}^1; \mathbf{A}^i)}{P(\mathbf{R}|\mathbf{V}^2; \mathbf{A}^j)} \right|, \quad i, j = 1, 2. \tag{7.6}$$

We find that the decoder matched to the noise model which has minimum LSM_p

will be more likely to have poorer performance. This intuitive observation is generally true for the VA and APP decoders under realistic channel noise environments. Therefore, according to the *minimax* concept, we need to select a possible noise pattern corresponding to the worst case which has minimum separation between the likelihood of the two paths, since it is more difficult to distinguish between the two paths. Then, based on the likelihood separation metric of path metrics, the minimax robust decoder can be designed through the following 3-step procedure [66].

step (a) Calculate the LSM for each pair of paths

$$LSM_p(\mathbf{A}) = \left| \log \frac{P(\mathbf{R}|path\ 1; \mathbf{A})}{P(\mathbf{R}|path\ 2; \mathbf{A})} \right|, \tag{7.7}$$

for all possible noise parameter vector $\mathbf{A}$.

step (b) Select the worst LSM_p:

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}}(LSM_p(\mathbf{A})), \tag{7.8}$$

step (c) Decode based on the likelihood ratio test using the noise model $\hat{\mathbf{A}}$ from *step (b)*.

To compute equation (7.8) we need to evaluate all possible $\mathbf{A}_i$ combinations, which could be very complicated. For the example given in Fig. 7-2, we need to calculate LSM_p for $4^3 = 64$ cases.

Furthermore, in a VA or APP decoder, normally there are many possible error events. Thus one set of $\mathbf{A}$ which minimises the likelihood separation for one error event often cannot minimise the likelihood separation for other error events. This makes the calculation even more complicated.

B. Minimax robust decoder based on branch likelihood ratio

In this subsection, a sub-optimal, practically feasible minimax robust decoder based on the theories from the previous subsection will be proposed. This is the key algorithm which will be discussed extensively.

In trellis-based decoding algorithms, if we select the minimum likelihood separation for each time instance, then it will likely minimise the likelihood separation of most error events. Based on this intuitive observation, the following algorithm was proposed in [66]:

First, for the case of Fig. 7-2, we define the *LSM* for each pair of branch metric that start from or stop at the same state:

$$LSM_b(\mathbf{A}_i) = \left| \log \frac{P(R_i|V_i \in B_0 \ (or \ B_1); \mathbf{A}_i)}{P(R_i|V_i \in B_2 \ (or \ B_3); \mathbf{A}_i)} \right|. \tag{7.9}$$

The decoder then makes a decision based on the following procedure:

step (a) Calculate the $LSM_b(\mathbf{A}_i)$ for all possible noise parameter vectors $\mathbf{A}_i$.

step (b) Select the parameter vector $\hat{\mathbf{A}}_i$ with minimum LSM_b for each received signal:

$$\hat{\mathbf{A}}_i = \arg\min_{\mathbf{A}_i}[LSM_b(\mathbf{A}_i)], \tag{7.10}$$

step (c) Use the above parameter vector to compute the branch metric in a standard decoding algorithm. In the VA, just replace the branch metric calculation by $-\log P(R_i|V_i; \hat{\mathbf{A}}_i)$.

The important implication of *step (c)* is that we do not need to change the trellis optimisation part of traditional decoders; all we need to change is the branch metric, or in most systems, a metric table in the decoder. If we know the possible noise types beforehand, this metric table can be calculated off-line so there will be no additional complexity. Therefore, the minimax robust decoders will be an easy extension to current decoder implementation.

For the half rate convolutional code [66] or the case of Fig. 7-2, we note that the LSM_b can be straightforwardly calculated from a pair of branch metrics that start or stop in the same state. In the trellis codes, however, if there are $\tilde{k} \geq 2$ encoded bits, then there are $\eta = 2^{\tilde{k}}$ transitions that start or stop in the same state. A case with $\eta = 4$ is shown in Fig. 7-3.

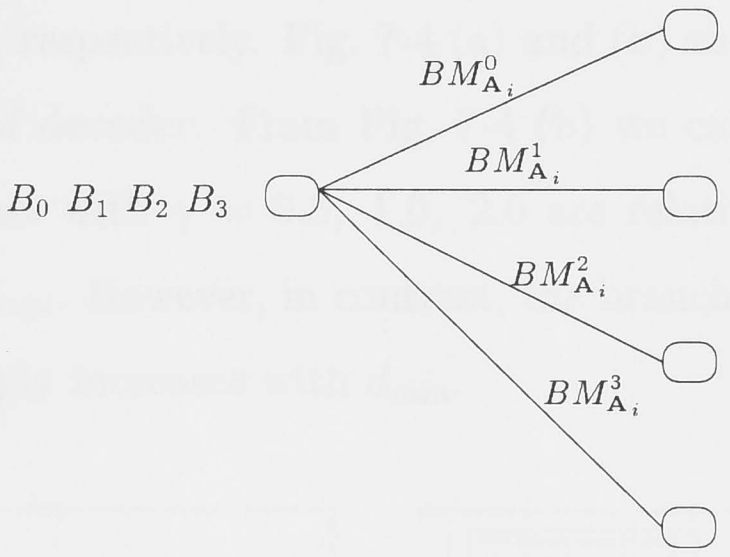


Figure 7-3: Trellis diagram with four transitions from one state

For this case, we may calculate the average LSM of these η branch metrics as

$$\overline{LSM_b}(\mathbf{A}_i) = \frac{1}{\binom{\eta}{2}} \sum_{l,j=1;l\neq j}^{\eta} \left| \log \frac{P(R_i|V_i \in B_l; \mathbf{A}_i)}{P(R_i|V_i \in B_j; \mathbf{A}_i)} \right|, \tag{7.11}$$

which can then be used to determine the noise parameter vector. It is clear that the computational complexity of calculating $\overline{LSM_b}(\mathbf{A}_i)$ is exponentionally increasing with η. Therefore, as a simplification, we can use the following LSM_b to detect the noise pattern.

$$LSM_b(\mathbf{A}_i) = \left| \log \frac{\max\{P(R_i|V_i \in B_l; \mathbf{A}_i)\}}{\min\{P(R_i|V_i \in B_j; \mathbf{A}_i)\}} \right|, \quad l, j = 1, \cdots, \eta, \tag{7.12}$$

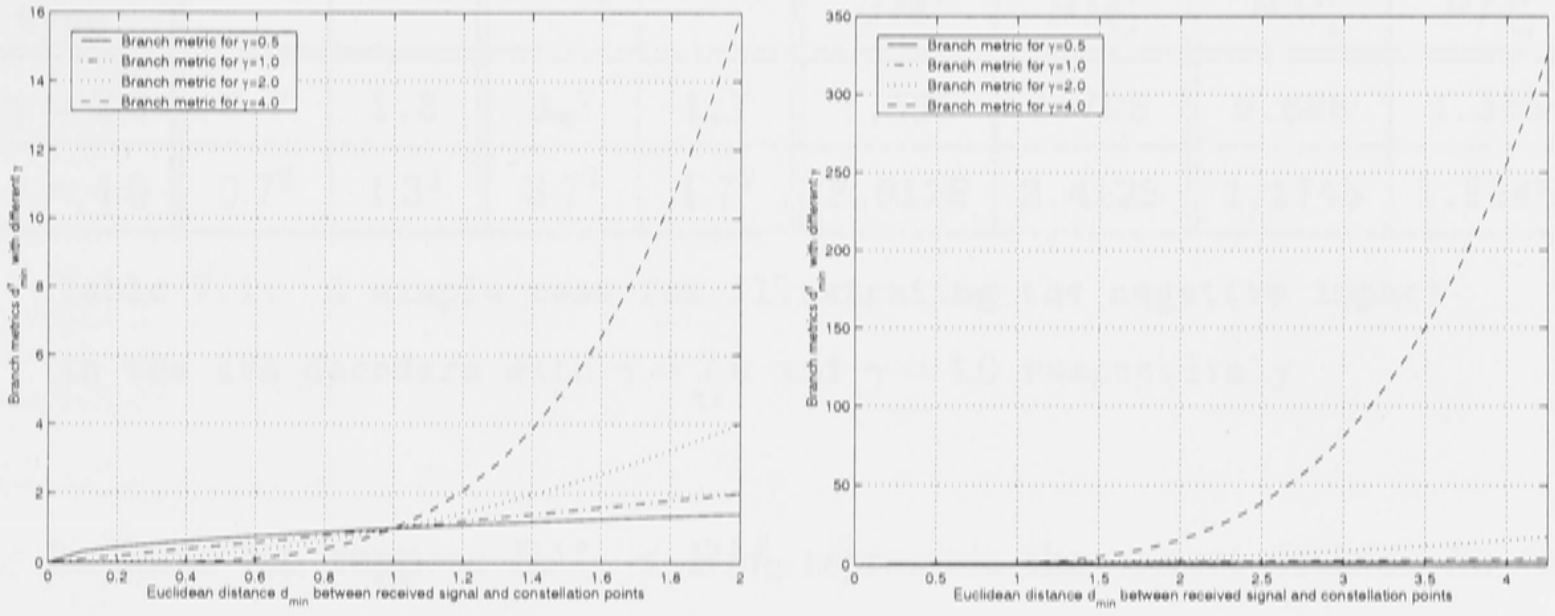
where the maximum and minimum branch metrics are selected to calculate the LSM_b. Through simulation, we find that such a simplification has negligible impact on the error performance.

7.3.4 Scale consideration in the robust IVA

In chapters 4 and 5, we have explained that the scale parameter α is crucial to the performance of the IVA. When the IVA is employed in the decoder corresponding to a different type of noisy channel (γ is set to 0.5, 1.0, 2.0, and 4.0 in this chapter), should we adjust α according to the type of decoder?

In a 1-D signal constellation, suppose the squared Euclidean distance between the received signal and one signal point is d^2_{min}, then the branch metrics used in the VA or IVA are $\sqrt{d_{min}}$, d_{min}, d^2_{min}, d^4_{min} for the decoders with

$\gamma = 0.5,\ 1.0,\ 2.0,\ 4.0$, respectively. Fig. 7-4 (a) and (b) show the branch metrics for these four types of decoder. From Fig. 7-4 (b) we can see that the branch metrics of the decoders with $\gamma = 0.5,\ 1.0,\ 2.0$ are relatively slowly increasing with the increase of d_{min}. However, in contrast, the branch metric of the decoder with the $\gamma = 4.0$ sharply increases with d_{min}.



(a) Branch metrics with $d_{min} \in (0.0, 2.0)$

(b) Branch metrics with $d_{min} \in (0.0, 4.25)$

Figure 7-4: Branch metrics d^{γ}_{min} in the decoders with $\gamma = 0.5,\ 1.0,\ 2.0$ and 4.0

In the iterative decoding process of the IVA, we know that one branch metric (BM_1) is updated by adding another scaled branch metric (BM_s) to it. Whether the *extrinsic* information present in the BM_s is positive or negative with respect to BM_1 depends on the "selector" $\widehat{W}$ which is calculated from the values of other decoded symbols. If $\widehat{W}$ is wrong (i.e., $\widehat{W} \neq W$), then BM_s has an adverse impact on BM_1. Therefore, the scale parameter α is necessary. Suppose one updated branch metric is $BM_1' = BM_1 + \alpha \cdot BM_s$, and also suppose BM_s has a negative impact on BM_1. For the decoder with $\gamma = 2.0$, an appropriate α is chosen to suppress such a negative impact. For the decoder with $\gamma = 4.0$, however, if the same scale is applied, the negative impact probably cannot be eliminated due to the rapid "amplification" of any negative impact (see Fig. 7-4).

Table 7.1 gives a simple example that illustrates amplification of negative

influence in the decoder with $\gamma = 4.0$, compared with the decoder with $\gamma = 2.0$. In this case, the two branch metrics BM_1 and BM_2 correspond to one received signal, and another two branch metrics BM_s^1 and BM_s^2 are fed back as *extrinsic* metrics to update BM_1 and BM_2, respectively.

Decoder type	BM_1	BM_2	BM_s^1	BM_s^2	$\alpha = 0.25$		$\alpha = 0.05$	
					BM_1'	BM_2'	BM_1'	BM_2'
$\gamma = 2.0$	0.7	1.3	3.7	1.7	1.625	1.725	0.885	1.385
$\gamma = 4.0$	0.7^2	1.3^2	3.7^2	1.7^2	3.9125	2.4125	1.1745	1.8345

Table 7.1: A simple case for illustrating the negative impact in the IVA decoders with $\gamma = 2.0$ and $\gamma = 4.0$ respectively

In Table 7.1, suppose $BM_1 < BM_2$ represents the correct decision for the received signal, and $BM_s^1 > BM_s^2$ means a negative impact on that received signal. In the decoder with $\gamma = 2.0$, the relationship between BM_1' and BM_2' remains unchanged with α equalling either 0.25 or 0.05, namely, BM_1' is still smaller than BM_2'. In the decoder with $\gamma = 4.0$, however, due to amplification of the negative impact (here it is about 5 times), BM_1' is larger than BM_2' if α equals 0.25, which reverses the relationship between BM_1 and BM_2, and consequently will degrade the error performance of the IVA. Therefore, α must be decreased in this case. We can see BM_1' is smaller than BM_2' if $\alpha = 0.05$. That means amplification of negative impact in the decoder with $\gamma = 4.0$ has been reduced by using a smaller scale.

However, if α is small, there is another problem arising in the IVA. As we mentioned in chapters 4 and 6, if the scale is too small and the *extrinsic* metrics BM_s contain positive information, the positive impact from *extrinsic* information will be too weak to assist the decoder. Just like the example in Table 7.1, now suppose $BM_1 < BM_2$ represents the wrong decision for the received signal, and $BM_s^1 > BM_s^2$ implies a positive impact on the received signal. If α equals 0.05, the *extrinsic* information contained in BM_s^1 and BM_s^2 will be too weak for BM_1

and BM_2.

With the power four operation in the decoder with $\gamma = 4.0$, it is natural that amplification of positive impact can also result. Still using the example in Table 7.1 except exchanging the values of BM_s^1 and BM_s^2 (which now represent positive information), we can get $BM_1' = 1.2125$ and $BM_2' = 5.1125$ with $\alpha = 0.25$, respectively. However, due to the comparison-selection procedure (i.e., selecting the minimum value) in the IVA, amplifications of positive information tend to be suppressed. Therefore, in this case the decoder with $\gamma = 4.0$ does not benefit much from amplification of positive information.

From Fig. 7-4, we can see that the problem (i.e., amplification of negative impact through feedback of *extrinsic* branch metrics) is not significant in constellations with small size, such as 8-PSK or 16-QAM. However, in constellations with medium or large size, we have to consider the negative impact caused by amplification. Therefore, in terms of the discussion above and Fig. 7-4 (b), we can conclude that the scales in the decoders with $\gamma = 0.5$ and $\gamma = 1.0$ could be slightly larger than the one in the decoder with $\gamma = 2.0$, but the scale in the decoder with $\gamma = 4.0$ must be much smaller than that of the decoder with $\gamma = 2.0$.

Figure 7-5 illustrates an experimental result, in which the Ungerboeck 16-state trellis code $(h^0, h^1, h^2) = (23, 04, 16)$ with eight subsets [32] are examined under four types of noise to show the distribution of branch metrics in a 256-QAM constellation. To be compatible with Fig. 7-4, the branch metrics presented in Fig. 7-5 are the minimum Euclidean distance in each subset.

From Fig. 7-5, we can see that in the $\gamma = 4.0$ channel more than 50% minimum Euclidean distances are located in the range between 3.0 and 4.25, which implies an amplification of 9 times and 18 times respectively, compared with the $\gamma = 2.0$ channel. Thus, we may anticipate that in the decoder with $\gamma = 4.0$, the performance of the IVA could probably be worse than that of other decoders with $\gamma = 0.5,\ 1.0$ and 2.0, even though it is matched to the $\gamma = 4.0$ channel. It is hard to mathematically determine how much the amplification of negative or

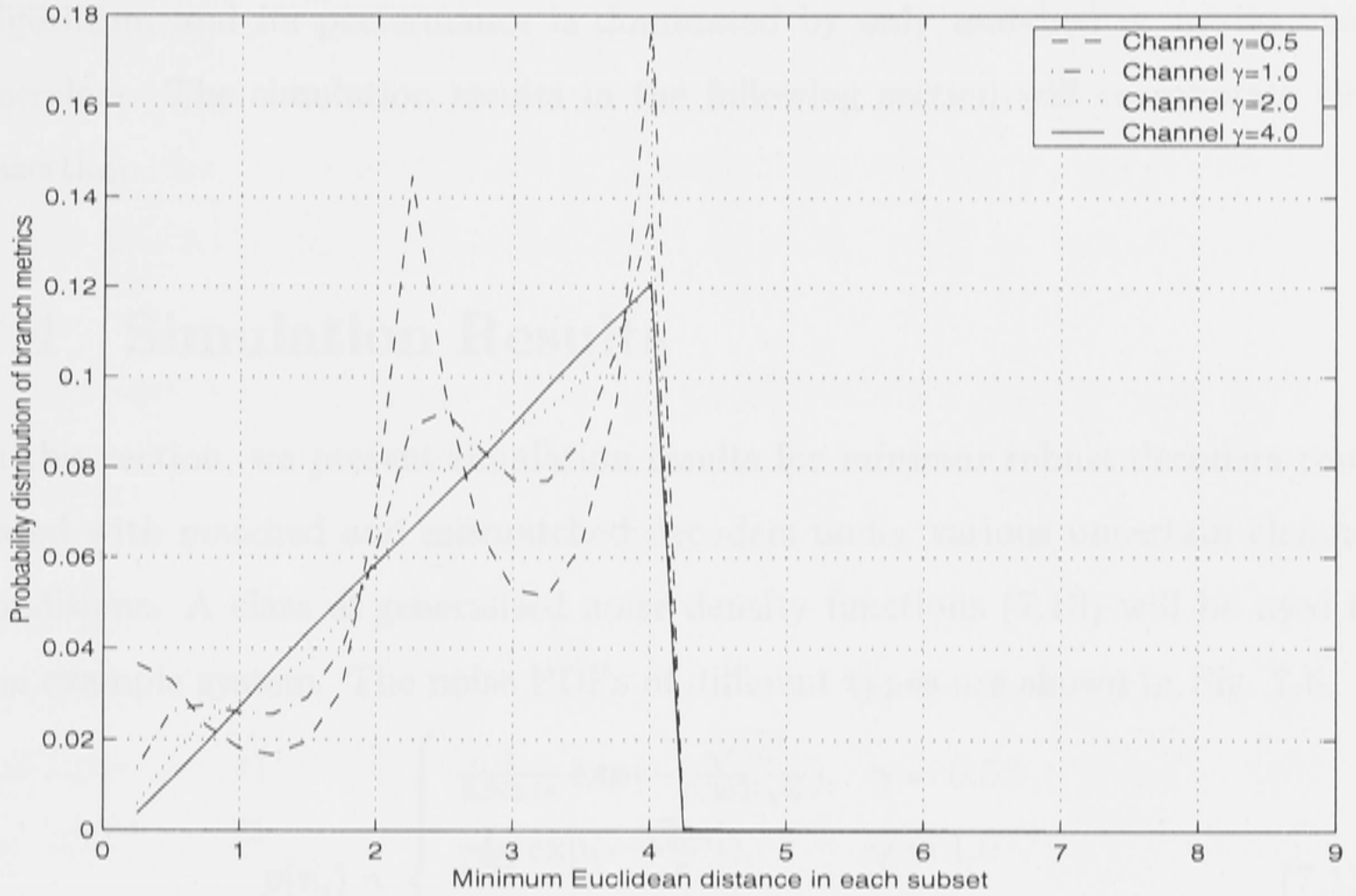


Figure 7-5: Distribution of branch metrics with different type of noise at SNR=11.2 dB

positive impact affects the performance of the IVA. In the following section, our simulation results will show how the performance of the IVA is affected jointly by the degree of mismatch between decoders and channel, as well as the amplification of negative or positive impact from fedback *extrinsic* information.

In Chapter 5, we have studied the modified iterative min-sum algorithm. It contains a procedure of information normalisation which is used to scale down the *extrinsic* information fed into the same symbol node several times. The scaling parameter d in the modified iterative min-sum algorithm is determined by the length of the error event with minimum distance minus the number of symbols in the error path which agrees with the correct path. The *extrinsic* information fed back into a symbol node is calculated from the weights of other symbol nodes through the min-sum algorithm. The amplification of negative or positive impact in the weight of a symbol node has been greatly mitigated through the min-sum algorithm (add-compare-select operation). Therefore, the phenomenon we discussed above for the IVA should not appear in the modified iterative min-sum

algorithm, and its performance is dominated by only matched or mismatched decoders. The simulation results in the following section will corroborate this assertion.

7.4 Simulation Results

In this section, we present simulation results for *minimax* robust decoders compared with matched and mismatched decoders under various uncertain channel conditions. A class of generalised noise density functions (7.13) will be used in the example system. The noise PDFs of different types are shown in Fig. 7-6.

$$p(n_i) = \begin{cases} \frac{1}{0.3651\sigma}\exp(-\frac{\sqrt{|n_i|}}{0.3021\sqrt{\sigma}}), & \gamma = 0.5 \\ \frac{1}{\sqrt{2}\sigma}\exp(-\frac{\sqrt{2}|n_i|}{\sigma}), & \gamma = 1.0 \\ \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{n_i^2}{2\sigma^2}), & \gamma = 2.0 \\ \frac{1}{3.1182\sigma}\exp(-\frac{n_i^4}{8.7539\sigma^4}). & \gamma = 4.0 \end{cases} \tag{7.13}$$

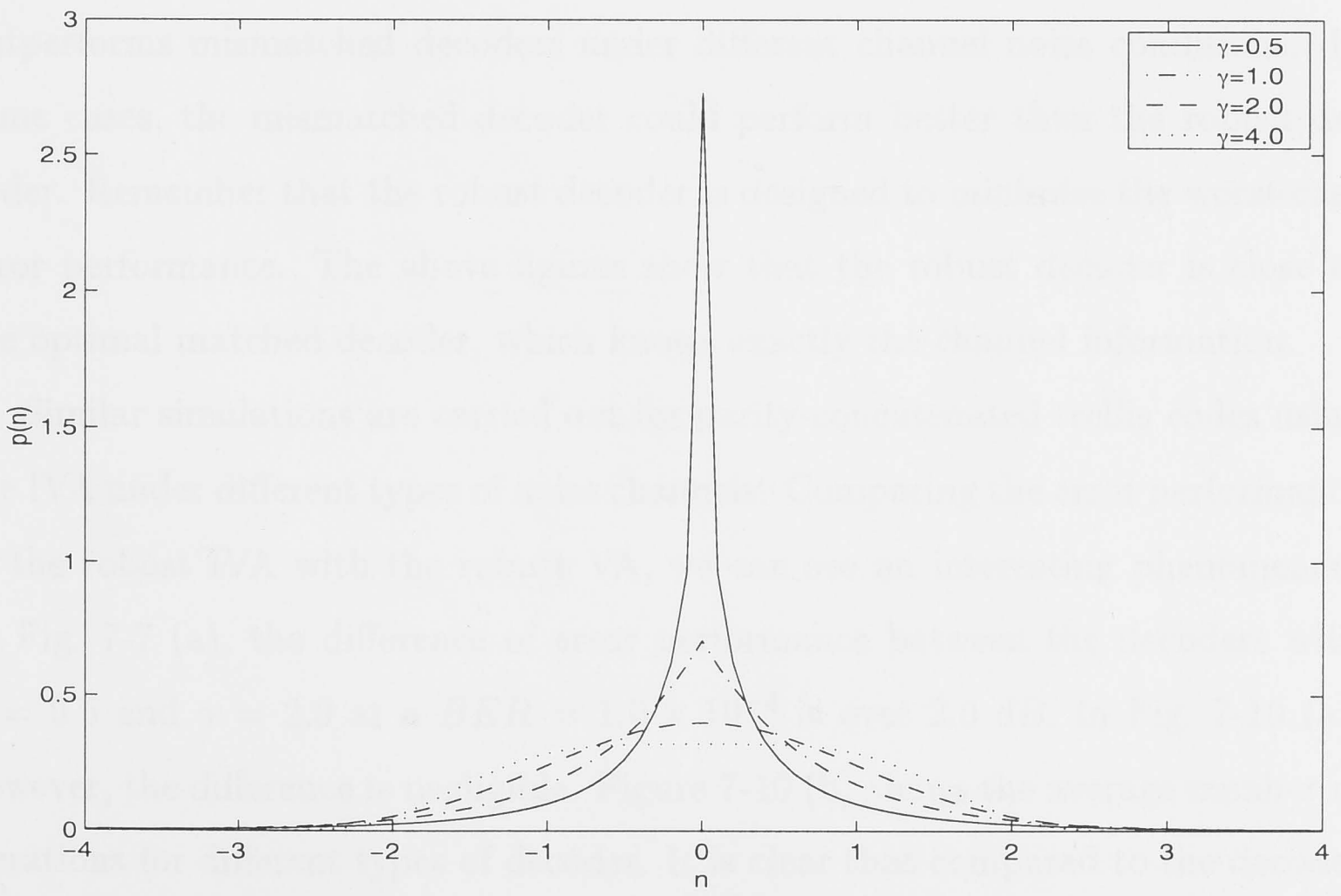


Figure 7-6: Different types of noise with $\gamma = 0.5,\ 1.0,\ 2.0,\ 4.0$ ($\sigma = 1.0$)

The trellis code under study is a 16-state Ungerboeck code with $(h^0, h^1, h^2) = (23, 04, 16)$ using a 128-QAM constellation. For the parity-concatenated trellis

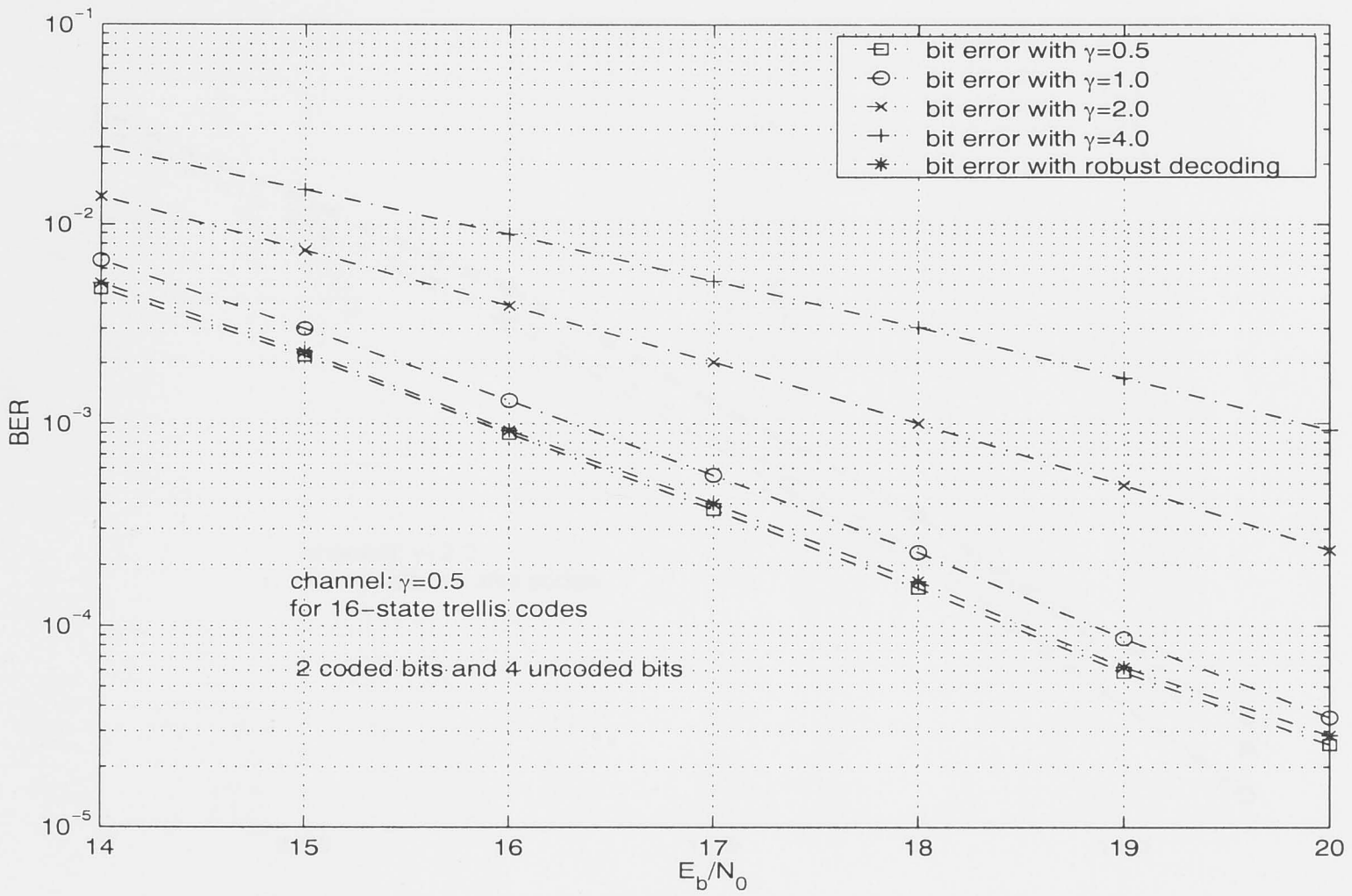
code, a double-parity-check structure with $m = 20$, $q = 20$ and $l = 50$ (i.e., 20,000 symbols per block) is employed. 16-state trellis shaping is also applied with the concatenated trellis codes. The peak number of iterations is set to 50, except the case of the $\gamma = 0.5$ channel, in which the peak number of iterations is 30.

First, we study the robust VA decoder under different types of noise channels. In Fig. 7-7 and Fig. 7-8, we present the cases where the channel noise is fixed for all symbol durations. In Fig. 7-9, we present a case where the mixed types of noise affect the decoder.

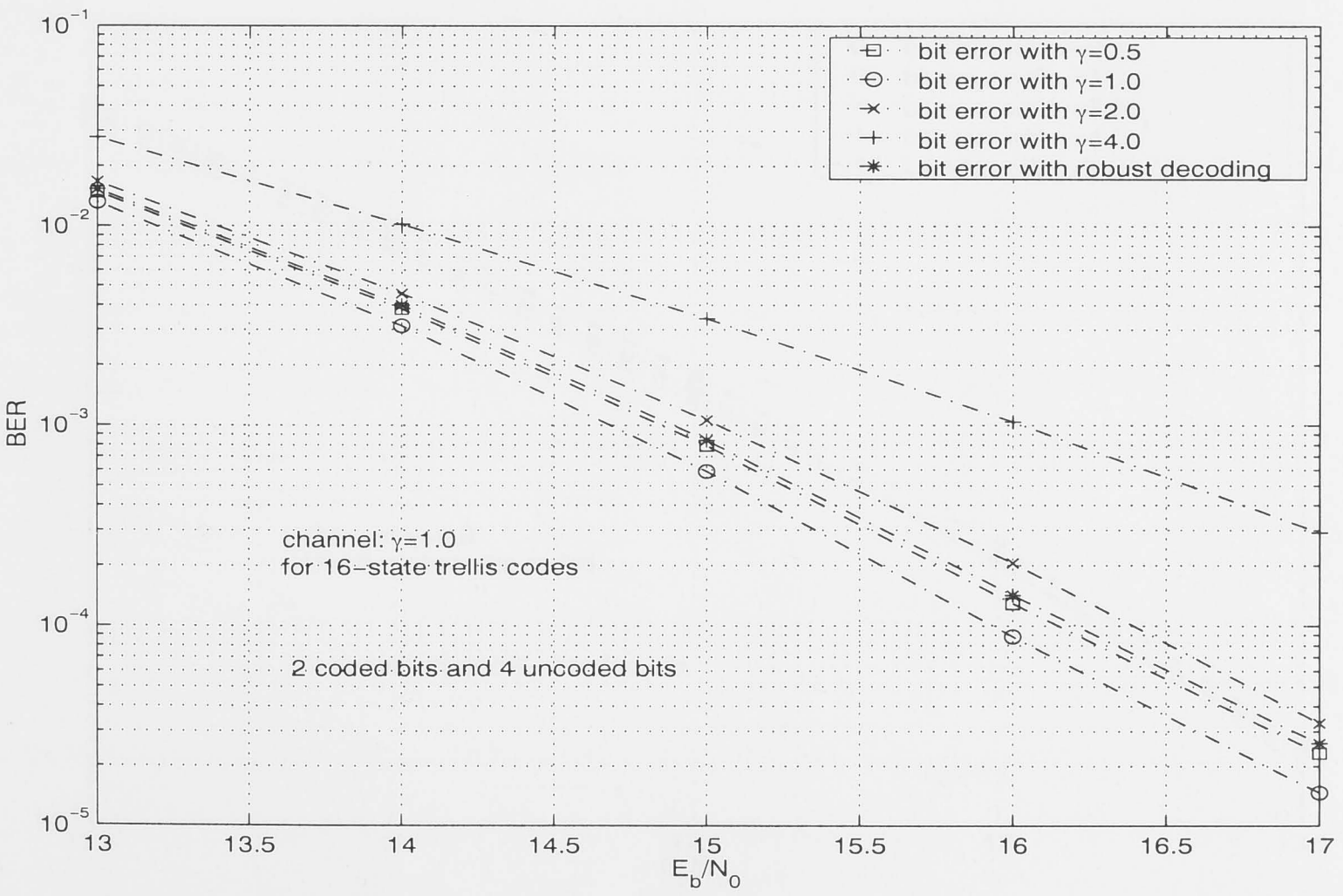
From Figures 7-7 and 7-8, we can see that the channel disturbed by the short-tail type of noise ($\gamma = 4.0$) achieves better performance than the channel disturbed by the long-tail type of noise ($\gamma = 0.5$). In addition, to the short-tail noise channel, the performance degradation is not significant when the mismatched decoder is employed.

We can also find in Figures 7-7, 7-8 and 7-9 that the robust decoder generally outperforms mismatched decoders under different channel noise conditions. In some cases, the mismatched decoder could perform better than the robust decoder. Remember that the robust decoder is designed to minimise the worst-case error performance. The above figures show that the robust decoder is close to the optimal matched decoder, which knows exactly the channel information.

Similar simulations are carried out for parity-concatenated trellis codes using the IVA under different types of noise channels. Comparing the error performance of the robust IVA with the robust VA, we can see an interesting phenomenon. In Fig. 7-7 (a), the difference of error performance between the decoders with $\gamma = 0.5$ and $\gamma = 2.0$ at a $BER = 1.0 \times 10^{-4}$ is over 2.0 dB. In Fig. 7-10 (a), however, the difference is negligible. Figure 7-10 (b) shows the average number of iterations for different types of decoder. It is clear that compared to the decoder with $\gamma = 0.5$, the decoder with $\gamma = 2.0$ takes many more iterations to converge, although it finally converges with almost same error performance as that of the decoder with $\gamma = 0.5$. In addition, for the channel with $\gamma = 0.5$, as the SNR

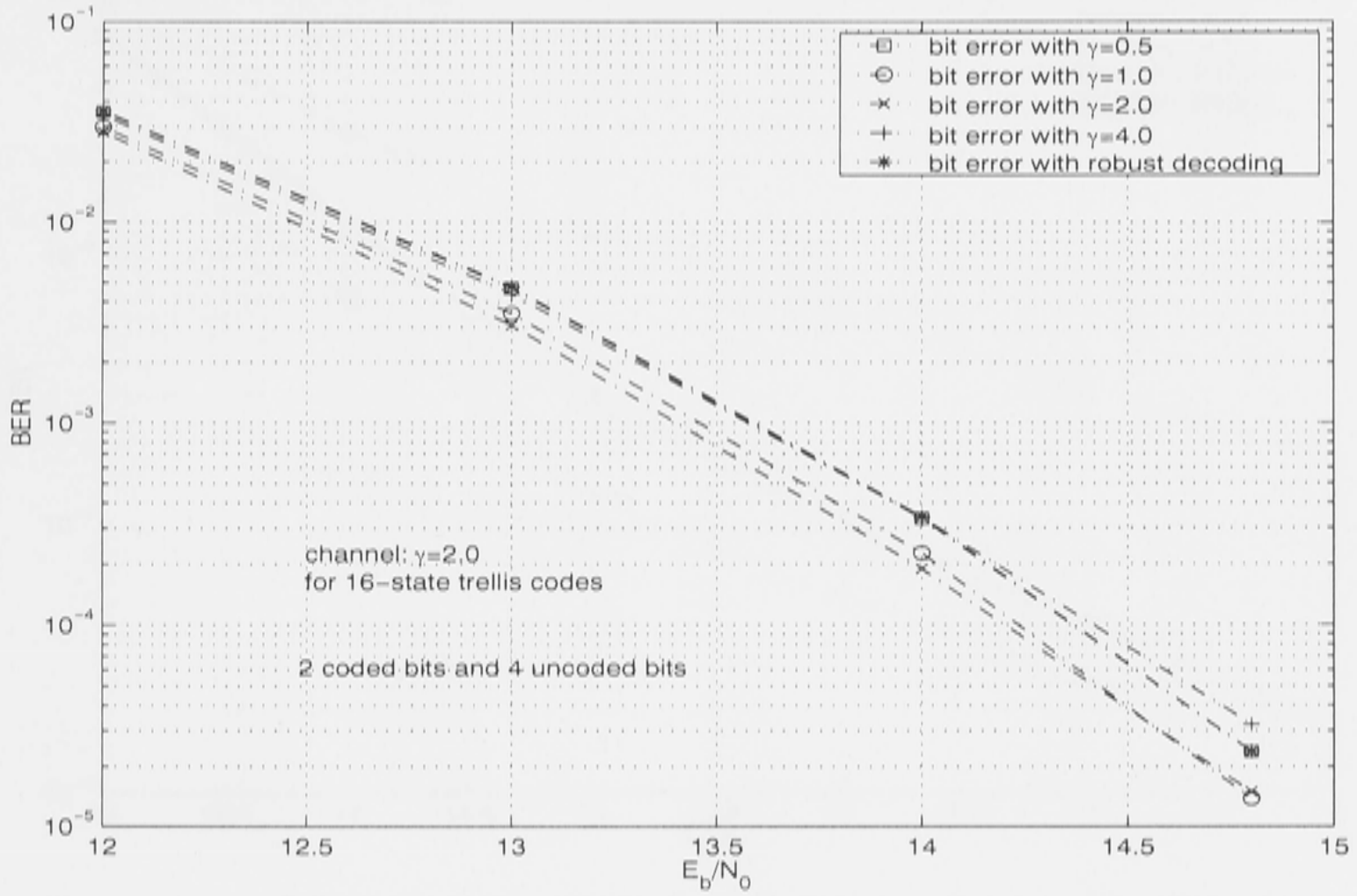


(a) Channel $\gamma = 0.5$

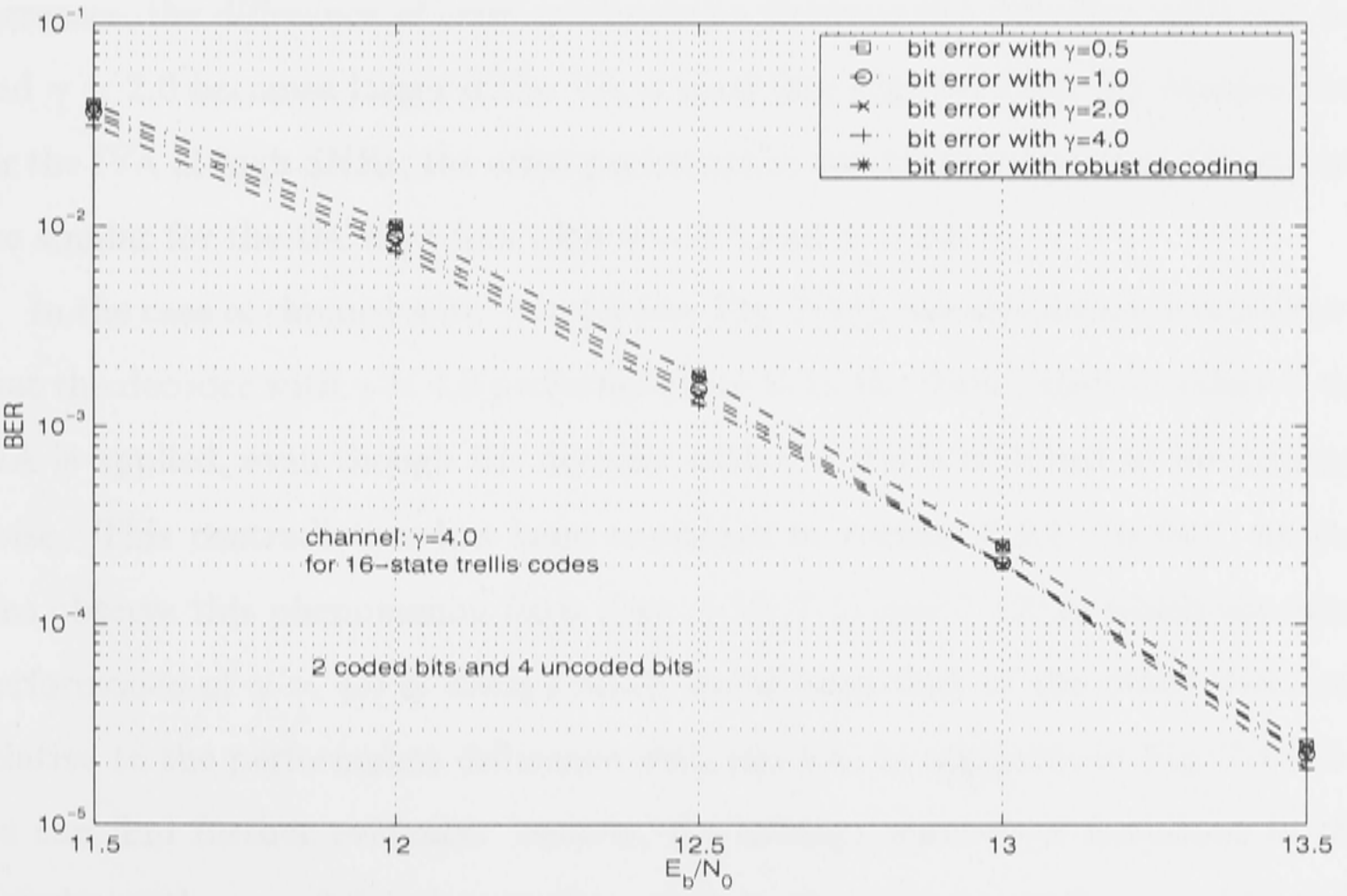


(b) Channel $\gamma = 1.0$

Figure 7-7: BER performance of the robust VA decoder for the $\gamma = 0.5$ and 1.0 channels



(a) Channel $\gamma = 2.0$



(b) Channel $\gamma = 4.0$

Figure 7-8: BER performance of the robust VA decoder for the $\gamma = 2.0$ and 4.0 channels

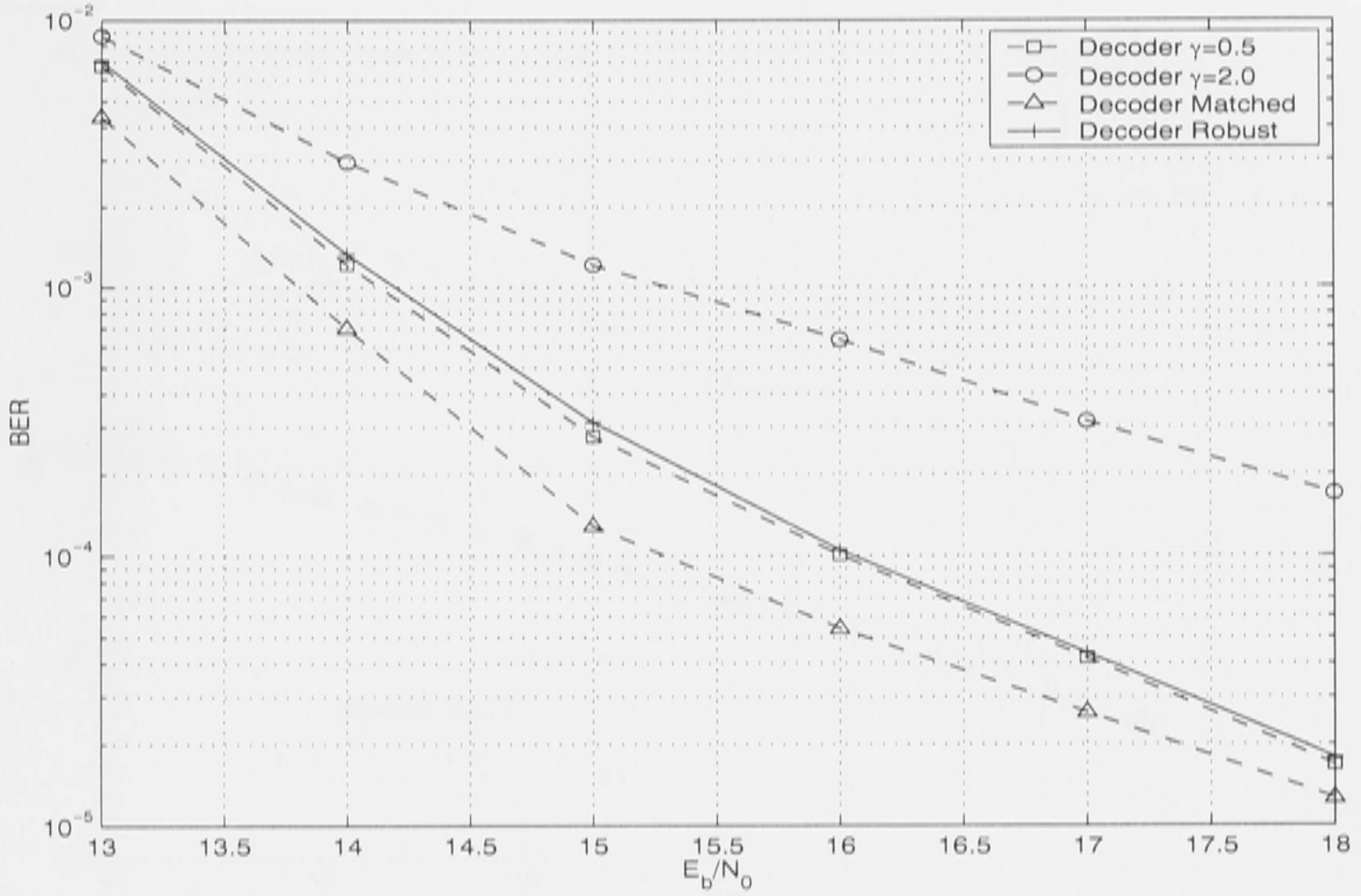
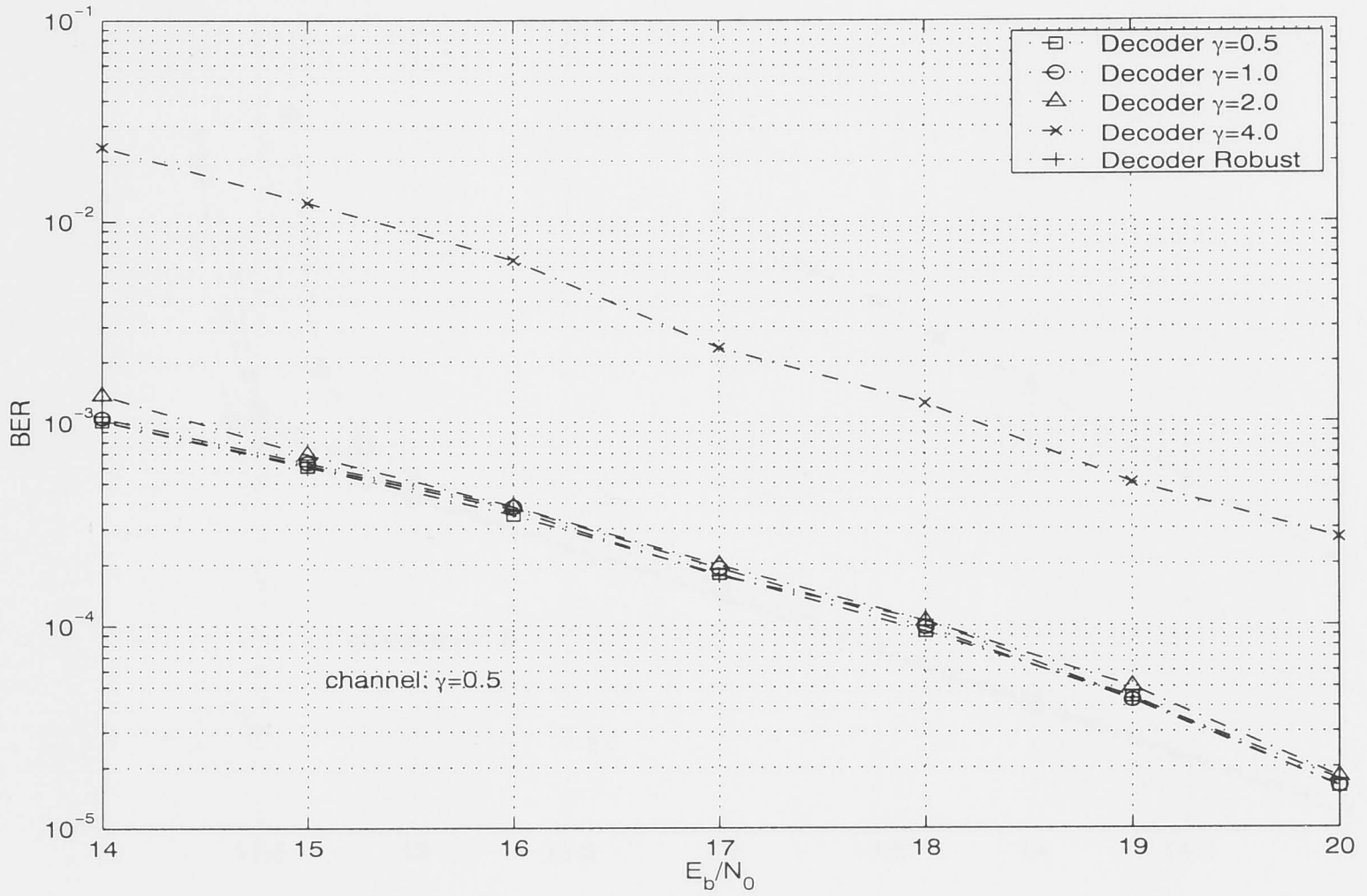


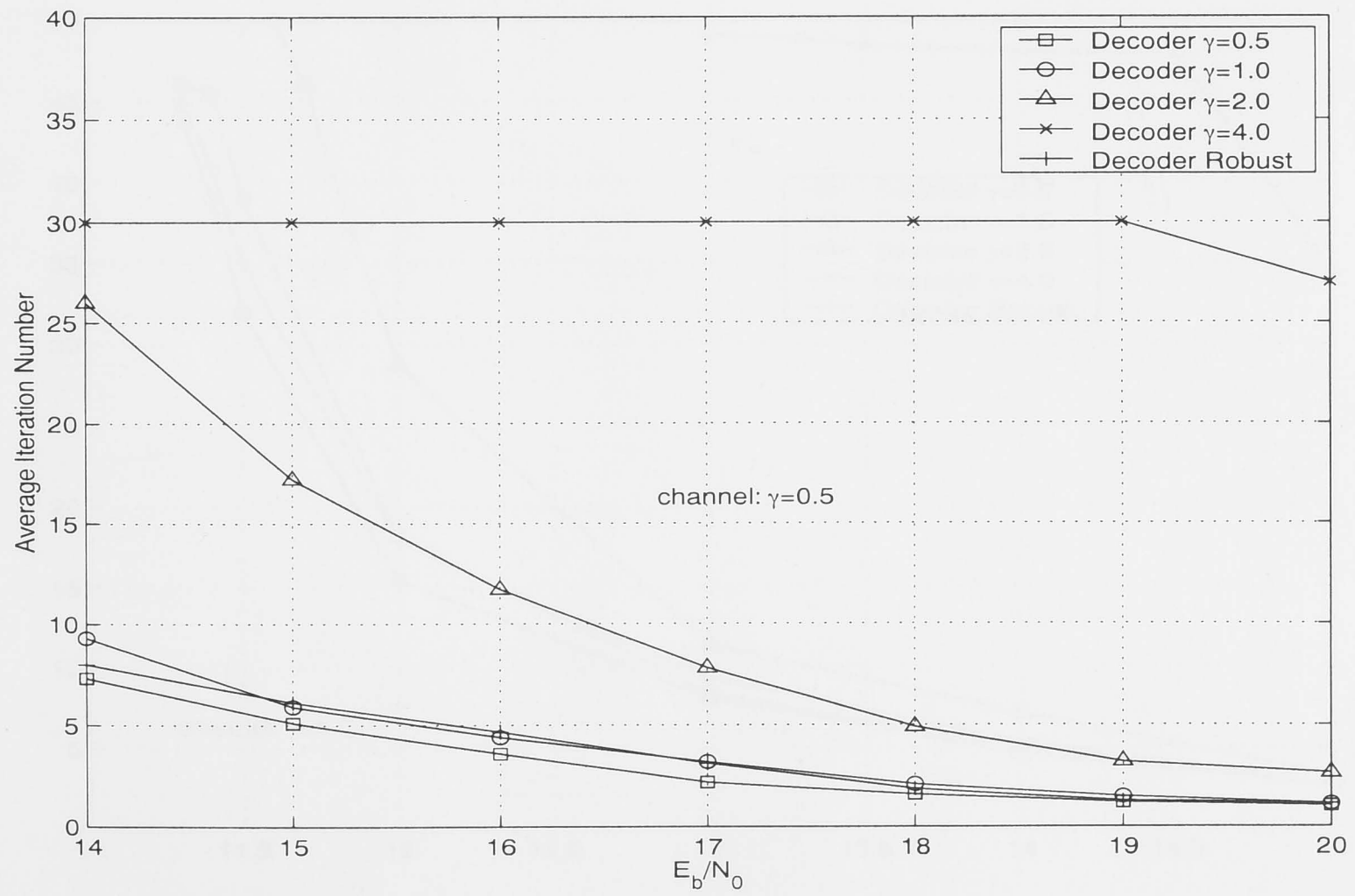
Figure 7-9: BER performance of the robust VA decoder in a channel with mixed type of noise, $P_r(\gamma = 2) = 80\%$, $P_r(\gamma = 0.5) = 20\%$

increases, the difference of error performance between the decoders with $\gamma = 0.5$ and $\gamma = 2.0$ becomes larger if the VA is used (see Fig. 7-7 (a)). By comparison, for the IVA at high SNRs, the error performance and average number of iterations are similar for the the decoders with $\gamma = 0.5$ and $\gamma = 2.0$.

In the case of channel with $\gamma = 4.0$ (see Fig. 7-13), the simulation results show that the decoder with $\gamma = 4.0$ performs worse than the three other decoders if the IVA is applied, even though the decoder with $\gamma = 4.0$ is matched to the channel noise. This contradiction has been explained in section 7.3.4. In fact, we can also observe this phenomenon from Figs. 7-10, 7-11 and 7-12, in which the error performance of $\gamma = 4.0$ is always much worse than that of the other decoders, relative to the performance difference with the VA. In addition, in Fig. 7-13 (b), we can find further evidence: namely, the average number of iterations in the decoder with $\gamma = 2.0$ is larger than that in the decoder with $\gamma = 1.0$, even though the $\gamma = 2.0$ decoder performs slightly better than the $\gamma = 1.0$ decoder in term of bit error performance. This result is caused jointly by mismatched

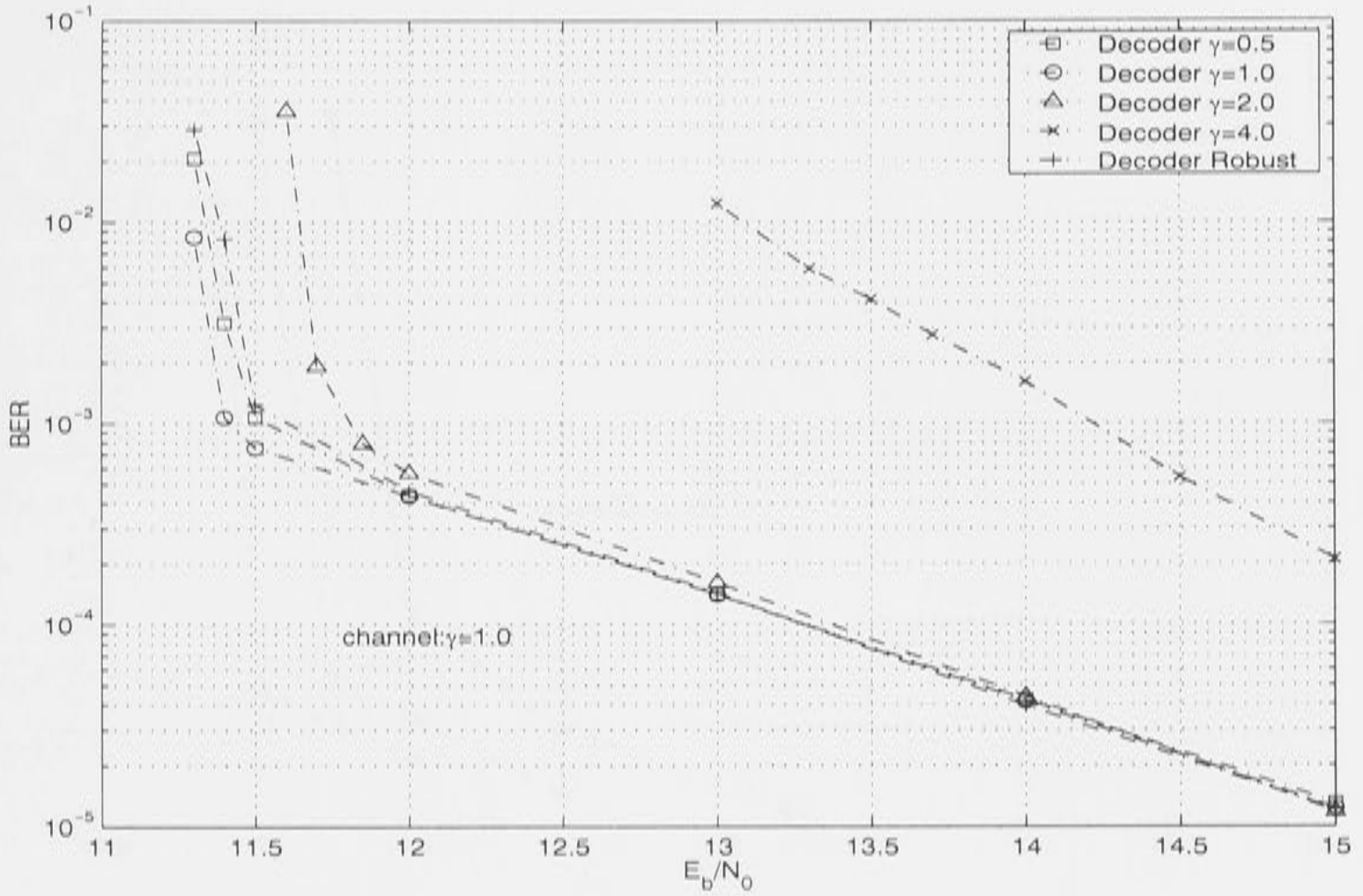


(a) BER performance

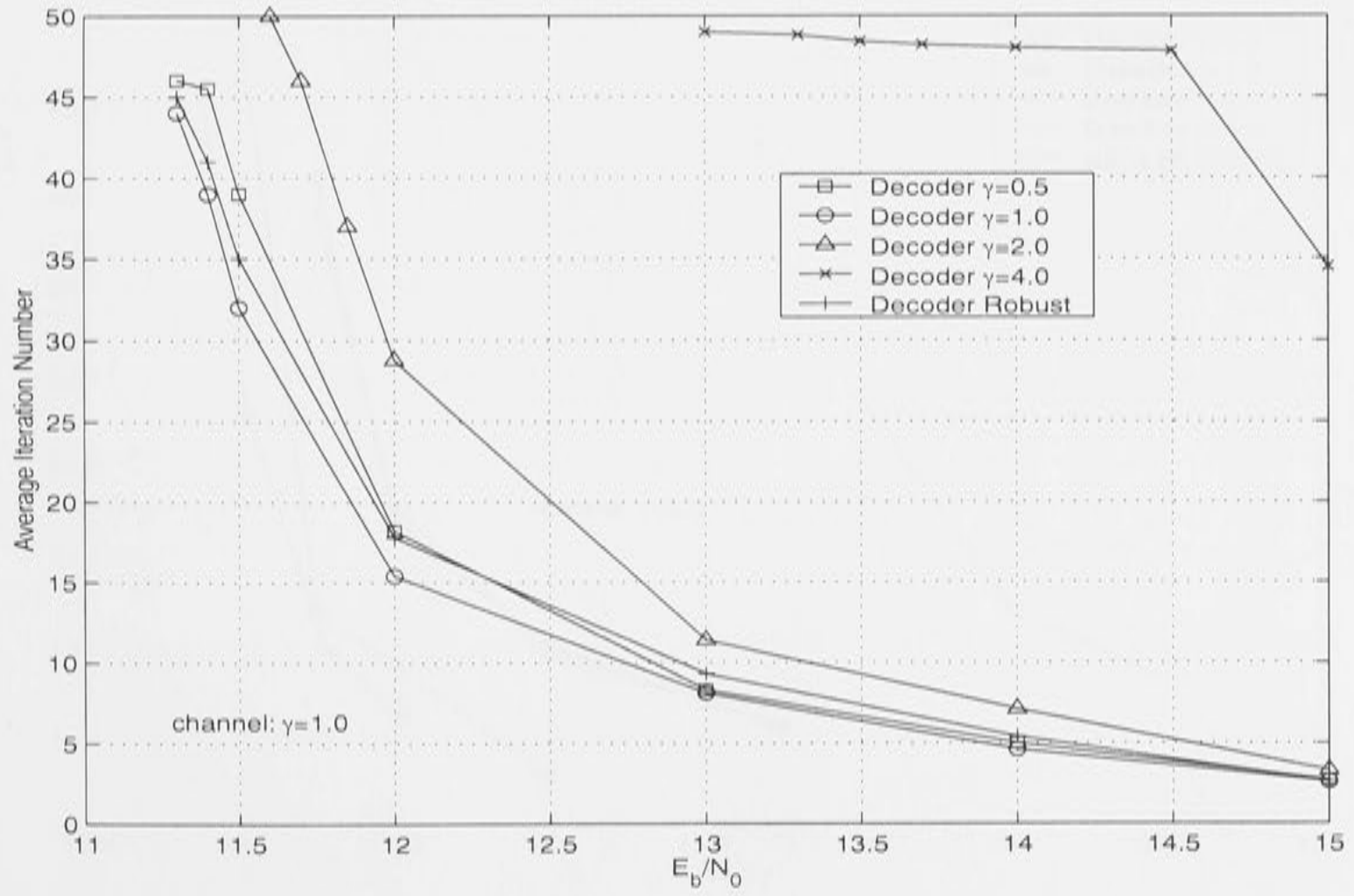


(b) Average number of iterations

Figure 7-10: Performance of the robust IVA decoder for the channel $\gamma = 0.5$

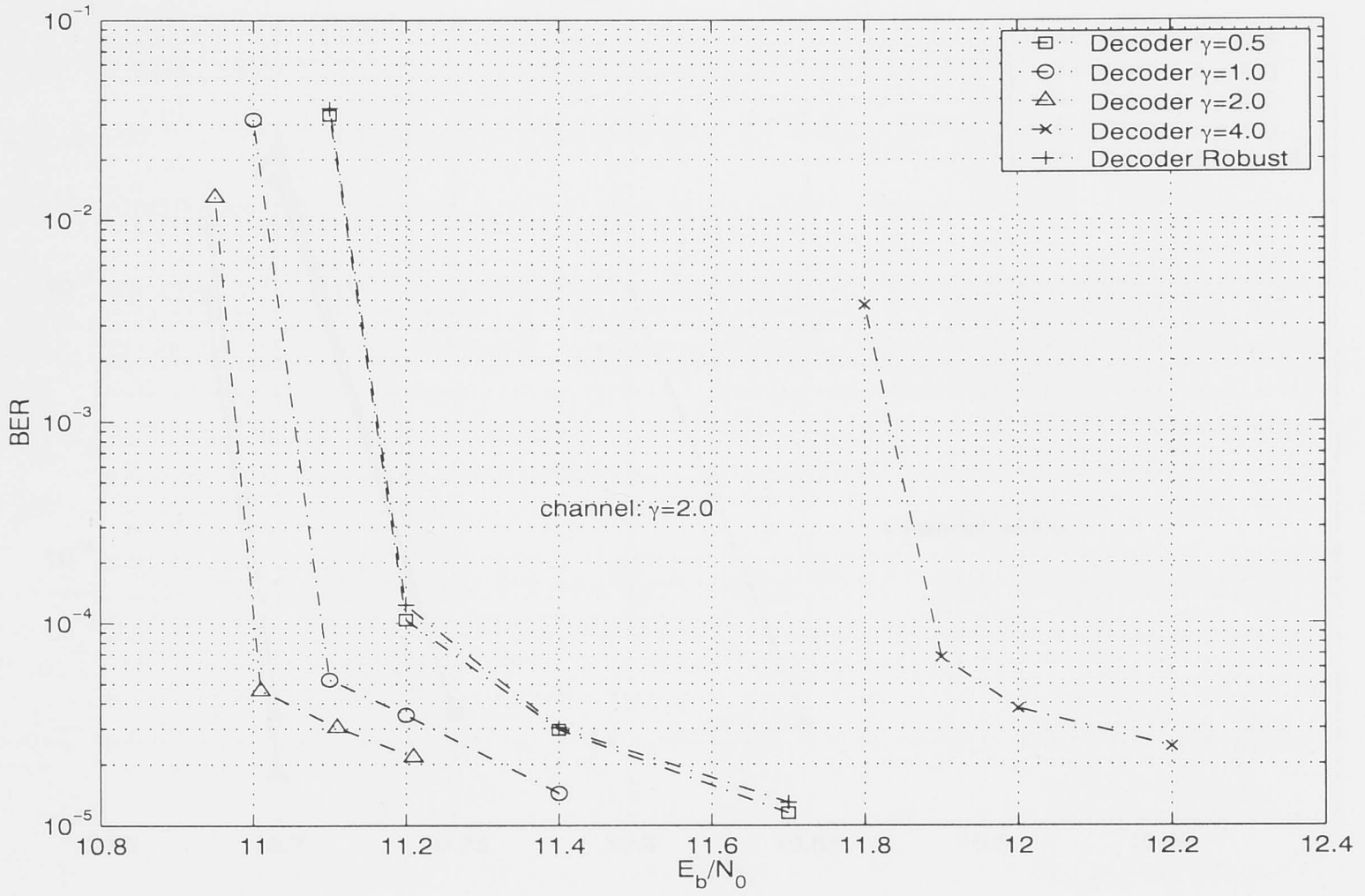


(a) BER performance

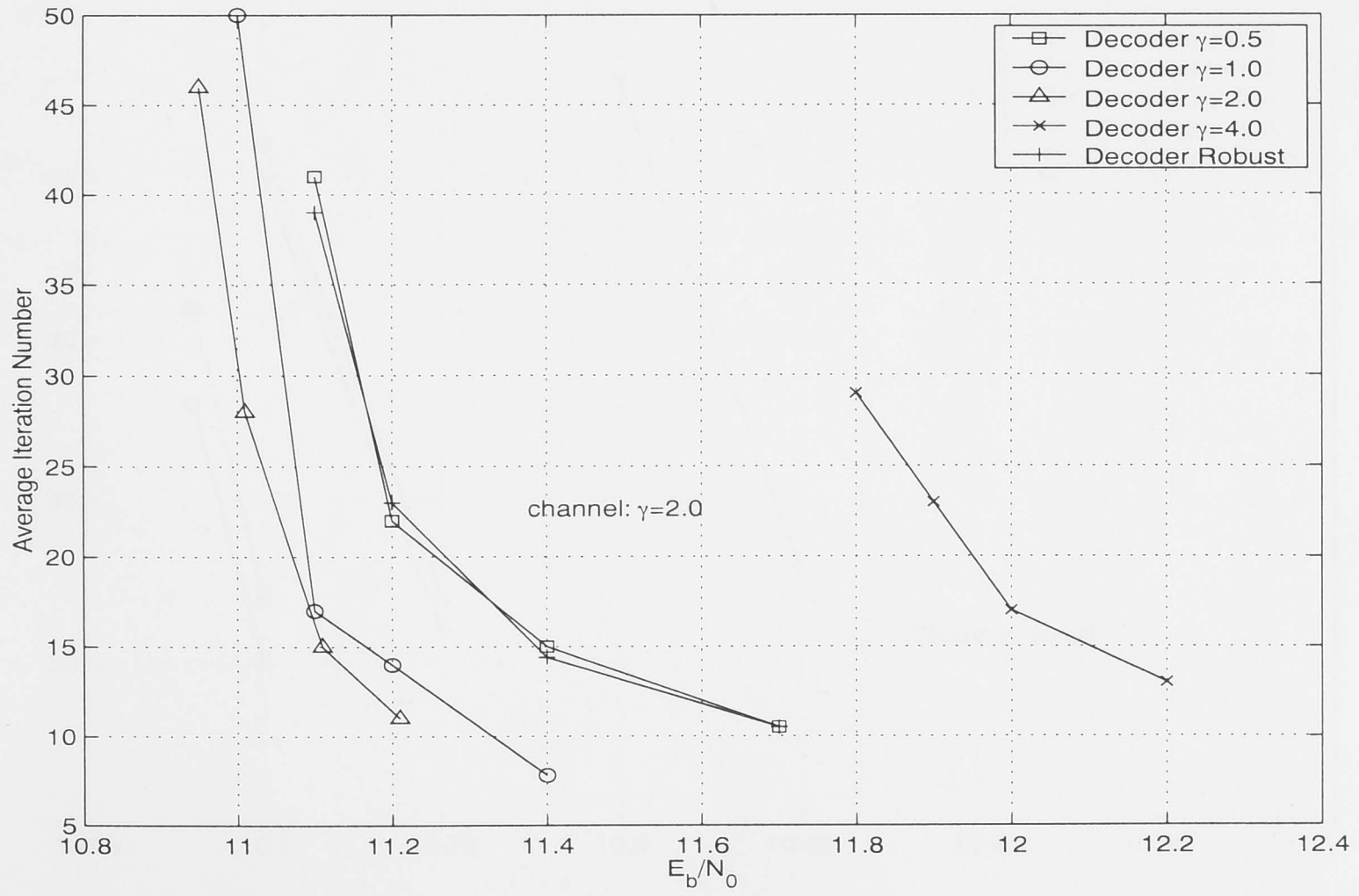


(b) Average number of iterations

Figure 7-11: Performance of the robust IVA decoder for the channel $\gamma = 1.0$

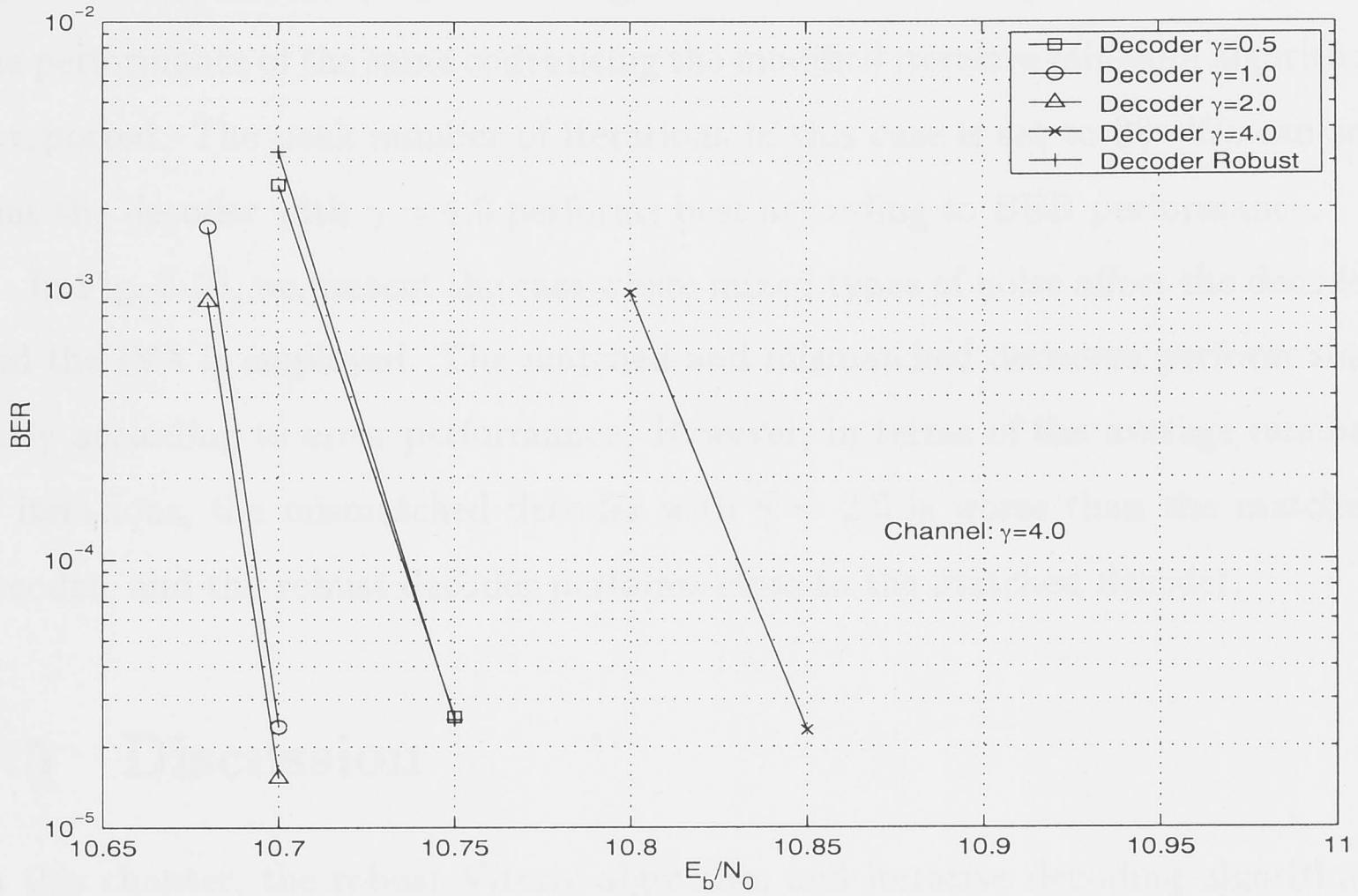


(a) BER performance

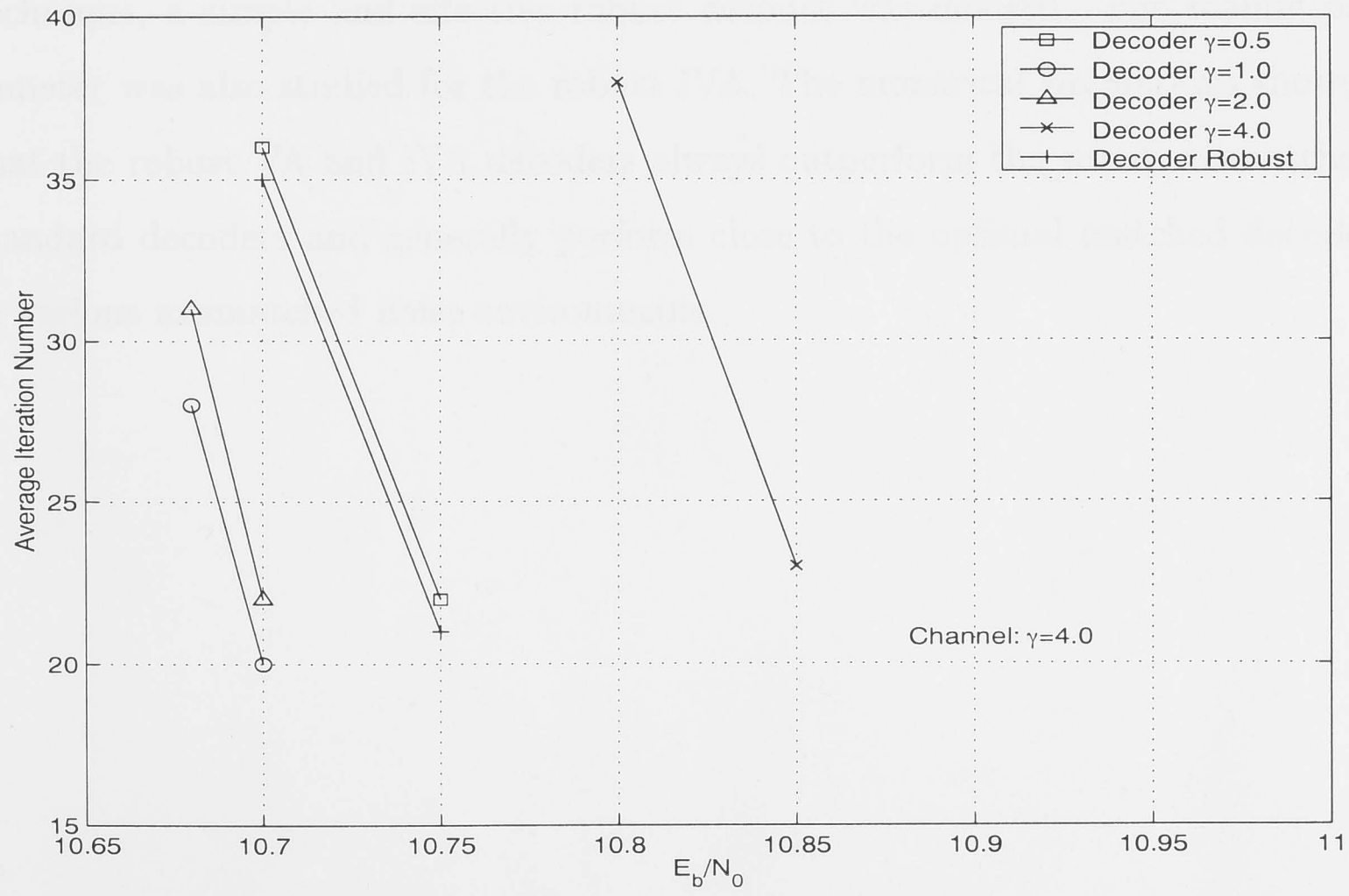


(b) Average number of iterations

Figure 7-12: Performance of the robust IVA decoder for the channel $\gamma = 2.0$



(a) BER performance



(b) Average number of iterations

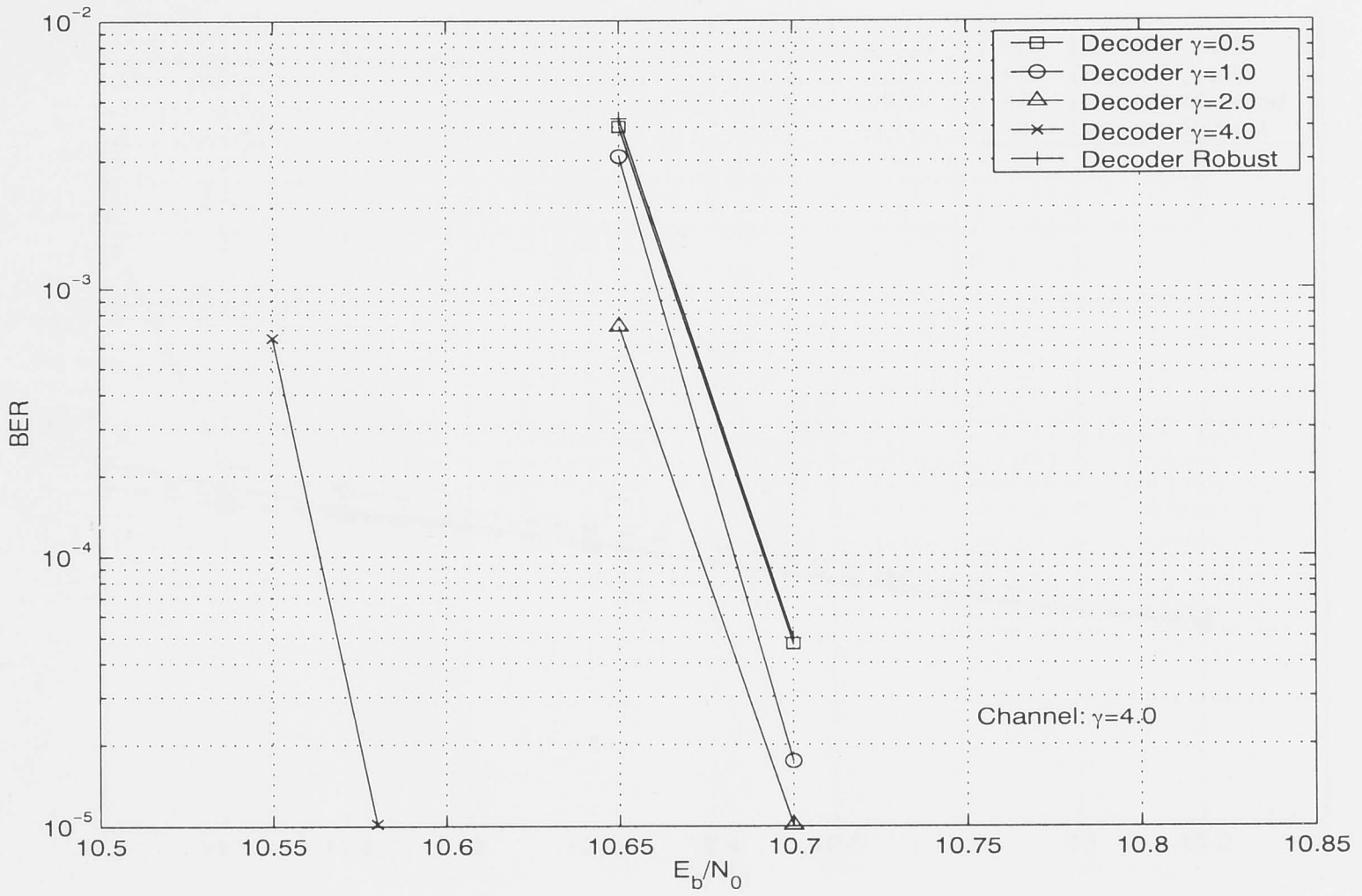
Figure 7-13: performance of the robust IVA decoder for the channel $\gamma = 4.0$

decoders and amplification of the negative impact. For comparison, in Fig. 7-14 the performance of the same codes using the modified iterative min-sum algorithm is reported. The peak number of iterations in this case is set to 30. We can see that the decoder with $\gamma = 4.0$ performs best according to BER performance.

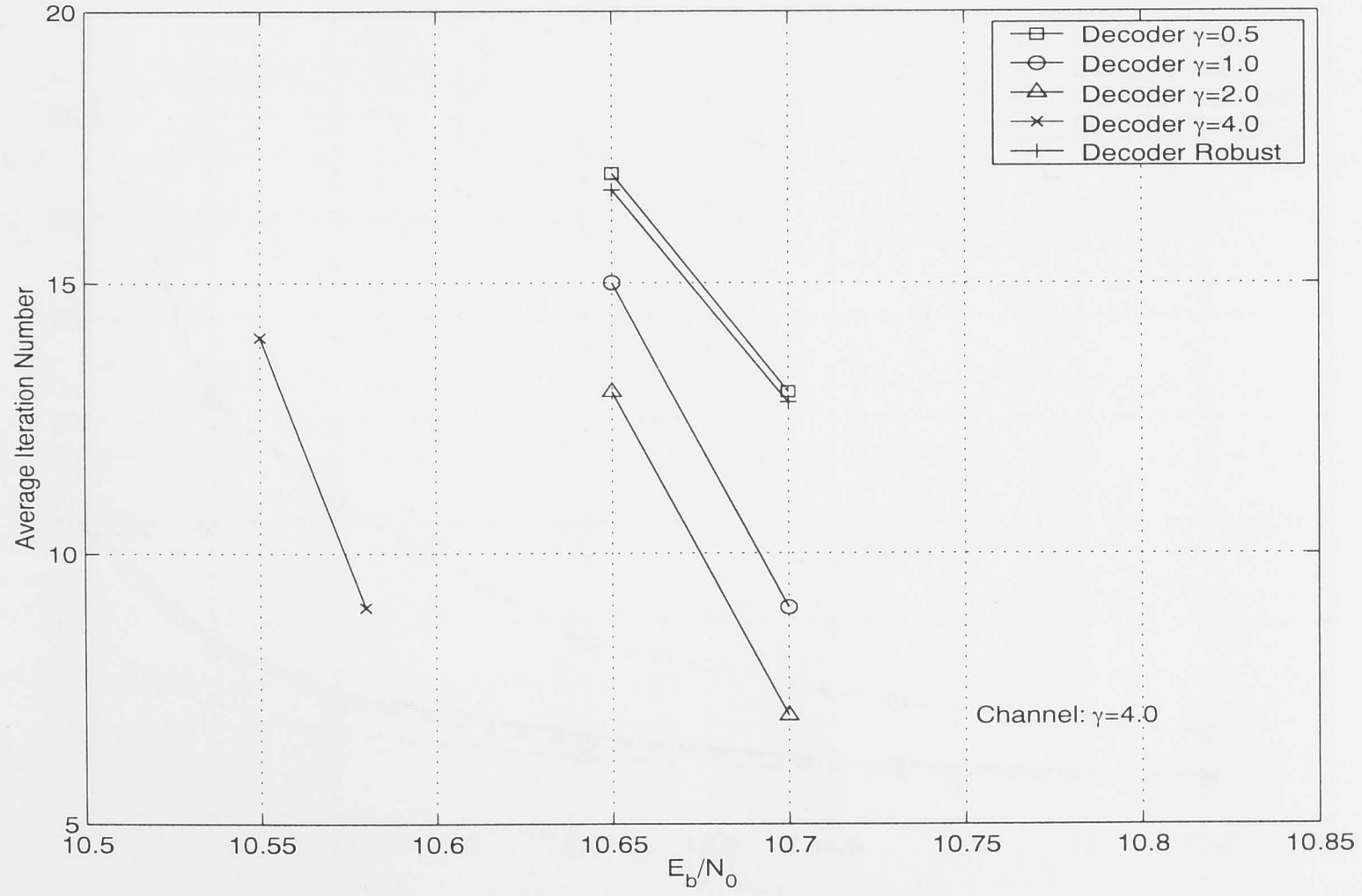
In Fig. 7-15, we present the case where mixed types of noise affect the decoder and the IVA is employed. The matched and mismatched decoders perform similarly according to error performance. However, in terms of the average number of iterations, the mismatched decoder with $\gamma = 2.0$ is worse than the matched decoder, and the robust decoder performs close to the matched decoder.

7.5 Discussion

In this chapter, the robust Viterbi algorithm and iterative decoding algorithms emphasising the IVA were studied for conventional trellis codes and parity-concatenated trellis codes under an uncertain noise environment. Using the *minimax* technique, a simple and effective robust decoder was devised. The scaling parameter was also studied for the robust IVA. The numerical simulations showed that the robust VA and IVA decoders always outperform the worst mismatched standard decoders and generally perform close to the optimal matched decoder in various mismatched noise environments.

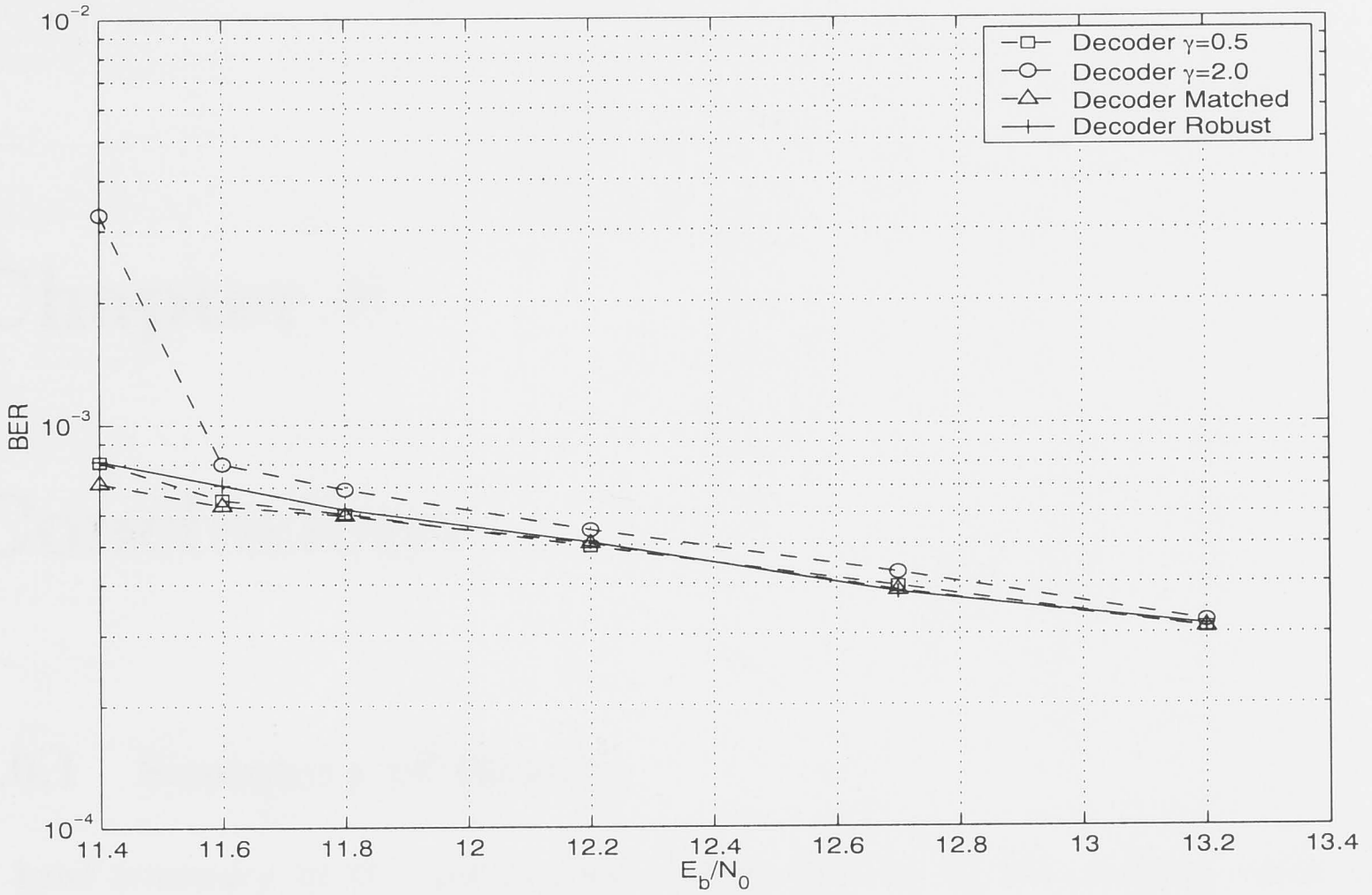


(a) BER performance

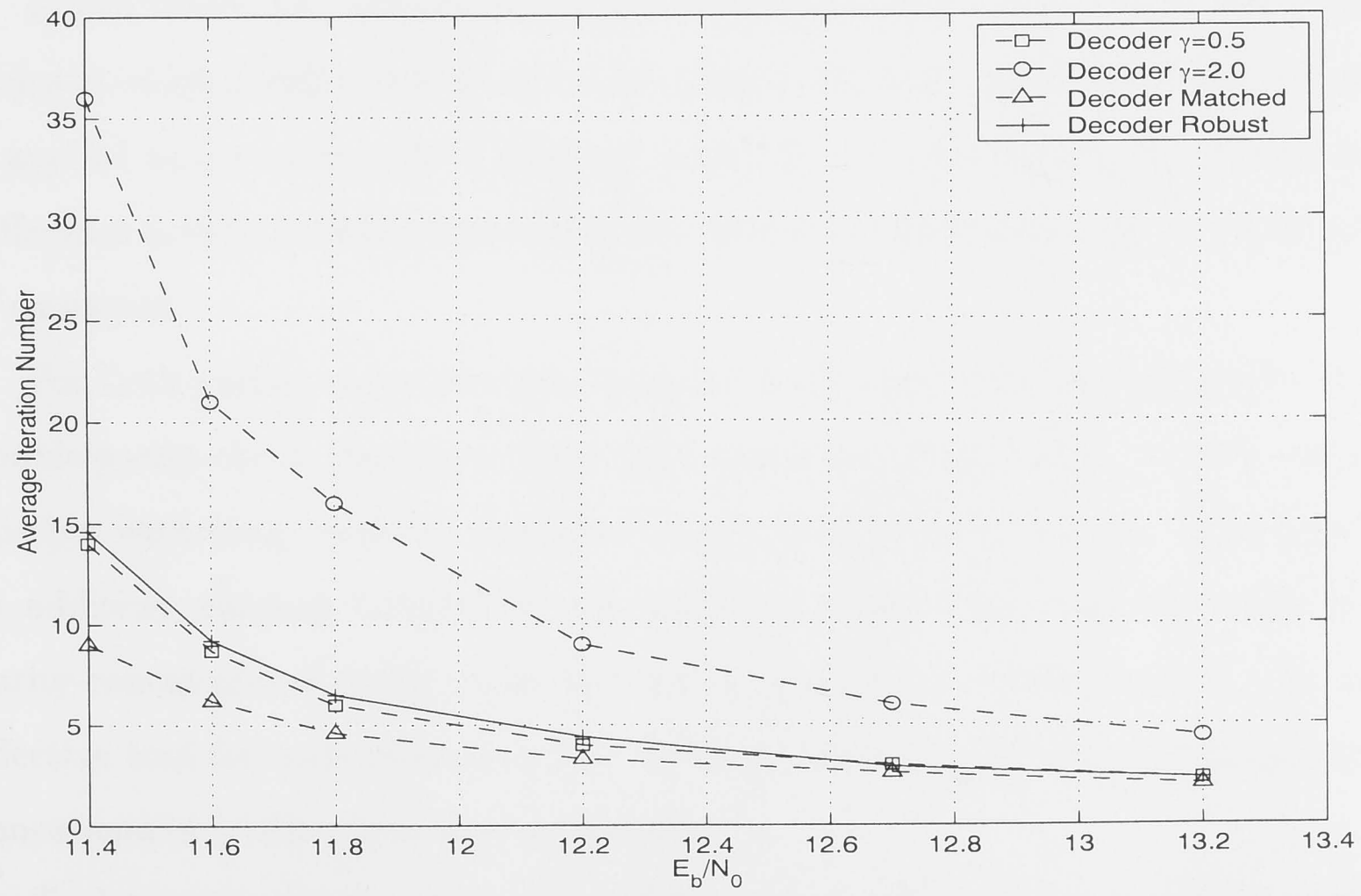


(b) Average number of iterations

Figure 7-14: Performance of the robust iterative min-sum decoder for the channel $\gamma = 4.0$



(a) BER performance



(b) Average number of iterations

Figure 7-15: Performance of the robust IVA decoder in a mixed type of noise channel, $P_r(\gamma = 2) = 80\%$, $P_r(\gamma = 0.5) = 20\%$

Chapter 8

Conclusions

8.0.1 Summary of Results

A brief summary of the accomplished work is given in this chapter, with an emphasis on the contributions to the subject of iterative decoding algorithms for parity-concatenated trellis codes.

In this thesis, we constructed the parity-concatenated 2-D and M-D Wei trellis codes in which a trellis code is used as the inner code and a simple even parity code is applied as the outer code. Compared with turbo TCM schemes, one significant difference is that a conventional block interleaver rather than pseudo-random one is employed.

For both packet and continuous transmissions, the corresponding single- and double-parity-check structures have been designed, respectively. Packet transmission with long, medium and short block lengths have also also considered. In addition, shaping techniques and multilevel codes were combined with the parity-concatenated trellis codes for further performance improvement. As an effective tool for code interpretation, TWL graph representations of the parity-concatenated trellis codes were presented.

To deal with serious error propagation caused by small cycles in TWL graphs, several iterative decoding algorithms have been proposed. One of them is the iterative Viterbi algorithm (IVA). The only difference between the IVA and the

standard VA is the calculation of branch metrics. In the IVA the branch metrics of symbols are randomly selected to be fed back as *extrinsic* information, which effectively avoids the negative impact from other symbols. Both parity-concatenated 2-D trellis codes and M-D Wei trellis codes can be decoded by the IVA.

The other way to deal with small cycles in TWL graphs is the modified iterative two-way algorithms, which were developed from conventional ITWAs. A normalisation function determined by the length of error event of trellis codes is employed in the modified ITWAs, and good results have been achieved. From the perspective of graph interpretation, the IVA can be viewed as a simplified case of the iterative min-sum algorithm in which the local weights (branch metrics) of one symbol node rather than sum of upstream and downstream weights are applied as *extrinsic* information.

For the parity-concatenated trellis codes at a high spectral efficiency, error floors often appear at the region close to the Shannon capacity limit due to the parallel transition errors in trellis codes. The upper bound on the error floor has been analytically determined, and an appropriate multilevel code is selected to bring down the error floor to a very low level.

Based on the graph representations, five iterative decoding algorithms, including the IVA and modified iterative min-sum/sum-product algorithms, have been developed, and their computational complexity and performance were compared. It can be seen that the iterative sum-product algorithm has the highest computational complexity, and the IVA has the lowest complexity. Compared with the standard VA, these IDAs can achieve significant gains for both parity-concatenated 2-D trellis codes and M-D trellis codes.

In simulations, with trellis shaping, the performance of 1.25 dB away from the Shannon capacity limit at a $BER = 3.0 \times 10^{-5}$ can be achieved by the IVA for parity-concatenated Ungerboeck 256-state trellis codes with $20,000$-symbol block length, but an error floor occurs. Further, using a simple binary BCH code, the error floor can be reduced to 10^{-9} with very little additional cost. For the same

trellis codes with 2,000- and 200-symbol block length, the performances about 1.7 dB and 2.1 dB away from the Shannon limit have been achieved by the IVA, respectively. For the popular 4-D 16-state Wei trellis codes, the numerical results showed that about 2.2 dB net gain can be obtained using the IVA. All these designs have been achieved with low complexity and computation.

Through simulations, we find that for continuous transmission or packet transmission with long block length, generally the modified iterative min-sum/sum-product algorithms can get slightly better performance than the IVA. However, for packet transmission with short block length, the IVA outperforms the modified ITWAs. In addition, for both packet and continuous transmission, the complexity of the ITWAs is much higher than that of the IVA, expecially for the parity-concatenated M-D trellis codes. Therefore, according to the tradeoff between computation complexity and performance, the IVA is preferable among five IDAs. Due to its low complexity, the IVA can be applied to many current standard systems, such as in high-rate voice band modems or ADSL modems, without or with very little modification.

Several other important advantages of the IVA include: (a) the decoder knows the erroneous packets with high accuracy; (b) it does not require any noise variance estimation; (c) it can recover partial block data. The evident disadvantages of the IVA include: (a) high peak complexity value for some cases and (b) no soft output.

Finally, a robust Viterbi algorithm and robust iterative decoding algorithms emphasising the IVA have been studied for conventional trellis codes and parity-concatenated trellis codes under an uncertain noisy environment. Using the *minimax* technique, a simple and effective robust decoder was devised. The numerical simulations showed that these robust VA and IVA decoders always outperform the worst mismatched standard decoders and generally perform close to the optimal matched decoder in various mismatched noise environments.

8.0.2 Areas for Further Research

In this thesis, we have investigated a class of iterative decoding algorithms, named the iterative Viterbi algorithm (IVA). We showed that it can achieve near Shannon limit performance with very low complexity. It can be potentially applied into many standards such as the ADSL modem, cable modem, etc. In the future, we will investigate how to apply it in these real systems. One of the key problems is to study how the convolutional interleaver affects the algorithm. The other one is to study how to implement the IVA with lowest hardware complexity.

Regarding our work in the area of iterative decoding based on graphs, we will further investigate whether the scaling process is necessary for other types of codes such as short LDPC and short turbo codes.

Bibliography

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

[2] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes", *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260-264, Feb. 1998.

[3] A. M. Aji, G. B. Horn, and R. J. McEliece, "Iterative Decoding on Graphs with a Single Cycle", *Proceeding of IEEE ISIT'98*, pp. 276, MIT, Boston, USA, Aug. 16-21, 1998.

[4] A. K. Calderbank and L. H. Ozarow, "Nonequiprobable signalling on the Gaussian channel", *IEEE Trans. on Inform. Theory*, Vol. 36, pp. 726-740, July, 1990.

[5] A. K. Khandani and P. Kabal, "Shaping multidimensional signal spaces—Part I: Optimum shaping, shell mapping and Part II: Shell-addressed constellations", *IEEE Trans. on Inform. Theory*, Vol. 39, pp. 1799-1819, Nov. 1993.

[6] A. S. Barbulescu and S. S. Pietrobon, "Interleaver design for three dimensional turbo-codes", *Proc. 1995 IEEE International Symposium on Information Theory*, Whistler, British Columbia, Canada, p. 37, Sept. 1995.

[7] B. Vucetic, *Designs of Turbo codes and Turbo Trellis Coded Modulation*, to be published.

[8] B. Vucetic, "Iterative decoding algorithm", in *PIMRC'97*, Helsinki, Finland, pp. 99-120, Sept. 1997.

[9] B. J. Frey, *Graphical Models for Machine Learning and Digital Communication*, Cambridge, MA: MIT Press, 1998.

[10] B. J. Frey and D. J. C. MacKay, "Trellis-Constrained Codes", in *Proc. of the 35th Allerton Conference on Commun., Control and Computing 1997*, Chanpaign-Urbana, Illinois.

[11] B. J. Frey and D. J. C. MacKay, "A revolution: belief propagation in graphs with cycles", in *Advances in Neural Information Processing Systems 10*, MIT Press: Cambridge, MA, 1998.

[12] B. J. Frey and F. R. Kschischang, "Probability propagation and iterative decoding", *Proceedings of the 34th Allerton Conference on Communication, Control and Computing 1996*, Champaign-Urbana, Illinois, 1996.

[13] B. J. Frey, R. Koetter, and A. Vardy, "Skewness and pseudocodewords in iterative decoding", pp. 148, *ISIT'98*, Cambridge, MA, USA, 1998.

[14] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes", *IEEE Trans. on Commun.*, vol. 44, no. 10, pp. 1261-1271, OCT. 1996.

[15] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-correcting Coding and Decoding: Turbo codes", *Proc. IEEE Int. Conf. Commun. (ICC)*, Geneva, Switzerland, 1993, pp. 1064-1070.

[16] D. J. C. MacKay, "Good codes based on very sparse matrices", *IEEE Trans. Inform. Theory*, vol. 45, pp. 399-431, March 1999.

[17] D. J. C. MacKay and C. P. Hesketh, "Performance of low density parity check codes as function of actual and assumed noise levels", *IEEE Trans. Commun.*, to be published.

[18] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes", *Electronics Letters*, vol. 33, no. 6, pp. 457-458, March 1997.

[19] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of constructions of irregular Gallager codes", *IEEE Trans. on Commun.*, vol. 47, no. 10, pp. 1449-1454, Oct. 1999.

[20] D. Divsalar and R. J. McEliece, "Effective free distance of turbo-codes", *Electronics Letters*, vol. 32, no. 5, pp. 445-446, Feb. 1996.

[21] E. Biglieri, D. Divsalar, P. J. McLane, and M. K. Simon, *Introduction to Trellis-coded Modulation with Applications*, Macmillan Publishing Company, New York, 1991.

[22] F. Jelinek, "Bootstrap trellis decoding", *IEEE Trans. on Inform. Theory*, Vol. 21, pp. 318-325, May 1975.

[23] F. Jelinek and J. Cocke, "Bootstrap hybrid decoding for symmetrical binary input channels", *Information and Control*, Vol. 18, pp. 261-298, April 1971.

[24] F. R. Kschischang and Brendan J. Frey, "Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models", *IEEE Journal on Selected Areas in Commun.*, Vol. 16, No. 2, pp. 219-230, Feb. 1998.

[25] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm", Submitted to *IEEE Trans. on Inform. Theory.*

[26] F. Q. Wang and D. J. Costello, Jr., "Sequential decoding of trellis codes at high spectral efficiencies", *IEEE Trans. on Inform. Theory*, vol. 43, no. 6, pp. 2013-2019, Nov. 1997.

[27] F. Q. Wang and D. J. Costello, Jr., "New rotationally invariant four dimensional trellis codes", *IEEE Trans. on Inform. Theory*, vol. 42, pp. 291-300, Jan. 1996.

[28] F. Q. Wang and D. J. Costello, Jr., "Probabilistic Construction of Large Constraint Length Trellis Codes for Sequential Decoding", *IEEE Trans. on Comm.*, Vol. 43, No. 9, pp. 2439-2447, Sept. 1995.

[29] F. Q. Wang and D. J. Costello, Jr., "Sequential decoding with trellis shaping", *IEEE Trans. on Inform. Theory*, vol. 41, no. 6, pp. 2037-2040, Nov. 1995.

[30] F. Q. Wang and D. J. Costello, Jr., "A hybrid M-algorithm/sequential decoder for convolutional and trellis codes", *1990 Int. Symposium on Inform. Theory and Its Applications (ISITA'90)*, pp. 67-69, Hawaii, U.S.A., Nov. 27-30, 1990.

[31] G. Ungerboeck, "Channel coding with multilevel/phase signals", *IEEE Trans. on Inform. Theory*, Vol. IT-28, pp. 55-67, Jan. 1982.

[32] G. Ungerboeck, "Trellis coded modulation with redundant signal sets, Part I: Introduction and Part II: State of the art", *IEEE Commun. Mag.*, Vol. 25, pp. 5-22, Feb. 1987.

[33] G. D. Forney, Jr., "Lower bounds on error probability in the presence of large intersymbol interference", *IEEE Trans. Commun. Technol.*, Vol. 20, pp. 76-77, Feb. 1972.

[34] G. D. Forney, Jr., "The Viterbi algorithm", *Proc. IEEE*, pp. 268-278, Mar. 1973.

[35] G. D. Forney, Jr., *Concatenated codes*, MIT Press, Cambridge, Mass, 1963

[36] G. D. Forney, Jr., "Trellis shaping", *IEEE Trans. on Inform. Theory*, Vol. 38, pp. 281-300, Mar. 1992.

[37] G. D. Forney, Jr., "Final report on a coding system design for advance solar mission", *Rep. NAS2-3637, Contract NASA CR73167*, NASA Ames Res. Ctr., Moffett Field, CA, 1967.

[38] G. D. Forney, Jr., "On iterative decoding and the two-way algorithm", *Proc. Intl. Symp. Turbo codes and related topics*, Brest, France, Sept. 1997.

[39] G. D. Forney, Jr., "Approaching AWGN channel capacity with coset codes and multilevel coset codes", to appear in *IEEE Trans. on Inform. Theory*, manuscript was received in Feb, 1999.

[40] G. D. Forney, Jr. and G. Ungerboeck, "Modulation and coding for linear Gaussian channels", *IEEE Trans. on Inform. Theory*, Vol. IT-44, No. 6, pp. 2389-2415, Oct., 1998.

[41] G. D. Forney, Jr., F. R. Kschischang, and B. Marcus, "Iterative decoding of tail-biting trellises", *Proceedings of Information Theory Workshop*, San Diego, CA, USA, Feb. 1998.

[42] G. D. Forney, Jr. and L. F. Wei, "Multidimensional constellations – Part I: Introduction, figures of merit, and generalized cross constellations", *IEEE J. on Select. Areas in Commun.*, vol. 7, pp. 877-892, Aug. 1989.

[43] G. J. Pottie and D. P. Taylor, "Multilevel codes based on partitioning", *IEEE Trans. on Inform. Theory*, Vol. IT-35, No. 1, pp. 87-98, Jan., 1989.

[44] G. Solomon, H. C. A. van Tilborg: "A connection between block and convolutional coded", *SIMA J. of Appl. Math*, pp. 358-369, Vol. 37, No. 2, Oct. 1979.

[45] H. A. Cabral, D. J. Costello, Jr., and P.R. Chevillat, "Bootstrap hybrid decoding using the multiple stack algorithm", *Proceeding of IEEE ISIT'97*, pp. 494, Ulm, Germany, June 29 - July 4.

[46] H. Nickl, J. Hagenauer, and F. Burkert, "Approaching Shannon's capacity limit by 0.27 dB using simple Hamming codes", *IEEE Commun. Lett.*, vol. 1, pp. 130-132, Sept. 1997.

[47] H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes", *IEEE Trans. on Inform. Theory*, Vol. IT-23, No. 3, pp. 371-377, May 1977.

[48] J. E. Porath, "Algorithm for converting convolutional codes from feedback to feedforward form and vice versa", *Electronics Letters*, Vol. 25, No. 15, pp. 1008-1010, 20th July 1989.

[49] J. D. Andersen, "Turbo codes extended with outer BCH code", *Electronics Letters*, Vol. 32, No. 22, pp. 2059-2060, October 1996.

[50] J. G. Proakis, *Digital Communications*, the third edition, McGraw-Hill, New York, 1995.

[51] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, CA: Morgan Kaufmann, 1988.

[52] J. Yuan, B. Vucetic, and W. Feng, "Combined turbo codes and interleaver design", *IEEE Trans. on Commun.*, vol. 47, no. 4, pp. 484-487, Apr. 1999.

[53] J. H. Ma and J. K. Wolf, "On Tail-biting Convolutional codes", *IEEE Trans. on Comm.*, Vol. COM-34, pp. 104-111, 1986.

[54] J. B. Anderson and K. E. Tepe, "Properties and Error Performance of the Tail-biting BCJR Decoder", submitted to *IT transaction*, manuscript received in Dec. 1998.

[55] J. Y. Wang and M. C. Lin, "On constructing trellis codes with large free distances and low decoding complexities", *IEEE Trans. on Commun.*, vol. 45, no. 9, pp. 1017-1020, Sept. 1997.

[56] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications", *Proc. Globecom'89*, Dallas, TX, pp. 47.1.1-7, Nov. 1989.

[57] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes", *IEEE Trans. Inform. Theory*, Vol. 42, March, pp. 429-445, 1996.

[58] J. Hagenauer, P. Robertson, and L. Papke, "Iterative ('Turbo') decoding of systematic convolutional codes with the MAP and SOVA algorithms", *Proc. of the 1994 ITG Conference on Source and Channel Coding* Munich, pp. 1-9, Oct. 1994.

[59] J. Lodge, R. Young, P. Hoeher, and J. Hagenauer, "Separable MAP 'filters' for the decoding of product and concatenated codes", in *Proc. IEEE Int. Conf. Commun.*, pp. 1740-1745, 1993.

[60] J. B. Anderson and S. M. Hladik, "Tail-biting MAP decoders", *IEEE J. on Select. Areas in Commun.*, No. 16, pp. 297-302, Feb. 1998.

[61] L. Ping, Sammy Chan, and Kwan L. Yeung, "Iterative decoding of multi-dimensional concatenated single parity check codes", *IEEE ICC'98*, Atlanta, Ga., pp.131-135, 1998.

[62] L.Wei, "Near Optimal Limited-Search Detection, Part II: Convolutional Codes", Submitted to *IEEE Trans. on Inform. Theory*,

[63] L. Wei, "On bootstrap iterative Viterbi algorithm", *IEEE ICC'99*, Vancouver, Canada.

[64] L. Wei, "Near Shannon limit iterative Viterbi algorithm for Forney concatenated systems," submitted to *IEEE Trans. on Inform. Theory.*

[65] L. Wei, T. Aulin and H. Qi, "On the Effect of Truncation Length on the Exact Performance of a Convolutional Code", *IEEE Trans. on Inform. Theory*, pp.1678-1682, Sept. 1997.

[66] L. Wei, Z. Li, M. James, and I. Petersen, "A minimax robust decoding algorithm", *IEEE Trans. on Inform. Theory*, May 2000.

[67] L. F. Wei, "Trellis coded Modulation with multidimensional constellations", *IEEE Trans. on Inform. Theory*, Vol. IT-33, pp. 483-501, July 1987.

[68] L. F. Wei, "Rotationally invariant convolutional channel coding with expanded signal space — Part I: 180 degrees and Part II: Nonlinear codes", *IEEE J. on Select. Areas in Commun.*, vol. SAC-2, pp. 659-686, Sept. 1984.

[69] L. C. Perez, J. Seghers, and D. J. Costello, Jr., "A distance spectrum interpretation of turbo codes", *IEEE Trans. on Inform. Theory*, vol. IT-42, pp. 1698-1709, Nov. 1996.

[70] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Trans. on Inform. Theory*, pp. 284-287, March 1974.

[71] M. Rouanne and D. J. Costello, Jr., "An algorithm for computing the distance spectrum of trellis codes", *IEEE Journ. on Selec. Areas in Commun.*, vol. 7, no. 6, pp.929-940, Aug. 1989.

[72] M. Sipser and D. A. Spielman, "Expander codes", *IEEE Trans. on Inform. Theory*, vol. IT-42, pp. 1710-1722, Nov. 1996.

[73] M. C. Davey and D. MacKay, "Low density parity check codes over $GF(q)$", *IEEE Communications Letters*, vol. 2, no. 6, pp. 165-168, June 1998.

[74] M. Eyuboglu, G. D. Forney, P. Dong, and G. Long, "Advanced modulation techniques for V.Fast", *Eur. Trans. on Telecom.*, pp. 243-256, May 1993.

[75] M. C. Reed and J. Asenstorfer, "A novel variance estimator for turbo-code decoding", *International Conference on telecommunications*, April 1997.

[76] M. P. C. Fossorier, M. Mihaljević, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation", *IEEE Trans. on Commun.*, vol. 47, no. 5, pp. 673-680, May 1999.

[77] N. Wiberg, "Codes and decoding on general graphs", Ph.D. dissertation, Linköping Univ., Sweden, 1996.

[78] N. Wiberg, H. A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs", in *Proc. of the IEEE Int. Symp. on Inform. Theory*, p. 468, 1995.

[79] P. Jung, "Novel low complexity decoder for turbo-codes", *Electron. Lett.*, vol. 31, pp. 86-87, Jan. 1995.

[80] P. Robertson and T. Wörz, "Novel coded modulation scheme employing turbo codes", *Electron. Lett.*, vol. 31, no. 18, pp. 1546-1547, Aug. 1995.

[81] P. Robertson and T. Wörz, "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes", *IEEE Journal on Selected Areas in Comm.*, Vol. 16, No. 2, pp. 206-218, Feb. 1998.

[82] Q. Wang, L. Wei, and R. A. Kennedy, "Near optimal decoding for trellis-coded modulation using the BIVA and trellis shaping", *Applied Algebra, Algebraic Algorithms and Error-correcting codes - 13th International Symposium, AAECC-13*, pp. 191-200, Honolulu, Hawaii, U.S.A., Nov. 1999.

[83] Q. Wang, L. Wei, and R. A. Kennedy, "Iterative Viterbi Decoding, Trellis Shaping and Multilevel Structure for High-Rate Concatenated TCM", submitted to *IEEE Trans. on Comm.*.

[84] Q. Wang and L. Wei, "Complexity and Performance Comparison of Graph-Based Iterative Decoding Algorithms for Parity-Concatenated Trellis Codes", submitted to *IEEE Trans. on Inform. Theory.*

[85] Q. Wang and L. Wei, "Iterative Viterbi Algorithm for Concatenated Multi-dimensional TCM", accepted by *ISIT'2000*, June 25-30, 2000.

[86] R. G. Gallager, "Low-Density Parity-Check Codes", *IRE Trans Info. Theory* IT-8:21-28, Jan 1962.

[87] R. G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, MA:MIT Press, 1963.

[88] R. M. Tanner, "A recursive approach to low complexity codes", *IEEE Trans. Inform. Theory*, Vol. 27, pp. 533-547, Sept. 1981.

[89] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes", *Proc. of GLOBECOM'94*, San Francisco, California, vol. 1, pp. 339-343, Nov. 1994.

[90] R. H. Deng and D. J. Costello, "High rate concatenated coding systems using bandwidth efficient trellis inner codes", *IEEE Trans. on Commun.*, vol. 37, no. 5, pp. 420-427, May 1989.

[91] R. J. McEliece, D. J. C. MacKay, and J. F. Cheng, "Turbo Decoding as an Instance of Pearl's "Belief Propagation" Algorithm", *IEEE Journal on Selected Areas in Comm.*, Vol. 16, No. 2, pp. 140-152, Feb. 1998.

[92] R. Johannesson and K. Sh. Zigangirov, "Introduction to Convolutional Coding", *IEEE Press*, Piscataway, NJ, 1999.

[93] R. Lucas, M. Bossert, and M. Breitbach, "On iterative soft-decision decoding of linear binary block codes and product codes", *IEEE J. Select. Areas Commun.*, vol. 16, pp. 276-298, Feb. 1998.

[94] R. K. Boel, M. R. James, and I. R. Petersen, "Robustness and risk-sensitive filtering", submit for publication, 1998.

[95] S. LeGoff, A. Glavieux, and C. Berrou, "Turbo codes and high efficiency modulation", *IEEE ICC'94*, pp. 645-649, New Orleans, LA, May 1994.

[96] S. Benedetto and G. Montorsi, "Serial Concatenation of Block and Convolutional Codes", *Electron. Lett.*, Vol. 32, pp. 887-888, 1996.

[97] S. Benedetto and G. Montorsi, "Unveiling turbo-codes: Some results on parallel concatenated coding schemes", *IEEE Trans. on Inform. Theory*, vol. 42, no. 2, pp. 409-428, Mar. 1996.

[98] S. Benedetto and G. Montorsi, "Design of parallel concatenated codes", *IEEE Trans. on Commun.*, pp. 591-600, May 1996.

[99] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Bandwidth efficient parallel concatenated coding schemes", *Electron. Lett.*, vol. 31, Nov. 23, 1995.

[100] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Parallel concatenated trellis coded modulation", *Proceeding of IEEE ICC'96*, pp. 974-978, 1996.

[101] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenated Trellis coded modulation with iterative decoding", *Proceeding of IEEE ISIT'97*, p. 8, Ulm, Germany, June 29 - July 4, 1997.

[102] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding", *IEEE Trans. on Inform. Theory*, vol. 44, no. 3, pp. 909-926, May 1998.

[103] S. Benedetto, R. Garello, and G. Montorsi, "A search for good convolutional codes to be used in the construction of turbo codes", *IEEE Trans. Commun.*, vol. 46, no. 9, pp. 1101-1105, Sep. 1998.

[104] S. Couturier, D. J. Costello, Jr., and Fu-Quan Wang, "Sequential decoding with trellis shaping", *IEEE Trans. on Inform. Theory*, Vol. IT-41, No.6, pp. 2037-2040, Nov. 1995.

[105] S. M. Aji, G. B. Horn, and R. J. McEliece, "On the convergence of iterative decoding on graphs with a single cycle", pp. 276, *ISIT'98*, Cambridge, MA, USA, 1998.

[106] T. A. Summers and S. G. Wilson, "SNR mismatch and online estimation in Turbo decoding", *IEEE Trans. on Commun.*, Vol. 46, No. 4, pp. 421-423, April 1998.

[107] U. Wachsmann and J. Huber, "Power and bandwidth efficient digital communication using turbo codes in multilevel codes", *Euro. Trans. Telecomm.* vol. 6, pp. 557-567, Sept. 1995.

[108] U. Wachsmann, Robert F. H. Fischer, and J. B. Huber, "Multilevel codes: theoretical concepts and practical design rules", *IEEE Trans. on Inform. Theory*, Vol. IT-45, No.5, pp. 1361-1391, July 1999.

[109] W. Blackert and S. Wilson, "Turbo trellis coded modulation", *Proc. CISS'96*, 1996.

[110] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1992.

[111] X. D. Li and J. A. Ritcey, "Trellis-coded modulation with bit interleaving and iterative decoding", *IEEE J.S.A.C.*, vol. 17, no. 4, pp. 715-724, Apr. 1999.

[112] X. L. Huang and Nam Phamdo, "Turbo decoders which adapt to noise distribution mismatch" *IEEE Commun. Letters* Nov. 1998.

[113] Y. Weiss, "Correctness of local probability propagation in graphical models with loops", submitted to *Neural Computation*, 1998. http://www-bcs.mit.edu/ yweiss

[114] Z. Li, L. Wei, M. James, and I. R. Petersen, "On robust decoding algorithms", *International Conference on Communications ICC'99*, Vancouver, Canada, June 1999.

[115] *ITU-T Recommendation V.34*, Sept. 1994.

[116] *Cable-TV access method and physical layer specification*, IEEE Project 802.14/a Draft 2 Revision 2, July 1997.
`http://www.catv.org/modem/standards`

[117] *ANSI TI.413*, 1995.

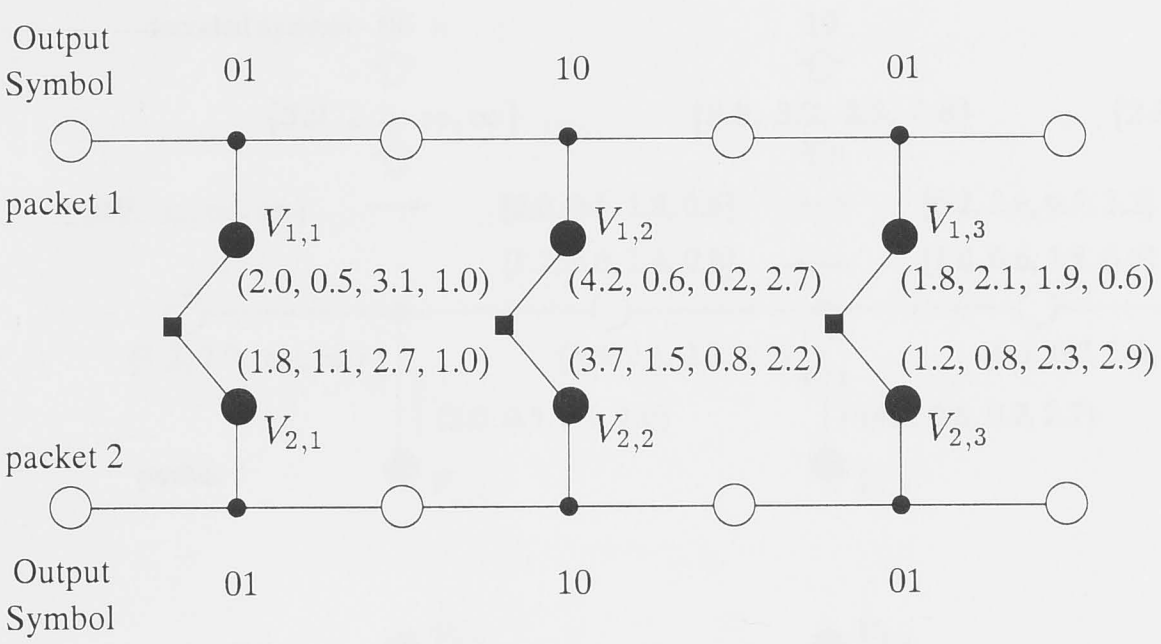
Appendix A

Conventional Iterative Min-sum Algorithm for a Parity-Concatenated Trellis Code

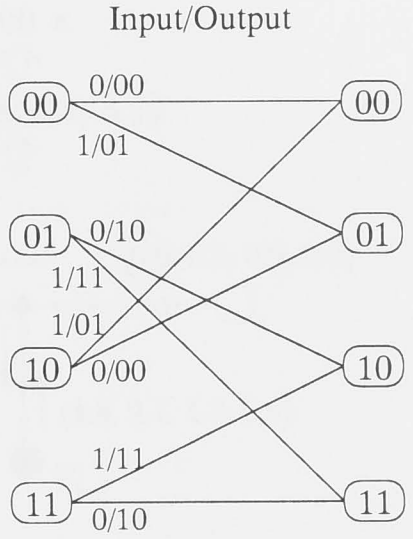
In section 5.1.3, the iterative two-way algorithms are described for decoding the parity-concatenated trellis codes. This appendix gives an example to illustrate how the iterative min-sum algorithm works for the parity-concatenated trellis codes.

Assume that one bit of each symbol is inputted into a linear rate 1/2, 4-state trellis encoder (i.e., $k = 1$, $\tilde{k} = 1$ and $\nu = 2$), and the output symbols are mapped into a QPSK constellation. Figure A-1 (a) shows the TWL graph representation for the parity-concatenated trellis codes on a single-parity-check structure without tail-biting, in which $m = 2$ and $l = 3$. The corresponding trellis diagram of the 4-state trellis encoder is displayed in Fig. A-1 (b).

Due to output symbols in packet 1 and packet 2 satisfy the PC constraint, the outputs of two packets are the same in this example. In the receiver, the smallest branch metric of each of four subsets is calculated as the local likelihood weight of coded symbol node, which represents a four-symbol space: 00, 01, 10 and 11, as shown in Fig. A-1 (a). It will be helpful to regard each edge as having the same set of values as the symbol node to which it is connected.



(a)



(b)

Trellis diagram for the 4-state trellis encoder

(x, x, x, x) denotes the local likelihood weights corresponding to four possible values 00, 01, 10 and 11 of coded symbol node, respectively.

Figure A-1: TWL graph representation for the parity-concatenated trellis codes on a single-parity-check structure with $m = 2$ and $l = 3$

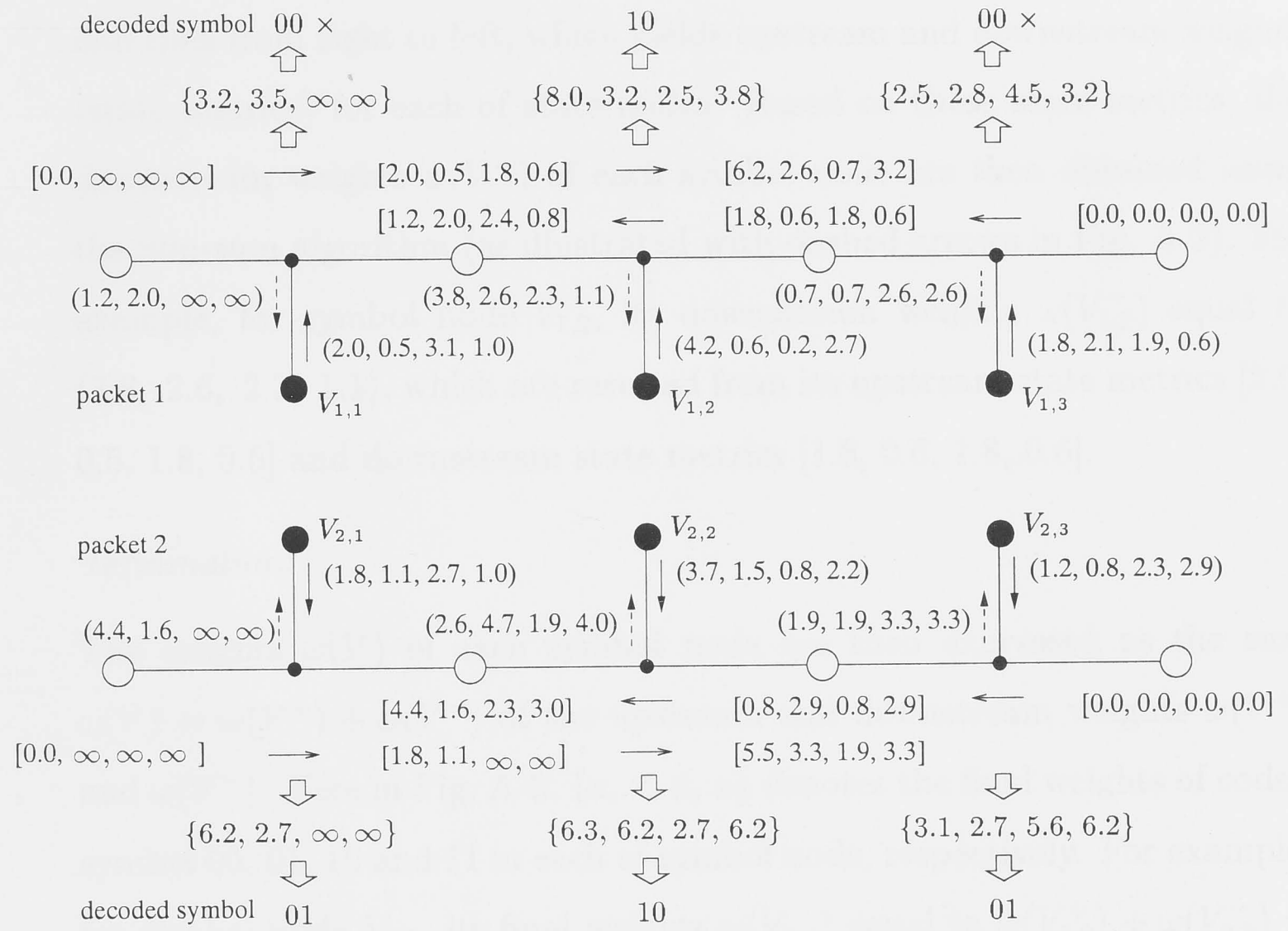
Figure A-2 illustrates the process of iterative decoding for the codes proposed in Fig. A-1. In the first iteration, two packets are decoded independently using the min-sum algorithm, as illustrated in Fig. A-2 (a).

· *Initialization:*

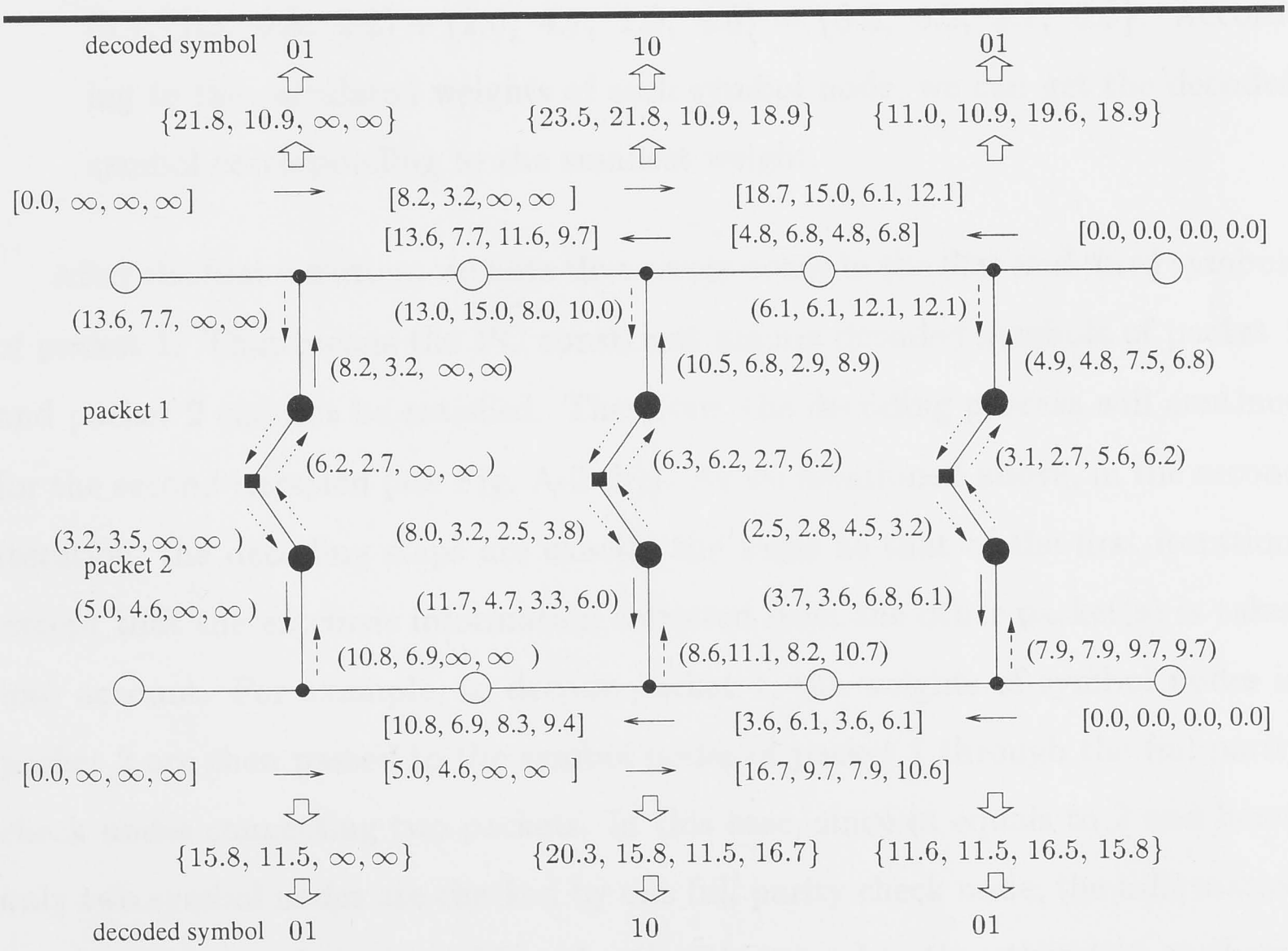
For each of leaf node, the symbol weights, which can be viewed as the upstream weights $\omega(V^+)$ of each symbol node, are simply transferred to the neighbouring edge (as illustrated with solid arrows in Fig. A-2). No computations are required. The local weights of parity check node are set zero. The local weights (state metrics) of state nodes in both ending are initialised as appropriate. Here, [x, x, x, x] in Fig. A-2 denotes the state metric of states 00, 01, 10 and 11, respectively.

· *Updating:*

For each of packets, based on the incoming weights into the parity check nodes, the min-sum (Viterbi algorithm) are performed first from left to right



(a) After first iteration



(b) After second iteration

Figure A-2: Iterative min-sum algorithm for decoding the parity-concatenated trellis codes on a single-parity-check structure with $m = 2$ and $l = 3$

and then from right to left, which yields upstream and downstream weights (state metrics) for each of state nodes. Based on these state metrics, the downstream weights $\omega(V^-)$ of each symbol node are then obtained using the min-sum algorithm (as illustrated with dashed arrows in Fig. A-2). For example, for symbol node $V_{1,2}$, its downstream weights $\omega(V_{1,2}^-)$ equal to (3.8, 2.6, 2.3, 1.1), which are resulted from its upstream state metrics [2.0, 0.5, 1.8, 0.6] and downstream state metrics [1.8, 0.6, 1.8, 0.6].

- *Termination:*

  The weights $\omega(V)$ of each symbol node are then expressed as the sum $\omega(V) = \omega(V^+) + \omega(V^-)$ of the upstream and downstream weights $\omega(V^+)$ and $\omega(V^-)$. Here in Fig. A-2, {x, x, x, x} denotes the final weights of coded symbol 00, 01, 10 and 11 in each of symbol node, respectively. For example, for symbol node $V_{2,2}$, its final weights $\omega(V_{2,2})$ equal to $\omega(V_{2,2}^+) + \omega(V_{2,2}^-) = (3.7,\ 1.5,\ 0.8,\ 2.2) + (2.6,\ 4.7,\ 1.9,\ 4.0) = \{6.3,\ 6.2,\ 2.7,\ 6.2\}$. According to the calculated weights of each symbol node, we can get the decoded symbol corresponding to the smallest weight.

After the first iteration, we note that errors occur in the first and third symbols of packet 1. That means the PC constraint among decoded symbols of packet 1 and packet 2 can not be satisfied. Therefore, the decoding process will continue for the second iteration (see Fig. A-2 (b)). As we mentioned above, in the second iteration, the decoding steps are exactly the same as that of the first iteration, except that the *extrinsic* information collected from the other packet(s) is taken into account. For example, to decode packet 1, the weights of symbol nodes in packet 2 are then passed to the symbol nodes of packet 1 through the full parity check nodes connecting two packets. In this case, since m equals to 2 and hence only two symbol nodes are checked by one full parity check node, the information on one symbol node is then able to be directly passed to the other one, as shown with dotted arrows in Fig. A-2 (b). After the second iteration, we can see that the errors occurred in packet 1 are corrected.