On Distance Measurement Methods for Turbo Codes

Youssouf Ould Cheikh Mouhamedou

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

November 2005

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

To,
My beloved parents Salma Bint Habib and Mohamed Ould Cheikh Mouhamedou
and
in loving memory of my grandparents.

Abstract

New digital communication applications, such as multimedia, require very powerful error correcting codes that deliver low error rates while operating at low to moderate signal-to-noise ratios (SNRs). Turbo codes have reasonable complexity and can achieve very low error rates if a proper interleaver design is in place. The use of well-designed interleavers result in very low error rates, especially for medium to long interleavers where turbo codes offer the greatest potential for achieving high minimum distance ($d_{\min}$) values.

The reliable determination of a code's error performance at very low error rates using simulations may take months or may not be practical at all. However, the knowledge of $d_{\min}$ and its multiplicities can be used to estimate the error rates at high SNR. This thesis is concerned with efficient and accurate distance measurement methods for turbo codes. Since high values of $d_{\min}$ can be caused by high input weight values, say up to 20 or higher, if a brute force algorithm is used the accurate determination of $d_{\min}$ requires that all possible input sequences of input weight up to 20 be tested. Testing all possible input sequences becomes impractical as the size of the interleaver and the value of input weight increase. Thus, the accurate determination of the distance spectrum, or at least $d_{\min}$ and its multiplicities, is a significant problem, especially for interleavers that yield high $d_{\min}$. Based on Garello's true distance measurement method, this thesis presents an efficient and accurate distance measurement method for single- and double-binary turbo codes that uses proper trellis termination such as dual-termination or tail-biting. This method is applied to determine the distance spectra for the digital video broadcasting with return channel via satellite (DVB-RCS) standard double-binary turbo codes. It is also used to design new interleavers for DVB-RCS that yield a significant improvement in error performance compared to those in the standard.

This method fits particularly well with tail-biting turbo codes that use structured interleavers. This is because the distance properties repeat and this method can use this knowledge to reduce the search space. The reduction in search space results in significant reduction in complexity (i.e., execution time), which allows the determination of high $d_{\min}$ values in reasonable time. This efficiency is demonstrated for both single- and double-binary turbo codes, using structured interleavers that have high $d_{\min}$ values for various code rates. This method reduces the execution times by a factor of 40 to 400.

Sommaire

Les récentes applications des communications numériques, comme le multimédia, requièrent des codes correcteurs d'erreurs puissants ayant des taux d'erreur binaire faibles à des rapports signal/bruit moyens et faibles. Les codes turbo ont une complexité acceptable et permettent d'atteindre des taux d'erreur binaire très faibles lorsque munis d'un entrelaceur approprié. L'utilisation d'entrelaceurs performants se traduit par de faibles taux d'erreur binaire particulièrement pour les entrelaceurs de longueur moyenne à élevée qui permettre aux codes turbos d'atteindre des valeurs élevées de distance libre minimum ($d_{\min}$).

L'estimation exacte de la performance d'un code correcteur d'erreurs à très faibles taux d'erreur binaire par le biais de simulations peut prendre des mois ou s'avérer irréalisable. Cependant, la connaissance de $d_{\min}$ et de ses multiplicités peut être utilisée pour estimer le taux d'erreur binaire des codes aux rapports signal/bruit élevés. Cette dissertation étudie certaines méthodes efficaces et précises de calcul de la distribution des poids des codes turbo. Les valeurs élevées de $d_{\min}$ peuvent être dues aux poids d'entrée élevés, jusqu'à 20 ou plus. Si une approche force brute était utilisée pour l'estimation de $d_{\min}$, celle-ci requerrait de tester toutes les séquences d'entrée ayant un poids de 20 et plus. Cette approche devient irréalisable à mesure que la dimension de l'entrelaceur et les valeurs des poids d'entrée augmentent. Par conséquent, l'estimation précise du spectre des distances libres ou du moins $d_{\min}$ et ses multiplicités, est un problème de taille pour les entrelaceurs à $d_{\min}$ élevé. Cette dissertation présente une méthode efficace et précise, basée sur la méthode de distance libre de Garello, du calcul de la distance libre des codes turbo binaire simple et duo-binaire qui utilisent une méthode de terminaison du treillis tel que la dualité ou la circularité. La méthode est appliquée au calcul du spectre des distances libres du code turbo duo-binaire du canal de retour de la norme de radio diffusion par satellite (DVB-RCS). La méthode est également utilisée pour concevoir de nouveaux entrelaceurs pour la norme DVB-RCS qui possèdent un taux d'erreur binaire grandement réduit comparé à celui de la norme.

La méthode convient en particulier aux codes turbo à terminaison circulaire qui utilisent des entrelaceurs structurés parce que la méthode utilise les caractéristiques répétitives de la distribution de poids pour réduire l'espace de recherche. La diminution de l'espace de rechercher entraîne une réduction de complexité importante qui permet l'estimation de $d_{\min}$ dans un délai raisonnable. Cette efficacité est illustrée pour divers taux de codage des codes turbo binaire et duo-binaire à entrelaceur structuré et $d_{\min}$ élevé. La méthode réduit par un facteur de 40 à 400 la période de recherche.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Prof. Peter Kabal, for his motivation, support and guidance. I would also like to express my deep appreciation to Dr. John Lodge at the Communications Research Centre (CRC) in Ottawa not only for giving me the excellent opportunity to work as a part of his team, where I learned theory and applied it to practical and interesting problems, but also for his motivation and financial support. I am very grateful to my supervisor at the CRC, Dr. Stewart Crozier, not only for his guidance, patience, and support, but also for reviewing draft versions of this thesis. It was a great experience to work with Prof. Kabal and Dr. Crozier. Without their help and encouragement, this thesis could not have been completed in a timely manner.

I would like to thank Dr. Paul Guinand at the CRC for the numerous discussions on turbo decoding and reviews that improved the outcome of this thesis. I am very thankful to Prof. Roberto Garello at the Dipartimento di Elettronica of Politecnico di Torino for graciously elaborating on some finer points related to his distance measurement method. Special thanks to Prof. Fabrice Labeau for suggesting that I approach Prof. Kabal as a supervisor for my Ph.D. I am also thankful to my Ph.D. committee members: Prof. Jan Bajcsy and Prof. Frank Ferrie. Special thanks to my external examiner Prof. A. K. Khandani of University of Waterloo, and the members of my Ph.D. defence committee: Prof. P. Kabal, Prof. B. Champagne, Prof. J. Bajcsy, Prof. T. Le-Ngoc, Prof. K. Siddiqi, and Prof. A. Ghrayeb of Concordia University for their valuable comments and suggestions.

I would like to thank Andrew Hunt at the CRC for many fruitful discussions on fundamental questions of coding. I am very thankful to Ken Gracie at CRC for his detailed reviews that improved the quality of both my papers and this thesis. The advice, feedback and reviews of Dr. Ron Kerr at the CRC are greatly appreciated. I am thankful to Dr. François Patenaude at the CRC for his French translation of the thesis abstract.

Special thanks to my friends Abdulkareem Adinoyi at Carleton university and Abdul Rafeeq Abdul-Shakoor at CRC. I would like to thank my fellow graduate students, including Aziz Shalwani, Rickey Der, and Kamal Deep Singh Sahambi. I would like to extend my thanks to my colleagues at the CRC, including Nandeesh Kumar, Pascal Chahine, Charles Benoit, Dr. Hossein Najaf-Zadeh, and Dr. Nikhil Adnani.

Last but certainly not the least, I am indebted to my parents, my aunt – whom I sincerely call, from the depth of my heart, beloved mother – Fatima Bint Habib, my sisters, and brothers for their unfailing confidence, encouragement and love. The opportunities that they have given me and their unlimited sacrifices are the reasons where I am and what I have accomplished so far.

Contents

1 Introduction **1**
1.1 The Structure of a Digital Communication System 2
1.2 Shannon Limit 6
1.3 Motivation and Research Objective 9
1.4 Research Contributions 12
1.5 Overview of Thesis Structure 14

2 Turbo Codes **15**
2.1 Turbo-Code Encoder Structure 15
2.1.1 The Constituent Encoders 16
2.1.2 Interleaving 18
2.1.3 Trellis Termination 23
2.1.4 Puncturing 27
2.2 Turbo Decoding 27
2.3 MAP Algorithm 29
2.4 Soft-decision and Hard-decision Decoding 39
2.4.1 Coding Gain with Soft-decision Decoding 39
2.4.2 Coding Gain with Hard-decision Decoding 41
2.4.3 Soft-decision Decoding versus Hard-decision Decoding 42

3 DVB-RCS Turbo Decoding **44**
3.1 DVB-RCS Encoding Scheme 47
3.1.1 Circular Coding 47
3.1.2 Encoding Process 50
3.1.3 Encoder Structure 50

3.1.4 DVB-RCS Interleaving and Puncturing . . . 51
3.2 Decoding of DVB-RCS Turbo Codes . . . 52
3.2.1 MAP and Log MAP decoding . . . 54
3.2.2 Max-Log MAP . . . 58
3.2.3 Initialization and Iterative Decoding . . . 60
3.2.4 Enhanced Max-Log MAP and Enhanced Log MAP Decoding . . . 63
3.3 Simulation Results . . . 64
3.3.1 Fixed Scale Factor over all Iterations . . . 67
3.3.2 Iteration Dependent Scale Factor . . . 68
3.3.3 Early Stopping . . . 68
3.3.4 The Effect of Overlap on Error Performance . . . 72
3.4 Conclusion . . . 74

4 Distance Measurement Methods for Turbo Codes 78
4.1 Background . . . 79
4.2 Garello's True Method . . . 81
4.2.1 Turbo-code Encoder . . . 81
4.2.2 Computing a Lower Bound on Minimum Distance . . . 83
4.2.3 Recursive Construction of Minimum Distance . . . 89
4.2.4 Techniques to Reduce the Computational Complexity and Memory Requirement . . . 91
4.2.5 Comparison Between Theory and Simulation Results . . . 99
4.2.6 Distance Results . . . 100
4.3 Berrou's Error-Impulse Method . . . 105
4.4 Garello's All-zero Iterative Decoding Method . . . 110
4.5 Crozier's Double-Impulse Iterative Decoding Method . . . 111
4.5.1 Distance Results . . . 113
4.5.2 Complexity Comparison . . . 117
4.6 Conclusion . . . 118

5 An Efficient and Accurate Distance Measurement Method 120
5.1 Background . . . 120
5.2 Complexity Reduction . . . 121

5.2.1 Distance Properties 122
5.2.2 How to Determine the Correct Multiplicity 123
5.3 Remarks on Complexity Reduction when Z is used 126
5.4 Example Distance and Complexity Results 126
5.5 Conclusion 129

6 Conclusions **130**
6.1 Research Achievements 130
6.2 Future Work 132
6.3 Contribution to the Literature 133

References **135**

List of Figures

1.1 Model of a digital communication system. 2
1.2 Binary symmetric channel. 6
1.3 Normalized channel capacity versus channel SNR. 8
1.4 Region of turbo code performance. Error rates are in base 10 logarithmic scale and SNRs (dB) are in linear scale. 10

2.1 Turbo-code encoder structure. 16
2.2 8-state constituent encoder used in UMTS turbo codes 18
2.3 Dithered relative prime (DRP) interleavers. 23
2.4 Turbo decoder structure. 29
2.5 Mapping the first two bits (c_1, c_2) and the last two bits (c_3, c_4) of a 16-QAM signal point to two independent 4-PAM signals. 36

3.1 Double-binary CRSC encoder for DVB-RCS turbo code and its corresponding trellis diagram. 48
3.2 A simple system model of DVB-RCS. 54
3.3 Different branches involved in the computation of the Log-Likelihood Ratio of the information symbol (10) for DVB-RCS turbo codes. 55
3.4 Alpha(0) and Beta(0) extracted from trellis-diagram. 56
3.5 Computing of alphas and betas for circular decoder. 62
3.6 Iterative decoder for DVB-RCS turbo code. 63
3.7 Performance of ML-MAP decoding for ATM packet size and various code rates using 8 iterations. 65
3.8 Performance of ML-MAP decoding for MPEG packet size and various code rates using 8 iterations. 66

3.9 Comparison of four decoding algorithms for DVB-RCS with ATM packet size. 69
3.10 Comparison of four decoding algorithms for DVB-RCS with MPEG packet size. 70
3.11 Performance of EML-MAP decoding using ramped and fixed scale factor. . 71
3.12 Performances of EML-MAP decoding without early stopping is compared to that of EML-MAP decoding with early stopping (B=2). 73
3.13 Performance of ML-MAP decoding for ATM packet sizes using rate 1/3 and various overlap lengths. 75
3.14 Performance of ML-MAP decoding for ATM packet sizes using rate 4/5 and various overlap lengths. 76
3.15 Performance of ML-MAP decoding for MPEG packet sizes using rate 1/3 and various overlap lengths. 77

4.1 A general turbo-code encoder with two recursive convolutional encoders. . 82
4.2 Trellis diagram, termination-bits and its corresponding weights for an encoder with generator polynomials (feedback,feedforward)=$(7,5)_{octal}$. 86
4.3 Examples showing how to compute $\mathrm{MWE2}^2_{tb}$ and $\mathrm{MWE2}^2_{\mathrm{umts}}$ for tail-biting and UMTS-termination, respectively. 87
4.4 Examples showing how to compute $\mathrm{MWE1}^2_{tb}$ and $\mathrm{MWE1}^2_{\mathrm{umts}}$ for tail-biting and UMTS-termination, respectively. 90
4.5 Example showing how to use the modified definition to compute $\overline{\mathrm{MWE1}}^2_{tb}$ and $\overline{\mathrm{MWE2}}^2_{tb}$ for $\boldsymbol{u}^2 = (0,1,0,\times,\times,\times,\times)$ and the interleaver $\pi = (3,1,6,0,5,2,4)$. 93
4.6 Cross section of a labelled binary tree for ENC1, where ENC1 starts in state s_1 and the basis input sequence is $\boldsymbol{u}^2 = (0,1,0,\times,\cdots,\times)$. 95
4.7 Figure illustrating how to obtain $\overline{\mathrm{MWE2}}^j_0$ and $\overline{\mathrm{MWE2}}^j_1$. 97
4.8 The forward and backward MVA can be stopped early at trellis sections ES_LEFT and ES_RIGHT, respectively. 97
4.9 FER and BER for MPEG-sized random interleaver of code rate R_c=1/3 (QPSK/AWGN). 99
4.10 FER and BER for interleavers of size 512 and code rate R_c=1/3 (BPSK/AWGN).102
4.11 FER and BER for ATM packets of code rate R_c=1/3 (QPSK/AWGN). . . 106
4.12 FER for MPEG packets of code rate R_c=1/3 (QPSK/AWGN). 107
4.13 BER for MPEG packets of code rate R_c=1/3 (QPSK/AWGN). 108

4.14 Two events for the DVB-RCS standard packet of size 64 symbols (128 bits). 111
4.15 For DVB-RCS standard encoder and a random interleaver of size 48 symbols (96 bits), the depicted two events cause the true $d_{\min}$. The first encoder starts and ends in state '001'. The second encoder starts and ends in state '011'. 116
4.16 For DVB-RCS standard encoder and a random interleaver of size 48 symbols (96 bits), the depicted two events cause the true $d_{\min}$. Both encoders start and end in the all-zero state. 116

List of Tables

1.1 Capacity limit for some selected code rates R_c using *continuous-input continuous-output* and *binary-input continuous-output* AWGN channel. 8

3.1 Circulation state correspondence table. 50
3.2 Turbo code permutation parameters. 52
3.3 Turbo code puncturing patterns. 53

4.1 Distance results ($d_{\min}/A_{\mathrm{d}_{\min}}/W_{\mathrm{d}_{\min}}$), and the next four distance terms, for random interleaver with code rate $R_c = 1/3$. 100
4.2 Distance results ($d_{\min}/A_{\mathrm{d}_{\min}}/W_{\mathrm{d}_{\min}}$), and the next two distance terms, for 4 UMTS interleavers and DRP interleavers with code rate $R_c = 1/3$. 101
4.3 Distance results ($d_{\min}/A_{\mathrm{d}_{\min}}/W_{\mathrm{d}_{\min}}$) for the 12 DVB-RCS standard interleavers. 103
4.4 Distance results ($d_{\min}/A_{\mathrm{d}_{\min}}/W_{\mathrm{d}_{\min}}$) for DVB-RCS with exhaustive search for DRP interleavers with $M = 4$. 104
4.5 Distance results ($d_{\min}$) for rate 1/3 DVB-RCS codes with an exhaustive search for DRP interleavers with M=1, 2 and 4. 104
4.6 Comparison of minimum distances obtained with Crozier's double-impulse method (DIM) using limited and full range search for the second impulse. . 113
4.7 Distances shown here are for the code rate 1/3. DVB-RCS standard interleavers were used with normal early stopping. 114
4.8 Distances for MPEG size (K=752 2-bit symbols) using DVB-RCS standard interleaver and standard puncturing. 115
4.9 Distances shown here are obtained using DVB-RCS standard encoder and code rate 1/3. 1000 random interleavers were tested. 116

4.10 Distances shown here are obtained using DVB-RCS standard encoder and code rate 1/3. 20 DRP interleavers were tested. 117
4.11 Distance results ($d_{\min}$) and CPU times in minutes for rate 1/3 DVB-RCS turbo-code encoder with various MPEG-sized interleavers (752 2-bit symbols).118

5.1 Minimum distances, multiplicities and CPU times in minutes for the DVB-RCS turbo-code encoder with the MPEG-sized standard interleaver. 128
5.2 Minimum distances, multiplicities and CPU times in minutes for the DVB-RCS turbo-code encoder with new MPEG-sized DRP interleavers. 128
5.3 Minimum distances, multiplicities and CPU times in minutes for the UMTS turbo-code encoder with new MPEG-sized DRP interleavers. 128

Glossary of Acronyms

The following acronyms and terms are used within this thesis.

3GPP	Third Generation Partnership Project
AWGN	Additive White Gaussian Noise
APP	*A Posteriori* Probability
ATM	Asynchronous Transfer Mode
BCJR	Bahl, Cocke, Jelinek and Raviv
BER	Bit Error Rate
BPSK	Binary Phase-Shift Keying
BSC	Binary Symmetric Channel
CRSC	Circular Recursive Systematic Convolutional
dB	Decibel
DEC	Decoder
DD	Dithered-Diagonal
DIM	Double Impulse Method
DMC	Discrete Memoryless Channel
DRP	Dithered Relative Prime
DSL	Digital Subscriber Line
DVB-RCS	Digital Video Broadcasting with Return Channel Via Satellite
ECC	Error Correction Coding
EIM	Error Impulse Method
EL-MAP	Enhanced Log Maximum *A Posteriori*
EML-MAP	Enhanced Max-Log Maximum *A Posteriori*
ENC	Encoder
ETSI	European Telecommunications Standards Institute

XOR	Exclusive OR
FER	Frame Error Rate
FRS	Full Range Search
FSK	Frequency-Shift Keying
GEO	Geostationary Earth Orbit
HSR	High-Spread Random
Hz	Hertz
ISI	Intersymbol Interference
IW	Input Weight
LDPC	Low-density parity-check
LMDS	Local Multipoint Distribution Service
LLR	Log Likelihood Ratio
LRS	Limited Range Search
MAP	Maximum *A Posteriori*
ML	Maximum Likelihood
ML-MAP	Max-Log Maximum *A Posteriori*
MPEG	Motion Picture Experts Group
MF-TDMA	Multi-Frequency Time Division Multiple Access
MVA	Modified Viterbi Algorithm
MW^j	Minimum weight of turbo-code encoder caused by the input sequence $\boldsymbol{u}^j = (u_0, \cdots, u_j, 0, \cdots, 0)$
$MWE1^j$	Minimum weight of first encoder caused by the input sequence $\boldsymbol{u}^j$
$MWE2^j$	Minimum weight of second encoder caused by the input sequence $\boldsymbol{u}_\pi^j$
PAM	Pulse Amplitude Modulation
PCCC	Parallel Concatenated Convolutional Codes
PSD	Power Spectral Density
PSK	Phase-Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-shift Keying
RP	Relative Prime
RSC	Recursive Systematic Convolutional
SCCC	Serially Concatenated Convolutional Codes
SF	Scale Factor

SIM	Single Impulse Method
SISO	Soft-In Soft-Out
SIT	Satellite Interactive Terminal
SNR	Signal-to-Noise Ratio
SOVA	Soft-Output Viterbi-Algorithms
TCM	Trellis-Coded Modulation
TDM	Time Division Multiplex
TIM	Triple Impulse Method
TUB	Truncated Union Bound
UMTS	Universal Mobile Telecommunications System
WE1	Hamming weight resulting from first encoder
WE2	Hamming weight resulting from second encoder

Glossary of Symbols

The following mathematical symbols are used in this thesis.

Greek Symbols

σ^2	Variance of the AWGN
η	Spectral efficiency
$\eta_{\max}$	maximum spectral efficiency
ϵ	Probability of bit error for a binary symmetric channel (crossover probability)
τ_r	Number of rows in a matrix
τ_c	Number of columns in a matrix
δ	Number of delays in an RSC encoder
Δ	Number of states of an RSC encoder
Ω	Denotes the set of all states of an RSC encoder
ϕ^0	Set of all possible input symbols
ϕ	Set of all possible non-zero input symbols
$\alpha_t(s)$	Forward state probability
$\beta_{t-1}(s')$	Backward state probability
$\gamma_t^{(z)}(s', s)$	Transition probability from state s' to state s at time t caused by input symbol z
π	Interleaver
π^{-1}	De-interleaver (inverse interleaver)
χ	A non-zero symbol in an input $\boldsymbol{u}_{\min}$ that causes $d_{\min}$
μ	The average number of tested amplitudes in Berrou's Error-Impulse method
λ	The average number of iterations

Latin Symbols

$A_{\min}$	Number of minimum distance codewords
B	Baseband bandwidth
W	Number of consecutive sets of hard decisions that must agree before stopping
C	Channel capacity
$C(\widetilde{N}, \widetilde{K})$	Linear code
$C(\widetilde{N}, \widetilde{K}, t)$	Linear code capable of correcting at most t-error
$d^E(\boldsymbol{x}_i, \boldsymbol{x}_k)$	Euclidian distance between the vector $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$
$d^H(\boldsymbol{c}_i, \boldsymbol{c}_k)$	Hamming distance between the two codewords $\boldsymbol{c}_i$ and $\boldsymbol{c}_k$
$d_{\min}, d^H_{\min}$	Minimum Hamming distance
D	Delay operator
D^n	Delay of n symbol times
E	Average energy of un-coded or coded bit
E_b, E_b^u	Average energy per un-coded information bit
E_b^c	Average energy for coded bit
F	Number of constituent encoders
G^{soft}	Coding gain with soft-decision decoding
G^{hard}	Coding gain with hard-decision decoding
$\boldsymbol{G}$	Generator matrix
I	I-Component of an M-ary modulation
$\boldsymbol{I}$	Identity matrix
k	Constraint length
k	Size of an input symbol in bits that enters RSC encoder
K	Size of information sequence in symbols
$\widetilde{K}$	Size of information sequence in bits
$\ell(i)$	The number of consecutive zero symbols immediately preceding a non-zero symbols at position i in $\boldsymbol{u}^j_{\min}$
$\ell_e(i)$	The number of consecutive zero symbols immediately preceding an error event that starts at position i in $\boldsymbol{u}^j_{\min}$
L	Period of the feedback polynomial
L_{apo}	*A posteriori* information

$L^t_{apo}, L^{(z)}(\hat{u}_t)$	*A posteriori* information for input symbol z at time t (i.e., $u_t = z$)
L_{apr}	*A priori* information
L^t_{apr}	*A priori* information for input symbol u_t
L_{ex}	Extrinsic information
L_{ex1}	Extrinsic information produced by first decoder
L_{ex2}	Extrinsic information produced by second decoder
L_{in}	Intrinsic information
L^t_{in}	Intrinsic information at time t
m	Number of bits represented by a single constellation point
m	Number of output sequences generated by first encoder
M	Number of constellation points
M	Number of repeating index increments for structured interleaver
M	Overlap size in symbols for tail-biting
n	Number of output sequences generated by second encoder
N	Size of codeword in symbols
$\widetilde{N}$	Size of codeword in bits
N_0	One-sided noise power spectral density
P_b	Bit error probability
P_n	Average noise power
$\boldsymbol{P}_i$	Parities of the i constituent encoder
P_s	Average signal power
P_w	Message error probability
q	Number of parity bits generated by constituent encoder at any time instant
q	Total number of delays in first and second encoder
Q	Q-Component of an M-ary modulation
R	Length of read dither vector
R_b	Data rate
R_c	Code rate
R_c^{-1}	Reciprocal of code rate
R_e	Data rate at the output of the channel encoder
R_m	Data rate at the output of the modulator
R_s	Symbol rate

s	Encoder state at time t
s'	Encoder state at time $(t-1)$
S	Amount of separation greater than S must be between any S consecutive elements in a S-random interleaver
$\boldsymbol{S}_c$	Circular state
$S''_{new}(i,j)$	Spread between two elements i and j in interleaver
$S'_{new}(i)$	Minimum spread associated with index i
S_{new}	Overall minimum spread
$\boldsymbol{u}$	Input sequence to the encoders
$\boldsymbol{u}_{\min}$	Input sequence that causes $d_{\min}$
$\hat{u}_t$	Estimated symbol at time t
$\boldsymbol{v}_t$	Output of RSC encoder at time t caused by input symbol u_t
W	Length of write dither vector
$W_{\min}$	Information bit multiplicity (i.e., sum of hamming weights of input sequences causing minimum distance)
$\boldsymbol{y}$	The output of the demodulator
$\boldsymbol{y}^{P1}$	Received parities of first encoders
$\boldsymbol{y}^{P2}$	Received parities of second encoders
$\boldsymbol{y}^{s}$	Received systematic part
$\boldsymbol{y}^{s}_{\pi}$	Interleaved version of the received systematic part

Operators

$\approx$	Approximately equal to
$>>$	Much greater than
$<<$	Much less than
$\max(\cdot)$	Returns the value of the biggest element of the argument
$\min(\cdot)$	Returns the value of the smallest element of the argument
$\arg\max_x(f(x))$	Return the argument x that maximizes the function $f(x)$
$\lfloor x \rfloor$	The largest integer $\leq x$
$[x]_K$	x modulo K
$\|\cdot\|$	Multidimensional Euclidian distance
$\binom{N}{n}$	Binomial coefficient

$\oplus$	XOR operation
$Q(\cdot)$	Gaussian Q function
$\ln(x)$	Base e, logarithm of x

Chapter 1

Introduction

The efficient design of a communication system that enables reliable high-speed services is challenging. 'Efficient design' refers to the efficient use of primary communication resources, namely, power and bandwidth. The reliability of such systems is usually measured by the required signal-to-noise ratio (SNR) to achieve a specific error rate. Also, a bandwidth efficient communication system with perfect reliability, or as reliable as possible, using as low a SNR as possible is desired. In order to enable a fair comparison between the reliability of different communication systems, the SNR is usually expressed as $\frac{E_b}{N_0}$, where E_b is the average energy per information bit and N_0 the one-sided noise power spectral density (PSD).

Error correction coding (ECC) is a technique that improves the reliability of communication over a noisy channel. The use of the appropriate ECC allows a communication system to operate at very low error rates, using low to moderate SNR values, enabling reliable high-speed multimedia services over a noisy channel. Although there are different types of ECC that have their roots in diverse mathematical disciplines, they all have one key ingredient in common, namely, achieving a high minimum Hamming distance[1] that occurs only for few codewords (i.e., low multiplicity). This is because achieving high Hamming distance between codewords results in high Euclidean distance between modulated codewords and it takes a lot of noise sample together to cause an error.

In this chapter, a basic communication system, the Shannon capacity limit and the motivation for this thesis are discussed. A list that summarizes the main contributions of

[1]Minimum Hamming distance is the minimum number of bits that must be changed in order to convert one codeword into another.

this work is introduced.

1.1 The Structure of a Digital Communication System

The *information source* generates a message containing information that is to be transmitted to the receiver. The information source can be an *analog source* that generates analog signals such as audio or video. An analog communication system transmits such a signal directly over the channel using analog modulation such as amplitude, frequency, or phase modulation. The information source can also be a *discrete source* that generates a sequence of symbols from a finite symbol alphabet. A teletype machine, where the output consists of a sequence of symbols from a given alphabet and the numbers 0 through 9, is an example of a discrete source.

In a digital communication system, shown in Fig. 1.1, the outputs of an analog or discrete source are converted into a sequence of bits. This sequence of bits might contain too much redundancy. The redundancy in the source might not be useful for achieving high reliability. Ideally, the *source encoder* removes redundancy and represents the source output sequence with as few bits as possible. Note that the redundancy in the source is different from the redundancy inserted intentionally by the error correcting code.

The *encrypter* encodes the data for security purposes. Encryption is the most effective way to achieve data security [1]. The three components, information source, source encoder and encrypter can be seen as a single component called the *source*. The binary sequence $\boldsymbol{u}$ is the output of the source. The number of bits the source generates per second is the data rate R_b and is in units of bits per second (bps or bits/s).

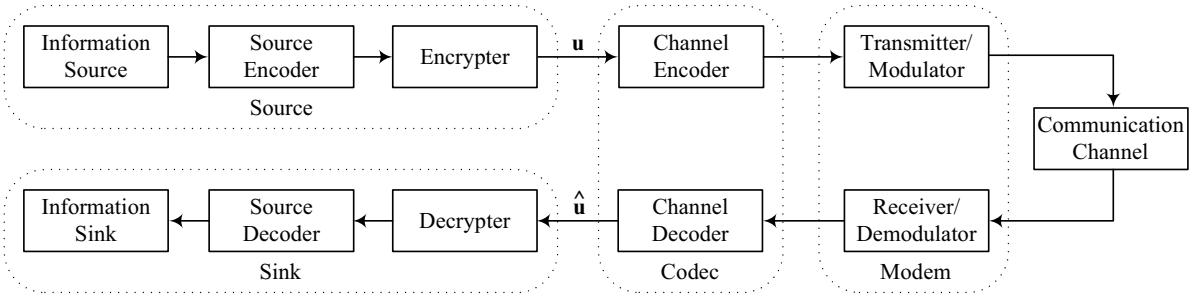


Fig. 1.1 Model of a digital communication system.

The primary goal of the *channel encoder* is to increase the reliability of transmission

within the constraints of signal power, system bandwidth and computational complexity. This can be achieved by introducing structured redundancy into transmitted signals [2]. Channel coding is used in digital communication systems to correct transmission errors caused by noise, fading and interference. In digital storage systems, channel coding is used to correct errors caused by storage medium defects, dust particles and radiation. The channel encoder assigns to each message of $\widetilde{K}$ bits a longer message of $\widetilde{N}$ bits called a *codeword*. This usually results in either a lower data transmission rate or increased channel bandwidth relative to an un-coded system. The data rate at the output of the channel encoder is $R_e = \frac{R_b}{R_c}$ bits/s, where $R_c = \frac{\widetilde{K}}{\widetilde{N}}$ is the *code rate*. To make the communication system less vulnerable to channel impairments, the channel encoder generates codewords that are as different as possible from one another.

Since the transmission medium is a waveform medium, the sequence of bits generated by the channel encoder can not be transmitted directly through this medium. The main goals of *modulation* are not only to match the signal to the transmission medium, enable simultaneous transmission of a number of signals over the same physical medium and increase the data rate, but also to achieve this by the efficient use of the two primary resources of a communication system, namely, *transmitted power* and *channel bandwidth*. The channel bandwidth is the band of frequencies allocated for the transmission of the message signal. The *modulator* maps m encoded bits into $M = 2^m$ distinct waveforms. Consequently, the symbol rate at the output of the modulator is $R_s = \frac{R_b}{R_c \cdot m}$ symbols/s. The M waveforms can be obtained by varying the amplitude as in pulse amplitude modulation (PAM), the phase as in phase-shift keying (PSK), the frequency as in frequency-shift keying (FSK) or both amplitude and phase (PAM-PSK), which is also known as quadrature amplitude modulation (QAM) [3]. For transmission that is free of intersymbol interference, Nyquist's criterion sets a limit of one pulse per single orthogonal signaling dimension. Since $\sin(2\pi f_c t)$ and $\cos(2\pi f_c t)$ (f_c is the carrier frequency) are orthogonal and can be sent using the same bandwidth, one can send two pulses (one on in-phase and one on quadrature-phase) at the same time. Thus, one can send at maximum $2B$ pulses per second in a bandwidth of B Hz. This bandwidth refers to the one-sided positive frequencies of the real signal. Therefore, the minimum signal bandwidth is $B = \frac{R_s}{2} = \frac{R_b}{2R_c \cdot m}$. This suggests that error correcting codes can be used with higher order modulation schemes (M-ary) without any increase in the bandwidth, which is often fixed, as is the case for telephone channel (B is about 3 kHz). For example, a communication system with data rate R_b can be realized using the

same bandwidth with:

(a) un-coded QPSK modulation, which also results in a symbol rate of $R_s = \frac{R_b}{m} = \frac{R_b}{2}$ symbols/s;

(b) error correcting code of code rate $R_c = 2/3$ and 8-PSK modulation, which also results in a symbol rate of $R_s = \frac{R_b}{R_c \cdot m} = \frac{R_b}{\left(\frac{2}{3}\right)\cdot 3} = \frac{R_b}{2}$ symbol/s.

In (b), the potential increase in the bandwidth by a factor of R_c^{-1} due to the use of the error correcting code is compensated by the use of 8-PSK modulation. However, higher M-ary modulation schemes are more susceptible to noise than the simple and robust binary phase-shift keying (BPSK) and quadrature phase-shift keying (QPSK). This is because the minimum Euclidean distance between the symbol points of M-ary modulation is smaller than that for BPSK/QPSK for a fixed average transmitted power. Thus, in order to achieve a coding gain, the greater susceptibility of M-ary modulation to error *must be more* than compensated for by the error correcting code. For the example above, the use of un-coded 8-PSK instead of un-coded QPSK results in an $\frac{E_b}{N_0}$ loss of about 3.6 dB[2] at a bit error rate (BER) of 10^{-5} [4]. The use of a convolutional code of constraint length $k = 6$ (i.e., memory 5) and a code rate of $R_c = 2/3$ together with QPSK leads to a coding gain in $\frac{E_b}{N_0}$ of about 4.2 dB at BER $= 10^{-5}$ compared to un-coded QPSK [5]. Assuming the same coding gain holds for 8-PSK, one can expect that the use of a 32-state 2/3-rate convolutional code together with 8-PSK results in a coding gain of about $(4.2 - 3.6) = 0.6$ dB in $\frac{E_b}{N_0}$ at a BER $= 10^{-5}$ compared to un-coded QPSK. This example shows that a 'trivial' mapping of coded bits to the M-ary signal points does not necessarily result in a significant reduction in $\frac{E_b}{N_0}$.

In 1982, Ungerboeck presented a new coding method called trellis-coded modulation (TCM) [6, 7, 8, 9] which does not expand the bandwidth yet achieves a significant coding gain. The strength of this method lies in the treatment of modulation and channel coding as a *combined* entity rather than two separate operations. Ungerboeck presented a set of rules, known as *set partitioning*, that allow a 'non-trivial' mapping of coded bits into the M-ary signal points, and mapping of these points to the trellis of the convolutional encoder (usually of rate $\frac{m}{m+1}$). With Ungerboeck's original mapping rule, a 32-state 8-PSK TCM yields a coding gain of 3.3 dB at a BER of 10^{-5} compared to un-coded QPSK [10]. This

[2]The SNR is usually measured in decibels, according to $\frac{E_b}{N_0}$ (dB) $= 10 \log_{10}\left(\frac{E_b}{N_0}\right)$.

coding gain of 3.3 dB is significant compared to the case discussed above with a gain of only 0.6 dB. As shown in [10] and [11], different mapping rules can improve the SNR required to achieve a given BER by a further 0.5 dB. The asymptotic gain with 8-PSK TCM compared to un-coded QPSK is 4.59 dB [12].

A *communication channel* refers to the combination of physical medium (copper wires, radio medium or optical fiber) and electronic or optical devices (equalizers, amplifiers) that are part of the path followed by a signal (Fig. 1.1). Channel noise, fading and interference corrupt the transmitted signal and cause errors in the received signal. This thesis considers only AWGN type channels, which ultimately limit system performance. Note that many interference sources and background noise can be modelled as AWGN due to the central limit theorem. A major contributor to AWGN is often the thermal noise due to the electronic equipment such as amplifiers used in the transmitter and receiver. A communication channel is a system in which the output depends probabilistically on the input. If the probability distribution of the output depends only on the input at that time and is independent of previous channel inputs or outputs, the channel is said to be *memoryless*. Suppose that the modulator and demodulator are considered to be parts of a memoryless channel. The combined channel has a discrete-time input and a discrete-time output and can be represented by a discrete memoryless channel (DMC). Such a composite channel is characterized by the set of possible inputs, the set of possible outputs and the conditional distribution of the output given the input.

The simplest channel model is obtained when the probability of error for binary value 0 and 1 are the same and the channel is memoryless. This channel is shown in Fig. 1.2 and is known as the binary symmetric channel (BSC). As an example, the use of BPSK or Gray labelled QPSK signaling over AWGN channel and a matched filter receiver results in bit error probability of $Q\left(\sqrt{\frac{2E}{N_0}}\right)$ [13], where $Q(x)$ is the tail probability of a Gaussian random variable with zero mean and variance $\sigma^2 = 1$ (i.e., $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt$), and $E = R_c \cdot E_b$ if coding is applied and $E = E_b$ if no coding is used. Since the bit error probability is symmetric with respect to the value of the transmitted bit, $p_{X|Y}(1|0) = p_{X|Y}(0|1) = \epsilon = Q\left(\sqrt{\frac{2E}{N_0}}\right)$, where $X \in \{0, 1\}$ and $Y \in \{0, 1\}$ are the input and output of the BSC channel, respectively.

At the receiving end of the communication system, the *demodulator* processes the channel-corrupted transmitted waveform and makes a hard or soft decision on each symbol. If the demodulator makes a hard decision, its output is a binary sequence and the sub-

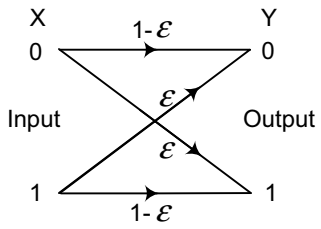


Fig. 1.2 Binary symmetric channel.

sequent channel decoding process is called *hard-decision decoding*. A hard decision in the demodulator results in some irreversible information loss. If the demodulator passes the soft output of the matched filter to the decoder, the subsequent channel decoding process is called *soft-decision decoding*.

The *channel decoder* works separately from the modulator/demodulator and has the goal of estimating the output of the source encoder based on the encoder structure and a decoding algorithm. In general, with soft-decision decoding, approximately 2 dB and 6 dB of coding gain with respect to hard-decision decoding can be obtained in AWGN channels and fading AWGN channels, respectively [14].

If encryption is used, the *decrypter* converts encrypted data back into its original form. The *source decoder* transforms the sequence at its input based on the source encoding rule into a sequence of data, which will be used by the *information sink* to construct an estimate of the message. These three components, decrypter, source decoder and information sink can be represented as a single component called the *sink*, as far as the rest of the communication system is concerned. The binary sequence $\widehat{\boldsymbol{u}}$ is the input to the sink.

1.2 Shannon Limit

Both the communication channel and the signal that travels through it have their own bandwidth. The bandwidth B of a communication channel defines the frequency limits of the signals that it can carry. In order to transfer data very quickly, a large bandwidth is required. Unfortunately, every communication channel has a limited bandwidth, whether constrained artificially by communications standards, such as the digital video broadcasting with return channel via satellite (DVB-RCS) standard [15], or physically by the transmission medium such as copper wire. Amplifiers or repeaters, used to boost signals so that they can travel longer distances, also impose bandwidth constraints. Limitations on the

data rate may be quantified by the *spectral efficiency* $\eta = \frac{R_b}{B}$ bits/s/Hz, where R_b is the data rate at the output of the source (see Fig. 1.1).

In his landmark paper, Shannon proved in 1948 that if the data rate R_b at the output of a source is less than a quantity called the *channel capacity*, C, then communication over an AWGN channel with an error probability as small as desired is possible with proper encoding and decoding [16, 17]. This fundamental result showed that noise sets a limit on the data rate but not on the error probability, as had been widely believed before [18]. For a continuous input to an AWGN channel, the channel capacity is given by the Shannon-Hartley formula [19], which assumes a *random* codeword of length approaching infinity and *maximum likelihood* (ML) decoding. This formula may be expressed as

$$C = B \log_2 \left(1 + \frac{P_s}{P_n}\right) \qquad \text{bits/s.} \tag{1.1}$$

where P_s is the average signal power and P_n is the average noise power.

Assuming that the data rate is equal to the channel capacity (i.e., $R_b = C$), equation (1.1) can be expressed as

$$\frac{C}{B} = \log_2 \left(1 + \frac{C}{B}\frac{E_b}{N_0}\right) \qquad \text{bits/s/Hz.} \tag{1.2}$$

From (1.2) it follows that the minimum required SNR, $\frac{E_b}{N_0}$, for error-free transmission is -1.59 dB [18], assuming that the bandwidth B approaches infinity and the code rate approaches zero. Shannon's theorem sets a limit for the coding gain that can be achieved by error correcting codes over an AWGN channel relative to un-coded modulation schemes.

Practical systems operate with a non-zero error probability. For instance, un-coded QPSK has a maximum spectral efficiency of $\eta_{\max} = 2$ bits/s/Hz and requires $\frac{E_b}{N_0} = 9.6$ dB to achieve BER=10^{-5} in AWGN. With coding, and no limitations on the signalling levels, the same spectral efficiency can be achieved at $\frac{E_b}{N_0} = 1.8$ dB (Fig. 1.3), implying an achievable coding gain of $\frac{E_b}{N_0} = (9.6 - 1.8) = 7.8$ dB.

The capacity limit depends on the code rate. Table 1.1 shows that a rate-1/2 code requires a higher SNR by about 0.7 dB, 1.0 dB and 1.8 dB compared to rate-1/3, rate-1/4 and the ultimate capacity limit (code rate approaching 0), respectively, assuming a *binary-input continuous-output* AWGN channel [20]. Since the Shannon coding theorem does not say how to design specific codes achieving the maximum possible data rate at arbitrarily

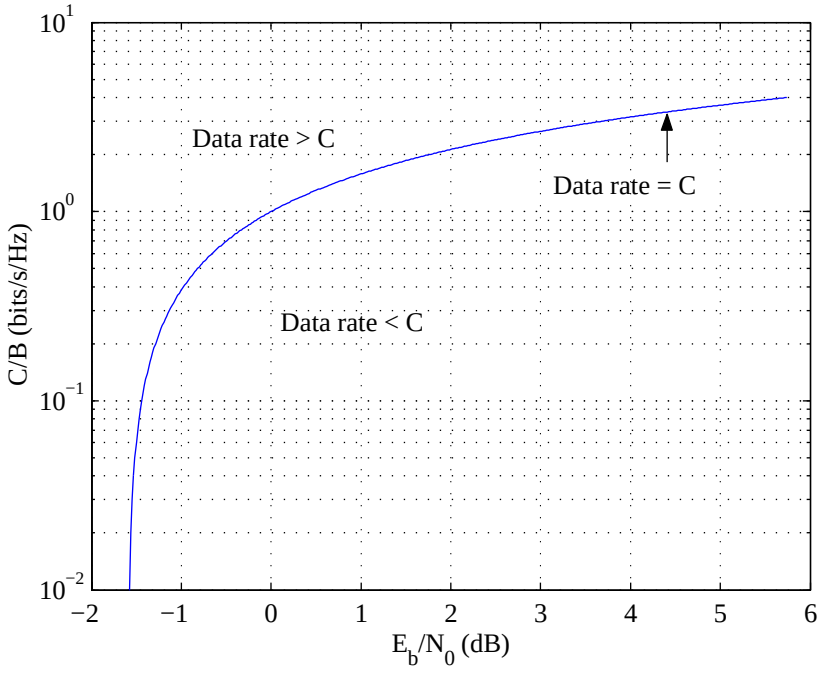


Fig. 1.3 Normalized channel capacity versus channel SNR.

Table 1.1 Capacity limit for some selected code rates R_c using *continuous-input continuous-output* and *binary-input continuous-output* AWGN channel. Table is taken from [20].

	Capacity limit in dB	
R_c	Continuous input	Binary input
1/2	0.00	0.19
1/3	-0.55	-0.50
1/4	-0.82	-0.79
1/6	-1.08	-1.07
0	-1.59	-1.59

small error probabilities, a great deal of effort has been expended to find efficient encoding and decoding methods that approach the Shannon limit.

An important breakthrough in coding since the introduction of TCM in 1982 has been the introduction of a new class of parallel concatenated convolutional codes (PCCC), called turbo codes in 1993 [21]. The code consists of three parts: (a) the original information sequence, (b) redundancy bits generated by the first constituent encoder that has the original information sequence as an input and (c) redundancy bits generated by the second constituent encoder that has a permuted version of the original information sequence as an input. Turbo codes achieve good performance with reasonable complexity. This is because (a) medium to high minimum distances can be achieved with low multiplicities, and (b) the constituent decoders can exchange soft information in an iterative manner which allows practical decoding that comes close to ML performance. Berrou *et al.* [21, 22] showed that a rate 1/2 turbo code[3] operating in AWGN can achieve a BER of 10^{-5} at an SNR of $\frac{E_b}{N_0} = 0.7$ dB. The capacity limit is $\frac{E_b}{N_0} = 0.19$ dB for the binary-input AWGN channel considered. Thus, it was shown that turbo codes can come within 0.5 dB of the binary-input capacity limit, at least for a BER of 10^{-5}. In other words, Berrou *et al.* showed for the first time that a practical communications system that operates very close to the capacity limit was possible. More recently, it has been shown in [23] that the use of 8-state[4] serially concatenated convolutional codes (SCCC) [24] with an interleaver size of 10^6 and 300 iterations can achieve a BER of 10^{-5} at an SNR of $\frac{E_b}{N_0} = 0.27$ dB, which is within 0.1 dB of the capacity limit.

1.3 Motivation and Research Objective

Turbo codes have reasonable complexity and provide powerful error correcting capability for various block lengths and code rates. Their typical error performance can be divided in the three regions as shown in Fig. 1.4.

- At very low to low SNRs, the error performance is very poor and is certainly not suitable for most communication systems.

[3]Two 16-state constituent encoders with (feedback polynomial, feedforward polynomial)=$(37, 21)_8$ and an interleaver size of 256x256=65536 were used. QPSK and 18 iterations using a modified BCJR (Bahl, Cocke, Jelinek and Raviv) algorithm were used.

[4](feedback polynomial, feedforward polynomial)=$(17, 07)_8$

- At low to medium SNRs, the error performance curve descends steeply, providing very low error rates at moderate SNRs. The region associated with the start and end of the steep error performance curve is known as *waterfall* region. In this region, the performance is mainly determined by the convergence behavior of the decoder. Generally, the longer the interleaver, the better the convergence, and the better (steeper) the waterfall performance is. This is mainly because of the nature of iterative decoding and has little to do with the potential improvement in distance properties for longer interleavers.

- At medium SNRs, the error performance curve starts to flatten severely; further improvements in the error performance require a significant increase in the SNR. The region associated with this severe flattening of the error performance curve is known as *error flare* or *error floor* region. The better the distance properties, the lower the error flare is. In other words, high minimum distances with low multiplicities are important for lowering the error flare.

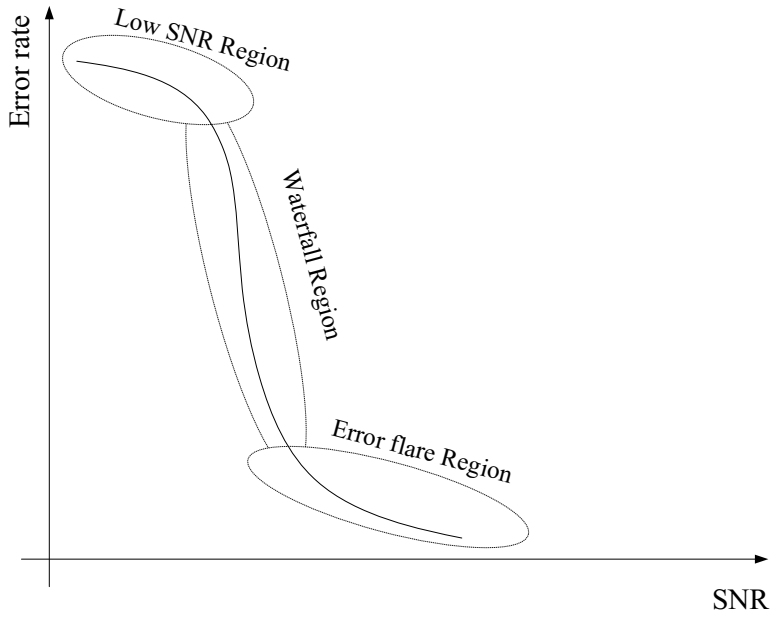


Fig. 1.4 Region of turbo code performance. Error rates are in base 10 logarithmic scale and SNRs (dB) are in linear scale.

Turbo code complexity and error performance depend mainly on the following seven factors. For details on turbo encoding and decoding, see Chapter 2.

- Constituent encoder constraint length: The complexity and memory requirements increase as the constraint length increases.

- Constituent encoder polynomials: Generally, the selection of the appropriate polynomials depends on the SNR value. In other words, constituent encoder polynomials affect both convergence (waterfall) performance and flare performance.

- Constituent decoder algorithm: The choice of the constituent decoder algorithm affects both complexity and convergence (waterfall) performance.

- Data puncturing: The larger the number of bits punctured, the higher the code rate is, but also the higher the SNR required to achieve the error performance of the un-punctured code.

- Number of iterations: Generally, higher number of iterations results in better waterfall performance, but it also leads to higher complexity and latency (delay).

- Interleaver size: Generally, longer interleavers improve the convergence and lower the error flare, but they also lead to higher memory requirement and latency.

- Interleaver design: Well-designed interleavers provide both good convergence and lower error flare without any increase in complexity. The memory required to store interleaver indices can also be reduced significantly if interleaver is structured.

From this discussion, it follows that error performance can be improved without any increase in complexity, memory requirement or latency if the interleaver is designed properly (assuming the use of appropriate constituent encoder polynomials). Thus, it is important to have a proper interleaver design in place that yields good distance properties.

Interleavers that yield high minimum distances can achieve very low error rates at moderate SNRs. Unfortunately, reliable software simulations that determine very low error rates may take months or not be applicable at all. However, experience indicates that the first term or first few terms of the distance spectrum can be used in the union bound to approximate those very low error rates. This is of particular use for the design of turbo code interleavers. Frequently the problem of finding good turbo code interleavers reduces to determining which of a class of interleavers gives the best performance. This can be predicted by finding which of the interleavers produces the highest first or first terms of

distance spectrum. In order to determine these terms, it is necessary to have a distance measurement method. It is also important that the method have low computational complexity that allows the computation of high distances in a reasonable time.

This thesis aims to develop efficient and accurate distance measurement methods for turbo codes. The new method developed in this thesis is applied to single-binary turbo codes included in the universal mobile telecommunications system (UMTS) standard [25] (also known as third generation partnership project 3GPP) as well as the double-binary turbo codes included in the digital video broadcasting with return channel via satellite (DVB-RCS) standard [15]. Another aim of this thesis is the design of new interleavers that yield significant improvement in error performance compared to the DVB-RCS standard interleavers. To achieve this aim, the dithered relative prime (DRP) approach [26] has been chosen as the basis for the new design. DRP interleavers are structured interleavers that are memory efficient because the interleaver can be stored and generated "on the fly" using only a few parameters instead of storing all the indices of the interleaver. The structure of DRP interleavers can also be exploited to reduce the complexity in computing the distance spectrum.

1.4 Research Contributions

The main contributions of this thesis can be summarized as follows:

- Garello's true distance measurement method [27] is extended to dual-terminated and tail-biting turbo codes. The extension is applied to both single-binary turbo codes and DVB-RCS double-binary turbo codes (Chapter 4 and [28]). This new method has lower complexity than the original one.

- A complete list of the first few terms of the distance spectra for all DVB-RCS standard block sizes and code rates is determined using this new method (Chapter 4 and [28]). These results complete the partial results in [29]. The results presented in [28] are the first complete results.

- New interleavers based on the DRP approach are designed for all DVB-RCS standard block sizes and code rates. Various interleavers are designed for motion picture experts group (MPEG) packets of length 1504 information bits. These interleavers are the

best ones known and provide a significant improvement in performance compared to the DVB-RCS standard interleavers (Chapter 4 and [30]).

- The reliability of Crozier's double-impulse iterative decoding method [31, 32] used for distance measurement is improved. The improved Crozier's method is compared with Garello's true distance method [27], Berrou's error-impulse method [33] and Garello's all-zero iterative decoding method [34] (Chapter 4 and [30]).

- A new efficient and accurate distance measurement method for tail-biting turbo codes that use structured interleavers such as DRP interleavers and the DVB-RCS standard interleavers was developed. The significant reduction in complexity achieved with this method was demonstrated by testing the best known interleavers for UMTS and DVB-RCS. This method allows the determination of high minimum distances in reasonable time (Chapter 5 and [35]). This is the fastest known distance measurement method for tail-biting turbo codes that use structured interleavers [35]).

- An efficient DVB-RCS turbo decoder has been implemented. This turbo decoder has been used to investigate various techniques that improve the error performance and reduce the decoding complexity. Furthermore, it has been used to investigate the capability of iterative decoding to provide reliable estimate for minimum distance.

- Efficient decoding techniques that improve error performance without an increase in complexity are applied to the DVB-RCS standard turbo codes. These techniques improve upon the known log maximum *a posteriori* (MAP) decoding, max-log MAP decoding and are referred to as enhanced log MAP and enhanced max-log MAP decoding, respectively, (Chapter 3 and [36]).

- An efficient early stopping rule that reduces the average complexity of decoding without degradation in error performance is applied to DVB-RCS turbo codes. The reduction in computational complexity is significant, especially for simulation purposes because the reliable determination of low error rates is very computational intensive (Chapter 3 and [36]).

- For tail-biting turbo codes, the turbo decoder does not have any knowledge about the start and end states of either encoders. To get reliable estimate for the probability of start and end state of each constituent encoder, a pre-run over a number of trellis

sections is required. This number of trellis sections is known as overlap size. The relation between overlap size and (a) packet length, (b) code rate and (c) SNR value is investigated using simulation (Chapter 3).

1.5 Overview of Thesis Structure

In Chapter 2, the building blocks of turbo codes, namely, constituent encoders, interleavers and puncturing units are discussed. The MAP decoding algorithm and iterative decoding are also discussed. To show the importance of soft-decision decoding, it is compared to hard-decision decoding for antipodal signalling (i.e., BPSK and Gray labelled QPSK) over an AWGN channel.

Chapter 3 introduces efficient decoding techniques for DVB-RCS double-binary turbo codes. The effect of tail-biting overlap size on performance is investigated using simulation results. An efficient early stopping technique is also applied to DVB-RCS.

Chapter 4 presents Garello's true distance measurement method for turbo codes. This method is extended to dual-terminated and tail-biting turbo codes. A simple technique that reduces the computational complexity is introduced. Distance results for selected UMTS block sizes and all DVB-RCS standard block sizes and code rates are presented. Distance results are also presented for newly designed DRP interleavers for all DVB-RCS standard block sizes and code rates. Error performances are compared for asynchronous transfer mode (ATM) packet sizes (424 information bits) and MPEG packet sizes (1504 information bits) using DVB-RCS standard interleavers as well as the newly designed DRP interleavers. Furthermore, this chapter compares distances obtained with the improved Crozier's double-impulse method to those obtained with Garello's true distance method, Berrou's error-impulse method and Garello's all-zero iterative decoding method.

Chapter 5 introduces an efficient and accurate distance measurement method for tail-biting turbo codes that use structured interleavers such as DRP and DVB-RCS standard interleavers. Distance results and execution times for selected well-designed interleavers for UMTS and DVB-RCS are presented.

Finally, Chapter 6 summarizes the contributions of this work, draws conclusions and discusses topics for future research.

Chapter 2

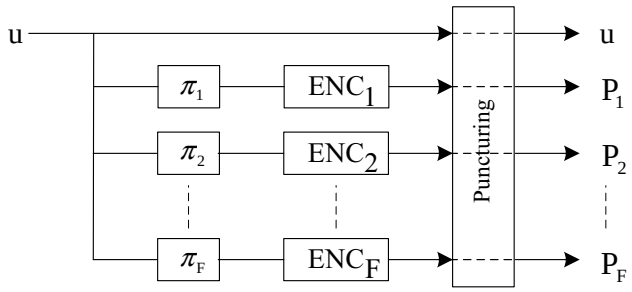
Turbo Codes

It is well known from information theory that a random code of sufficient length is capable of approaching the "Shannon limit", provided one uses *maximum likelihood* (ML) decoding. Unfortunately, the complexity of ML decoding increases with the size of codeword up to the point where decoding becomes impractical. Thus, a practical decoding of long codes requires that the code possess some structure. Coding theorists have been trying to develop codes that combine two 'seemingly' conflicting principles: (a) randomness, to achieve high coding gain and so approach the Shannon limit, and (b) structure to make decoding practical. In 1993, Berrou *et al.* introduced a new coding scheme that combines these two seemingly conflicting principles in an elegant way. They introduced randomness through an interleaver and structure by employing parallel concatenated convolutional codes. These codes are called turbo codes and offer an excellent tradeoff between complexity and error correcting capability. Concatenated codes are very powerful error correcting codes that are capable of closely approaching the Shannon limit by using iterative decoding [21, 23].

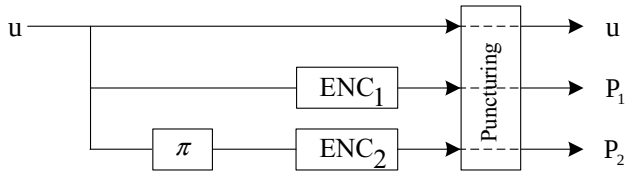
2.1 Turbo-Code Encoder Structure

A turbo-code encoder consists of three building blocks: constituent encoders, interleavers and a puncturing unit. The constituent encoders are used in parallel and each interleaver, π_i $(i = 1, \cdots, F)$, scrambles the information symbols before feeding them into the corresponding constituent encoder. The puncturing unit is used to achieve higher code rates. In general, turbo codes can have more than two parallel constituent convolutional encoders

ENC_i $(i = 1, \cdots, F)$, where each encoder is fed with a scrambled version of the information symbols $\boldsymbol{u}$. Fig. 2.1(a) shows the general structure of turbo codes, where the outputs $\boldsymbol{u}$, $\boldsymbol{P}_i$ $(i = 1, \cdots, F)$ are known as the systematic part and the parity part, respectively. In practice, most applications use only two constituent encoders where only the input to the second encoder is scrambled (see Fig. 2.1(b)).



(a) General structure of turbo codes.



(b) Typical structure of turbo codes.

Fig. 2.1 Turbo-code encoder structure.

2.1.1 The Constituent Encoders

A *non-recursive* or *feed-forward* convolutional encoder can be driven back to the all-zero state by δ zero symbols[1], where δ is the number of delay elements in the encoder. This means that the encoder starts and ends in the all-zero state for an input sequence $\boldsymbol{u} = (0, \cdots, 0, 1, 0, \cdots, 0)$ of weight 1 leading to a codeword of very low weight. This causes both the input sequence $\boldsymbol{u}$ and its permuted version to generate an output with very low weight. Since code with low weight codewords (i.e., low minimum distance) has poor error correcting capability, the use of feed-forward convolutional encoders in turbo codes is not desirable.

[1]It is common that at each time instant a single-binary symbol (i.e., 0 or 1) enters the convolution encoder. However, the input symbol need not be a single-binary symbol. For example, at each time instant a double-binary symbol (i.e., 00,01,10 or 11) enters the convolution encoders used in the DVB-RCS standard turbo codes [15].

Turbo codes use *recursive systematic convolutional* (RSC) encoders. The use of *recursive* or *feed-back* encoders prevents the encoders from being driven back to the all-zero state by zero symbols. Thus, assuming the RSC codes are terminated in the all-zero state, the input sequence $\boldsymbol{u}$ must have at least two non-zero symbols. Since $\boldsymbol{u}$ is permuted before entering ENC2, it is likely that one of the RSC code outputs will have high weight. This discussion does not mean that turbo codes exhibit very high minimum distances. In fact, achieving high minimum distances requires the use of a well designed interleaver of sufficient length. Finding such an interleaver is not trivial. The systematic part helps the iterative decoding to provide better convergence. Note that the systematic part prevents the tubo codes from being catastrophic[2] if no data puncturing is involved. If the systematic part is punctured, two different input sequences can produce the same codeword making the codes catastrophic [37]. Since repetition codes are not good codes, the systematic part from only one of the constituent encoders is transmitted.

In information theory and coding theory, convolutional codes are often expressed in terms of a set of polynomial generators that depend on the delay operator D, where D^n indicates a delay of n symbol times. A feed-forward convolutional encoder can be transformed easily to an RSC encoder with the same distance properties by dividing all polynomial generators by one of the polynomials. For example, a convolutional encoder with two feed-forward generator polynomials $g_1(D)$ and $g_2(D)$ is the 'same' as the RSC encoder with transfer function $G(D) = (1, g_2(D)/g_1(D))$, where $g_2(D)$ is the feed-forward generator polynomial and $g_1(D)$ is the feed-back generator polynomial. The 'same' means only that the distance properties are unchanged, the mapping of input sequence to output sequence is different. An example of an RSC encoder is the constituent encoder used in the universal mobile telecommunications system (UMTS) [25] standard turbo codes (see Fig. 2.2). This code has the transfer function $G(D) = (1, (1 + D + D^3)/(1 + D^2 + D^3))$. It is convenient and common to express the polynomial generators in octal representation. Thus, the feed-back polynomial $g_1(D) = 1 + D^2 + D^3$ is $(1011) = (13)_8$ and the feed-forward polynomial $g_2(D) = 1 + D + D^3$ is $(1101) = (15)_8$. It is also convenient and common to describe the functionality of a convolutional encoder using the *state-transition* diagram or the *trellis* diagram [38].

Due to the presence of the interleaver, good convolutional encoders are not necessarily good constituent encoders. The selection of constituent encoders from a distance spectrum

[2] A catastrophic code occurs when two or more information sequences are mapped to the same codeword.

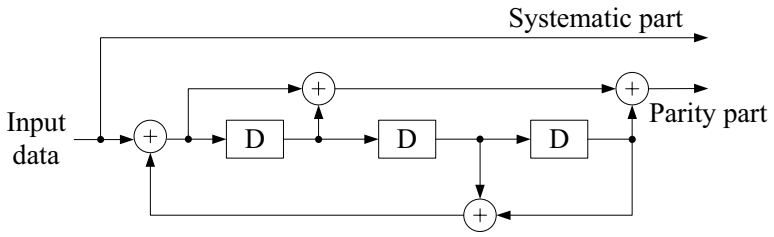


Fig. 2.2 8-state constituent encoder used in UMTS turbo codes

perspective and a decoding perspective is thoroughly discussed in [39, 40] and [41, 42], respectively. An important parameter for the design of constituent encoders is the period L of the feedback polynomial. It is common to use a primitive feedback polynomial because they provide the largest possible L for a given number of states δ (i.e., $L = 2^{\delta} - 1$). However, this does not suggest the use of high δ values such as 5, 6 or higher. A high δ value yields high minimum distance and so lowers the error flare, but also significantly increases both complexity and the memory requirements. A moderate δ value typically provides better convergence than a high δ value. The use of a moderate δ value (i.e., $\delta = 3$) is practical and provides good error performance at low SNR as well as high minimum distance for good error performance at high SNR, provided that the interleaver is designed properly.

2.1.2 Interleaving

Interleaving refers to the process of permuting symbols in the information sequence before it is fed to the second constituent encoder. The primary function of the interleaver is the creation of a code with good distance properties. Note that interleaving alone can not achieve good distance properties unless it is used together with recursive constituent encoders. The interleaver can be represented in many ways. In this thesis, the interleaver is a vector π, where $\pi(i)$ is the position in the information sequence that is interleaved to position i. In other words, the information sequence $\boldsymbol{u} = (u_0, u_1, \cdots, u_{K-1})$ is interleaved to $\boldsymbol{u}_\pi = (u_{\pi(0)}, u_{\pi(1)}, \cdots, u_{\pi(K-1)})$, where K is the number of symbols in the information sequence $\boldsymbol{u}$. This also means that the symbol at position i in $\boldsymbol{u}$ is interleaved to position $\pi^{-1}(i)$ in $\boldsymbol{u}_\pi$, where π^{-1} is the de-interleaver that acts on the interleaved information sequence and restores the sequence to its original order. There is a growing body of literature on the design of interleavers. The approaches used in the literature vary from simple random interleaving to highly structured interleaving.

Random interleavers are generated by selecting the elements (i.e., indices) of the interleaver in a random manner without any restriction on the selected elements. Usually, they yield codewords with low weight caused by input sequences of weight 2. In the literature they are sometimes called pseudorandom interleavers and they provide useful benchmarks. Another type of interleaver are the "spread" interleavers, known as S-random interleavers [43, 44]. These interleavers spread low weight input patterns to generate higher weight codewords. Thus, they can achieve better performance than average random interleavers. The elements of the S-random interleaver are selected based on the criterion that each element is not within $\pm S$ of any of the previously selected S elements. The interleaver can be generated as follows. Select a random integer from the set $\{0, \cdots, K-1\}$ as the first element in the interleaver and delete it from the set. Each subsequent element in the interleaver is selected randomly from the remaining integers in the set and compared with the S previously selected elements. If the selected element is within $\pm S$, then it is rejected and a new element is selected until the selection criterion is satisfied. Repeat this process until all K integers are selected. Note that backtracking might be required for a successful completion of the process. Unfortunately, the search time increases with the desired amount of separation, S, and the interleaver length K. Another drawback is that there is no guarantee that the search process will finish successfully (experiments have shown that the choice $S < \sqrt{N/2}$ usually produces a solution in a reasonable amount of time [45]). The S-random interleaver approach has been modified in [46, 47] by adding another criterion that considers the constituent encoders. The goal of this additional criterion is to eliminate low-weight codewords with significant contributions to the error performance. The elimination of a specific codeword can be done by breaking up the input pattern that generates that codeword. This additional criterion, however, increases the search time of the algorithm and is not guaranteed to eliminate all low-weight codewords.

A simple form of structured interleaver is the rectangular interleaver, often referred to in the literature as block interleaver. It is defined by a matrix of τ_r rows and τ_c columns such that the interleaver size is $K = \tau_r \times \tau_c$. This interleaving is performed as follows. Write the data to be interleaved into the matrix row by row, and read the interleaved data column by column, or vice versa. Rectangular interleavers that write in row by row and read column by column from the upper left corner can be expressed as $\pi(i) = i\tau_c + \lfloor i/\tau_r \rfloor \bmod K$, where $\lfloor x \rfloor$ is the "floor" function[3] of x. Such an interleaver, usually, yields a large number

[3] floor(x) is the largest integer less than or equal to x.

of low distance codewords caused by input sequences of weight 4 [45]. This is a direct consequence of the nature of the rectangular interleaver. Thus, increasing the size of the interleaver does not improve the distance properties and so does not improve the error performance. Another type of structured interleaver is the circular-shifting interleaver [45]. The permutation is described by $\pi(i) = (pi + s) \bmod K$, where s and p are known as offset and step size, respectively. The value of p is chosen relatively prime[4] to K to ensure that each element in the interleaver differs from the others. Crozier *et al.* in [48] refer to this type of interleaver as a relative prime (RP) interleaver. This interleaver can permute weight-2 input sequences that generate low distance into weight-2 input sequences that generate high distance. However, this type of interleaver is less likely to permute an input sequence of weight higher than 2 with low codeword weights into another input sequence with high codeword weights [45]. Another type of interleavers that have algebraic structure was introduced in [49]. This new algebraic interleaver design permutes a sequence of symbols with nearly the same statistical distribution as a random interleaver and perform as well as or better than the average of a set of random interleavers [49].

Interleaver design based on minimizing a cost function that takes into account the most significant error patterns for the desired interleaver was proposed in [50]. The interleaver grows iteratively to the desired length, with the cost function being minimized at each iteration step. The complexity of this interleaver design methodology is high and is suitable only for short interleavers. A short interleaver that yields a coding gain of 0.5 dB compared to a random interleaver at a bit error rate (BER) of 10^{-5} was designed in [50]. Interleaver design that is optimum in the sense of breaking up the weight-2 input sequences was introduced in [51]. In [51], a short interleaver based on this design rule was designed and simulated. The results show a significant improvement in performance compared to a random interleaver [51]. It is also noted in [51] that breaking up only the weight-2 input sequences is not sufficient to achieve good distance properties. This is because input sequences of weight higher than 2 are not broken up and can still lead to low codeword weights.

Achieving good distance properties is a common criterion for interleaver design. This fits very well with the concept of maximum likelihood (ML) decoding. Unfortunately, turbo decoding is not guaranteed to perform a ML decoding, because of the independence assumption made on the sequence to be decoded and the probabilistic information (known as

[4]Two positive integers are relatively prime if their greatest common divisor is 1.

extrinsic information[5]) passed between constituent decoders. This suggests an additional design criterion based on the correlation between the extrinsic information. An interleaver with good correlation properties (i.e., minimizing correlation between the extrinsic information of constituent decoders) was designed in [52]. Simulation results show that such interleavers perform better than S-random interleavers (improvement of about 0.1 dB).

It was recognized, shortly after the introduction of turbo codes, that spreading the elements of the interleaver results in both fast convergence and good distance properties. There are numerous structured interleaver approaches that have high-spread properties. Examples for such interleaver are: dithered golden interleavers [48], high-spread random (HSR) and dithered-diagonal (DD) interleavers [53] and dithered-relative prime (DRP) interleavers [26]. These interleavers maximize the overall (minimum) spread, defined below [26], and provide good error performance. The spread between two elements i and j in an interleaver is defined as:

$$S''_{new}(i,j) = |\pi(i) - \pi(j)| + |i - j|. \tag{2.1}$$

The minimum spread associated with index i is then

$$S'_{new}(i) = \min_{j, j \neq i} \; S''_{new}(i,j). \tag{2.2}$$

The overall (minimum) spread is defined as:

$$S_{new} = \min_{i} \; S'_{new}(i). \tag{2.3}$$

The following discussion concerns the DRP design approach. Let $\lfloor x \rfloor$ be the largest integer $\leq x$ and $[x]_K$ be the x modulo K operation. The DRP design approach is shown in Fig. 2.3 and consists of three steps [26]:

- First, the input vector, $\boldsymbol{v}_{in}$, is permuted locally using a small read dither vector, $\boldsymbol{r}$, of length R that is typically 2, 4, 8 or 16. The interleaved version $\boldsymbol{v}_a$ can be expressed

[5]Extrinsic information is discussed in detail in Section 2.3.

as

$$\begin{aligned} &\boldsymbol{v}_a(i) = \boldsymbol{v}_{in}(\pi_a(i)), \\ &\text{where} \\ &\pi_a(i) = R\lfloor i/R \rfloor + r\left([i]_R\right), \qquad i = 0, \cdots, K-1. \end{aligned} \tag{2.4}$$

- Next, the resulting vector, $\boldsymbol{v}_a$, is permuted using a relative prime (RP) interleaver, with starting index s and relative prime increment p, to obtain good spread. The interleaved version $\boldsymbol{v}_b$ can be expressed as

$$\begin{aligned} &\boldsymbol{v}_b(i) = \boldsymbol{v}_a(\pi_b(i)), \\ &\text{where} \\ &\pi_b(i) = [pi + s]_K \qquad i = 0, \cdots, K-1. \end{aligned} \tag{2.5}$$

- Finally, the resulting vector, $\boldsymbol{v}_b$, is permuted locally using a small write dither vector, $\boldsymbol{w}$, of length W. The interleaved version is the output vector $\boldsymbol{v}_{out}$ and can be expressed as

$$\begin{aligned} &\boldsymbol{v}_{out}(i) = \boldsymbol{v}_b(\pi_c(i)), \\ &\text{where} \\ &\pi_c(i) = W\lfloor i/W \rfloor + w\left([i]_W\right), \qquad i = 0, \cdots, K-1. \end{aligned} \tag{2.6}$$

Consequently, the output vector $\boldsymbol{v}_{out}$ can be expressed using the input vector $\boldsymbol{v}_{in}$ and the equivalent overall interleaver vector, π, referred to as the DRP interleaver.

$$\begin{aligned} &\boldsymbol{v}_{out}(i) = \boldsymbol{v}_{in}(\pi(i)), \\ &\text{where} \\ &\pi(i) = \pi_a(\pi_b(\pi_c(i))), \qquad i = 0, \cdots, K-1. \end{aligned} \tag{2.7}$$

Note that the interleaver length, K, must be a multiple of both R and W.

For any dither vectors $\boldsymbol{r}$ and $\boldsymbol{w}$, it can be shown that DRP interleavers have the following property:

$$\pi(i + M) = [\pi(i) + pM]_K \qquad i = 0, \cdots, K-1, \tag{2.8}$$

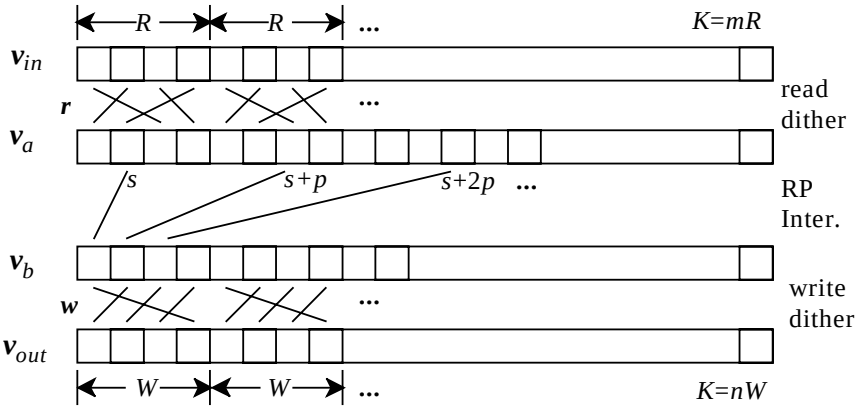


Fig. 2.3 Dithered relative prime (DRP) interleavers. Figure is taken from [26].

where M is the least common multiple of R and W, and the integer values p and K must be relative primes to ensure that the interleaver references all symbol elements. This nice property results in a significant reduction in memory requirement because only M repeating index increments need to be stored instead of the entire K indices. In other words, DRP interleavers are memory-efficient and can be generated on the fly using only M-index increments. In this thesis, the DRP approach is the basis for the new interleavers designed for single- and double-binary turbo codes. This is partly because its structure makes the search for good interleavers in the interleaver space relatively easy.

2.1.3 Trellis Termination

The truncation of *ordinary convolutional codes* without tail bits[6] results in significant degradation in error performance. This degradation occurs mainly for the bits near the unterminated end. For turbo codes the degradation in error performance is less. This is because turbo decoding uses constituent decoders that exchange *a priori* probabilities (extrinsic information) [54]. The higher the quality of the exchanged extrinsic information, the less is the degradation in error performance. This discussion does not conclude or suggest that trellis termination is not important for turbo codes. It just gives an intuitive explanation that the main purpose of trellis termination in turbo codes is not to combat the degradation in decoding performance that might occur near the end of the input sequence. Rather, the

[6]Tail bits are the extra bits needed to be encoded to drive the encoder to the all-zero state.

primary purpose is to avoid poor distance properties. This is demonstrated by the following example.

Assume a rectangular interleaver, where the data to be interleaved are written into the matrix row by row, and read in columns from the upper left corner. This interleaver interleaves the last symbol in the input sequence to the same position. If both trellises are not terminated, then the minimum distance is very low, namely 3^7, and is caused by the input sequence $\boldsymbol{u} = (0, \cdots, 0, 1)$. This is a simple example that demonstrates the importance of proper trellis termination from a distance spectrum perspective. This poor minimum distance can be improved by simply reading the columns from the lower right corner instead of from the upper left corner. A good interleaver must prevent the moving of symbols near the end of the input sequence to positions near the end of the interleaved input sequence. Even so, this will not yield good distance properties, unless the trellises are terminated properly. From this discussion, it follows that poor distance properties are most likely to occur if neither of the trellises is terminated.

Since turbo codes use constituent recursive convolutional encoders, the problem associated with terminating both trellis at the same time is rather complicated. This is because of the presence of the interleaver. There are various strategies of trellis termination. Some of them are discussed next.

Both encoders are terminated with individual tail symbols

It is well known that any recursive convolutional encoder of δ delays is guaranteed to be driven from any state s_x to any state s_y by a sequence of symbols, referred to as tail symbols, that are no longer than δ symbols. The tail symbols that drive the encoder from the state s_x to the all-zero state can be pre-computed for all possible 2^δ states and stored in a table. The first encoder (ENC1) and the second encoder (ENC2) start encoding in the all-zero state and are driven back to the all-zero state by δ_1 and δ_2 tail symbols, respectively. Here, δ_1 and δ_2 are the number of delays in ENC1 and ENC2, respectively. These δ_1 and δ_2 tail symbols are not included in the interleaver but are sent together with their parities to the decoder. This trellis termination has been adopted in the UMTS standard [25] and is referred to in this thesis as UMTS-termination. Compared to the case where none of the trellises are terminated, the minimum distance here is increased only by the extra weight

[7] Assuming the use of rate-1/2 constituent encoders.

resulting from δ_1 and δ_2 tail symbols and their corresponding parities. However, this type of termination still can produce low minimum distance because both trellises are terminated independently. Another drawback of this type of termination is the reduction in code rate, especially for short interleavers. Assuming the use of rate-1/2 convolutional encoders, the overall code rate is $R_c = K/(3K + 2\delta_1 + 2\delta_2)$.

Only first encoder is terminated

A common trellis termination method found in the literature is to terminate ENC1 and to leave ENC2 unterminated. ENC1 starts encoding in the all-zero state and is driven back to the all-zero state after encoding the K information symbols by δ_1 tail symbols. The δ_1 tail symbols are included in the sequence entering the interleaver, which implies that the interleaver length is $K + \delta_1$. The interleaved sequence is fed to ENC2 that starts in the all-zero state and is left unterminated in an unknown state after encoding the $K + \delta_1$ interleaved symbols. Terminating ENC1 reduce the likelihood of obtaining poor distance properties because the minimum distance is guaranteed to be caused by an input sequence of weight greater than or equal to 2. Assuming good spread, it is unlikely that both non-zero symbols in the un-interleaved input sequence are interleaved to positions near the end of the interleaved input sequence. For this reason, most small distances are eliminated. A drawback of this type of termination is the reduction in code rate, especially for short interleavers. Assuming the use of rate-1/2 convolutional encoders, the overall code rate is $R_c = K/(3(K + \delta_1))$.

Both encoders are terminated with a single set of termination symbols

Both encoders start and end encoding into the all-zero state by inserting $q = \delta_1 + \delta_2$ termination symbols to the K information symbols [55]. The resulting input sequence has $K + q$ symbols and consequently the interleaver length is $K + q$. It is shown in [55] that there is a q-by-q matrix that maps the q termination symbols to the final total state (the final state of both encoders). The first step in the encoding process is to choose the q termination positions within the $K+q$ positions. The encoding is performed in two stages. In the first stage, the K information symbols are placed in the non-termination positions and the q termination positions are filled with zero symbols. This input sequence of length $K+q$ is encoded by the turbo-code encoder that uses an interleaver of length $K+q$ and the

resulting total state is kept. This total state is multiplied with the inverse of q-by-q matrix to obtain the values of the termination symbols. In the second and final stage, the termination symbols are inserted into the input sequence and turbo encoding is performed to produce the final codeword. The q termination positions are independent of the information symbols, so they can be selected during the encoder design. Any q independent positions that span the termination space will work. Thus, there are many possible choices for the q termination positions. However, it is common to choose them close to the end of the non-interleaved $K+q$ symbols. This will reduce the extra complexity resulting from the second encoding [56]. This trellis termination is referred to in this thesis as dual-termination. This type of trellis termination provides good distance properties (assuming proper interleaver design) because both encoders are terminated and all encoded symbols are included in the interleaver. Again, this termination strategy results in a further reduction in code rate, especially for short interleavers. Assuming the use of rate-1/2 convolutional encoders, the overall code rate is $R_c = K/(3(K+\delta_1+\delta_2))$.

Both encoders are terminated using tail-biting

Circular coding makes it possible to start the encoding at a data dependent state, which is called the circular state $\boldsymbol{S}_c$, and end the encoding in the same state without any reduction in code rate. If the information sequence contains K symbols, where K is not a multiple of the period of the encoding recursive generator, then the existence of circular state $\boldsymbol{S}_c$ is guaranteed. Since the value of the circular state $\boldsymbol{S}_c$ depends on the sequence to be encoded, determining the circular state requires a pre-encoding operation. For the pre-encoding the encoder is initialized to the all-zero state. Then the data sequence goes through the encoder. At the end of the pre-encoding stage the encoder state is $\boldsymbol{S}_K^0$. The circular states $\boldsymbol{S}_c$ is then obtained from the expression $\boldsymbol{S}_c = \left(\boldsymbol{I} \oplus \boldsymbol{G}^K\right)^{-1} \cdot \boldsymbol{S}_K^0$ [57], where $\oplus$ denotes the bit-by-bit XOR operation. Here $\boldsymbol{I}$ and $\boldsymbol{G}$ are the identity matrix and the generator matrix, respectively. The circular states can be pre-computed and stored for all interleaver lengths of interest. For the case where K is a multiple of the period of the encoding recursive generator, alternative circular coding schemes presented in [56, 58] can be applied. This type of trellis termination has two advantages, namely, (a) no reduction in code rate and (b) yields good distance properties, provided that the interleaver is well designed. One disadvantage is that the decoder needs extra processing to estimate the

starting and ending states, which are unknown to it.

2.1.4 Puncturing

Puncturing refers to the process of removing certain bits from the codeword. The purpose of puncturing is to increase the overall code rate. It is common to puncture only the parity symbols of the first and second encoders. However, a significant improvement in the distance properties can be achieved if a small number of systematic symbols is punctured [37]. This is especially true for high code rates because the minimum distance is usually caused by low input weight. This means that the number of punctured systematic symbols can be increased without or with a small loss in the contribution of systematic part to the overall minimum distance. Increasing the number of punctured systematic symbols means that fewer parity symbols are punctured. This results in an improvement in the distance properties because the minimum distance is mainly determined by the parity contribution, especially for well designed interleavers.

In [59], a rate-1/2 turbo code that uses 4-state constituent encoders was designed considering the puncturing of the systematic part. The simulated results showed only a small improvement in flare performance. This is partly because only random interleavers were investigated. Results for 8 and 16-state turbo codes, for higher code rates with puncturing of the systematic symbols, were presented in [26, 60, 37]. In practice, the desirable code rate is usually achieved by puncturing the systematic part and the parities resulting from the first and second encoder using puncturing masks that repeat. However, care must be taken when the systematic part is punctured. This is because certain puncturing masks for the systematic part can lead to catastrophic puncturing, where different input sequences produce the same codeword [37]. Catastrophic puncturing can be systematically avoided, if highly structured interleaver, such as dithered relative prime (DRP) interleavers are used [37].

2.2 Turbo Decoding

As mentioned earlier, tubo codes are used to construct “large codes” that can be decoded with reasonable complexity using iterative decoding of the constituent codes that compose the turbo codes. The important element in turbo decoding are the constituent *soft-in soft-out* (SISO) decoders (Fig. 2.4). Each SISO takes as inputs:

- The output of the demodulator corresponding to the transmitted systematic part $\boldsymbol{y}^S$.
- The output of the demodulator corresponding to the transmitted parities associated with the SISO (i.e., $\boldsymbol{y}^{P1}$ for SISO1 and $\boldsymbol{y}^{P2}$ for SISO2).
- *A priori* information L_{apr}. That is, $L^t_{apr} = \ln \frac{p(u_t=1)}{p(u_t=0)}$ for single-binary turbo codes and $L^t_{apr} = \left\{ \ln \frac{p(u_t=01)}{p(u_t=00)}, \ln \frac{p(u_t=10)}{p(u_t=00)}, \ln \frac{p(u_t=11)}{p(u_t=00)} \right\}$ for double-binary turbo codes. Here $t = 0, \cdots, K-1$ where K is the size of the information sequence in symbols.

Let $\boldsymbol{y}_{\text{SISO}}$ be the part of received sequence corresponding to the SISO (i.e., $\boldsymbol{y}_{\text{SISO}} = \boldsymbol{y}^S, \boldsymbol{y}^{P1}$ for SISO1 and $\boldsymbol{y}_{\text{SISO}} = \boldsymbol{y}^S, \boldsymbol{y}^{P2}$ for SISO1). Each SISO is capable of producing two soft-outputs:

- A soft-output called extrinsic information, L_{ex}, suitable for exchange between constituent decoders.
- *A posteriori* information that is suitable for hard-decision. That is, $L^t_{apo} = \ln \frac{p(u_t=1|\boldsymbol{y}_{\text{SISO}})}{p(u_t=0|\boldsymbol{y}_{\text{SISO}})}$ for single-binary turbo codes and $L^t_{apo} = \left\{ \ln \frac{p(u_t=01|\boldsymbol{y}_{\text{SISO}})}{p(u_t=00|\boldsymbol{y}_{\text{SISO}})}, \ln \frac{p(u_t=10|\boldsymbol{y}_{\text{SISO}})}{p(u_t=00|\boldsymbol{y}_{\text{SISO}})}, \ln \frac{p(u_t=11|\boldsymbol{y}_{\text{SISO}})}{p(u_t=00|\boldsymbol{y}_{\text{SISO}})} \right\}$ for double-binary turbo codes. Again, here $t = 0, \cdots, K-1$ where K is the size of information sequence in symbols.

The SISO considered in this thesis is based on the maximum *a posteriori* (MAP) algorithm described in detail in Section 2.3. In the case where the codeword is punctured before transmission, the demodulator output corresponding to the punctured bits is set to zero.

The iterative decoding is performed as follows. The SISO1 takes as input the received systematic part $\boldsymbol{y}^S$, the parities $\boldsymbol{y}^{P1}$ belonging to ENC1 and the *a priori* information L_{apr1}. In the first iteration the L_{apr1} values are initialized with zeros, assuming that all symbols entering ENC1 are equally probable. SISO1 produces the extrinsic information L_{ex1} that is interleaved, using the same interleaver π used by the turbo-code encoder, and passed to SISO2. SISO2 takes as inputs the interleaved received systematic $\boldsymbol{y}^S_\pi$, the parities $\boldsymbol{y}^{P2}$ belonging to ENC2 and the extrinsic L_{ex1} provided by SISO1 as *a priori* information L_{apr2}. SISO2 produces the extrinsic information L_{ex2} that is de-interleaved, using the inverse interleaver π^{-1}, and passed to SISO1 as *a priori* information. This procedure is repeated iteratively. After a number of iterations, SISO2 produces *a posteriori* information L_{apo2}. The L_{apo2} values are de-interleaved, using π^{-1}, to match the order of the systematic part.

Then, for each time instant $t = 0, \cdots, K-1$ a hard-decision is performed based on the L^t_{apo2} values (Fig. 2.4). The next paragraph discusses the relation between the number of iterations and error rates. An iteration, also called a full-iteration, refers to two SISO decodings. A half-iteration refers to a single SISO decoding.

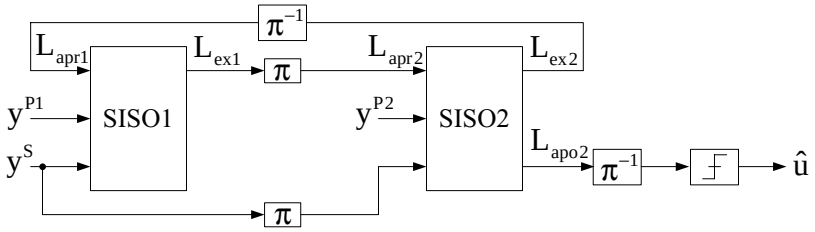


Fig. 2.4 Turbo decoder structure.

Turbo code error performance can be divided into three main regions: (a) low SNR region; (b) medium SNR region; and (c) high SNR region. In region (a), the performance is very poor and can be improved by increasing the number of iterations; however the overall error performance is certainly not suitable for most communication systems. In region (b), known as *waterfall* region, the error rates drops rapidly (see Fig. 1.4). In this region, many iterations are required to achieve good performance and increasing the number of iterations does improve the performance. Region (c), known as *error flare* or *error floor* region, is characterized by severe flattening of the error rate curves (see (Fig. 1.4). In this region the iterative decoder typically converges after a few iterations and further iterations usually lead to only a negligible improvement in performance. The error flare is mainly characterized by the minimum distance of the turbo code, which in turn is determined by the interleaver used (assuming that the constituent encoders and puncturing patterns have already been chosen).

2.3 MAP Algorithm

Bahl, Cocke, Jelinek and Raviv presented an optimal algorithm [61] for estimating the *a posteriori* probabilities of states and state transitions of a Markov source observed through a discrete memoryless channel. Since the codewords generated by a convolutional code can be viewed as the output of a Markov source, this algorithm is suitable for decoding convolutional codes. This algorithm, often referred to as the BCJR algorithm, was modified

by Berrou [21] to produce *a posteriori* probabilities for each information symbol. The modified algorithm, often referred to as the *maximum a posteriori* (MAP) algorithm, is optimal in the sense of minimizing the symbol error rate. In the turbo code literature, the MAP algorithm is also referred to as the *a posteriori* probability (APP) algorithm or the BCJR algorithm.

While the Viterbi algorithm [62, 63, 64] finds the most probable *information sequence*, or maximum likelihood (ML) sequence, that was sent, the MAP algorithm finds the most probable *information symbol*, at each time index, given the received coded sequence. The MAP algorithm is a *soft-in soft-out* (SISO) algorithm that is suitable for the iterative decoding used in turbo codes, while the original Viterbi algorithm is a hard-output algorithm (i.e., it does not produce *a posteriori* probabilities) and therefore it is not suitable for iterative decoding. The max-log MAP algorithm [65, 66] is a reduced complexity version of the MAP algorithm. There are two modifications to the Viterbi algorithm which produce soft-output Viterbi-algorithms SOVAs. One is Battail's SOVA [67] and the other is Hagenauer's SOVA [68]. The max-log MAP algorithm is about twice [65] as complex as the SOVA algorithm [68] and provides a good tradeoff between performance and complexity. The use of Hagenauer's SOVA algorithm leads to a performance degradation of a few tenths [65] of a decibel compared to the max-log MAP algorithm. A modified SOVA algorithm with complexity lower than that of the max-log MAP algorithm and a small degradation in error performance was presented in [69] and further discussed in [70].

The MAP algorithm is well described in the literature [21, 71, 72, 73, 74] for single-binary recursive systematic convolution (RSC) codes and binary antipodal signalling. This section discusses the MAP algorithm without putting any constraints on the type of RSC encoder (i.e., single-binary, double-binary,...) or type of M-ary modulation scheme.

Let $\boldsymbol{u} = (u_0, \cdots, u_{K-1})$ be the K input symbols to a RSC encoder. Let $s' = s_{t-1}$ be the encoder state at time $(t-1)$ and $s = s_t$ the encoder state at time t due to the information symbol u_t. Note that at each time instant t, the information symbol u_t and its corresponding coded bits v_t are fully determined by the pair (s', s). Let $\boldsymbol{y}$ be the received vector representing the transmitted coded sequence $\boldsymbol{u}$ and z be an element of the set ϕ^0 of all possible input symbols. The log likelihood ratio (LLR) for the APP of the information

symbol $u_t = z$, given the received $\boldsymbol{y}$, is expressed as [66]:

$$L^{(z)}(\hat{u}_t) \equiv \ln \frac{p(u_t = z \mid \boldsymbol{y})}{p(u_t = 0 \mid \boldsymbol{y})} = \ln \frac{\sum\limits_{(s' \to s, u_t = z)} p_t(s', s, \boldsymbol{y})}{\sum\limits_{(s' \to s, u_t = 0)} p_t(s', s, \boldsymbol{y})}, \tag{2.9}$$

where $p(u_t = z \mid \boldsymbol{y})$ is the APP of the information symbol z given $\boldsymbol{y}$. Note that $L^{(z=0)}(\hat{u}_t)$ is zero and therefore does not need to be calculated explicitly. The sum of the joint probabilities $p_t(s', s, \boldsymbol{y})$ in the numerator is taken over all transitions from state s' to state s caused by non-zero symbol z, whereas $p_t(s', s, \boldsymbol{y})$ in the denominator considers only the transitions caused by the zero symbol. These soft-output LLR are the basis for a hard-decision, $\hat{u}_t = \arg\max\limits_{z \in \phi^0} L^{(z)}(\hat{u}_t)$. If $\arg\max\limits_{z \in \phi^0} L^{(z)}(\hat{u}_t)$ returns more than one value, $\hat{u}_t$ can be set to any one of these values.

Assuming a transmission over a memoryless channel, the joint probability $p_t(s', s, \boldsymbol{y})$ can be expressed as the product of three independent probabilities [66]:

$$\begin{aligned} p_t(s', s, \boldsymbol{y}) &= p(s', y_{j<t}) \cdot p(s, y_t \mid s') \cdot p(y_{j>t} \mid s) \\ &= \underbrace{p(s', y_{j<t})} \cdot \underbrace{p(s \mid s') p(u_t = z \mid s', s) p(y_t \mid u_t = z)} \cdot \underbrace{p(y_{j>t} \mid s)} \\ &= \alpha_{t-1}(s') \quad \cdot \quad \gamma_t^{(z)}(s', s) \quad \cdot \quad \beta_t(s) \end{aligned} \tag{2.10}$$

where $y_{j<t}$ is the received sequence corresponding to the first $(t-1)$ trellis sections, $y_{j>t}$ is the received sequence corresponding to the last $(K-t)$ trellis sections and y_t is the received sequence corresponding to the trellis section at time t. The terms $\alpha_{t-1}(s') = p(s', y_{j<t})$ and $\beta_t(s) = p(y_{j>t} \mid s)$ are known as the *forward* and *backward* state probabilities, respectively, and $\gamma_t^{(z)}(s', s) = p(s \mid s') p(u_t = z \mid s', s) p(y_t \mid u_t = z)$ is the *transition* probability. Thus, equation (2.9) can also be expressed as [66]:

$$L^{(z)}(\hat{u}_t) = \ln \frac{\sum\limits_{(s' \to s, u_t = z)} \alpha_{t-1}(s') \cdot \gamma_t^{(z)}(s', s) \cdot \beta_t(s)}{\sum\limits_{(s' \to s, u_t = 0)} \alpha_{t-1}(s') \cdot \gamma_t^{(0)}(s', s) \cdot \beta_t(s)}. \tag{2.11}$$

The forward state probability $\alpha_t(s)$ represents the probability of being in the state s at time t, given the knowledge of the previous received sequence corresponding to the first t

trellis sections. It can be computed recursively in the following manner [61]:

$$\alpha_t(s) = \sum_{s'} \alpha_{t-1}(s') \cdot \gamma_t^{(z)}(s', s), \qquad t = 1, \cdots, K-1. \tag{2.12}$$

In other words, each new alpha at state s is the sum of the previous alphas multiplied by the transition probabilities corresponding to the transitions form s' to s. Assuming that the encoder starts in the zero state, the initial forward state probabilities are $\alpha_0(s' = 0) = 1$ and $\alpha_0(s' \neq 0) = 0$.

The backward state probability $\beta_{t-1}(s')$ represents the probability of being in the state s' at time $(t-1)$, given the knowledge of the future received sequence corresponding to the last $(K-t+1)$ trellis sections. It can be computed recursively backwards in the following manner [61]:

$$\beta_{t-1}(s') = \sum_{s} \beta_t(s) \cdot \gamma_t^{(z)}(s', s), \qquad t = K, \cdots, 2 \tag{2.13}$$

The values of betas are computed in the same way as those of alphas, but by starting at the end of the trellis and going in the reverse direction. Assuming a terminated trellis, the initial backward state probabilities are $\beta_K(s = 0) = 1$ and $\beta_K(s \neq 0) = 0$. If the trellis is not terminated, all $\beta_K(s)$ are assumed to be equally probable, implying that $\beta_K(s) = 1/\Delta$, where Δ is the number of states of the RSC encoder. The initialization of alphas and betas for tail-biting [56, 57, 58] is discussed in the next chapter, when DVB-RCS [15] standard double-binary turbo codes are introduced.

The transition probability is expressed as [61]

$$\gamma_t^{(z)}(s', s) = p(s \mid s') \cdot p(u_t = z \mid s', s) \cdot p(y_t \mid u_t = z). \tag{2.14}$$

If the transition $s' \rightarrow s$ is determined by the input symbol z', the probability $p(u_t = z \mid s', s)$ is either one or zero depending on whether $u_t = z'$ or $u_t \neq z'$. Assuming a trellis without parallel transitions, which is the case for convolutional codes, the probability $p(s \mid s')$ is the same as the *a priori* probability $p(u_t = z)$ if $p(u_t = z \mid s', s) = 1$ [65]. Thus,

$$\gamma_t^{(z)}(s', s) = p(u_t = z) \cdot p(y_t \mid u_t = z). \tag{2.15}$$

A turbo decoder uses constituent MAP decoders that exchange probabilities in an iterative manner, which improves the reliability of the *a posteriori* probabilities. Each MAP

decoder uses the same information symbols, also called the systematic symbols, to generate the *a posteriori* probabilities. Passing these probabilities between the constituent MAP decoders results in reuse of the probabilities associated with the systematic part over and over, which in turn affects the reliability of the *a posteriori* probabilities. Thus, the probabilities passed between the constituent decoder should be properly composed. In other words, only a specific part of the MAP decoder soft-output, $L^{(z)}(\hat{u}_t)$, should be exchanged between the constituent decoders. This specific part is known as *extrinsic* information. How to extract the extrinsic information from the $L^{(z)}(\hat{u}_t)$ is discussed next.

The information symbol $u_t = z$ enters the RSC encoder that produces the sequence $\boldsymbol{v}_t = (u_t, v_t)$, where u_t is the systematic part and v_t the parity part. Since there is a unique mapping from $u_t = z$ to $\boldsymbol{v}_t$, the probability $p(y_t \mid u_t = z)$ is the same as $p(y_t \mid \boldsymbol{v}_t)$. Let $\boldsymbol{y}(u_t)$ and $\boldsymbol{y}(v_t)$ be the parts of the received vector $\boldsymbol{y}$ that corresponds to the transmitted systematic part u_t and parity part v_t, respectively. The term $p(y_t \mid \boldsymbol{v}_t)$ can be expressed as the product of two probabilities: (a) the probability of the systematic part, $\gamma_t^{(z)}(u_t) = p(\boldsymbol{y}(u_t) \mid u_t)$, and (b) the probability of the parity part, $\gamma_t^{(z)}(v_t) = p(\boldsymbol{y}(v_t) \mid v_t)$. Using the abbreviation $\gamma_t^{(z)}(apr) = p(u_t = z)$, equation (2.15) can be written as:

$$\gamma_t^{(z)}(s', s) = \gamma_t^{(z)}(apr) \cdot \gamma_t^{(z)}(u_t) \cdot \gamma_t^{(z)}(v_t). \tag{2.16}$$

Inserting (2.16) in (2.11) results in:

$$L^{(z)}(\hat{u}_t) = \ln \frac{\sum\limits_{(s' \to s, u_t = z)} \alpha_{t-1}(s') \cdot \gamma_t^{(z)}(apr) \cdot \gamma_t^{(z)}(u_t) \cdot \gamma_t^{(z)}(v_t) \cdot \beta_t(s)}{\sum\limits_{(s' \to s, u_t = 0)} \alpha_{t-1}(s') \cdot \gamma_t^{(0)}(apr) \cdot \gamma_t^{(0)}(u_t) \cdot \gamma_t^{(0)}(v_t) \cdot \beta_t(s)}. \tag{2.17}$$

Note that $\gamma_t^{(z)}(apr)$ and $\gamma_t^{(z)}(u_t)$ in the numerator of (2.17) are the same for all trellis branches at time t. Similarly, $\gamma_t^{(0)}(apr)$ and $\gamma_t^{(0)}(v_t)$ in the denominator of (2.17) are the same for all trellis branches at time t. These terms can be taken out of the summations. This results in the following expression for the soft-output LLR:

$$\underbrace{L^{(z)}(\hat{u}_t)}_{A\ posteriori\ \text{LLR}} = \underbrace{\ln \frac{\gamma_t^{(z)}(apr)}{\gamma_t^{(0)}(apr)}}_{A\ priori\ \text{LLR}} + \underbrace{\ln \frac{\gamma_t^{(z)}(u_t)}{\gamma_t^{(0)}(u_t)}}_{Intrinsic\ \text{LLR}} + \underbrace{\ln \frac{\sum\limits_{(s' \to s, u_t = z)} \alpha_{t-1}(s') \cdot \gamma_t^{(z)}(v_t) \cdot \beta_t(s)}{\sum\limits_{(s' \to s, u_t = 0)} \alpha_{t-1}(s') \cdot \gamma_t^{(0)}(v_t) \cdot \beta_t(s)}}_{Extrinsic\ \text{LLR}}. \tag{2.18}$$

$$L^t_{apo} = L^t_{apr} + L^t_{in} + L^t_{ex}. \tag{2.19}$$

The *a posteriori* LLR, abbreviated L^t_{apo}, is composed of three LLRs. The first one, abbreviated L^t_{apr}, is the *a priori* LLR. The second one, abbreviated L^t_{in} and called *intrinsic* information, contains the contribution of the systematic part given y_t. The third one, abbreviated L^t_{ex} and called *extrinsic* information, provides the pure probabilistic contribution of the parity, and the rest of the systematic, symbols given both the encoder structure and the received sequence $\boldsymbol{y}$. When iterating, the extrinsic information from the previous MAP decoder becomes the *a priori* information for the current MAP decoder. That is,

$$L^t_{apo}(curr.) = L^t_{ex}(prev.) + L^t_{in}(curr.) + L^t_{ex}(curr.). \tag{2.20}$$

The calculation of the bit LLR for an arbitrary constellation is not the focus of this thesis. However, some pragmatic aspects, that are in line with common practice, are discussed. The next discussion focuses on how to evaluate the probabilities $\gamma^{(z)}_t(u_t)$ and $\gamma^{(z)}_t(v_t)$ for an arbitrary M-ary modulation.

In general, these probabilities (i.e.,$\gamma^{(z)}_t(u_t)$ and $\gamma^{(z)}_t(v_t)$) are evaluated using bit probabilities. The following example demonstrates why the evaluation is based on bit probabilities. Assume that u_t consists of two bits $u_t = (u^1_t, u^2_t)$ and that the modulation scheme is 8-PSK. Consider the transmission of the 3 symbols $u_t, u_{t+1}, u_{t+2} = (u^1_t, u^2_t, u^1_{t+1}, u^2_{t+1}, u^1_{t+2}, u^2_{t+2})$. Since the modulator maps every 3 bits to a single point in a two dimensional space (I- and Q-component), $(u^1_t, u^2_t, u^1_{t+1})$ and $(u^2_{t+1}, u^1_{t+2}, u^2_{t+2})$ are transmitted and received as $\overline{y}$ and $\overline{\overline{y}}$, respectively. The determination of $\gamma^{(z)}_t(u_t)$ and $\gamma^{(z)}_{t+2}(u_{t+2})$ involves only $\overline{y}$ and $\overline{\overline{y}}$, respectively. However, the determination of $\gamma^{(z)}_{t+1}(u_{t+1})$ is not obvious because u^1_{t+1} is transmitted in the received $\overline{y}$ and u^2_{t+1} is transmitted in the received $\overline{\overline{y}}$. A solution to this problem is to determine the bit probabilities and combine them to get the desired symbol probabilities. That is, $\gamma^{(z)}_{t+1}(u_{t+1}) = p(\boldsymbol{y}(u_{t+1}) \mid u_{t+1})$ is approximated by the product of $\gamma^{(z)}_{t+1}(u^1_{t+1}) = p(\boldsymbol{y}(u^1_{t+1}) = \overline{y} \mid u^1_{t+1})$ and $\gamma^{(z)}_{t+1}(u^2_{t+1}) = p(\boldsymbol{y}(u^2_{t+1}) = \overline{\overline{y}} \mid u^2_{t+1})$.

Let k and q be the number of bits in the information symbol u_t and parity v_t, respectively. The probabilities $\gamma^{(z)}_t(u_t)$ and $\gamma^{(z)}_t(v_t)$ are $\prod_{i=1}^{k} \gamma^{(z)}_t(u^i_t)$ and $\prod_{i=1}^{q} \gamma^{(z)}_t(v^i_t)$, respectively. This is strictly true only if the bits are associated with different received samples with independent noise on each. For example, with 64-QAM there are 6 bits involved but only 2 independent noise samples per symbol. Thus, these bit probabilities are not independent. In fact, the bits are often interleaved (channel interleaving) before mapping

them to constellation points, so that the bits are spread out amongst different constellation points. This way the bit probabilities can be considered to be essentially independent, so that they can be simply combined later.

For binary antipodal signalling such as BPSK and Gray labelled QPSK, the determination of the bit probabilities is straight forward because each coded bit c_t is mapped to the value $x_t = (1-2c_t)\sqrt{E} = (-1)^{c_t}\sqrt{E}$, where E is the energy of a modulated bit. The value x_t is corrupted by noise during the transmission and the received value is denoted by $\boldsymbol{y}(c_t)$. Assume that the noise is AWGN with variance $\sigma^2 = \frac{N_0}{2}$, N_0 being the one-sided power spectral density of the noisy channel. The probabilities $\gamma_t(c_t = 1)$ and $\gamma_t(c_t = 0)$ are given by

$$\begin{aligned} \gamma_t(c_t = 1) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{\boldsymbol{y}(c_t)+\sqrt{E}}{\sigma}\right)^2} \\ \gamma_t(c_t = 0) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{\boldsymbol{y}(c_t)-\sqrt{E}}{\sigma}\right)^2} \end{aligned} \tag{2.21}$$

It follows that the bit LLR is

$$\ln \frac{\gamma_t(c_t = 1)}{\gamma_t(c_t = 0)} = -4\frac{\sqrt{E}}{2\sigma^2}\boldsymbol{y}(c_t) = -4\frac{\sqrt{E}}{N_0}\boldsymbol{y}(c_t). \tag{2.22}$$

Note that BPSK and Gray labelled QPSK are widely used in practice, but they are not as bandwidth efficient as 64-QAM, for example.

For high order M-ary modulations, the estimation of bit probabilities from the received symbol increases the complexity and memory requirements, unless designed appropriately. If the I- and Q-component of the M-ary constellation cannot be mapped to two independent one-dimensional components, such as is the case for high order M-PSK (8-PSK, 16-PSK,$\cdots$, etc), all possible M signal points have to be used to obtain an estimate for each bit probability. Since this process has to be repeated for each received point, the complexity increases significantly. However, the complexity and memory requirements can be reduced for constellations made of two independent components such as 16-point quadrature amplitude modulation (QAM) and 64-point QAM, where they can be modelled as two independent 4-PAM and 8-PAM signals. To estimate the bit probability of 64-QAM, only 8 possible amplitudes have to be used, which means a reduction in complexity by a factor of $\sqrt{M}$. The following example shows how to estimate the bit probabilities of a 16-QAM

constellation.

The modulator maps four coded bits (c_1, c_2, c_3, c_4) into a single point and transmits it over the channel. The constellation points can be labelled in such a manner that (c_1, c_2) are mapped to an I-component and (c_3, c_4) are mapped to a Q-component of the 16-QAM constellation (Fig. 2.5). Since I- and Q-components are statistically independent, the bit probability can be estimated given the received point (y_I, y_Q) [75].

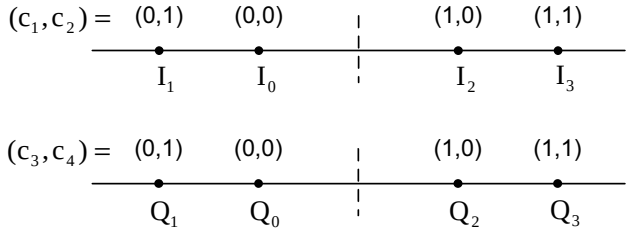


Fig. 2.5 Mapping the first two bits (c_1, c_2) and the last two bits (c_3, c_4) of a 16-QAM signal point to two independent 4-PAM signals.

$$\begin{aligned}
\gamma(c_1 = 1) &= p(y_I \mid I_2) + p(y_I \mid I_3) \\
\gamma(c_1 = 0) &= p(y_I \mid I_1) + p(y_I \mid I_0) \\
\gamma(c_2 = 1) &= p(y_I \mid I_1) + p(y_I \mid I_3) \\
\gamma(c_2 = 0) &= p(y_I \mid I_0) + p(y_I \mid I_2).
\end{aligned} \tag{2.23}$$

Similarly, $\gamma(c_3)$ and $\gamma(c_4)$ can be estimated for both values 0 and 1 using the y_Q component of the received point (y_I, y_Q).

Note that when using higher order modulations, different bits will typically have different average reliabilities. This leads to a nontrivial problem of how to best match the constellation mapping to information bits and parity bits. It has been observed that mapping the information bits to the most protected position yields the best convergence. Mapping the parity bits to the most protected position yields the best flare. In other words, there is a trade-off between convergence (waterfall) and flare performance.

Log MAP **algorithm**

The log MAP algorithm was introduced to reduce the complexity of the MAP algorithm. Using the logarithmic domain, the multiplications and divisions become additions and

subtractions, which results in a reduction in complexity. In addition, the use of logarithms tends to help with overflow and precision issues.

The log MAP algorithm is obtained by applying the natural logarithm to the equations for the transition probabilities, the alphas and the betas:

$$\overline{\gamma}_t^{(z)}(s', s) \equiv \ln \gamma_t^{(z)}(s', s) = \ln p(y_t \mid u_t = z) + \ln p(u_t = z) \tag{2.24}$$

$$\overline{\alpha}_t(s) \equiv \ln \alpha_t(s) = \ln \sum_{s'} e^{\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^{(z)}(s', s)} \tag{2.25}$$

$$\overline{\beta}_{t-1}(s') \equiv \ln \beta_{t-1}(s') = \ln \sum_{s} e^{\overline{\beta}_t(s) + \overline{\gamma}_t^{(z)}(s', s)}. \tag{2.26}$$

It follows that the soft-output LLR for the log MAP algorithm is:

$$L^{(z)}(\hat{u}_t) = \ln \frac{\sum\limits_{(s' \to s, u_t = z)} e^{\overline{\alpha}_{t-1}(s')} \cdot e^{\overline{\gamma}_t^{(z)}(s', s)} \cdot e^{\overline{\beta}_t(s)}}{\sum\limits_{(s' \to s, u_t = 0)} e^{\overline{\alpha}_{t-1}(s')} \cdot e^{\overline{\gamma}_t^{(0)}(s', s)} \cdot e^{\overline{\beta}_t(s)}}. \tag{2.27}$$

The problem associated with the evaluation of the sum of exponentials can be solved using the expression [76]:

$$\ln \left(e^{a_1} + e^{a_2}\right) = \max(a_1, a_2) + \ln \left(1 + e^{-|a_1 - a_2|}\right), \tag{2.28}$$

where the *correction term* is $\ln \left(1 + e^{-|a_1 - a_2|}\right)$ can be approximated by a small lookup table with a negligible affect on performance [74]. If the number of exponentials exceeds two, the sum of these exponentials can be obtained recursively as follows, where $f(x) = \ln \left(1 + e^{-|x|}\right)$

$$\begin{aligned} \ln \left(e^{a_1} + e^{a_2}\right) &\equiv \max^*(a_1, a_2) = \max(a_1, a_2) + f(a_1 - a_2) \\ \ln \left(e^{a_1} + e^{a_2} + e^{a_3}\right) &= \ln \left(e^{\max^*(a_1, a_2)} + e^{a_3}\right) = \max^*(\max^*(a_1, a_2), a_3) \\ &\textit{etc.} \end{aligned} \tag{2.29}$$

As shown in [65, 74], the true values of the correction term $\left(1 + e^{-|a_1 - a_2|}\right)$ can be approximated using pre-calculated values that are stored in a one-dimensional lookup table,

which results in reduced complexity. Simulation results [65] have shown that the use of a lookup table with only 8 values results in very little degradation compared to a true MAP algorithm.

Since the evaluation of the conditional probability $p(y_t \mid u_t = z)$ requires the knowledge of the noise variance σ^2, both the MAP and the log MAP algorithms require the knowledge of the channel's SNR. If the channel characteristics do not vary over time, it is sufficient to use the SNR of the design point, otherwise, good variance estimators are available in [77, 78]. However, accurate estimation of the SNR is not necessary. It has been shown that iterative decoding is more tolerant of an overestimation of the SNR than an underestimation [79, 80].

Max-Log MAP **algorithm**

The max-log MAP decoding algorithm [65, 66] is an approximation to the MAP decoding algorithm and is used in practice because of its reduced computational complexity. The reduction in complexity is due, in part, to the transformation of multiplications and divisions to additions and subtractions, but is mainly due to avoiding the computation of the exponential function by using the approximation

$$\ln \sum_j e^{a_j} \approx \max_j (a_j). \tag{2.30}$$

However, the approximation leads to a degradation in the error rate performance. In the next chapter a technique that reduces this degradation is introduced [36].

Applying the *max*-function gives the following expressions for $\overline{\gamma}$, $\overline{\alpha}$ and $\overline{\beta}$

$$\overline{\gamma}_t^{(z)}(s', s) = \ln p(y_t \mid u_t = z) + \ln p(u_t = z) \tag{2.31}$$

$$\overline{\alpha}_t(s) \approx \max_{s'} \left(\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^{(z)}(s', s) \right) \tag{2.32}$$

$$\overline{\beta}_{t-1}(s') \approx \max_{s} \left(\overline{\beta}_t(s) + \overline{\gamma}_t^{(z)}(s', s) \right) \tag{2.33}$$

and

$$L^{(z)}(\hat{u}_t) \approx \max_{(s' \to s, u_t = z)} \left(\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^{(z)}(s', s) + \overline{\beta}_t(s) \right) - \max_{(s' \to s, u_t = 0)} \left(\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^{(0)}(s', s) + \overline{\beta}_t(s) \right). \tag{2.34}$$

Note that the max-log MAP algorithm is basically the log MAP algorithm without the

correction term (or lookup table). As opposed to the MAP and log MAP algorithms, the max-log MAP algorithm does not require the knowledge of the SNR and is insensitive to scaling of the intrinsic LLRs.

2.4 Soft-decision and Hard-decision Decoding

This section discusses the importance of soft-decision decoding. This is done by comparing the SNR required with soft-decision decoding with that required for hard-decision decoding. Binary antipodal signalling over an AWGN channel and *maximum likelihood* (ML) decoding are assumed. BPSK and Gray labelled QPSK modulation are examples of binary antipodal signalling.

2.4.1 Coding Gain with Soft-decision Decoding

In soft-decision decoding the demodulator passes the sampled, match filtered analog values of the received vector $\boldsymbol{y}$ directly to the decoder. The ML-decoder chooses the message signal that is closest to the received vector $\boldsymbol{y}$ in the *Euclidean* distance sense.

Define $C(\widetilde{N}, \widetilde{K})$ as a linear code, where $\widetilde{N}$ is the codeword length in bits and $\widetilde{K}$ is the number of information bits. Let the vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ represent the transmitted signals $x_i(t)$ and $x_k(t)$, respectively, in an orthonormal system that spans the signal space. The message error probability is bounded by the union bound [81]:

$$P_w \leq \frac{1}{M} \sum_{i=0}^{M-1} \sum_{k=0, k\neq i}^{M-1} Q\left(\frac{d^E(\boldsymbol{x}_i, \boldsymbol{x}_k)}{\sqrt{2N_0}}\right), \tag{2.35}$$

where $M = 2^{\widetilde{K}}$ is the number of binary messages, $Q(x) = \int_x^\infty \frac{e^{-t^2/2}}{\sqrt{2\pi}} dt$ and $d^E(\boldsymbol{x}_i, \boldsymbol{x}_k) = \|\boldsymbol{x}_i - \boldsymbol{x}_k\|$ is the Euclidean distance between the vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$. For binary antipodal signalling, each bit b is mapped to $(1-2b)\sqrt{E} = (-1)^b\sqrt{E}$, where E is the energy used to transmit each modulated bit. Therefore, the Euclidian distance can be expressed as:

$$d^E(\boldsymbol{x}_i, \boldsymbol{x}_k) = 2\sqrt{E d^H(\boldsymbol{c}_i, \boldsymbol{c}_k)}, \tag{2.36}$$

where $d^H(\boldsymbol{c}_i, \boldsymbol{c}_k)$ is the Hamming distance (number of bits that are different) between the two codewords $\boldsymbol{c}_i$ and $\boldsymbol{c}_k$. Note that for a linear code, all codewords see the same

neighborhood, that is $d^H(\boldsymbol{c}_i, \boldsymbol{c}_k) = d^H(\boldsymbol{c}_0, \boldsymbol{c}_j = \boldsymbol{c}_i \oplus \boldsymbol{c}_k)$, where $\boldsymbol{c}_0$ is the all-zero codeword and $\oplus$ denotes the bit-by-bit XOR operation. Therefore, the message/codeword error probability is:

$$P_w \leq \sum_{j=1}^{M-1} Q\left(\frac{2\sqrt{E d^H(\boldsymbol{c}_0, \boldsymbol{c}_j)}}{\sqrt{2N_0}}\right). \tag{2.37}$$

The function $Q(x)$ asymptotically approaches $e^{-\frac{x^2}{2}}$ as x approaches ∞. Therefore, the codeword error probability is upper bounded by the sum of $(M-1)$ exponentials. Since each probability decays exponentially as the Hamming distance $d^H(\boldsymbol{c}_0, \boldsymbol{c}_j)$ grows, the minimum Hamming distance $d^H_{\min} = \min_{j \neq 0} d^H(c_0, c_j)$ dominates the sum. Therefore, the asymptotic codeword error probability is:

$$P_w \approx A_{\min} Q\left(\sqrt{2 d^H_{\min} \frac{E}{N_0}}\right), \tag{2.38}$$

where $A_{\min}$ is the number of codewords at $d^H_{\min}$. It follows that the asymptotic information bit error probability is:

$$P_b \approx \frac{W_{\min}}{\widetilde{K}} Q\left(\sqrt{2 d^H_{\min} \frac{E}{N_0}}\right), \tag{2.39}$$

where $W_{\min}$ is the sum of the Hamming weights of all input sequences causing $d^H_{\min}$. For a fixed $\widetilde{K}$ and large SNR, the term $W_{\min}/\widetilde{K}$ is negligible for evaluating performance as a function of asymptotic SNR and therefore the information bit probability can be approximated as

$$P_b \approx Q\left(\sqrt{2 d^H_{\min} \frac{E}{N_0}}\right). \tag{2.40}$$

If coding is employed, then $E = E^c_b = R_c \cdot E^u_b$, where E^c_b is the energy per coded bit, E^u_b is the energy per un-coded bit and R_c is the code rate. Using $e^{-\frac{x^2}{2}}$ for x approaching ∞, the asymptotic information bit error probability P_b can be expressed as

$$P_b \leq e^{-R_c d^H_{\min} \frac{E^u_b}{N_0}}, \text{as} \quad \frac{E^u_b}{N_0} \to \infty \tag{2.41}$$

The information bit error probability for un-coded case is given by [82]

$$P_b = Q\left(\sqrt{2\frac{E_b^u}{N_0}}\right). \tag{2.42}$$

Thus, the asymptotic information bit error probability for un-coded case is:

$$P_b \leq e^{-\frac{E_b^u}{N_0}}, \text{as} \qquad \frac{E_b^u}{N_0} \to \infty \tag{2.43}$$

From (2.41) and (2.43) it follows that the asymptotic information bit error probability achieved for the un-coded case at an $\text{SNR}^u = \frac{E_b^u}{N_0}$ can be achieved at an $\text{SNR}^c = \left(\frac{E_b^u}{N_0}\right)/(R_c d_{\min}^H)$. Therefore, the asymptotic coding gain with soft-decision decoding can be expressed as:

$$G^{\text{soft}} = 10\log_{10}\left(\text{SNR}^u\right) - 10\log_{10}\left(\text{SNR}^c\right) = 10\log_{10}\left(R_c d_{\min}^H\right) \qquad \text{dB}. \tag{2.44}$$

2.4.2 Coding Gain with Hard-decision Decoding

In hard-decision decoding, the demodulator makes a hard-decision on the components of the received vector $\boldsymbol{y}$, that is it decides whether bit 0 or bit 1 was transmitted, and passes these binary bits to the decoder. The ML-decoder chooses the codeword that minimizes the Hamming distance between the codeword and the binary sequence received from demodulator.

Let $C(\widetilde{N}, \widetilde{K}, t)$ be a code that is capable of correcting at most t-errors. Assuming that the bit errors are independent and that the channel is BSC with transition probability ϵ (bit error probability), the codeword error probability is upper bounded by [4]

$$P_w \leq \sum_{n=t+1}^{\widetilde{N}} \binom{\widetilde{N}}{n} \epsilon^n (1-\epsilon)^{\widetilde{N}-n}. \tag{2.45}$$

As the SNR approaches ∞, the transition probability ϵ approaches 0. Therefore, for a fixed $\widetilde{N}$ and t, the asymptotic codeword error probability is $P_w \approx \epsilon^{t+1}$. Without investigating how the information bit error probability P_b compares to the codeword error probability P_w, it is obvious that $P_b \leq P_w$ and can be approximated as $P_b \approx \epsilon^{t+1}$. For binary antipodal

signalling, the transition probability is given by [4]

$$\epsilon = Q\left(\sqrt{2\frac{E}{N_0}}\right). \tag{2.46}$$

If coding is employed ($E = E_b^c = R_c \cdot E_b^u$), the asymptotic information bit error probability can be approximated as:

$$P_b \approx e^{-(t+1)R_c\frac{E_b^u}{N_0}}, \text{as} \qquad \frac{E_b^u}{N_0} \to \infty \tag{2.47}$$

Using (2.42), the asymptotic bit error probability for the un-coded case can be approximated as:

$$P_b \approx e^{-\frac{E_b^u}{N_0}}, \text{as} \qquad \frac{E_b^u}{N_0} \to \infty \tag{2.48}$$

From (2.47) and (2.48) it follows that the asymptotic coding gain with hard-decision decoding is given by:

$$G^{\text{hard}} = 10\log_{10}\left(R_c(t+1)\right) \qquad \text{dB} \tag{2.49}$$

2.4.3 Soft-decision Decoding versus Hard-decision Decoding

Assuming binary antipodal signalling, AWGN channel and ML decoding it follows from (2.44) and (2.49) that the coding gain with soft-decision decoding over hard-decision decoding is:

$$G^{\text{soft vs. hard}} = 10\log_{10}\left(\frac{d_{\min}^H}{t+1}\right) \qquad \text{dB}. \tag{2.50}$$

If the code $C(\widetilde{N}, \widetilde{K}, t)$ is a linear code with minimum distance $d_{\min}^H$, then the hard-decision error-correcting capability of this code is guaranteed to be $t = \left\lfloor \frac{d_{\min}^H - 1}{2} \right\rfloor$ [83], where $\lfloor x \rfloor$ is the largest integer $\leq x$. Therefore, soft-decision decoding is asymptotically (SNR approaches ∞) about 3 dB more efficient that hard-decision decoding. At realistic SNRs a figure of 2 dB is more appropriate [2].

The above discussion demonstrates that the correction capability of a code can be improved significantly by applying soft-decision decoding. In practice, the soft output of the matched filter must be quantized. The finer the quantization, the higher the coding gain. In practice, 4 and 5 bits of quantization is usually sufficient for binary antipodal signalling

schemes [84]. Finer quantization are required for higher order modulation schemes, such as 16- or 64-QAM

Chapter 3

DVB-RCS Turbo Decoding

The broadband access market is dominated by Digital Subscriber Line (DSL) and cable modem technology, with increasing competition from terrestrial wireless Local Multipoint Distribution Service (LMDS) technologies, and emerging competition from satellite wireless technologies. The technical limitations of DSL (e.g., distance from the switching office), cable modem (e.g., shared line capacity), and LMDS (e.g., lack of line of sight to towers) have led to service degradation, which has resulted in slower customer acceptance.

Given rapid growth in the broadband access market and the limitations of existing technologies, satellite broadband access has emerged as an alternative technology with particular advantages. One is that a satellite can deploy service over a large area, once a single hub infrastructure is in place. Others are the low cost and the constant service quality for all user in the system. This is in contrast to a DSL system, where service quality depends on the distance to the switching office.

Digital video broadcasting with return channel via satellite (DVB-RCS) is an open standard that provides a satellite return channel to systems based on the digital video broadcasting (DVB) standard for satellite transmission. It is published and maintained by the European Telecommunications Standards Institute (ETSI) for digital video broadcasting [15]. DVB-RCS supports a wide range of applications such as web browsing, videoconferencing, teleworking and distance education. DVB-RCS systems enable two-way communications between a hub station and remote terminals, referred to as satellite interactive terminals (SITs). The communication path from the hub station towards the SITs is referred to as the forward link. The communication path from the SITs back to the hub

station is referred to as the return link. The hub station continuously transmits on the forward link using a time division multiplex (TDM) access technique. The SITs transmit as needed, sharing the return link resources using a multi-frequency time division multiple (MF-TDMA) access technique. On the forward link, packets are encapsulated in an MPEG DVB data-stream received by all SITs. On the return link, packets are encapsulated in ATM or MPEG data-steam to the hub. Typical data rates forDVB-RCS range from 1 to 10 Mbps on the forward link and 144 kbps to 2Mbps on the return link.

For deep-space and satellite communications where power savings are especially important, turbo codes allow for increased energy efficiency without compromising performance. The DVB-RCS return link uses 8-state double-binary turbo codes of various code rates (1/3 to 6/7) and block sizes (12 to 216 bytes) that include ATM and MPEG sizes. Compared to classical turbo codes, which use single-binary codes, double-binary turbo codes have a number of advantages:

- Double-binary turbo codes double the decoding rates in a hardware implementation, because they allow memory access of two bits at each time instant. The reason for such decoding rates is the fact that the extrinsic information, which must be passed to the next decoder after *interleaving* or *de-interleaving*, represents two bits at each time instant. Doubling the decoding rates leads to a reduction in the latency of the decoder by one half.

- Double-binary turbo codes reduce the sensitivity to puncturing [85]. This can be explained as follows. Since the rate 1/2 double-binary recursive systematic convolutional (RSC) encoder produces two parity streams, most of the code rates can be obtained by simply ignoring one of these parity streams and puncturing the other (if necessary). Ignoring one of the two parity streams results in a new RSC encoder with a single parity stream. This single parity stream is less punctured compared to similar single-binary convolutional RCS encoders, which results in less sensitivity to puncturing.

- Double-binary turbo codes reduce the correlation effects between component decoders, which leads to improved convergence [86].

For practical purposes, it is important to reduce the computational complexity of turbo decoding. An approach for reducing the computational complexity of maximum *a posteriori*

(MAP) decoding [61] has been introduced in [74] for single-binary turbo codes, where there are only two branches entering and leaving each state. The idea behind this approach is to avoid the computation of $\ln(e^a + e^b)$ by using the expression [76]

$$\ln\left(e^{a_1} + e^{a_2}\right) = \max(a_1, a_2) + \ln\left(1 + e^{-|(a_1 - a_2)|}\right), \tag{3.1}$$

where the *correction term* $\ln\left(1 + e^{-|a_1 - a_2|}\right)$ is approximated in a small lookup table with a negligible effect on performance.

As opposed to single-binary codes where only two branches enter and leave each state, in double-binary turbo codes there are four branches entering and leaving each state. As a result of the four branches, the following expression must be evaluated for double-binary codes

$$\ln \sum_{i=1}^{4} e^{a_i} = \max(a_1, a_2, a_3, a_4) + \ln \sum_{i=1}^{4} e^{a_i - \max(a_1, a_2, a_3, a_4)}. \tag{3.2}$$

This expression can be evaluated in recursive manner using (2.29) in Section 2.3. However, this has the following drawbacks:

- Due to the presence of the four values a_1, a_2, a_3, a_4 a much larger lookup table is required to guarantee a reliable approximation for $\ln \sum_{i=1}^{4} e^{a_i}$. This results in an increase in memory requirements compared to single-binary turbo codes.

- An increase in computational complexity as a result of the evaluation of the expression $\ln \sum_{i=1}^{4} e^{a_i}$ in a recursive way. This is because of (a) the extra computation implied by $(a_i - \max(a_1, a_2, a_3, a_4))$ where $i = 1, \cdots, 4$, and (b) the search for the correction term in the lookup table.

The discussion above shows that the use of true MAP decoding for double-binary turbo codes results in both extra computational complexity and extra memory requirements. It is clear that these drawbacks can be avoided by using max-log MAP decoding. However, the use of max-log MAP decoding results in a degradation in error performance compared to true MAP decoding. This chapter discusses techniques that improve the max-log MAP decoding performance. The key element here is the fact that these techniques do not increase the computational complexity of max-log MAP decoding and yet provide error performance very close to that of true MAP decoding. This techniques are applied to

DVB-RCS turbo codes for ATM and MPEG packet sizes, and simulation results are presented. A further reduction in computational complexity can be achieved by applying an effective *early stopping criterion*. An efficient early stopping criterion that does not affect the error performance is applied and simulation results are presented for DVB-RCS turbo codes. Finally, the relation between the error performance and overlap length needed for decoding of tail-biting turbo codes is investigated for various packet sizes and code rates.

3.1 DVB-RCS Encoding Scheme

3.1.1 Circular Coding

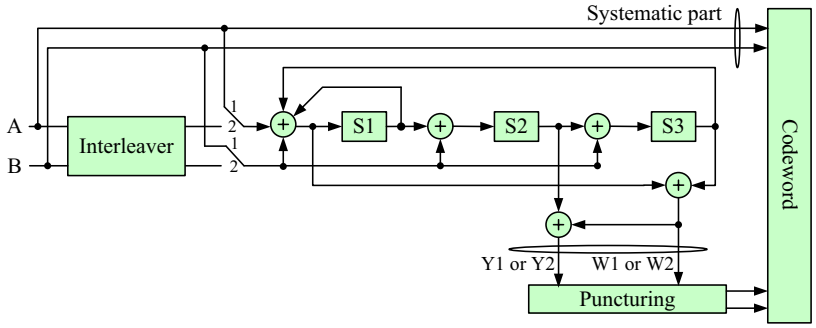
Due to block-oriented encoding, where the data stream is encoded block wise, the convolutional component codes are likely to end in unknown states at the end of the encoding process. Since the states of the encoders at the end of the encoding process are unknown to the constituent decoders, the bits at the end of the block are not as well protected. This results in a degradation in error rate performance if no precautions are taken. The most common method to overcome this problem is to force the encoder to a known state at the end of the encoding stage by the addition of termination bits, which are then sent to the decoder. This lowers the code rate and hence the data rate, especially for short blocks, resulting in a decrease in the spectral efficiency [87]. Circular coding makes it possible to start the encoding at a data dependent state, which is called the *circular state* $\boldsymbol{S}_c$, and end the encoding in the same state while avoiding this drawback.

To determine the circular state $\boldsymbol{S}_c$ for the DVB-RCS turbo-code encoder shown in Fig. 3.1(a), note that the state of the encoder at time t denoted by $\boldsymbol{S}_t$ depends on the previous state $\boldsymbol{S}_{t-1}$ and the input couple (A_t, B_t). At time t the contents of the registers of the encoder are [57]:

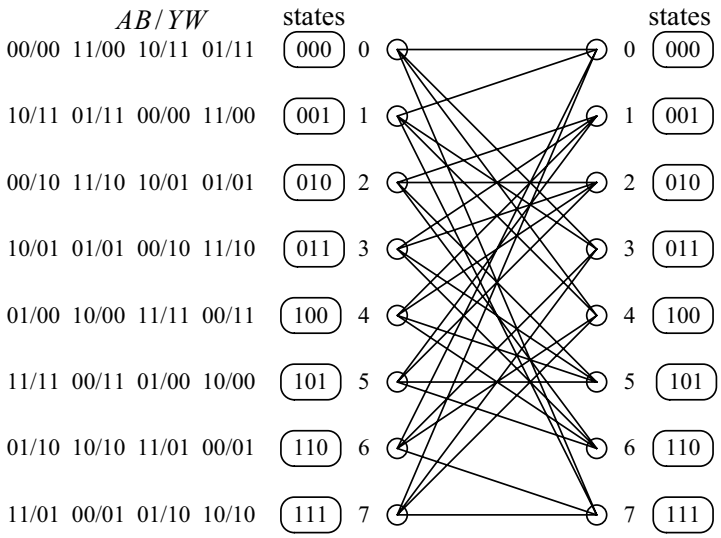
$$\begin{aligned} S_{1,t} &= S_{1,t-1} \oplus S_{3,t-1} \oplus A_t \oplus B_t \\ S_{2,t} &= S_{1,t-1} \oplus B_t \\ S_{3,t} &= S_{2,t-1} \oplus B_t, \end{aligned} \tag{3.3}$$

where $\oplus$ denotes bit-by-bit XOR operation. This equation can be written in a compact way as

$$\boldsymbol{S}_t = \boldsymbol{G} \cdot \boldsymbol{S}_{t-1} \oplus \boldsymbol{X}_t, \tag{3.4}$$



(a)



(b)

Fig. 3.1 Double-binary CRSC encoder for DVB-RCS turbo code and its corresponding trellis diagram.

where the current state $\boldsymbol{S}_t$, the generator matrix $\boldsymbol{G}$ and the input vector $\boldsymbol{X}_t$ are given by the following expressions:

$$\boldsymbol{S}_t = \begin{bmatrix} S_{1,t} \\ S_{2,t} \\ S_{3,t} \end{bmatrix}; \boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}; \boldsymbol{X}_t = \begin{bmatrix} A_t \oplus B_t \\ B_t \\ B_t \end{bmatrix}.$$

When (3.4) is employed, the following expressions are obtained:

$$\begin{aligned} \boldsymbol{S}_1 &= \boldsymbol{G} \cdot \boldsymbol{S}_0 \oplus \boldsymbol{X}_1 \\ \boldsymbol{S}_2 &= \boldsymbol{G} \cdot \boldsymbol{S}_1 \oplus \boldsymbol{X}_2 \\ &\vdots \\ \boldsymbol{S}_K &= \boldsymbol{G} \cdot \boldsymbol{S}_{K-1} \oplus \boldsymbol{X}_K. \end{aligned} \tag{3.5}$$

$\boldsymbol{S}_K$ can be written in a compact form as a function of the initial state $\boldsymbol{S}_0$ and the input couples entering the encoder between time 1 and K.

$$\boldsymbol{S}_K = \boldsymbol{G}^K \cdot \boldsymbol{S}_0 \oplus \sum_{l=1}^{K} \boldsymbol{G}^{K-l} \cdot \boldsymbol{X}_l \tag{3.6}$$

If the encoding starts in the state $\boldsymbol{S}_0 = \boldsymbol{S}_c$ and ends in the state $\boldsymbol{S}_K = \boldsymbol{S}_c$, then the following expression can be derived from (3.6):

$$\boldsymbol{S}_c = \left(\boldsymbol{I} \oplus \boldsymbol{G}^K\right)^{-1} \cdot \sum_{l=1}^{K} \boldsymbol{G}^{K-l} \cdot \boldsymbol{X}_l. \tag{3.7}$$

The circular encoding is performed in two stages. First, the encoder is initialized in the all-zero state and the input sequence in encoded. The final state $\boldsymbol{S}_K^0 = \sum_{l=1}^{K} \boldsymbol{G}^{K-l} \cdot \boldsymbol{X}_l$ is used to determine the circular state $\boldsymbol{S}_c$ as follows

$$\boldsymbol{S}_c = \left(\boldsymbol{I} \oplus \boldsymbol{G}^K\right)^{-1} \cdot \boldsymbol{S}_K^0. \tag{3.8}$$

In the second and final stage, the encoder is initialized with the circular state $\boldsymbol{S}_c$ and the input sequence is encoded again. In this way, the encoder is guaranteed to start and end in the the same state, namely, the circular state $\boldsymbol{S}_c$.

The existence of such a circular state $\boldsymbol{S}_c$ is guaranteed only if $\left(\boldsymbol{I} \oplus \boldsymbol{G}^K\right)$ is invertible. If K is a multiple of the period of the encoding recursive generator, then such a circular state $\boldsymbol{S}_c$ may not exist. This limitation can be overcome by using the alternative circular coding scheme presented in [56, 58].

3.1.2 Encoding Process

If the data to be encoded contains K couples, where K is not a multiple of the period of the encoding recursive generator, then the existence of circular state $\boldsymbol{S}_c$ is assured. Since the value of the circular state $\boldsymbol{S}_c$ depends on the contents of the sequence to be encoded, determining the circular state requires a *pre-encoding* operation. For the pre-encoding, the encoder is initialized to the all-zero state. Then the data sequence goes through the encoder. At the end of the pre-encoding stage the encoder state is $\boldsymbol{S}^0_{K_1}$ for unpermuted and $\boldsymbol{S}^0_{K_2}$ for permuted data sequence. The circular states $\boldsymbol{S}_{c_1}$ and $\boldsymbol{S}_{c_2}$ corresponding to the first and second RSC encoders are then obtained from the expression $\boldsymbol{S}_{c_i} = \left(\boldsymbol{I} \oplus \boldsymbol{G}^K\right)^{-1} \cdot \boldsymbol{S}^0_{K_i}$, where $i \in \{1, 2\}$. Based on the length K and the final state of pre-encoding, the circular state $\boldsymbol{S}_{c_i}$ can be obtained from Table 3.1.

Table 3.1 Circulation state correspondence table.

$K \mod 7$	$\boldsymbol{S}^0_K$							
	0	1	2	3	4	5	6	7
1	0	6	4	2	7	1	3	5
2	0	3	7	4	5	6	2	1
3	0	5	3	6	2	7	1	4
4	0	4	1	5	6	2	7	3
5	0	2	5	7	1	3	4	6
6	0	7	6	1	3	4	5	2

3.1.3 Encoder Structure

The standardized DVB-RCS encoder is composed of two identical double-binary circular recursive systematic convolutional (CRSC) encoders. Each pair of bits is fed to the encoder

twice in uninterleaved mode (switch in position 1), and twice in interleaved mode (switch in position 2), as shown in Fig. 3.1(a). The data sequence is fed to the encoder in the form of packets of length K couples, where K is not a multiple of seven to make sure that a circular state $\boldsymbol{S}_c$ exists. The output of the encoder at time t is $C_t = (A_t, B_t, Y_{1t}, W_{1t}, Y_{2t}, W_{2t})$ and consists of 6 bits, the input couple $d_t = (A_t, B_t)$, also called the *systematic* part, the redundancy couple (Y_{1t}, W_{1t}) resulting from the first CRSC encoder and the redundancy couple (Y_{2t}, W_{2t}) resulting from the second CRSC encoder. The redundancy bits are also called *parity* bits.

Fig. 3.1(b) shows the trellis diagram of the 8-state double-binary convolutional encoder described above. The input to the encoder (A, B) and their corresponding trellis outputs (Y, W) are shown on the left of each state. The corresponding state transitions are shown for each state. The state transitions from top to bottom, exiting from each state, are represented by the numbers from left to right.

3.1.4 DVB-RCS Interleaving and Puncturing

DVB-RCS interleaving

The permutation is done on two levels, the first one inside the couple (level 1), the second one between couples (level 2). Let K be the number of data couples in each block at the encoder input (each block contains $2K$ information bits). The two levels of interleaving are described as follows [15]:

Level 1:

For $j = 0, \ldots, K-1$

- If $j \mod 2 = 0$, invert the couple $(A_j, B_j) = (B_j, A_j)$

Level 2:

For $j = 0, \ldots, K-1$

- If $j \mod 4 = 0$, then set $P = 0$
- If $j \mod 4 = 1$, then set $P = K/2 + P_1$
- If $j \mod 4 = 2$, then set $P = P_2$
- If $j \mod 4 = 3$, then set $P = K/2 + P_3$
- Set $i = (P_0 \cdot j + P + 1) \mod K$

The j^{th} information symbol is permuted to the new position, i. Table 3.2 provides the combinations of the default parameters P_0, P_1, P_2 and P_3 to be used.

Table 3.2 Turbo code permutation parameters.

Frame size in couples (K)	P_0	$\{P_1, P_2, P_3\}$
48 (12 bytes)	11	{24,0,24}
64 (16 bytes)	7	{34,32,2}
212 (53 bytes)	13	{106,108,2}
220 (55 bytes)	23	{112,4,116}
228 (57 bytes)	17	{116,72,188}
424 (106 bytes)	11	{6,8,2}
432 (108 bytes)	13	{0,4,8}
440 (110 bytes)	13	{10,4,2}
752 (188 bytes)	19	{137,224,600}
848 (212 bytes)	19	{2,16,6}
856 (214 bytes)	19	{428,224,652}
864 (216 bytes)	19	{2,16,6}

DVB-RCS puncturing

The puncturing patterns for the seven rates defined in the DVB-RCS standard [15] (R_c=1/3, 2/5, 1/2, 2/3, 3/4, 4/5, and 6/7) are shown in Table 3.3. The puncturing patterns are the same for both sets of parity (Y_{1t}, W_{1t}) and (Y_{2t}, W_{2t}). A '1' indicates that the redundant bit is transmitted and a '0' indicates that the redundant bit is not transmitted. The first and second rows of each matrix in Table 3.3 gives the repeating puncturing patterns for Y and W, respectively. Note that the code rates 3/4 and 6/7 are exact only if K is a multiple of 3 and 6, respectively. Since the standardized frame sizes are a multiple of 8, the other code rates are exact.

3.2 Decoding of DVB-RCS Turbo Codes

Define $u_t = (A_t, B_t)$ as the transmitted information couple at time t and $c_t = (A_t, B_t, Y_t, W_t)$ as its corresponding output from the CRSC encoder. The encoder outputs are modulated by Gray labelled quadrature phase shift keying (QPSK) with absolute mapping according

Table 3.3 Turbo code puncturing patterns.

Code Rate	puncturing patterns
1/3	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
2/5	$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$
1/2	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
2/3	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$
3/4	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
4/5	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
6/7	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

to the relation $x = 1-2a$, where $a \in \{0,1\}$ and $x \in \{1,-1\}$ are the input and output of the modulator, respectively. The modulated bits are then transmitted over a noisy channel. As shown in Fig. 3.2, at the receiver side, the observed samples are $y_t = (1-2A_t+n_1, 1-2B_t+n_2, 1-2Y_t+n_3, 1-2W_t+n_4,)$, where n_i are samples of the noise from the channel (Fig. 3.2).

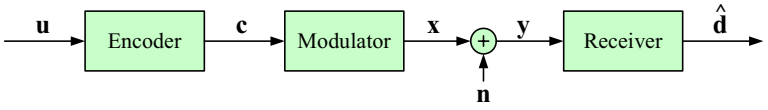


Fig. 3.2 A simple system model of DVB-RCS.

3.2.1 MAP and Log MAP decoding

Turbo codes [21], [22] use constituent codes that allow soft decoding to provide reliability estimates that can be passed between the constituent decoders. The Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm [61] or maximum *a posteriori* (MAP) decoding (also *a posteriori* probability (APP) decoding) computes the log-likelihood ratio (LLR) as

$$L_{apo}^{(z,z')}(\hat{u}_t) \equiv \ln \frac{p(u_t = z \mid \boldsymbol{y})}{p(u_t = z' \mid \boldsymbol{y})}, \tag{3.9}$$

where z and z' are elements of the set of all information symbols. Note that $L^{(z,z)}(u_t)$ is zero and does not need to be considered, because it does not provide any information. For double-binary codes such as DVB-RCS, there are 12 LLRs resulting from the combinations $(z \neq z') \in \phi^0 = \{00, 01, 10, 11\}$. However, all useful information is contained in the three *a posteriori* LLRs given by

$$L_{apo}^{(z)}(\hat{u}_t) \equiv \ln \frac{p(u_t = z \mid \boldsymbol{y})}{p(u_t = 00 \mid \boldsymbol{y})}, \tag{3.10}$$

where $z \in \phi = \{01, 10, 11\}$.

In order to write (3.10) in a simple form, assume that the CRSC encoder goes from state $s' = s_{t-1}$ at time $(t-1)$ to state $s = s_t$ at time t via the input symbol $z = u_t \in \phi^0$. As discussed in detail in Section 2.3, equation (3.10) can be expressed in a simple form

suitable for both software and hardware implementations, namely

$$L_{apo}^{(z)}(\hat{u}_t) = \ln \frac{\sum_{(s' \to s, z)} \alpha_{t-1}(s') \cdot \gamma_t^{(z)}(s', s) \cdot \beta_t(s)}{\sum_{(s' \to s, 00)} \alpha_{t-1}(s') \cdot \gamma_t^{(00)}(s', s) \cdot \beta_t(s)}. \tag{3.11}$$

Fig. 3.3 shows, for DVB-RCS turbo codes, the different branches involved in the computation of the *log-likelihood* ratio $L_{apo}^{(10)}(\hat{u}_t)$ of the information symbol (10). The dashed and solid lines represent the transitions $(s' \to s)$ caused by the inputs (10) and (00), respectively.

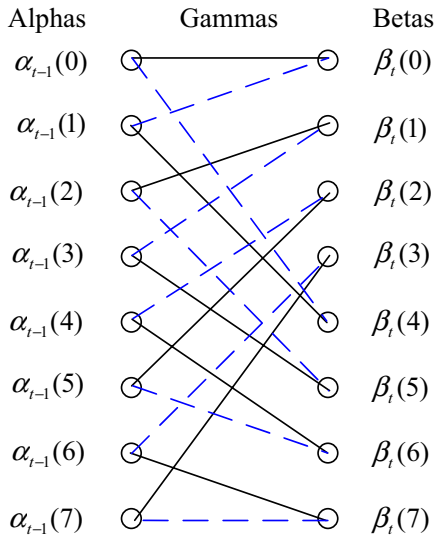


Fig. 3.3 Different branches involved in the computation of the Log-Likelihood Ratio of the information symbol (10) for DVB-RCS turbo codes. The dashed and solid lines represent the transitions $(s' \to s)$ caused respectively by the inputs (10) and (00), respectively

Transition probability (gamma)

In the following discussion, it is assumed that the channel is additive white Gaussian noise (AWGN). The vector $c_t = (A_t, B_t, Y_t, W_t)$ corresponding to the transition $(s' \to s)$ caused by the input symbol $u_t = (A_t, B_t)$ is modulated to $x_t = (1-2A_t, 1-2B_t, 1-2Y_t, 1-2W_t)$ and sent over the AWGN channel. The probability that y_t is received, given u_t, is given by

the probability density function

$$p(y_t \mid u_t) = p(y_t \mid x_t) = \prod_{i=0}^{3} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{y_t[i]-x_t[i]}{\sigma}\right)^2} \tag{3.12}$$

where $x_t[i]$ and $y_t[i]$ are the i^{th} elements of x_t and y_t, respectively.
The transition probability is given by the equation

$$\begin{aligned} \gamma_t^{(z)}(s', s) &= p(s \mid s') \cdot p(y_t \mid s', s) \\ &= p(y_t \mid u_t = z) \cdot p(u_t = z) \end{aligned} \tag{3.13}$$

How to obtain $p(y_t \mid u_t = z)$ will be discussed later.

Forward recursion

Each new alpha metric is the sum of the previous alphas multiplied by the branch metrics along each branch from the *four* previous states s' to the current state s.

$$\alpha_t(s) = \sum_{s'} \alpha_{t-1}(s') \cdot \gamma_t(s', s) \tag{3.14}$$

An example for $\alpha_t(0)$ is shown in Fig. 3.4

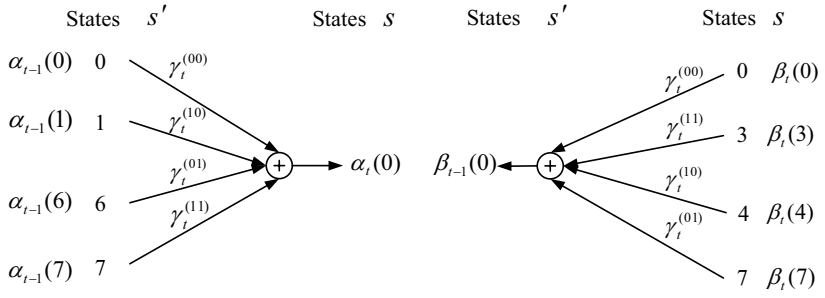


Fig. 3.4 Alpha(0) and Beta(0) extracted from trellis-diagram.

Backward recursion

The computation of the beta metrics is similar to that of the alphas but starting at the end of the trellis and going in the reverse direction, i.e.,

$$\beta_{t-1}(s') = \sum_{s} \beta_t(s) \cdot \gamma_t(s', s). \tag{3.15}$$

An example for $\beta_{t-1}(0)$ is shown in Fig. 3.4.

Extrinsic information

MAP decoding provides probabilistic information about the transmitted information symbol u_t, called *extrinsic* information $L_{ex}^{(z)}(u_t)$, based on the *a posteriori* LLR information $L_{apo}^{(z)}(\hat{u}_t)$, the *a priori* LLR information $L_{apr}^{(z)}(u_t)$ and the intrinsic information $L_{in}^{(z)}(u_t)$ resulting from the systematic part of the received samples. Specifically,

$$L_{ex}^{(z)}(\hat{u}_t) = L_{apo}^{(z)}(\hat{u}_t) - L_{apr}^{(z)}(u_t) - L_{in}^{(z)}(u_t). \tag{3.16}$$

The *a priori* LLR information can be obtained from

$$L_{apr}^{(z)}(u_t) = \ln \frac{p(u_t = z)}{p(u_t = 00)}, \tag{3.17}$$

and the intrinsic information resulting from the systematic part can be obtained from

$$L_{in}^{(z)}(u_t) = \ln \frac{p(y_t^s \mid u_t = z)}{p(y_t^s \mid u_t = 00)}, \tag{3.18}$$

where y_t^s is the systematic part of the received codeword corresponding to the transmitted information symbol $u_t = z$. This extrinsic information $L_{ex}^{(z)}(u_t)$ will be passed to the next decoder as a new estimate for the *a priori* probability of the information symbol $u_t = z$.

The log MAP (L-MAP) algorithm is obtained by applying the natural logarithm to the equations for the gammas, alphas and betas:

$$\begin{aligned} \overline{\gamma}_t^{(z)}(s', s) &\equiv \ln \gamma_t^{(z)}(s', s) \\ &= \ln p(y_t \mid u_t = z) + \ln p(u_t = z) \end{aligned} \tag{3.19}$$

$$\begin{aligned}\overline{\alpha}_t(s) &\equiv \ln \alpha_t(s) \\ &= \ln \sum_{s'} e^{\overline{\alpha}_{t-1}(s')+\overline{\gamma}_t^{(z)}(s',s)}\end{aligned} \tag{3.20}$$

$$\begin{aligned}\overline{\beta}_{t-1}(s') &\equiv \ln \beta_{t-1}(s') \\ &= \ln \sum_{s} e^{\overline{\beta}_t(s)+\overline{\gamma}_t^{(z)}(s',s)}.\end{aligned} \tag{3.21}$$

Consequently, the *a posteriori* LLR for the L-MAP is

$$L_{apo}^{(z)}(\hat{u}_t) = \ln \frac{\sum\limits_{(s'\to s,z)} e^{\overline{\alpha}_{t-1}(s')} \cdot e^{\overline{\gamma}_t^{(z)}(s',s)} \cdot e^{\overline{\beta}_t(s)}}{\sum\limits_{(s'\to s,00)} e^{\overline{\alpha}_{t-1}(s')} \cdot e^{\overline{\gamma}_t^{(00)}(s',s)} \cdot e^{\overline{\beta}_t(s)}}. \tag{3.22}$$

The L-MAP can be easily simplified, resulting in a significant reduction in computational complexity. This simplification is known as max-log MAP and will be discussed in the next section. The use of the logarithm also tends to reduce the severity of overflow and precision issues.

3.2.2 Max-Log MAP

The max-log MAP (ML-MAP) decoding algorithm is an approximation to the MAP decoding algorithm and is widely used in practice because of its lower computational complexity. The reduction in complexity is due, in part, to the transformation of multiplication to addition, but is mainly due to avoiding the computation of the exponential function by using the approximation

$$\ln \sum_j e^{a_j} \approx \max_j (a_j). \tag{3.23}$$

However, this approximation leads to degradation in the error rate performance. Later a technique that reduces this degradation will be introduced.

Applying the *max*-function gives the following expressions for $\overline{\alpha}$, $\overline{\beta}$ and $\overline{\gamma}$

$$\overline{\gamma}_t^{(z)}(s',s) = \ln p(y_t \mid u_t = z) + \ln p(u_t = z) \tag{3.24}$$

$$\overline{\alpha}_t(s) \approx \max_{s'} \left(\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^{(z)}(s',s) \right) \tag{3.25}$$

$$\overline{\beta}_{t-1}(s') \approx \max_{s} \left(\overline{\beta}_t(s) + \overline{\gamma}_t^{(z)}(s', s) \right) \tag{3.26}$$

and

$$\begin{aligned} L_{apo}^{(z)}(\hat{u}_t) \approx & \max_{(s' \to s, z)} \left(\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^{(z)}(s', s) + \overline{\beta}_t(s) \right) \\ & - \max_{(s' \to s, 00)} \left(\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^{(00)}(s', s) + \overline{\beta}_t(s) \right). \end{aligned} \tag{3.27}$$

Taking the natural logarithm of (3.12) results in the expression

$$\begin{aligned} \ln p(y_t | u_t = z) &= \ln p(y_t | x_t) \\ &= 4 \ln \left(\frac{1}{\sqrt{2\pi}\sigma} \right) - \sum_{i=0}^{3} \frac{1}{2\sigma^2} \left(y_t[i] - x_t[i] \right)^2. \end{aligned} \tag{3.28}$$

Dropping the constant term $4 \ln \left(\frac{1}{\sqrt{2\pi}\sigma} \right)$ and eliminating the constant factor $\frac{1}{2\sigma^2}$, (3.28) simplifies to

$$\ln p(y_t | u_t = z) = \ln p(y_t | x_t) \approx \sum_{i=0}^{3} - \left(y_t[i] - x_t[i] \right)^2. \tag{3.29}$$

The intrinsic information $L_{in}^{(z)}(u_t)$, as given in (3.18), can be simplified for each individual symbol $z \in \phi = \{01, 10, 11\}$. The intrinsic information $L_{in}^{(01)}(u_t)$ can be expressed in the following simple form, by ignoring the constant factor $\frac{1}{2\sigma^2}$, which does not influence the reliability of extrinsic values obtained from (3.16), if ML-MAP is used:

$$\begin{aligned} L_{in}^{(01)}(u_t) &= \ln \frac{p(y_t^s \mid u_t = 01)}{p(y_t^s \mid u_t = 00)} = \ln \frac{p(y_t^s \mid x_t = 1, -1)}{p(y_t^s \mid x_t = 1, 1)} \\ &= -\frac{1}{2\sigma^2}(y_t^s[0] - 1)^2 - \frac{1}{2\sigma^2}(y_t^s[1] + 1)^2 + \\ &\frac{1}{2\sigma^2}(y_t^s[0] - 1)^2 + \frac{1}{2\sigma^2}(y_t^s[1] - 1)^2 \\ &= -\frac{1}{2\sigma^2} \cdot 4 y_t^s[1] \\ &\approx -4 y_t^s[1]. \end{aligned} \tag{3.30}$$

Similarly, $L_{in}^{(10)}(u_t)$ and $L_{in}^{(11)}(u_t)$ can be expressed as

$$\begin{aligned} L_{in}^{(10)}(u_t) &\approx -4y_t^s[0] \\ L_{in}^{(11)}(u_t) &\approx -4(y_t^s[0] + y_t^s[1]). \end{aligned} \tag{3.31}$$

Note that the extrinsic values in (3.16) depend on $L_{in}^{(z)}(u_t)$ in (3.30), (3.31) and $L_{apo}^{(z)}(\hat{u}_t)$ in (3.27), which in turn depends on (3.29). To get consistent extrinsic values, the *same* factor $\frac{1}{2\sigma^2}$ in (3.29), (3.30) and (3.31) should be ignored. Thus, the factor 4 in (3.30), (3.31) must be retained.

The use of the simplified (3.29), (3.30) and (3.31) leads to a further reduction in the computational complexity. It is important to note that the standard deviation σ of the AWGN channel is not involved in the computation of the ML-MAP algorithm and so the signal to noise ratio (SNR) is not required, unlike the MAP and L-MAP algorithms.

3.2.3 Initialization and Iterative Decoding

A Priori probabilities of information symbols

In iterative decoding [88] for turbo codes the *extrinsic* information $L_e^{(z)}(u_t)$ from the previous decoder is passed to the next decoder as the *a priori* probability $p(u_t = z)$ of the transmitted information symbol $u_t = z$:

$$L_{ex}^{(z)}(u_t) \equiv \ln \frac{p(u_t = z)}{p(u_t = 00)}. \tag{3.32}$$

For simplicity, $L_{ex}^{(z)}$ is used as an abbreviation for $L_{ex}^{(z)}(u_t)$. Since

$$p(u_t = 00) + p(u_t = 01) + p(u_t = 10) + p(u_t = 11) = 1, \tag{3.33}$$

and

$$p(u_t = z) = p(u_t = 00) \cdot e^{L_{ex}^{(z)}}, \tag{3.34}$$

it follows that [89]

$$\begin{aligned} p(u_t = 00) &= \frac{1}{1 + e^{L_{ex}^{(01)}} + e^{L_{ex}^{(10)}} + e^{L_{ex}^{(11)}}} = b \\ p(u_t = 01) &= b \cdot e^{L_{ex}^{(01)}} \\ p(u_t = 10) &= b \cdot e^{L_{ex}^{(10)}} \\ p(u_t = 11) &= b \cdot e^{L_{ex}^{(11)}}. \end{aligned} \tag{3.35}$$

Applying (3.23) to (3.35) leads to the following expressions [89]

$$\begin{aligned} \ln p(u_t = 00) &= -\max(0, L_{ex}^{(01)}, L_{ex}^{(10)}, L_{ex}^{(11)}) \\ \ln p(u_t = 01) &= L_{ex}^{(01)} - \max(0, L_{ex}^{(01)}, L_{ex}^{(10)}, L_{ex}^{(11)}) \\ \ln p(u_t = 10) &= L_{ex}^{(10)} - \max(0, L_{ex}^{(01)}, L_{ex}^{(10)}, L_{ex}^{(11)}) \\ \ln p(u_t = 11) &= L_{ex}^{(11)} - \max(0, L_{ex}^{(01)}, L_{ex}^{(10)}, L_{ex}^{(11)}). \end{aligned} \tag{3.36}$$

The scalar b in (3.35) is a fixed value for each trellis stage. Setting b to any non-zero value will not have any effect on the error rate performance. To avoid the computation of the *max*-function in (3.36) and thereby lower the computational complexity, the scalar b can be set to 1, which leads to the following simple expressions:

$$\begin{aligned} \ln p(u_t = 00) &= 0 \\ \ln p(u_t = 01) &= L_{ex}^{(01)} \\ \ln p(u_t = 10) &= L_{ex}^{(10)} \\ \ln p(u_t = 11) &= L_{ex}^{(11)}. \end{aligned} \tag{3.37}$$

For the first half-iteration, it is assumed that all information symbols are equally likely, because there is no *a priori* information available (i.e., $L_{ex}^{(00)} = L_{ex}^{(01)} = L_{ex}^{(10)} = L_{ex}^{(11)} = 0$). Using (3.36) or (3.37) for the first half-iteration leads to

$$\begin{aligned} \ln p(u_t = 00) &= 0 \\ \ln p(u_t = 01) &= 0 \\ \ln p(u_t = 10) &= 0 \\ \ln p(u_t = 11) &= 0. \end{aligned} \tag{3.38}$$

For subsequent iterations of the first decoder and all iterations of the second decoder, equation (3.37) is used.

After some number of iterations a hard decision is made based on the *a posteriori* LLRs of all information symbols. For simplicity, $L_{apo}^{(z)}$ is used as an abbreviation for $L_{apo}^{(z)}(\hat{u}_t)$

$$\begin{aligned} L_{apo}^{(z)} &= \max(L_{apo}^{(00)} = 0, L_{apo}^{(01)}, L_{apo}^{(10)}, L_{apo}^{(11)}) \\ \hat{u}_t &= z. \end{aligned} \tag{3.39}$$

Overlapping for CRSC codes

Let the frame to be decoded have K symbols. For circular coding the decoder does not know the initial state of the encoder at time $t = 0$ or $t = K$, so *pre-computing* of alphas and betas over M trellis stages must be done to get reliable initialization values for them. The pre-computing of alphas starts at time $t = (K - M)$ and betas at time $t = M$. The alpha values at time $t = K$ become the initial alpha values at time $t = 0$ and the beta values at time $t = 0$ become the initial beta values at time $t = K$, as shown in Fig. 3.5.

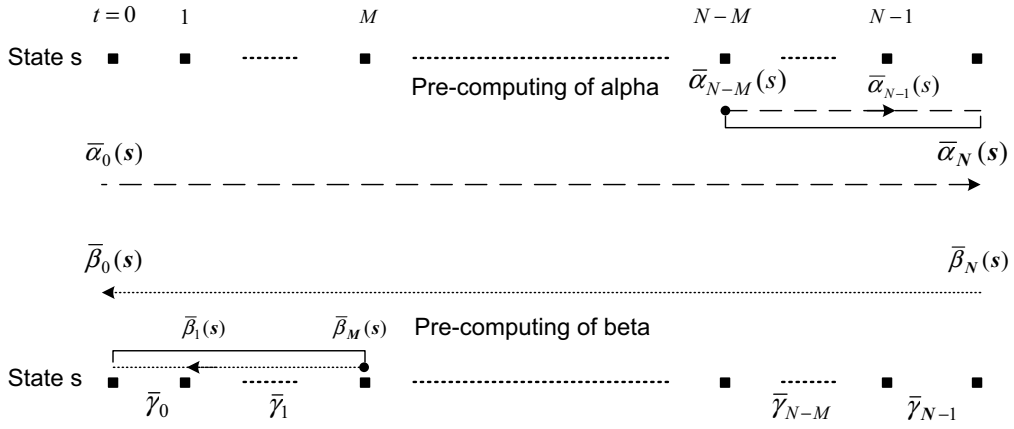


Fig. 3.5 Computing of alphas and betas for circular decoder.

Iterative decoding

Fig. 3.6 shows a block diagram of the iterative decoder for the DVB-RCS double-binary turbo code. Let $\boldsymbol{u}$ be the transmitted data sequence. For the first half-iteration, the *a priori* LLRs $\boldsymbol{L}_{apr}^{1(z)}(\boldsymbol{u})$, $z \in \{01, 10, 11\}$, are initialized to zero. The first decoder (DEC1) computes the *a posteriori* LLR $\boldsymbol{L}_{apo}^{1(z)}(\hat{\boldsymbol{u}})$ based on $\boldsymbol{L}_{apr}^{1(z)}(\boldsymbol{u})$, the received systematic symbols $\boldsymbol{y}^S$, the

received first set of parity $\boldsymbol{y}^{P1}$ and the structure of first encoder. DEC1 then computes the extrinsic information $\boldsymbol{L}_{ex}^{1(z)}(\boldsymbol{u}) = \boldsymbol{L}_{apo}^{1(z)}(\hat{\boldsymbol{u}}) - \boldsymbol{L}_{apr}^{1(z)}(\boldsymbol{u}) - \boldsymbol{L}_{in}^{1(z)}(\boldsymbol{u})$. The extrinsic information from DEC1 is interleaved and passed to the second decoder (DEC2) where it is used as the *a priori* information $\boldsymbol{L}_{apr}^{2(z)}(\boldsymbol{u})$. Interleaving of the extrinsic LLRs occurs in two levels. In the first level, at each time instant t where $(t \mod 2) = 0$, the extrinsic values $\boldsymbol{L}_{ex}^{(00)}$, $\boldsymbol{L}_{ex}^{(01)}$, $\boldsymbol{L}_{ex}^{(10)}$ and $\boldsymbol{L}_{ex}^{(11)}$ becomes $\boldsymbol{L}_{ex}^{(00)}$, $\boldsymbol{L}_{ex}^{(10)}$, $\boldsymbol{L}_{ex}^{(01)}$ and $\boldsymbol{L}_{ex}^{(11)}$, respectively. In the second level, the entire extrinsic information are permuted according to level 2 in 3.1.4.

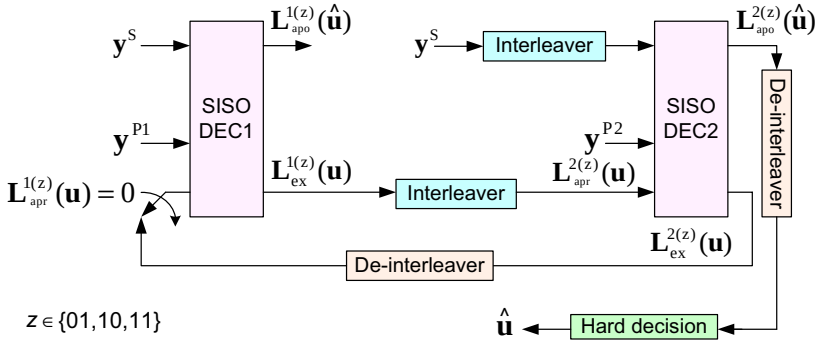


Fig. 3.6 Iterative decoder for DVB-RCS turbo code.

DEC2 computes the *a posteriori* log-likelihood-ratio $\boldsymbol{L}_{apo}^{2(z)}(\hat{\boldsymbol{u}})$ based on interleaved extrinsic information from the first decoder $\boldsymbol{L}_{apr}^{2(z)}(\boldsymbol{u})$, the permuted received systematic symbols, the received second set of parity $\boldsymbol{y}^{P2}$ and the structure of second encoder. DEC2 computes the extrinsic information $\boldsymbol{L}_{ex}^{2(z)}(\boldsymbol{u}) = \boldsymbol{L}_{apo}^{2(z)}(\hat{\boldsymbol{u}}) - \boldsymbol{L}_{apr}^{2(z)}(\boldsymbol{u}) - \boldsymbol{L}_{in}^{2(z)}(\boldsymbol{u})$. DEC1 will use the de-interleaved extrinsic information from DEC2 as *a priori* information $\boldsymbol{L}_{apr}^{1(z)}(\boldsymbol{u})$ in the subsequent iterations. This procedure is repeated for each iteration. After a number of iterations, a hard decision is made based on the de-interleaved *a posteriori* log-likelihood-ratio. DEC1 and DEC2 have the same structure since the two component codes of the DVB-RCS turbo code are the same. If puncturing is involved, the demodulator output corresponding to the punctured parities is initially set to zero.

3.2.4 Enhanced Max-Log MAP and Enhanced Log MAP Decoding

In iterative decoding [88], both decoders need *a priori* probabilities of the information symbols to compute the *a posteriori* LLRs. Each decoder produces a reliability value, called *extrinsic* information, for each information symbol and passes it to the other decoder. In this way each decoder takes advantage of the extrinsic information computed by the other.

The extrinsic output of the first decoder in the first iteration depends *only* on the systematic bits and the first set of parity, due to the fact that the extrinsic input to DEC1 is set to *zero* in the first iteration. The extrinsic output of the second decoder depends on the systematic bits, the second set of parity and the *non-zero* extrinsic input coming from DEC1, which in turn depends on the systematic bits. This means that information derived from the systematic bits will be used again by the second decoder, which makes the extrinsic output of DEC2 sub-optimal. The extrinsic output of DEC2 in the first iteration will be used by DEC1 in the second iteration, making the extrinsic output of DEC1 in the second and subsequent iterations sub-optimal. In other words, the iterative decoding for turbo codes with MAP decoding is optimal only for the first *half-iteration*. For all other iterations it is sub-optimal, due to the fact that the extrinsic input into the decoders is dependent on information derived from the systematic bits, i.e., the extrinsics for the two decoders are correlated.

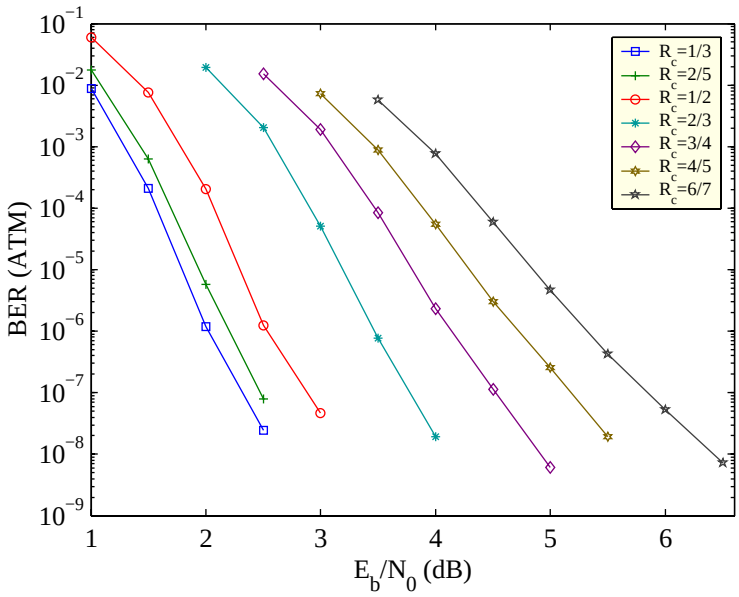
The idea behind enhanced log MAP (EL-MAP) and enhanced max-log MAP (EML-MAP) is to scale the extrinsic output of each decoder with an appropriate value, so that each decoder can give a better extrinsic estimate to the next decoder. The improvement of EML-MAP over ML-MAP is probably because the scaled extrinsic reduces the overestimation of the extrinsic produced by ML-MAP compared with L-MAP.

The idea of scaling the extrinsic information was first introduced in [90], where the extrinsic output of a soft-output-Viterbi-algorithm (SOVA) [68] is scaled by an appropriate scale factor (SF). Scaling the extrinsic information for ML-MAP was introduced in [91]. Similar enhancements have also been presented in [92], [93]. EML-MAP is applied here to DVB-RCS (double-binary turbo code) using appropriate SFs [36].

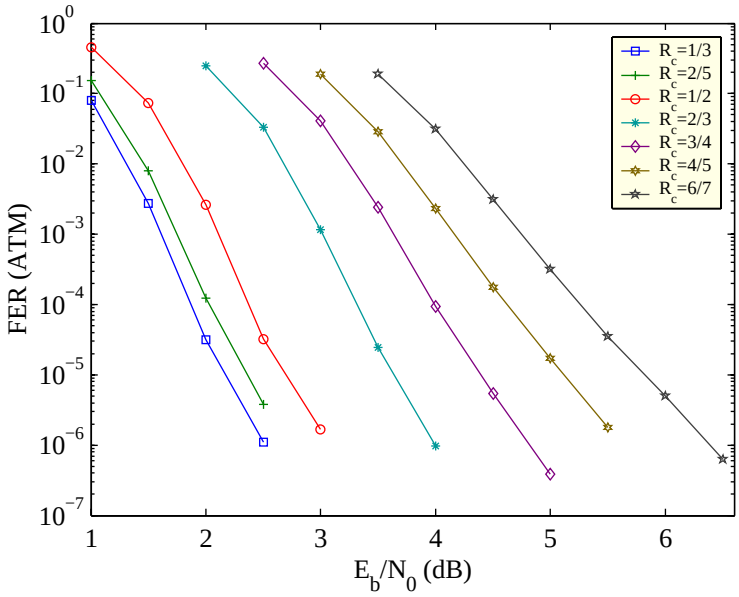
Note that iterative decoding of turbo codes with log MAP (L-MAP) is not optimal in the sense of making a *maximum likelihood* decision [94]. However, scaling the extrinsic information of L-MAP with an appropriate SF improve the iterative decoding performance. This approach is referred to in this thesis as enhanced log MAP (EL-MAP).

3.3 Simulation Results

The BER and FER for the seven standardized code rates were simulated using ML-MAP for the ATM packet size of 424 bits and the MPEG packet size of 1504 bits (see Fig. 3.7 and Fig. 3.8). The number of iterations was set at 8.

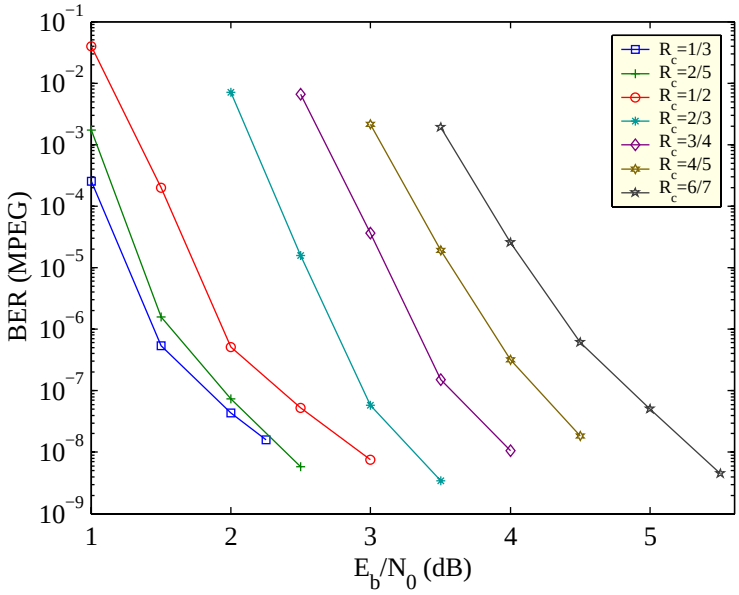


(a)

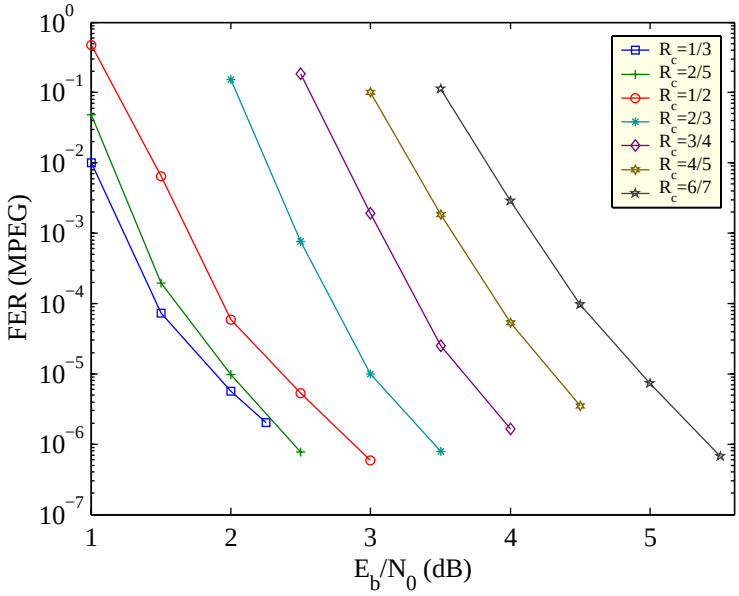


(b)

Fig. 3.7 Performance of ML-MAP decoding for ATM packet size and various code rates using 8 iterations. The size of overlap is 150 bits (75 symbols) for all code rates. For all code rates, at least 100 ATM packet errors were counted at the highest simulated SNR value.



(a)



(b)

Fig. 3.8 Performance of ML-MAP decoding for MPEG packet size and various code rates using 8 iterations. The size of overlap is 150 bits (75 symbols) for all code rates. For all code rates, at least 30 MPEG packet errors were counted at the highest simulated SNR value.

The four algorithms (ML-MAP, EML-MAP, L-MAP and EL-MAP) was compared for rate 1/3 using ATM and MPEG packet sizes. The maximum number of iterations was fixed at 8. For computing the initial α_i and β_i values, the overlap for circular decoding is set to 100 bits (50 symbols) for both ATM and MPEG packets. To reduce the statistical differences, the same noise sequence was used for the four algorithms at each SNR value. The results of these comparisons are shown below. Each subsection discusses different decoding parameters and their effect on performance.

3.3.1 Fixed Scale Factor over all Iterations

Simulations were run with SFs from 0.05 to 0.95 (in steps of 0.05). As expected, it has been found that the appropriate value of SF depends on the value of SNR. The higher the SNR, the higher the SF should be. It was found that the SF of 0.75 and 0.9 are good choices over all iterations for un-punctured EML-MAP and EL-MAP, respectively.

EML-MAP **versus ML-**MAP

Compared to ML-MAP, EML-MAP (SF=0.75) has an improvement of about 0.2 dB at moderate SNRs for both BER and FER for both packet sizes. As mentioned above the higher the SNR, the higher the SF. Thus, the higher the SNR, the smaller the improvement with EML-MAP compared to ML-MAP. Simulation results are shown in Fig. 3.9 and Fig. 3.10.

EML-MAP **versus L-**MAP

For both packet sizes, EML-MAP (SF=0.75) reduces the gap from L-MAP to about 0.1 dB at low to medium SNRs for both BER and FER. Also, the higher the SNR, the lower is the gap between EML-MAP and L-MAP. At high SNRs EML-MAP outperforms L-MAP for both packet sizes (see Fig. 3.9 and Fig. 3.10).

EL-MAP **versus L-**MAP

For both packet sizes, EL-MAP (SF=0.9) has an improvement of about 0.1 dB versus L-MAP for both BER and FER at high SNRs (see Fig. 3.9 and Fig. 3.10). Note that the BER/FER of EML-MAP (SF=0.75) and EL-MAP (SF=0.9) can be better than the

BER/FER of L-MAP. This is because iterative decoding is sub-optimal due to the correlation effects between the component decoders.

3.3.2 Iteration Dependent Scale Factor

Fig. 3.11 shows that an iteration dependent SF with constant step size over the interval [0.70, 0.85] gives a small improvement in error rate performance compared to a fixed SF of 0.75. Note that the first and the last value of the interval correspond to the SFs for the first and the eighth iteration, respectively. Also, the extrinsic information from the first and the second decoder use the same scale factor.

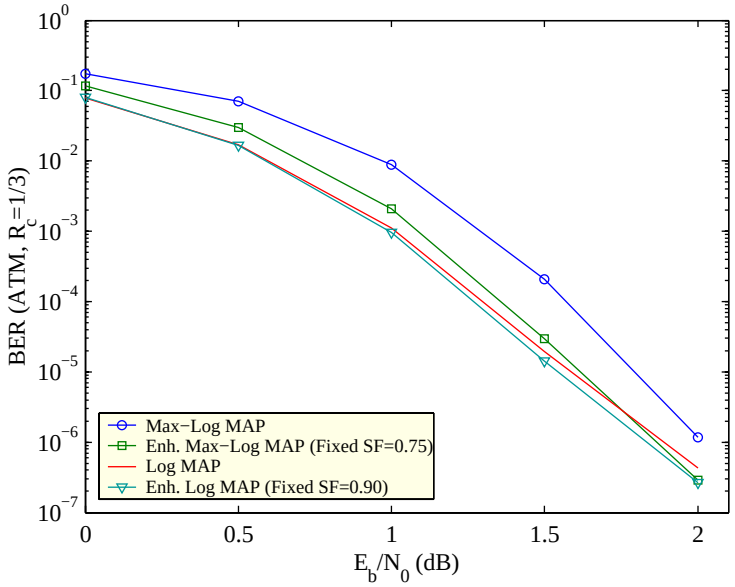
3.3.3 Early Stopping

Computational complexity can be substantially reduced by using early stopping to reduce the average number of iterations. The effect of early stopping [95] is tested based on the EML-MAP approach. Define B as the number of consecutive sets of hard decisions that must agree before stopping. Thus, the minimum value for B is 2. The hard decisions are made based on the LLR *a posteriori* information $\boldsymbol{L}_{apo}$, which are the sum of the *scaled* (SF=0.75) extrinsic outputs of the previous decoder $\boldsymbol{L}_{ex}^{(prev.)}(scaled)$, the intrinsic information $\boldsymbol{L}_{in}$ and the *unscaled* (SF=1.00) extrinsic outputs from the current decoder $\boldsymbol{L}_{ex}^{(curr.)}(unscaled)$, i.e.,

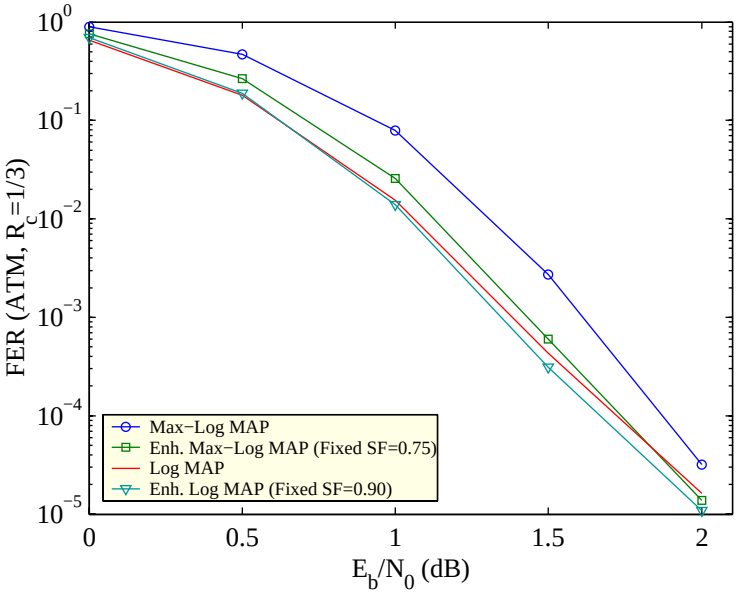
$$\boldsymbol{L}_{apo} = \boldsymbol{L}_{in} + \boldsymbol{L}_{ex}^{(prev.)}(scaled) + L_{ex}^{(curr.)}(unscaled). \tag{3.40}$$

Note that the hard decisions are based on unscaled extrinsic outputs from the current decoder. It was found that using scaled extrinsic outputs from the current decoder led to a degradation in error rate performance with early stopping [34].

Fig. 3.12 shows almost no degradation in BER/FER by choosing $B = 2$. Note that the curves for early stopping and no early stopping in Fig. 3.12 are essentially superimposed. Using $B = 2$ and a maximum number of full-iterations equal to 8 leads to an average reduction in computational complexity by a factor of 2 to 4 for SNR values above 1.5 dB, which is a significant reduction. Note that with B=2 the extra processing is only a single half-iteration on average compared to ideal (genie) early stopping. With genie early stopping, it is assumed that the transmitted codeword is known at the receiver and that

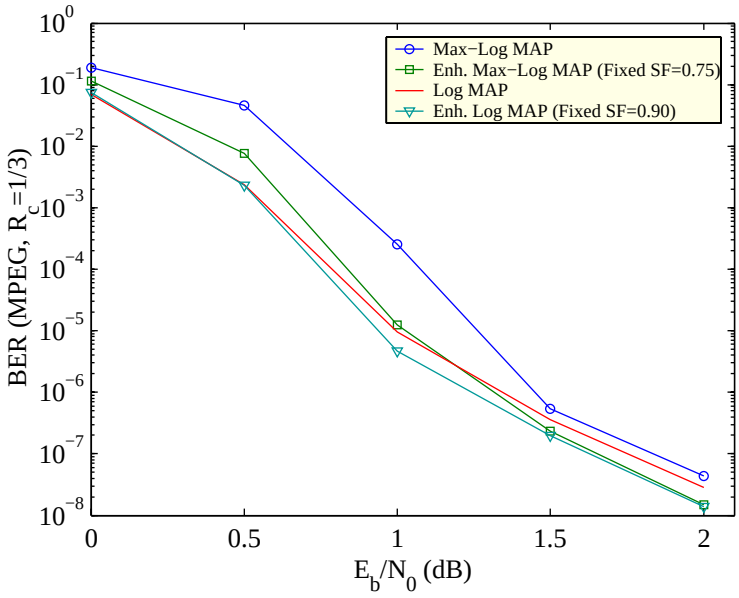


(a)

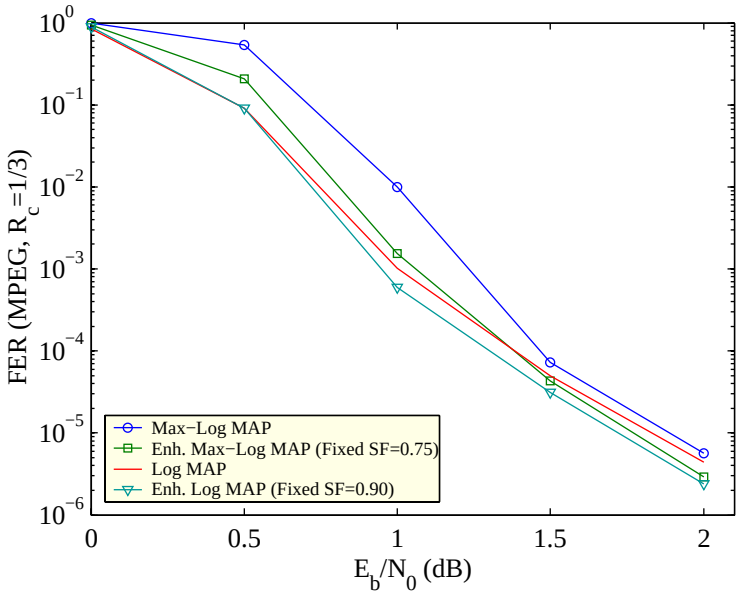


(b)

Fig. 3.9 Comparison of four decoding algorithms for DVB-RCS with ATM packet size. The size of overlap is 100 bits (50 symbols). The code rate is 1/3 and the number of iterations is 8. 10 million ATM packets were simulated at 2 dB.

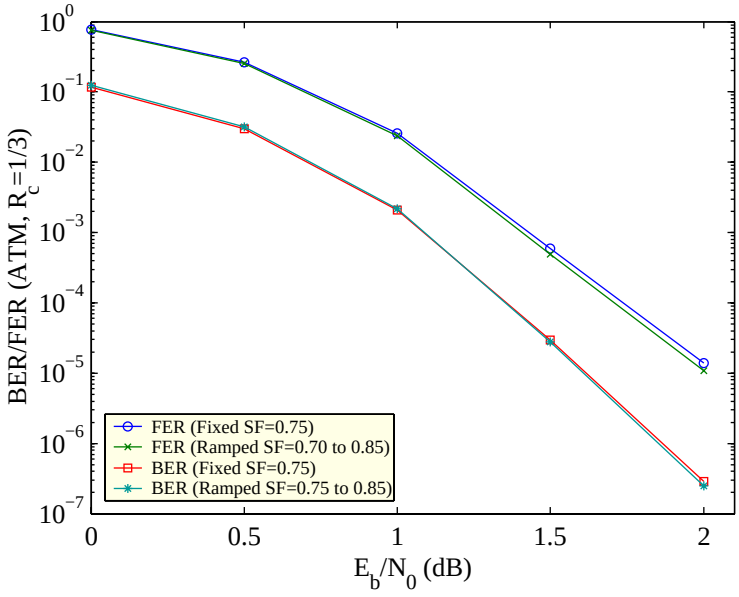


(a)

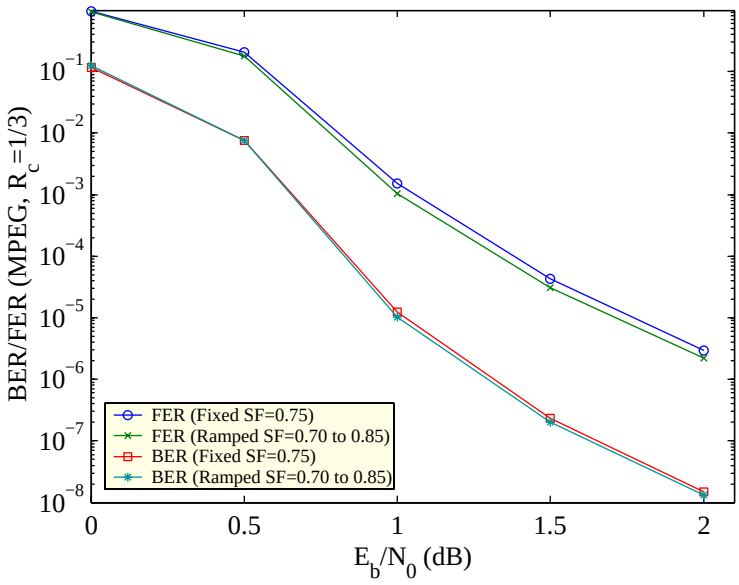


(b)

Fig. 3.10 Comparison of four decoding algorithms for DVB-RCS with MPEG packet size. The size of overlap is 100 bits (50 symbols). The code rate is 1/3 and the number of iterations is 8. 15 million MPEG packets were simulated at 2 dB.



(a)



(b)

Fig. 3.11 Performance of EML-MAP decoding using ramped and fixed scale factor. The size of overlap is 100 bits (50 symbols) for ATM and MPEG packets. The number of iterations is 8. 10 million ATM packets and 15 million MPEG packets were simulated at 2 dB.

the decoder stops when at any half-iteration the transmitted codeword is produced.

Assuming ML-MAP or EML-MAP, it has been shown for single-binary turbo codes that a values of B higher than 2 is required for high SNRs [34] where the error performance curve starts to flare. However, it has been shown recently that $B = 2$ is sufficient to avoid any noticeable error rate degradation for all SNR values if path ambiguities in the decoding trellis are removed before applying the early stopping criterion. That is, the early stopping criterion is applied only if the *a posteriori* LLR values are not ambiguous (i.e., all *a posteriori* LLRs are different) [84]. Similar behavior is expected with double-binary turbo codes.

Note that this early stopping case ($B = 2$) works well only for ML-MAP and EML-MAP, where the SISO decisions correspond to the maximum likelihood codeword [65], as with a Viterbi decoder [64]. Higher values for B are needed for L-MAP and EL-MAP, because the SISO decisions correspond to the most likely individual bit decisions, not the most likely bit sequence.

From previous work, it has been observed that early stopping with $B = 2$ works well for dual-terminated [55] single-binary turbo codes. A value of $B = 2$ is also expected to work well with single-binary turbo codes. If trellises are not terminated properly, $B > 2$ may be required.

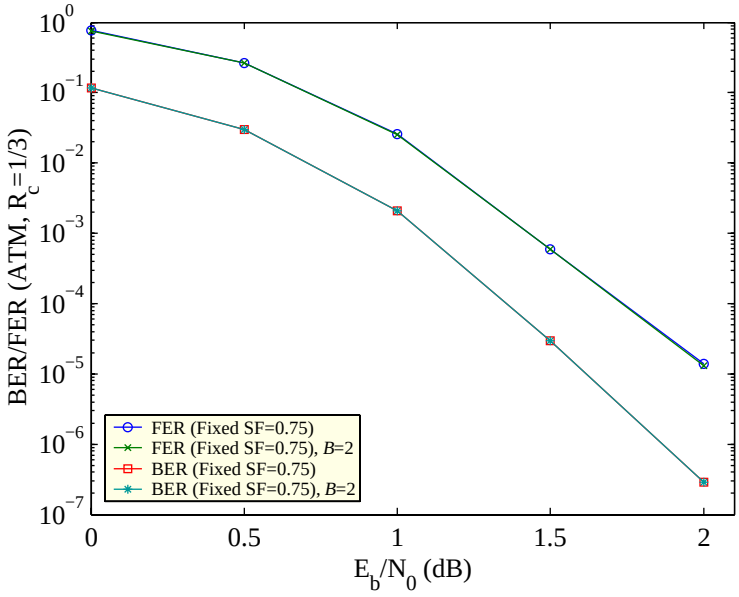
3.3.4 The Effect of Overlap on Error Performance

Due to the nature of tail-biting, the turbo decoder does not have any knowledge about the start and end states of either encoders. To get reliable initial values for the alphas and betas at the beginning of the decoding process, a pre-computation of the alphas and betas over a number of trellis sections must be done.

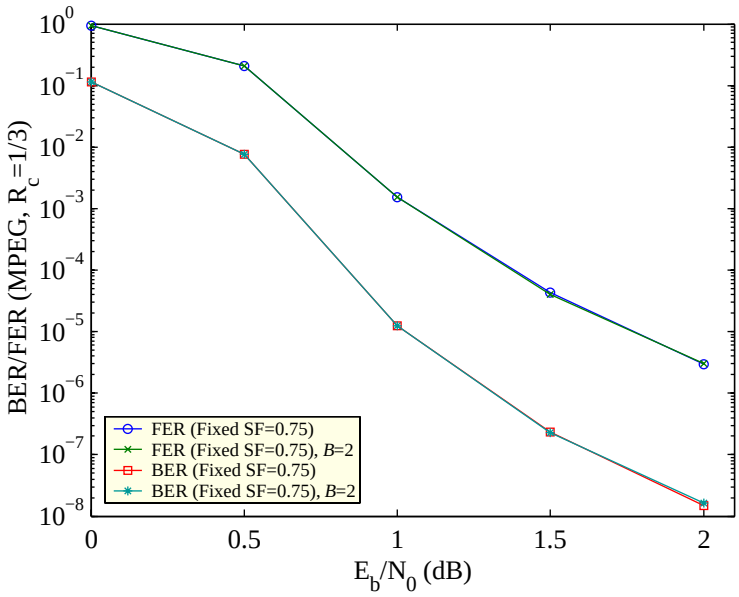
As shown for ATM packets in Fig. 3.13 and Fig. 3.14, setting the size of overlap to 20 bits is enough for rate 1/3, whereas an overlap size of at least 50 bits is needed for rate 4/5. Fig. 3.15 shows for MPEG packets that an overlap size of 20 bits is enough for rate 1/3. An overlap size of 150 bits (75 symbols) is a safe choice over all code rates for both ATM and MPEG packets.

From the performance results shown in Figs. 3.13, 3.14, and 3.15 the following points can be made:

- There is a severe flaring of the error rate curves at medium to high SNRs if the



(a)



(b)

Fig. 3.12 Performances of EML-MAP decoding without early stopping is compared to that of EML-MAP decoding with early stopping (B=2). The size of overlap is 100 bits (50 symbols) for ATM and MPEG packets. The maximum number of iterations is 8. 10 million ATM packets and 15 million MPEG packets were simulated at 2 dB.

pre-computation of alphas and betas is ignored (i.e., size of overlap is 0 bit).

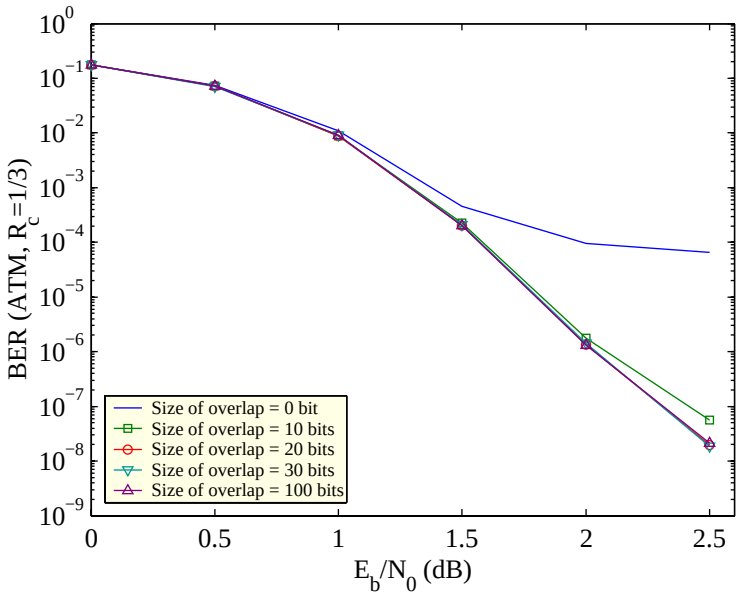
- The best overlap length depends on both the code rate and the value of SNR.
- The greater the overlap length, the better the error performance obtained at high SNRs. For low SNRs, increasing the overlap length did not improve the error performance.
- The best overlap length depends on packet length. In general, longer packets require longer overlap size in order to minimize the degradation in error performance.

3.4 Conclusion

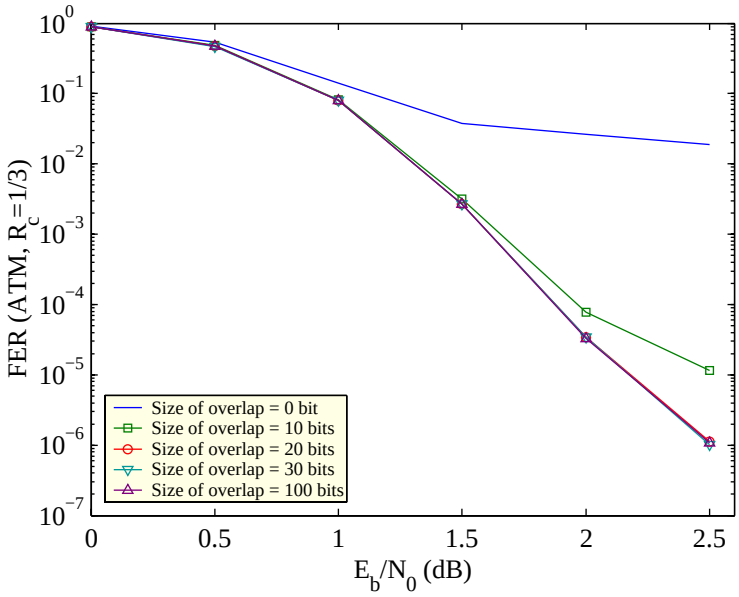
The EML-MAP and EL-MAP decoding algorithms have been introduced for DVB-RCS turbo codes. The BER/FER performance of EML-MAP is very close to that of L-MAP. At high SNRs, both EML-MAP and EL-MAP outperform L-MAP. Compared to EML-MAP with a fixed SF of 0.75, slightly better results have been obtained with ramp scale factors of a constant step size, where a SF of 0.70 for the first full-iteration and a SF of 0.85 for the eighth full-iteration have been used.

A simple and effective early stopping criterion that reduces the average computational complexity for EML-MAP at medium to high SNRs by a factor of 2 to 4 with almost no degradation in BER/FER has been introduced for DVB-RCS turbo codes, and its performance given for a maximum number of full-iteration equal to 8. Note that this reduction in average computational complexity is proportional to the number of full-iterations.

The error performance of tail-biting turbo codes depends on the length of the overlap. The best overlap length depends on the code rate and SNR value. For high SNRs and longer packets, an increase in the overlap length improved the observed error rate performance.

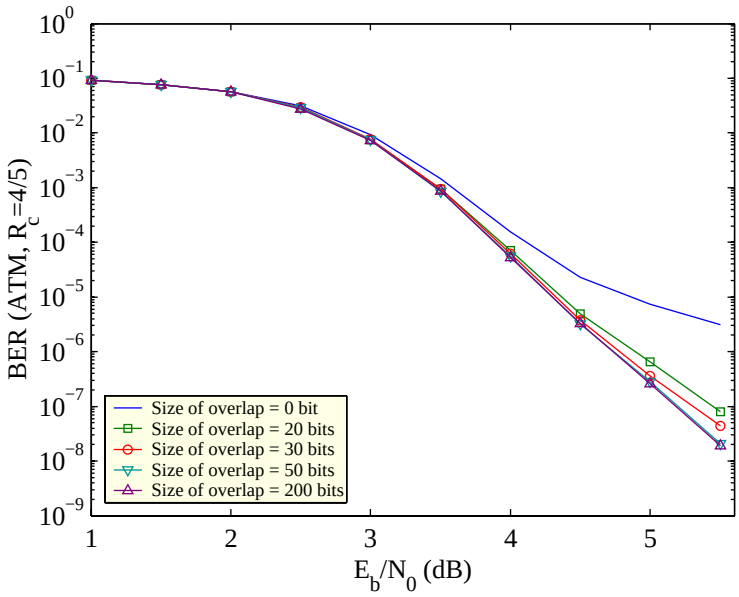


(a)

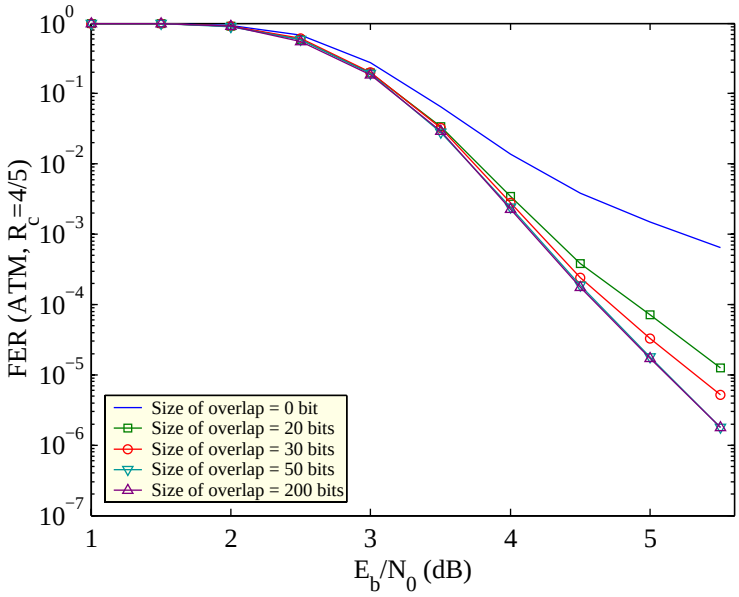


(b)

Fig. 3.13 Performance of ML-MAP decoding for ATM packet sizes using various overlap lengths. The sizes of overlap are 0, 10, 20, 30 and 100 bits. The code rate is 1/3 and the number of iteration is 8. 50 ATM packet errors were counted at the highest simulated SNR value.

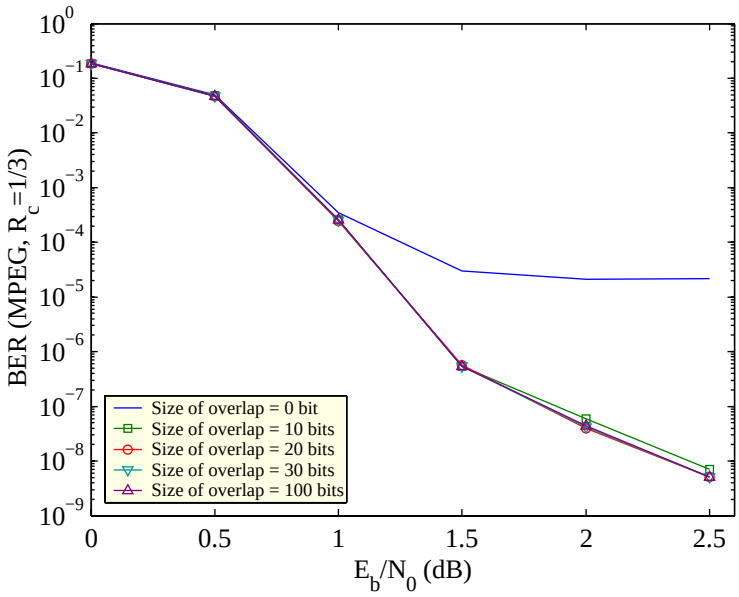


(a)

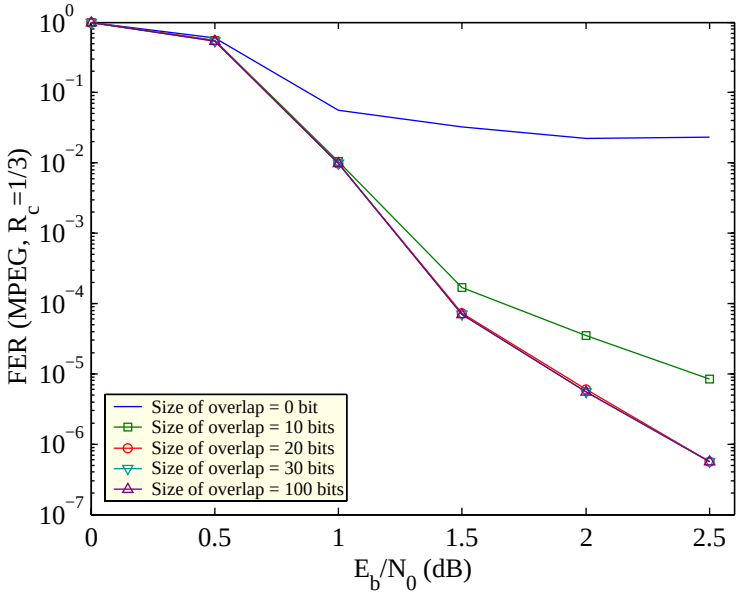


(b)

Fig. 3.14 Performance of ML-MAP decoding for ATM packet sizes using various overlap lengths. The sizes of overlap are 0, 20, 30, 50 and 200 bits. The code rate is 4/5 and the number of iteration is 8. 100 ATM packet errors were counted at the highest simulated SNR value.



(a)



(b)

Fig. 3.15 Performance of ML-MAP decoding for MPEG packet sizes using various overlap lengths. The sizes of overlap are 0, 10, 20, 30 and 100 bits. The code rate is 1/3 and the number of iteration is 8. 25 MPEG packet errors were counted at the highest simulated SNR value.

Chapter 4

Distance Measurement Methods for Turbo Codes

Consider the transmission of a linear binary code $C(\widetilde{N}, \widetilde{K})$ ($\widetilde{N}$ is the codeword length in bits, $\widetilde{K}$ is the number of information bits) over the additive white gaussian noise (AWGN) channel using binary phase-shift keying (BPSK) or quadrature phase-shift keying (QPSK) modulation. Applying maximum-likelihood (ML) decoding, the frame error rate (FER) and bit error rate (BER) are upper bounded by the union bounds [18]:

$$\text{FER} \leq \sum_{d \geq d_{\min}} A_{\text{d}} Q\left(\sqrt{2d \frac{\widetilde{K}}{\widetilde{N}} \frac{E_{\text{b}}}{N_0}}\right) \tag{4.1}$$

$$\text{BER} \leq \sum_{d \geq d_{\min}} \frac{W_{\text{d}}}{\widetilde{K}} Q\left(\sqrt{2d \frac{\widetilde{K}}{\widetilde{N}} \frac{E_{\text{b}}}{N_0}}\right) \tag{4.2}$$

Here, $d_{\min}$ is the minimum distance of the code, the multiplicity A_{d} is the number of codewords with Hamming weight d, the information bit multiplicity W_{d} is the sum of the Hamming weights of the A_{d} input sequences generating the codewords with Hamming weight d, the function $Q(x)$ is given by the expression $\frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt$, E_{b} is the energy per information bit and N_0 is the one-sided noise power spectral density. The function $Q(\sqrt{x})$ decreases exponentially with x, thus the first term or first few terms of (4.1) and (4.2) can be used to approximate FER and BER at high SNRs. However, it is important to keep in mind that turbo codes [21, 22] use iterative soft decoding [88], which is sub-optimal

compared to ML decoding. See [96] for examples and further discussion.

The rapid growth in multimedia applications requires error correcting codes that achieve very low error rates at moderate SNR values. Unfortunately, the reliable determination of very low error rates using software simulation may take months or may not be feasible at all. However, simulation results indicate that the first term or first few terms of (4.1) and (4.2) give a good approximation to the performance of turbo codes at high SNRs (see Section 4.2.5). This is useful for the design of turbo code interleavers. Frequently, the problem of finding good turbo code interleavers reduces to determining which of a class of interleavers gives the best performance. This can be predicted by finding which of the interleavers produces the smallest first few terms of (4.1) and (4.2). In order to determine these terms, it is necessary to have a distance measurement method. It is also important that the method have low computational complexity to allow the determination of high distances in a reasonable time.

This chapter discusses Garello's true distance measurement method [27] in detail. This method is extended to tail-biting turbo codes and a technique that reduces the computational complexity is presented [28]. The first term of the distance spectrum for all DVB-RCS standard packets and code rates is presented. New interleavers are designed for all DVB-RCS standard packets and code rates. Simulation results show that the new interleavers for ATM and MPEG packet sizes provide significantly better error performance than the DVB-RCS standard interleavers. Furthermore, Berrou's error-impulse method [33], Garello's all-zero iterative decoding method [97], Crozier's double-impulse method [31, 32] and an improvement to Crozier's method [30] are discussed. Minimum distances obtained with these methods for DVB-RCS turbo codes are compared with those obtained with Garello's true method. The complexities of the various methods are compared.

4.1 Background

Finding efficient and reliable methods to determine the true distance spectrum, or even the true minimum distance, $d_{\min}$, for turbo codes is a big challenge. This is particularly true for long interleavers with potentially high minimum distances.

A number of brute force approaches to computing $d_{\min}$ have been proposed. The basic idea is to consider all possible candidate sequences for the first constituent encoder, then interleave and encode each sequence with the second constituent encoder to find the total

turbo code distance. These brute force approaches are practical, even for large block sizes, as long as $d_{\min}$ remains small (i.e., close to the minimum possible $d_{\min}$). For well designed interleavers, however, the computational complexity quickly becomes unacceptable as the value of $d_{\min}$ increases. See [96] and [27] and the references therein for examples and further discussion.

Assuming a random interleaver of size approaching infinity, one can expect that the $d_{\min}$ of a turbo code is equal to $d_{\min}(2)$, where $d_{\min}(2)$ is the distance due to input sequences of weight 2 [98]. This approach leads to a significant reduction in computational complexity, because only $\binom{\widetilde{K}}{2}$ of the $(2^{\widetilde{K}}-1)$ non-zero input sequences must be tested. Unfortunately, this approach is not applicable for interleavers of small to medium sizes. In fact, $d_{\min}$ can be produced by an input sequence of any weight, even when the interleavers are random and very large. The necessity of considering input-weights larger than two has been confirmed analytically in [99]. Thus $d_{\min}(2)$ gives a loose upper bound on the true $d_{\min}$. Furthermore, from a practical point of view, it is easy to increase $d_{\min}(2)$ by designing interleavers with high spread [26, 100, 60]. In this case $d_{\min}(2)$ tends to be a very loose upper bound on $d_{\min}$. However, this approach can still be used in practice to first discard bad interleavers (i.e., low $d_{\min}(2)$) and the remaining interleavers can then be tested by a true distance measurement method.

The first significant approach to determine the minimum distance was introduced by Robertson [73]. This brute force approach is practical, even for long interleavers, as long as the minimum distance remains small. The complexity of this approach quickly becomes unacceptable if the minimum distance is due to high input-weights, which is often the case for well-designed interleavers. Similar approaches have been presented in [101] and [102]. Robertson's approach was improved in [103] by introducing a *back tracking* algorithm that efficiently computes the distances caused by low input-weights.

Another method based on combining low input-weight patterns that lead to low-weight codewords has been presented in [100]. An essential aspect of this approach is to determine which combinations of low input-weight patterns should be considered. It has been observed that these combinations depend on the *spread*, defined as:

$$S_{new} = \min_{(i,j\neq i)} (\mid \pi(i) - \pi(j) \mid + \mid i - j \mid), \forall i, j \in \{0, \cdots, \widetilde{K}-1\}, \tag{4.3}$$

where π represents an interleaver permutation. A high spread constraint easily eliminates

many of the worst input-weight combinations. However, some input-weight combinations do not improve with spread. Improving the distance for these cases requires specific distance tests to be performed. Fortunately, many of the remaining worst-case input-weight combinations are fairly easy to test with reasonable computational complexity. See [100] for the recommended cases to test and further details. This method has been found to give a fairly tight upper bound on the true $d_{\min}$ and thus is very useful for designing good interleavers in a reasonable time. The approach is very efficient, demonstrated by the fact that it was possible to find a distance upper bound of 110 for a 16-state single-binary turbo code with $\widetilde{K}$=32768 bits [100]. Also, for large interleavers with sufficiently high spread, the upper bounds are guaranteed to be the true minimum distances for all cases up to and including an input-weight of 6. Unfortunately, the higher the upper bound that is achieved, the less likely the bound is to be tight. That is, the more likely the true $d_{\min}$ will be caused by one of the higher input-weight cases not tested.

4.2 Garello's True Method

A novel and efficient method to compute the true $d_{\min}$, the true multiplicity $A_{d_{\min}}$ and the true information bit multiplicity $W_{d_{\min}}$ based on the notion of constrained subcodes has been presented by Garello [27]. This method has been improved and extended in [104, 105] to high rate turbo codes using high rate non-punctured constituent codes. It has also been improved and extended in [29, 28] to tail-biting turbo codes.

Details of Garello's distance measurement method are explained and techniques that reduce the computational complexity of this method are presented. Furthermore, Garello's method for trellis-terminated [44] turbo codes is extended to tail-biting [56, 57, 58] turbo codes. The improved method is applied to both single- and double-binary turbo codes, such as those used in the UMTS/3GPP standard [25] and in the DVB-RCS standard [15], respectively.

4.2.1 Turbo-code Encoder

In general turbo codes use the information sequence and two recursive convolutional encoders in parallel. The method presented here is for trellis-terminated and tail-biting turbo codes and works with puncturing (see Fig. 4.1). For any input sequence $\boldsymbol{u}$ =

$(u_0, \cdots, u_{\widetilde{K}-1}) \in [GF(2)]^{\widetilde{K}}$, where $GF(2)$ represents the binary Galois Field, the turbo-code encoder generates the following three outputs:

- The input sequence itself $\boldsymbol{u}$ (called the systematic bits).
- The m-output sequences $(\boldsymbol{P}11, \cdots, \boldsymbol{P}1m)$ generated by the first feedback convolutional encoder (ENC1) from the input sequence $\boldsymbol{u}$. The z^{th}-output sequence (called z^{th} parity of ENC1) consists of $\widetilde{K}$ bits, also $\boldsymbol{P}1z = (p1z_0, \cdots, p1z_{\widetilde{K}-1})$, where $z \in \{1, \cdots, m\}$.
- The n-output sequences $(\boldsymbol{P}21, \cdots, \boldsymbol{P}2n)$ generated by the second feedback convolutional encoder (ENC2) from the permuted input sequence $\boldsymbol{u}_\pi$, where π consists of $\widetilde{K}$ unique elements in $\{0, \cdots, \widetilde{K}-1\}$ representing the permutation indices. The l^{th}-output sequence (called l^{th} parity of ENC2) consists of $\widetilde{K}$ bits, also $\boldsymbol{P}2l = (p2l_0, \cdots, p2l_{\widetilde{K}-1})$, where $l \in \{1, \cdots, n\}$.

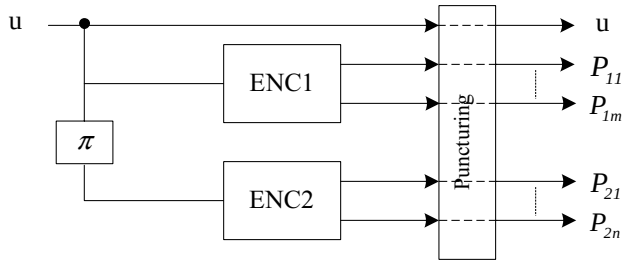


Fig. 4.1 A general turbo-code encoder with two recursive convolutional encoders.

The trellis termination described in [44] starts encoding in the all-zero state and uses additional δ_1 and δ_2 termination-bits to flush the first and the second encoders into the all-zero state, respectively. Here, the first and second encoders are of memory δ_1 and δ_2, respectively. These termination-bits are not included in the interleaver but are sent together with their parities to the decoder. This trellis termination has been adopted in the UMTS/3GPP standard [25] and is referred to in this thesis as UMTS-termination.

A better termination method was introduced in [55]. This method, referred to in this thesis as dual-termination, also uses additional $(n\delta_1 + m\delta_2)$ termination-bits to flush both encoders into the all-zero state. However, these termination-bits are included in the interleaver, which means that they are included with the systematic part of the codeword.

This makes the codeword with dual-termination $(\delta_1 + \delta_2)$ bits longer than the codeword with UMTS-termination. This results in a small decrease in the code rate compared to UMTS-termination, but it usually increases the $d_{\min}$ compared to UMTS-termination.

UMTS-termination [44] and dual-termination [55], both result in a decrease in code rate. Furthermore, UMTS-termination impedes the achievement of high $d_{\min}$. With the various tail-biting approaches [56, 57, 106, 107, 108, 109, 58] each encoder starts and stops in the same data dependent state, without using any termination-bits. This avoids any decrease in the code rate and allows the achievement of high $d_{\min}$.

4.2.2 Computing a Lower Bound on Minimum Distance

This lower bound is a key element in lowering the computational complexity and will be used in the next section to determine the exact $d_{\min}$ in a recursive manner.

With UMTS-termination and dual-termination, both encoders start in the all-zero state and are forced back to the all-zero state after encoding the original input sequence. However, with UMTS-termination the termination-bits are not included in the interleaver. With tail-biting, the first encoder (ENC1) must start and stop in the same state. The second encoder (ENC2) must also start and stop in the same state. Any combination of starting states for the two encoders is allowed. As an example, for an 8-state turbo code there are 64 possible starting state combinations.

Turbo codes are linear codes, thus the minimum distance $d_{\min}$ is given by the codeword with the minimum Hamming weight. The goal is to find *all* input sequences that cause $d_{\min}$ from the $(2^{\widetilde{K}} - 1)$ possible non-zero input sequences. Define $\boldsymbol{u} = (u_0, \cdots, u_j, \cdots, u_{\widetilde{K}-1})$ and $\boldsymbol{u}_\pi = \left(u_{\pi(0)}, \cdots, u_{\pi(j)}, \cdots, u_{\pi(\widetilde{K}-1)}\right)$ as the input sequences into the single-binary ENC1 and ENC2, respectively, where π is an interleaver of length $\widetilde{K}$. In other words, any information bit u_j entering ENC1 at time j will enter ENC2 at time $\pi^{-1}(j)$. Let $w(\boldsymbol{u}) = \text{WE1} + \text{WE2}$ be the Hamming weight of the codeword generated by the input sequence $\boldsymbol{u}$, where WE1 is the sum of the Hamming weights of $\boldsymbol{u}$ and its corresponding parities generated by ENC1,

$$\text{WE1} = \sum_{i=0}^{\widetilde{K}-1} u_i + \sum_{l=1}^{m} \sum_{i=0}^{\widetilde{K}-1} p1l_i, \tag{4.4}$$

and WE2 is the Hamming weight of the parities generated by ENC2,

$$\text{WE2} = \sum_{l=1}^{n} \sum_{i=0}^{\widetilde{K}-1} p2l_i. \tag{4.5}$$

Consider an input sequence $\boldsymbol{u}^j$, where only the first $(j+1)$ information bits $\boldsymbol{u}^{\leq j} = (u_0, \cdots, u_j)$ are known and the other $(\widetilde{K}-j-1)$ bits $\boldsymbol{u}^{>j} = (u_{j+1}, \cdots, u_{\widetilde{K}-1}) = (\times, \cdots, \times)$ are unknown ($\times$ can be 0 or 1). The aim is to find the unknown information bits $\boldsymbol{u}^{>j}$ in $\boldsymbol{u}^j$ that minimize the weight $W^j = \text{WE1}^j + \text{WE2}^j$. Define MWE1^j and MWE2^j as the minimum output-weight generated respectively by the input sequences $\boldsymbol{u}^j$ into ENC1 and $\boldsymbol{u}^j_\pi$ into ENC2. Note that the input bits at positions $(j+1, \cdots, \widetilde{K}-1)$ in $\boldsymbol{u}^j$ leading to MWE1^j are not necessarily the same input bits at positions $(\pi^{-1}(j+1), \cdots, \pi^{-1}(\widetilde{K}-1))$ in $\boldsymbol{u}^j_\pi$ leading to MWE2^j, because MWE1^j and MWE2^j are computed separately based only on the common knowledge of the known bits $\boldsymbol{u}^{\leq j}$. If the $\boldsymbol{u}^{\leq j}$ agree with the first $(j+1)$ information bits of any input sequence $\boldsymbol{u}_{\min}$ that causes $d_{\min}$, then $\text{MW}^j = \text{MWE1}^j + \text{MWE2}^j$ is a lower bound for $d_{\min}$, i.e.,

$$\text{MW}^j \leq (w(\boldsymbol{u}_{\min}) = d_{\min}). \tag{4.6}$$

Computation of MWE2^j for tail-biting

The computation of MWE2^j_{tb} for tail-biting turbo codes is obtained by applying the following modified *forward* Viterbi algorithm (MVA) [27]. Each branch of the trellis of ENC2 is labelled with the usual weight except the irrelevant branches of ENC2 at sections $(\pi^{-1}(0), \cdots, \pi^{-1}(j))$ are labelled with an effectively infinite value (in practice, it is enough to set this value to $\widetilde{N}$). The irrelevant branches are the branches associated with bits of the complementary input sequence $\widetilde{\boldsymbol{u}}^{\leq j}$, where $\widetilde{\boldsymbol{u}}^{\leq j} = \left((1, \cdots, 1) \oplus \boldsymbol{u}^{\leq j}\right)$ and $\oplus$ denotes the bit-by-bit XOR operation. Furthermore, assume that the encoder starts and ends in the state s_x. The steps to compute MWE2^j_{tb} are:

1. Initialize $t = 0$; $w(s_x) = 0$; $w(s \neq s_x) = \infty$.

2. Increase t by 1.

   - Compute all weights of each state s by adding the weight of the branch entering s from state s' and the weight of the state s' at time $(t-1)$, then set $w(s)$ to

the smallest weight. Repeat until $t = \widetilde{K}$.

3. The value MWE2_{tb}^{j} is $w(s_x)$.

This procedure is repeated for all starting states of ENC2 and then the minimum is selected to be MWE2_{tb}^{j}.

The encoding process of dual-termination [55] is different from that of tail-biting because of the use of flush bits that force both encoders simultaneously to start and end in the all-zero state. However, the distance spectrum of dual-termination is basically a subset of the distance spectrum of tail-biting. The determination of MWE2_{dt}^{j} for dual-termination is done by applying the three step algorithm described above for tail-biting, where the start and end state s_x is the all-zero state.

Computation of MWE2^{j} for UMTS-termination

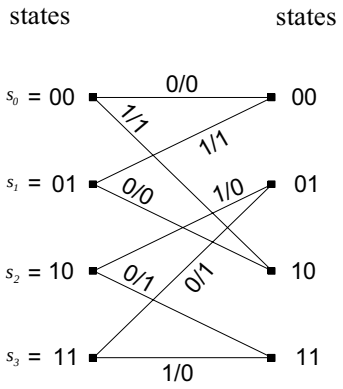
Given any state, the termination-bits [44] and their corresponding weights can be determined offline. Define $w_{\text{umts}}(s_\sigma)$ as the weight resulting from driving the encoder from the state s_σ to the state s_0. The computation of $\text{MWE2}_{\text{umts}}^{j}$ for UMTS-terminated turbo codes is done by adding the offline computed weights $w_{\text{umts}}(s_\sigma)$ to the weights $w(s_\sigma)$ obtained at time $t = \widetilde{K}$ from the already described forward MVA ($w(s_0) = 0$; $w(s \neq s_0) = \infty$) and then selecting the minimum value

$$\text{MWE2}_{\text{umts}}^{j} = \min_{s_\sigma \in \Omega_2} (w(s_\sigma) + w_{\text{umts}}(s_\sigma)), \tag{4.7}$$

where Ω_2 denote the set of all states of ENC2.

Example showing how to compute MWE2^{j} for tail-biting and UMTS-termination

To show how to compute MWE2_{tb}^{j} and $\text{MWE2}_{\text{umts}}^{j}$, consider a 4-state single-binary turbo code (for simplicity) where both encoders have the same generator polynomials (feedback, feedforward)=$(7,5)_{octal}$. The trellis diagram for the constituent encoder and the extra weights resulting from termination-bits for UMTS-termination are depicted in Fig. 4.2(a) and Fig. 4.2(b), respectively. Assume for tail-biting that ENC2 starts and ends in the state s_2, $\boldsymbol{u}^2 = (0, 1, 0, \times, \times, \times, \times)$ and $\pi = (3, 1, 6, 0, 5, 2, 4)$. For UMTS-termination, ENC2 starts in the all-zero state and is forced back to the all-zero state at the end of encoding the original input sequence.



(a) Trellis diagram.

states	termination-bits	output-weight
00	00	0+0=0
01	10	1+1=2
10	11	2+1=3
11	01	1+2=3

(b) Termination-bits and its corresponding weights (input+parity).

Fig. 4.2 Trellis diagram, termination-bits and its corresponding weights for an encoder with generator polynomials (feedback,feedforward)=$(7,5)_{octal}$.

To obtain $\mathrm{MWE2}^2_{tb}$ for tail-biting, as shown in Fig. 4.3(a), the forward MVA ($w(s_2) = 0$; $w(s \neq s_2) = \infty$) is applied leading to an output-weight of 2 at state s_2 and so $\mathrm{MWE2}^2_{tb} = 2$.

The computation of $\mathrm{MWE2}^2_{\mathrm{umts}}$ for UMTS-termination is illustrated in Fig. 4.3(b). The forward MVA ($w(s_0) = 0$; $w(s \neq s_0) = \infty$) is applied leading to the weights $w(s) = (2, 1, 3, 2)$ corresponding to the states (s_0, s_1, s_2, s_3) at the end of the trellis. These weights $w(s)$ are added to the already precomputed weights $w_{\mathrm{umts}}(s) = (0, 2, 3, 3)$ leading to the weights $(2, 3, 6, 5)$ at states (s_0, s_1, s_2, s_3). Thus, $\mathrm{MWE2}^2_{\mathrm{umts}}$ is 2.

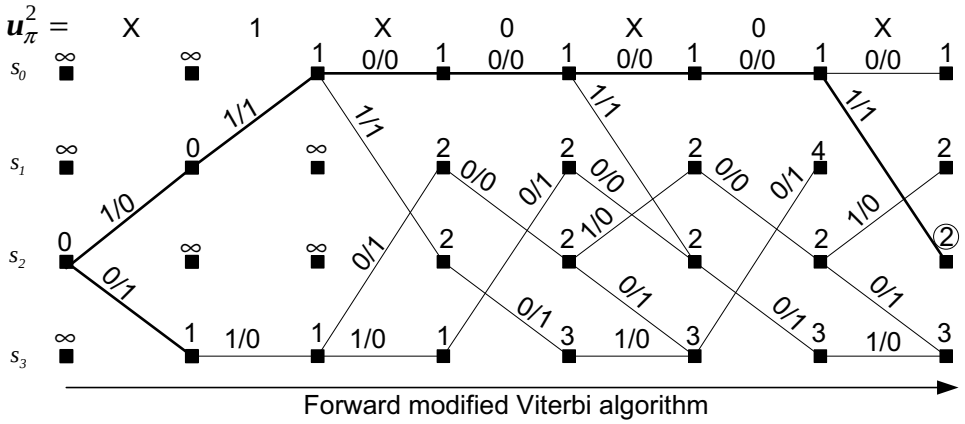
Noting that the path leading to $\mathrm{MWE2}^2_{\mathrm{umts}}$ in Fig. 4.3(b) starts and ends in the state s_0, it follows that $\mathrm{MWE2}^2_{dt}$ for dual-termination is also 2.

Computation of $\mathrm{MWE1}^j$ for tail-biting

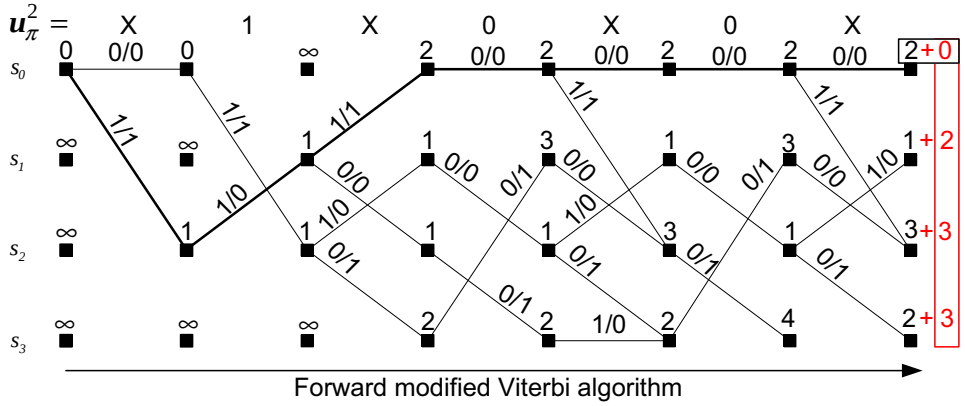
The computation of $\mathrm{MWE1}^j_{tb}$ for tail-biting turbo codes consists of three parts:

- The output-weight $\mathrm{ENC1}(\boldsymbol{u}^{\leq j})$ resulting from encoding the known input sequence $\boldsymbol{u}^{\leq j}$ with ENC1. ENC1 starts encoding $\boldsymbol{u}^{\leq j}$ at time $t = 0$ in state s_x and ends encoding at time $t = j$ in state s_y. The computation of $\mathrm{ENC1}(\boldsymbol{u}^{\leq j})$ and the determination of s_y is straightforward.

- The minimum output-weight $\mathrm{ENC1}(\boldsymbol{u}^{>j})$ resulting from encoding the unknown input sequence $\boldsymbol{u}^{>j}$ with ENC1, which starts encoding $\boldsymbol{u}^{>j}$ at time $t = j$ in the state s_y and



(a) Computing $\text{MWE2}_{tb}^2 = 2$ for tail-biting by applying forward MVA. Branches are labelled with weight of input/parity.



(b) Computing $\text{MWE2}_{\text{umts}}^2 = 2$ for UMTS-termination by applying forward MVA. Branches are labelled with weight of input/parity.

Fig. 4.3 Examples showing how to compute MWE2_{tb}^2 and $\text{MWE2}_{\text{umts}}^2$ for tail-biting and UMTS-termination, respectively. For tail-biting, ENC2 starts and ends in state s_2. For UMTS-termination, ENC2 starts and ends in s_0. The input sequence is $\boldsymbol{u}^2 = (0, 1, 0, \times, \times, \times, \times)$ and the interleaver is $\pi = (3, 1, 6, 0, 5, 2, 4)$.

ends encoding in the state s_x at time $t = \widetilde{K}$. Applying the *backward* MVA ($t = \widetilde{K}$; initial state=s_x; $w(s_x) = 0$; $w(s \neq s_x) = \infty$; decreasing t by 1 until $t = j$) gives a minimum weight at time $t = j$ for the state s_y. This weight is the needed minimum output-weight ENC1($\boldsymbol{u}^{>j}$).

- The $\mathrm{MWE1}^j_{tb}$ is the sum of ENC1($\boldsymbol{u}^{\leq j}$) and ENC1($\boldsymbol{u}^{>j}$).

This procedure is repeated for all starting states of ENC1 and then the minimum is selected to be $\mathrm{MWE1}^j_{tb}$.

As discussed before, the distance spectrum of dual-termination is basically a subset of the distance spectrum of tail-biting. The computation of $\mathrm{MWE1}^j_{dt}$ for dual-termination is the same as $\mathrm{MWE1}^j_{tb}$, where the only start and end state s_x allowed is the all-zero state.

Computation of $\mathrm{MWE1}^j$ for UMTS-termination

The computation of $\mathrm{MWE1}^j_{\mathrm{umts}}$ for UMTS-terminated turbo codes is in principle the same as the one for tail-biting, assuming ENC1 starts encoding $\boldsymbol{u}^{\leq j}$ in the all-zero state. The only difference is the initialization of the backward MVA for the computation of ENC1($\boldsymbol{u}^{>j}$). The backward MVA should be initialized at time $t = \widetilde{K}$ with the extra weights $w_{\mathrm{umts}}(s)$ resulting from the termination-bits.

Example showing how to compute $\mathrm{MWE1}^j$ for tail-biting and UMTS-termination

To show how to compute $\mathrm{MWE1}^j_{tb}$ and $\mathrm{MWE1}^j_{\mathrm{umts}}$, consider the same 4-state single-binary turbo code and the input sequence $\boldsymbol{u}^2$ used earlier. Assume for tail-biting that ENC1 starts and ends in the state s_1. For UMTS-termination ENC1 starts in the all-zero state and is forced back to the all-zero state at the end of encoding the original input sequence.

To obtain $\mathrm{MWE1}^2_{tb}$, ENC1 starts encoding $\boldsymbol{u}^{\leq 2}$ in state s_1 and ends encoding in state s_2 with output-weight of 1. The minimum weight resulting from $\boldsymbol{u}^{>2}$ is obtained by applying the backward MVA with the initialization ($w(s_1) = 0$; $w(s \neq s_1) = \infty$), which leads to an output-weight of 3 at state s_2 and therefore $\mathrm{MWE1}^2_{tb} = 1 + 3 = 4$ (Fig. 4.4(a)). Consequently, $\mathrm{MW}^2_{tb} = \mathrm{MWE1}^2_{tb} + \mathrm{MWE2}^2_{tb} = 4 + 2 = 6$.

For the computation of $\mathrm{MWE1}^2_{\mathrm{umts}}$, ENC1 starts encoding $\boldsymbol{u}^{\leq 2}$ in state s_0 and ends encoding in state s_3 with output-weight of 3. The minimum weight resulting from $\boldsymbol{u}^{>2}$ is obtained by applying the backward MVA with the initialization $w(s) = w_{\mathrm{umts}}(s) =$

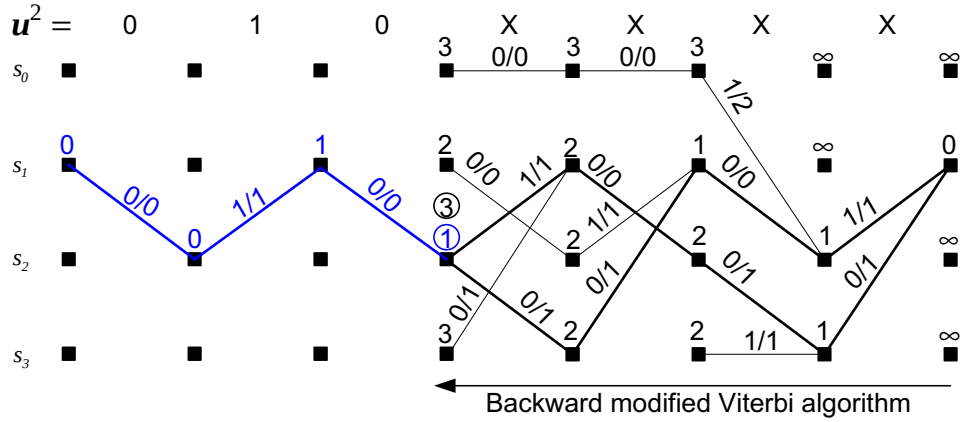
$(0,2,3,3)$, which leads to an output-weight of 3 at state s_3 and therefore $\mathrm{MWE1}^2_{\mathrm{umts}} = 3+3 = 6$ (Fig. 4.4(b)). Consequently, $\mathrm{MW}^2_{\mathrm{umts}} = \mathrm{MWE1}^2_{\mathrm{umts}} + \mathrm{MWE2}^2_{\mathrm{umts}} = 6+2 = 8$.

Noting that the path leading to $\mathrm{MWE1}^2_{\mathrm{umts}}$ in Fig. 4.4(b) starts and ends in the state s_0, it follows that $\mathrm{MWE1}^2_{dt}$ for dual-termination is also 6. Consequently, $\mathrm{MW}^2_{dt} = \mathrm{MWE1}^2_{dt} + \mathrm{MWE2}^2_{dt} = 6+2 = 8$.

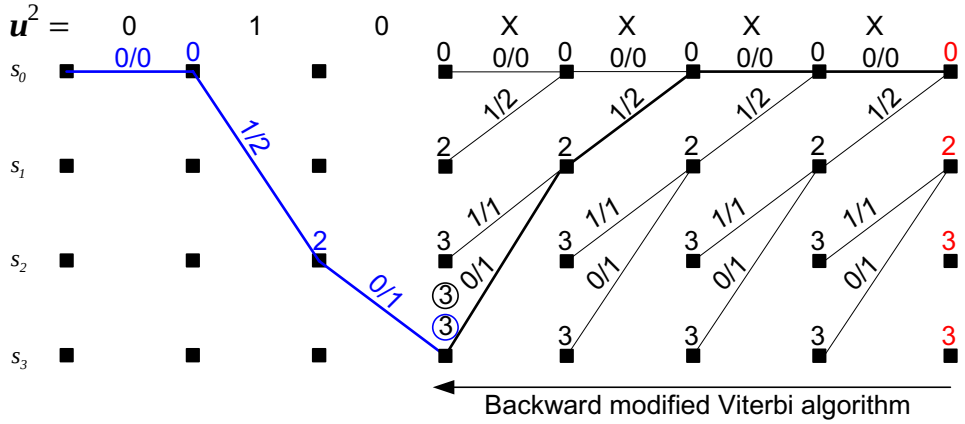
4.2.3 Recursive Construction of Minimum Distance

The search for $d_{\min}$ consists of an iterative construction of input sequences that generate codewords of weight $d_{\min}$. Assume d^* is an upper bound for $d_{\min}$ and IW^* is the maximum allowed input-weight that can cause $d_{\min}$. Any input sequence $\boldsymbol{u} = \boldsymbol{u}^j$ fulfilling the criteria (weight of $\boldsymbol{u}^j$) $\leq IW^*$ and $\mathrm{MW}^j \leq d^*$ could generate $d_{\min}$ and will be the basis for the next iteration $\boldsymbol{u}^{j+1}_{0/1}$, where $\boldsymbol{u}^{j+1}_{0/1} = (\boldsymbol{u}^{\leq j}, 0/1, \times, \cdots, \times)$. If both $\boldsymbol{u}^{j+1}_0$ and $\boldsymbol{u}^{j+1}_1$ fulfill the criteria, then keep $\boldsymbol{u}^{j+1}_1$ to be used as a basis for another iteration later and set $\boldsymbol{u} = \boldsymbol{u}^{j+1}_0$ as the current basis for the next iteration $\boldsymbol{u}^{j+2}_{0/1} = (\boldsymbol{u}^{\leq j+1}, 0/1, \times, \cdots, \times)$. If only $\boldsymbol{u}^{j+1}_b = \boldsymbol{u}^{j+1}_0$ or $\boldsymbol{u}^{j+1}_b = \boldsymbol{u}^{j+1}_1$ fulfill the criteria, then set $\boldsymbol{u} = \boldsymbol{u}^{j+1}_b$ as the current basis for the next iteration $\boldsymbol{u}^{j+2}_{0/1}$ and there is no need to keep $\boldsymbol{u}^{j+1}_{\overline{b}}$ ($\overline{b}$ denotes the complement of bit b). The iterations continue until $\boldsymbol{u}^{\widetilde{K}-1}_{0/1} = (\boldsymbol{u}^{\leq \widetilde{K}-2}, 0/1)$. If the $\mathrm{MW}^{\widetilde{K}-1}$ resulting from $\boldsymbol{u}^{\widetilde{K}}_0$ or $\boldsymbol{u}^{\widetilde{K}}_1$ is lower than d^*, then set d^* to $\mathrm{MW}^{\widetilde{K}-1}$. This leads to a reduction in the number of bases to be considered in subsequent iterations and thus fewer input sequences need to be tested, leading to reduced computational complexity. To make sure that all possible $(2^{\widetilde{K}}-1)$ non-zero input sequences are tested, the following input sequences $\boldsymbol{u}^0 = (1, \times, \cdots, \times)$, $\boldsymbol{u}^1 = (0, 1, \times, \cdots, \times)$, $\boldsymbol{u}^2 = (0, 0, 1, \times, \cdots, \times), \cdots,$ $\boldsymbol{u}^{\widetilde{K}-2} = (0, \cdots, 0, 1, \times)$, $\boldsymbol{u}^{\widetilde{K}-1} = (0, \cdots, 0, 0, 1)$ must be used as bases for subsequent iterations. The unique offsets corresponding to $\boldsymbol{u}^{\leq 0} = (1)$, $\boldsymbol{u}^{\leq 1} = (0, 1)$, $\boldsymbol{u}^{\leq 2} = (0, 0, 1), \cdots, \boldsymbol{u}^{\leq \widetilde{K}-2} = (0, \cdots, 0, 1)$, $\boldsymbol{u}^{\leq \widetilde{K}-1} = (0, \cdots, 0, 0, 1)$ must be used to guarantee the exact values of distance d, multiplicity A_d and information bit multiplicity W_d.

To find the exact distance spectrum for turbo codes using UMTS-terminated [44] it is enough to consider the single case, where both encoders are initialized in the all-zero state and driven back to the all-zero state at the end of encoding the original input sequence. To find the exact distance spectrum for tail-biting turbo codes [56, 57, 58] the algorithm is run $(\Delta_1 \cdot \Delta_2)$ times, where Δ_1 and Δ_2 are the number of states of ENC1 and ENC2, respectively. To find the exact distance spectrum for dual-terminated turbo codes [55], the



(a) Computing $\mathrm{MWE1}^2_{tb} = \mathrm{ENC1}(\boldsymbol{u}^{\leq 2}) + \mathrm{ENC1}(\boldsymbol{u}^{>2}) = 1 + 3 = 4$ for tail-biting by applying backward MVA. Branches are labelled with weight of input/(input+parity).



(b) Computing $\mathrm{MWE1}^2_{\mathrm{umts}} = \mathrm{ENC1}(\boldsymbol{u}^{\leq 2}) + \mathrm{ENC1}(\boldsymbol{u}^{>2}) = 3 + 3 = 6$ for UMTS-termination by applying backward MVA. Branches are labelled with weight of input/(input+parity).

Fig. 4.4 Examples showing how to compute $\mathrm{MWE1}^2_{tb}$ and $\mathrm{MWE1}^2_{\mathrm{umts}}$ for tail-biting and UMTS-termination, respectively. For tail-biting, ENC1 starts and ends in state s_1. For UMTS-termination, ENC1 starts and ends in s_0. The input sequence is $\boldsymbol{u}^2 = (0, 1, 0, \times, \times, \times, \times)$ and the interleaver is $\pi = (3, 1, 6, 0, 5, 2, 4)$.

algorithm is run only one time, where both encoders start and end in the all-zero state. It has been observed for non-punctured tail-biting turbo codes that most of the computation is used in finding $d_{\min}$ for the case where both encoders are assumed to start in the all-zero state. This is because the number of surviving bases to be tested is significantly higher than that for other starting state combinations. Thus, the computational complexity with tail-biting is typically not much more than with dual-termination.

4.2.4 Techniques to Reduce the Computational Complexity and Memory Requirement

Some techniques that are used by Garello to reduce the computational complexity and memory requirements are discussed. A new technique that further reduces the computational complexity is also presented.

Garello's modified definitions for WE1 and WE2

Garello's modified definitions for WE1 and WE2 are explained. The computational complexity of the distance measurement method presented here depends strongly on the number of bases that must be considered for later testing. This number in turn depends on the value of MW^j for each basis $\boldsymbol{u}^j, j \in \{0, \cdots, \widetilde{K}-1\}$. The aim now is to increase the value of MW^j, because the higher MW^j is, the lower the number of bases that must be considered and thus the fewer the input sequences that must be tested.

Note that any single-binary recursive convolutional code with memory δ can be driven from any state s_x to any state s_y by an input sequence of length and weight less than or equal to δ. Given $\boldsymbol{u}^j$, ENC1 is guaranteed to be driven into the all-zero state with δ input bits. Also, considering the weight of the systematic bits resulting from the $(\widetilde{K}-j-1)$ unknown consecutive input bits in $\mathrm{MWE1}^j$ will not bring any significant weight. It is better to consider their weight in $\mathrm{MWE2}^j$, because the $(\widetilde{K}-j-1)$ unknown consecutive bits get scattered and enter ENC2 in non-consecutive order leading to higher $\mathrm{MWE2}^j$. The modified definition for the weight of ENC1 is

$$\overline{\mathrm{WE1}} = \left[\begin{array}{l} \text{-Weight of both systematic and parity of ENC1 for known input bits, plus} \\ \text{-Weight of only parity of ENC1 for unknown input bits.} \end{array} \right. \tag{4.8}$$

This modified definition ($\overline{\text{WE1}}$) leads to a lower weight for $\overline{\text{MWE1}}^j$ compared to MWE1^j from the previous definition (WE1), i.e.,

$$\overline{\text{MWE1}}^j \leq \text{MWE1}^j. \tag{4.9}$$

The modified definition for the weight of ENC2 is

$$\overline{\text{WE2}} = \left[\begin{array}{l} \text{-Weight of only parity of ENC2 for known input bits, plus} \\ \text{-Weight of both systematic and parity of ENC2 for unknown input bits.} \end{array} \right. \tag{4.10}$$

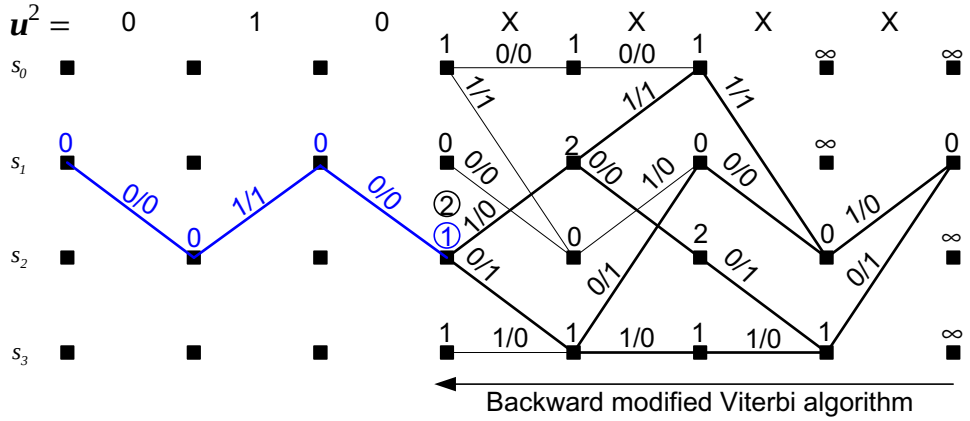
On average, the modified definition ($\overline{\text{WE2}}$) leads to significantly higher weight for $\overline{\text{MWE2}}^j$ compared to MWE2^j in the previous definition (WE2), i.e.,

$$\overline{\text{MWE2}}^j >> \text{MWE2}^j. \tag{4.11}$$

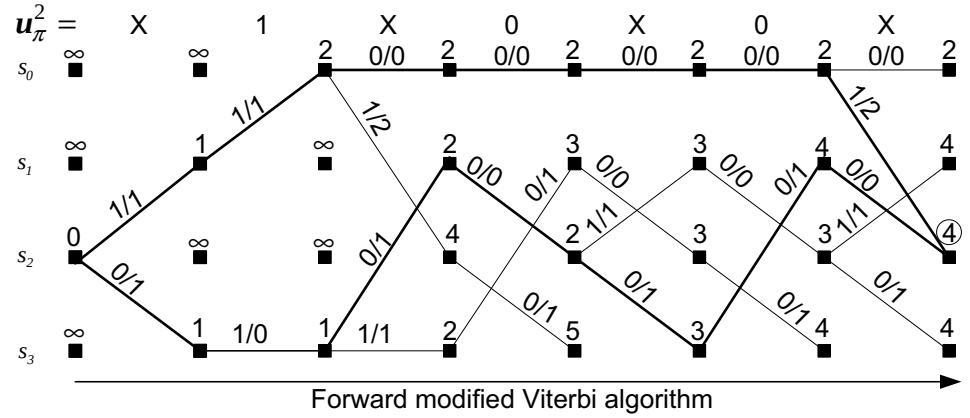
On average, the lost weight in (4.9) is more than compensated for by the gained weight in (4.11). The effect of this compensation becomes especially clear if the number of unknown bits ($\widetilde{K}-j-1$) in $\boldsymbol{u}^j$ is significantly larger than δ for most j (*i.e.*, ($\widetilde{K}-j-1$) $>> \delta$), which is the case for all interleavers of practical length. On average, the modified definitions increase the minimum weight $\overline{\text{MW}}^j = \overline{\text{MWE1}}^j + \overline{\text{MWE2}}^j$ resulting from the sum of minimum weights of ENC1 and ENC2. This eliminates the test of many input sequences, which in turn significantly reduces the computational complexity. That is,

$$\overline{\text{MW}}^j \geq \text{MW}^j. \tag{4.12}$$

To show the effect of these definitions for tail-biting, lets apply them on the same 4-state turbo code used earlier using the same basis input sequence $\boldsymbol{u}^2 = (0, 1, 0, \times, \times, \times, \times)$. As shown in Fig. 4.5(a), $\overline{\text{MWE1}}^2_{tb} = \text{ENC1}(\boldsymbol{u}^{\leq 2}) + \text{ENC1}(\boldsymbol{u}^{>2}) = 1 + 2 = 3$ is less than the MWE1^2_{tb} of 4 (see Fig. 4.4(a)). However, the $\overline{\text{MWE2}}^2_{tb}$ is 4 (see Fig. 4.5(b)), which is higher than the MWE2^2_{tb} of 2 (see Fig. 4.3(a)). This example shows that even for a very short interleaver ($\widetilde{K} = 7$), $\overline{\text{MW}}^2_{tb} = 7$ is obtained using the new definitions instead of $\text{MW}^2_{tb} = 6$ obtained with the previous definitions. Similarly, an example can be shown for UMTS-termination. The increase in $\overline{\text{MW}}^j$ compared to MW^j is even more significant for longer interleavers.



(a) Computing $\overline{\text{MWE1}}_{tb}^{2} = \text{ENC1}(\boldsymbol{u}^{\leq 2}) + \text{ENC1}(\boldsymbol{u}^{>2}) = 1 + 2 = 3$ by applying backward MVA. Branches are labelled with weight of input/(input+parity) for known bits and with weight of input/parity for unknown bits.



(b) Computing $\overline{\text{MWE2}}_{tb}^{2} = 4$ by applying forward MVA. Branches are labelled with weight of input/parity for known bits and with weight of input/(input+parity) for unknown bits.

Fig. 4.5 Example showing how to use the modified definition to compute $\overline{\text{MWE1}}_{tb}^{2}$ and $\overline{\text{MWE2}}_{tb}^{2}$ for $\boldsymbol{u}^2 = (0, 1, 0, \times, \times, \times, \times)$ and the interleaver $\pi = (3, 1, 6, 0, 5, 2, 4)$.

Memory reduction

This technique was introduced by Garello to achieve the following two goals:

- Reduce the memory required to keep the new bases.
- Avoid unnecessary repeated computation of the value ENC1($\boldsymbol{u}^{\leq j}$).

For all possible bases $\boldsymbol{u}^0, \boldsymbol{u}^1, \cdots, \boldsymbol{u}^{\widetilde{K}-2}, \boldsymbol{u}^{\widetilde{K}-1}$ the memory required without the memory reduction technique is:

- $\widetilde{K}$ bits to save the $\boldsymbol{u}^{\leq j}$ known bits.
- Buffers of sizes $(2, 3, \cdots, \widetilde{K}-1)$ bits to save the new bases. This results in a memory requirement of $\sum_{i=2}^{\widetilde{K}-1} i = \frac{(\widetilde{K}-2)(\widetilde{K}+1)}{2}$ bits.

The overall memory required in bits is:

$$M = \widetilde{K} + \frac{(\widetilde{K}-2)(\widetilde{K}+1)}{2} \tag{4.13}$$

To lower the memory requirement during the recursive construction of $d_{\min}$, consider all possible next iterations $\boldsymbol{u}_1^{t=j+1}, \boldsymbol{u}_1^{t=j+2}, \cdots, \boldsymbol{u}_1^{t=\widetilde{K}-1}$ as branches of a binary tree. Each branch B_t is labelled with four variables (t, I_t, W_t, s_{t+1}) where I_t is the input-weight at time t resulting from $\boldsymbol{u}^{\leq t}$ (the $t+1$ known bits), W_t is the output-weight (i.e., weight of input and parity) of ENC1 generated by the input sequence $\boldsymbol{u}^{\leq t}$ and s_{t+1} is the state of ENC1 after encoding $\boldsymbol{u}^{\leq t}$. For each basis input sequence $\boldsymbol{u}_1^{t=j+1}, \boldsymbol{u}_1^{t=j+2}, \cdots, \boldsymbol{u}_1^{t=\widetilde{K}-1}$ fulfilling the criteria (weight of $\boldsymbol{u}^t$) $\leq IW^*$ and $\overline{\mathrm{MW}}^t \leq d^*$, instead of saving it, it is enough to label its corresponding branch B_t with the four values (t, I_t, W_t, s_{t+1}).

To demonstrate the labelling strategy, consider the same 4-state single-binary turbo code described earlier, where ENC1 starts and ends in state s_1 and $\boldsymbol{u}^2 = (0, 1, 0, \times, \cdots, \times)$ is the basis input sequence. A cross section of the binary tree is shown in Fig. 4.6.

After the last iteration $\boldsymbol{u}_{0/1}^{t=\widetilde{K}-1}$ is done, where all unknown input bits become known, trace back the last saved branch $B_{t'}$ and extract from it the new basis. The new basis for the next iteration can be extracted easily by using the first t' bits from the known $\boldsymbol{u}_{0/1}^{t=\widetilde{K}-1}$ and concatenating them with the input sequence $(1, \times, \cdots, \times)$, which gives

$$\boldsymbol{u}^{j=t'} = (\boldsymbol{u}^{\leq (t'-1)}, 1, \times, \cdots, \times) \tag{4.14}$$

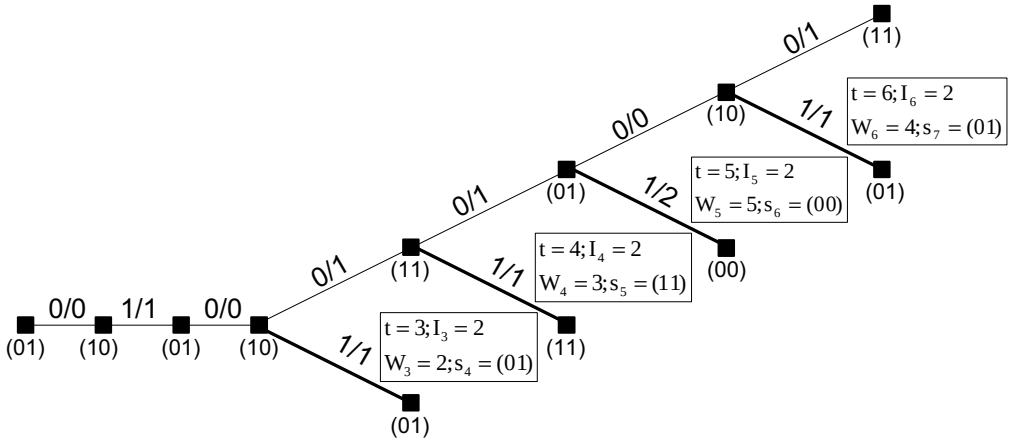


Fig. 4.6 Cross section of a labelled binary tree for ENC1, where ENC1 starts in state s_1 and the basis input sequence is $\boldsymbol{u}^2 = (0, 1, 0, \times, \cdots, \times)$. Branches are labelled with weight of input/(input+parity). Each node of the binary tree represents the current state of the ENC1. The branches corresponding to the input bit '1' fulfilling the criteria (weight of $\boldsymbol{u}^t$) $\leq IW^*$ and $\overline{\mathrm{MW}}^t \leq d^*$ are the thick branches labelled with (t, I_t, W_t, s_{t+1}).

One advantage of using a labelled binary tree is the fact that only a single buffer of $\widetilde{K}$ bits is needed for all basis input sequences $\boldsymbol{u}^0, \boldsymbol{u}^1, \cdots, \boldsymbol{u}^{\widetilde{K}-1}$ instead of the M bits as previously mentioned in (4.13). Of course, labelling the needed $(\widetilde{K}-2)$ branches requires an extra $3(\widetilde{K}-2)+2(\widetilde{K}-2) = 5(\widetilde{K}-2)$ bytes (assuming that time index t needs 2 bytes and the other 3 labels require only 1 byte). The labelling technique requires a memory of about $5\widetilde{K}$ bytes, but this is still negligible compared to the previously required M bits in (4.13). The saving is significant even for medium sized interleavers.

Recall that the main objective is to compute $\overline{\mathrm{MW}}^j = \overline{\mathrm{MWE1}}^j + \overline{\mathrm{MWE2}}^j$, where $\overline{\mathrm{MWE1}}^j = \mathrm{ENC1}(\boldsymbol{u}^{\leq j}) + \mathrm{ENC1}(\boldsymbol{u}^{>j})$. Another advantage of using a labelled binary tree is its lower computational complexity, due to the fact that it is enough to compute $\overline{\mathrm{MWE2}}^j$ to obtain the needed $\overline{\mathrm{MW}}^j$, because $\mathrm{ENC1}(\boldsymbol{u}^{\leq j})$ is given by the value of $W_{t'=j}$ labelling the branch $B_{t'=j}$ and $\mathrm{ENC1}(\boldsymbol{u}^{>j})$ at state $s_{t'+1=j+1}$ is already computed offline as suggested earlier.

Other useful techniques to reduce complexity

The following techniques have also been used to reduce the computational complexity in Garello's distance measurement routine:

1. Finding a tight upper bound for $d_{\min}$ at the beginning of the iterative process reduces the number of input sequences to be tested and thus lowers the computational complexity. Hence, use $\boldsymbol{u}^{j=\widetilde{K}-1} = (0, \cdots, 0, 0, 1)$, $\boldsymbol{u}^{j=\widetilde{K}-2} = (0, \cdots, 0, 1, \times)$, $\boldsymbol{u}^{j=\widetilde{K}-3} = (0, \cdots, 0, 1, \times, \times), \cdots, \boldsymbol{u}^{j=0} = (1, \times, \cdots, \times)$ successively as the basis for the next iteration $\boldsymbol{u}_{0/1}^{j+1}$, where $\times$ can be 0 or 1. This tends to quickly lower the upper bound d^* because the first bases tested will have large numbers of consecutive leading zero bits and very few trailing unknown bits.

2. To reduce the computational complexity of $\overline{\text{MWE1}}^j$ for all $j \in \{\widetilde{K}-1, \widetilde{K}-2, \cdots, 0\}$, ENC1($\mathbf{u}^{>j} = (\times, \cdots, \times)$) can be computed *offline* for all states and all trellis sections $\{\widetilde{K}-1, \widetilde{K}-2, \cdots, 0\}$ by applying the backward MVA. Similarly, the value ENC2($\mathbf{u}^{>j} = (\times, \cdots, \times)$) for all $j \in \{\widetilde{K}-1, \widetilde{K}-2, \cdots, 0\}$ can also be computed *offline*. The values of ENC2($\mathbf{u}^{>j} = (\times, \cdots, \times)$) are used to stop earlier the computation of $\overline{\text{MWE2}}^j$.

3. The $\overline{\text{MWE2}}_0^j$ resulting from the input sequence $\boldsymbol{u}_0^j$ and the $\overline{\text{MWE2}}_1^j$ resulting from the input sequence $\boldsymbol{u}_1^j$ can be computed separately by applying the forward MVA, but this leads to unnecessarily repeated computation over trellis sections $(0, \cdots, \pi^{-1}(j)-1)$ and $(\pi^{-1}(j)+1, \cdots, \widetilde{K}-1)$. To lower the computational complexity, $\overline{\text{MWE2}}_0^j$ and $\overline{\text{MWE2}}_1^j$ can be computed in a single run by applying (see Fig. 4.7):

a. Forward MVA over trellis sections $0, \cdots, \pi^{-1}(j)-1$.

b. Backward MVA over trellis sections $\widetilde{K}-1, \cdots, \pi^{-1}(j)+1$.

c. Combining the results of (a) and (b) at trellis section $t = \pi^{-1}(j)$ for case $\times = 0$ to get $\overline{\text{MWE2}}_0^j$ and for case $\times = 1$ to get $\overline{\text{MWE2}}_1^j$.

4. The forward MVA used to determine $\overline{\text{MWE2}}^j$ can be stopped earlier at the end of the trellis section ES_LEFT if the minimum weight resulting from the summation of (a) current weights at end of trellis section ES_LEFT in ENC2, and (b) the ENC2($\mathbf{u}^{>\text{ES\_LEFT}}$) weights computed offline and (c) $\overline{\text{MWE1}}^j$ is higher than d^* (see Fig. 4.8).

Similarly, the backward MVA used to determine $\overline{\text{MWE2}}^j$ can be stopped earlier at the beginning of the trellis section ES_RIGHT if the minimum weight resulting from the summation of (a) current weights at beginning of trellis section ES_RIGHT in ENC2, and (b) the weights ENC2($\mathbf{u}^{\leq\text{ES\_RIGHT}}$) and (c) $\overline{\text{MWE1}}^j$ is higher than d^* (see Fig. 4.8).

It has been observed that the combination of an early stopping at trellis section ES_LEFT together with an early stopping at trellis section ES_RIGHT does not lower the average

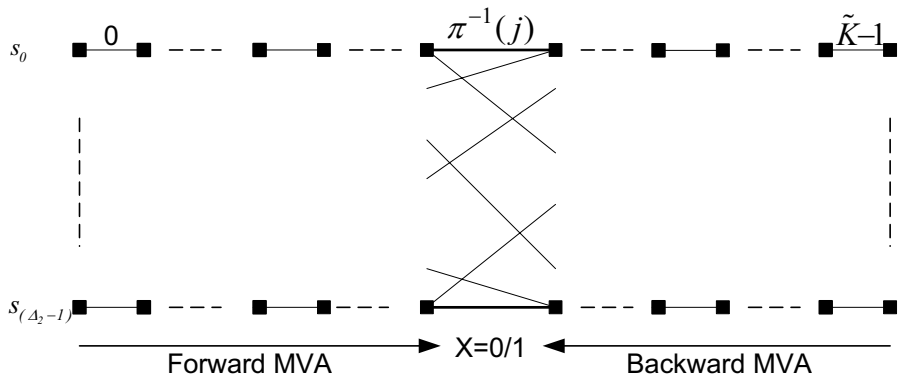


Fig. 4.7 This Figure illustrate how to obtain $\overline{\text{MWE2}}_0^j$ and $\overline{\text{MWE2}}_1^j$. Applying (a) forward MVA and (b) backward MVA, then combining the results of (a) and (b)at trellis section $\pi^{-1}(j)$ for input bits 0 and 1 leads to $\overline{\text{MWE2}}_0^j$ and $\overline{\text{MWE2}}_1^j$, respectively.

computational complexity. In fact, the computational complexity is the same for interleavers of lengthes $\widetilde{K} \leq 200$ and is significantly higher for longer interleavers. Thus, it is recommended to use early stopping only at trellis section ES_LEFT or ES_RIGHT, but not both at the same time. In this thesis, the computation of $\overline{\text{MWE2}}^j$ is stopped earlier only at the end of trellis section ES_LEFT.

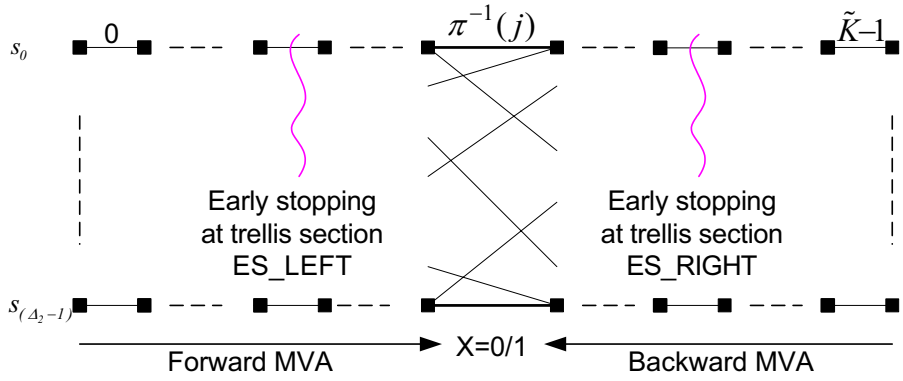


Fig. 4.8 The forward and backward MVA can be stopped early at trellis sections ES_LEFT and ES_RIGHT, respectively. This thesis uses early stopping only at trellis sections ES_LEFT.

A new technique to reduce complexity

The complexity involved with the determination of $\overline{\mathrm{MW}}_{0/1}^{j}$ is mainly due to the computation of $\overline{\mathrm{MWE2}}_{0/1}^{j}$. The new technique proposed here reduces the complexity resulting from the determination of $\overline{\mathrm{MWE2}}_{0/1}^{j}$ by avoiding unnecessary re-computation of forward and backward MVA.

During the computation of $\overline{\mathrm{MWE2}}_{0/1}^{j}$ the forward MVA starts at trellis section LEFT=0 and ends at trellis section ES_LEFT or $(\pi^{-1}(j)-1)$ and the backward MVA starts at trellis section RIGHT=($\widetilde{K}$-1) and ends at trellis section $(\pi^{-1}(j)+1)$. The computation of $\overline{\mathrm{MWE2}}_{0/1}^{j+1}$ will also use LEFT=0 and RIGHT=($\widetilde{K}$-1), which leads to a re-computation of backward and forward MVA over many trellis sections. To avoid this re-computation, the following strategy is proposed:

- The computation of $\overline{\mathrm{MWE2}}_{0/1}^{j}$ for all bases $\boldsymbol{u}^{j=\widetilde{K}-2} = (0,\cdots,0,1,\times)$, $\boldsymbol{u}^{j=\widetilde{K}-3} = (0,\cdots,0,1,\times,\times),\cdots,\boldsymbol{u}^{j=0} = (1,\times,\cdots,\times)$ and (bases kept from previous iterations) is done by setting LEFT=0 and RIGHT=($\widetilde{K}$-1). The computed weights for all states at begin end end of each trellis section must be stored in a $(\widetilde{K}+1)-by-\Delta_2$ matrix (i.e., MVA_MATRIX[$\widetilde{K}$+1][Δ_2]) .

- The computation of $\overline{\mathrm{MWE2}}_{0/1}^{j+1}$ uses the new values for LEFT and RIGHT, which depend on the positions $\pi^{-1}(j)$ and ES_LEFT from the previous iteration and the current position $\pi^{-1}(j+1)$ and can be obtained using the following simple structure:

  - LEFT=min$(\pi^{-1}(j+1),\pi^{-1}(j),ES_LEFT)$.
  - RIGHT=max$(\pi^{-1}(j+1),\pi^{-1}(j))$.

  The forward and backward MVA are initialized with MVA_MATRIX[LEFT] at trellis section LEFT and MVA_MATRIX[RIGHT+1] at trellis section RIGHT+1, respectively. The values of the MVA_MATRIX must be updated for the trellis sections (LEFT,$\cdots$,RIGHT) during the computation of $\overline{\mathrm{MWE2}}_{0/1}^{j+1}$, so they can be used for the computation of $\overline{\mathrm{MWE2}}_{0/1}^{j+2}$, if needed.

Simulation results with DRP interleavers [26, 100, 60] of various lengths show that this strategy reduces the average computational complexity by about a factor of 2.

4.2.5 Comparison Between Theory and Simulation Results

For comparison purposes, simulation results for a random interleaver were compared to the analytical union bounds given in (4.1) and (4.2) that used the first and the first five terms of the distance spectrum listed in Table 4.1. TUB(1) and TUB(5) refer to truncated union bound that uses the first and the first five terms of the distance spectrum, respectively. As expected, Fig. 4.9 shows that the simulation results agree with the analytical bounds in the flare region (i.e., at high SNR values). Furthermore, Fig. 4.9 shows that increasing the number of distance spectrum terms used in the union bound reduces the gap between the simulation results and the analytical bounds. Thus, it is desirable to use as many terms as possible when the error performance is predicted based on the analytical union bound. This in turn shows the importance of efficient distance measurement methods, capable of computing high distances in reasonable times.

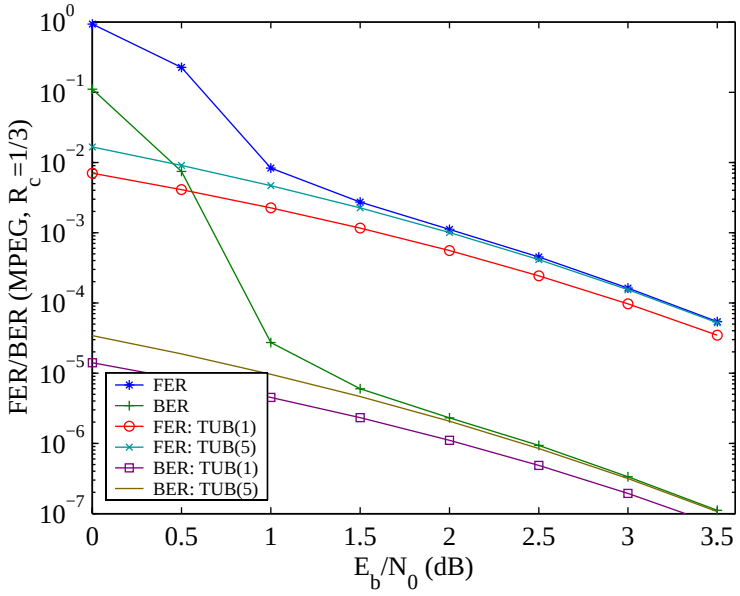


Fig. 4.9 FER and BER for MPEG-sized random interleaver of code rate R_c=1/3 (QPSK/AWGN). The size of overlap is 75 symbols (150 bits) and the number of iterations is 8. Enhanced max-log MAP (EML-MAP) decoding with an extrinsic scale factor of 0.75 was used by the DVB-RCS turbo decoder. 1000 MPEG packet errors were counted at each simulated SNR value. The TUB(1) and TUB(5) curves are the truncated union bounds using the first and the first five terms of the distance spectrum, respectively.

Another way to check the results is to determine the distance between the transmitted

Table 4.1 Distance results ($d_{\min}/A_{\mathrm{d}_{\min}}/W_{\mathrm{d}_{\min}}$), and the next four distance terms, for a $\widetilde{K}$=1504 bit random interleaver with code rate $R_c = 1/3$.

$\widetilde{K}$	first term	second term	third term	forth term	fifth term
1504	12/3/9	13/2/6	15/4/15	16/3/8	17/4/12

codeword and the decoded codeword in the simulation when a packet is in error. Obviously, this distance provides an upper bound on the true $d_{\min}$. At high SNR values, this upper bound is usually very tight and often is equal to the true $d_{\min}$, especially when $d_{\min}$ is not very high. The minimum distance obtained from the simulation, with the random interleaver, did agree with the true $d_{\min}$ obtained algorithmically. This provides a good indication that the method is working properly.

Note that both methods described above were used only for comparison purposes and they do not verify the computed minimum distance. They just increase the confidence that the method was implemented correctly.

4.2.6 Distance Results

The new interleaver design for the DVB-RCS [15] turbo codes is based on the dithered relative prime (DRP) approach [26, 100, 60]. DRP interleavers are highly structured and ideal for designing low-memory interleaver banks for turbo codes. Each interleaver can be stored and implemented using only a few parameters. These parameters can be computed at run-time if desired. The interleaver bank resolution is determined by the dither window size, M. An M value of 4 works well for blocks of length $\widetilde{K} < 400$ and an M value of 8 is better for blocks of length $400 \leq \widetilde{K} \leq 1500$. An M value of 16 or higher is recommended for larger blocks.

Distance results are shown for some UMTS standard interleavers and all DVB-RCS standard interleavers and code rates. The code rate is R_c and the 3 values ($d_{\min}/A_{\mathrm{d}_{\min}}/W_{\mathrm{d}_{\min}}$) represent minimum distance, codeword multiplicity and information bit multiplicity, respectively.

UMTS/3GPP results

Table[1] 4.2 shows that the distance results for the UMTS standard interleavers barely improve with tail-biting. This is because the lowest distances are usually caused by the interleaver itself, and not the termination technique.

Table 4.2 also shows that the distance results for the DRP interleavers are consistently better than those shown for the UMTS standard. This is especially true for tail-biting. Thus, it is recommended to use UMTS encoders and decoders that use tail-biting [56, 57, 58] or dual-termination [55].

Table 4.2 Distance results ($d_{\min}/A_{\mathrm{d}_{\min}}/W_{\mathrm{d}_{\min}}$), and the next two distance terms, for 4 UMTS interleavers and DRP interleavers with code rate $R_c = 1/3$. $\widetilde{K}$ is in bits.

	standard UMTS interleavers		DRP interleavers	
$\widetilde{K}$	UMTS-termination	tail-biting	UMTS-termination	tail-biting
	12/1/4	12/1/4	14/4/8	25/160/640
80	16/3/12	16/4/16	16/1/2	26/160/960
	17/7/23	17/11/33	17/3/7	27/440/2200
	16/1/2	17/2/6	21/4/8	31/80/400
160	17/2/6	20/3/12	23/6/18	32/160/640
	20/4/16	21/10/30	24/4/16	33/480/3040
	24/1/4	24/1/4	24/1/2	38/800/4480
320	25/2/6	25/1/3	25/2/6	39/2160/15920
	26/2/4	26/2/4	26/4/10	40/400/2560
	27/9/81	27/9/81	30/2/6	42/160/960
640	28/160/640	28/162/648	31/2/4	43/320/1920
	29/8/24	29/5/15	32/3/8	44/480/1920

Fig. 4.10 shows, for interleavers of length 512, an improvement with the DRP interleaver greater than 0.5 dB at FERs below 10^{-6} compared to the UMTS standard interleaver. The UMTS standard interleaver has a $d_{\min}$ of 21 with $A_{\mathrm{d}_{\min}}$=1 and $W_{\mathrm{d}_{\min}}$=3 for both UMTS-termination [25] and dual-termination [55]. The DRP interleaver with $M = 8$ has a $d_{\min}$ of 42 with $A_{\mathrm{d}_{\min}}$=410 and $W_{\mathrm{d}_{\min}}$=2460 for dual-termination. Since the minimum distance with

[1]Special thanks to the communications and signal processing group at the Communications Research Centre (CRC) in Ottawa for providing the DRP interleavers used in Table 4.2 and Fig. 4.10 as well as the simulation software used to generate Fig. 4.10. Distance results for tail-biting in Table 4.2 are determined using Garello's extended method proposed in this chapter.

UMTS-termination $d_{\min}$(UMTS) is always less than or equal to the minimum distance with dual-termination, then it is fair to compare the error rate performance of both interleavers based on dual-termination. Note that the constituent decoders used enhanced max-log maximum a posteriori (MAP) decoding with an extrinsic scale factor of 0.8.

The truncated union bounds (TUBs) in Fig. 4.10 used only the first term of the distance spectrums. The gaps between the analytical bounds and the simulated curves are expected to be reduced when more terms of the distance spectrums are used. Unfortunately, the determination of higher terms is very computationally intensive because the $d_{\min}$ value is already quite high.

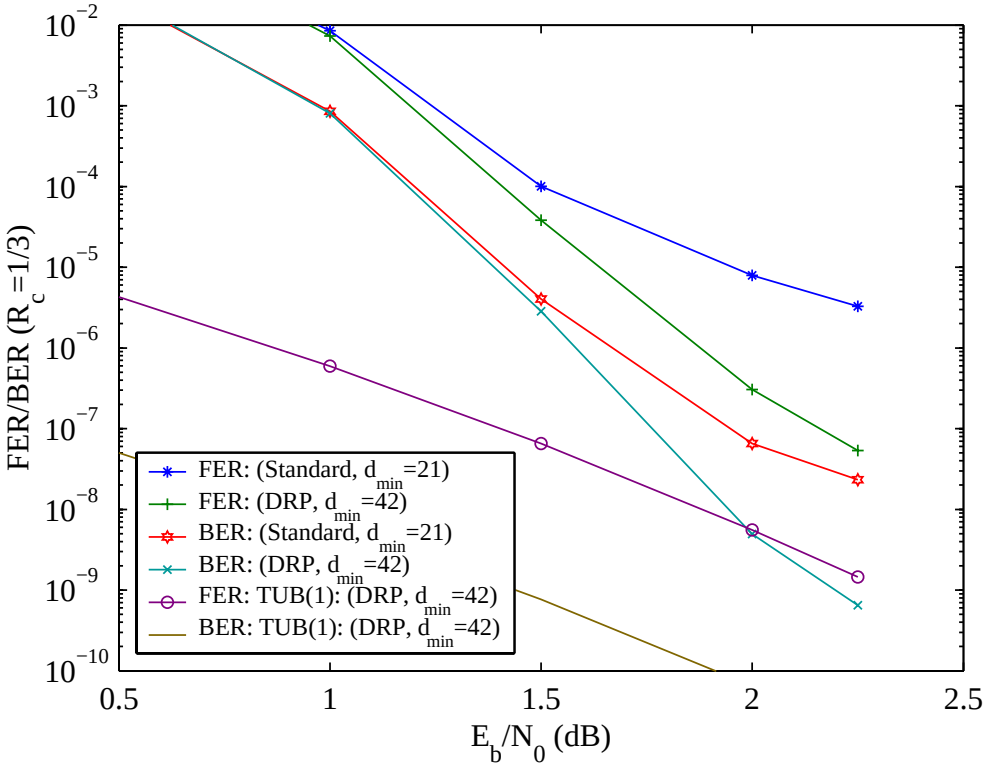


Fig. 4.10 FER and BER for interleavers of size 512 and code rate R_c=1/3 (BPSK/AWGN). The number of iterations is 16. Enhanced max-log MAP (EML-MAP) decoding with an extrinsic scale factor of 0.8 was used by the turbo decoder. 600 million packets were simulated for both interleavers at 2.25 dB. Both interleavers were used with dual-termination. The TUB(1) curves are the truncated union bounds using only the first term of the distance spectrum.

DVB-RCS results

For double-binary turbo codes the bits within the symbols can also be manipulated, for example as per the original DVB-RCS standard. Let $K = \widetilde{K}/2$ be the interleaver length in symbols (2 bits). Since symbols are interleaved, the dither window operates on M symbols ($2M$ bits).

Table 4.3 shows the distance results for the twelve DVB-RCS standard interleavers for the seven standard code rates. Some of the distance results shown in Table 4.3 have also been independently computed by Rosnes, *et al.* [29] using a different modified version of Garello's original algorithm. The results in Table 4.3 agree exactly with the subset of cases considered in [29].

Table 4.3 Distance results ($d_{\min}/A_{\mathrm{d}_{\min}}/W_{\mathrm{d}_{\min}}$) for the 12 DVB-RCS standard interleavers. K is in 2-bit symbols.

K	$R_c = 1/3$	$R_c = 2/5$	$R_c = 1/2$	$R_c = 2/3$	$R_c = 3/4$	$R_c = 4/5$	$R_c = 6/7$
48	21/72/240	17/48/192	13/72/168	8/120/360	4/8/32	4/12/36	3/16/32
64	25/192/1248	18/32/192	14/32/128	8/64/256	5/4/13	4/16/64	3/2/5
212	31/106/954	25/159/1325	18/159/954	11/159/901	7/10/50	6/159/742	4/9/27
220	31/110/990	25/165/1265	19/165/1265	11/220/1210	7/10/35	6/110/550	4/2/8
228	30/114/855	24/57/342	18/171/1197	10/57/342	7/19/57	6/171/798	5/247/836
424	30/212/1696	24/212/1696	18/212/1696	13/530/3710	8/21/84	7/212/954	5/80/287
432	31/108/972	27/324/3132	18/108/972	12/432/2160	8/36/144	6/108/324	5/72/288
440	28/110/1100	22/110/1100	16/110/1100	12/110/440	8/27/108	8/1100/5500	4/10/40
752	33/376/3384	27/376/3384	19/376/3384	12/188/1316	9/27/171	9/3572/20680	6/199/826
848	36/848/7420	28/636/5088	20/636/5088	13/212/1272	9/1/4	8/212/848	5/67/176
856	33/428/3852	27/428/3852	19/428/3852	12/214/1498	9/8/40	9/3210/17762	5/16/64
864	36/864/7560	28/648/5184	20/648/5184	13/216/1296	9/72/144	8/648/2160	6/288/1008

Table 4.4 shows additional distance results for the first 9 standard interleaver sizes for DVB-RCS. These minimum distances were obtained using DRP interleavers that were found with an exhaustive search over all possible dither patterns with $M = 4$ (except for some high code rates for K=752 2-bit symbols the search was not exhaustive). All 4 repeating possibilities of swapping the 2 bits within a symbol over 2 consectutive symbols were also considered. For all the packet sizes and code rates, the minimum distances shown in Table 4.4 are as good or better than those shown in Table 4.3 for the standard interleavers.

Results were also generated for DRP interleavers with M values of 1 and 2 and a code

Table 4.4 Distance results ($d_{\min}/A_{\mathrm{d}_{\min}}/W_{\mathrm{d}_{\min}}$) for DVB-RCS with exhaustive search for DRP interleavers with $M = 4$. Standard puncturing was used. K is in 2-bit symbols.

K	$R_c = 1/3$	$R_c = 2/5$	$R_c = 1/2$	$R_c = 2/3$	$R_c = 3/4$	$R_c = 4/5$	$R_c = 6/7$
48	24/24/144	20/192/960	14/96/288	8/24/60	6/64/224	5/108/420	4/156/520
64	28/384/2528	22/64/384	17/256/1920	9/64/256	6/16/67	5/16/48	3/1/2
212	36/1007/7420	28/53/530	22/1908/14416	12/106/318	8/24/96	8/2173/9858	4/1/4
220	36/715/6380	28/110/880	21/220/1210	12/55/275	8/9/42	8/1815/8305	4/1/3
228	36/342/1710	29/627/3933	22/1995/14649	12/285/1197	9/133/684	8/2337/12084	6/1216/5092
424	36/106/636	30/848/5088	23/530/4770	14/318/2014	9/2/11	9/1802/9434	5/1/4
432	36/108/648	30/864/5184	23/756/7020	14/540/3348	10/72/360	9/2160/10584	6/36/180
440	36/330/2970	30/880/5280	23/880/7260	14/330/2090	9/1/2	9/1760/9130	6/246/1155
752	36/3196/24064	30/1504/9024	22/3760/28388	14/188/1692	10/137/793	10/7332/41924	6/108/479

rate of 1/3. Table 4.5 shows that the (empirical) upper bounds on $d_{\min}$ with M=1,2 and 4 are 28, 32 and 36, respectively. Note that DRP interleavers with $M = 1$ correspond to simple relative prime interleavers. Results for M=1,2 and 4 were obtained with an exhaustive search considering the four repeating possibilities of swapping the 2 bits within a symbol over 2 consecutive symbols. To get a $d_{\min}$ higher than 36, the dither window size must be greater than $M = 4$. Unfortunately, an exhaustive search with $M = 16$ or even $M = 8$ is impossible in a reasonable time due to the very large number of dither patterns to be tested. Thus, the search was limited to randomly selected dither patterns. A $d_{\min}$ of 40 with $A_{\mathrm{d}_{\min}}$=1175 and $W_{\mathrm{d}_{\min}}$=8084 was obtained for MPEG-sized packets (K=752 2-bit symbols) using the DRP approach with $M = 16$, whereas the $d_{\min}$ for the DVB-RCS standard interleaver is just 33.

Table 4.5 Distance results ($d_{\min}$) for rate 1/3 DVB-RCS codes with an exhaustive search for DRP interleavers with M=1, 2 and 4. K is in 2-bit symbols.

M	K=48	K=64	K=212	K=220	K=228	K=424	K=432	K=440	K=752
1	24	27	28	28	28	28	28	28	28
2	24	27	32	32	32	32	32	32	32
4	24	28	36	36	36	36	36	36	36

The ATM-sized DRP interleaver [110] used in Fig. 4.11 can be generated using (4.15), where the first 4 indices of the interleaver are {54, 0, 181, 135}, M=4 and r=100. The MPEG-sized DRP interleaver [110] used in Fig. 4.12 and Fig. 4.13 can be generated us-

ing (4.15), where the first 16 indices of the interleaver are {314, 230, 464, 3, 40,700, 577, 431, 194, 263, 665, 510, 68, 397, 629, 107}, M=16 and r=144. For both DRP interleavers, the 2 bits in each symbol were swapped before applying interleaving.

$$\pi(i+M) = (\pi(i)+r) \mod K, \quad i = 0, \cdots, K-1. \tag{4.15}$$

Fig. 4.11 shows, for an ATM packet with K=212 2-bit symbols, an improvement with the DRP interleaver greater than 0.15 dB at FERs below 10^{-6} compared to the DVB-RCS standard interleaver. Fig. 4.12 shows, for an MPEG packet with K=752 2-bit symbols, an improvement with the DRP interleaver greater than 0.4 dB at FERs below 10^{-6} compared to the DVB-RCS standard interleaver. Figs. 4.11, 4.12 and 4.13 were generated using enhanced max-log MAP decoding [36] with an extrinsic scale factor of 0.75.

The truncated union bounds (TUBs) in Figs. 4.11, 4.12 and 4.13 used only the first term of the distance spectrums. The gaps between the analytical bounds and the simulated curves are expected to be reduced when more terms of the distance spectrums are used. Unfortunately, the determination of higher terms is very computationally intensive because the $d_{\min}$ values are already quite high.

For comparison purposes, the performance of an S-random interleaver with $d_{\min}$=21, $A_{d_{\min}}$=1 and $W_{d_{\min}}$=6 was also simulated. Fig. 4.12 and Fig. 4.13 show that the S-random interleaver performs better than the standard interleaver, and closer to the DRP interleaver, at low SNR values. This is mainly because of the low multiplicity of the S-random interleaver, but the S-random interleaver also provides better convergence in the waterfall region. However, at high SNRs the performance of the S-random interleaver is worse than both the standard and DRP interleavers. This is because of the low $d_{\min}$ given by the S-random interleaver.

4.3 Berrou's Error-Impulse Method

Berrou *et al.* introduced in [33] a fast method for estimating the minimum distance based on the ability of a soft-in decoder to overcome error-impulse inputs. This method inserts a low-amplitude impulse into the all-zero codeword at a specific index to see if the decoder can correct it. This amplitude is increased in steps of 1 until the decoder fails. The lowest amplitude at which the decoder fails is recorded. Testing all data indices in the all-zero

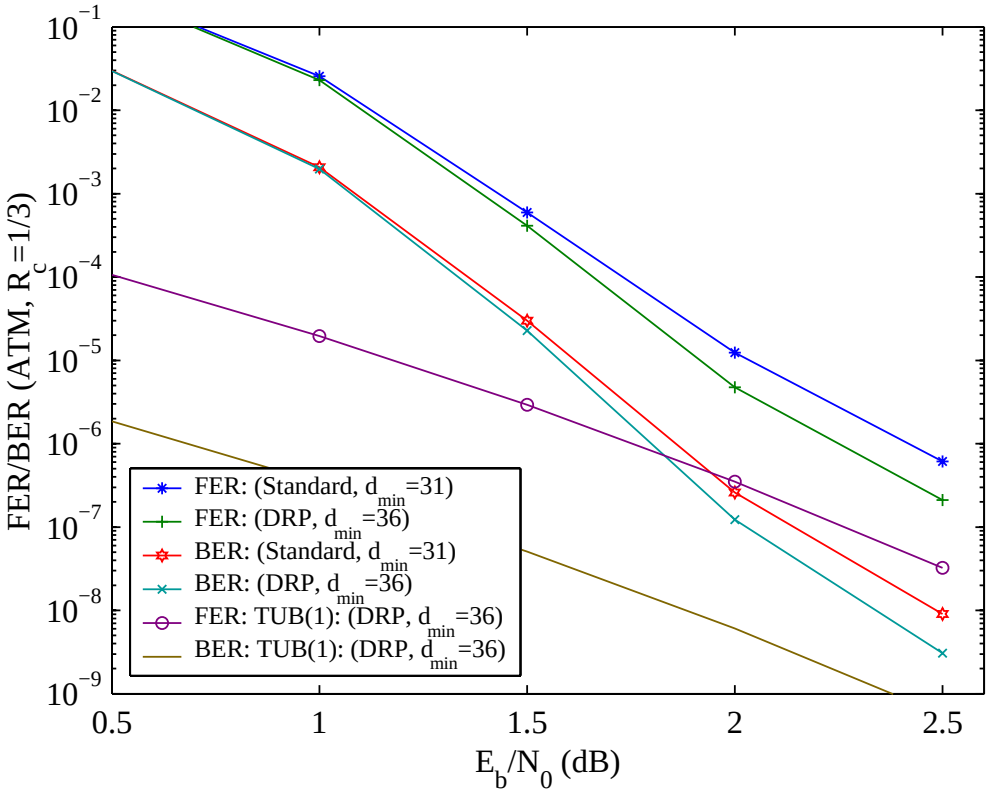


Fig. 4.11 FER and BER for ATM packets of code rate R_c=1/3 (QPSK/AWGN). The size of overlap is 50 symbols (100 bits) and the number of iterations is 8. Enhanced max-log MAP (EML-MAP) decoding with an extrinsic scale factor of 0.75 was used by the DVB-RCS turbo decoder. 100 million packets were simulated for both interleavers at 2.5 dB. The TUB(1) curves are the truncated union bounds using only the first term of the distance spectrum.

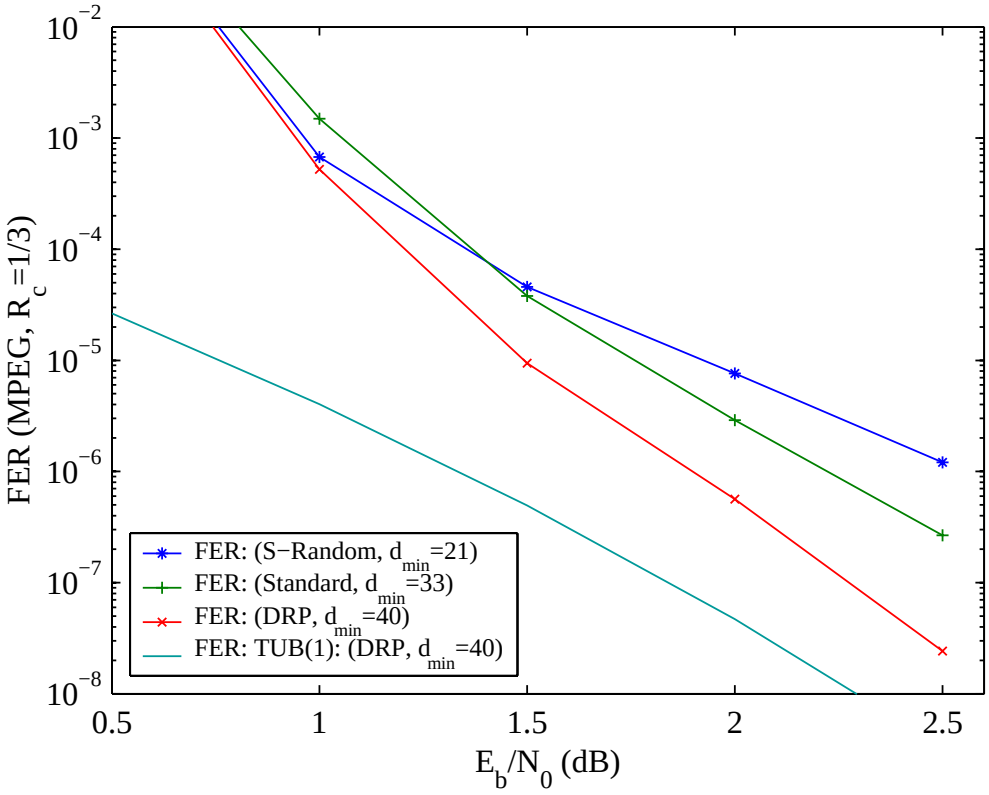


Fig. 4.12 FER for MPEG packets of code rate R_c=1/3 (QPSK/AWGN). The size of overlap is 75 symbols (150 bits) and the number of iterations is 8. Enhanced max-log MAP (EML-MAP) decoding with an extrinsic scale factor of 0.75 was used by the DVB-RCS turbo decoder. The numbers of packet errors at 2.5 dB were 100, 50 and 15 for the *S*-Random, standard and the new DRP interleaver, respectively. The TUB(1) curve is the truncated union bound using only the first term of the distance spectrum.

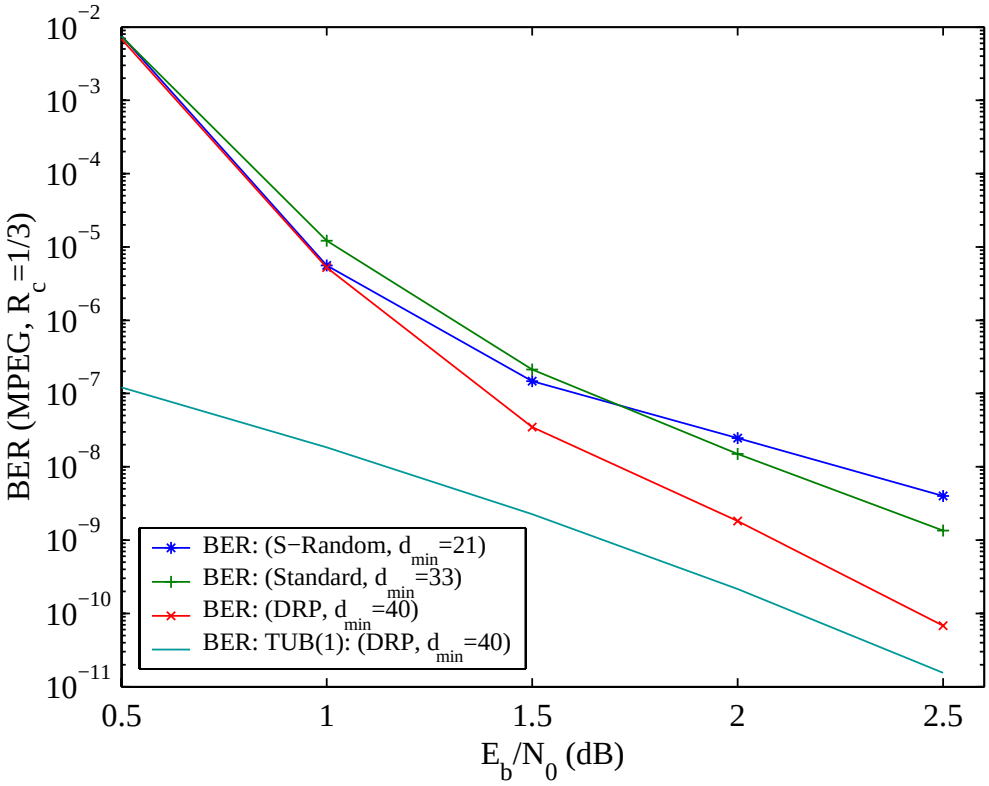


Fig. 4.13 BER for MPEG packets of code rate R_c=1/3 (QPSK/AWGN). The size of overlap is 75 symbols (150 bits) and the number of iterations is 8. Enhanced max-log MAP (EML-MAP) decoding with an extrinsic scale factor of 0.75 was used by the DVB-RCS turbo decoder. The numbers of packet errors at 2.5 dB were 100, 50 and 15 for the *S*-Random, standard and the new DRP interleaver, respectively. The TUB(1) curve is the truncated union bound using only the first term of the distance spectrum.

codeword in this manner gives a list of amplitudes. The lowest amplitude gives an estimate of $d_{\min}$.

A short description of this method follows. Define $\boldsymbol{x} = (1, 1, \ldots, 1)$ as the modulated codeword generated by the all-zero input sequence and $\boldsymbol{y} = (1, 1, \ldots, 1, 1 - E_i, 1 \ldots, 1)$ as the input to the decoder (E_i is called the error-impulse at position i and is a real number). Assuming that $d_{\min}$ lies in the interval $[d_0, d_1]$, where d_0 and d_1 are two integers and a ML decoder is used, then $d_{\min}$ can be determined with the following algorithm

‖ set $E_{\min} = d_1 + 0.5$;
‖ for $i = 0$ to $(\widetilde{N} - 1)$ do

– $E = d_0 - 0.5$;
– set $[(\hat{\boldsymbol{x}} = \boldsymbol{x}) = \text{TRUE}]$;
– while $[(\hat{\boldsymbol{x}} = \boldsymbol{x}) = \text{TRUE}]$ and $(E \leq E_{\min} - 1.0)$ do

- $E = E + 1.0$;
- $\boldsymbol{y} = (1, \ldots, 1, 1 - E, 1 \ldots, 1)$; where $(1 - E)$ is in position i;
- ML decoding of $\boldsymbol{y} \Rightarrow \hat{\boldsymbol{x}}$;
- If $(\hat{\boldsymbol{x}} \neq \boldsymbol{x})$ then $[(\hat{\boldsymbol{x}} = \boldsymbol{x}) = \text{FALSE}]$;

end while
– $E_{\min} = E$

end for
‖ $d_{\min}$ is the integer part of E_{min}

For turbo codes, because they are systematic, it is enough to consider the indexes of the $\widetilde{K}$ information bits instead all $\widetilde{N}$ indexes of the codeword. Thus, the first loop is reduced to: for $i = 0$ to $(\widetilde{K} - 1)$.

Since the iterative decoding complexity is linear in the codeword length, the complexity of this method is $O(\lambda\mu\widetilde{K}^2)$, where λ is the average number of iterations, μ is the average number of tested amplitudes (typically proportional to $d_{\min}$) and $\widetilde{K}$ is the number of information bits. To achieve good convergence, the maximum λ must be high. However, the average number of iterations can be significantly reduced using early stopping techniques, especially for amplitudes lower than the estimated $d_{\min}$.

It was shown in [33] that this method is guaranteed to find the true minimum distance if the decoder uses true maximum likelihood (ML) decoding. Unfortunately, turbo decoding is not guaranteed to perform a ML decoding because of the iterative nature of the decoding process. Thus, the relationship between the distance obtained with this method and the

true $d_{\min}$ remains uncertain. It has been observed that this method is usually pessimistic, but distances higher than $d_{\min}$ have also been found. Even so, the approach may prove to be very useful for finding good interleavers.

4.4 Garello's All-zero Iterative Decoding Method

Garello *et al.* introduced a method [97] similar to Berrou *et al.*'s method [33]. Instead of increasing the amplitude of the error-impulse in steps of 1 until the decoder fails to converge to the all-zero codeword [33], this method [97] intentionally sets the amplitude of the error-impulse to a high value so that the decoder can not converge to the all-zero codeword. Since the decoder fails, it estimates a non-zero input sequence. Encoding this input sequence leads to a non-zero codeword. The hamming weight of this codeword is an upper bound for the true $d_{\min}$.

A short description of this method follows. Let d^* be an upper bound on $d_{\min}$ and $w(\hat{\boldsymbol{c}})$ be the hamming weight of the estimated codeword $\hat{\boldsymbol{c}}$. The vector $\boldsymbol{y}$ represents the input to the decoder, assuming that the all-zero codeword is modulated using antipodal signaling. The index i corresponds to the information bit forced to be '1' and $A_{\min}$ is an estimate of the minimum distance multiplicity. If log MAP decoding is applied then a value for the noise variance, σ^2, must be selected.

‖ choose σ^2, if log MAP decoding is used;
‖ set $E_i \approx 2d^*$. set $d_{\min} = d^*$. set $A_{\min} = 1$;
‖ for $i = 0$ to $(\widetilde{K} - 1)$ do

- set $\boldsymbol{y} = (1, \ldots, 1, -E_i, 1 \ldots, 1)$, where $(-E_i)$ is in position i;
- iterative decoding of $\boldsymbol{y} \Rightarrow \hat{\boldsymbol{x}}$;
- encoding of $\hat{\boldsymbol{x}} \Rightarrow \hat{\boldsymbol{c}}$;
- determine $w(\hat{\boldsymbol{c}})$;
- if $w(\hat{\boldsymbol{c}}) < d_{\min}$, set $d_{\min} = w(\hat{\boldsymbol{c}})$ and $A_{\min} = 1$;
- else if $w(\hat{\boldsymbol{c}}) = d_{\min}$, update $A_{\min}$;

end for

Note that setting the amplitude of the error-impulse too high may lead to precision problems during the decoding, especially when log MAP decoding [61] is used. This method is slightly more reliable with log MAP decoding than with max-log MAP decoding. However, this does not mean that log MAP decoding is recommended. This is because the minimum

distance estimated with log MAP decoding is still a very loose upper bound on the true $d_{\min}$, at least for well designed interleavers. Also, more iterations can be used with max-log MAP for the sam complexity.

The complexity of this method is $O(\lambda \widetilde{K}^2)$, but it only works well for interleavers with low minimum distances. The reason is that these poor distances are usually due to only two short events, one associated with the un-interleaved input sequence entering ENC1 and the other associated with the interleaved input sequence entering ENC2. Forcing the information bit to be '1' at index i, by setting the amplitude of the error-impulse to a high value, and testing all indexes $i \in \{0, ..., \widetilde{K} - 1\}$ usually allows the decoder to find these two events. Sometimes, this method is not able to find the true minimum distance, even when it is due to only two error events, especially for short interleavers, where the spread is low. Fig. 4.14 illustrates a case found with two error events for the DVB-RCS standard packet of size 64 symbols (128 bits), where the true $d_{\min}$ is 25 and the estimated $d_{\min}$ is 30.

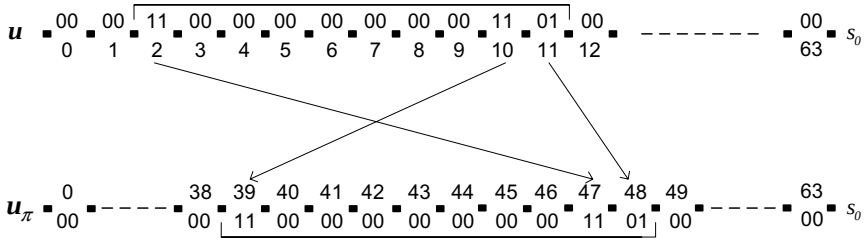


Fig. 4.14 Two events for the DVB-RCS standard packet of size 64 symbols (128 bits). Both encoders start and end in the all-zero state. The pattern above gives the true $d_{\min}$ of 25. The $d_{\min}$ estimated by the all-zero iterative decoding method is 30.

4.5 Crozier's Double-Impulse Iterative Decoding Method

Crozier *et al.* in [31, 32] improved upon the all-zero iterative decoding method [97] by introducing second (and third) impulse(s). Forcing the information bit to be '1' at a single index helps the decoder to converge to two events, one in the first encoder (ENC1) and the other in the second encoder (ENC2). Since high distances are often a result of more than two events, forcing two information bits to be '1' helps the decoder to converge to more events. The double-impulse method described in [31, 32] considered two impulses within a limited range of each other, where the maximum range required was about twice the

expected $d_{\min}$ value. This limited range approach keeps the complexity low, but typically only allows three separate events (one in ENC1 and two in ENC2) to be directly influenced by an impulse. Table 4.6 shows that the reliability of the double-impulse method can be improved by considering a full range search for the second impulse [30]. This is because a full range search typically allows up to four separate events (two in ENC1 and two in ENC2) to be directly influenced by an impulse. A short description of the full range double-impulse method follows, where i and j are the indexes for the two information bits forced to be '1s'.

‖ choose σ^2, if log MAP decoding is used;
‖ set $E_i = E_j \simeq 2d^*$. set $d_{\min} = d^*$. set $A_{\min} = 1$;
‖ for $i = 0$ to $(\widetilde{K} - 1)$ do

‖ for $j = i$ to $(\widetilde{K} - 1)$ do

- set $\boldsymbol{y} = (1, \ldots, 1, -E_i, 1, \ldots, 1, -E_j, 1, \ldots, 1)$, where $(-E_i)$ and $(-E_j)$ are in positions i and j;
- iterative decoding of $\boldsymbol{y} \Rightarrow \hat{\boldsymbol{x}}$;
- encoding of $\hat{\boldsymbol{x}} \Rightarrow \hat{\boldsymbol{c}}$;
- determine $w(\hat{\boldsymbol{c}})$;
- if $w(\hat{\boldsymbol{c}}) < d_{\min}$, set $d_{\min} = w(\hat{\boldsymbol{c}})$ and $A_{\min} = 1$;
- else if $w(\hat{\boldsymbol{c}}) = d_{\min}$, update $A_{\min}$;

end for

end for

With this method the decoder must decode $\widetilde{K} \cdot (\widetilde{K}+1)/2$ packets, which results in complexity of $O(\lambda \widetilde{K}^3)$. However, the complexity is reduced when testing structured interleavers such as dithered relative prime (DRP) interleavers [60] for tail-biting turbo codes because the distance properties repeat every M indexes (typical values for M are 4, 8, 16 and 32). Since only M indexes in the first loop need to be tested, the complexity is $O(\lambda M K^2)$. It has been observed for this method and the all-zero iterative decoding method that a high value for λ does not necessarily increase the reliability. Thus, it is usually sufficient to set λ to a moderate value (e.g., 16 or 32).

This method, as well as the other iterative methods, works best for interleavers with high spread. It has also been found that this method works very well with max-log MAP decoding, where the knowledge of SNR is not required.

Note[2] that the double-impulse method with a limited range search was able to find the true minimum distance for most of the interleavers tested in [32] (see Fig. 1 and Fig. 2 in [32]). Table 4.6 shows a few examples selected from [32], where the limited range search was not able to provide the true $d_{\min}$ values. The minimum distances, $d_{\min}$(true or TIM), are obtained partially by Garllo's true method and partially by Crozier's triple-impulse method (TIM) [32]. Minimum distances obtained with double-impulse method using limited and full range search are referred to as $d_{\min}$(LRS) and $d_{\min}$(FRS), respectively.

Table 4.6 Comparison of minimum distances obtained with Crozier's double-impulse method (DIM) using limited and full range search for the second impulse. Double-impulse method with limited and full range search as well as TIM method used 16 full iterations. $\widetilde{K}$ is in bits.

$\widetilde{K}$	504	728	1024	1256	1512	1616	1728	2120	2160
$d_{\min}$(true or TIM)	40	43	46	49	51	51	52	52	52
$d_{\min}$(LRS)	42	44	48	50	52	52	54	54	53
$d_{\min}$(FRS)	40	43	46	49	51	52	52	53	53

4.5.1 Distance Results

The all-zero iterative decoding method and double-impulse iterative decoding method with full range search are referred to in this section as the single-impulse method (SIM) and the double-impulse method (DIM), respectively. Using the DVB-RCS turbo-code encoder, the true $d_{\min}$ values, obtained using the method in [28], are compared with the distances obtained with the error-impulse method (EIM), the SIM and the DIM. The EIM uses max-log MAP decoding with early stopping (ES), where B is the number of consecutive sets of hard decisions that must agree before stopping [95, 36, 34]. Genie ES can also be used with the EIM. With genie ES the decoder stops when at any half-iteration the all-zero codeword is produced. The maximum number of full iterations was set to 256 for EIM and 16 for SIM and DIM. $\overline{d}_{\text{method}}$ and σ_{method} are the average distance and the standard deviation for a specific method, respectively, and $K = \widetilde{K}/2$ is the number of 2-bit information symbols.

[2]Special thanks to the communications and signal processing group at the Communications Research Centre (CRC) in Ottawa for providing the DRP interleavers used in Table 4.6 as well the distance results shown in Table 4.6.

DVB-RCS standard interleavers

The distances shown in Table 4.7 are for the twelve DVB-RCS standard interleavers. The code rate is 1/3 and early stopping with different B values were investigated. The use of ES reduces the computational complexity of the EIM. The EIM is used with both normal ES and genie ES.

Table 4.7 shows that the distances estimated using the EIM with normal ES are pessimistic. The EIM with normal ES and $B = 16$ gives the same distances as with $B = 8$, thus in the following sections only $B = 8$ was used. The EIM with genie ES gives pessimistic results for packet sizes less than or equal to 228 2-bit symbols and optimistic results for packet sizes greater than or equal to 424 2-bit symbols. The SIM tends to be very optimistic. In fact, it always provides an upper bound on the true $d_{\min}$. The DIM was able to find the true $d_{\min}$ for all twelve DVB-RCS standard interleavers.

Table 4.7 Distances shown here are for the code rate 1/3. DVB-RCS standard interleavers were used with normal ES, where B is the number of consecutive sets of hard decisions that must agree before stopping. The sign – indicates that there is no difference between the corresponding result and the result estimated with normal ES and B=4. K is in 2-bit symbols.

	true $d_{\min}$	d_{EIM} with normal ES			d_{EIM} with genie ES	d_{SIM}	d_{DIM}
K		$B = 4$	$B = 8$	$B = 16$			
48	21	17	–	–	18	21	21
64	25	16	–	–	17	31	25
212	31	25	24	24	30	188	31
220	31	23	22	22	28	257	31
228	30	25	–	–	29	84	30
424	30	24	–	–	32	364	30
432	31	25	–	–	33	497	31
440	28	25	24	24	32	509	28
752	33	28	–	–	38	204	33
848	36	28	27	27	37	634	36
856	33	28	–	–	38	332	33
864	36	27	–	–	35	332	36

Table 4.8 shows the distances with the MPEG packet size for the seven standard code rates (R_c). Each distance estimated with EIM using normal ES is pessimistic. The EIM with genie ES gives optimistic results for the lower code rates and pessimistic results for the higher code rates. Note that the SIM continues to give very optimistic results with

puncturing. The DIM was able to find the true $d_{\min}$ for all but one of the seven code rates. For the rate 3/4 case, the estimated minimum distance was only $d_{\min} + 1$.

Table 4.8 Distances for MPEG size (K=752 2-bit symbols) using DVB-RCS standard interleaver and standard puncturing. Normal ES with various B were used. The sign – indicates that there is no difference between the corresponding result and the result with normal ES and B=4.

	true $d_{\min}$	d_{EIM} with normal ES			d_{EIM} with genie ES	d_{SIM}	d_{DIM}
R_c		$B = 4$	$B = 8$	$B = 16$			
1/3	33	28	–	–	38	204	33
2/5	27	23	–	–	30	397	27
1/2	19	17	–	–	20	219	19
2/3	12	11	–	–	14	394	12
3/4	9	7	–	–	8	32	10
4/5	9	7	6	6	9	110	9
6/7	6	4	–	–	5	12	6

Random interleavers

Tables 4.7 and 4.8 indicate that the distances estimated with the EIM and genie ES can be misleading. They are a mix of optimistic and pessimistic results. Thus, in this section the investigation of EIM was limited to normal ES with $B = 8$. The four methods were tested with 1000 random interleavers for the code rate 1/3.

Table 4.9 indicates that the longer the interleaver is, the better the distance estimated with EIM. One possible explanation is that increasing the size of a random interleaver does not imply a significant increase in $d_{\min}$, but it improves the reliability of the extrinsic information, which in turn helps convergence.

The SIM produced the true $d_{\min}$ in all cases, except for two short interleavers of size 48 2-bit symbols. It is interesting to note that the two events (one in ENC1 and one in ENC2) leading to the true $d_{\min}$ for both interleavers were short events. This means that the SIM fails sometimes even when $d_{\min}$ is caused by only two short events. These two cases are shown in Fig. 4.15 and Fig. 4.16. However, the estimated minimum distance tends to be a tight upper bound on $d_{\min}$ for random interleavers.

Since the SIM is a subset of the DIM, and both methods give true upper bound on $d_{\min}$, it is enough to test these two interleavers of size 48 2-bit symbols with the DIM. The

Table 4.9 Distances shown here are obtained using DVB-RCS standard encoder and code rate 1/3. K is in 2-bit symbols. The m random interleavers were tested using normal ES with $B = 8$. The signs – and • indicate that there is no difference between the estimated $\overline{d}_{\text{method}}$, σ_{method} and the true ones. The abbreviations "agr." and "disagr." refer to the number of agreements and disagreements compared to true $d_{\min}$.

		true $d_{\min}$		EIM($B = 8$)		SIM		EIM(B=8)		SIM	
K	m	$\overline{d}_{\min}$	$\sigma_{\min}$	$\overline{d}_{\text{EIM}}$	σ_{EIM}	$\overline{d}_{\text{SIM}}$	σ_{SIM}	agr.	disagr.	agr.	disagr.
48	1000	10.545	1.445	10.057	1.122	10.547	1.447	627	373	998	2
212	1000	10.743	1.509	10.604	1.435	–	•	888	112	1000	0
752	1000	10.854	1.543	10.808	1.507	–	•	961	39	1000	0
1024	1000	10.874	1.554	10.854	1.546	–	•	981	19	1000	0
1504	1000	10.781	1.541	10.768	1.536	–	•	988	12	1000	0

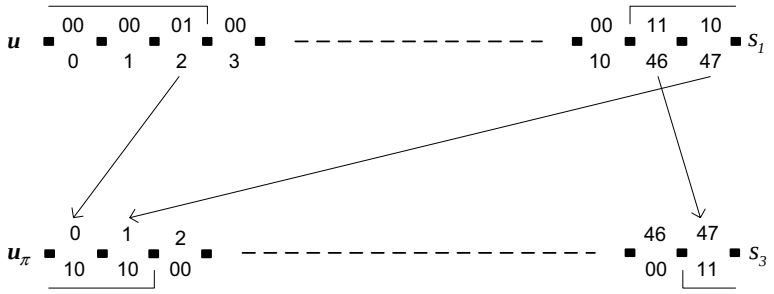


Fig. 4.15 For DVB-RCS standard encoder and a random interleaver of size 48 symbols (96 bits), the depicted two events cause the true $d_{\min}$. The first encoder starts and ends in state '001'. The second encoder starts and ends in state '011'. The pattern above gives the true $d_{\min}$ of 11. The distance estimated by the single-impulse method (SIM) is 12.

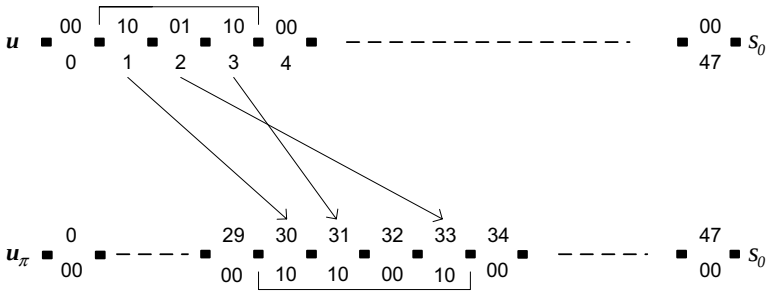


Fig. 4.16 For DVB-RCS standard encoder and a random interleaver of size 48 symbols (96 bits), the depicted two events cause the true $d_{\min}$. Both encoders start and end in the all-zero state. The pattern above gives the true $d_{\min}$ of 12. The distance estimated by the single-impulse (SIM) method is 13.

DIM found the true $d_{\min}$ for these two interleavers. Thus, the DIM found the true $d_{\min}$ in all cases.

Dithered relative prime interleavers

The four methods were tested with 20 DRP interleavers for the code rate 1/3. All 20 interleavers were unique in the tail-biting sense, meaning that any shift or offset or combination of both to any interleaver does not result in any other interleaver. The EIM used normal ES with $B = 8$.

Table 4.10 shows that EIM and SIM were not able to find the true $d_{\min}$ for any of the tested DRP interleavers. The results show that EIM and SIM are very pessimistic and optimistic, respectively. The DIM was able to find the true $d_{\min}$ for all tested interleavers.

Further investigation, showed that the DIM is also able to find the true minimum distance, even when the distances are high. The DIM found the true $d_{\min}$ values of 36 and 40 for DRP-interleavers of sizes 212 and 752 2-bit symbols, respectively.

Table 4.10 Distances shown here are obtained using DVB-RCS standard encoder and code rate 1/3. K is in 2-bit symbols. The m DRP interleavers were tested with normal ES and $B = 8$. The abbreviations "agr." and "disagr." refer to the number of agreements and disagreements compared to true $d_{\min}$.

						EIM(B=8)		SIM		DIM	
K	m	$\overline{d}_{\min}$	$\overline{d}_{\text{EIM}}$	$\overline{d}_{\text{SIM}}$	$\overline{d}_{\text{DIM}}$	agr.	disagr.	agr.	disagr.	agr.	disagr.
48	20	22	15.2	51.9	22	0	20	0	20	20	0
212	20	31	23.2	154.9	31	0	20	0	20	20	0
752	20	36	26.4	280.5	36	0	20	0	20	20	0

4.5.2 Complexity Comparison

The EIM and SIM have the lowest complexity and are very fast methods. However, they provide very poor estimates for $d_{\min}$, especially for well-designed interleavers. Thus, they are not considered in the comparison. This comparison is restricted to the Garello's true distance measurement method and the more reliable DIM. In this comparison, the DIM used 16 iterations. Note that an increase in the number of iterations usually leads to better reliability of the DIM, especially for interleavers that yield high $d_{\min}$ values.

Rate 1/3 DVB-RCS standard turbo-code encoder was used together with 3 MPEG-sized interleavers, namely, the standard interleaver and two new DRP interleavers. The reported

CPU times were obtained with a 2.4 GHz Pentium 4 (Xeon) processor. T_{DIM} and T_{GARELLO} refer to CPU times (in minutes) required with the DIM and the Garello's true method, respectively.

Table 4.11 shows that the execution times with Garello's true method depend strongly on the value of $d_{\min}$. The test of the new MPEG-sized DRP interleavers that yield $d_{\min}$=36 and 40 increased the execution times by a factor of 4 and 29, respectively compared to the standard interleavers that yield a $d_{\min}$=33. This demonstrates that the execution time increases very rapidly with $d_{\min}$. Thus, an efficient distance measurement method is required to allow the test of high $d_{\min}$ in a reasonable time.

Table 4.11 shows that the execution times with DIM are the same for all interleavers and are much less than the lowest reported T_{GARELLO}. These results are encouraging because of the low complexity and the good reliability of the DIM. However, the DIM is not guaranteed to provide the true $d_{\min}$. Even if it finds the true $d_{\min}$, the multiplicities remains uncertain. Since the aim is to develop a distance measurement method that determines at least the true minimum distance, the focus of the next chapter is to further reduce the complexity of the Garello's true method to allow the accurate determination of high minimum distances in reasonable times.

Table 4.11 Distance results ($d_{\min}$) and CPU times in minutes for rate 1/3 DVB-RCS turbo-code encoder with various MPEG-sized interleavers (752 2-bit symbols). The DIM used 16 iterations.

	DVB-RCS standard interleaver ($d_{\min}$=33)	DRP interleaver ($d_{\min}$=36)	DRP interleaver ($d_{\min}$=40)
T_{DIM}	199	199	199
T_{GARELLO}	348	1400	10153

4.6 Conclusion

Garello's true distance measurement method for turbo codes was discussed and extended to tail-biting turbo codes. Garello's techniques to reduce the computational complexity were discussed. A new technique that reduces the average computational complexity by a factor of 2 was presented. This reduction in computational complexity is significant because the measurement of high distances (i.e., $d_{\min} > 51$) can take months (or even years). Distance results for some UMTS standard interleavers were presented. Distance results for the twelve DVB-RCS standard interleavers over all standard code rates were presented.

New DVB-RCS interleavers of lengths 212 (ATM) and 752 (MPEG) 2-bit symbols were designed based on DRP approach. The new DVB-RCS interleaver design achieves, for rate-1/3 ATM packets, an improvement of at least 0.15 dB at FERs below 10^{-6} compared to the UMTS standard interleaver. The new DVB-RCS interleaver design, for rate-1/3 MPEG packet achieves, an improvement of at least 0.4 dB at FERs below 10^{-6} compared to the DVB-RCS standard interleaver.

Four distance measurement methods were compared based on the DVB-RCS standard interleavers, random interleavers and dithered relative prime (DRP) interleavers. Garello's method always finds the true $d_{\min}$ value, but it is the most computationally intensive. The error-impulse method (EIM) is usually pessimistic, whereas the single-impulse method (SIM) is usually optimistic and it always gives a true upper bound on $d_{\min}$. These two iterative methods can be useful for estimating low minimum distances, typically associated with random interleavers. For random interleavers with low distances, the reliability of both methods increases with the size of the interleaver. The double-impulse method (DIM) with full range search is much more reliable and was able to find the true minimum distance in almost all cases.

The EIM and SIM have low complexity that grows linearly with interleaver size, whereas the DIM has a moderate complexity that grows in a quadratic manner with interleaver size. The complexity of Garello's true distance measurement method depends mainly on the value of $d_{\min}$ and increases very rapidly with $d_{\min}$. The next chapter introduces a new method that reduces the complexity of Garello's true method to allow the determination of high minimum distances in reasonable time.

Chapter 5

An Efficient and Accurate Distance Measurement Method for Tail-biting Turbo Codes that use Structured Interleavers

An efficient and accurate distance measurement method for tail-biting turbo codes that use structured interleavers is presented. This method takes advantage of the structure in the interleaver as well as the circular property of tail-biting. The efficiency of this method is demonstrated for both single- and double-binary turbo codes using the best known MPEG-sized interleavers (1504 information bits) [60, 37, 28, 30] for various code rates.

5.1 Background

There is a growing body of literature on the design of interleavers that yield high minimum distances ($d_{\min}$) [60, 111]. Such interleavers are important for lowering the error flare of turbo codes, allowing them to achieve very low error rates at low to moderate signal-to-noise ratios (SNRs). A significant challenge is to determine their distance spectra or at least their $d_{\min}$ values and corresponding multiplicities.

Berrou *et al.* introduced in [33] a fast method that uses error-impulse inputs. This method is suitable for iterative decoding and has been applied to single-binary turbo codes

in [31, 32] and to double-binary turbo codes in [33, 30]. It has been shown in [31, 32, 30] that this method usually gives a lower bound on $d_{\min}$, but distances higher than $d_{\min}$ have also been found. A similar method has been presented by Garello *et al.* in [97]. This method tends to be very optimistic, see [31, 32, 30]. The accuracy of these two methods is very poor for high $d_{\min}$ interleavers [31, 32, 30]. More accurate iterative decoding methods with low complexity were presented in [31, 32]. As shown in [31, 32, 30], these more accurate methods find the correct $d_{\min}$ most of the time. However, the accuracy of these methods remains uncertain, especially for long interleavers that yield high $d_{\min}$ values. Even if they find the true $d_{\min}$, they cannot be guaranteed to find the correct multiplicities.

A novel and accurate distance measurement method was introduced by Garello *et al.* in [27] for single-binary turbo codes. It has been improved significantly by Rosnes in [112] and extended to tail-biting and double-binary turbo codes in [29, 28]. This method tests all possible non-zero input data sequences $\boldsymbol{u}^{K-1} = (0, \cdots, 0, \chi)$, $\boldsymbol{u}^{K-2} = (0, \cdots, 0, \chi, \times)$, $\cdots$, $\boldsymbol{u}^{1} = (0, \chi, \times, \cdots, \times)$, $\boldsymbol{u}^{0} = (\chi, \times, \cdots, \times)$. Here, K is the interleaver length in symbols and 0 represents the zero-symbol (i.e., $\{0\}$ for single-binary turbo codes and $\{00\}$ for double-binary turbo codes). The variable χ is either $\{1\}$ for single-binary turbo codes or an element of $\{01, 10, 11\}$ for double-binary turbo codes. The variable $\times$ is an element of $\{0, 1\}$ or $\{00, 01, 10, 11\}$ for single- or double-binary turbo codes, respectively. For more details, see [27, 28] and Chapter 4. This method provides the true $d_{\min}$ and the true multiplicities. However, for interleavers that yield high $d_{\min}$ values, the complexity increases rapidly with $d_{\min}$, making the test impractical. This complexity can be reduced significantly for tail-biting turbo codes [56, 57, 58] that use highly structured interleavers. This is because the distance properties repeat every few data symbols. However, one must be careful when computing the multiplicities. It is not as simple as just testing a small number of indices. Examples showing this problem are discussed in the next section and a solution is also presented.

5.2 Complexity Reduction

The new method is based on Garello's true distance measurement method [27, 28]. In fact, the core of the algorithm remains the same as Garello's algorithm for each symbol index tested. The new method efficiently determines the true $d_{\min}$ and the true multiplicities for tail-biting turbo codes that use structured interleavers. Structured interleavers, such

as dithered relative prime (DRP) interleavers [60], standard digital video broadcast with return channel via satellite (DVB-RCS) interleavers [15] and almost regular permutation (ARP) interleavers [111] have the following property:

$$\pi\left([i+M]_K\right) = \left[\pi(i) + Mp\right]_K, i = 0, \ldots, K-1 \tag{5.1}$$

where $[x]_K$ is x modulo K, and M is the number of repeating index increments required to implement the interleaver π. K must be a multiple of M and the integer values p and K must be relative primes to ensure that the interleaver references all symbol indices.

5.2.1 Distance Properties

Let L be the least common multiple of M, the various mask lengths used to puncture the data and parity symbols, and the pre-interleaving mask length used in non single-binary turbo codes to manipulate the bits within a symbol before the actual interleaving. For structured interleavers, the distance properties of tail-biting turbo codes repeat every L indices if K is a multiple of L. This can be explained as follows.

i From (5.1) it follows that a circular shift over qM positions (q is an integer) of an input sequence $\boldsymbol{u}$ entering the first encoder (ENC1) results in a circular shift over qpM positions of the corresponding input sequence $\boldsymbol{u}_\pi$ entering the second encoder (ENC2).

ii From the circular property of tail-biting it follows that circular shifts of $\boldsymbol{u}$ and $\boldsymbol{u}_\pi$ result in circular shifts of their parities.

Form (i) and (ii), it follows that the distance resulting from a circular shift of $\boldsymbol{u}$ over qM positions is the same as the distance resulting from a non-shifted $\boldsymbol{u}$, provided that no puncturing or pre-interleaving is involved. If puncturing or pre-interleaving is involved, the distance is guaranteed to be shift-invariant, if the circular shift of $\boldsymbol{u}$ goes over qL positions. Thus, the distance properties repeat every L indices.

To summarize, without puncturing, the distance properties of tail-biting turbo codes repeat every M indices. With puncturing, they repeat every L indices. Thus, the $d_{\min}$ is guaranteed to be found if the first L indices are tested for all $\Delta_1 \cdot \Delta_2$ state combinations, where Δ_1 and Δ_2 are the number of starting (and ending) states in ENC1 and ENC2, respectively.

An error event refers to the input symbols associated with a path in the trellis that departs from the all-zero state and returns to the all-zero state without passing through the all-zero state. Each input sequence $\boldsymbol{u}_{\min}$ that causes $d_{\min}$ has at least Z consecutive zero symbols that are not a part of any error events. Note that Z can be as small as 0 for very short interleavers, but is typically much greater than zero for most interleavers of interest. This Z determines the number of state combinations that need to be considered and the locations of the L indices to be tested. If $Z < (L-1)$, the first L indices $\{L-1, \cdots, 0\}$ must be tested considering all $\Delta_1 \cdot \Delta_2$ state combinations. If $Z \geq (L-1)$, which is usually the case even for fairly short interleavers, only the state combinations where ENC1 starts and ends in the all-zero state need to be considered (i.e., Δ_2 state combinations). This leads to a reduction in complexity, especially if puncturing is involved. It is also enough to test the L indices $\{Z, \cdots, Z-L+1\}$. This reduces the complexity even further, especially for large Z. This is because a large number of leading symbols are know to be zero-symbols, as discussed earlier.

5.2.2 How to Determine the Correct Multiplicity

As mentioned above, care must be taken when determining the multiplicities. A '*shift*' of an input sequence refers to a circular shift of the input sequence by a multiple of L positions. Any input sequence that causes $d_{\min}$ can be used to *represent* all shifts of that input sequence that also cause $d_{\min}$. The multiple shifts of this input sequence will be counted later by multiplying by K/L. The goal now is to count only one representative from each unique set of shifted input sequences. The following two examples demonstrate the details associated with the determination of such representative input sequences for two cases, namely, $Z < (L-1)$ and $Z \geq (L-1)$. For the examples considered below, let L be 4.

Case ($Z < L-1$):

Assume that $d_{\min}$ is caused by the representative input sequence $\boldsymbol{u}_{\min}$. Recall that all state combinations must be considered. Thus, H shifts of $\boldsymbol{u}_{\min}$ will be found where H is the number of χ variables in $\boldsymbol{u}_{\min}$ that are immediately preceded by at least b consecutive zero symbols that satisfy $b \geq [i]_L$, where i is the position of χ in $\boldsymbol{u}_{\min}$. This means that each χ in $\boldsymbol{u}_{\min}$ could cause a shift of $\boldsymbol{u}_{\min}$ to be found. This is demonstrated using the

universal mobile telecommunications system (UMTS) 8-state polynomial generators [25]. Assume that $d_{\min}$ is caused by the representative single-binary input sequence $\boldsymbol{u}_{\min} = (1_0, 0, 0, 1, 0, 0, 1_2, 1, 0, 1_1, 0, 1)$, where subscripts are used for reference purposes. When testing the first $L = 4$ indices, three shifts of $\boldsymbol{u}_{\min}$ will be found (i.e., $H = 3$):

- $\boldsymbol{u}^2_{\min} = (0, 0, 1_2, 1, 0, 1_1, 0, 1, 1_0, 0, 0, 1)$
- $\boldsymbol{u}^1_{\min} = (0, 1_1, 0, 1, 1_0, 0, 0, 1, 0, 0, 1_2, 1)$
- $\boldsymbol{u}^0_{\min} = (1_0, 0, 0, 1, 0, 0, 1_2, 1, 0, 1_1, 0, 1)$

when indices 2, 1 and 0 are tested, respectively. This is because 1_2, 1_1 and 1_0 at positions 6, 9 and 0 in $\boldsymbol{u}_{\min}$ are immediately preceded by at least $[6]_4 = 2$, $[9]_4 = 1$ and $[0]_4 = 0$ zeros, respectively. Since $\boldsymbol{u}^2_{\min}$, $\boldsymbol{u}^1_{\min}$ and $\boldsymbol{u}^0_{\min}$ are shifts of $\boldsymbol{u}_{\min}$ by 4, 8 and 0 positions to the left, respectively, three shifts of $\boldsymbol{u}_{\min}$ are found. However, the goal is to count only one representative of $\boldsymbol{u}_{\min}$. One efficient solution is to recognize that when $\boldsymbol{u}^2_{\min}$ is found, $\boldsymbol{u}^1_{\min}$ and $\boldsymbol{u}^0_{\min}$ will also be found. Similarly, when $\boldsymbol{u}^1_{\min}$ is found, $\boldsymbol{u}^2_{\min}$ and $\boldsymbol{u}^0_{\min}$ will also be found. As well, when $\boldsymbol{u}^0_{\min}$ is found, $\boldsymbol{u}^2_{\min}$ and $\boldsymbol{u}^1_{\min}$ will also be found. To count $\boldsymbol{u}_{\min}$ only once, each shift of $\boldsymbol{u}_{\min}$ that is found is counted only $1/H$ of the time, where H is the total number of shifts found. In this example, $H = 3$ and $\boldsymbol{u}_{\min}$ is counted only once by counting it $\frac{1}{3}$ of the time each of the three times a shift of it is found.

Case ($Z \geq L - 1$):

Since ENC1 starts and ends in the all-zero state, only χ at the beginning of an error event could cause a shift of $\boldsymbol{u}_{\min}$ to be found. Again, this is demonstrated using the UMTS 8-state polynomial generators. Assume that $d_{\min}$ is caused by the representative single-binary input sequence $\boldsymbol{u}_{\min} = (0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1) = (0, e_1, 0, 0, 0, 0, e_2, e)$, where $e_1 = (1, 1, 1, 0, 1)$, $e_2 = (1, 1, 0, 0, 0, 1)$ and $e = (1, 0, 1, 1)$ are distinct error events. In this example, Z is 4 and the indices to be tested are $\{4, 3, 2, 1\}$. Only two shifts of $\boldsymbol{u}_{\min}$ will be found:

- $\boldsymbol{u}^2_{\min} = (0, 0, e_2, e, 0, e_1, 0, 0)$
- $\boldsymbol{u}^1_{\min} = (0, e_1, 0, 0, 0, 0, e_2, e)$

when indices 2 and 1 are tested, respectively. Since $\boldsymbol{u}^2_{\min}$ and $\boldsymbol{u}^1_{\min}$ are shifts of $\boldsymbol{u}_{\min}$ by 8 and 0 to the left, respectively, two shifts of $\boldsymbol{u}_{\min}$ are found. However, the goal is to count only one representative of $\boldsymbol{u}_{\min}$. As before, one solution is to recognize that when $\boldsymbol{u}^2_{\min}$ is found, $\boldsymbol{u}^1_{\min}$ will also be found and vice versa. In this example, $H = 2$ and $\boldsymbol{u}_{\min}$ is counted only once by counting it $\frac{1}{2}$ of the time each of the two times a shift of it is found.

Given that an arbitrary $\boldsymbol{u}^j_{\min}$ was found while testing index j, the question now is how to recognize the other shifts of $\boldsymbol{u}^j_{\min}$ that will also be found. The answer is as follows for the two cases.

Case ($Z < L - 1$):

Let $\ell(i)$ be the number of consecutive zero symbols immediately preceding a χ at position i in $\boldsymbol{u}^j_{\min}$, where $i = j + 1, \cdots, K - 1$ are tested for χ. From the first example given above, it follows that a shift of $\boldsymbol{u}^j_{\min}$ is guaranteed to be found during the test of index $[i]_L$ if $\ell(i) \geq [i]_L$.

Case ($Z \geq L - 1$):

Let $\ell_e(i)$ be the number of consecutive zero symbols immediately preceding an error event that starts at position i in $\boldsymbol{u}^j_{\min}$, where $i = j+1, \cdots, K-1$ are tested for the start of an error event. A circular shift of position i must result in a new position $i' \in \{Z, \cdots, Z - L + 1\}$. From the second example given above, it follows that a shift of $\boldsymbol{u}^j_{\min}$ is guaranteed to be found during the test of index i' if $\ell_e(i) \geq i'$. It can be shown that

$$i' = i - L \cdot \lfloor (i - Z + L - 1)/L \rfloor , \tag{5.2}$$

where $\lfloor x \rfloor$ is the largest integer less than or equal to x.

Recall that H is the total number of shifts found. Each time an input sequence $\boldsymbol{u}_{\min}$ that causes $d_{\min}$ is found, H is determined and the codeword multiplicity is increased by $\frac{1}{H}$. Also, the information bit multiplicity is increased by $w(\boldsymbol{u}_{\min})/H$, where $w(\boldsymbol{u}_{\min})$ is the Hamming weight of $\boldsymbol{u}_{\min}$. The overall true codeword multiplicity ($A_{d_{\min}}$) and the true information bit multiplicity ($W_{d_{\min}}$) are obtained by multiplying the multiplicities determined above by K/L.

The approach is easily extended to higher distances so that more terms of the distance

spectrum can be determined.

5.3 Remarks on Complexity Reduction when Z is used

A deterministic polynomial-time algorithm to compute the minimum distance would be a good step toward the ideal solution. This is because such an algorithm could be used to construct good turbo codes by choosing an interleaver at random and checking if the associated code has a large minimum distance. Unfortunately, no such algorithm is known. More details about the hardness of computing the minimum distance for the class of binary linear codes can be found in [113, 114, 115].

The complexity of Garello's distance measurement method is hard to determine analytically. Thus, the reduction in complexity due to the use of Z is discussed based on a brute force method. Applying the standard Garello's method requires testing $2^K - 1$ non-zero input sequences, whereas the use of Z requires testing only $2^{(K-Z)} - 1$ non-zero input sequences. Consequently, the use of Z yields a reduction in complexity by a factor of about 2^Z. This clearly shows that the use of even a moderate Z value has the potential to result in a significant reduction in complexity.

Obviously, Garello's method does not explicitly test all possible non-zero input sequences. Thus, the reduction in complexity is upper bounded by 2^Z. The brute force method used above was introduced to demonstrate that the use of Z could yield a significant reduction in complexity. The actual observed reductions in complexity are typically much smaller, but still very significant, as shown in Tables 5.1, 5.2 and 5.3, for example.

5.4 Example Distance and Complexity Results

Distance results and execution times are reported for both a double-binary turbo code that uses the DVB-RCS 8-state polynomial generators [15] and a single-binary turbo code that uses the UMTS 8-state polynomial generators [25]. Results are presented for MPEG-sized (1504 information bits) interleavers, various code rates, R_c, and several Z values for the new method. The reported CPU times (in minutes) T_{OLD} and $T_{\text{NEW}}(Z)$ required with the old and new methods, respectively, were obtained with a 2.4 GHz Pentium 4 (Xeon) processor. Note that all methods have been implemented in C programming language. As a consistency check, for cases where original Garello's true method is applicable, this new

method gave identical results (in much shorter time).

Table 5.1 shows the results for the double-binary DVB-RCS 8-state turbo-code encoder with the MPEG-sized standard interleaver. A value of $L = 4$ symbols is sufficient for all the standard code rates reported in Table 5.1. The $T_{\text{NEW}}(Z)$ results are for $Z = L - 1 = 3$ symbols (6 bits) and $Z = 150$ symbols (300 bits). Comparing T_{OLD} with $T_{\text{NEW}}(Z = 3)$ shows that the execution times are reduced significantly using a Z value of only 3 symbols. Further reductions in the execution times is achieved using $Z = 150$ symbols. As expected, the measured distances and multiplicities were exactly the same for all three methods used.

Table 5.2 shows the results obtained with new MPEG-sized DRP interleavers for the DVB-RCS encoder. The reported code rates use DVB-RCS standard puncturing masks. A value of $L = 4$ symbols is sufficient for all the code rates reported in Table 5.2, except for rate 1/3 where $L = 8$ symbols. As an example, for rate 2/5, the use of $Z = 3$ and $Z = 150$ symbols reduced the execution times by factors of 25 and 400, respectively, compared to the old method. Note that for rate 1/3, the new DRP interleaver gives a $d_{\min}$ of 40, whereas the standard interleaver gives a $d_{\min}$ of 33.

Table 5.3 shows the results for the single-binary UMTS 8-state turbo-code encoder with new MPEG-sized DRP interleavers. A value of $L = 8$ symbols is sufficient for all the code rates reported in Table 5.3. The accurate determination of $d_{\min} = 51$ would not be possible in reasonable time without the use of the new method. The reported T_{OLD} values for code rates 1/3, 2/5 and 1/2 are optimistic estimates obtained by testing only a subset of indices.

The use of Z values of 150 symbols, 150 symbols and 200 bits in Tables 5.1, 5.2 and 5.3, respectively, show a typical reduction in execution time by a factor of 40 to 400. Those Z values were obtained using safe lower bounds on Z based on the constituent encoders and the structure of the interleavers. The true Z values are likely much higher. Future work includes finding tighter lower bounds for Z, so the complexity can be reduced even further.

This method described above is the fastest known distance measurement method for tail-biting turbo codes that use structured interleavers. In [112], Rosnes reduced significantly the complexity of Garello's true distance measurement method [27]. Combining this new method with that of Rosnes [112] will further reduce the execution times.

Table 5.1 Minimum distances, multiplicities and CPU times in minutes for the DVB-RCS turbo-code encoder with the MPEG-sized standard interleaver.

R_c	1/3	2/5	1/2	2/3	4/5
$d_{\min}$	33	27	19	12	9
$A_{d_{\min}}$	376	376	376	188	3572
$W_{d_{\min}}$	3384	3384	3384	1316	20680
T_{OLD}	351	353	120	52	240
$T_{\text{NEW}}(Z=3)$	6.95	7.18	2.10	1.66	3.65
$T_{\text{NEW}}(Z=150)$	3.36	3.00	0.91	0.38	2.25

Table 5.2 Minimum distances, multiplicities and CPU times in minutes for the DVB-RCS turbo-code encoder with new MPEG-sized DRP interleavers.

R_c	1/3	2/5	1/2	2/3	4/5
$d_{\min}$	40	30	22	14	10
$A_{d_{\min}}$	1128	1504	3760	188	7332
$W_{d_{\min}}$	7332	9024	28388	1692	41924
T_{OLD}	10153	751	482	854	1215
$T_{\text{NEW}}(Z=3)$	482	29	14	17	22
$T_{\text{NEW}}(Z=150)$	270	1.80	2.91	7.20	10.88

Table 5.3 Minimum distances, multiplicities and CPU times in minutes for the UMTS turbo-code encoder with new MPEG-sized DRP interleavers.

R_c	1/3	2/5	1/2	2/3	4/5
$d_{\min}$	51	38	28	14	9
$A_{d_{\min}}$	940	376	1692	376	2068
$W_{d_{\min}}$	7708	2256	9588	1692	10152
T_{OLD}	302400	129600	34560	1421	504
$T_{\text{NEW}}(Z=7)$	12108	6468	1353	17	6.3
$T_{\text{NEW}}(Z=200)$	5578	908	651	10.35	3.53

5.5 Conclusion

A new and very efficient distance measurement method for tail-biting turbo codes that use structured interleavers has been presented. This new method takes advantage of the interleaver structure and the circular property of tail-biting. In this way, the search space for Garello's true method is reduced drastically, which in turn results in a significant reduction in complexity. The efficiency of this method was demonstrated for both single- and double-binary turbo codes, using structured interleavers that yield high minimum distances for various code rates. The execution times were reduced by a factor of 40 to 400. This means much larger interleavers with distances higher than 51 can be tested using this true $d_{\min}$ measurement method.

Chapter 6

Conclusions

This chapter summarizes the research achievements of the thesis and suggests some directions for future research work. A list of publications made during the course of this thesis is provided.

6.1 Research Achievements

In this thesis, efficient distance measurement methods have been introduced for both single- and double-binary turbo codes that use proper trellis termination such as dual-termination or tail-biting. Furthermore, various techniques that improve error performance and lower the decoding complexity have been introduced for the digital video broadcasting with return channel via satellite (DVB-RCS) standard double-binary turbo codes. The research achievements made during the course of this thesis are the following:

- Efficient distance measurement method for dual-terminated and tail-biting turbo codes based on Garello's true minimum distance method. This extended method reduces the computational complexity of Garello's true method by a factor of 2. This reduction in computational complexity is significant because the determination of minimum distances higher than 51 can easily take months (or even years) on a fast computer. This method has been applied to the double-binary tail-biting turbo codes used in the DVB-RCS standard. Minimum distances and multiplicities were determined for all twelve DVB-RCS standard interleavers and seven code rates.

- New interleavers have been designed for all DVB-RCS standard block sizes and code

rates. These interleavers have been designed based on the dithered relative prime (DRP) approach using the distance measurement method mentioned above. From a practical point of view, DRP interleavers are memory efficient because they can be stored and generated on the fly using only a few parameters instead of storing all the indices of the interleaver. The minimum distances of the new DRP interleavers are as good or better than those for the standard interleavers, for all the packet sizes and code rates. For rate-1/3, the new DRP interleavers designed for ATM and MPEG packet sizes achieve an improvement of at least 0.15 dB and 0.4 dB, respectively, at FERs below 10^{-6} compared to the standard interleavers. Thus, the new DRP interleavers designed for ATM and MPEG are the best ones known in the literature.

- Improving the reliability of Crozier's double-impulse iterative decoding method for distance measurement. The reliability of this new method has been investigated for various code rates using different types of interleavers such as random interleavers, DVB-RCS standard interleavers and DRP interleavers. Distance results show that this improved method is much more reliable than other iterative methods such as Berrou's error-impulse method or Garello's all-zero iterative decoding method. In fact, the improved method provides the true minimum distance most of the time. This method has (a) low complexity compared to Garello's true method and (b) provides a true upper bound on the minimum distance. These two features make it a very powerful tool for designing good interleavers in reasonable time. This is because bad interleavers (i.e., low minimum distances) can be rejected easily and only good interleavers (i.e., potentially high minimum distances) are retained to be evaluated using a true distance measurement method.

- A new and very efficient distance measurement method for tail-biting turbo codes that use structured interleavers such as DRP interleavers and the DVB-RCS standard interleavers has been introduced. This method takes advantage of (a) the structure of interleavers, (b) the circular property of tail-biting and (c) the number of consecutive zeros between error event(s). The accurate determination of $d_{\min} = 51$ and its corresponding true multiplicities for MPEG-sized interleaver would not be possible in reasonable time without the use of the new method. The efficiency of this method has been demonstrated for both single- and double-binary turbo codes, using MPEG-sized structured interleavers that yield high minimum distances for various

code rates. The execution times have been reduced by a factor of 40 to 400. This significant reduction in execution times will enable the testing of longer interleavers that yield minimum distances higher than 51 in acceptable time.

- Extended efficient decoding techniques, previously applied to single-binary, to double-binary to improve the performance of DVB-RCS standard double-binary turbo codes. These techniques are practical because they do not increase the decoding complexity. These techniques improve upon the known log maximum *a posteriori* (MAP) decoding and max-log MAP decoding, and are referred to as enhanced log MAP (EL-MAP) and enhanced max-log MAP (EML-MAP) decoding, respectively. The EL-MAP and EML-MAP decodings have been compared to the log MAP (L-MAP) and max-log MAP (ML-MAP) decodings using the DVB-RCS standard interleavers for the asynchronous transfer mode (ATM) (424 information bits) and the motion picture experts group (MPEG) (1504 information bits) packet sizes. For rate-1/3, EL-MAP has an improvement of about 0.1 dB at frame error rates (FERs) below 10^{-4} compared to ML-MAP, for both packet sizes. For rate-1/3, EML-MAP has an improvement of about 0.2 dB at moderate signal-to-noise ratios (SNRs) compared to ML-MAP, for both packet sizes.

- Extended an early stopping stopping technique, previously applied to single-binary, to double-binary to significantly reduce the average decoding complexity of the DVB-RCS standard turbo codes without degradation in error performance. The reduction in complexity is achieved by allowing the decoder to stop early, before reaching the maximum number of iterations. This technique is useful for simulation purposes, where the determination of reliable low error rates is very computationally intensive (i.e., it can take months even when a very fast decoder is used).

6.2 Future Work

To extend the work presented in this thesis, the following topics can be considered for future work. The suggested directions for future research work are summarized below.

- Since the execution time for the true distance measurement method presented in Chapter 5 depends strongly on the true value of the minimum distance, the determination of minimum distance can become impractical for long interleavers since they

can offer the greatest potential in achieving high minimum distances. However, it has been shown in Chapter 5 that a significant reduction in execution time can be achieved for tail-biting turbo codes that use structured interleavers if one takes advantage of the number of consecutive zeros, Z, preceding or following an error event in an input sequence that causes the minimum distance. Since (a) the minimum distance grows approximately with the base-3 logarithm of the interleaver size [116] and (b) Z increases with interleaver size, the use of a tight lower bound on Z will reduce the execution time further allowing the testing of long interleavers in acceptable time. Thus, the development of methods that provide a tight lower bound on Z is important for lowering the average execution time.

- A modification of Berrou's error-impulse method has been used in [117] to determine the minimum distance of low-density parity-check (LDPC) codes [118]. As shown in Chapter 4, Crozier's double-impulse method determines the minimum distance of turbo codes most of the time. A similar approach with two or more impulses is also expected to work well for LDPC codes. Thus, it is of interest to apply Crozier's multiple-impulse methods to LDPC codes and compare the reliability and complexity to that of the method presented in [117].

- A method that reduces significantly the complexity of Garello's true distance measurement method was presented by Rosnes in [112]. A new and efficient method that exploits the structure of interleaver and the circular property of tail-biting has been presented in Chapter 5. This new method is also based on Garello's true method. It is interesting to combine this new method with that of Rosnes [112]. This will tremendously reduce the complexity and thereby allows the test of long interleavers with high minimum distances in much shorter time than the known methods.

6.3 Contribution to the Literature

Conference Papers

1. Y. Ould-Cheikh-Mouhamedou, P. Guinand, and P. Kabal, "Enhanced Max-Log-APP and enhanced Log-APP decoding for DVB-RCS," *Proc. 3rd Int. Symp. turbo codes*, September 2003, pp. 259-262, (Brest, France).

2. Y. Ould-Cheikh-Mouhamedou, S. Crozier and P. Kabal, "Distance Measurement Method For Double Binary Turbo Codes and A New Interleaver Design For DVB-RCS," *Proc. IEEE Globecom*, November 29 - December 3 2004, p. 7, (Dallas, Texas, USA).

3. Y. Ould-Cheikh-Mouhamedou, S. Crozier and P. Kabal, "Comparison of Distance Measurement Methods for Turbo codes," 9^{th} *Canadian Workshop on Inform. Theory (CWIT'05)*, June 2005, pp. 36-39, (Montreal, Quebec, Canada).

Submitted Journal Papers

1. Y. Ould-Cheikh-Mouhamedou, S. Crozier and P. Kabal, "Efficient Distance Measurement Method for Turbo Codes That use Structured Interleavers," *IEEE Commun. Letters*, Submitted in June, 2005.

References

[1] D. R. Stinson, *Cryptography - Theory and Practice*. CRC Press, 2002.

[2] G. C. Clark and J. B. Cain, *Error-Correction Coding for Digital Communications*. Plenum Press, 1988.

[3] S. Haykin, *Communication Systems*. John Wiley & Sons, 2001.

[4] B. Sklar, *Digital Communications, Fundamentals and Applications*. Prentice Hall PTR, 2001.

[5] I. M. Jacobs, "Practical applications of coding," *IEEE Trans. Inform. Theory*, vol. 20, p. 305310, May 1974.

[6] G. Ungerboeck and I. Csajka, "On improving data-link performance by increasing the channel alphabet and introducing sequence coding," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT'76)*, June 1976. (Ronneby, Sweden).

[7] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. 28, pp. 55–67, Jan. 1982.

[8] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets, part I," *IEEE Commun. Mag.*, vol. 25, pp. 5–11, Feb. 1987.

[9] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets, part II," *IEEE Commun. Mag.*, vol. 25, pp. 12–21, Feb. 1987.

[10] W. Zhang, *Finite State Systems in Mobile Communications*. Ph.D. Dissertation, University of South Australia, Feb. 1996.

[11] J. Du and M. Kasahara, "Improvements in the information-bit error rate of trellis coded modulation systems," *Trans. of IEICE*, vol. E-72, pp. 609–614, May 1989. (Japan).

[12] E. Biglieri, D. Divsalar, P. J. McLane, and M. K. Simon, *Introduction to Trellis-Coded Modulation with Applications*. Macmillan, 1991.

[13] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, 1983.

[14] J. G. Proakis, *Digital Communications*. McGraw Hill, 2001.

[15] European Telecommunications Standards Institute, *Interaction channel for satellite distribution systems*. ETSI EN 301 790, V1.3.1, Mar. 2003.

[16] C. E. Shannon, "A mathematical theory of communications, part I," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[17] C. E. Shannon, "A mathematical theory of communications, part II," *Bell System Technical Journal*, vol. 27, pp. 623–657, 1948.

[18] B. Vucetic and J. Yuan, *Turbo Codes: Principles and Applications*. Kluwer, 2000.

[19] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*. Wiley, 1965.

[20] S. Dolinar, D. Divsalar, and F. Pollara, "Code performance as a function of block size," in *JPL, The Telecommunications and Mission Operations Progress Report: Technical Report 42-133*, pp. 1–23, May 15 1998.

[21] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun. (ICC'93)*, pp. 1064–1070, May 1993. (Geneva, Switzerland).

[22] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Commun.*, vol. 44, pp. 1261–1271, Oct. 1996.

[23] S. T. Brink, "Rate one-half code for approaching the Shannon limit by 0.1dB," *Electronics Letters*, vol. 36, pp. 1293–1294, July 2000.

[24] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 909–926, May 1998.

[25] 3^{rd} *Generation Partnership Project (3GPP) Technical Specification Group: Universal Mobile Telecommunications System (UMTS); Multiplexing and Channel Coding (FDD), TS 25.212 v3.4.0*. Sept. 2000.

[26] S. Crozier and P. Guinand, "High-performance low-memory interleaver banks for turbo-codes," in *Proc.* 54^{th} *IEEE Vehicular Technology Conf. (VCT'01)*, pp. 2394–2398, Oct. 2001. (Atlantic City, New Jersey, USA),(www.crc.ca/fec).

[27] R. Garello, P. Pierleoni, and S. Benedetto, "Computing the free distance of turbo codes and serially concatenated codes with interleavers: Algorithms and applications," *IEEE J. on Selected Areas Commun.*, vol. 19, pp. 800–812, May 2001.

[28] Y. Ould-Cheikh-Mouhamedou, S. Crozier, and P. Kabal, "Distance measurement method for double binary turbo codes and a new interleaver design for DVB-RCS," in *Proc. IEEE Globecom*, Nov. 29-Dec. 3 2004. (Dallas, Texas).

[29] E. Rosnes and O. Ytrehus, "An efficient algorithm for tailbiting turbo code weight distribution calculation," in *Proc.* 3^{rd} *Int. Symp. turbo codes*, pp. 439–442, Sept. 2003. (Brest, France).

[30] Y. Ould-Cheikh-Mouhamedou, S. Crozier, and P. Kabal, "Comparison of distance measurement methods for turbo codes," in 9^{th} *Canadian Workshop on Inform. Theory (CWIT'05)*, pp. 36–39, June 2005. (Montreal, Quebec, Canada).

[31] S. Crozier, P. Guinand, and A. Hunt, "Computing the minimum distance of turbo-codes using iterative decoding techniques," in *Proc.* 22^{nd} *Biennial Symp. Commun.*, pp. 306–308, May 31-June 3 2004. (Kingston, Ontario, Canada),(www.crc.ca/fec).

[32] S. Crozier, P. Guinand, and A. Hunt, "Estimating the minimum distance of turbo-codes using double and triple impulse methods," *IEEE Commun. Letters*, vol. 9, pp. 631–633, July 2005.

[33] C. Berrou, S. Vaton, M. Jézéquel, and C. Douillard, "Computing the minimum distance of linear codes by the error impulse method," in *Proc. IEEE Globecom*, pp. 10–14, Nov. 2002. (Taipei, Taiwan).

[34] K. Gracie, S. Crozier, and P. Guinand, "Performance of an MLSE-based early stopping technique for turbo codes," in *Proc.* 60^{th} *IEEE Vehicular Technology Conf. (VCT'04)*, Sept. 2004. (Los Angeles, California).

[35] Y. Ould-Cheikh-Mouhamedou, S. Crozier, and P. Kabal, "Efficient distance measurement method for turbo codes that use structured interleavers," *IEEE Commun. Letters*, Submitted in June 2005.

[36] Y. Ould-Cheikh-Mouhamedou, P. Guinand, and P. Kabal, "Enhanced Max-Log-APP and enhanced Log-APP decoding for DVB-RCS," in *Proc.* 3^{rd} *Int. Symp. turbo codes*, pp. 259–262, Sept. 2003. (Brest, France).

[37] S. Crozier, P. Guinand, and A. Hunt, "On designing turbo-codes with data puncturing," in 9^{th} *Canadian Workshop on Inform. Theory (CWIT'05)*, pp. 32–35, June 2005. (Montreal, Quebec, Canada).

[38] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. IEEE Press, Piscataway, N. J., 1999.

[39] 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, pp. 1101–1105, Sept. 1998.

[40] M. S. C. Ho, S. S. Pietrobon, and T. Giles, "Improving the constituent. codes of turbo encoders," *IEEE Globecom*, vol. 6, pp. 3525–3529, Nov. 1998.

[41] S. T. Brink, "Iterative decoding trajectories of parallel concatenated codes," in *Proc. of the* 3^{rd} *Int. ITG Conf. on Source and Channel Coding (SCC)*, pp. 75–80, Jan. 2000. (Munich, Germany).

[42] J. Hokfelt, "On the design of turbo codes," in *Ph.D. Dissertation*, Aug. 2000. Lund University, Sweden.

[43] D. Divsalar and F. Pollara, "Multiple turbo codes for deep-space communications," in *JPL, TDA progress report 42-121*, May 15 1995.

[44] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. IEEE Int. Conf. Commun. (ICC'95)*, pp. 54–59, June 1995. (Seattle, WA).

[45] S. Dolinar and D. Divsalar, "Weight distribution for turbo codes using random and nonrandom permutations," in *JPL, TDA progress report 42-122*, pp. 56–65, August 15 1995.

[46] J. Yuan, B. Vucetic, and W. Feng, "Combined turbo codes and interleaver design," *IEEE Trans. Commun.*, vol. 47, pp. 484–487, Apr. 1999.

[47] W. Feng, J. Yuan, and B. Vucetic, "A code-matched interleaver design for turbo codes," *IEEE Trans. Commun.*, vol. 50, pp. 926–937, June 2002.

[48] S. Crozier, J. Lodge, P. Guinand, and A. Hunt, "Performance of turbo codes with relative prime and golden interleaving strategies," in *Proc. of the* 6^{th} *Int. Mobile Satellite Conf. (IMSC '99)*, pp. 268–275, June 1999. (Ottawa, Ontario, Canada),(www.crc.ca/fec).

[49] O. Y. Takeshita and D. J. Costello, Jr., "New classes of algebraic interleavers for turbo-codes," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT'98)*, p. 419, Aug. 1998.

[50] F. Daneshgaran and M. Mondin, "Design of interleavers for turbo codes based on a cost function," in *Proc.* 1^{st} *Int. Symp. turbo codes*, pp. 255–258, Sept. 1997. (Brest, France).

[51] A. K. Khandani, "Design of the turbo-code interleaver using Hungarian method," *Electronics Letters*, vol. 34, pp. 63–65, Jan. 1998.

[52] J. Hokfelt, O. Edfors, and T. Maseng, "Interleaver design for turbo codes based on the performance of iterative decoding," in *Proc. IEEE Int. Conf. Commun. (ICC'99)*, pp. 93–97, June 1999. (Vancouver, BC, Canada).

[53] S. Crozier, "New high-spread high-distance interleavers for turbo codes," in *Proc.* 20^{th} *Biennial Symp. Commun.*, pp. 3–7, May 2000. (Kingston, Ontario, Canada),(www.crc.ca/fec).

[54] J. Hokfelt, O. Edfors, and T. Maseng, "A survey on trellis termination alternatives for turbo codes," in *Proc. IEEE Vehicular Technology Conf. (VTC'99)*, pp. 2225–2229, May 1999. (Houston, Texas).

[55] P. Guinand and J. Lodge, "Trellis termination for turbo encoders," in *Proc.* 17^{th} *Biennial Symp. Commun.*, pp. 389–392, May 30-June 1 1994. (Queens University, Kingston, Canada) , (www.crc.ca/fec).

[56] S. Crozier, P. Guinand, J. Lodge, and A. Hunt, "Construction and performance of new tail-biting turbo codes," in *Proc. of the* 6^{th} *Int. Workshop on Digital Signal Processing Techniques for Space Applications (DSP'98)*, Sept. 1998. (Estec, Noordwijk, Netherlands),(www.crc.ca/fec).

[57] C. Berrou, C. Douillard, and M. Jézéquel, "Multiple parallel concatenation of circular recursive convolutional (CRSC) codes," *Annals Telecommun.*, vol. 54, pp. 166–172, March-April 1999.

[58] J. Sun and O. Y. Takeshita, "Extended tail-biting schemes for turbo codes," *IEEE Commun. Letters*, vol. 9, pp. 252–254, Mar. 2005.

[59] I. Land and P. Hoeher, "Partially systematic rate 1/2 turbo codes," in *Proc.* 2^{nd} *Int. Symp. turbo codes*, pp. 287–290, Sept. 2000. (Brest, France).

[60] S. Crozier and P. Guinand, "Distance upper bounds and true minimum distance results for turbo-codes designed with DRP interleavers," in *Proc.* 3^{rd} *Int. Symp. turbo codes*, pp. 169–172, Sept. 2003. (Brest, France).

[61] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[62] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. 13, pp. 260–269, Apr. 1967.

[63] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technology*, vol. COM-19, pp. 751–772, Oct. 1971.

[64] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 218–278, Mar. 1973.

[65] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and suboptimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun. (ICC'95)*, pp. 1009–1013, June 1995. (Seattle, WA).

[66] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.

[67] G. Battail, "Ponderation des symboles décodés par lalgorithme de Viterbi," in *Annales Télécommunic*, vol. 42, pp. 31–38, Jan. 1987.

[68] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Globecom*, pp. 1680–1686, Nov. 1989. (Dallas, Texas).

[69] M. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and max-log-MAP decodings," *IEEE Commun. Letters*, vol. 2, pp. 137–139, May 1998.

[70] L. Cong, C. Long, and W. Xiaofu, "Further results on the equivalence between SOVA and max-log-MAP decodings," in *Proc. IEEE ICCT 2000*, vol. 2, pp. 1689–1692, Aug. 2000.

[71] P. Robertson, "Improving decoder and code structure of parallel concatenated recursive systematic (turbo) codes," in *IEEE Trans. of Int. Conf. on Universal Personal Commun.*, pp. 183–187, Sept. 1994. (San Diego, CA).

[72] J. Hagenauer, E. Offer, and L. Papke, "Iterative (turbo) decoding of systematic convolutional codes with the MAP and SOVA algorithms," in *Proc. of Int. ITG Conf. on Source and Channel Coding (SCC)*, pp. 1–9, Oct. 1994. (Munich, Germany).

[73] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Proc. IEEE Globecom*, pp. 1298–1303, Nov.-Dec. 1994. (San Francisco, CA).

[74] P. Robertson and P. Hoeher, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *European Trans. Telecommun.*, vol. 8, pp. 119–125, Mar.-Apr. 1997.

[75] A. S. Barbulescu, J. A. Torres, F. Hirzel, and V. Demjanenko, "Turbo codes 2000," *VOCAL Technologies, white paper: Temporary Document CF-036*, 08-12 jan 2001. (Clearwater, Florida).

[76] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structures for ISI channels," *IEEE Trans. Commun.*, vol. 42, pp. 1661–1671, Feb./Mar./Apr. 1994.

[77] M. Reed and J. Asenstorfer, "A novel variance estimator for turbo-code decoding," in *Int. Conf. Telecommun. (ICT'97)*, pp. 173–178, Apr. 1997. (Melbourne, Australia).

[78] T. A. Summers and S. G. Wilson, "SNR mismatch and online estimation in turbo decoding," *IEEE Trans. Commun.*, vol. 46, pp. 421–423, Apr. 1998.

[79] M. Jordan and R. Nichols, "The effects of channel characteristics on turbo code performance," in *Proc. IEEE Military Commun. Conf.*, pp. 17–21, Oct. 1996.

[80] A. Worm, P. Hoeher, and N. Wehn, "Turbo-decoding without SNR estimation," *IEEE Commun. Letters*, vol. 4, pp. 193–195, June 2000.

[81] J. B. Anderson and A. Svensson, *Coded modulation systems*. Kluwer, 2003.

[82] J. G. Proakis and M. Salehi, *Communication System Engineering*. Prentice Hall, 1994.

[83] S. G. Wilson, *Digital Modulation and Coding*. Prentice-Hall, 1996.

[84] K. Gracie, A. Hunt, and S. Crozier, "MLSE-based early stopping for turbo codes - finite quantization effects," in 9^{th} *Canadian Workshop on Inform. Theory (CWIT'05)*, pp. 231–234, June 2005. (Montreal, Quebec, Canada).

[85] P. J. Lee, "Constructions of rate $(n-1)/n$ punctured convolutional codes with minimum required SNR criterion," *IEEE Trans. Commun.*, vol. 36, pp. 1171–1174, Oct. 1988.

[86] C. Berrou, M. Jézéquel, C. Douillard, and S. Kérouédan, "The advantages of non-binary turbo codes," in *Proc. IEEE Inform. Theory Workshop*, pp. 61–63, Sept. 2001. (Cairns, Australia).

[87] C. Douillard, M. Jézéquel, and C. Berrou, "The turbo code standard for DVB-RCS," in *Proc.* 2^{nd} *Int. Symp. turbo codes*, pp. 551–554, Sept. 2000. (Brest, France).

[88] J. Lodge, R. Young, P. Hoeher, and J. Hagenauer, "Using seperable MAP 'filters' for the decoding of product and concatinated codes," in *Proc. IEEE Int. Conf. Commun. (ICC'93)*, pp. 1740–1745, May 1993. (Geneva, Switzerland),(www.crc.ca/fec).

[89] M. R. Soleymani, Y. Gao, and U. Vilaipornsawai, *Turbo Coding for Satellite and Wireless Communications*. Kluwer Academic Publishers, 2002.

[90] L. Papke and P. Robertson, "Improved decoding with the SOVA in a parallel concatenated (turbo-code) scheme," in *Proc. IEEE Int. Conf. Commun.*, pp. 102–106, June 1996. (Dallas, TX).

[91] A. Hunt, "Hyper-codes: High-performance low-complexity error-correcting codes," in *Master's Thesis*, May 1998. (Carleton University, Ottawa, Ontario, Canada),(www.crc.ca/fec).

[92] S. Crozier, A. Hunt, K. Gracie, and J. Lodge, "Performance and complexity comparison of block turbo-codes, hyper-codes and tail-biting convolutional codes," in *Proc.* 19^{th} *Biennial Symp. Commun.*, pp. 84–88, May 31-June 3 1998. (Kingston, Ontario, Canada),(www.crc.ca/fec).

[93] J. Vogt and A. Finger, "Improving the max-log-map turbo decoder," *Electronics Letters*, vol. 36, pp. 1937–1939, Nov. 2000.

[94] J. Hokfelt, O. Edfors, and T. Maseng, "Turbo codes: Correlated extrinsic information and its impact on iterative decoding performance," in *Proc. IEEE Vehicular Technology Conf. (VTC'99)*, pp. 1871–1875, May 1999. (Houston, Texas).

[95] K. Gracie, S. Crozier, and A. Hunt, "Performance of a low-complexity turbo decoder with a simple early stopping criterion implemented on a SHARC processor," in *Proc. Sixth Int. Mobile Satellite Conf.*, pp. 281–286, June 1999. (Ottawa, Canada).

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

[97] R. Garello and A. Vila, "The all-zero iterative decoding algorithm for turbo code minimum distance computation," in *Proc. IEEE Int. Conf. Commun. (ICC'04)*, pp. 361–364, June 2004. (Paris, France).

[98] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, pp. 591–600, May 1996.

[99] M. Breiling and J. Huber, "Combinatorial analysis of the minimum distance of turbo codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2737–2750, Nov. 2001.

[100] S. Crozier and P. Guinand, "Distance bounds and the design of high-distance interleavers for turbo-codes," in *Proc.* 21^{st} *Biennial Symp. Commun.*, pp. 10–14, June 2002. (Kingston, Ontario, Canada),(www.crc.ca/fec).

[101] F. Daneshgaran and M. Mondin, "An efficient algorithm for obtaining the distance spectrum of turbo codes," in *Proc.* 1^{st} *Int. Symp. turbo codes*, pp. 251–254, Sept. 1997. (Brest, France).

[102] P.-C. Yeh, A. O. Yilmaz, and W. Stark, "On the error floor analysis of turbo codes: Weight spectrum estimation (wse) scheme," in *Proc. of the Int. Symp. on Inform. Theory*, p. 439, June 2003. (Yokohama, Japan).

[103] M. Breiling and J. Huber, "A method for determining the distance profile of turbo codes," in *Proc. of the* 3^{rd} *Int. ITG Conf. on Source and Channel Coding (SCC)*, pp. 219–224, Jan. 2000. (Munich, Germany).

[104] E. Rosnes and O. Ytrehus, "On algorithms for determination of turbo code weight distribution," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT'02)*, p. 82, June 2002. (Lausanne, Switzerland).

[105] E. Rosnes and O. Ytrehus, "Improved algorithms for high rate turbo code weight distribution calculation," in *Proc.* 10^{th} *Int. Conf. Telecommun. (ICT'03)*, pp. 104–111, Feb. 2003. (Papeete, French Polynesia).

[106] J. Anderson and S. Hladik, "MAP tailbiting decoders," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT'97)*, p. 224, June 29-July 4 1997.

[107] Y.-P. Wang, R. Ramesh, A. Hassan, and H. Koorapaty, "On MAP decoding for tail-biting convolutional codes," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT'97)*, p. 225, June 29-July 4 1997.

[108] H.-A. Loeliger, "New turbo-like codes," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT'97)*, p. 109, June 29-July 4 1997.

[109] J. Anderson and S. Hladik, "Tailbiting MAP decoders," *IEEE J. on Selected Areas Commun.*, vol. 16, pp. 297–302, Feb. 1998.

[110] S. Crozier and P. Guinand, *High-performance low-memory interleaver banks for turbo-codes*. U.S. Patent 6,857,087, Feb. 2005.

[111] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, and M. Jézéquel, "Designing good permutations for turbo codes: towards a single model," in *Proc. IEEE Int. Conf. Commun. (ICC'04)*, pp. 341–345, June 2004. (Paris, France).

[112] E. Rosnes and O. Ytrehus, "Improved algorithms for the determination of turbo-code weight distributions," *IEEE Trans. Commun.*, vol. 53, pp. 20–26, Jan. 2005.

[113] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. 24, pp. 384–386, May 1978.

[114] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1757–1766, Nov. 1997.

[115] I. Dumer, D. Micciancio, and M. Sudan, "Hardness of approximating the minimum distance of a linear code," *IEEE Trans. Inform. Theory*, vol. 49, pp. 22–37, Jan. 2003.

[116] A. Perotti and S. Benedetto, "A new upper bound on the minimum distanceof turbo codes," *IEEE Trans. Inform. Theory.*, vol. 50, pp. 2985–2997, Dec. 2004.

[117] X.-Y. Hu, M. P. C. Fossorier, and E. Eleftheriou, "On the computation of the minimum distance of low-density parity-check codes," in *Proc. IEEE Int. Conf. Commun. (ICC'04)*, pp. 767–771, June 2004.

[118] R. G. Gallager, "Low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 8, pp. 21–28, Jan. 1962.