MAP Joint Source-Channel Arithmetic Decoding for Compressed Video

Hossein Kourkchi

A Thesis
In the Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montreal, Quebec, Canada

May 2018

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Hossein Kourkchi

Entitled: MAP Joint Source-Channel Arithmetic Decoding for Compressed Video

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical & Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Ketra Schmitt

_____ External Examiner
Dr. Stéphane Coulombe

_____ External to Program
Dr. Chun-Yi Su

_____ Examiner
Dr. M.N.S. Swamy

_____ Examiner
Dr. Wei-Ping Zhu

_____ Thesis Co-Supervisor
Dr. M. Omair Ahmad

_____ Thesis Co-Supervisor
Dr. William Lynch

Approved by _____
Dr. Wei-Ping Zhu, Graduate Program Director

Wednesday, July 4, 2018 _____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

ABSTRACT

MAP joint source-channel arithmetic decoding for compressed video

Hossein Kourkchi, Ph.D.
Concordia University, 2018

In order to have robust video transmission over error prone telecommunication channels several mechanisms are introduced. These mechanisms try to detect, correct or conceal the errors in the received video stream.

In this thesis, the performance of the video codec is improved in terms of error rates without increasing overhead in terms of data bit rate. This is done by exploiting the residual syntactic/semantic redundancy inside compressed video along with optimizing the configuration of the state-of-the art entropy coding, *i.e.,* binary arithmetic coding, and optimizing the quantization of the channel output. The thesis is divided into four phases.

In the first phase, a breadth-first suboptimal sequential maximum *a posteriori* (MAP) decoder is employed for joint source-channel arithmetic decoding of H.264 symbols. The proposed decoder uses not only the intentional redundancy inserted via a forbidden symbol (FS) but also exploits residual redundancy by a syntax checker. In contrast to previous methods this is done as each channel bit is decoded. Simulations using intra prediction modes show improvements in error rates, *e.g.,* syntax element error rate reduction by an order of magnitude for channel SNR of 7.33dB. The cost of this improvement is more computational complexity spent on the syntax checking.

In the second phase, the configuration of the FS in the symbol set is studied. The delay probability function, *i.e.*, the probability of the number of bits required to detect an error, is calculated for various FS configurations. The probability of missed error detection is calculated as a figure of merit for optimizing the FS configuration. The simulation results show the effectiveness of the proposed figure of merit, and support the FS configuration in which the FS lies entirely between the other information carrying symbols to be the best.

In the third phase, a new method for estimating the *a priori* probability of particular syntax elements is proposed. This estimation is based on the interdependency among the syntax elements that were previously decoded. This estimation is categorized as either reliable or unreliable. The decoder uses this prior information when they are reliable, otherwise the MAP decoder considers that the syntax elements are equiprobable and in turn uses maximum likelihood (ML) decoding. The reliability detection is carried out using a threshold on the local entropy of syntax elements in the neighboring macroblocks.

In the last phase, a new measure to assess performance of the channel quantizer is proposed. This measure is based on the statistics of the rank of true candidate among the sorted list of candidates in the MAP decoder. Simulation results shows that a quantizer designed based on the proposed measure is superior to the quantizers designed based on maximum mutual information and minimum mean square error.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my supervisors Dr. William E. Lynch and Dr. M. Omair Ahmad for the continuous support and directing my research, and for their patience, enthusiasm, motivations. Beside my supervisors, I would like to thank the professors with whom I interacted at Concordia.

I would like to thank my friends and colleagues for allowing me to share my experiences with them and for their spiritual support.

I would like to thank my brother for motivating and guiding me in my studies. Finally, I would like to dedicate this work to my parents for their continuous support throughout my life.

Table of Contents

List of Figures ..... **x**

List of Tables ..... **xv**

List of Symbols ..... **xvii**

List of Abbreviations ..... **xxii**

CHAPTER 1. Introduction **1**

1.1. Background 1

1.2. Literature review 3

1.3. Research objectives 10

1.4. Outline 12

CHAPTER 2. Predictive video coding overview and binary arithmetic coding with forbidden symbol **14**

2.1. Introduction 14

2.2. Overview on block structures of predictive video coders 14

2.3. *Intra*/*Inter* Prediction 20

2.3.1. *Intra* prediction 20

2.3.2. *Inter* prediction 26

2.4. Entropy coding 29

2.4.1. CAVLC 29

2.4.2. CABAC 30

2.5. Arithmetic coding (AC) 31

2.6. Binary arithmetic coding (BAC) with forbidden symbol (FS) 44

2.7. Summary 46

CHAPTER 3. Improving MAP arithmetic decoding of H.264 *intra* modes using residual redundancy 47

3.1. Introduction 47

3.2. MAP sequential binary arithmetic decoding 50

3.2.1. Metric-first algorithm (stack algorithm (SA)) 54

3.2.2. Depth-first algorithm 55

3.2.3. Breadth first algorithm (M-algorithm (MA)) 56

3.2.4. Comparing the searching techniques 57

3.3. MAP arithmetic decoding of H.264 *intra* modes using residual redundancy 58

3.3.1. MAP binary arithmetic decoding structure 59

3.3.2. Candidate generation and the decoding metric 60

3.3.3. Channel modeling 63

3.4. Simulations and results 65

3.5. Summary 71

CHAPTER 4. Optimizing the forbidden symbol configuration in joint source-channel arithmetic codes 73

4.1. Introduction 73

4.2. Binary arithmetic coding with generalized forbidden symbol configuration 77

4.2.1. Error detection delay 78

4.2.2. Probability of the missed error detection 83

4.3. Simulation results 89

4.4. Summary 92

CHAPTER 5. Improvement of the MAP decoder using interdependencies of syntax elements 94

5.1. Introduction 94

5.2. Using the interdependencies of syntax elements in MAP decoder 96

5.3. Transmitting *a priori* probability of *intra* modes as side information 102

5.4. PMF estimation of *intra* modes 104

5.4.1. Adaptive *intra* mode PMF estimation 104

5.4.2. Calculation of *intra* mode transition probability function 109

5.5. Adjusting the proposed method by local entropies 114

5.6. Simulation results 119

5.7. Summary 125

CHAPTER 6. Design of the quantizer on the channel output for joint source channel arithmetic decoding 127

6.1. Introduction 127

6.2. Available channel quantization designs 131

6.2.1. MMSE method 133

6.2.2. MMI method 137

6.3. Channel quantizer designs based on rank of true candidate 141

6.3.1. Dispersion of true candidate (DTC) 144

6.3.2. Probability of true candidate ranked zero or one..... 147

6.4. Simulation results..... 162

6.5. Summary 173

CHAPTER 7. Conclusion 175

7.1. Summary 175

7.2. Concluding remarks 177

7.3. Future works 179

References 181

List of Figures

Figure 2-1. Block diagram of block based predictive encoder. 16

Figure 2-2. Block diagram of block based predictive decoder. 16

Figure 2-3. Frame decomposition to five slices. 17

Figure 2-4. Syntax structure of H.264 20

Figure 2-5. Directional 4×4 *intra* prediction modes. 21

Figure 2-6. *Intra* predication modes and their index. 22

Figure 2-7. Directional 16×16 *intra* prediction modes. 22

Figure 2-8. *Intra* modes of the video *football,* (a) the 8^{th} frame, (b) the 9^{th} frame, (c) the 10^{th} frame. 24

Figure 2-9. *Intra* mode signaling process, (a) encoder, (b) decoder, (c) current mode at block X and its neighbors A, B, and C. 25

Figure 2-10. Macroblock partitioning in *inter* prediction mode. 26

Figure 2-11. Motion vector representation, (a) Three consecutive frames of video *foreman,* (b) motion vectors and (c) color maps of the direction and intensity of motion vectors. . 28

Figure 2-12. Overview of CABAC. 31

Figure 2-13. Arithmetic coding ESI subdivision. 33

Figure 2-14. The relation between EBI and ESI. 36

Figure 2-15. The relation between DBI and DSI. 39

Figure 2-16. The drifted DBI when the fourth received bit is in error. 40

Figure 2-17. Rescaling criteria in arithmetic coding for (a) E_0 (b) E_1 (c) E_2. 42

Figure 2-18. Overview of arithmetic encoding procedure. 43

Figure 2-19. Binary arithmetic encoding with forbidden symbol μ. 45

Figure 3-1. Error propagation in 3 consecutive frames by flipping only one bit of the compressed video stream Forman (a) Original video, (b) Error free decoded video, (c) Decoded erroneous data stream. 48

Figure 3-2. Block diagram of the coding and transmission system. 51

Figure 3-3. Decoding tree construction in sequential decoding technique. 54

Figure 3-4. Vertical and horizontal *intra* modes are impossible in blocks A and B respectively. 59

Figure 3-5. Block diagram of the MAP decoder..... 60

Figure 3-6. Sequential decoding tree for M-algorithm, M=2. 60

Figure 3-7. SEER versus SNR for three different MAP decoders and ε=0.1. 68

Figure 3-8. Decoding tree with length 40 for SNR=4.3232 dB, M=16 and ε=0.05. 69

Figure 3-9. Decoding tree with length 40 for SNR=4.3232 dB, M=16 and ε=0.1. 70

Figure 3-10. Decoding tree with length 40 for SNR=4.3232 dB, M=16 and ε=0.2. 70

Figure 3-11. Decoding tree with length 40 for SNR=4.3232 dB, M=16 and ε=0.4. 71

Figure 4-1. (a) Distributing and placing of the FS subintervals FS_1, FS_2 and FS_3 in the current ESI of length w_i, (b) The second step of iterative interval generation in binary arithmetic coding with distributed forbidden symbol. 78

Figure 4-2. Two types of interval drifts, (a) from valid interval to valid one and (b) from valid to invalid interval. 80

Figure 4-3. A plot of DPF versus n, value of delay, for $\epsilon = 0.1$ and $p_0 = 0.8$. 82

Figure 4-4. A plot of the PMD, $\pi(q_1, q_2)$, when $p_0 = 0.8$, $\epsilon = 0.1$ and $r = 20$..... 85

Figure 4-5. Optimum q_1, q_2 and q_3 versus p_0 for $\epsilon = 0.1$ and $r = 20$. 85

Figure 4-6. FS distribution after 4 levels of subdivision of DSI for (a) *FS at beginning* (b) *FS at end* and (c) *FS in middle* when $p_0=0.7$. 87

Figure 4-7. FS distribution after 4 levels of subdivision of DSI for (a) *FS at beginning* (b) *FS at end* and (c) *FS in middle* when $p_0=0.9$. 88

Figure 4-8. Plot of PER versus channel SNR for FS probability $\epsilon = 0.1$ and $p_0=0.9$. .. 90

Figure 4-9. Plot of PER versus channel SNR for FS probability $\epsilon = 0.1$ and $p_0=0.8$. ... 91

Figure 4-10. Plot of PER versus channel SNR for FS probability $\epsilon = 0.1$ and $p_0=0.7$. 91

Figure 4-11. Plot of PER versus channel SNR for FS probability $\epsilon = 0.1$ and $p_0=0.6$. . 92

Figure 5-1. Block diagram of the MAP decoder using *a priori* probability of syntax elements. 97

Figure 5-2. Spatially adjacent MBs to the current MB. 105

Figure 5-3. Transition between *intra* modes of current MB and adjacent MBs. 108

Figure 5-4. PMF of angular *intra* mode index difference. 114

Figure 5-5. PER and normalized entropy versus frame number for ML algorithm and MAP-Ad methods for video *foreman*. 116

Figure 5-6. PER and normalized entropy versus frame number for ML algorithm and MAP-Ad methods for video *football*. 117

Figure 5-7. The block diagram of the *intra* mode PMF estimation in MAP-Ad-Entp method with normalized entropy threshold *th%*. 119

Figure 6-1. Block diagram of the transmission system using the quantized channel. 132

Figure 6-2. Quantizer function with 4 decision levels, and conditional PDF of channel output given the sent bit for BPSK modulation. 132

Figure 6-3. Optimum Δ versus channel SNR for MMSE quantizer. 137

Figure 6-4. Transition of the equivalent channel of a quantized channel. 138

Figure 6-5. Optimum Δ versus channel SNR for MMI quantizer..... 141

Figure 6-6. DTC versus Δ for SNR = 5.208 dB and $j = 2,3, \dots ,7$. 145

Figure 6-7. Optimum Δ versus channel SNR for MDTC (j=2,3, …, 7), MMI and MMSE quantizers. 147

Figure 6-8. Calculation of RTC using decoding tree for $\mathbf{z}$=[r_1, r_4] given $\boldsymbol{u} = \mathbf{0}$. 150

Figure 6-9. Precise $P(RTC = 1)$ and its approximation, $\psi(\Delta, j)$, for channel SNR= 6.7895 dB. 159

Figure 6-10. Optimum Δ versus channel SNR for MDTC (j=2, 3,4,5), MPRTC1 (j=2, 3,4,5), MMI and MMSE quantizer designs. 162

Figure 6-11. Plot of PER versus Δ for various channel SNRs, M=128 and FS probability (a) $\epsilon = 0.1$ (b) $\epsilon = 0.05$. 165

Figure 6-12. Plot of PER versus Δ for various channel SNRs, M=64 and FS probability (a) $\epsilon = 0.1$ (b) $\epsilon = 0.05$. 166

Figure 6-13. Plot of PER versus Δ for various channel SNRs, M=32 and FS probability (a) $\epsilon = 0.1$ (b) $\epsilon = 0.05$. 167

Figure 6-14. Optimum Δ versus channel SNR for MPRTC1 (j=3, 4), MMI and MMSE methods and PER-Sim, for various M and, (a) $\epsilon = 0.1$ and (b) $\epsilon = 0.05$. 169

Figure 6-15. PER versus channel SNR for three methods fixed quantizer $\Delta = 0.3468$, adaptive quantizers $\Delta=\Delta_{opt}$:MMI and $\Delta=\Delta_{opt}$:MPRTC1(j=3) for (a) M=32, (b) M=64 and (c) M=128. 171

List of Tables

Table 2.1. Relation between *intra* modes (m) and their index (im) 22

Table 3.1. Error rates and average decoding time (T) for the algorithms *FS* Only, *FS+ Final SC*, and for the proposed method *FS+Full SC* at various values of probability of FS, various values of M, and at channel SNR = 6.7895 dB 67

Table 5.1. Comparing the performance of the ML decoder and MAP decoder uses transmitted *a priori* probabilities of *intra* modes 104

Table 5.2. *Corruption ratio* and *improvement percentag*e of MAP-Ad and MAP-Ad-Entp-th% with respect to ML algorithm in various values of the entropy thresholds for five different videos 121

Table 5.3. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 4.32dB 123

Table 5.4. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 5.20 dB 123

Table 5.5. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 6.78dB 124

Table 5.6. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 7.34dB 124

Table 5.7. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 8.39dB 125

Table 6.1. Conditions on $\boldsymbol{z}$ when RTC=1 for $j = 2$ and their probability 150

Table 6.2. $\boldsymbol{z}$'s results in RTC=1 for $j = 3$ and their probability given $\boldsymbol{u} = \boldsymbol{0}$ 151

Table 6.3. Optimum Δ and for various channel SNRs for MPRTC1(j=2,…,5), MMI and MMSE methods and PER-Sim, for M=32, M=64 and M=128 and $\epsilon = 0.05$ 173

List of Symbols

Ω	Absolute angular difference between *intra* mode indexes
$\boldsymbol{n}$	Additive white Gaussian noise (AWGN)
ψ	Approximation of $P(RTP = 1)$
κ	Average spacing of the data points
$\boldsymbol{s}_{\tilde{\mathbf{u}},k}$	Bin sequence generated by decoding the k^{th} bit of $\tilde{\mathbf{u}}$
$\boldsymbol{s}_{\tilde{\boldsymbol{u}}^j}$	Bin sequences corresponds to $\tilde{\boldsymbol{u}}^j$
b	Bit inversion position
$\boldsymbol{v}$	BPSK modulated signal
$\tilde{\mathbf{u}}$	Candidate of bit sequence
$\boldsymbol{y}$	Channel output
$\boldsymbol{z}$	Channel quantizer output
T	Computation time per packet
$\varphi(.)$	Cumulative distribution function
$\boldsymbol{r} = [r_1, \dots, r_{\ell_r}]$	Demodulated hard bit sequence
$\delta_d(.)$	Discrete Dirac function
γ	Dispersion of RTC away from zero
θ	DPF
ν'	Drifted value of ν

E_b	Energy per bit
$H(.)$	Entropy function
H_{MB_x}	Entropy of *intra* modes of MB_x
D	Error detection delay
$\hat{P}(.)$	Estimated PMF
$\hat{\boldsymbol{s}}$	Estimation of bin sequence $\boldsymbol{s}$
$\hat{\boldsymbol{u}}$	Estimation of bit sequence $\boldsymbol{u}$
$\hat{\boldsymbol{x}}$	Estimation of syntax element sequence $\boldsymbol{x}$
μ	Forbidden Symbol
$hist(.)$	Histogram function
$inmd(.)$	Index of *intra* mode a block
$indx(m)$	Index of *intra* mode m
$\beta(v, b)$	indicator function shows that altering the b^{th} bit of v results in a valid subinterval
$\alpha(v)$	Indicator function shows v reside in a valid subinterval
$mode(.)$	*intra* mode of a block
ℓ_s	Length of bin sequence
ℓ_u	Length of bit sequence
ℓ_x	Length of syntax element sequence
w_k	length of the ESI at the SSS$= k$
$\tilde{l}_j$	Lower limit of the EBI after j bits

l_k	Lower limit of the ESI
$\mathcal{E}$	Mean square error
$m_{\tilde{\boldsymbol{u}}}$	Metric of candidate $\tilde{\boldsymbol{u}}$
ℓ_j	Minimum length of $\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}$ for all path in stage j of decoding tree
$I(Z_i; U_i)$	Mutual information between Z_i and U_i
$\breve{H}_{MB_x}$	Normalized entropy of *intra* modes of MB_x
$th\%$	Normalized entropy threshold
ζ	Number of bit 1 in the first $j-1$ elements of $\tilde{\boldsymbol{u}}$
ς	Number of bit 1 in the first $j-2$ elements of $\tilde{\boldsymbol{u}}$
ρ	Number of bits repressing the intervals
F	Number of rescaling without emitting a bit
M	Number of saved candidates for M-Algorithm
π	PMD
P_0	Probabilities of binary symbol 0
P_1	Probabilities of binary symbol 1
$P(.)$	Probability mass function
δ	Probability of EOPS
ε	Probability of forbidden symbol
$\Delta^{(i)}$	Quantization parameter in i^{th} iteration
Δ	Quantization parameter of a 4 level symmetric quantizer
$\boldsymbol{x}_{\tilde{\mathbf{u}},k}$	SE sequence generated by decoding the k^{th} bit of $\tilde{\mathbf{u}}$

$\boldsymbol{s} = [s_1, \dots, s_{\ell_s}]$	Sequence of bins with length ℓ_s
$\boldsymbol{u} = [u_1, \dots, u_{\ell_u}]$	Sequence of bits with length ℓ_u
$\overline{\boldsymbol{x}}^{n-1}$	Sequence of first $n-1$ elements of $\overline{\boldsymbol{x}}$
$\boldsymbol{x} = [x_1, \dots, x_{\ell_x}]$	Sequence of syntax elements with length ℓ_x
$\mathcal{B}_{\ell_u}$	Set of all bit sequences with the length of ℓ_u
τ	Smoothing parameter
$\mathcal{A}$	Source symbol alphabet
ν	Stored value in the AC register
$[\![\overline{\boldsymbol{x}}^{n-1}]\!]$	Subset of $\overline{\boldsymbol{x}}^{n-1}$ whose elements are spatiotemporally adjacent with the SE $\overline{x}_n$
$Q(.)$	Tail probability of the standard normal distribution
$\overline{x}_n$	The n^{th} element of $\overline{\boldsymbol{x}}$
$\widetilde{\boldsymbol{u}}^j$	The First j bits of $\widetilde{\boldsymbol{u}}$
$\tilde{u}_k$	The k^{th} bit of $\widetilde{\mathbf{u}}$
y_k	The k^{th} bit of $\mathbf{y}$
ν_b	The b^{th} bit of ν
e	Threshold of difference between $\Delta^{(i)}$ and $\Delta^{(i+1)}$
ρ_k	Transition probability $k \in \{1,2,3,4\}$
$\overline{\boldsymbol{x}}$	Truncated SE sequence
$\overline{\boldsymbol{x}}_{\widetilde{\boldsymbol{u}}^j}$	Truncated SE sequence with length ℓ_j

$\tilde{h}_j$	Upper limit of the EBI after j bits
h_k	Upper limit of the ESI

List of Abbreviations

AC	Arithmetic Coding
AMVP	Advanced Motion Vector Prediction
APP	*A Priori* Probability
ARQ	Automatic Repeat reQuest
ASO	Arbitrary Slice Ordering
AVC	Advanced Video Coding
AWGN	Additive White Gaussian Noise
BAC	Binary Arithmetic Coding
BCJR	Bahl-Cocke-Jelinek-Raviv
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CBP	Coded Block Pattern
CDF	Cumulative Distribution Function
CM	Current Mode
CTU	Coding Tree Unit
CU	Coding Unit
DBI	Decoder Bit Interval

DCG	Demodulation and Candidates Generator
DCT	Discrete Cosine Transform
DTTL	Decoding Tree Tail Length
DPB	Decoded Picture Buffer
DPCM	Differential Pulse Code Modulation
DPF	Delay Probability Function
DSI	Decoder Symbol Interval
DTC	Dispersion of True Candidate
EBI	Encoder Bit Interval
EOPS	End of Packet Symbol
ESI	Encoder Symbol Interval
FEC	Forward Error Correction
FMO	Flexible Macroblock Ordering
FPS	Frames per Second
FS	Forbidden Symbol
FSM	Finite-State Machine
GSS	Golden Section Search
HD	High Definition
HEVC	High Efficiency Video Compression
HSV	Hue Saturation Value
JSCAC	Joint Source-Channel Arithmetic Coding

JSCD	Joint Source-Channel Decoding
JVT	Joint Video Team
LDPC	Low Density Parity Check
MA	M-Algorithm
MAP	Maximum *a posteriori*
MB	Macroblock
MDTC	Minimum Dispersion of True Candidate
MHP	Most Highly Probable
ML	Maximum Likelihood
MMI	Maximum Mutual Information
MMSE	Minimum Mean Square Error
MPM	Most Probable Mode
MPRTC	Maximum Probability of Rank Of True Candidate
MSE	Mean Square Error
MV	Motion Vector
NAL	Network Adaptation Layer
PER	Packet Error Rate
PMD	Probability of Missed Error Detection
PMF	Probabilities Mass Function
PPS	Picture Parameter Set
QCIF	Quarter Common Intermediate Format

QP	Quality Parameter
RTC	Rank of True Candidate
SA	Stack Algorithm
SC	Syntax Checker
SCR	Syntax Checking Rate
SEER	Syntax Element Error Rate
SER	Symbol Error Rate
SISO	Soft Input Soft Output
SM	Signaled Mode
SNR	Signal to Noise Ratio
SPS	Sequence Parameter Set
SSS	Source Symbol Stage
VCL	Video Coding Layer
VLC	Variable Length Coding

CHAPTER 1. Introduction

1.1. Background

According to a report from Cisco [1], in 2021 video content will account for more than 80 percent of all of the world's internet traffic, and the amount of video on demand traffic will be equivalent to 7.2 billion DVDs per month. This shows the importance of video compression to save channel resources.

Video compression standards are widely used in many video communication applications such as digital television, mobile TV, videoconferencing and internet video streaming. In all of the applications, the compressed video is always or likely to be transmitted through a communication channel which often introduces errors. These standards are also used for video storage on DVDs and NAND flash memories *etc*. Faults in reading process from the memory is inevitable.

Compressed video is fragile, that is, only a few bits in error can result in large areas of corruption in the decompressed video. In other words, errors are propagated. This corruption is in the form of partially or entirely missed/distorted video frame/frames. This phenomenon is even more noticeable in extremely noisy environments. An example of this is when the degradation in the path between the digital television transmitter and the viewer is high due to distance or environmental conditions. Another example is transmission of video signals over IP networks. Data loss may occur due to network issues such as

congestion in which a network node or link carries more data than its capacity [2] resulting in dropped packet. The reason behind this error propagation is the elimination of the redundancy by the compressor.

Video compression removes different types of redundancy, minimizing the degradation in the quality of video by rate-distortion optimization techniques [3]. Most video compression standards such as H.264 and high efficiency video coding (HEVC) remove the spatiotemporal redundancy using both predictive coding and transform coding [4]. *Intra* prediction removes the spatial redundancy inside a frame, and *inter* prediction removes the temporal redundancy between consecutive frames. Finally, an entropy coder reduces the remaining redundancy [3]. Context adaptive arithmetic coding (CABAC), a state-of-the-art entropy coding method, is used in H.264 and in its successor, HEVC [5]. Arithmetic coding (AC) maps a sequence of symbols to an interval using a recursive subdivision algorithm. The length of this interval is proportional to the probability of the symbol sequence when the symbols are ideally independent [6]. The number of bits per symbol generated by AC is exactly equal to the entropy of the source [7] if the symbols are independent.

Although, these procedures eliminate redundancy efficiently, the drawback of compression efficiency is the vulnerability of the compressed video stream to the presence of transmission errors. Notably, errors are propagated spatially and temporally in the decompressed video [8], [9] owing to both the predictive coding and the entropy coding. Furthermore, if the AC decoding loses synchronization, burst errors can result [10].

1.2. Literature review

There are several approaches to alleviate the effects of channel errors. The first approach is to only moderate the effect of errors without any effort to correct them, *e.g.,* partitioning, error detection and concealment [11]-[13]. The second is to detect erroneous data packets using error detection techniques and resend them. The third is to correct the errors using error correction codes like forward error correction (FEC) [14], although they impose overhead by adding redundancy to the video stream. On the other hand, in the Fourth approach, the video decompressor corrects the errors using the redundancy available in the compressed video stream which is called residual redundancy. In the following, these approaches are further elaborated.

The most straightforward way to moderate error propagation is partitioning, *i.e.,* subdividing the video into disjoint subsets and compressing each one independently [15]. Some partitioning algorithms are embedded as tools in the video coding standards such as data partitioning using slices and tiles, arbitrary slice ordering (ASO), and flexible macroblock ordering (FMO). Their goal is not only to isolate the errors but also to speed up the decoding process using parallel processing techniques [16]- [18]. However, partitioning techniques only minimize the effect of error propagation without any effort to correct the errors. Moreover, some of these algorithms are not widely used due to their high complexity and coding inefficiency, *e.g.*, FMO is not included in HEVC [5], [19].

In [13] the effect of errors is diminished through error concealment. In this method, the corrupted areas of the video are reconstructed from the uncorrupted areas by using

temporal and spatial interpolation algorithms [15]. However, error concealment does not correct the errors; it only moderates the perceptual effect of the errors.

The automatic repeat request (ARQ) protocol, the second type of error handling approach, is widely used for error control in communication systems, *e.g.,* global system for mobile communications (GSM) [20]. In this approach the decoder makes a request to the encoder to resend that packet of the data which is detected erroneous [21]. However, this technique is impractical in the applications in which only a one-way link is available, *e.g.,* broadcasting. Two-way links between encoder and decoder is required to establish this back and forth process.

The third approach, channel coding, is one of the most common ways to make data transmission resilient against channel noise [14]. In channel coding, a data sequence is mapped to another form by adding patterns of redundancy. However, channel coding decreases the error rate at the inevitable cost of bit-rate increase. This overhead is a heavy cost especially in very low data bit rate applications [22].

To address the problems in the above three approaches, joint source-channel decoding (JSCD) techniques have been introduced to take advantage of the residual redundancy in the compressed data to detect and also to correct the errors [23]-[25]. The JSCD technique may also use the redundancy intentionally added along with residual redundancy [26], [27]. Therefore, by exploiting residual redundancy, smaller amounts of intentionally added redundancy are required. Consequently, the JSCD techniques moderate the error rates while imposing less overhead to the video communication.

The JSCD algorithms are categorized to two main types, cooperative [28]-[31] and non-cooperative [23], [27], [32] and [33]. In cooperative methods, both the channel decoder and the source decoder exchange data for error correction, while in non-cooperative methods errors are corrected in the source decoder and the source decoder independently. Non-cooperative JSCD algorithms require only source decoding to exploit the residual redundancy.

In [28] and [29], a maximum *a posteriori* (MAP) cooperative decoder is introduced for joint source channel arithmetic codes (JSCAC). In this method, the conventional arithmetic decoder [6] is modified to a soft input soft output (SISO) decoder in order to use the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [34]. The SISO arithmetic decoder and a channel decoder are coupled in the turbo coding fashion [35]. In this approach, soft information between the source and channel decoders are transferred through a scrambler iteratively.

In [30] and [31], the cooperative JSCD scheme is adopted for CABAC in H.264. A list of candidates consisting of the most probable bit sequences are generated based on the channel decoder output, *i.e.,* a sequence of soft bits. The soft bit sequence is modified based on the best candidate conforming the H.264 syntax. The modified soft bit sequence is fed back to the channel decoder to generate a new soft bit sequence and this process continues for several iterations. To handle the computational complexity and the memory requirement, the decoder limits the number of the candidates in each iteration. This restriction results in reducing the efficiency of the error correction.

In the cooperative JSCD algorithms channel and source SISO decoder must be coupled. Thus, in the applications in which the SISO channel decoder is not available implementation of such algorithms is not appropriate. In such a case, non-cooperative JSCD algorithms are applicable.

In [32] a non-cooperative joint source channel arithmetic coding (JSCAC) algorithm is proposed only for error detection. In this algorithm, a forbidden symbol (FS) is added to the arithmetic coding symbol alphabet. This is done by assigning a portion of the AC interval to the FS. Errors are detected when an FS is received at the decoder. Dedicating a part of the symbol alphabet to the FS is a form of intentionally added redundancy. The amount of redundancy is controllable by the probability of the FS. Subsequently, this method is utilized in error correction of AC [36].

M. Grangetto and P. Cosman [37] applied the FS idea to MAP arithmetic decoding. In [33], the MAP arithmetic decoder is implemented bit-by-bit, by means of a binary decoding tree whose nodes correspond to the state of the AC after decoding bit stream candidates. However, this MAP algorithm is computationally complex because of the exponential growth of the tree with the increasing length of the data stream.

In [27], G. Olmo *et al* moderated the complexity of the MAP decoder using a suboptimal sequential search algorithm [38]. In the search algorithm, only part of the decoding tree is constructed, keeping only a limited number of the best nodes (with the highest $a\ posteriori$ probability as decoding metric) and discarding the rest. Likewise, in the elimination process the nodes resulting in the FS are eliminated. Consequently, the FS

also moderates the complexity by pruning the decoding tree. However, in these JSCAC methods the residual redundancy is not used for the error correction.

In [23], P. Duhamel *et al* introduced a MAP arithmetic decoder to exploit the residual redundancy in the binarization schemes of H.264. However, the decoder limits its scope to the binarizers, and it does not check the syntax elements, which are the basic elements of the video compression standard.

To correct errors using residual redundancy, JSCD generates candidates using the decoding tree and eliminates the candidates that violate the video compression standard syntax. The capability of error detection using this method in various parts of video decompression is studied in [8]. Depending on how often and which syntax is checked various methods have been introduced as explained below.

In [24], the MAP arithmetic decoder is modified by discarding the final nodes of the decoding tree that do not conform the syntax of H.264 using a syntax checker (SC). There are two problems with this method. Firstly, since the SC is active only at the final stage of the decoding tree, this can result in the decoder not discarding erroneous candidates in as timely a fashion as it could have. Secondly, errors close to the end of the data stream have reduced probability of detection. This is because of a delay to detect the desynchronization of the arithmetic decoder after an error.

A way to improve the performance of the MAP arithmetic decoder is to minimize the error detection delay, *i.e.,* the number of bits required to be decoded after an error happens to detect the FS. The lower the delay, the better the error detection probability. In [10], the error detection delay is modeled by a geometric distribution [39]. However, in this model

the effect of the FS subinterval location in the AC interval on the delay is not considered. Several works have been done to study the FS subinterval location summarized below.

In [40], instead of using only one FS subinterval, it is divided into several disjoint FS subintervals. There are various FS configurations depending on the position of FS subintervals in the AC interval. In this method, the effectiveness of the FS configuration is evaluated by the symmetry of the total lengths of the FS subintervals about the midpoint of the AC interval. However, the most symmetric FS configuration does not necessarily provide the minimum error detection delay.

A look-ahead technique for a faster error detection is proposed in [41], and the error detection delay of several FS configurations are also studied using simulations. It is shown that placing the FS subinterval between the other subintervals provides the best configuration among those studied for AC without the use of the look-ahead technique.

In [42], AC with FS is simplified and modeled by a finite-state machine (FSM) and its effectiveness is analyzed by the calculation of an upper bound for the bit error rate. This upper bound is used for optimizing the FS configuration. However, this optimization is only valid for the FSM model.

Another approach for outperforming the MAP decoder error resilience is to estimate the *a priori* probability as accurately as possible. This probability is part of the decoding metric, *i.e., a posteriori* probability. If the MAP decoder works in syntax element level, *a priori* probability of the syntax elements can be used. These probabilities can be calculated either at the encoder [43] or estimated at the decoder using the previously decoded data [44].

In [43], a first-order Markov process is used to model the statistics of the motion vectors in H.264. In this method the parameters of the statistical model for the motion vectors are calculated at the encoder and transmitted as side information to the decoder. However, this imposes an overhead on the system.

In [44], various syntax elements of H.264 are statistically modeled. The variable length code (VLC) [3] is used as entropy coder. Unlike AC, the encoder maps each symbol to a variable length code. As a result, the parameters of the model are extracted codeword by codeword from the previously decoded syntax elements. However, the statistical model may not always be reliable for all conditions, *e.g.,* non-stationary processes [39], and may mislead the decoder. Since this method works codeword by codeword, it is not applicable for AC.

In the MAP methods mentioned above, some have the decoder working with soft channel bits, and some with hard channel bits. Soft decoding results in better error resilience [27], but in order to process the soft bits using digital signal processing units, the channel output have to be quantized. Designing the channel quantizer to minimize the error rates is mathematically challenging. To overcome this problem several proxy measures have been introduced in the past as described below.

The most well-known measure for quantizer optimization is mean square error (MSE) [45]. This measure is widely used in quantization of multimedia signals like image and video [46]. However, minimum mean square error (MMSE) does not guarantee minimum error rate in the channel quantization application.

Instead of MSE, some information theoretic measures have been used for channel quantization design [47]-[50]. The communication channel and the quantizer are combined and considered as an equivalent discrete channel. The performance of the quantizer is evaluated by the equivalent channel characteristics like channel capacity, mutual information and channel cutoff rate [7]. However, in all of these methods, the effect of the quantization on the MAP decoder and in turn error rates is not taken into account in the design. Consequently, these measures do not guarantee minimum error rate.

Several aspects of JSCAD have been studied in the past as discussed in this section. Finally, several techniques have been introduced for the channel quantizer design. The drawbacks of the mentioned approaches and the proposed solutions are addressed in the next section, research objectives.

1.3. Research objectives

The objective of this thesis is to improve the performance of the MAP joint source channel arithmetic decoder with FS in terms of error rates without imposing the overhead in terms of data bit rate. This is done by efficiently applying the SC to exploit the residual redundancy along with optimizing the FS configuration to improve the performance of the MAP arithmetic decoder in terms of error rates. Further, the *a priori* probability of syntax elements are estimated. Finally, a proxy measure is introduced to design the channel quantizer. These objectives are achieved in four phases as explained below.

In the first phase, to decrease the overhead imposed by FS a MAP arithmetic decoder is proposed in which FS and SC are combined for decoding a particular syntax element, namely *intra* prediction modes. JSCD of this syntax element is studied because of two

reasons. Firstly, in [8] it is shown that the errors result in syntactic errors in the *intra* modes often. Secondly, correction of the errors in an *intra* frame not only benefits that frame directly, but also improves quality of those frames which use that frame as a reference in prediction procedure. In this method, FS and SC are used to eliminate invalid candidates not only at the end of the decoding tree (unlike [24]), but also throughout the development of the tree.

In the second phase, the FS configuration is optimized to reduce the error rate. Indeed, a new figure of merit that measures the probability of missed error detection (PMD) is proposed for analyzing the FS configuration. In determining this measure the error detection delay is considered as a function of FS configuration. This is unlike [39] where the delay is modeled independent of the FS configuration. The best FS configuration based on this measure is the one in which the FS subinterval is placed between the other subintervals, which is consistent with the simulation results for MAP decoder. This configuration prevents FS subintervals to be clustered. It is shown that the optimized FS configuration reduces the error rate considerably without imposing additional overhead to the system. The proposed measure can model the error detection delay for various FS configuration which can be used in designing the MAP decoding tree.

In the third phase, the *a priori* probability of syntax elements is calculated through the construction of the decoding tree. Unlike [44], AC is used as the entropy coder and the decoding tree is constructed channel bit by channel bit and *a priori* probabilities are calculated in each stage from the interdependencies in the previously decoded syntax elements. A method is introduced to overcome inaccurate estimates that mislead the MAP

decoder. In this method, when the previously decoded syntax elements are too diverse, the computed probabilities are deemed unreliable. The entropy of the previously decoded syntax elements is used as the diversity measure. In an unreliable case, the decoder neglects the *a priori* probabilities estimated.

The fourth phase of the thesis is dedicated to optimizing the channel quantizer. The quantizer is placed before the MAP arithmetic decoder, and it is designed to outperform the error resilience of the decoder. Unlike past methods, the MAP decoder metric and the construction process of the decoding tree is taken into account in the quantizer design. In this phase, new proxy measures for performance of the channel quantizer is proposed. The first measure is the dispersion of the true candidate (DTC) in the sorted list of candidates in the decoding tree. DTC is calculated using the second moment of the rank of the true candidate among the sorted candidates. However, calculating DTC is computationally complex specially for long bit sequences. The second proposed measure is the probability that the true candidate is among two best candidates. This measure is analytically determined and used to optimize the channel quantizer for MAP arithmetic decoder. Simulation results justify the effectiveness of the proposed measure.

1.4. Outline

The thesis is organized as follows. In the next section, Chapter 2, H.264 is briefly explained with more details given on some elements like *intra* and *inter* prediction, entropy coding, and arithmetic coding. Chapter 3 considers the MAP arithmetic decoder. In this section, improvement over the conventional MAP decoder by using the redundancy in syntax elements is provided. The binary arithmetic coding (BAC) with distributed FS and

the proposed figure of merit for FS configuration are presented in Chapter 4. The proposed MAP decoding using *intra* mode statistic estimation is proposed in Chapter 5. In Chapter 6, the proposed figure of merits for the quantizer optimization based on statistics of rank of true candidate is provided. Finally, the thesis summary, conclusion and future work are brought in Section 7.

CHAPTER 2. Predictive video coding overview and binary arithmetic coding with forbidden symbol

2.1. Introduction

Block based predictive video coding standards are for a wide range of applications from low complexity and low-bitrate mobile video applications to high definition (HD) broadcast services. The joint video team (JVT) finalized the video coding standard H.264 for formal approval submission as H.264/AVC (advanced video coding) [18] in March 2003. The latest edition of the standard was published in April 2017 [16]. The successor of H.264 is H.265/HEVC (high efficiency video compression) whose last version was released in December 2016 [51].

In this chapter, information about block based predictive video compression standards H.264 and HEVC are provided, and modules related to this work specifically the entropy coder are studied in detail. Section 2.2 provides a general overview on predictive video coding. Two types of prediction, *intra* and *inter*, are explained in Section 2.3, and Section 2.4 provides two different types of entropy coding. The entropy coding used in this work, arithmetic coding, is brought in Section 2.5, and finally the error resilient version of this entropy coding is explained in Section 2.6.

2.2. Overview on block structures of predictive video coders

Block based predictive video encoders, *e.g.,* H.264 and HEVC, consist of prediction, transform, quantization and source encoding blocks as shown in Figure 2-1. To reconstruct the decoded video sequence, complementary processes of source decoding, dequantization,

inverse transform and reconstruction are carried out by the video decoder as depicted in Figure 2-2.

Block based predictive video encoding takes advantage of predictive and transform coding simultaneously [4], [52]. The particular concern in predictive coding is error propagation, *i.e.,* when a single error occurs in compressed video stream it would be propagated to neighboring pixels or frames. In H.264 and HEVC, in order to minimize the effect of error propagation, every input frame to the encoder is decomposed into several slices [3], and every slice is encoded independently. Thus, every slice is decodable independent of the other slices (if any) of the same frame. Since there is no interdependency between slices, errors cannot be propagated from one slice to other slices. The number of slices and their shapes are arbitrarily chosen at the compressor side, as shown in Figure 2-3. Frame partitioning by slices not only is for enhancing the error robustness but also can be used for parallel processing purposes. In HEVC, to boost parallel processing capabilities, frames are partitioned to rectangular tiles by vertical and horizontal borders and each tile divided to slices. The tiles and their slices are independently encoded and in turn at the receiver can be decoded in parallel fashion.

In H.264, the fundamental element of a slice is the macroblock (MB) which is a 16×16 region of pixels. For prediction of a MB in a slice the predictor must limit its scope to the current slice in that frame. Thus, losing a slice does not affect decoding of the other slices in that frame. So, by using several slices and preventing the interdependencies between them, the effect of error propagation in decoder side can be minimized. In HEVC, the fundamental coding unit (CU) size is ranging from 8×8 to 64×64 pixels. Multiple CUs in

hierarchal fashion form a coding tree unit (CTU) whose size is 64×64. In other words, CU in HEVC is similar to MB in H.264 but its size is variable.

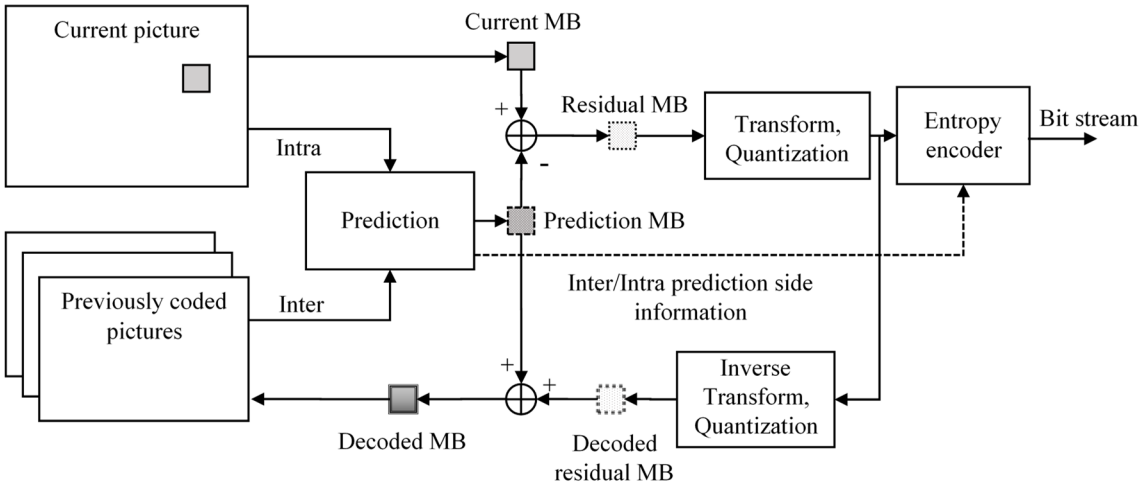


Figure 2-1. Block diagram of block based predictive encoder.

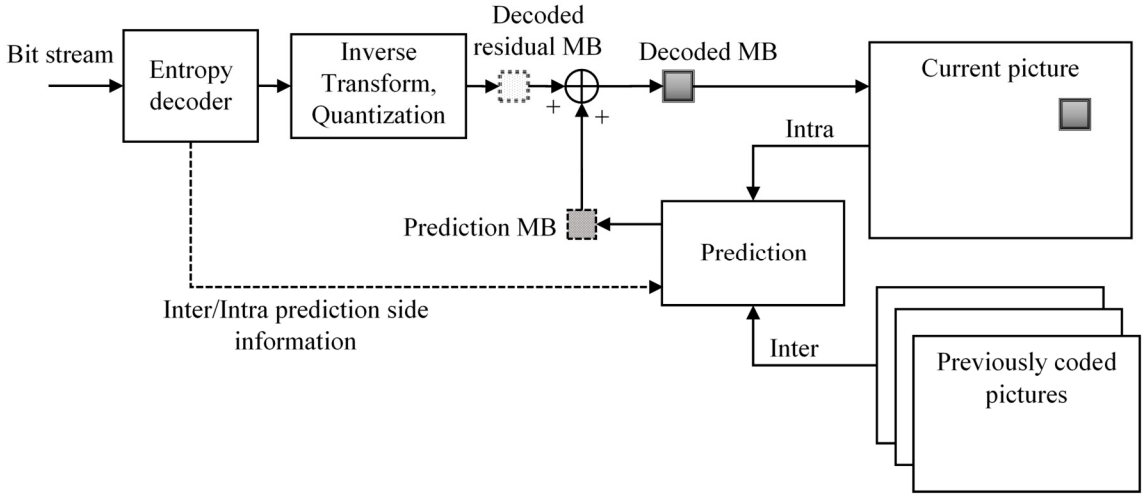


Figure 2-2. Block diagram of block based predictive decoder.

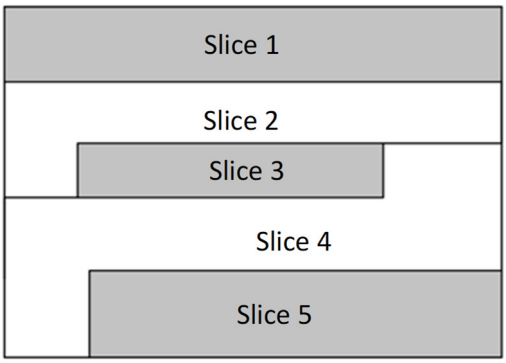


Figure 2-3. Frame decomposition to five slices.

In H.264, a prediction of every MB based on previously coded data is formed either from the current frame or from other frames that have already been coded and transmitted. The former is *intra* prediction and latter is *inter* prediction. The encoder calculates the residual data by subtracting the predicted MB from the original MB. Since adjacent macroblocks are temporally and spatially correlated, residual data is generally low energy and often is sparse in the transformed domain. The residue for each MB is divided into 4×4 or 8×8 blocks, and then transformed by using an integer transform, a scaled approximation to the discrete cosine transform (DCT). In certain cases, part of the output of this integer transform is further transformed using a Hadamard Transformation. The transform coefficients are scaled and quantized. The transformation and quantization are designed in a way to minimize the computational complexity for processors with limited integer precision. By using an integer transform core, the floating-point operations are avoided and the total number of multiplications is reduced [3].

In HEVC every CU can be partitioned down into 4×4 prediction blocks (PB) by recursive horizontal and vertical halving process. The same types of prediction, *i.e., intra* and *inter*, are available in HEVC similar to H.264. *Intra* PB subdivisions are symmetric while *inter* PB subdivisions are either symmetric or asymmetric.

The last step of the encoder is binarization and entropy coding which this work is concentrated on. There are two options for entropy coding in H.264. One of them is context adaptive variable length coding (CAVLC) and the other is context adaptive binary arithmetic coding CABAC. In HEVC, only CABAC is used as entropy coding [5]. CABAC provides better compression efficiency than CAVLC does. However, it has higher computational complexity and it is more fragile in the presence of noise [53], [18]. In this work arithmetic coding as the core of CABAC is used as the entropy coder.

A video sequence is represented in a hierarchal format called video coding syntax. The basic elements of this structure are syntax elements which determine different aspects of the coded sequence and the way that the element is coded. In Figure 2-4 an overview of the hierarchal structure of the H.264 syntax is shown.

At the highest level, a compressed video stream consists of a series of packets or network adaptation layer units, (NAL Units). Parameter sets, sequence parameter set (SPS) and picture parameter set (PPS), signal key parameters used by the decoder to correctly decode the video data and slices. The SPS contains information common for the entire video stream such as frame size, maximum number of reference frames *etc*. PPS is made up of common parameters that may apply to a sequence or subset of coded frames, such as

entropy coding type, number of active references. Video coding layer (VCL) NAL units contains video coding data.

At the next level, a slice represents all or part of a coded video frame and consists of a number of coded macroblocks. A slice header contains the frame number, the coded frame type, the number of the first macroblock in the slice, and other information required to resynchronize the parameters of the decoder and common to all macroblocks in the slice. The slice header is followed by the slice data containing compressed corresponding data of macroblocks.

At the lowest level, a macroblock contains macroblock type, macroblock prediction information, coded block pattern (CBP), quantization parameter (QP), and coded residual data. There are three main macroblock types, *I* (*intra* predictive), *P* (predictive), and *B* (bidirectional predictive). In type *I*, the current slice is used to predict the current macroblock in various *intra* modes. *Inter* prediction is used in types *P* and *B*. the prediction in type *P* is just based on the previous frames whether the prediction in type *B* is based on the past and future frames. The *inter* prediction information consists of motion vectors. Residual information consist of chrominance (Cb and Cr) and luminance (Y) parts. CBP indicates which transformed chrominance and luminance residual blocks are non-zero.

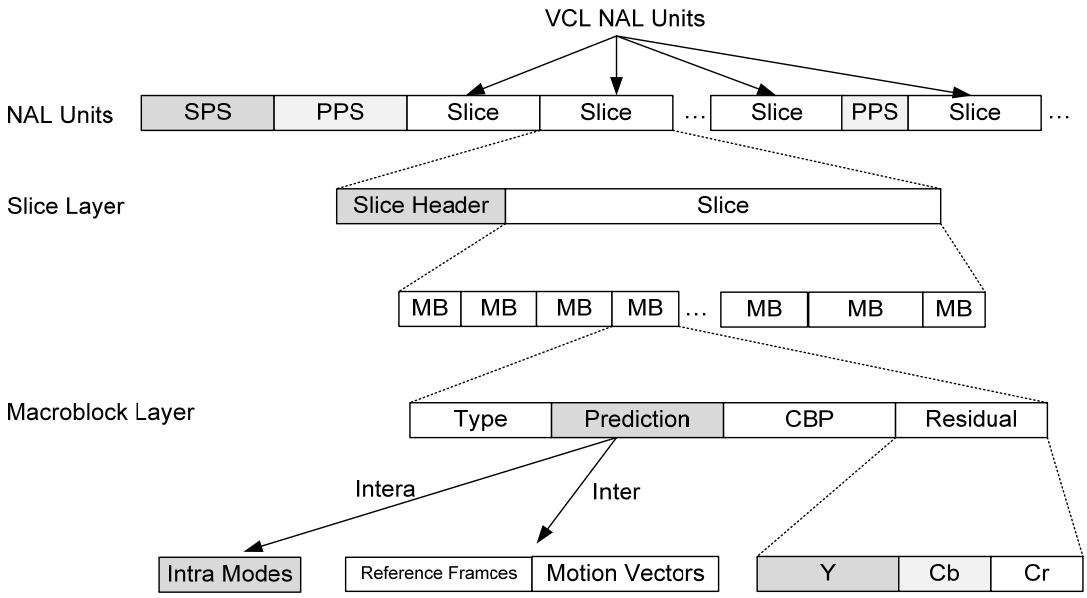


Figure 2-4. Syntax structure of H.264

2.3. *Intra*/*Inter* Prediction

The prediction can be done based on current frame or previously encoded/decoded frames [54]. Accordingly, two types of prediction, *intra* and *inter*, are possible which are explained in this section.

2.3.1. *Intra* prediction

In H.264, for *intra* prediction, each MB is partitioned into either sixteen 4×4 *intrablocks* or four 8×8 *intrablocks* or is left as one 16×16 *intrablock*. A smaller prediction block size (4×4) gives a more accurate prediction that translates to fewer bits to represent residual data. However, the choice of prediction for every 4×4 block must be signaled to the decoder requiring more bits to be transmitted.

In the 4×4 and 8×8 cases, each *intrablock* is predicted based on the previously compressed pixels from the same slice using one of 9 possible *intra* modes as depicted in Figure 2-5. The predicted samples *a-p* are formed by a linear combination of available prediction samples *A-M, e.g.,* if diagonal down-right mode is selected the sample d is predicted by $d = round(\frac{B}{4} + \frac{C}{2} + \frac{D}{4})$ [16]. The arrows indicate the direction of the prediction. *Intra* 8×8 modes are very similar to *Intra* 4×4 modes. Modes 0, 1 and 3 to 8 are directional modes and mode 2 is DC mode. As presented in Figure 2-6, the directional prediction modes are indexed from 0 to 7 in the order of their direction and the DC mode is indexed as the last mode, *i.e.,* mode 8th. The *intra* mode index is denoted by i_m which is between 0 and 8. The relation between *intra* modes and their index is represented in Table 2.1. The *intra* mode indexing is used in Chapter 5.

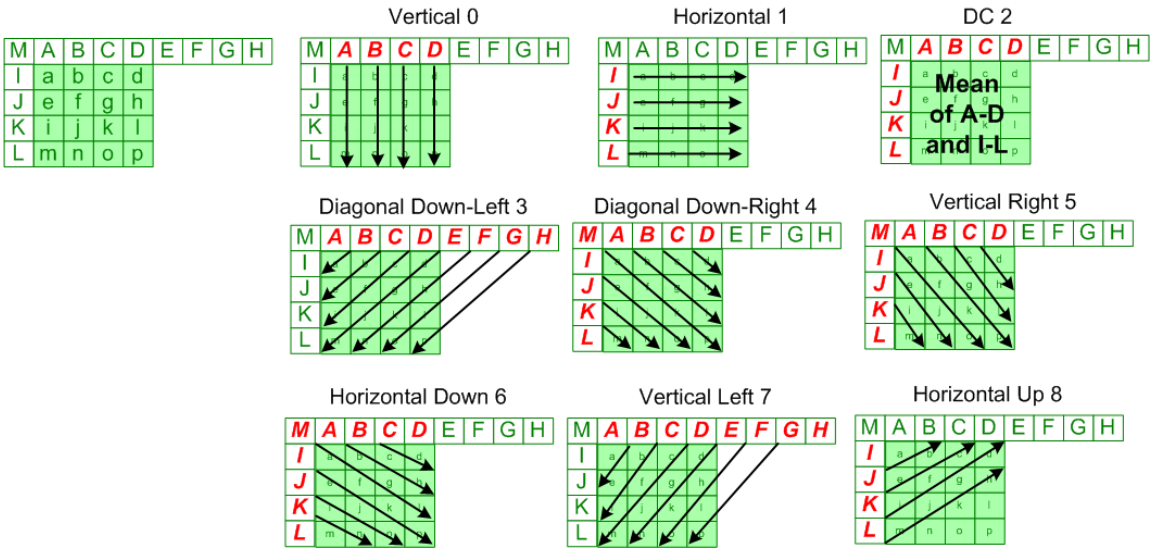


Figure 2-5. Directional 4×4 *intra* prediction modes.

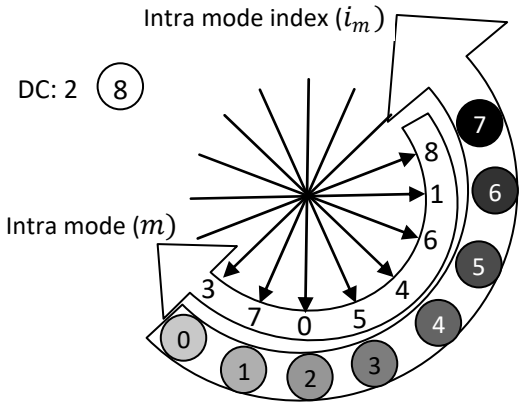


Figure 2-6. *Intra* predication modes and their index.

Table 2.1. Relation between *intra* modes (m) and their index (i_m)

m	0	1	2	3	4	5	6	7	8
i_m	2	6	8	0	4	3	5	1	7

In the 16×16 case, four directional modes are possible which are depicted in Figure 2-7. The prediction is done using a linear combination of available pixels on the top or left side of the MB.

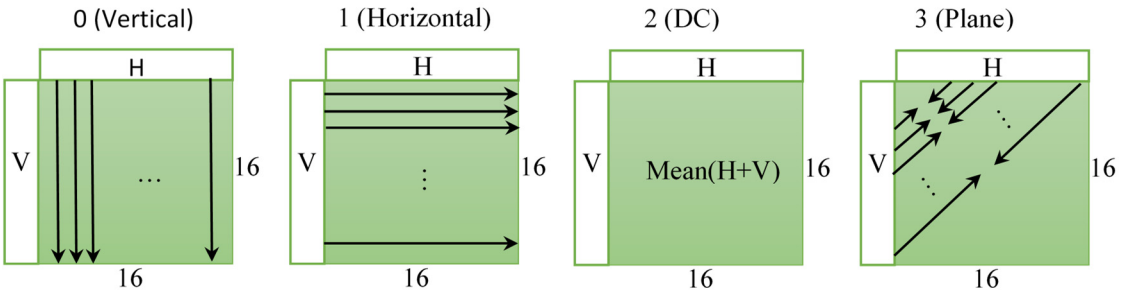


Figure 2-7. Directional 16×16 *intra* prediction modes.

The encoder attempts to choose the appropriate *intra* prediction modes to minimize the total number of bits for the prediction and the residual data, by a rate distortion mechanism [55]. The prediction modes and residual data are entropy encoded and sent in the bitstream.

In order to decrease interdependency between the slices, the predictor limits its scope to the current slice. This means that at any given *intrablock*, several modes may be unavailable as they require using pixels from a different slice. *e.g.*, the vertical mode cannot be used for blocks in the top edge of a slice or horizontal mode for blocks in the most left side of a slice. The invalidity of these modes can be used to eliminate erroneously decoded bitstreams. This error correction technique is explained in Chapter 3. In [8], it is shown that the syntax/semantic errors in *intra* prediction modes are among the most probable syntax errors.

In Figure 2-8 *intra* modes of the 8^{th}, 9^{th}, and 10^{th} frames, of video *football* (3fps) [56] is represented. Each gray color represents a directional *intra* mode as depicted in Figure 2-6. It can be easily seen that there is a noticeable correlation between *intra* modes of spatially and temporally adjacent blocks.

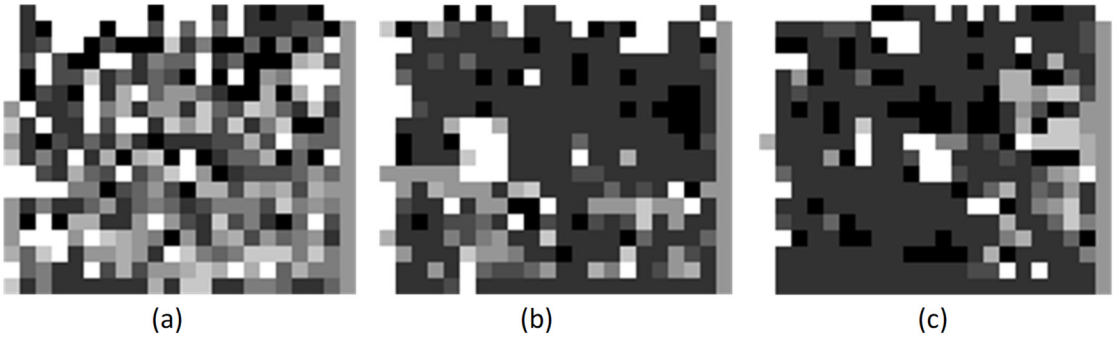


Figure 2-8. *Intra* modes of the video *football,* (a) the 8th frame, (b) the 9th frame, (c) the 10th frame.

In order to get rid of the redundancy caused by correlation of neighboring *intrablocks*, H.264 uses a predictive signaling method as shown in Figure 2-9. For each current mode (CM) at block X, the encoder calculates the most probable mode (MPM), defined as the minimum of prediction mode of block A and B. if either these adjacent blocks are unavailable, DC mode is set as the MPM. For every block the encoder sends a binary flag indicates that the MPM of block X is equal to the CM or not. Afterward, if the MPM is not as same as CM, it generates a 3-bits number, denoted as signaled mode (SM), which represents the difference between the MPM and CM. Since there are 9 *intra* modes, it seems that more than 3 bits are needed to signal the difference between MPM and CM. However, as the flag shows that MPM and CM are not equal this difference is between 1 and 9. Using a mechanism shown in Figure 2-9 this number is signaled using only a 3 bit number.

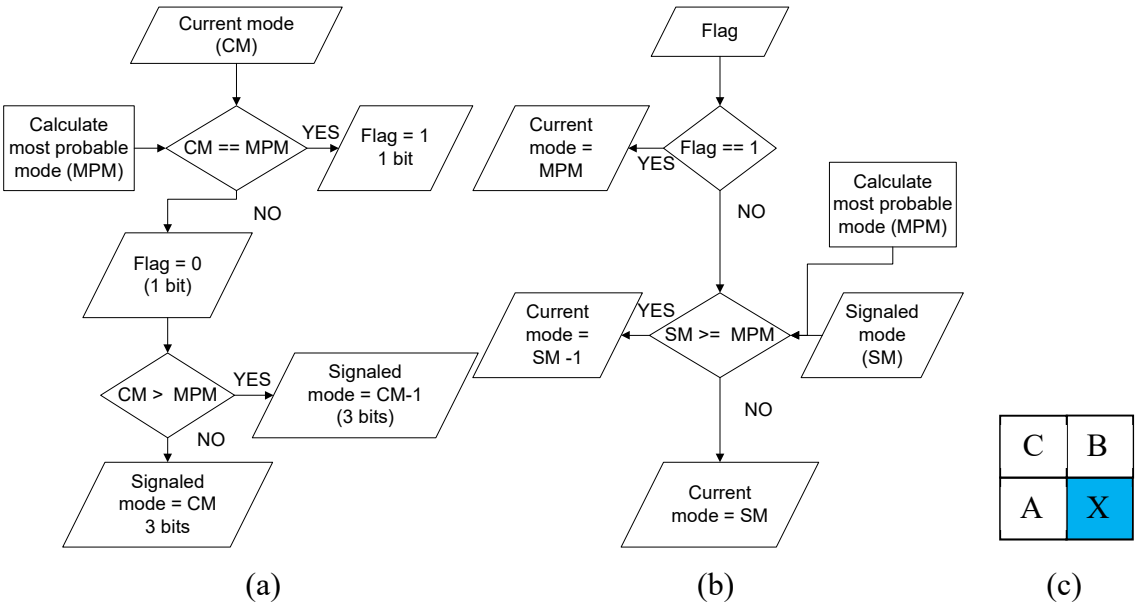


Figure 2-9. *Intra* mode signaling process, (a) encoder, (b) decoder, (c) current mode at block X and its neighbors A, B, and C.

In HEVC, *intra* PB size range is between 4×4 and 32×32. The number of directional modes is almost four times the number of *intra* modes in H.264, *i.e.,* 33 directional modes as well as a planar mode and a DC mode. While this increase provides better *intra* prediction especially for directional image texture, signaling the selected *intra* modes requires more bits to be transmitted. To moderate this overhead, more sophisticated *intra* mode signaling is used. In HEVC, 3 distinct MPMs are derived from *intra* modes of the top and left adjacent PUs to the current *intra* mode and then they are sorted. If the current *intra* mode is equal to one of them index of that MPM which is an integer between 0 and 2 is signaled, otherwise a 5-bit number representing the current *intra* mode minus 3 is signaled.

2.3.2. *Inter* prediction

In *inter* prediction a block of samples is predicted from a picture that has previously been coded and transmitted, *i.e.,* a reference picture (see Figure 2-1). This process is made up of selecting a prediction region, generating a prediction block and subtracting this from the original block of samples to form a residue. In H.264, an MB can be either partitioned into 16×8, 8×16 or 8×8 blocks or kept as a 16×16 MB as shown in Figure 2-10. The smaller the block size, the more precise the prediction. However, signaling data of each prediction separately is an overhead.

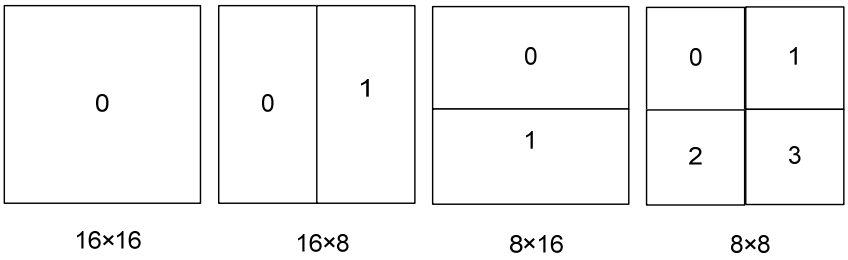


Figure 2-10. Macroblock partitioning in *inter* prediction mode.

The reference pictures are chosen from a decoded picture buffer (DPB), a list of previously coded pictures. This list consists of two lists, namely list 0 and list 1. List 0 and list 1 includes the pictures before and after the current picture in terms of display order respectively. In P (predictive) macroblocks the predictor uses only one picture from list 0, whereas in B (bi-directional) macroblocks both lists 0 and 1 are used. A weighted prediction may be utilized optionally according to the temporal distance between the current and reference picture(s) [3].

The offset between the position of the current partition and the prediction region in reference picture is a motion vector (MV). The precision of the MVs can be one-pixel or half-pixel or quarter-pixel position. In H.264, half-pixel and quarter-pixel position can be interpolated using the samples of the reference pictures. In Figure 2-11, the motion vector of three consecutive P frames of the typical video *foreman* is represented. In this representation the HSV (hue, saturation and value) color code is used. The hue and value components indicate the direction and length of the motion vector respectively. The brighter the block, the longer the MV. It can be easily seen that the MVs are highly correlated. This is because neighboring blocks belong to the same object moving together. Due to high interdependencies between MVs of adjacent partitions they are encoded differentially and transmitted alongside the residual information. H.264 does the MV prediction by calculating the median of three spatially adjacent MVs. As only one type of MV prediction is used in H.264, MV prediction signaling is not needed. In HEVC, more sophisticated MV prediction technique is used which is called advanced motion vector prediction (AMVP). AMVP derives two more probable MV among five spatially neighboring blocks and a temporally MV from two temporal collocated blocks, finally an index to the MV prediction is signaled.

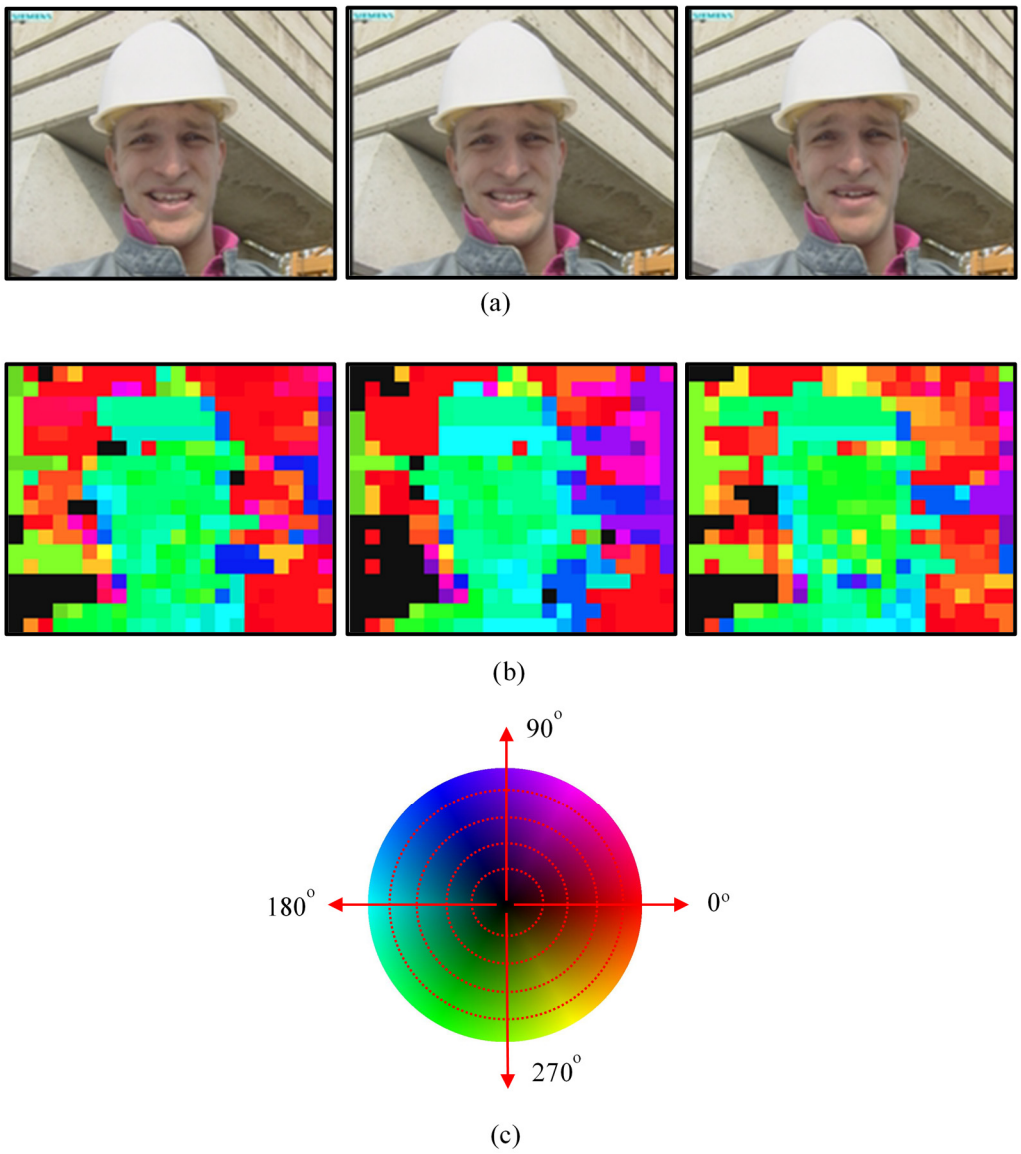


Figure 2-11. Motion vector representation, (a) Three consecutive frames of video *foreman,* (b) motion vectors and (c) color maps of the direction and intensity of motion vectors.

2.4. Entropy coding

The last module of video encoders is the entropy coding. The objective of this module is to reduce the redundancy in the encoded data stream using lossless data compression techniques. There are two entropy coding methods in H.264. One of them is context adaptive variable length coding (CAVLC) and the other is context adaptive binary arithmetic coding (CABAC). CABAC has better compression efficiency than CAVLC, although it has more computational complexity and it is more fragile against noise [57]. HEVC uses only CABAC as entropy encoder. This work focusses on CABAC and its source coding engine, *i.e.* arithmetic coding.

2.4.1. CAVLC

For encoding the residual data, they are firstly mapped into a one-dimensional list using a scanning pattern. This list is then coded using variable length coding (VLC). VLC maps input symbols to sequences of codewords. This assignment is based on a predefined table which assigns shorter codewords to more probable symbols. The optimum VLC encoder is Huffman coding [45] which uses the statistics of the symbols for the codeword assignments. Since the statistics of symbols vary temporally and spatially these tables should be changed adaptively. VLC tables are chosen adaptively based on the local statistics. Since representing every codeword needs at least 1 bit, this coding scheme cannot reach the entropy bound unless it encodes many symbols together [4] which is infeasible in CAVLC.

2.4.2. CABAC

CABAC entropy coding in H.264 consists of three major operations, binarization, context modeling and arithmetic coding (AC). The simplified block diagram of CABAC scheme is shown in Figure 2-12. Binarizer maps syntax elements to binary symbols (bins) which are forwarded for arithmetic encoding. CABAC changes the internal context model adaptively based on the previously encoded bins, the current syntax element type and its local statistics. The context model determines the bin probabilities used in arithmetic coding. Context models and binarization schemes for each syntax element are defined in the standard. There are about 300 context models for the various syntax elements [16].

Alternatively, the bypass coding engine may be used to speed up the encoding and decoding process by bypassing the context modeling and using faster version of the arithmetic coding, bypass coder.

In order to have efficient entropy coding, the context modeling block must estimate the conditional probabilities as accurately as and as fast as possible. Therefore, to fulfill this requirement alphabet reduction in CABAC is performed, by using the binarization block. Different binarization schemes are used for various syntax elements. There are four basic binarization types: the unary code, the truncated unary code, the k^{th} order Exp-Golomb code and the fixed-length code [57]. Since studying the binarization schemes is out of the scope of this work, it is not explained more here. In the next section, the arithmetic coding (AC) is explained in detail.

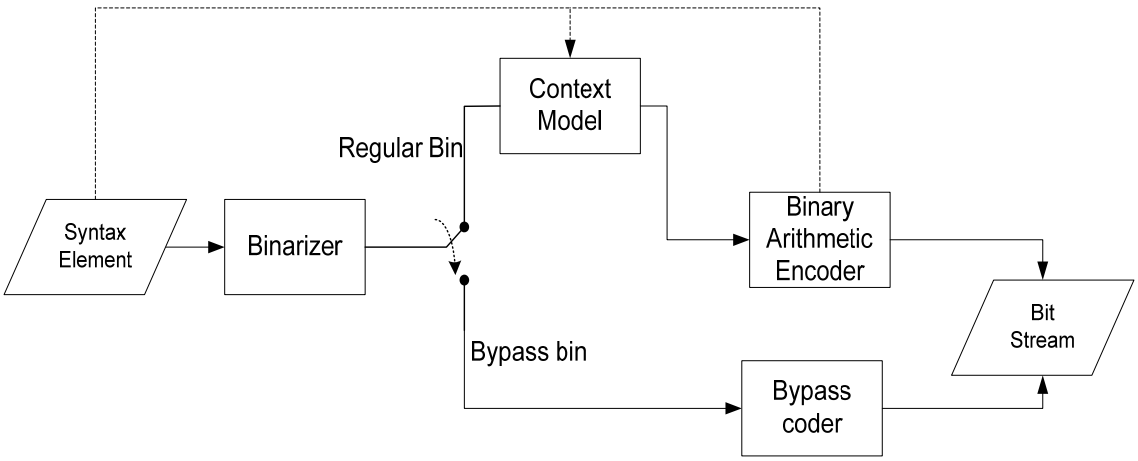


Figure 2-12. Overview of CABAC.

2.5. Arithmetic coding (AC)

The main notion of arithmetic coding was introduced by Shannon in 1948 [58]. This idea is to use the cumulative distribution function (CDF) of source symbols to generate a bit sequence with minimum length, known as Shannon-Fano coding [7]. This is done by mapping the symbols to intervals whose length is proportional to the symbols probability. The more likely the symbol, the larger the intervals and in turn smaller number of bits required.

Suppose the finite source symbol alphabet is $\mathcal{A} = \{a_1, \dots, a_N\}$ with the corresponding probabilities mass function (PMF) $P(a_i)$. The CDF is defined as $\varphi(i) = \sum_{k=1}^{i} P(a_k)$ for $1 \leq i \leq N$, and $\varphi(0) = 0$.

The objective of arithmetic coding is to map particular sequences of source symbols $\boldsymbol{s} = [s_1, \dots, s_{\ell_s}]$, $s_i \in \mathcal{A}$, with length ℓ_s symbols into disjoint intervals where the lengths of the intervals are proportional to the probability of the sequence of source symbols $\boldsymbol{s}$.

These are named Encoder Symbol Intervals (ESIs). The lower and upper limits of the ESI after encoding k symbols are denoted by l_k and h_k, respectively. The ESI is determined by a recursive procedure [36] described as follow,

$$\begin{aligned} l_0 &= 0 \\ h_0 &= 1 \\ l_k &= l_{k-1} + (h_{k-1} - l_{k-1}) \times \varphi(Indx(s_k) - 1) \\ h_k &= l_{k-1} + (h_{k-1} - l_{k-1}) \times \varphi\big(Indx(s_k)\big), \end{aligned} \tag{2-1}$$

where $Indx(s_k)$ returns the index of symbol s_k in the alphabet set $\mathcal{A}$.The calculation of l_k and h_k using s_k represents the k[th] stage of this process. This will be referred to as the source symbol stage (SSS). The difference between l_k and h_k , *i.e.,* $w_k = h_k - l_k$, is the length of the ESI at the SSS$= k$. It can be shown using the definition of the CDF that

$$P(s_k) = \frac{h_k - l_k}{h_{k-1} - l_{k-1}}. \tag{2-2}$$

Furthermore, if the input symbols s_k are independent then w_k is proportional to the probability of $\boldsymbol{s}^k$ defined as the first k symbols of $\boldsymbol{s}$**.** This recursive process is shown in Figure 2-13 for ternary arithmetic coding with symbol PMF $P(a_1) = 0.7$, $P(a_2) = 0.2$ and $P(a_3) = 0.1$. In this example, a sequence of symbols with the length of $\ell_s = 5$, $\boldsymbol{s} = [a_1, a_3, a_1, a_2, a_1]$, is encoded. The final ESI is shown by lower and upper limits $l_5 =$

0.6643 and $h_5 = 0.67116$ respectively. If the decoder receives any number in this interval the symbol sequence can be decoded unambiguously. Since less precision and in turn less number of bits are required to represent the larger intervals, assigning larger intervals to more probable symbols make the source coding scheme efficient.

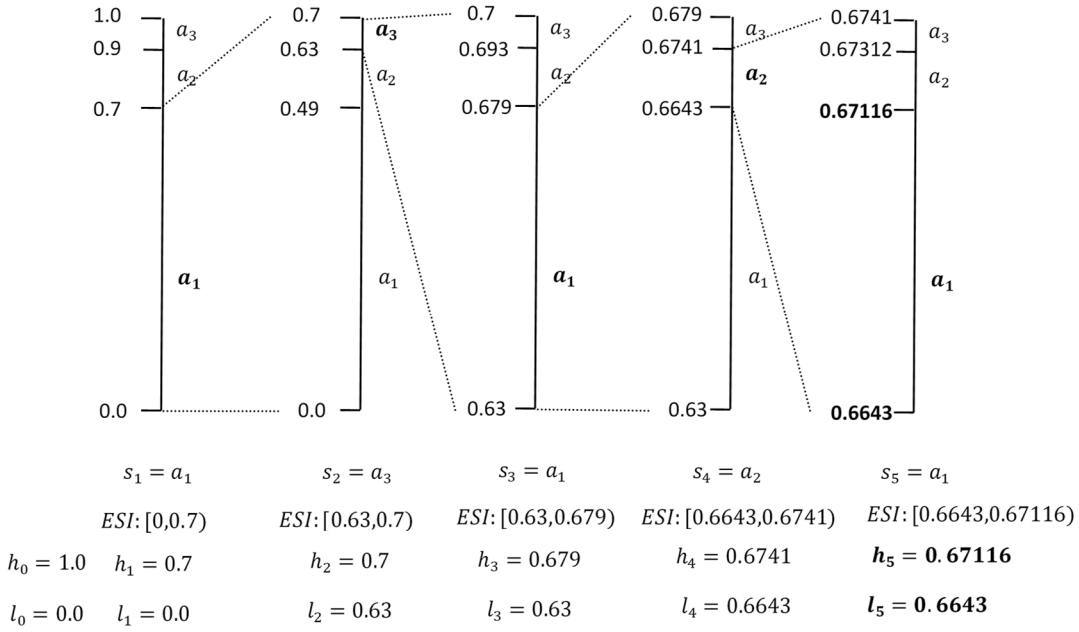


Figure 2-13. Arithmetic coding ESI subdivision.

To send the message represented by the ESI over a channel, it needs to be translated into a bitstream. This is done by creating a number in binary format inside the ESI and expressing it in sufficient precision such that one is certain that the only the current ESI could contain that number even if infinite precision were used.

To this end, the concept of an EBI is used. The EBI is an interval associated with the binary number being generated inside the ESI. When that number is expressed with *j* bits

of precision, the ESI is the interval of all real numbers that could be expressed starting with these j bits.

The lower and upper limits of the EBI after j bits are denoted by $\tilde{l}_j$ and $\tilde{h}_j$ respectively. The relation between EBI and ESI is represented in Figure 2-14. The encoder works symbols clock wise (*i.e.,* according to the SSS) and generates ESI symbol by symbol. At each SSS it selects the largest EBI (*i.e*. using the fewest number of leading bits) where the ESI is fully within the EBI. If at a particular SSS the largest such EBI becomes smaller the new bits expressing that smaller EBI are emitted to the channel. If at a particular SSS the new ESI is not within a smaller EBI then the previous SSS then the EBI remains the same as no bits are emitted. In Figure 2-14, EBI selection process is shown for the symbol sequence $\boldsymbol{s} = [a_1, a_1, a_1, a_3]$. The encoding process is summarized as the following pseudo code.

-**Step1**: (initialization)

Set $[l_0, h_0) = [0,1)$, $[\tilde{l}_0, \tilde{h}_0) = [0,1)$, $k = 0, j = 0$

-**Step2**: (ESI evolution)

$k = k + 1$;

Find $[l_k, h_k)$ from $[l_{k-1}, h_{k-1})$ and s_k based on (2-1)

-**Step3**: (EBI selection)

If $[l_k, h_k) \in [\tilde{l}_j, (\tilde{l}_j + \tilde{h}_j)/2)$ then

$j = j + 1;$

$\tilde{h}_j = (\tilde{l}_j + \tilde{h}_j)/2;$

Emit $u_j = 0$

Go to step 3

If $[l_k, h_k) \in [(\tilde{l}_j + \tilde{h}_j)/2, \tilde{h}_j)$ then

$j = j + 1$;

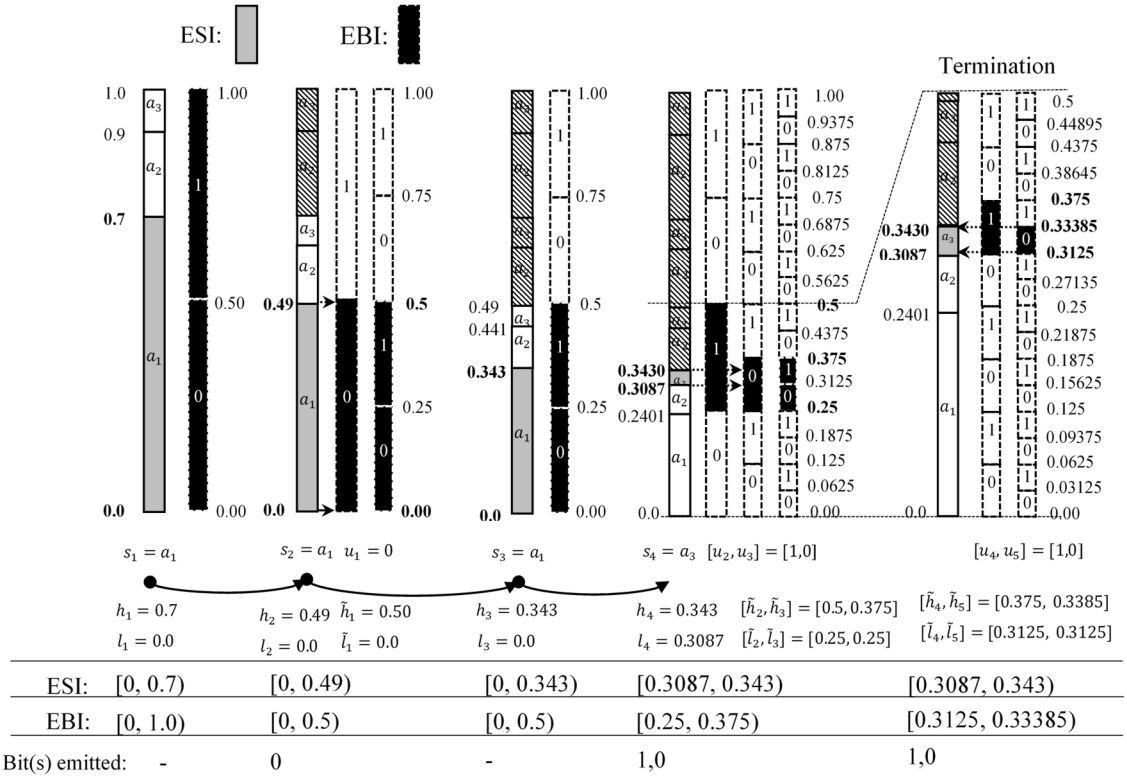
$\tilde{l}_j = (\tilde{l}_j + \tilde{h}_j)/2$;

Emit $u_j = 1$

Go to step 3

Go to Step 2.

To terminate this process, the encoder must send a number inside the final ESI, $[l_{\ell_s}, h_{\ell_s})$. Therefore, after generating the final ESI (in this example $l_4 = 0.3087, h_4 = 0.343$), the EBI gets finer until it resides thoroughly inside the last ESI.



ESI:	[0, 0.7)	[0, 0.49)	[0, 0.343)	[0.3087, 0.343)	[0.3087, 0.343)
EBI:	[0, 1.0)	[0, 0.5)	[0, 0.5)	[0.25, 0.375)	[0.3125, 0.33385)
Bit(s) emitted:	-	0	-	1,0	1,0

Figure 2-14. The relation between EBI and ESI.

The decoder deals with two kinds of interval, namely the decoder bit interval (DBI), and decoder symbol interval (DSI). With no channel errors the DBI is the same as the EBI. This will then result in the DSI being the same as the ESI with some latency. However, in a noisy channel environment, the received bit sequence is denoted by $\boldsymbol{r} = [r_1, \ldots, r_{\ell_r}]$ could be different from $\boldsymbol{u}$ due to alterations caused by channel noise. The DBI is generated using the recursive process as,

$$\tilde{l}'_0 = 0$$

$$\tilde{h}'_0 = 1$$

$$\tilde{l}'_j = \begin{cases} \tilde{l}'_{j-1} & : \quad r_j = 0 \\ \dfrac{\tilde{l}'_{j-1} + \tilde{h}'_{j-1}}{2} & : \quad r_j = 1 \end{cases}$$

$$\tilde{h}'_j = \begin{cases} \dfrac{\tilde{l}'_{j-1} + \tilde{h}'_{j-1}}{2} & : \quad r_j = 0 \\ \tilde{h}'_{j-1} & : \quad r_j = 1, \end{cases} \tag{2-3}$$

which $\tilde{l}'_j$ and $\tilde{h}'_j$ are the lower and upper limits of the DBI after receiving and decoding j bits. In Figure 2-15, this procedure is shown for received bit sequence $\boldsymbol{r} = [0,1,0,1,0]$. The decoder generates DBIs bit by bit. DSIs are subdivided using the recursive process, similar to generating the ESI subdivision. The lower and upper limits of the DSI after decoding k bins are denoted by l'_k and h'_k . The decoded symbols are denoted by s'_k. By receiving every bit, the DBI gets smaller and the decoder selects the DSI as the one which the DBI is completely within. If the DBI resides completely within none of them (*e.g.,* the DBI at the 2nd bit $r_2 = 1$ in Figure 2-15), it jumps to the next bit without emitting any symbol and repeats this process.

-**Step1**: (initialization)

Set $[l'_0, h'_0) = [0,1)$; $[\tilde{l}'_0, \tilde{h}'_0) = [0,1)$;

$k = 0; j = 0;$

-**Step2**: (DBI evolution)

$j = j + 1$;

Find $[\widetilde{l'}_j, \widetilde{h'}_j)$ from $[\widetilde{l'}_{j-1}, \widetilde{h'}_{j-1})$ and r_j based on (2-3)

-**Step3**: (DSI selection)

For all $s'_k \in \mathcal{A}$:

$low = l_{k-1} + (h_{k-1} - l_{k-1}) \times \varphi(Indx(s'_k) - 1)$;

$high = l_{k-1} + (h_{k-1} - l_{k-1}) \times \varphi\big(Indx(s'_k)\big)$;

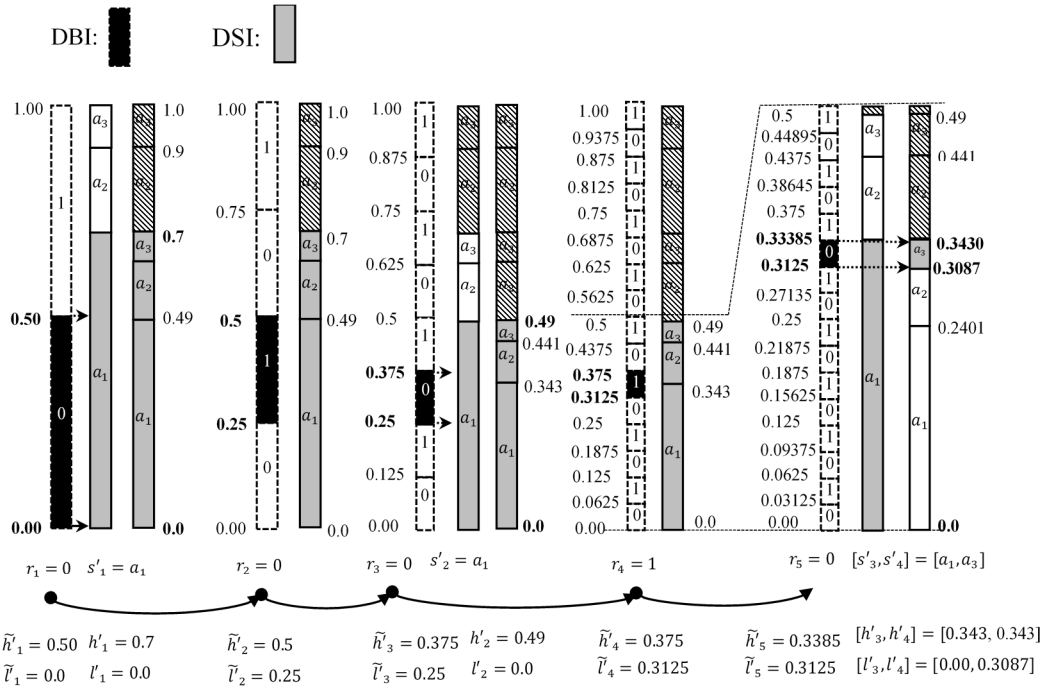
If $low \leq \widetilde{l'}_j$ and $\widetilde{h'}_j \leq high$ then

Emit s'_k and $k = k + 1$;

$l_k = low$ and $h_k = high$;

Go to Step3;

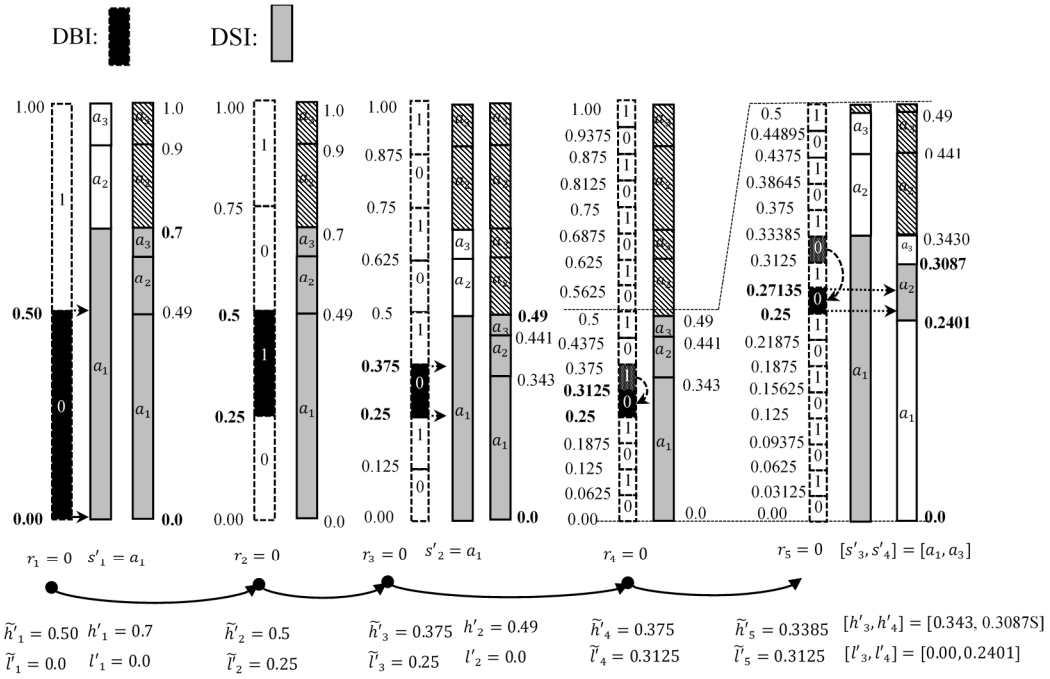
Go to Step 2;



DBI:	[0, 0.5)	[0.25, 0.5)	[0.25, 0.375)	[0.375, 0.3125)	[0.3125, 0.33385)
DSI:	[0, 0.7)	[0, 0.7)	[0, 0.49)	[0.0, 0.49)	[0.3087, 0.3430)
Symbol(s) decoded:	a_1	-	a_1	-	a_1, a_3

Figure 2-15. The relation between DBI and DSI.

In the decoder, the DSI is tracking the ESI by using DBI. The DBI could be deviated from the EBI due to channel noise. This deviation is dependent on the position of the erroneous bits. In such cases, the DSI would lose tracking on the ESI with a delay which results in erroneous decoded symbols. In Chapter 4, this delay is studied in detail. For example, suppose the 4th received bit in the above example is in error, *i.e.,* $r_4 = 0$ while $u_4 = 1$. As it is shown in Figure 2-16, the DBI is drifted from the EBI and the last decode symbol is in error.



DBI:	[0, 0.5)	[0.25, 0.5)	[0.25, 0.375)	[0.375, 0.3125)	[0.3125, 0.33385)
DSI:	[0, 0.7)	[0, 0.7)	[0, 0.49)	[0.0, 0.49)	[0.3087, 0.3430)
Symbol(s) decoded:	a_1	-	a_1	-	a_1, a_2

Figure 2-16. The drifted DBI when the fourth received bit is in error.

Unlike Huffman coding, arithmetic coding converts variable length symbol sequences to variable length codewords. Thus, it can approach to the entropy bound [7] more closely than Huffman coding [4]. The other advantage of arithmetic coding is that this algorithm operates symbol by symbol, *i.e.* no symbol packetizing is required.

However, the arithmetic coding and decoding method has a remarkable drawback. Implementing this algorithm by a fixed precision method is impossible, because by encoding more symbols the intervals become tinier, so more precision is required to represent that. A normalization technique should be used to solve this problem. To

overcome this issue, Witten *et al.* [6] modified the original arithmetic encoder by using a rescaling technique. The main idea behind their rescaling algorithm is to represent the interval by output bit 0 or 1 when the ESI is wholly in lower or upper half of the total interval, [0,0.5) and [0.5,1) respectively, then rescale the interval. In order to prevent underflow, another criterion is taken into account. When the difference between the upper and lower limits is less than a quarter of the unit interval, *i.e.* 0.25, and the interval is not entirely either in upper half or lower half, the encoder rescales the interval but emits no bit. The Encoder keeps the number of this rescaling, F, until the interval is confined to the lower or upper halves. The rescaling procedure is summarized as the following steps.

E_0: The interval is entirely inside lower half, *i.e.* $h_k < 0.5$ (Figure 2-17(a)). The encoder rescales the limits by doubling them $l_k = 2l_k$ and $h_k = 2h_k$ and emits bit $u_j = 0$. If the internal variable F is not zero, it emits a complement sequence of bits, [1,…,1], with length of F and reset F to 0.

E_1: The interval is entirely inside upper half, *i.e.* $l_k \geq 0.5$ (Figure 2-17 (b)). The encoder rescales the limits by shifting and doubling them $l_k = 2(l_k - 0.5)$ and $h_k = 2(h_k - 0.5)$ and emits bit $u_j = 1$. If the internal variable F is not zero, it emits a complement sequence of bits, {0…0}, with length of F and reset F to 0.

E_2: The interval straddles the midpoint of the unit interval and its length is less than 0.25, *i.e.* $0.5 \leq l_k < 0.5 \leq h_k < 0.75$ (Figure 2-17 (c)). In such a case, scaling is performed by $l_k = 2(l_k - 0.25)$ and $h_k = 2(h_k - 0.25)$ and the internal buffer F is increased one unit.

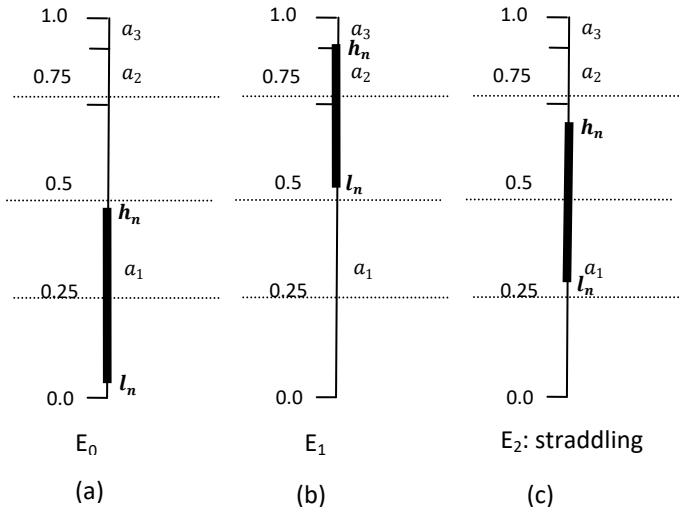


Figure 2-17. Rescaling criteria in arithmetic coding for (a) E_0 (b) E_1 (c) E_2.

The block diagram of arithmetic coding rescaling process is shown in Figure 2-18. The encoder always checks the above conditions, and emits bits and rescale the interval in E_0 or E_1. If E_3 happens, it means that the interval is about midpoint and its length is less than 0.25. In such cases, it only rescales the interval without emitting any bits, because more symbols are required to say that the final interval is on the lower half or on the upper half. It keeps track the number of E_3 rescaling, *i.e., F,* until E_0 or E_1 happens. If E_0 happens it means that the final interval is on the lower half after F times rescaling. In the other word, the final interval is about midpoint for F stages of rescaling and finally resides on the lower half. Thus, encoder emits 0 concatenated with an all 1s sequence with length of F. Similarly, if E_1 happens it emits 1 concatenated with an all 0s sequence with length of F. If none of these conditions (E_0, E_1 and E_2) happens the encoder only subdivides the current interval by encoding more symbols without any rescaling.

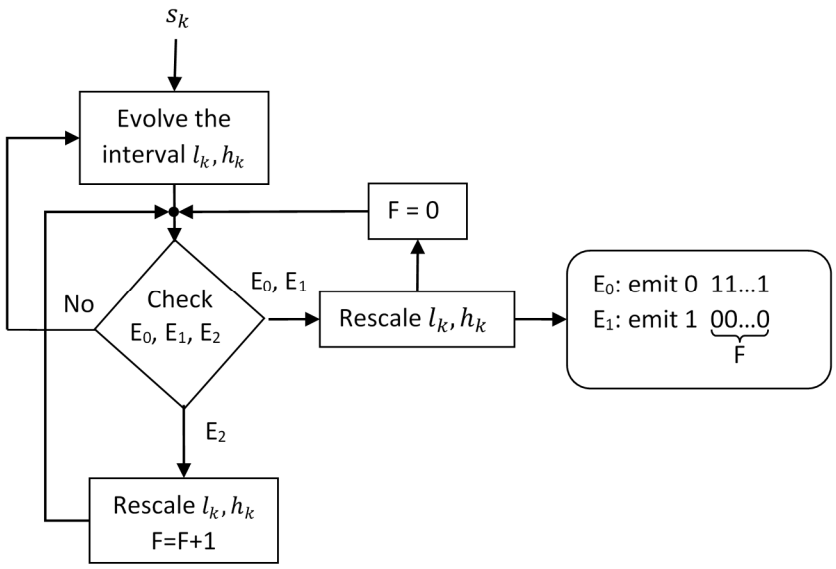


Figure 2-18. Overview of arithmetic encoding procedure.

It is noteworthy that all numbers and calculations are in integer precision. The upper and lower limits of intervals are represented by ρ-bits numbers. The encoder uses an internal binary shift register with the length of ρ to implement this algorithm. A dual shift register is utilized in the decoder side to follow the encoder shift register using the received bits. By means of the shift register scheme, the decoder is able to decode several bits simultaneously.

An approach to implement the decoder is bit clock decoding [33]. In this approach the bit sequence is progressively decoded bit by bit. This bit clock decoding is described previously in Figure 2-15. If the bit u_j, is equal to 0 or 1 the decoder selects the lower half or upper half of the current DBI interval respectively. This process continues until the input interval resides completely within one of the DSIs. To prevent the overflow and underflow problems, the decoder imitates the encoder interval rescaling for DBI and DSI. In the

following chapters, the bit clock decoding fashion is used to implement the sequential arithmetic decoder.

2.6. Binary arithmetic coding (BAC) with forbidden symbol (FS)

Arithmetic coding is highly efficient in removing the redundancy, however it has very poor resynchronization properties, *i.e.,* any alteration in the sent bits caused by the distortion in the channel leads to error propagation in the decoder side. In [32] a forbidden symbol (FS) is inserted into the source coding alphabet for error detection. The FS is never sent and so if it is detected at the decoder side, it indicates that the bit sequence is in error. As explained in the previous section, the AC divides the coding interval to subintervals and assign these subinterval to the symbols from the symbols alphabet. A portion of the ESI (and in turn DSI) is dedicated to an additional symbol that is never encoded, *i.e.* FS. In Figure 2-19, interval subdivision of binary arithmetic coding (BAC) with a FS is represented for three iterations. The subintervals assigned to the FS is named invalid subintervals and the others are named valid subintervals. The invalid subintervals are represented by shaded areas. On the decoder side if the DBI resides entirely inside an invalid subinterval, it indicates that the received bit sequence is in error. The greater the aggregate length of invalid subintervals, the better the error detection rate.

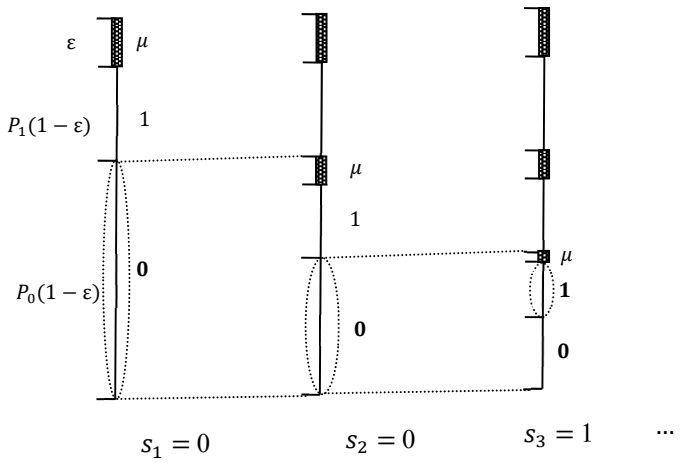


Figure 2-19. Binary arithmetic encoding with forbidden symbol μ.

In BAC with FS, the probability assigned to the FS, μ, is equal to ε. Accordingly, the length of the assigned subintervals to the symbols 0, 1 and μ are $P_0(1-\varepsilon)$, $P_1(1-\varepsilon)$ and ε respectively, where P_0 and P_1 are probabilities of binary symbols (bins) 0 and 1.

The decoder cannot detect the transmission error promptly. So, there is a delay between the bit in error and the point where it is detected [59]. In [10], this delay is modeled by a geometric distribution. The error detection delay is studied in Chapter 4.

Although the forbidden symbol enables the system to detect errors, it imposes a redundancy into the system, because the encoder dedicates a portion of the coding interval that should never be used. The amount of the coding redundancy per encoded bit is equal to $R_\mu = -\log_2(1-\varepsilon)$ bits/symbol [27]. Thus, there is a tradeoff between the error correction performance and the imposed redundancy.

FS does not guarantee detection of errors, which are close to the end of the bit sequence (namely packet), due to error detection delay. In [27], to overcome this kind of

errors, a termination mechanism using an end of packet symbol (EOPS) is used. The EOPS is encoded at the end of each packet with the assigned probability δ. At the decoder, a mismatch between the final decoded symbol and EOPS indicates errors close to the end of the packet.

BAC with FS is not embedded in video standards like H.264 and HEVC. To use this method, the standard BAC should be modified in both of the encoder and the decoder. To be compatible with the standard, the BAC decoder can only use syntactic/semantic checks to detect and correct errors [25].

In [27], FS is utilized not only for error detection, but also for error correction. A suboptimal sequential decoding is used to make a MAP arithmetic decoder. This decoding is provided in Chapter 3. An improvement using residual redundancies is also introduced in that chapter.

2.7. Summary

This chapter provided the background information related to this thesis. An introduction to block based predictive video compression including H.264 and its predecessor HEVC was presented. Some features of the H.264 used in following chapters were given. These features included two types of prediction and CABAC as entropy coding. An introduction to arithmetic coding as the core of CABAC in HEVC and H.264 was studied. As this thesis deals with coping transmission errors using joint source channel arithmetic coding, arithmetic coding with forbidden symbol was finally presented.

CHAPTER 3. Improving MAP arithmetic decoding of H.264 *intra* modes using residual redundancy

This chapter presents an improved MAP decoder to be used for joint source-channel arithmetic decoding for H.264 symbols. The proposed decoder uses not only the intentional redundancy inserted via a forbidden symbol but also exploits residual redundancy by a syntax checker. A breadth-first suboptimal sequential MAP decoder is employed. The decoder eliminates paths in the decoding tree that result in invalid syntax or that decode a forbidden symbol. In contrast to previous methods, this is done as each channel bit is decoded. Simulations using intra prediction modes show improvements in error rates, for example, syntax element error rate reduction by an order of magnitude for channel SNR of 7.33dB. The cost of this improvement is more computational complexity spent on the syntax checking.

3.1. Introduction

As explained in the previous chapter, arithmetic coding is embedded as the commonly used option for the entropy coding of a video stream in H.264. Arithmetic coding as a compression method removes redundancy efficiently. However, an undesirable result of high compression efficiency is that the compressed video stream is sensitive to transmission errors, *i.e.*, altering a few bits in the coded video stream results in large areas of corruption in the decompressed video [8]. For example, in Figure 3-1, only one-bit alteration of the compressed data stream results in corruption not only in the corresponding

current *intra* frame but also in the adjacent frames, due to the predictive mechanism of the video coder.

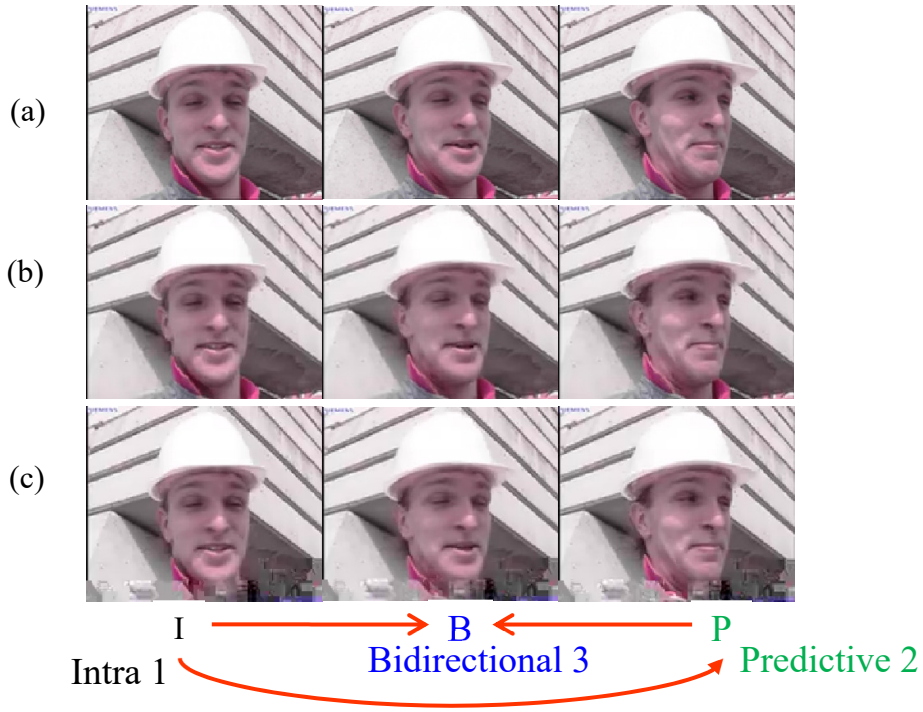


Figure 3-1. Error propagation in 3 consecutive frames by flipping only one bit of the compressed video stream Forman (a) Original video, (b) Error free decoded video, (c) Decoded erroneous data stream.

To overcome noisy environments, several approaches have been introduced. Some are embedded as tools in H.264 such as data partitioning, arbitrary slice ordering, or flexible macroblock ordering (FMO) [16]. Their main objective is to minimize the effect of error propagation without any effort to correct errors. Some of these, *e.g.*, FMO, are not widely used because of their high complexity and coding inefficiency, and are not included in the emerging standard, HEVC [5].

Other approaches include channel coding, automatic repeat request (ARQ), and error concealment. However, these approaches impose an overhead or conceal erroneous packets

instead of correcting the errors. To address these problems, joint source-channel decoding (JSCD) techniques have been introduced [23], [24], [26] and [27]. The main objective of JSCD is to exploit the available redundancy in the compressed bit stream to detect and correct erroneous bits. As explained in Section 2.6, a forbidden symbol can be inserted into the source coding alphabet for error detection and correction. A suboptimal sequential decoding technique is used for maximum *a posteriori* (MAP) estimation with the use of FS [27]. The FS should never be sent, and so if it is detected it indicates that the chosen bit sequence candidate is in error. It also imposes an overhead on the system.

Error detection can be done using not only FS, but also residual redundancy (which doesn't impose overhead). This redundancy is observed in the form of a nonuniform distribution or memory in the output bits of the compressor. One of the ways to extract the residual redundancy, is modeling the compressor output statistically and use that statistical model in the decompressor side [60], [61]. The other way to exploit the residual redundancy is correcting the bit errors results in syntax/semantic errors in the decompressed video.

In [8], commonly observed syntax errors are studied, *e.g., intra* prediction mode error, slice fragment error and illegal reference index error. It is shown that the syntax/semantic errors in *intra* modes are among the most probable syntax errors, accordingly exploiting the residual redundancy in *intra* modes is the focus of this chapter.

In [23], the error correction relies on the residual redundancy in the binarization scheme. In [24], the H.264 syntax is used to discard invalid candidates of decoded sequence by means of a syntax checker (SC). The decoder does not use SC as it constructs the

decoding tree; when the decoding tree is completely constructed, it discards the paths leading to syntax/semantic errors, and finally chooses the best candidate with the highest score as the winner. However, this can lead to the decoder not discarding erroneous sequences in as timely fashion as could be done, because SC is active only at the final stage of the decoding tree. This problem is managed by limiting the size of the packets, which imposes overhead. This approach is called *final SC* in this thesis.

In this chapter, a MAP decoder is proposed in which FS and SC are combined. In this method, FS and SC are used to eliminate invalid candidates not only at the end of the decoding tree, but also throughout the development of the tree. In contrast [24] no limit on the packet size is set. Since SC is always active at all stages of the decoding tree, the new approach is called *full SC*. This method improves the error resilience without overhead in terms of data bit rate although it does result in an additional computation load at the decoder.

3.2. MAP sequential binary arithmetic decoding

In MAP decoding, the goal is to choose the best bit sequence that maximizes the *a posteriori* probability [45] given the noisy information from the channel. The block diagram of the system is depicted in Figure 3-2. The binarizer maps the sequence of syntax elements $\boldsymbol{x} = [x_1, \dots, x_{\ell_x}]$ of length ℓ_x where x_i is a syntax element (for example an *intra* mode), to a sequence of bins $\boldsymbol{s} = [s_1, \dots, s_{\ell_s}]$ of length ℓ_s, where s_i is either 0 or 1. The binary arithmetic coder (BAC) with FS compresses the bin sequence to a bit sequence, $\boldsymbol{u} =$

$[u_1, \ldots, u_{\ell_u}]$ of length ℓ_u, where again u_i is either 0 or 1. ℓ_u is variable and dependent on the bin sequence $\boldsymbol{s}$. There is a one-to-one relation between sequences $\boldsymbol{x}$, $\boldsymbol{s}$, and $\boldsymbol{u}$**.** The bit sequence $\boldsymbol{u}$ after modulation goes to a communication channel, modeled by the addition of white Gaussian noise (AWGN), $\boldsymbol{n}$.

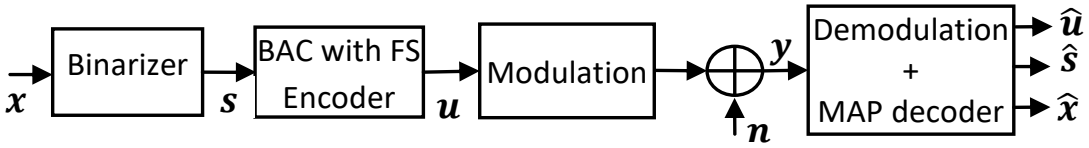


Figure 3-2. Block diagram of the coding and transmission system.

The objective of the decoder is to estimate $\boldsymbol{u}$ and the corresponding bin sequence $\boldsymbol{s}$ and syntax element sequence $\boldsymbol{x}$, represented respectively by $\hat{\boldsymbol{u}}$, $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{x}}$ in a way that maximize the *a posteriori* probability as

$$\hat{\boldsymbol{u}} = \arg\max_{\tilde{\boldsymbol{u}} \in \mathcal{B}_{\ell_u}} P(\tilde{\boldsymbol{u}}|\boldsymbol{y}), \tag{3-1}$$

where $\mathcal{B}_{\ell_u}$ is the set of all bit sequences with the length of ℓ_u. Using the Bayes' rule this probability is written as

$$\hat{\boldsymbol{u}} = \arg\max_{\tilde{\boldsymbol{u}} \in \mathcal{B}_{\ell_u}} \frac{P(\boldsymbol{y}|\tilde{\boldsymbol{u}})P(\tilde{\boldsymbol{u}})}{P(\boldsymbol{y})} = \arg\max_{\tilde{\boldsymbol{u}} \in \mathcal{B}_{\ell_u}} \frac{P(\boldsymbol{y}|\tilde{\boldsymbol{u}})P(\boldsymbol{s}_{\tilde{\boldsymbol{u}}})}{P(\boldsymbol{y})}, \tag{3-2}$$

where $\boldsymbol{s}_{\tilde{\boldsymbol{u}}}$, is the bin sequence corresponding to the bit sequence $\tilde{\boldsymbol{u}}$. The MAP decoder has access to the channel output $\boldsymbol{y}$ and to the *a priori* probabilities of the bins, *i.e* P_0, P_1. The metric in this relation consist of three terms, channel transition probability $P(\boldsymbol{y}|\tilde{\boldsymbol{u}})$ which is dependent on the characteristics of the communication channel and the modulation method, *a priori* probability $P(\boldsymbol{s}_{\tilde{\boldsymbol{u}}})$ of the source symbols and the denominator $P(\boldsymbol{y})$ which is constant for all realizations of $\tilde{\boldsymbol{u}}$ and thus insignificant in the maximization process. $P(\boldsymbol{s}_{\tilde{\boldsymbol{u}}})$ can be found by using the probability of symbols at the bin level (by assuming that the bins are independent) or even can be extended further to the next decoding level, *i.e.,* the syntax element level [62]. By finding the corresponding syntax element sequence $\boldsymbol{x}_{\tilde{\boldsymbol{u}}}$ the MAP decoding metric is written as

$$\hat{\boldsymbol{u}} = arg \max_{\tilde{\boldsymbol{u}} \in \mathcal{B}_{\ell_u}} \frac{P(\boldsymbol{y}|\tilde{\boldsymbol{u}})P(\boldsymbol{x}_{\tilde{\boldsymbol{u}}})}{P(\boldsymbol{y})}. \tag{3-3}$$

The MAP decoder can use the metric defined either in (3-2) or (3-3) depending on its access to the decoded syntax elements. In the next section, a MAP arithmetic decoder is proposed which uses a syntax checker to eliminate the invalid candidates. Here $P(\boldsymbol{x}_{\tilde{\boldsymbol{u}}})$ is equal to 0 in (3-3) for $\boldsymbol{x}_{\tilde{\boldsymbol{u}}}$ that are invalid.

Due to the large cardinality of $\mathcal{B}_{\ell_u}$, calculating metrics for the all bit sequences is impractical. In order to overcome this challenge, some suboptimal search techniques are introduced [38]. In these techniques, the generated candidates are limited to a subset $\mathcal{B}'_{\ell_u} \subseteq$

$\mathcal{B}_{\ell_u}$ that its cardinality is much lower than cardinality of $\mathcal{B}_{\ell_u}$. These techniques are explained in sections 3.2.1 to 3.2.4.

Using forbidden symbol in the decoding process helps to address this obstacle [27], because, in the MAP decoding process when the forbidden symbol μ is decoded, not only that candidate, but also the candidates with the same bit sequence prefix are eliminated promptly. The elimination of bin sequences resulting in a forbidden symbol means that we increase the likelihood that our search space includes the correct sequence.

The search space is constructed using a decoding tree. As explained in Section 2.5, the arithmetic decoder can decode the candidates bit by bit, *i.e.,* bit clock decoding, and in turn construct the decoding binary tree. Later in this chapter, it is shown that the decoding metric can be expanded to a sequential relation that can be calculated using the decoding tree as shown in Figure 3-3. The tree's columns, labeled *j* are called stages. A node at stage *j* represents a bit sequence with length *j,* $\tilde{\boldsymbol{u}}^j$, bit sequences and the internal state of the arithmetic decoder after decoding this *j*-bit sequence. Each state is evolved to the next two states depending on the value of the decoding bit at stage j, *i.e.* $\tilde{u}_j = 0\ or\ 1$. The actual candidate is found by tracing from the $m = 0$ node to the candidate node with solid/dashed branches being 0's/1's respectively. Thus, the metric of each node is calculated by decoding the first j bits of the candidates sequentially which is denoted by $m_{\tilde{\boldsymbol{u}}^j}$ which will come in Section 3.3.

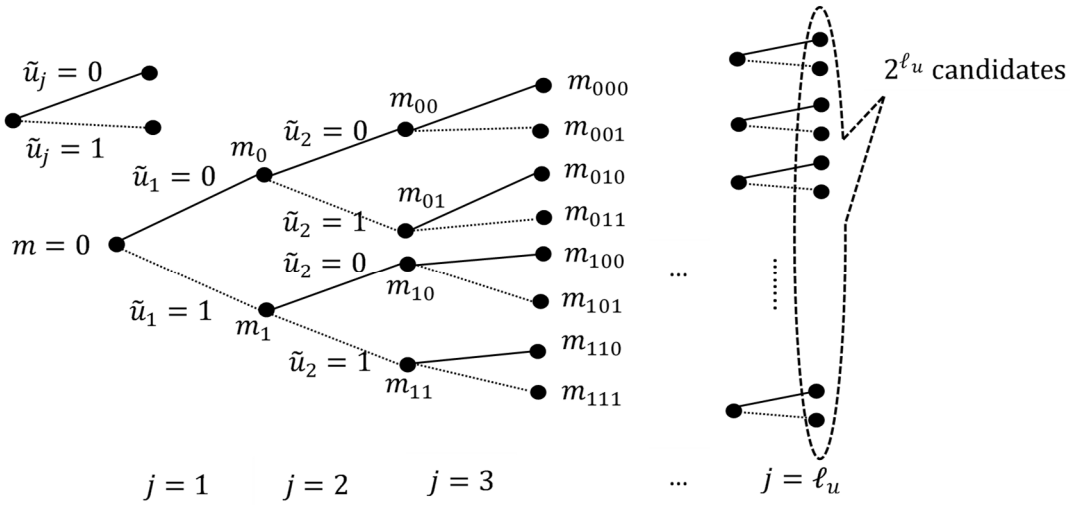


Figure 3-3. Decoding tree construction in sequential decoding technique.

At each branching process in the decoding tree, the decoder emits a variable length bin sequence whose *a priori* probability can be used in (3-2) as part of the decoding metric. This branching process continues until the decoder reaches the final channel bit. At that point, the candidate among the generated candidates with the highest metric is selected as the winner. When one reaches the end of the packet, the selected path corresponds to exactly one sequence of bits $\widehat{\boldsymbol{u}}$, and in turn, to one sequence of bins $\hat{\mathbf{s}}$, and finally, to one sequence of syntax elements $\widehat{\boldsymbol{x}}$.

As mentioned earlier, making the entire decoding tree is impractical due to its exponential growth with ℓ_u. There are three main suboptimal sequential searching techniques [38], called stack algorithm (SA), depth-first algorithm and M-algorithm (MA).

3.2.1. Metric-first algorithm (stack algorithm (SA))

In this algorithm the decoder extends the best path with highest accumulated metric, and branch forward as shown in Figure 3-3. In this scheme the decoder keeps a list of

ordered paths and extends the best one and calculates the new metric of the path and sorts all paths based on their metrics. If the extended path meets the stop criteria, like a forbidden symbol, that path would be pruned. The first path that meets the termination criteria, *i.e.,* decoding all bits without observing a forbidden state, is the winning path. Ideally a stack keeps all stored paths. However, the stack size is limited by the available memory and acceptable computational complexity of the decoder. Thus, there is a tradeoff between the complexity and performance of the decoder [38].

3.2.2. Depth-first algorithm

In this search algorithm, the decoder attempts to reach the deepest point in the decoding tree in a path by a greedy algorithm. The decoder goes along that path until it reaches the stop point, *e.g.,* observes a forbidden symbol. If the path metric hits that point, the decoder goes back to the most recent node along the path and goes forward along a partly different path from that node and tries the best one to reach an unobserved adjacent stop point.

During this forward and backward process, the decoder reaches the same nodes, states, over and over. So, the main disadvantage of this algorithm is that the decoder takes a large number of steps to regenerate the paths already visited. Since this algorithm keeps just a single path, it does not need large memory. Compared to the stack algorithm this algorithm has more complexity but it needs less memory to run [38].

3.2.3. Breadth first algorithm (M-algorithm (MA))

In this algorithm the maximum number of stored candidates and their corresponding paths is equal to M. In each step, the decoder extends forward all stored candidates in the stage of j and calculate the metric of the new candidates as depicted in Figure 3-3, and then, prunes the invalid nodes. After sorting all paths based on their metrics, only the best M paths will be kept. This process goes on until the decoder reaches the end of the stream. At the final point the best candidate with the highest metric will be selected as the winner, and its corresponding bit and bin sequences will be considered as the output of the decoder. By increasing the number of stored candidates, *i.e.,* M, the chance of missing the right pass will be decreased and the performance of the error correction code will be improved. However larger M results in a larger memory requirement and more computational complexity. Thus, there is a tradeoff between the performance and memory and complexity of the algorithm.

To calculate the maximum number of visited nodes in constructing the decoding tree, it is assumed no node results in invalid state. The decoding tree is divided to two parts. In the first part the decoding tree grows until the number of nodes reaches the limit M. It can be easily shown using the geometric series that the number of nodes in that part is $2^{(\lfloor \log_2 M \rfloor + 1)} - 1$. In the second part the M best nodes are kept and the other M nodes are eliminated. The length of this part is $\ell_u - \lfloor \log_2 M \rfloor$ and in turn the number of nodes visited in this part is $2M(\ell_u - \lfloor \log_2 M \rfloor)$. Therefore, the maximum total number of nodes visited is

$$
\begin{aligned}
&(2^{(\lfloor \log_2 M \rfloor + 1)} - 1) + 2M(\ell_u - \lfloor \log_2 M \rfloor) \\
&\cong 2M - 1 + 2M\ell_u - 2M\lfloor \log_2 M \rfloor \\
&= 2M\ell_u + 2M(1 - \lfloor \log_2 M \rfloor) - 1,
\end{aligned}
\tag{3-4}
$$

which is approximately equal to $2M\ell_u$ for large packet size, ℓ_u.

3.2.4. Comparing the searching techniques

Among the search algorithms discussed, the depth-first algorithm requires the least memory, because it keeps only one state and uses a back and forth fashion to find the best path. However, its computational complexity is high due to its back and forth behavior. As the available memory on most devices is sufficient to run the other algorithms, this technique is no longer widely used [38].

Since the stack algorithm extends just one path in each step, it is faster than the breadth-first algorithm (M-algorithm). However, M-algorithm has better error correction performance than the stack algorithm with the same size of storable states M. The other drawback of stack algorithm is the nondeterministic computational complexity, because the number of nodes visited varies in different level of the noise. The higher the noise level, the more nodes visited. But, in the M-algorithm the number of nodes visited at each depth is on average equal to 2M. The main computational load in M-algorithm is the sorting process and elimination of the invalid nodes. In the Stack algorithm, there is a sorting task as well but the sorting list is not always full and the main computational burden is the

elimination of invalid nodes [27]. Since the M-algorithm has more deterministic behavior and is more efficient, this method is utilized in this thesis.

In all of the suboptimal search algorithms, the true path might be missed or removed from the search scope. The decoder might detect such cases, if all stored candidates are invalid. This type of errors can be handled by an ARQ mechanism (if it is possible) or partially decoding the bit stream.

3.3. MAP arithmetic decoding of H.264 *intra* modes using residual redundancy

Similar to FS, decoding a syntax element leading to semantics/syntax errors is an indication of error occurrence, but without imposing an increase in the size of the compressed bitstream. This is used for error detection and correction in the proposed MAP decoding. In *intra* coding, prediction of the current MB is based on the previously compressed data from the current frame that have already been compressed. As explained in Chapter 2, each MB is partitioned into either sixteen 4×4 *intrablocks* or four 8×8 *intrablocks* or is left as one 16×16 *intrablock*. In the 4×4 and 8×8 cases, there are 9 possible directional *intra* modes (Figure 2-5), and in the 16×16 case, four directional modes are possible. The chosen prediction modes are compressed and sent in the bitstream.

Since the predictor limits its scope to the current slice, several *intra* modes may be unavailable as they require using pixels from a different slice. *e.g.*, the vertical mode cannot be used for blocks in the top edge of a slice or horizontal mode for blocks in the most left

side of a slice as shown in Figure 3-4. The invalidity of these modes is used to correct erroneously decoded bitstreams.

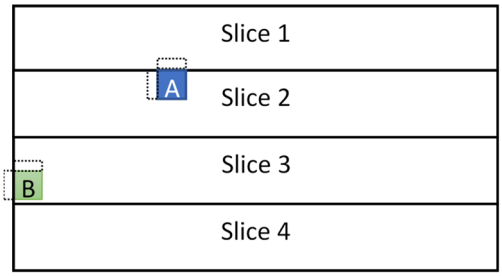


Figure 3-4. Vertical and horizontal *intra* modes are impossible in blocks A and B respectively.

3.3.1. MAP binary arithmetic decoding structure

The proposed MAP decoder uses Equations (3-2) and (3-3) for calculating the decoding metric. The MAP decoder has access to $\boldsymbol{y}$ and to the *a priori* probabilities of the bins and can check validation of the decoded syntax elements, *i.e., intra* modes. The block diagram of the MAP decoder is depicted in Figure 3-5. The demodulation and candidate generator (DCG) generates bit sequences of length j, $\widetilde{\boldsymbol{u}}^j$, and in turn bin sequences $\boldsymbol{s}_{\widetilde{\boldsymbol{u}}^j}$. The debinarizer converts $\boldsymbol{s}_{\widetilde{\boldsymbol{u}}^j}$ to syntax element sequences $\boldsymbol{x}_{\widetilde{\boldsymbol{u}}^j}$. Validation is done by discarding sequences whose $\boldsymbol{s}_{\widetilde{\boldsymbol{u}}^j}$ contains a forbidden symbol, or whose $\boldsymbol{x}_{\widetilde{\boldsymbol{u}}^j}$ contains an invalid syntax element.

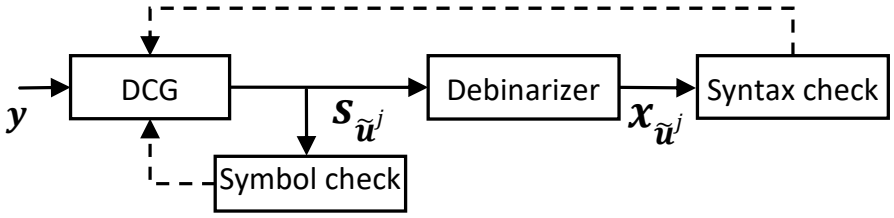


Figure 3-5. Block diagram of the MAP decoder.

3.3.2. Candidate generation and the decoding metric

The suboptimal sequential breath-first algorithm, MA, is used. As explained in Section 3.2.3, MA is a *breadth-first* technique keeps only the best M valid candidates at each stage of the decoding tree [38], as depicted in Figure 3-6 for M=2. Here at stage $j = 1$ the M=2 best paths are extended to form 2M=4 candidates at stage j. In this example one candidate is discarded as it results in a FS (shown by an unfilled a circle) and another because it resulted in an invalid syntax element (shown by a triangle). After these eliminations, valid candidates are evaluated by the metric.

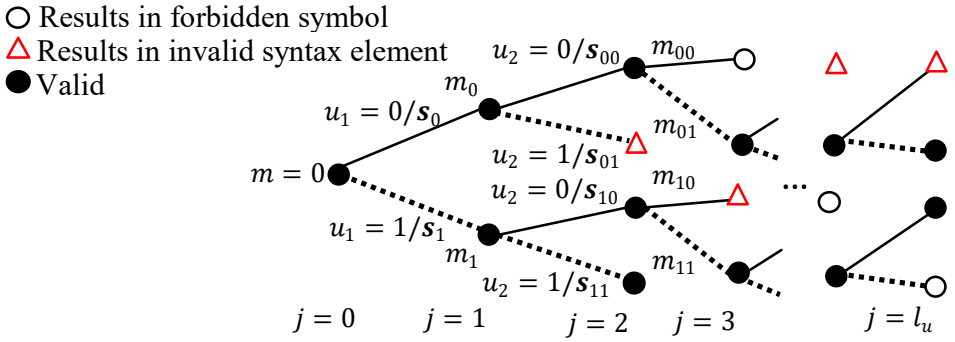


Figure 3-6. Sequential decoding tree for M-algorithm, M=2.

The metric of each candidate can be expressed in additive form, by taking the logarithm of (3-2),

$$m_{\tilde{\boldsymbol{u}}} = \log\big(P(\boldsymbol{y}|\tilde{\boldsymbol{u}})\big) + \log(\boldsymbol{s}_{\tilde{\boldsymbol{u}}}) - \log(P(\boldsymbol{y})). \tag{3-5}$$

Assuming the channel noise is white and bits of $\mathbf{y}$ are independent, the decoding metric at stage of j assigned to each candidate $\tilde{\boldsymbol{u}}^j$ is

$$m_{\tilde{\boldsymbol{u}}^j} = \sum_{k=1}^{j} \{\log\big(P(y_k|\tilde{u}_k)\big)\} + \log(P(\boldsymbol{s}_{\tilde{\boldsymbol{u}}^j})) - \sum_{k=1}^{j} \{\log(P(y_k))\}, \tag{3-6}$$

where y_k and $\tilde{u}_k$ are the k^{th} bit of $\mathbf{y}$ and $\tilde{\mathbf{u}}$, respectively and $\boldsymbol{s}_{\tilde{\boldsymbol{u}}^j}$ is the bin sequences corresponding to $\tilde{\boldsymbol{u}}^j$. As mentioned earlier, $\tilde{\boldsymbol{u}}^j$ is a bit sequence candidate of length j. The last term in Equation (3-6), $\sum_{k=1}^{j}\{\log(P(y_k))\}$, is identical for all candidates at the same stage j. Since in the M-algorithm the candidate metrics are compared and sorted in the same stage j, this term is insignificant and can be removed. However, in stack algorithm this term cannot be discarded, because the decoder stores and sorts the states in different stages of the decoding tree. Consideration of this term in the metric calculation is one of the drawbacks of the stack algorithm. As M-algorithm is used in this work, the metric can be written as

$$m_{\tilde{\boldsymbol{u}}^j} = \sum_{k=1}^{j} \{\log\big(P(y_k|\tilde{u}_k)\big)\} + \log(P(\boldsymbol{s}_{\tilde{\boldsymbol{u}}^j})). \tag{3-7}$$

Using the simplifying assumption that bins are independent the second term is decomposed as

$$\log(P(\boldsymbol{s}_{\tilde{\boldsymbol{u}}^j})) = \sum_{k=1}^{j} \{\log(P(\boldsymbol{s}_{\tilde{\boldsymbol{u}},k}))\}, \tag{3-8}$$

where $\boldsymbol{s}_{\tilde{\boldsymbol{u}},k}$ is the variable length bin sequence generated by decoding the k^{th} channel bit $\tilde{u}_k$. $\boldsymbol{s}_{\tilde{\boldsymbol{u}},k}$ is dependent on $\tilde{u}_k$ as well as on the internal state of the MAP decoder after decoding the first $k-1$ bits. $P(\boldsymbol{s}_{\tilde{\boldsymbol{u}},k})$ is calculated by using the *a priori* probability of bins. When the length of $\boldsymbol{s}_{\tilde{\boldsymbol{u}},k}$ is zero it shows that the internal state of BAC is changed by decoding $\tilde{u}_k$ without emitting any bins. In such cases, this term is neglected.

The metric is written in a differential form

$$m_{\tilde{\boldsymbol{u}}^j} = m_{\tilde{\boldsymbol{u}}^{j-1}} + \Delta m_{\tilde{\boldsymbol{u}},j} = \sum_{k=1}^{j} \Delta m_{\tilde{\boldsymbol{u}},k}, \tag{3-9}$$

where,

$$\Delta m_{\tilde{\boldsymbol{u}},k} = \log\big(\mathrm{P}(y_k|\tilde{u}_k)\big) + \log\Big(\mathrm{P}\big(\boldsymbol{s}_{\tilde{\boldsymbol{u}},k}\big)\Big). \tag{3-10}$$

Thus, the metric at each node is the metric at the parent node, $m_{\tilde{\boldsymbol{u}}^{j-1}}$, plus the incremental term, $\Delta m_{\tilde{\boldsymbol{u}},j}$. At each stage, the M valid nodes with the highest $m_{\tilde{\boldsymbol{u}}^j}$ are saved.

This process is continued until the decoder reaches the final channel bit. At that point, the path among the surviving valid paths that respects the end of packet symbol (EOPS) constraint with the highest metric is selected as the winner. When one reaches the end of the packet, the selected path corresponds to exactly one sequence of bits $\hat{\boldsymbol{u}}$, and in turn, to one sequence of bins $\hat{\boldsymbol{s}}$, and finally, to one sequence of syntax elements $\hat{\boldsymbol{x}}$.

The proposed approach takes advantage of FS and residual redundancy in syntax elements simultaneously in constructing the decoding tree. The *FS+final SC* approach [24], uses FS to eliminate candidates at each decoding stage, but only uses SC in the last stage of the decoding tree. It does not discard invalid paths as quickly as the proposed method does. This may result in the correct path being crowded out from the M best paths at some stage resulting in a greater likliehood of the final output being incorrect. This is especially likely for small values of M and long packets.

In the proposed method, *FS+full SC*, since SC is always active, the maximum number of times that SC runs is about $2M\ell_u$ (see Equation (3-4)), while this number is 2M for *FS+final SC* method. Thus, the proposed method is more computationally complex than *FS+final SC* method. The *FS+Full SC* method can have comparable performance to the *FS+final SC* method, but with a smaller M.

3.3.3. Channel modeling

In Equation (3-10), the term $\log\big(\mathrm{P}(y_k|\tilde{u}_k)\big)$ is dependent on the modulation and the channel characteristics. In channel modeling, the bit sequences are modulated using binary phase shift keying (BPSK) and transmitted over an AWGN channel. The SNR of the

received signal is E_b/N_0, which E_b is energy of bit. To decode the noisy received data from the channel two main approaches are possible, namely hard decoding and soft decoding.

3.3.3.1. Hard decoding

The received noisy bits y_k are demodulated to hard bits r_k by

$$r_k = \begin{cases} 1 & : \quad y_k \geq 0 \\ 0 & : \quad y_k < 0 \end{cases}. \tag{3-11}$$

Since a BPSK modulation is used, channel transition probability in (3-10) is as follow,

$$P(y_k|\tilde{u}_k) = \begin{cases} p & : \tilde{u}_k \neq r_k \\ 1-p & : \tilde{u}_k = r_k' \end{cases} \tag{3-12}$$

where p is probability of error and can be written as a function of SNR,

$$p = \mathrm{Q}\left(\sqrt{2 \times SNR}\right). \tag{3-13}$$

Q is the well-known tail probability of the standard normal distribution,

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} \exp\left\{\frac{-z^2}{2}\right\} \mathrm{dz}. \tag{3-14}$$

3.3.3.2. Soft decoding

In the soft decoding scheme, the MAP decoder has access to noisy soft bits as

$$y_k = \begin{cases} \sqrt{E_b} + n_k & : u_k = 0 \\ -\sqrt{E_b} + n_k & : u_k = 1 \end{cases}, \tag{3-15}$$

where n_k is zero mean additive white Gaussian noise with variance of $\sigma^2 = N_0/2$. By using the function $\bar{u}_k = \sqrt{E_b}(2\tilde{u}_k - 1)$ the channel transition probability turns out to be

$$\log\big(P(y_k|\tilde{u}_k)\big) = -\frac{\log(2\pi\sigma^2)}{2} - \frac{(y_k - \bar{u}_k)^2}{2\sigma^2}. \tag{3-16}$$

By discarding the constant term, *i.e.,* $-\frac{\log(2\pi\sigma^2)}{2}$, the soft decoding metric for MA turn out to be as

$$m_{\tilde{\boldsymbol{u}}^j} = \sum_{k=1}^{j} \left\{ -\frac{(y_k - \bar{u}_k)^2}{2\sigma^2} + \log\left(P\left(\boldsymbol{s}_{\tilde{\boldsymbol{u}},k}\right)\right) \right\}. \tag{3-17}$$

3.4. Simulations and results

In the simulation, 300 frames of the video *Foreman* in QCIF format are used. All frames are divided into slices and *intra* encoded with the quantization parameter (QP) value 28. As changing QP does not affect the decision on selection of intra modes the simulations are done for this QP. Each slice consists of one row of MBs using the reference software H.264, *i.e.,* JM [63]. *Intra* modes, after the signaling process in the H.264 standard and

binarization, are encoded by BAC with a FS. The tests are done for various FS probabilities $\varepsilon=5\times10^{-2}$, and $\varepsilon=2\times10^{-1}$ and EOPS probability $\delta = 0.01$. The used channel SNR's 7.335 dB, 6.7895 dB, 5.208 dB, 4.3232 dB and 1.312 dB correspond to the probability of channel bit error $p = 5\times10^{-4}$, 10^{-3}, 5×10^{-3}, 10^{-2} and 5×10^{-2}, respectively. Parameter M that controls the performance and complexity tradeoff, is set to 16, 64 and 256. The simulations are repeated, until at least 60 erroneous packets are decoded. Unlike [24], there is no constraint on packet size. The hard decoding scheme is used in all methods.

In Table 3.1 the performance of *FS* only [27], and *FS+final SC* [24], and the proposed method *FS+full SC* are compared in terms of error rates and average computational time (T) for channel SNR=6.7895 dB. Computation times are on a computer with a core i7, 2.8GHz CPU and are the average for one packet. Symbol error rate (SER) compares the decoded bins of the selected path to the uncorrupted bins ($\boldsymbol{s}$ and $\hat{\boldsymbol{s}}$ in Figure 3-2), while syntax element error rate (SEER) compares the syntax elements of the selected path to the uncorrupted sequence of syntax elements ($\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ in Figure 3-2). In calculation of packet error rate (PER) if there is at least one erroneous bin in the decoded packet, the entire packet is considered to be in error. The proposed method is seen to be superior to the other methods in term of these error rates.

Table 3.1. Error rates and average decoding time (T) for the algorithms *FS* Only, *FS+ Final SC*, and for the proposed method *FS+Full SC* at various values of probability of FS, various values of M, and at channel SNR = 6.7895 dB

	FS probability	Method	PER	SER	SEER	T(sec)
M =256	ε = 0.05	*FS*	6.33E-02	7.60E-03	1.00E-02	50.9
		FS+Final	3.67E-02	7.10E-03	9.46E-03	50.8
		FS+Full SC	7.38E-03	5.46E-04	7.03E-04	91.2
	ε = 0.2	*FS*	1.09E-03	1.89E-06	2.71E-05	70.5
		FS+Final	9.17E-04	1.58E-06	2.11E-06	70.5
		FS+Full SC	8.93E-04	1.57E-06	2.24E-06	114.6
M = 64	ε = 0.05	*FS*	1.21E-01	1.36E-01	3.83E-02	12.7
		FS+Final	1.16E-01	1.30E-01	3.82E-02	12.7
		FS+Full SC	3.20E-02	4.77E-02	8.74E-03	22.8
	ε = 0.2	*FS*	2.17E-03	2.00E-04	3.15E-04	17.5
		FS+Final	1.88E-03	1.99E-04	2.66E-04	17.5
		FS+Full SC	1.39E-03	5.44E-05	6.69E-05	28.4
M = 16	ε = 0.05	*FS*	6.67E-01	1.96E-01	2.63E-01	3.8
		FS+Final	6.67E-01	1.96E-01	2.63E-01	3.8
		FS+Full SC	2.73E-01	7.23E-02	9.18E-02	6.5
	ε = 0.2	*FS*	3.17E-02	7.62E-03	1.02E-02	4.9
		FS+Final	3.17E-02	7.62E-03	1.02E-02	4.9
		FS+Full SC	2.10E-02	5.10E-03	6.59E-03	7.8

The error rate improvement is more apparent when the probability of FS is lower, since for lower ε the pruning of the decoding tree is done increasingly by SC. The gain of *FS+final SC* over *FS* method is tiny for small M, because for small M, the likelihood of missing the correct path is high especially for long packets. *FS+Full SC* outperforms in small M as well as large M. As expected the computational complexity of the proposed method is the largest due to the SC process in all stages of the decoding tree. T is almost double for *FS+Full SC* as compared to *FS+Final SC* for the same channel conditions.

The tradeoff between complexity, performance and FS probability ε is seen in this table. For example, in M=16, ε=0.2 and *FS* only method, PER is as same as the PER in the

proposed method for M=64 and ε=0.05. Thus, this method has lower overhead due to lower ε, at the cost of higher complexity imposed by the SC and larger value of M.

In order to evaluate the performance of the system under different noise levels, SEER is plotted as a function of the channel SNR and it is shown in Figure 3-7 for the various methods, with probability of FS ε=0.1, M=16 and 256. For the highest value of SNR=7.335 dB, *FS+Full SC* improves SEER by an order of magnitude. As seen in this figure, the proposed method exhibits a gain of about 1dB over the *FS+final SC* and *FS* only.

Since the *FS+final SC* method checks syntax just at the final nodes of the decoding tree, it may take longer to eliminate incorrect paths. Therefore, this method does not provide a significant gain especially for lower SNR values where a large number of errors in a long packet are more probable.

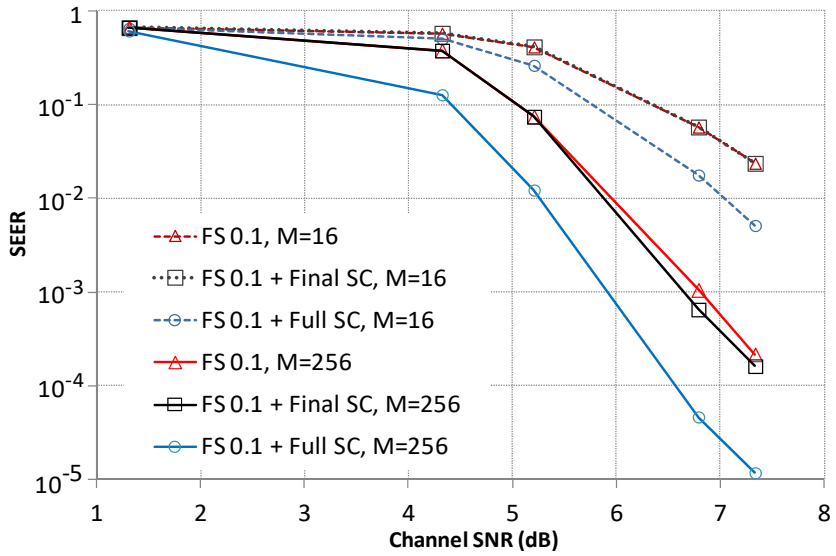


Figure 3-7. SEER versus SNR for three different MAP decoders and ε=0.1.

The structure of the decoding tree for various ε's is illustrated in Figure 3-8, Figure 3-9, Figure 3-10 and Figure 3-11. In these figures, the surviving paths are drawn after 40

levels, and M and channel SNR are respectively chosen to be 16 and 4.3232 dB. The solid and dashed branches correspond to channel bits 0 and 1, respectively. The winning path with the highest score is represented by thick cyan color. The distance between the first node that a non-winning path branches from the winning path to the end of decoding tree is called decoding tree tail length (DTTL). The minimum value of DTTL is $\log_2 M$ and its maximum is equal to the length of the decoding tree. Value of DTTL varies between 18 and 32 in these figures. By comparing these figures, it can be seen that decreasing value of ε results in growth in DTTL. This is because of the relation between the delay in detecting FS after an error and probability of FS. This delay is studied in Chapter 4.

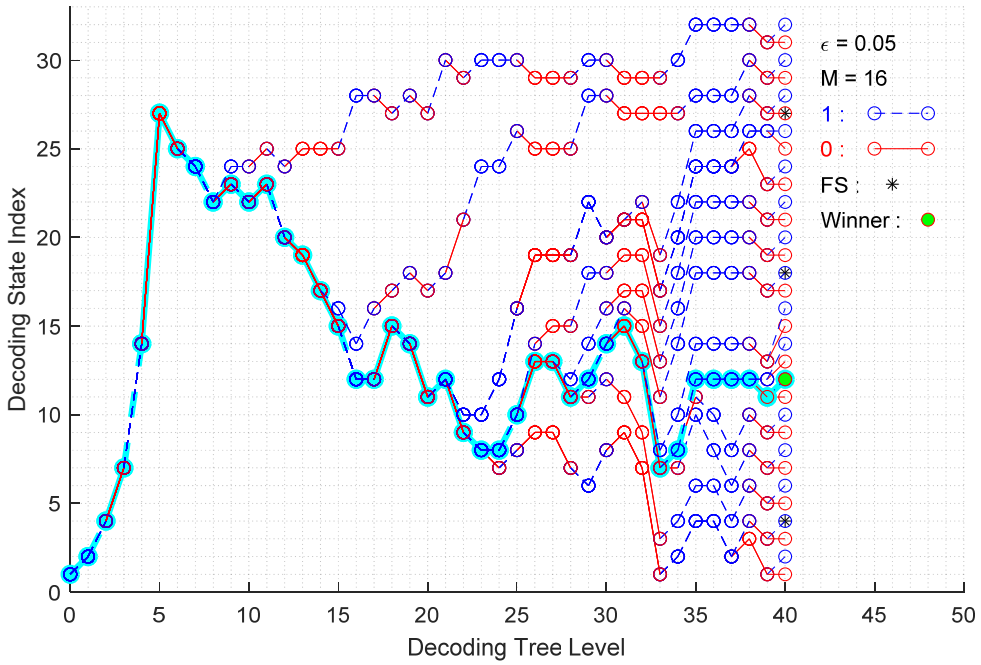


Figure 3-8. Decoding tree with length 40 for SNR=4.3232 dB, M=16 and ε=0.05.

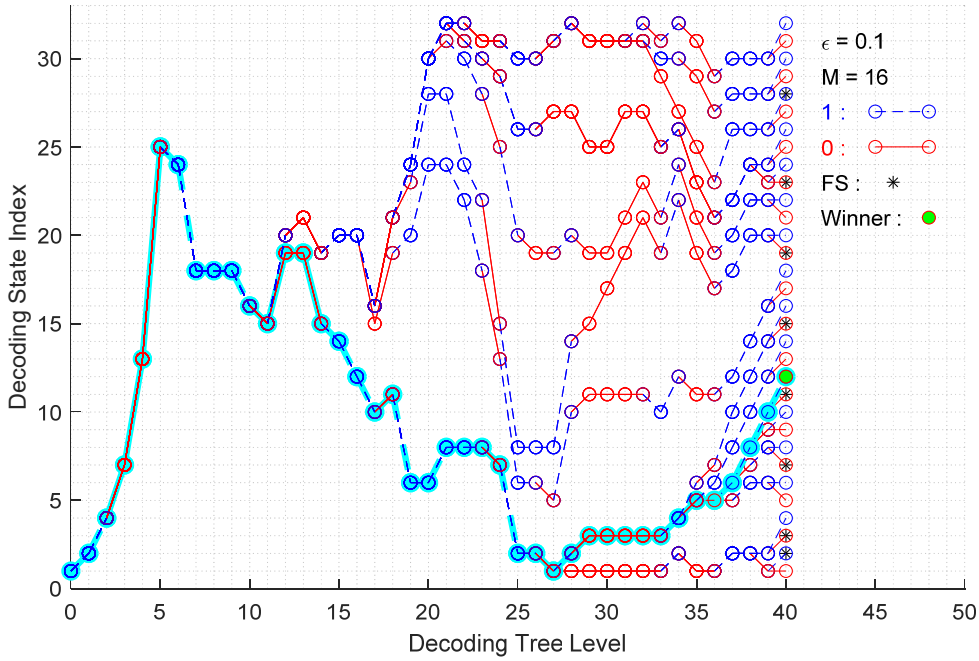


Figure 3-9. Decoding tree with length 40 for SNR=4.3232 dB, M=16 and ε=0.1.

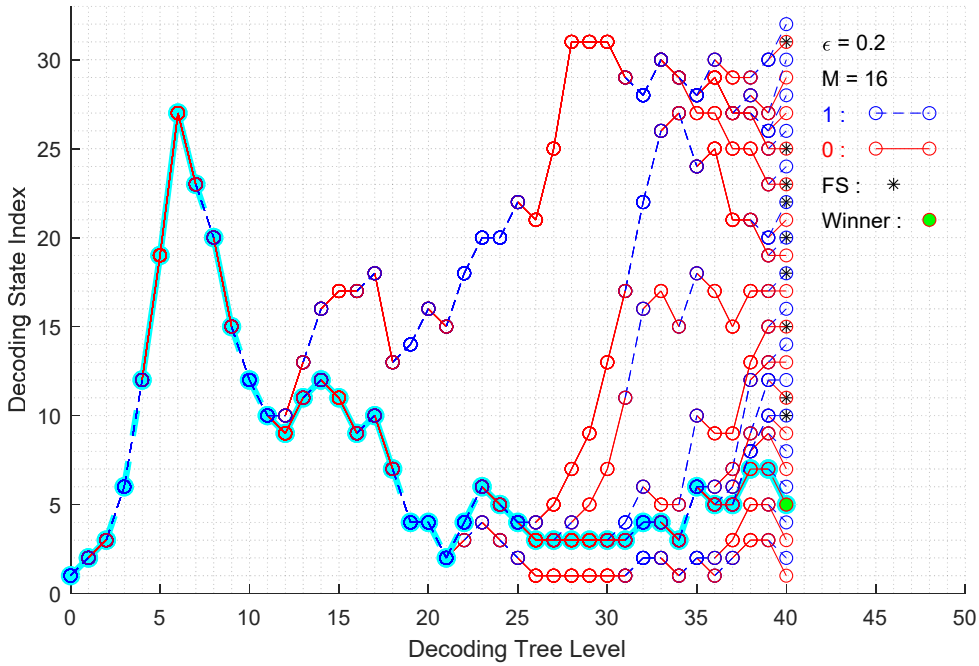


Figure 3-10. Decoding tree with length 40 for SNR=4.3232 dB, M=16 and ε=0.2.

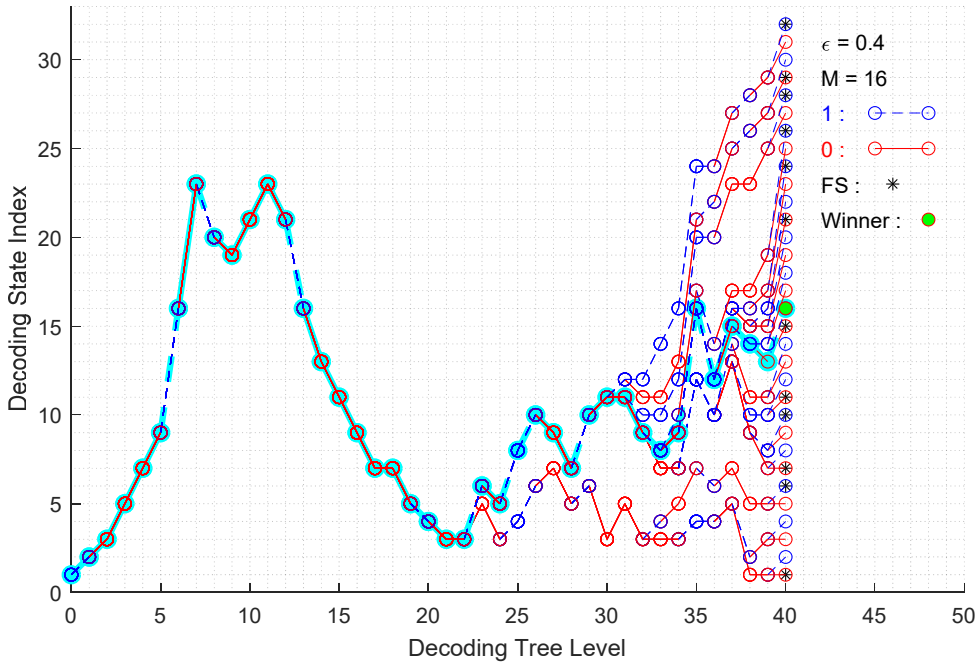


Figure 3-11. Decoding tree with length 40 for SNR=4.3232 dB, M=16 and ε=0.4.

3.5. Summary

In this chapter, a new method has been proposed for MAP arithmetic decoding by exploiting the residual redundancy in H.264 video streams. In the proposed method an arithmetic coding with forbidden symbol, and a breadth first sequential decoding algorithm have been used. Unlike the previous work [24], syntax checking is performed at all stages of the decoding tree. Although the syntax checker imposes a computational overhead to the decoder, the simulation results have shown considerable improvement in terms of PER, SER, and SEER. Accordingly, a tradeoff between computational complexity and performance has been shown. Depending on the available computational power and desired performance, the decoder can select whether to use *FS+Final SC* algorithm or *FS+Final SC* algorithm. The structure of decoding tree for MA has been represented for various

probabilities of FS. The inverse relation between DTTL and probability of FS has been shown. As future work, this measure can be calculated through construction of the decoding tree and if it exceeds a threshold some isolated branches can be eliminated.

CHAPTER 4. Optimizing the forbidden symbol configuration in joint source-channel arithmetic codes

As explained in previous chapters, in joint source-channel arithmetic coding, a forbidden symbol is added to the symbol set to make it more robust against transmission errors. By splitting the interval occupied by the FS into subintervals, various configurations are possible. In this chapter, the delay probability function (DPF), defined as the probability that the delay (the number of bits required to detect an error) is greater than its argument, is calculated for various FS configurations. A figure of merit is also proposed for optimizing the FS configuration. It is defined as the probability of missed error detection (PMD). Simulations are carried out by employing the breadth-first suboptimal sequential MAP decoding. The simulation results show the effectiveness of the proposed figure of merit, and support the FS configuration in which the FS interval lies entirely between the other information carrying symbols is the best.

4.1. Introduction

Joint source-channel arithmetic coding (JSCAC) techniques have been introduced in [23], [27], [32] and [64] that combine data compression capabilities and error correction. As explained in the previous chapter, in JSCAC, redundancy is intentionally added during source coding using a forbidden symbol [23], [27]. The decoder can use either this added redundancy or residual redundancy to enhance error resilience [64].

As explained in Chapter 2, AC subdivides iteratively internal an interval (ESI and DSI) and assigns the subintervals to the source coding symbol alphabet. In [32], a forbidden symbol (FS) is inserted into the source coding alphabet for error detection. This is done by assigning a portion of the ESI and DSI to the FS, which is called the invalid interval (FS subinterval). The other subintervals are assigned to information carrying symbols which are called valid intervals. The FS is never transmitted, and its detection indicates that the bit sequence is in error. In [10], the error detection capability of this technique has been analyzed, and an automatic repeat request (ARQ) protocol has been utilized for error correction. In [27], the decoder uses the FS to eliminate the decoded sequence candidates that result in a FS through the construction of a decoding tree.

In conventional methods, the FS subinterval is located in the current interval without splitting it. X. Wang *et al.* [40] have introduced an FS configuration by splitting the FS subinterval into several disjoint FS subintervals. In their method the FS subintervals are located on both sides of the valid subintervals. They have measured the effectiveness of their FS configuration by evaluating the ratio of the parts of the FS subinterval length that are in the lower and upper halves of the unit interval. In their method, the FS subinterval distribution is balanced about the midpoint of the AC internal interval. However, this FS configuration is not guaranteed to be optimal in terms of error rate and error detection delay.

A look-ahead technique for faster error detection is proposed in [41], and the error detection delay of several FS configurations are also studied. As explained in Chapter 2 there are two types of intervals at the decoder side, namely DSI and DBI. DSI is subdivided

iteratively and DBI is determined by the bit sequence received from the channel. In a conventional method, for decoding, the DBI must be completely within one of the subintervals of the DSI before the bit is decoded. But in this look-ahead method if the DBI partially covers FS subintervals and a valid subinterval, by considering that the bit sequence is not in error, and thus the symbol or symbols corresponding to the valid subinterval can be decoded, and that valid subinterval considered as a new DSI and subdivided ahead. With this look-ahead method, an error may be detected earlier, because by selecting and subdividing the DSI ahead more space would be occupied by invalid subintervals by decoding the same number of bits. However, in [41], using simulations it is shown this technique has no gain in the cases that the FS subinterval is located between the valid subintervals.

In [65], a forecasting forbidden symbols technique is introduced in which the effective invalid subinterval is expanded. By expanding the invalid subinterval ahead, the decoder can detect the FS earlier and in turn better error correction performance is achievable. However, in this method the FS configuration is not taken into account.

The AC internal state is determined by three numbers, namely upper and lower limits of the ESI (in turn DSI at the decoder side) and the number of rescaling processes without bit emission (F as explained in Section 2.5). There are limited number of options for the upper and lower limits of ESI due to the limited number of bits assigned to them, but F can be any positive integer. Thus, the number of states of AC is unlimited. In [42], the conventional AC with FS is modified by limiting F to F_{max} and it is modeled by a finite-state machine (FSM). However, limiting F_{max} adds redundancy into the compressed data

and thus lowers compression. The smaller F_{max} is, the greater the added redundancy. An upper bound for bit error rate is calculated for this FSM model. This upper bound is used for optimizing the FS configuration for small F_{max}s. However, this optimization is only valid for this FSM model with limited F and cannot be applied in practical AC. For example, for AC with $F_{max} = 2$, distributing the invalid subinterval in all three possible positions is the best configuration based on the FSM model, but in practical AC placing the invalid subinterval between valid ones is the best configuration.

In this chapter, a generalized FS configuration is analyzed using a new figure of merit that measures the probability of missed error detection (PMD). The best FS configuration based on PMD is the one in which the FS subinterval is placed entirely between the valid subintervals. Simulations are carried out using a breadth-first MAP sequential decoding. In these simulations error rates are measured for various FS configurations and different level of noise. The optimality of FS configuration based on simulation results is consistent with what is obtained from the figure of merit PMD. As changing the FS configuration does not change the amount of redundancy added, it does not impose an expansion of the data bit rate.

The organization of this chapter is as follows. In Section 4.2, a generalized configuration of BAC with FS is provided. The new measure for optimization of the FS configuration is introduced in this section. In Section 4.3 simulation results are presented followed by Section 4.4 that summarizes this chapter.

4.2. Binary arithmetic coding with generalized forbidden symbol configuration

The main objective of AC is to map the input sequence of symbols into a variable-length sequence of bits. This is an iterative task which generates disjoint subintervals, their length (ideally) being proportional to the *a priori* probability of the corresponding input symbol. This iterative procedure is explained in Chapter 2 in detail. Practical AC uses intervals of integers in $[0,2^{\rho})$ as ESI, EBI, DSI, and DEI, where ρ is the length of the internal AC register, and all the subinterval boundaries are integers [42], [6]. In JSCAC a portion of the ESI and DSI is dedicated to an additional symbol that is never encoded, *i.e.*, a FS [27] with an assigned probability ϵ. Observing the FS indicates that an error has occurred. The greater the probability ϵ, the better the error detection and correction rate.

As explained in Chapter 2 there is an iterative subdivision process for ESI and DSI. In this process, the ESI after encoding k bins is denoted by $I_k = [l_k,\ h_k)$ whose length is $w_k = h_k - l_k$, and in turn the DSI after decoding k bins is denoted by $I'_k = [l'_k,\ h'_k)$ whose length is $w'_k = h'_k - l'_k$. There are several possibilities for placing the FS subinterval in the ESI and DSI. The FS subintervals can be placed at the beginning, at the end or between the valid subintervals.

Here, the FS subinterval is divided into three subintervals, one to the left of the 0 valid subinterval (FS_1), one to the right of the valid 1 subinterval (FS_2) and one in between them (FS_3) as depicted in Figure 4-1 for ESI. Their lengths are proportional to ϵq_1, ϵq_2 and ϵq_3, respectively, where $q_1 + q_2 + q_3 = 1$ and $0 \leq q_1, q_2, q_3 \leq 1$. The length of valid subintervals 0 and 1 are proportional to $(1-\epsilon)p_0$ and $(1-\epsilon)p_1$ respectively where p_0 is

probability of bin 0 and p_1 is probability of bin 1 which is $p_1 = 1 - p_0$. Thus, for given p_0 and ϵ only two parameters q_1, q_2 need to be adjusted to find the best FS configuration. The DSI subdivision should be adjusted in the same way. The question is which FS configuration is the best one.

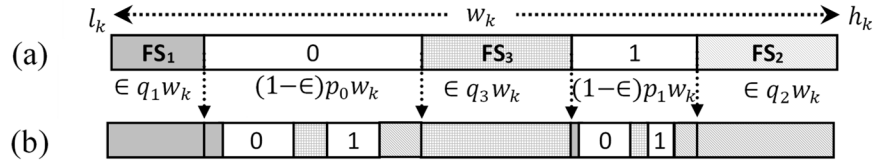


Figure 4-1. (a) Distributing and placing of the FS subintervals FS_1, FS_2 and FS_3 in the current ESI of length w_i, (b) The second step of iterative interval generation in binary arithmetic coding with distributed forbidden symbol.

As explained in Chapter 2, DBI is uniquely determined by the input bit sequence to the decoder. Any errors in the received bits causes the DBI to deviate from the EBI. If the DBI falls into an invalid subinterval in DSI, the decoder detects the error.

4.2.1. Error detection delay

A measure for evaluating the effectiveness of the coding is the error detection delay. Error detection delay, denoted by the random variable D, is the number of bits needed to detect an error after it occurs.

In [10], the error detection at any particular bit is assumed to be a Bernoulli trial [39] with parameter of ϵ, *i.e.* by receiving each bit after an error the DBI falls into an invalid

subinterval of DSI with probability of ϵ and it falls into a valid subinterval of DSI with probability of $1-\epsilon$. Thus, the probability mass function of D is that of a geometric random variable *i.e.,*

$$f_D(d) = \epsilon(1-\epsilon)^{d-1}, \tag{4-1}$$

where $d = 1,2,\dots,\infty$. The delay probability function (DPF) is defined as the probability that the delay, D, is greater than n, *i.e.,* $P[D > n]$. This yields

$$P[D > n] = (1-\epsilon)^n. \tag{4-2}$$

In this model, the FS configuration is not taken into account for calculating DPF.

Here, a new measure for DPF is proposed in which the FS configuration is considered. Suppose that the DSI subintervals are generated iteratively (similar to what was shown in Figure 4-1 for two consecutive input symbols). Boundaries of each subinterval are integers as explained formerly. The subdivision process continues as long as the length of all the subintervals are greater than a predefined threshold denoted by λ. Suppose ρ consecutive encoder output bits are stored in a binary shift register whose value is denoted by ν. This value corresponds to the EBI $\tilde{I}(\nu) = [\nu, \nu+1)$. A single-bit inversion (imposed by the channel noise) at position b, $0 \le b < \rho$, drifts ν to $\nu' = \nu + (-1)^{\nu_b} 2^b$, where $\nu_b \in \{0,1\}$ is the b^{th} bit of ν, and in turn, the EBI, $\tilde{I}(\nu)$, is drifted to the DBI $\tilde{I}(\nu') = [\nu', \nu'+1)$. This drift may cause the DBI to fall into a valid subinterval, or into an invalid subinterval of

DSI as shown in Figure 4-2 for $\rho = 6$. As lengths of $\tilde{I}(\nu')$ is equal to 1 and its starting and finishing points are integers, straddle case is impossible. In the other word the EBI may not fall into two adjacent subintervals. In Figure 4-2(a), the erroneous bit is at $b = 4$ which alters $\nu = (000010)_2 = 2$ to $\nu' = (010010)_2 = 18$ and it causes the DBI to fall into a valid subinterval. In Figure 4-2(b) the position of erroneous bit is $b = 5$ which alters $\nu = (001001)_2 = 9$ to $\nu' = (101001)_2 = 41$ and causing that DBI to fall into an invalid subinterval.

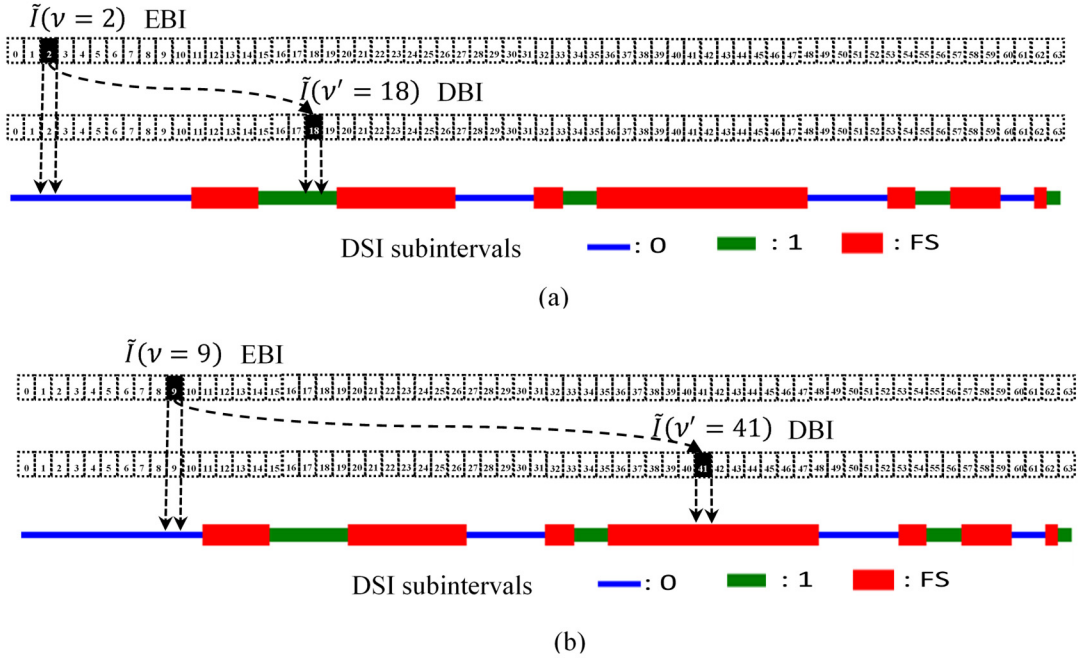


Figure 4-2. Two types of interval drifts, (a) from valid interval to valid one and (b) from valid to invalid interval.

To distinguish these two types of drifts, two indicator functions are defined as

$$\alpha(v) = \begin{cases} 1 & : \ \tilde{I}(v)\ residing\ in\ a\ valid\ subinterval \\ 0 & : \ otherwise \end{cases}, \tag{4-3}$$

and

$$\beta(v, b) = \begin{cases} 1 & : \ \tilde{I}(v')\ residing\ in\ a\ valid\ subinterval \\ 0 & : \ otherwise \end{cases}. \tag{4-4}$$

If $\tilde{I}(v')$ resides inside a valid subinterval, the decoder cannot detect the error at position b using these ρ bits.

Assuming that the length assignment of subintervals of DSI respects the bin probabilities, all of the intervals $\tilde{I}(v)$ in valid regions are equiprobable. Here, DPF, represented by θ is calculated using the indicator functions $\alpha(v)$ and $\beta(v, b)$ as

$$\theta(q_1, q_2, n) = \Pr(D > B | B = n) = \frac{\sum_v \big(\alpha(v).\beta(v, B = n)\big)}{\sum_v \alpha(v)}. \tag{4-5}$$

This is the ratio of the number of drifts of DBI from the valid subintervals to the valid ones and the total number of possible drifts of DBI from the valid subintervals by flipping one bit at position n.

DPF, $\theta(q_1, q_2, n)$, is calculated for the parameters $p_0 = 0.8$, $\epsilon = 0.1$, $\lambda = 200$ and $\rho = 20$, and it is depicted in Figure 4-3. Here in DSI and ESI subdivision, four FS configurations are studied, which are placing the FS subinterval beside the larger valid

subinterval (referred to as *FS at beginning*), beside the smaller valid subinterval (referred to as *FS at end*), between the valid subintervals (referred to as *FS in middle*), and at both sides of the valid subintervals [40] so that $q_1 = p_0/2$ and $q_2 = (1 - p_0)/2$ (referred to as *distributed FS in 3 places* [40]). DPF based on the geometric model in Equation (4-2) is also depicted in this figure.

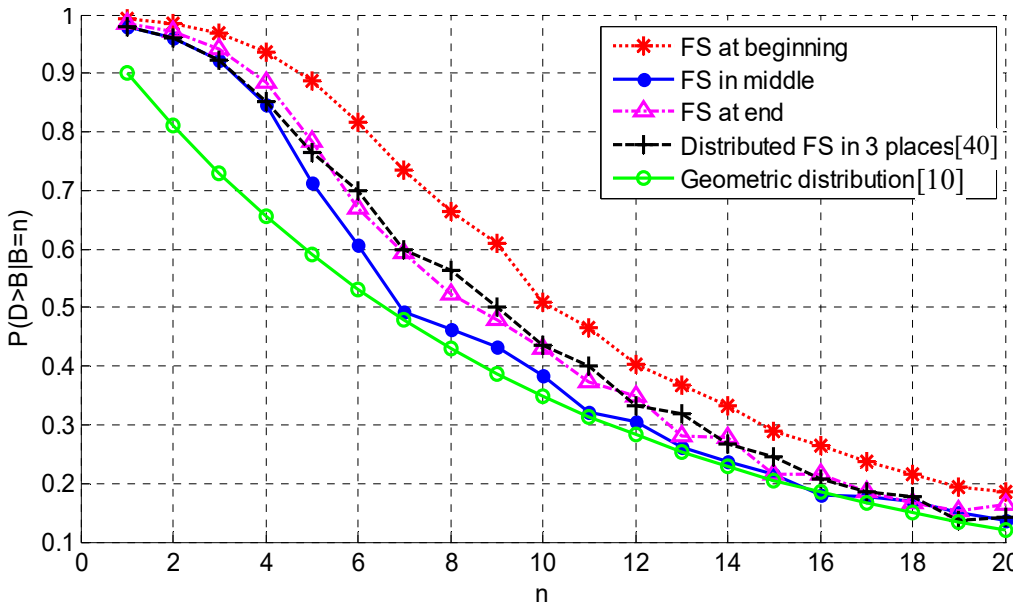


Figure 4-3. A plot of DPF versus n, value of delay, for $\epsilon = 0.1$ and $p_0 = 0.8$.

Using the proposed DPF measure, $\theta(q_1, q_2, n)$, delay of error detection can be calculated for each FS configuration. As expected, $\theta(q_1, q_2, n)$ is greater than the ideal case based on the geometric distribution. If the FS subintervals are subdivided to tiny parts and distributed uniformly, error detection delay would have geometric distribution. However, this is impossible due to the structure of subdivision process in arithmetic coding. Hence,

the closer DPF to geometric model, the better the FS configuration. The configuration *FS in middle* provides the closest DPF to the geometric model.

4.2.2. Probability of the missed error detection

In [42], the optimum q_1 and q_2 are calculated to minimize a union bound on bit error probability of an FSM model for AC. The dominant term in the union bound is the distance spectrum [38] that represents the average number of valid bit sequence pairs at the Hamming distance d_{free}, where d_{free} is the free distance [38], *i.e.,* the minimum Hamming distance between all two valid bit sequences.

The free distance d_{free} equals 1 for low redundancy cases with $\epsilon < 0.4$ [42]. In other words, two valid bit sequences have at least one different bit. In view of the relation between the free distance and the upper bound of bit error rate, a new figure of merit for FS configuration is now proposed. This figure of merit determines the probability of the missed detection (PMD) of an error caused by flipping only one bit in ρ bits. Suppose the random variable B represents the position of the erroneous bit (counted from the end) in the bit sequence of length ρ with only one bit in error, and $P(B = b) = \frac{1}{\rho}$. If the error detection delay is greater than b, the decoder cannot detect that error using the ρ bits. PMD which is a function of $q_1$and q_2, is defined as $\pi(q_1, q_2) = \sum_b P(D > B, B = b)$.

Suppose that all DSI subintervals are generated iteratively as previously explained. The likelihood of the drifted DBI residing in a valid DSI subinterval determines PMD, and

it is dependent on the FS configuration. All the possible combinations of (v, b) are examined, and $\pi(q_1, q_2)$ is calculated as

$$\pi(q_1, q_2) = \sum_b P(D > B|B = b)P(B = b) = \frac{\sum_b \sum_v \big(\alpha(v).\beta(v, b)\big)}{\rho \times \sum_v \alpha(v)}, \tag{4-6}$$

which, $P(D > B|B = b)$ is calculated in (4-5). The PMD defined above is the ratio of the number of DBI drifts from the valid DSI subintervals to the valid ones and the total number of possible drifts from the valid DSI subintervals by flipping one bit.

The PMD $\pi(q_1, q_2)$ is calculated for parameters $p_0 = 0.8$, $\epsilon = 0.1$, $\lambda = 200$ and $\rho = 20$, and it is depicted in Figure 4-4. In this calculation, $q_1$and q_2 are varied in steps of 0.05. It can be seen that the optimum point is around the origin ($q_1 = 0,\ q_2 = 0$), which means that the FS placed entirely between the valid symbols 0 and 1 is the best. The worst FS configuration is obtained by placing the FS beside the most probable symbol ($q_1 = 1$, $q_2 = 0$).

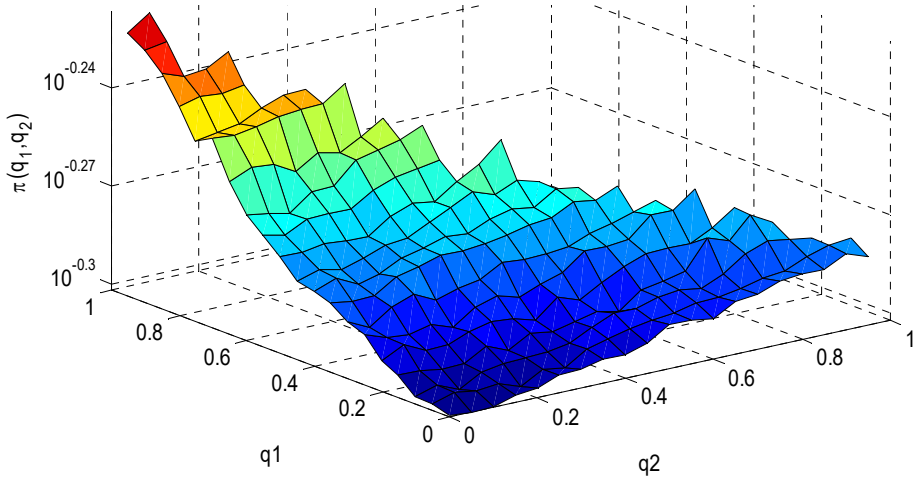


Figure 4-4. A plot of the PMD, $\pi(q_1, q_2)$, when p_0 =0.8, $\epsilon = 0.1$ and $r = 20$.

Figure 4-5 shows the effect of changing p_0 on the optimum q_1 and q_2 and the corresponding q_3 in terms of $\pi(q_1, q_2)$. The optimum q_1 and q_2 are almost equal to zero for all p_0. Thus, the best location for the FS is between the valid symbols irrespective of the probabilities of the valid symbols.

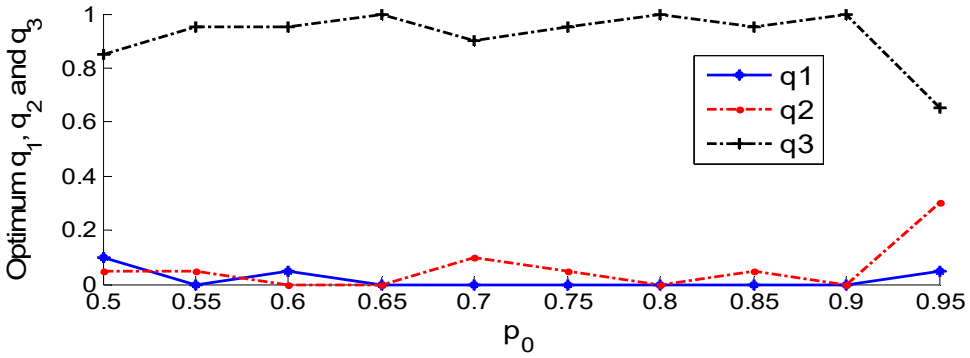


Figure 4-5. Optimum q_1, q_2 and q_3 versus p_0 for $\epsilon = 0.1$ and $r = 20$.

In Figure 4-6 various DSI subintervals after 4 levels of subdivision (4 decode bins) are illustrated for $p_0 = 0.7$. Distributing FS subintervals as evenly as possible causes the minimum distance between every pair of valid DSIs to be maximized, explicitly maximizing the free distance d_{free}. Increasing the free distance leads to improve the error detection. Based on this figure *FS at beginning* is the worst FS configuration, since in this case, the FS subintervals tend to cluster together and make a larger FS subinterval. *FS at end* results in clustering as well, but the clustered FS subintervals are not as large as *FS at beginning*, since probability of symbol 0 is greater than symbol 1. Thus, this configuration performs better than the *FS at beginning.* In the *FS in middle*, the clustering phenomenon is prevented, since in this case FS subintervals are not adjacent. This configuration tends to distribute the FS subintervals more evenly and in turn increases the minimum distance of valid subintervals which results in lower error rate.

In Figure 4-7 the DSI subintervals after 4 levels of subdivision are illustrated for $p_0 = 0.9$. Comparing this figure and Figure 4-6 shows that when the bin probabilities are more unbalanced, the clustering phenomenon makes wider concatenated FS subintervals. Thus, the FS configuration selection is more important when the bins are more unequally probable.

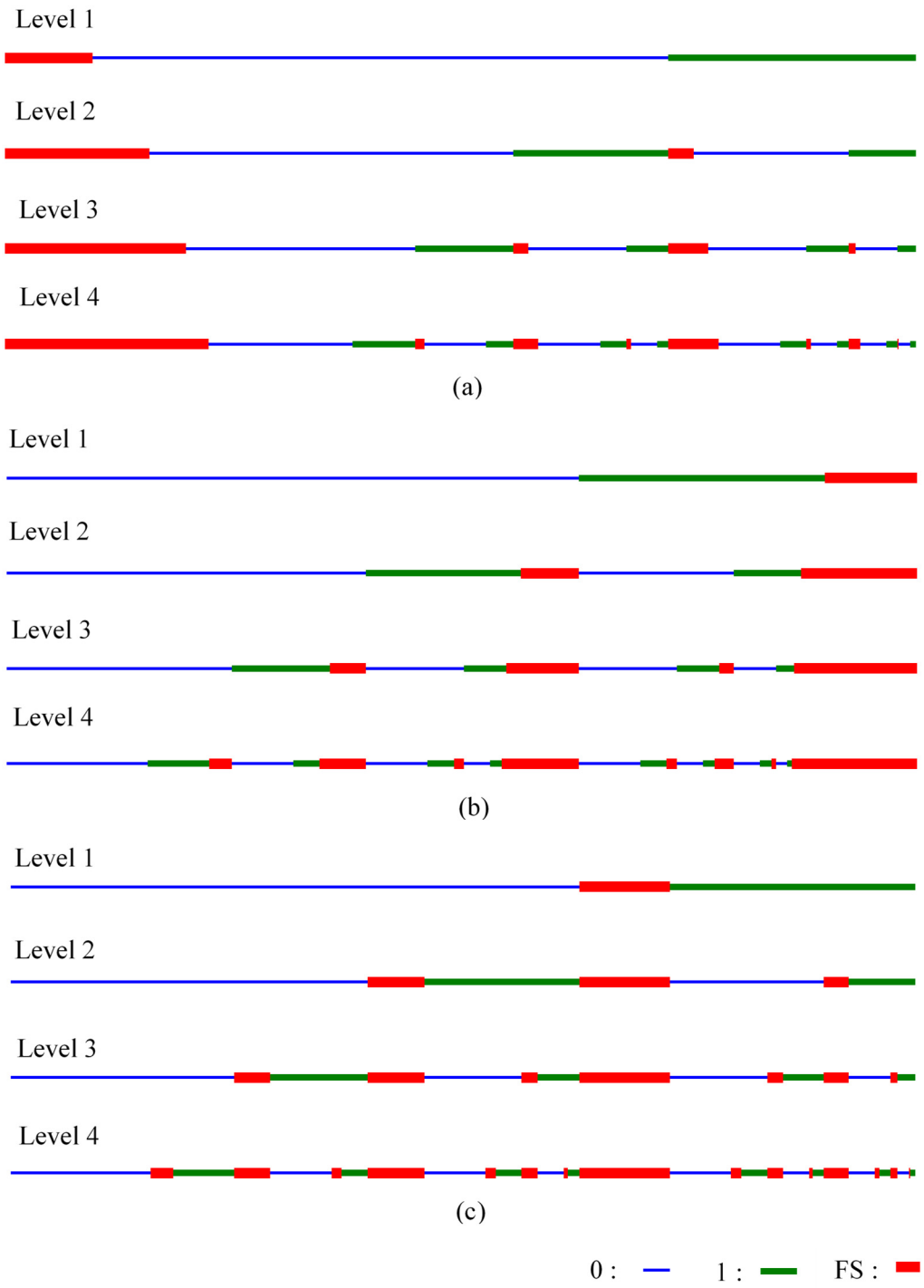


Figure 4-6. FS distribution after 4 levels of subdivision of DSI for (a) *FS at beginning* (b) *FS at end* and (c) *FS in middle* when $p_0=0.7$.

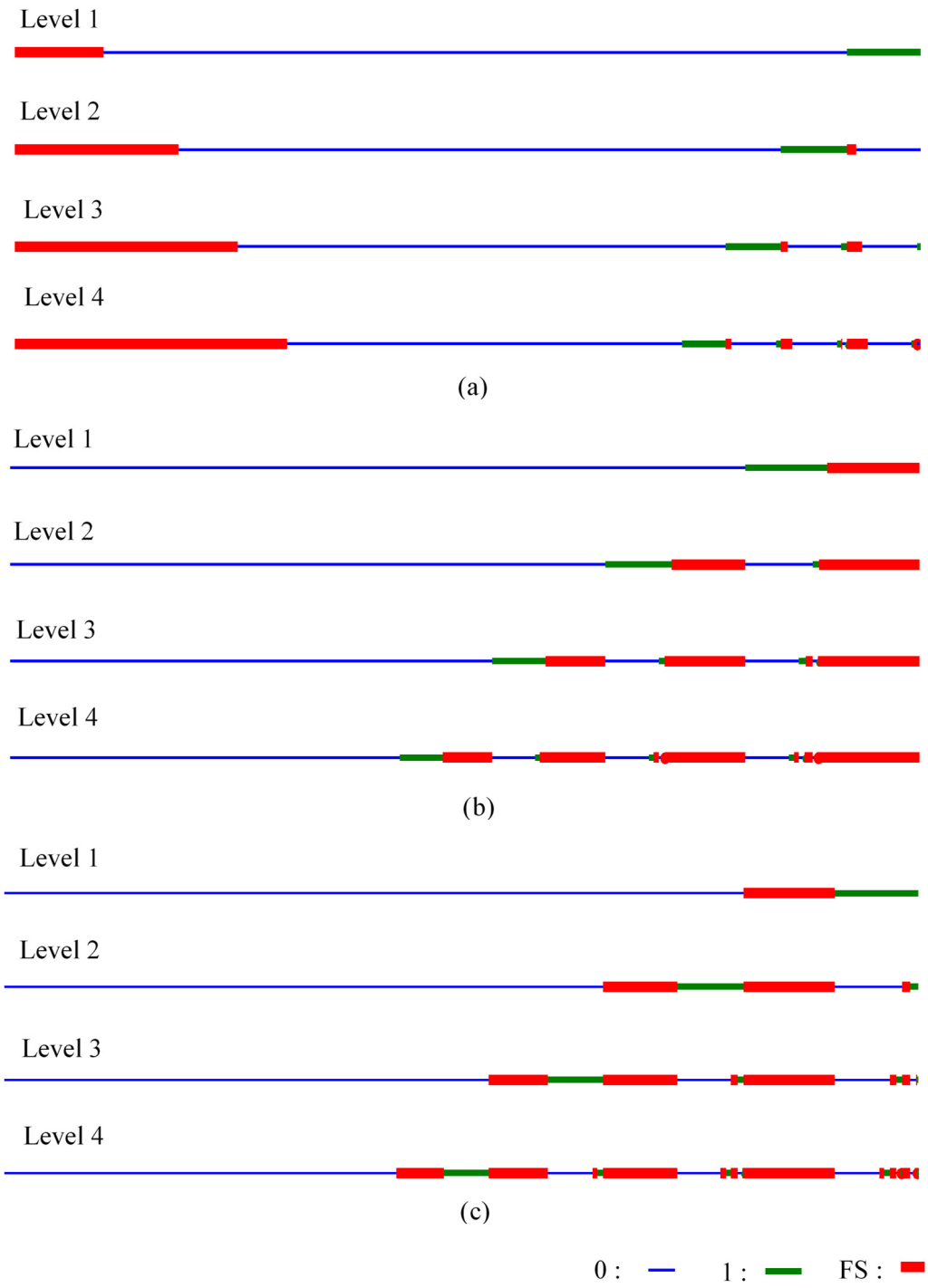


Figure 4-7. FS distribution after 4 levels of subdivision of DSI for (a) *FS at beginning* (b) *FS at end* and (c) *FS in middle* when $p_0=0.9$.

4.3. Simulation results

In the simulations, binary phase shift keying (BPSK) modulation is used. The received noisy signal y_k are demodulated to hard bits r_k. Therefore, the channel transition probability is given by Equation (3-12). In the AWGN channel, the probability of the error, p, can be written as a function of the signal-to-noise ratio (SNR), *i.e.*, $p = Q(\sqrt{2 \times SNR})$.

M is chosen as 8 in the MAP decoder using the M-algorithm which is explained in Chapter 3 and [27]. The bin sequence is randomly generated with various values of bin *a priori* probabilities $p_0 = 0.6, 0.7, 0.8, \text{and } 0.9$ and packetized with length of 250. The tests are conducted for FS probability $\epsilon = 0.1$ and EOPS probability $\delta = 0.01$. The simulations are repeated, until at least 2×10^4 erroneous packets are decoded. The decoder has *a priori* knowledge of the packet length and p_0. Figure 4-8 to Figure 4-11 show the plot of the packet error rate (PER) as a function of SNR for the various FS configurations mentioned earlier in Section 4.2, *i.e., FS at beginning, FS at end, FS in middle* and *distributed FS in 3 places* [40] when $p_0 = 0.9, 0.8, 0.7$ and 0.6 respectivley. In the calculation of PER, if there is at least one erroneous bin in the decoded packet, the entire packet is considered to be in error. It can be seen that the *FS in middle* is the best configuration among the four configurations for all SNRs. Use of the optimum FS configuration outperforms the PER especially for higher SNRs without any loss in compression efficiency. The simulation results indicate that the best place for the FS subinterval is in between the two valid

subintervals. By comparing these figures, it can be seen that when the probability of bins 0 and 1 are more unbalanced, changing the FS configuration has more effect on PER. Because, in such conditions the clustering phenomenon makes wider joint FS intervals as shown in Figure 4-6 and Figure 4-7. The only FS configuration without clustering is *FS in middle.*

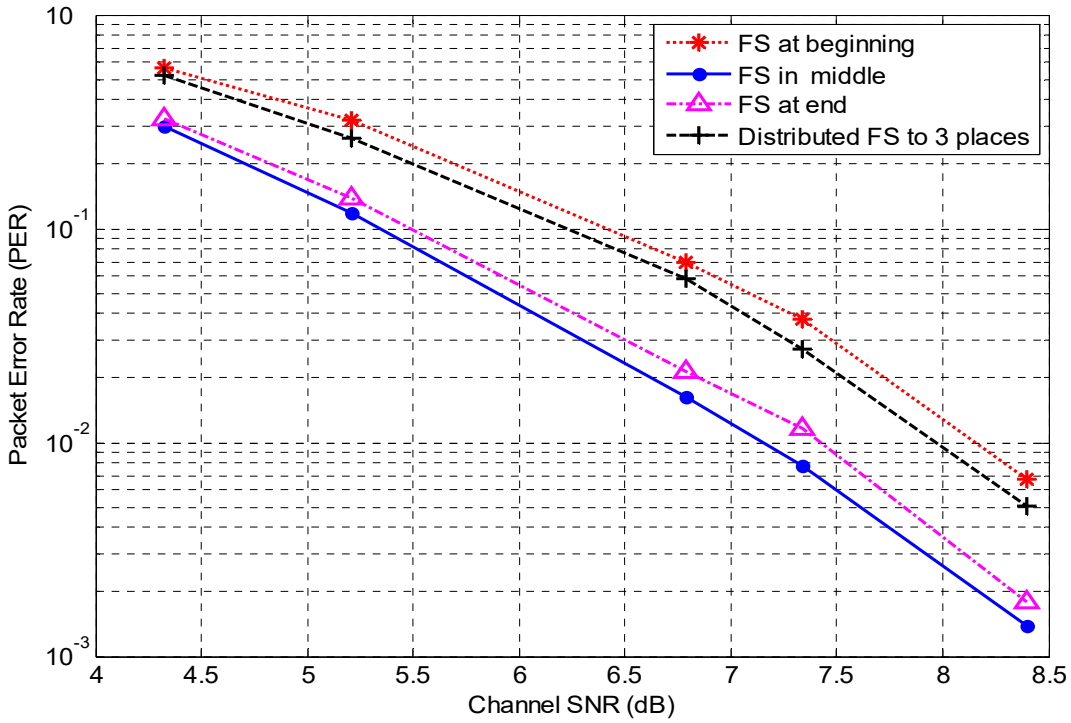


Figure 4-8. Plot of PER versus channel SNR for FS probability $\epsilon = 0.1$ and $p_0=0.9$.

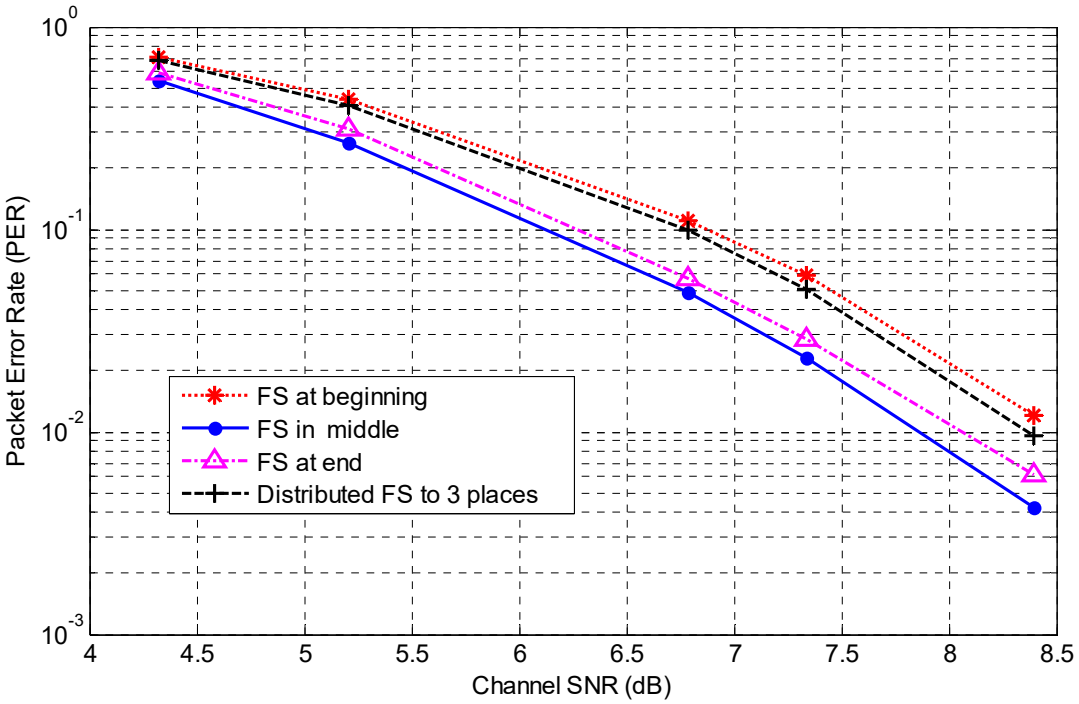


Figure 4-9. Plot of PER versus channel SNR for FS probability $\epsilon = 0.1$ and $p_0=0.8$.

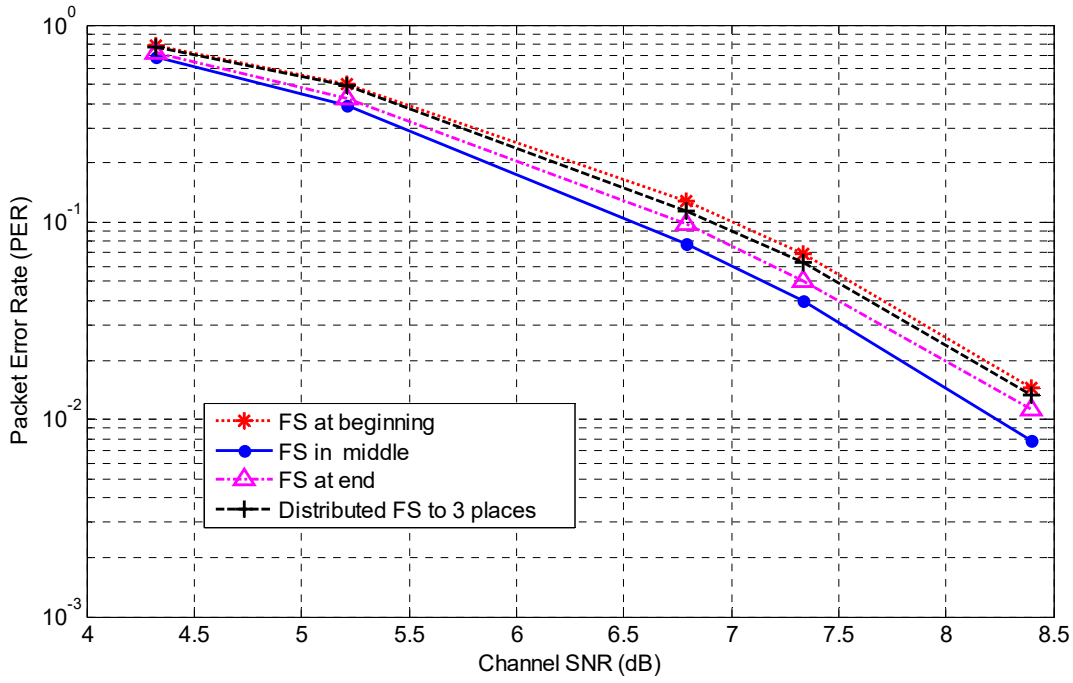


Figure 4-10. Plot of PER versus channel SNR for FS probability $\epsilon = 0.1$ and $p_0=0.7$.

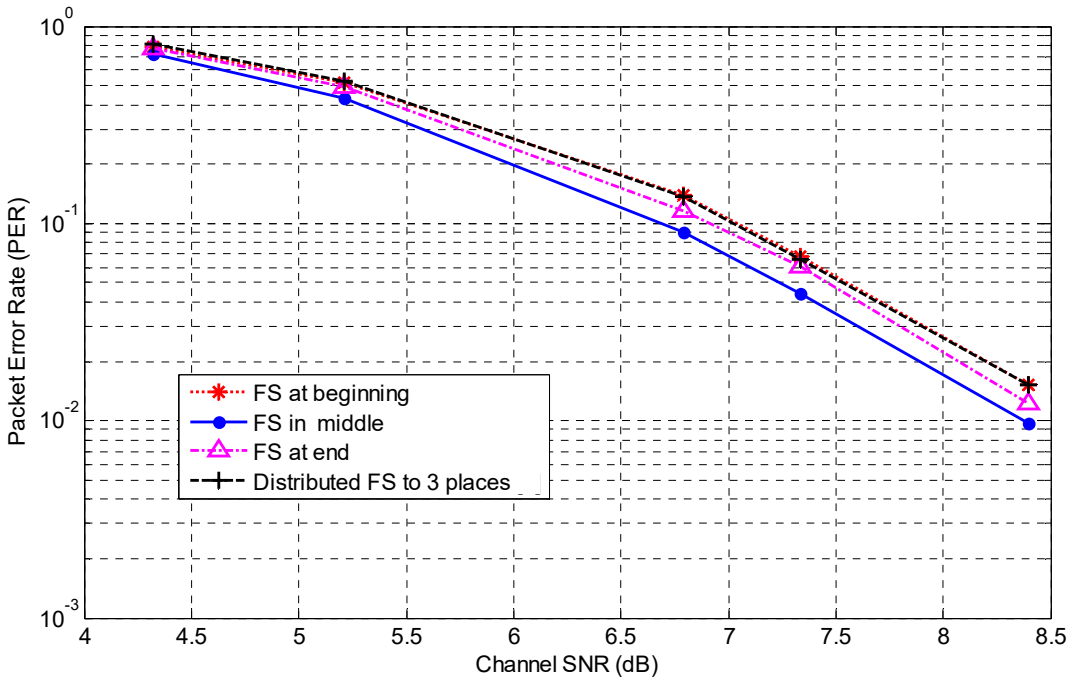


Figure 4-11. Plot of PER versus channel SNR for FS probability $\epsilon = 0.1$ and $p_0=0.6$.

4.4. Summary

In this chapter, a new figure of merit has been proposed for evaluating the FS configurations in arithmetic codes. This figure of merit measures the probability of the missed error detection caused by alteration of one bit in the data stream. The best FS position respecting the proposed figure of merit is in between the valid intervals. The simulation results have demonstrated the effectiveness of the proposed figure of merit and provide results consistent to the packet error rate. It has been shown using the optimized FS configuration reduces the error rate considerably without increasing the amount of redundancy or imposing additional overhead to the system. It is shown the FS configuration

selection is more important when the bins are unequally likely, due to clustering phenomenon.

The effect of changing FS configuration on the FS forecasting algorithm in [65] can be studied as the future work. In this algorithm, the effective FS interval expanded ahead. The FS configuration has effect on the expansion process. Thus, choosing the proper FS configuration could change performance of the system.

CHAPTER 5. Improvement of the MAP decoder using interdependencies of syntax elements

5.1. Introduction

The goal of a source encoder is to compress a signal. For this purpose, it exploits the redundancy in the input data and removes it. However, in practice, some residual redundancy remains in the source encoder output. This residual redundancy can be seen as being due to suboptimal source coding. This suboptimality could be a result of either the mismatch between the statistical models used in the encoder and the actual statistics of the encoder input, or of a constraint on the latency and complexity of the encoder [66]. As explained in Chapter 3, the decoder can use the residual redundancy for error correction of the data received.

The redundancy can be observed in the form of a non-uniform distribution or memory in the output bits of the encoder. In [60], a MAP decoder exploits the redundancy by modeling the output using a first-order Markov process [39]. In this model, only the dependency between the current decoding bit and previous ones are considered. In [61], a MAP decoder is introduced that uses the interdependency among the current bit, the past bits, and future bits to exploit the residual redundancy. In [67], the redundancy is exploited using a generic Markov model for source encoder output. In these methods, the residual redundancy is exploited by statistically modeling the encoder output without considering the semantic and syntactic information of the encoder input and in turn the decoder output.

In [43], the motion vectors in *inter* frames are modeled using a first-order Markov process. The correlation between motion vectors is used in a MAP decoder to reduce the error rate. In this method the parameters of the statistical model for the motion vectors are calculated at the encoder and transferred as side information to the decoder through an error-free channel. Obviously, this error-free transmission imposes an overhead on the system.

In [44], the syntax elements (SE) of H.264 are statistically modeled. The decoder divides the input bit sequence into variable length bit sequences (input codewords), each of which is decoded to an SE using the variable length code (VLC) [3]. At each stage, it generates all possible codeword candidates and their decoding metric and finally it keeps only the candidate maximizing the decoding metric and complying with the video compression standard. There are three problems with this method. First, errors in a codeword may result in syntactic errors in next stages but not in the current stage. Second, the statistical model may not always be reliable for all conditions, *e.g.,* non-stationary processes [39], and may mislead the decoder. Third, codeword-by-codeword decoding is not applicable for AC.

To address these issues, a sequential MAP decoder is proposed that utilizes a breadth-first algorithm (MA) for constructing the decoding tree as explained in Section 3.2.3. The decoding tree is constructed channel bit by channel bit and the decoding metric is calculated in each stage for the candidates based on decoder input and the *a priori* information exploited from the interdependencies in previously decoded syntax elements. In this chapter, the *intra* modes are modeled as highly correlated syntax elements (as shown

in Section 2.3.1). The statistics of each *intra* mode are calculated from previously decoded *intra* modes. A method is proposed to categorize the statistics as reliable and unreliable. In particular, when the previously decoded *intra* modes are too diverse, the calculated statistics are deemed unreliable for the current syntax element. To measure the diversity, the entropy of the *intra* modes of previously decoded neighboring *intra* blocks is calculated. If the entropy is more than a predefined threshold, the statistics are deemed unreliable. In the unreliable cases the decoder neglects the *a priori* probabilities estimated and only uses the channel information in the decoder metric. The performance of the proposed decoder is evaluated using simulation. The simulation results show 1% to 13% PER reduction comparing to ML decoding.

This chapter is organized as follows. In Section 5.2, the structure and metric calculation of the proposed MAP decoder is provided. In Section 5.3 , the effect of using *intra* mode statistics in the decoder is represented. Section 5.4 explains the method for estimating the statistics of the *intra* modes, and in Section 5.5 the method for categorizing the *intra* modes statistics is proposed and the MAP decoder is adjusted using this method. The simulation and results are provided in Section 5.6 and the chapter is summarized in Section 5.7.

5.2. Using the interdependencies of syntax elements in MAP decoder

In this section, a new MAP arithmetic decoder is proposed that uses the interdependencies between syntax elements as *a priori* information. The block diagram of the decoder is shown in Figure 5-1. Similar to the MAP decoder introduced in Chapter 3

and [64], the DCG generates the bit sequence candidates $\tilde{\boldsymbol{u}}^j$ with length j and in turn the corresponding bin sequences $\boldsymbol{s}_{\tilde{\boldsymbol{u}}^j}$ and syntax element sequences $\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}$. The symbol checker discards the candidates that produce a FS. The syntax checker eliminates candidates that violate the syntax or semantics of the H.264 standard. The *a priori probability* (APP) estimator calculates the probability of $\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}$ and feeds it back into the DCG to update the decoding metric of the candidate.

The candidate generation is done in a fashion similar to what was done in Chapter 3, using a decoding tree. The MA is used as the suboptimum search technique which keeps only the best M candidates at each level of the decoding tree (see Figure 3-6).

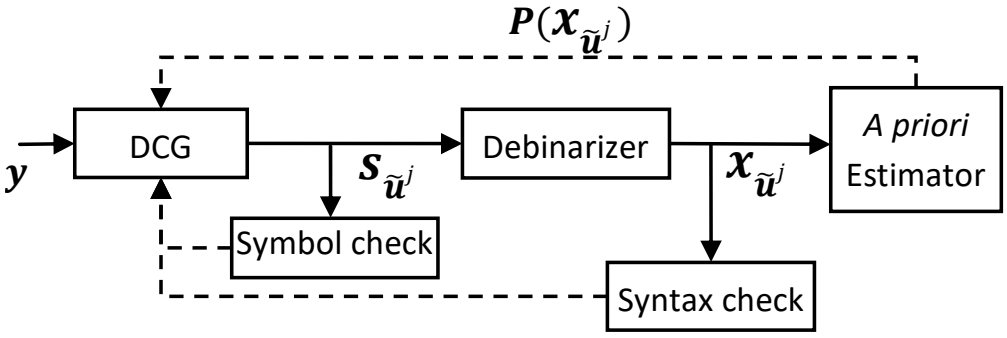


Figure 5-1. Block diagram of the MAP decoder using *a priori* probability of syntax elements.

To consider the interdependencies between syntax elements, Equation (3-3) is used. By applying the logarithm function on the argument of Equation (3-3), the metric corresponding to each candidate $\tilde{\boldsymbol{u}}$ in additive form as

$$m_{\tilde{\boldsymbol{u}}} = \log\big(P(\boldsymbol{y}|\tilde{\boldsymbol{u}})\big) + \log(P(\boldsymbol{x}_{\tilde{\boldsymbol{u}}})) - \log(P(\boldsymbol{y}))\,. \tag{5-1}$$

By using the assumptions that the channel noise is white and bits of $\boldsymbol{y}$ are independent, the decoding metric assigned to each node of the decoding tree at the stage of j corresponding to candidate $\tilde{\boldsymbol{u}}^j$ is written as

$$m_{\tilde{\boldsymbol{u}}^j} = \sum_{k=1}^{j}\{\log\big(P(y_k|\tilde{u}_k)\big)\} + \log\Big(P\big(\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}\big)\Big) - \sum_{k=1}^{j}\log\big(P(y_k)\big), \tag{5-2}$$

where y_k and $\tilde{u}_k$ are the k^{th} bit of $\boldsymbol{y}$ and $\tilde{\boldsymbol{u}}$, respectively, and $\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}$ is the syntax element sequence that results from decoding the first j bits of $\tilde{\boldsymbol{u}}$, *i.e.,* $\tilde{\boldsymbol{u}}^j$. The last term, *i.e.,* $\sum_{k=1}^{j}\log\big(P(y_k)\big)$, is identical for all candidates at the same stage j. Thus, it is insignificant and it is discarded in the metric calculation.

Three approaches are possible to estimate the middle term, *i.e.,* $\log\Big(P\big(\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}\big)\Big)$. The first one is to use the simplifying assumption that all syntax element sequences are equiprobable. The second approach is to calculate the *a priori* probabilities at the encoder and send them as side information. The third scenario is to estimate the *a priori* probabilities using the interdependencies between syntax elements. In the first scenario, this term can be neglected and in turn the decoder would be converted to a maximum likelihood (ML) decoder [45]. The candidate metric for the ML decoder with MA is

$$m_{\tilde{\boldsymbol{u}}^j} = \sum_{k=1}^{j} \log\big(P(y_k|\tilde{u}_k)\big), \tag{5-3}$$

in which only the channel transition probability is considered. The second scenario is undesirable, due to overhead caused by the side information. In the third scenario, the term $\log\big(P(\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j})\big)$ is expanded by means of the chain rule [39] as

$$\log\big(P(\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j})\big) = \log\big(P(\boldsymbol{x}_{\tilde{\boldsymbol{u}},1})\big) + \sum_{k=2}^{j} \log\Big(P\big(\boldsymbol{x}_{\tilde{\boldsymbol{u}},k}\big|\boldsymbol{x}_{\tilde{\boldsymbol{u}}^{k-1}}\big)\Big), \tag{5-4}$$

where $\boldsymbol{x}_{\tilde{\boldsymbol{u}},k}$ is the variable length syntax element sequence generated by decoding the k^{th} channel bit $\tilde{u}_k$. $\boldsymbol{x}_{\tilde{\boldsymbol{u}},k}$ is dependent not only on $\tilde{u}_k$ but also on the internal state of the MAP decoder after decoding the first $k-1$ bits of $\tilde{\boldsymbol{u}}$, *i.e.,* $\tilde{\boldsymbol{u}}^{k-1}$. The effect of the interdependency can be seen in the conditional probability $P\big(\boldsymbol{x}_{\tilde{\boldsymbol{u}},k}\big|\boldsymbol{x}_{\tilde{\boldsymbol{u}}^{k-1}}\big)$.

One of the challenges in the metric calculation of the MAP decoder is that the lengths of the generated syntax element sequences are different in each node of the decoding tree at stage j. In other words, $\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}$ in Equation (5-2) has various length for different realizations of $\tilde{\boldsymbol{u}}^j$. Since the probability of a syntax element sequence realization generally decreases with the growth of its length (considering the chain rule), it is decided that the maximization process should take the same number of syntax elements into consideration. Thus, Equation (5-2) is adjusted in such a way that the decoder uses syntax element sequences with the same length in calculation of the metric.

Suppose ℓ_j is the minimum length of $\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}$ for all paths in stage j, *i.e.*

$$\ell_{\mathrm{j}} = \min_{\tilde{\boldsymbol{u}}^j}(length(\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j})). \tag{5-5}$$

By considering the first ℓ_j syntax elements of $\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}$, a truncated sequence is constructed which is named $\overline{\boldsymbol{x}}_{\tilde{\boldsymbol{u}}^j}$. Indeed, sequences $\overline{\boldsymbol{x}}_{\tilde{\boldsymbol{u}}^j}$ for various $\tilde{\boldsymbol{u}}^j$ have the same length, ℓ_j. In decoding metric (see Equation (5-2)), $P\left(\boldsymbol{x}_{\tilde{\boldsymbol{u}}^j}\right)$ is replaced by $P\left(\overline{\boldsymbol{x}}_{\tilde{\boldsymbol{u}}^j}\right)$. In the interests of simplifying the notation, $\overline{\boldsymbol{x}}_{\tilde{\boldsymbol{u}}^j}$ is replaced by $\overline{\boldsymbol{x}}$. The probability $P(\overline{\boldsymbol{x}})$ is calculated based on the chain rule and as,

$$\log\left(P(\overline{\boldsymbol{x}})\right) = \log(P(\bar{x}_1)) + \sum_{n=2}^{\ell_j} \log\left(P(\bar{x}_n|\overline{\boldsymbol{x}}^{n-1})\right). \tag{5-6}$$

where $\overline{\boldsymbol{x}}^{n-1}$is the first $n-1$ elements of $\overline{\boldsymbol{x}}$ and $\bar{x}_n$ is the n^{th} element of $\overline{\boldsymbol{x}}$.

$P(\bar{x}_n|\overline{\boldsymbol{x}}^{n-1})$ represents the interdependency between each syntax element and its previously decoded predecessors. This probability can be calculated from a statistical model for the syntax elements which is explained in Section 5.4.

The syntax elements can be adjacent in two different senses. The first one is data stream adjacency when they are consecutive in the data stream. The second one is spatiotemporal adjacency, when they belong to a block or a macroblock to which they are spatially or temporally adjacent. Spatiotemporally adjacency does not necessarily result in

data stream adjacency and vice versa. Spatiotemporally adjacent syntax elements are often highly correlated. By assuming that there is only interdependency among the spatiotemporal adjacent syntax elements and ignoring the dependencies between non-spatiotemporal adjacent syntax elements the conditional probability $P(\bar{x}_n|\bar{\boldsymbol{x}}^{n-1})$ is rewritten as

$$P(\bar{x}_n|\bar{\boldsymbol{x}}^{n-1}) = P(\bar{x}_n|[\![\bar{\boldsymbol{x}}^{n-1}]\!]). \tag{5-7}$$

where $[\![\bar{\boldsymbol{x}}^{n-1}]\!]$ is the subset of $\bar{\boldsymbol{x}}^{n-1}$ whose elements are spatiotemporally adjacent with the syntax element $\bar{x}_n$. In other words, $[\![\bar{\boldsymbol{x}}^{n-1}]\!]$ is a sequence of syntax elements that are previously decoded and spatiotemporally adjacent to the current syntax element $\bar{x}_n$.

Depending on the syntax element type, various categories of spatiotemporal adjacency would be considered. For example, when the decoder decodes *intra* modes, only spatial adjacency is taken into account and the macroblocks at the left, right and up side of the current macroblock are considered as neighbors. Finally, the MAP decoding metric for MA in Equation (5-2) would be in the form of

$$m_{\tilde{\boldsymbol{u}}^j} = \sum_{k=1}^{j} \{\log(P(y_k|\tilde{u}_k))\} + \log(P(\bar{x}_1)) + \sum_{n=2}^{\ell_j} \{\log(P(\bar{x}_n|[\![\bar{\boldsymbol{x}}^{n-1}]\!]))\}. \tag{5-8}$$

5.3. Transmitting *a priori* probability of *intra* modes as side information

In the MAP decoding, the decoder uses *a priori* information in the calculation of the metric. This information can be transmitted to the decoder as *a priori* probabilities. Of course, transmitting this information to the decoder imposes an overhead to the system. Alternatively, the decoder can estimate this information from syntax elements that have already been decoded, saving the overhead of a side channel. A method to do this is explained in Section 5.4.

The effect of using the *a priori* information of syntax elements is demonstrated by a benchmark simulation. In this simulation the video *foreman* is *intra* encoded. The *intra* mode distribution in every MB is calculated at the encoder and sent to the decoder. The histogram of *intra* modes of blocks inside the current MB (MB_x) is used as an estimate of the local PMF of the *intra* modes.

In calculation of the histogram, the count of every mode (m) are presumed to be at least 1 as denoted in Equation (5-9),

$$hist(m, MB_x) = \max\left(1, \sum_{block\ b \in MB_x} \delta_d(mode(b) - m)\right), \tag{5-9}$$

where $mode(b)$ is the *intra* mode of block b and $\delta_d(.)$ is discrete Dirac function.

The calculated PMF is normalized histogram function, thus the minimum value of 1 condition, ensures that the PMF is not zero anywhere in its domain. As the value of the PMF is used as part of the decoding metric (which takes the logarithm of event

probabilities), this condition prevents the decoding metric to be $-\infty$ for any *intra* mode. In other words, this PMF calculation does not imply an impossible mode. Impossible modes are only implied by the position of the current *intra* block in the slice as explained in Section 3.3. The *intra* mode PMF of the current MB is calculated by normalizing the histogram function as

$$\hat{P}(mode(b) = m) = \frac{hist(m, MB_x)}{\sum_{\tilde{m}} hist(\tilde{m}, MB_x)} \quad : \; b \in MB_x. \tag{5-10}$$

where b is an *intra* block in the current MB. This function is calculated for every MB at the encoder and transmitted to the decoder to use it as the *a priori* probability of *intra* modes.

In Table 5.1, the error rates for the soft ML decoder and soft MAP decoder, using this *a priori* information, are compared. In this simulation, the probability of FS and EOPS are set to 0.001 and 0.01 respectively and SNR is 5.208dB which corresponds to channel error rate equal to 5×10^{-3}. The parameter M is set to 256 for the breadth first algorithm (MA). A significant reduction in error rates for the MAP decoder is borne out by comparing the results of the two algorithms shown in Table 5.1. However, This MAP decoder needs the distribution of the modes locally for every MB to use in the decoding metric. Transmitting this information imposes an additional overhead to the system. In order to prevent this overhead, this distribution is estimated based on the previously decoded *intra* modes, as explained in Section 5.4.

Table 5.1. Comparing the performance of the ML decoder and MAP decoder uses transmitted *a priori* probabilities of *intra* modes.

	ML	MAP
PER	9.17E-03	7.69E-03
SER	3.10E-04	1.45E-04
BER	3.88E-05	3.29E-05
SEER	3.80E-04	1.76E-04

5.4. PMF estimation of *intra* modes

In *intra* mode signaling, H.264 uses a simple prediction method for MPM which is explained in Chapter 2. However, it cannot remove all of the redundancy in the *intra* modes. Therefore, a more complex predictive method at the decoder can be used to decrease the error rate by exploiting the residual redundancy in *intra* modes.

In this section a method is proposed to calculate the PMF of the *intra* modes in the current MB from the decoded neighboring MBs. This *intra* mode distribution is the conditional probability $P(\bar{x}_n|[\![\overline{\boldsymbol{x}}^{n-1}]\!])$ of the MAP decoding metric provided in Equation (5-8). The more accurate the PMF estimation, the better the performance.

5.4.1. Adaptive *intra* mode PMF estimation

The encoder scans the MBs of a frame in a raster fashion, *i.e.* row by row. The syntax elements in three adjacent MBs at top, left and top-left side of the current MB are considered as spatially adjacent to the syntax elements in the current MB as shown in Figure 5-2. The current MB is denoted by MB_x and the union of the adjacent MBs is

denoted by MB_{adj}, *i.e.* $MB_{adj} = MB_L \cup MB_U \cup MB_{UL}$. In Equation (5-8), $\bar{x}_n$ is the mode of the n^{th} *intra* block in MB_x and $[\![\bar{\boldsymbol{x}}^{n-1}]\!]$ is set of all *intra* modes decoded in MB_{adj}.

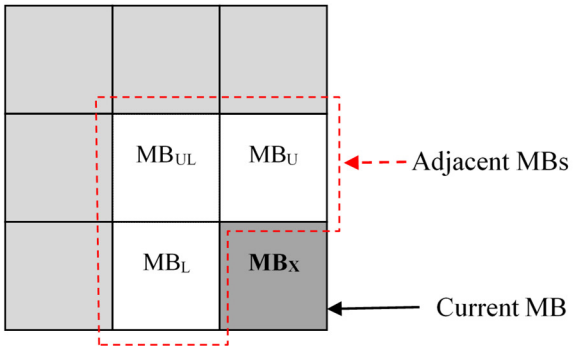


Figure 5-2. Spatially adjacent MBs to the current MB.

Similar to Equation (5-9), the *intra* mode histogram of these three neighboring MBs is calculated as

$$hist(m, MB_{adj}) = \max\left(1, \sum_{block\ b' \in MB_{adj}} \delta_d(mode(b') - m)\right). \quad (5\text{-}11)$$

The simplest estimation for *intra* mode PMF of the current MB based on its adjacent MBs is the normalized histogram as follows,

$$\hat{P}(mode(b) = m) = \frac{hist(m, MB_{adj})}{\sum_{\tilde{m}} hist(\tilde{m}, MB_{adj})} \quad : \quad b \in MB_x. \quad (5\text{-}12)$$

In this calculation it is assumed that all blocks in a MB have the same *intra* mode PMF, no matter where the block is located in the MB. This form of estimation is used in [44] as the *intra* mode *a priori* probability. In this estimation the transition between MB_{adj} and MB_x is not taken into account and the PMF calculated from MB_{adj} is used directly as an estimation for the PMF in MB_x.

To calculate the PMF of *intra* block modes in MB_x more accurately, a conditional probability that relates the *intra* block mode patterns in MB_{adj} and MB_x *intra* blocks should be used. These probabilistic relations imply that *intra* modes in MB_x is a Markov information source where its predecessor state is the *intra* block mode pattern of MB_{adj}. The number of possible patterns for *intra* modes in MB_{adj} is a large number. For example, if MB_{adj} consists of 4×4 *intra* blocks, the number of possibilities is equal to $(9)^{3\times16} \cong 6.3 \times 10^{45}$. The large number of possibilities makes precalculation of the conditional probabilities infeasible.

As mentioned in Section 2.3, the *intra* mode of adjacent *intra* blocks are highly correlated. In [68], the video compressor uses this correlation to speed up the *intra* mode selection process by limiting the search domain only to the most highly probable (MHP) modes. In this approach three *intra* modes are labeled as MHP modes. One mode is the mode of adjacent *intra* block, and the other two modes are *intra* modes neighboring the mode of adjacent *intra* block in angular manner (see Figure 2-5). In other words, this can be viewed as a probabilistic model. When probability of a specific directional mode is 1 in the adjacent *intra* block, probability of that mode and its two angular neighbors are more probable than the other modes. Similarly, there is a probabilistic relation between

directional *intra* modes of *intra* block in the current MB, and its neighbors which can be used to decrease the complexity of the Markov source model.

In order to moderate this complexity and also to statistically use the Markov information source model, instead of considering all possible patterns in MB_{adj} only the distribution of *intra* mode in MB_{adj} is taken into account. The relation between PMF of *intra* modes in MB_x and PMF of *intra* modes in MB_{adj} is modeled by a transition conditional probability function as shown in Figure 5-3. MB_{adj} is modeled as an information source which emits *intra* modes with a certain PMF, and the *intra* modes in MB_x are their deviated version. This deviation is modeled by a transition probability function. Emitting an *intra* mode by MB_{adj} can be interpreted as random selection of an *intra* block in MB_{adj}. Thus, the *intra* mode probability is estimated adaptively by normalizing the local histogram as

$$\hat{P}(mode(b') = m') = \frac{hist(m', MB_{adj})}{\sum_{\tilde{m}} hist(\tilde{m}, MB_{adj})} \quad : \quad b' \in MB_{adj}. \tag{5-13}$$

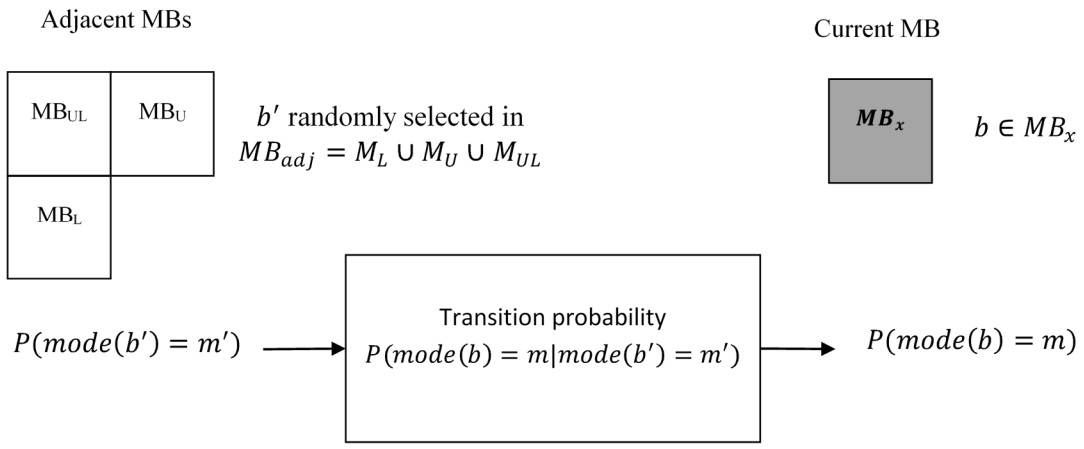


Figure 5-3. Transition between *intra* modes of current MB and adjacent MBs.

The transition probability function between block b in MB_x and a randomly selected block b' in MB_{adj} is represented by $P(mode(b) = m|mode(b') = m')$.

Using the total probability rule [39], the PMF of *intra* modes in the current MB is written as

$$\hat{P}(mode(b) = m) = \sum_{m'} P(mode(b) = m|mode(b') = m')\hat{P}(mode(b') = m') \; : \quad b \in MB_x \, , b' \in MB_{adj}, \tag{5-14}$$

which b is an *intra* block in MB_x and b' is a randomly chosen *intra* block out of MB_{adj}.

The first term is transition probability from the adjacent MBs to the current MB and the second term is PMF estimation of the adjacent MBs from Equation (5-13). If the transition probability function is

$$P(mode(b) = m|mode(b') = m') = \delta_d(m - m'), \tag{5-15}$$

the PMF estimator in equation (5-12) and (5-14) are equivalent. The remaining part of this section is dedicated to the calculation of this transition function.

5.4.2. Calculation of *intra* mode transition probability function

Intra modes are usually chosen by the encoder based on a rate-distortion optimization mechanism [55]. *Intra* mode direction in a MB are usually aligned to the image texture of the MB, *e.g.,* if the MB has lots of vertical lines the most probable *intra* mode in that MB is the vertical mode. In natural images, the texture varies gradually in adjacent MBs. Thus, directional *intra* modes of the current MB and its neighbors are highly correlated. If a directional mode is highly probable in the neighboring MBs, that mode and the modes close to it in an angular sense are highly probable in the current MB. In order to work with the *intra* modes in the angular order conveniently, instead of *intra* modes their indexes are considered in these calculations. The relation between the *intra* modes and their indexes is represented in Figure 2-6 and Table 2.1. The function $i_m = indx(m)$ maps the *intra* mode m to its index i_m. When i_m varies from 0 to 7, the directional *intra* modes varies in angular

order from diagonal-down to horizontal-up, and when $i_m = 8$ the *intra* mode is DC which is the only non-directional mode.

Due to the one-to-one relation between *intra* modes and their indexes, $\hat{P}(mode(b) = m)$ is equal to $\hat{P}(indx(mode(b)) = i_m)$. In the interests of simplifying the notation, $indx(mode(.))$ is replaced by function $inmd(.)$. Using this function, the PMF of *intra* modes in Equation (5-14) turns into

$$\hat{P}(mode(b) = m) = \hat{P}(inmd(b) = i_m) = \sum_{i'} P(inmd(b) = i_m | inmd(b') = i') \hat{P}(inmd(b') = i') \; : \; b \in MB_x \, , b' \in MB_{adj}. \tag{5-16}$$

The second term, $\hat{P}(inmd(b') = i')$, is calculated from the adjacent MB based on Equation (5-13). For the first term, $P(inmd(b) = i_m | inmd(b') = i')$, Three situations are possible which are separately analyzed as follows.

i) The first situation is $i' = 8$ which means the neighboring mode is DC. In such cases, since there is no directional information in the adjacent MBs, the PMF of *intra* modes in current MB is assumed to be uniform, *i.e.,*

$$P(inmd(b) = i_m | inmd(b') = 8) = \frac{1}{9} \; : \quad b \in MB_x \, , b' \in MB_{adj}. \tag{5-17}$$

ii) The second situation is $0 \leq i' \leq 7$ and $i_m = 8$ which means the neighboring mode is directional and the current mode is DC. As directional modes do not induce information in DC mode, the transition probability is $\frac{1}{9}$ similar to Equation (5-17). Thus, the transition probability would be in the form of

$$P(inmd(b) = 8 | inmd(b') = i') = \frac{1}{9} : \ 0 \leq i' \leq 7, b \in MB_x \, , b' \in MB_{adj}. \tag{5-18}$$

iii) The last situation is $0 \leq i' \leq 7$ and $0 \leq i_m \leq 7$, *i.e.,* current and neighboring modes are both directional. Angular difference function is defined as

$$angdiff(i_m, i') = \begin{cases} |i_m - i'| : 0 \leq |i_m - i'| \leq 4 \\ 8 - |i_m - i'| : 4 < |i_m - i'| \leq 7, \end{cases} \quad : \ 0 \leq i_m, i' \leq 7 \tag{5-19}$$

which shows angular difference between two *intra* mode indexes i_m and i'. Due to the circular positioning of the 8 directional modes, the angular difference between two *intra* mode indexes is positive and less than 4. Thus, when $|i_m - i'|$ is greater than 4, the angular difference is calculated as $8 - |i_m - i'|$. As current and adjacent *intra* modes are random variables, the output of this function is also a random variable whose range is between 0 and 4. This random variable is denoted by Ω. To measure the PMF of Ω for typical videos, several raw videos from database [56] (~11000 frames) in QCIF format are *intra* encoded, and the angular differences between directional *intra* modes of the blocks in every MB and

blocks in neighboring MBs are calculated. The PMF of Ω is calculated as the normalized histogram of the angular difference. This PMF is represented in Figure 5-4. When $\Omega = 4$, two directional modes in current and adjacent MBs are orthogonal which is least probable, and when $\Omega = 0$, two directional modes are in the same direction which is the most probable as expected. When $1 \leq \Omega \leq 3$, the *intra* mode index could alter either clockwise or counter-clockwise. The transition probability for this situation is expanded using the definition of conditional probability as

$$
\begin{aligned}
&P(inmd(b) = i_m | inmd(b') = i') = \\
&\qquad P(inmd(b) = i_m, 0 \leq inmd(b) \leq 7 | inmd(b') = i') = \\
&\qquad = P(inmd(b) = i_m | inmd(b') = i', 0 \leq inmd(b) \leq 7) \times \\
&\qquad P(0 \leq inmd(b) \leq 7 |\, inmd(b') = i') : \\
&\qquad\qquad 0 \leq i_m, i' \leq 7, b \in MB_x\,, b' \in MB_{adj}.
\end{aligned}
\tag{5-20}
$$

The second term, *i.e.,* $P(0 \leq inmd(b) \leq 7 |\, inmd(b') = i')$, is the complement event in Equation (5-18), and its probability is equal to $1 - \frac{1}{9} = \frac{8}{9}$. The first term is calculated using the *intra* mode angular difference distribution as

$$P(inmd(b) = i_m | inmd(b') = i', 0 \le inmd(b) \le 7) =$$

$$= \begin{cases} 0.5 \times P\big(\Omega = angdiff(i_m, i')\big) : 1 \le angdiff(i_m, i') \le 3 \\ P\big(\Omega = angdiff(i_m, i')\big) \quad : angdiff(i_m, i') = 0 \; or \; 4 \end{cases}$$

$$: 0 \le i_m, i' \le 7, b \in MB_x \, , b' \in MB_{adj} \quad (5\text{-}21)$$

The 0.5 coefficient behind the first statement is because when the angular difference is between 1 and 3, the *intra* mode direction may vary clockwise or counterclockwise which are assumed to be equiprobable. Consequently, the transition probability for this situation is

$$P(inmd(b) = i_m | inmd(b') = i'') =$$

$$= \begin{cases} \frac{4}{9} P\big(\Omega = angdiff(i_m, i')\big) \quad : 1 \le angdiff(i_m, i') \le 3 \\ \frac{8}{9} P\big(\Omega = angdiff(i_m, i')\big) \quad : angdiff(i_m, i') = 0 \; or \; 4 \end{cases}$$

$$: 0 \le i_m, i' \le 7, b \in MB_x \, , b' \in MB_{adj}. \quad (5\text{-}22)$$

The transition probability when both *intra* modes in the current MB and adjacent MBs are directional ($0 \le i_m, i' \le 7$) is only dependent on their angular difference.

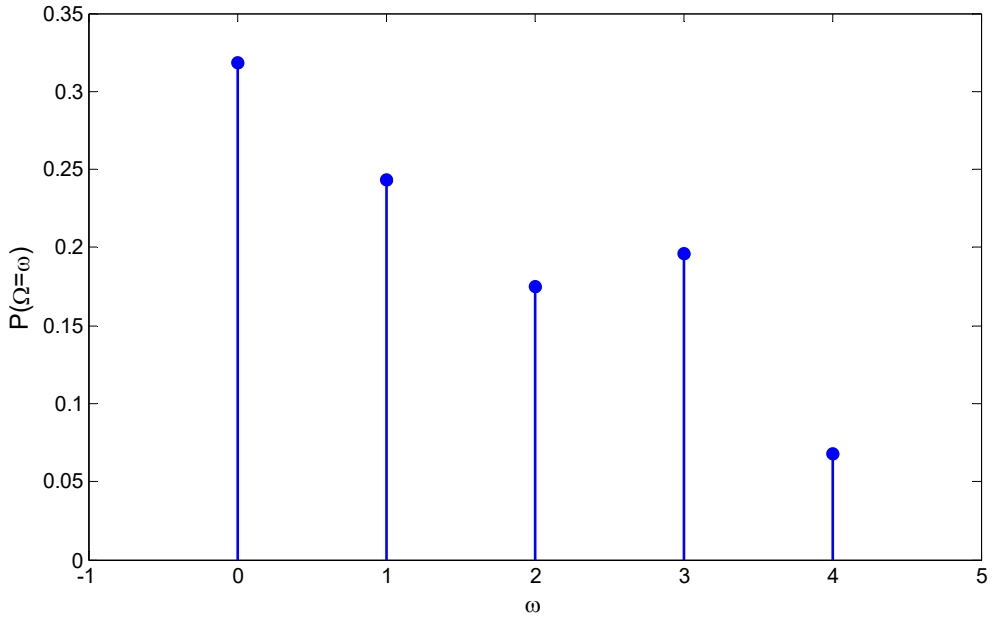


Figure 5-4. PMF of angular *intra* mode index difference.

By plugging the transition probabilities calculated for these three situations into Equation (5-16), PMF of *intra* modes of the current MB is estimated. As the *a priori* probabilities are calculated adaptively in this method, the proposed decoder is called MAP-Ad. This estimation might be unreliable, degrading performance of the decoder. In the next section, a method is provided to moderate these unreliable conditions.

5.5. Adjusting the proposed method by local entropies

The PMF estimated from the neighbors in Equation (5-16), *i.e.,* $\hat{P}(mode(b) = m)$, is used as *a priori* information in the MAP decoding metric. However, this estimate may be unreliable, resulting in a rise in error rates. In this section, a method is proposed to categorize MBs into two types, namely reliable MBs and unreliable MBs. The *intra* modes

of each MB is a source of information whose output is a discrete random variable with probabilities calculated in Section 5.4. The reliability categorization is done using the entropy of this information source [7].

In order to find the relation between the entropy of *intra* modes and performance of the proposed method, two typical video streams are analyzed frame by frame and the entropy for every frame is calculated. The PMF of *intra* modes for a frame is calculated as

$$\hat{P}(mode(b) = m | b \in Frame_x) = \frac{hist(m, Frame_x)}{\sum_{\tilde{m}} hist(\tilde{m}, Frame_x)} \quad , \tag{5-23}$$

where b a randomly selected *intra* block in frame $Frame_x$. The average *intra* mode entropy of a frame is calculated by using this probability as

$$H_{Frame_x} = -\sum_{\tilde{m}} \hat{P}(mode(b) = \tilde{m} | b \in Frame_x) \times \log\left(\hat{P}(mode(b) = \tilde{m} | b \in Frame_x)\right). \tag{5-24}$$

Temporally adjacent frames are very similar to each other in standard raw videos. In order to study performance of the decoding algorithm in frames with various textures, the frame rate is set to 3 frames per second (FPS). This is done by temporal down sampling of the standard videos. Every frame is *intra* coded as one slice. The AWGN channel SNR is set to 5.2080 dB.

In Figure 5-5 and Figure 5-6, the *intra* mode entropy of every frame is plotted. Also plotted are the PER of the proposed MAP-Ad method and the ML method as a function of frame number for two typical videos *foreman* and *football.* The MAP-Ad decoder performs better than ML, but not in all frames. It can be seen that there is an inverse relation between PER improvement and entropy. In other words, PER of MAP-Ad is higher than PER of ML for the frames with higher entropy. In such frames the adaptive PMF estimation in MAP-Ad method misleads the decoder. This is more obvious in the 9^{th}, 10^{th} and 11^{th} frames of video *football*, where the *intra* block modes are highly correlated as their spatial texture is very smooth as depicted in Figure 2-8. When the entropy of an information source is high, its outcomes are mostly equiprobable and in turn using a non-uniform PMF as *a priori* probabilities for MAP decoding mislead the decoder. Thus, the entropy can be used as a measure for reliability categorization.

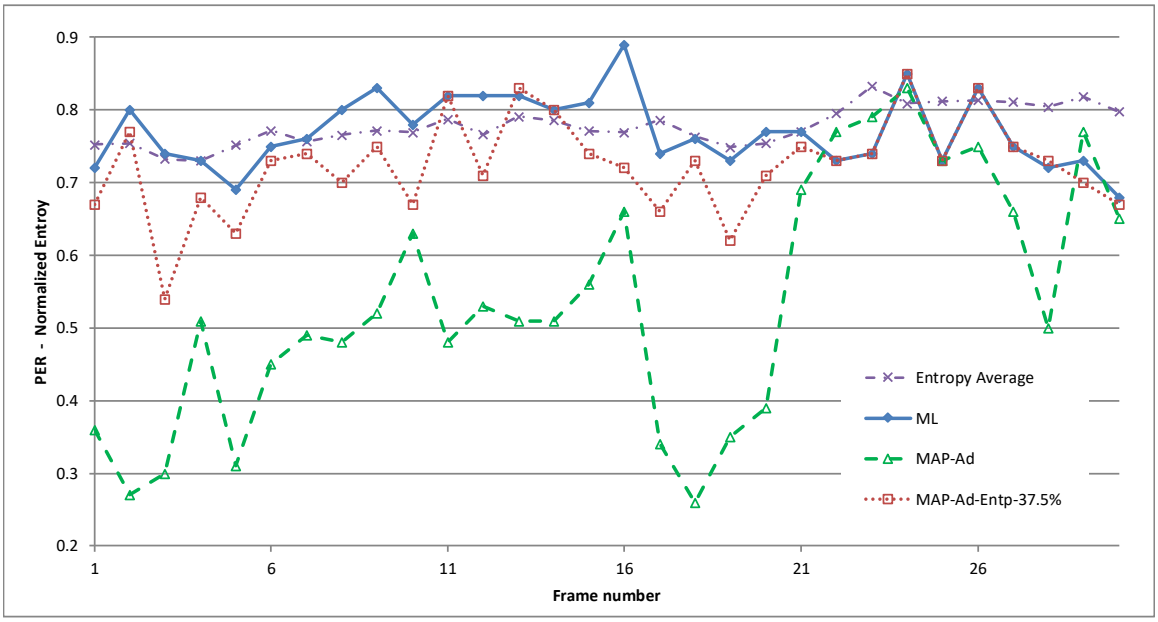


Figure 5-5. PER and normalized entropy versus frame number for ML algorithm and MAP-Ad methods for video *foreman*.

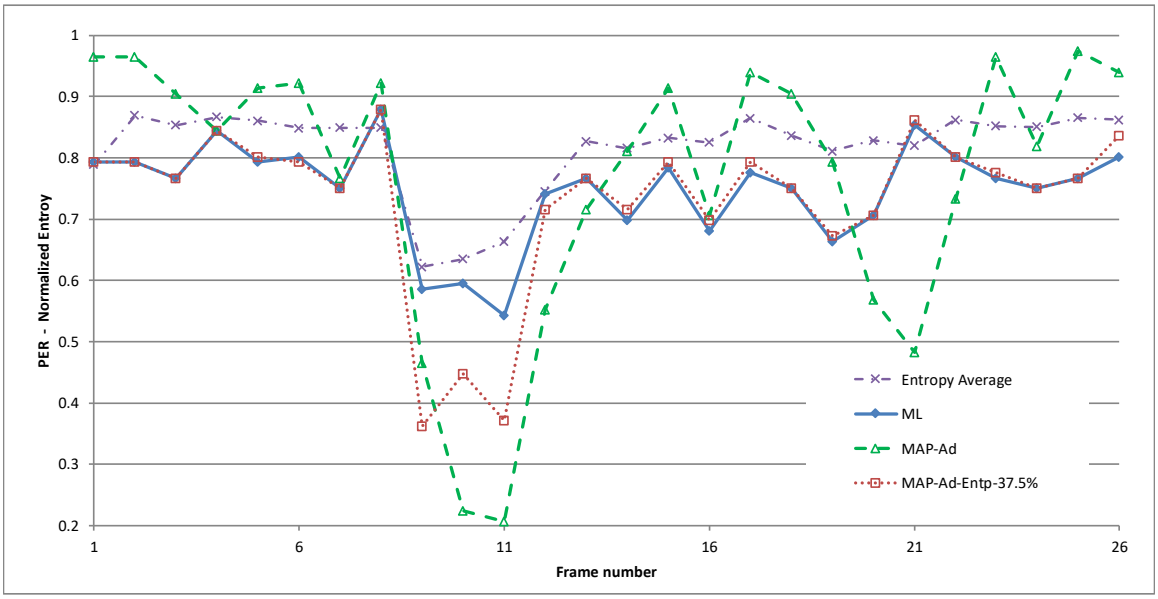


Figure 5-6. PER and normalized entropy versus frame number for ML algorithm and MAP-Ad methods for video *football.*

Here, the *intra* mode entropy is used to adjust the MAP-Ad decoding method. The proposed decoder switches between the MAP-Ad and ML based on the local *intra* mode entropies. This method is called MAP-Ad-Entp. In this method every MB is categorized as either reliable or unreliable based on its local entropy defied as

$$H_{MB_x} = -\sum_{\tilde{m}} \hat{P}(mode(b) = \tilde{m}) \times \log\left(\hat{P}(mode(b) = \tilde{m})\right) \; : \; b \in MB_x, \qquad (5\text{-}25)$$

which $\hat{P}(mode(b) = \tilde{m})$ is calculated for MB_x in Equation (5-16). The normalized local entropy is calculated as

$$\breve{H}_{MB_x} = \frac{H_{MB_x} - H_{min}}{H_{max} - H_{min}}, \tag{5-26}$$

where H_{max} and H_{min} are maximum and minimum values of the local entropy. H_{max} corresponds to the uniform PMF of *intra* modes and equals to $H_{max} = -\log\left(\frac{1}{9}\right) = 0.9542$, and H_{min} corresponds to the most unbalanced estimated PMF. For example, if 3 adjacent MBs contain 8×8 *intra* blocks and considering histogram Equation (5-9), minimum entropy is $H_{min} = -8 \times \frac{1}{39} \times \log\left(\frac{1}{39}\right) - \frac{38}{39} \times \log\left(\frac{38}{39}\right) = 0.7768$. Consequently, the normalized local entropy, $\breve{H}_{MB_x}$, would be between 0 and 1.

The block diagram of the proposed method (MAP-Ad-Ent) for *intra* mode PMF estimation is depicted in Figure 5-7. The *intra* mode histogram and *intra* mode PMF of the threeadjacent decoded MBs, *i.e.,* MB_U, MB_L and MB_{UL}, is calculated using Equation (5-11). The local entropy of a MB is calculated based on the calculated *intra* mode PMF of its adjacent MBs. If it is greater than a predefined threshold, $th\%$, the current MB is deemed unreliable and the decoder switches to ML *i.e.,* it is assumed that the *intra* modes are equiprobable. If the normalized local entropy is not greater than the threshold, the PMF estimation is corrected using the transition probability as describe in Equation (5-16). By using this strategy, the MAP decoder avoids employing unreliable information.

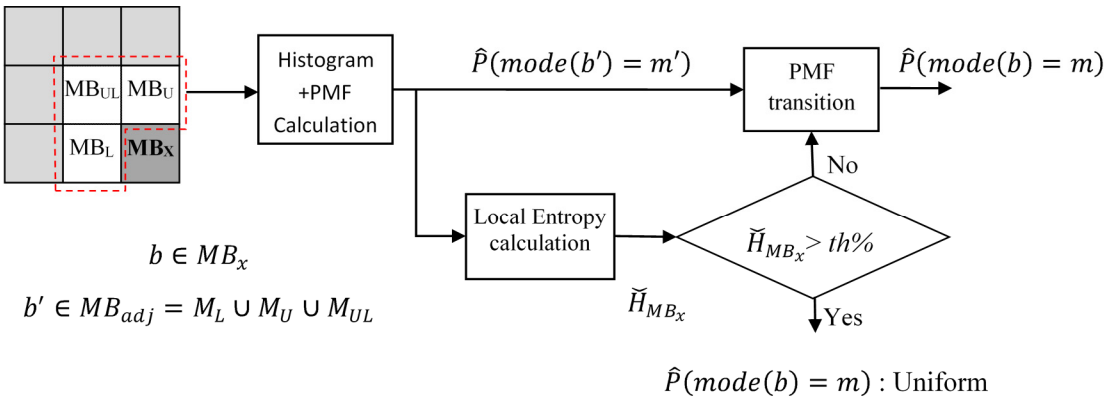


Figure 5-7. The block diagram of the *intra* mode PMF estimation in MAP-Ad-Entp method with normalized entropy threshold *th%*.

5.6. Simulation results

In the simulations 300 frames of the videos *foreman*, *football*, *car-phone, table-tennis*, and *crew* [56] in QCIF format, 30 FPS, are used. QCIF is chosen because it is a low-resolution video format and consequently the interdependency among intra blocks is low. In other words, QCIF is the one of the worst cases in terms of intra mode inter dependences. Therefore, it is expected that the results can be extended for higher resolution formats. All frames are considered as one slice and *intra* encoded with the QP value 28. The tests are done for FS probability 1×10^{-3} and EOPS probability 1×10^{-2}. The SNR of the channel is in the range 7.335 dB, 6.7895 dB, 5.208 dB, 4.3232 dB and 1.312 dB, which correspond to the probability of error $p = 5 \times 10^{-4}$, 10^{-3}, 5×10^{-3}, 10^{-2} and 5×10^{-2}, respectively. The value of M for MA is set to 16 for all simulations. The simulations run until at least 60 erroneous packets are decoded.

In Figure 5-5 and Figure 5-6, PER of MAP-Ad-Entp decoder at $th\% = 37.5\%$ is compared to MAP-Ad and ML decoder for two videos *foreman* and *football* for channel SNR 5.2080 dB. It can be seen that the PER of MAP-Ad-Entp is lower than or equal to PER of ML algorithm for most frames, while the PER of MAP-Ad algorithm is either lower or higher than PER of ML algorithm. It can be seen that the performance of the MAP-Ad algorithm is dependent on the intra mode entropy of the frames. The MAP-Ad decoder improves the PER over the ML decoder in the frames with low entropy while it degrades the frames with high entropy.

In order to measure the consistency of the PER improvement of a MAP decoder over the ML decoder, the ratio between the number of frames whose PER is greater than the PER of the ML decoder, and the total number of frames is calculated as

$$corruption\ ratio = \frac{|\{\forall fr \in \mathcal{F} |\ PER_{MAP}(fr) > PER_{ML}(fr)\}|}{|\mathcal{F}|}, \tag{5-27}$$

where, $\mathcal{F}$ is set of encoded frames. $PER_{MAP}(fr)$ and $PER_{ML}(fr)$ are, respectively, PER of the MAP decoder and PER of the ML decoder for frame $fr \in \mathcal{F}$. Cardinality of a set is show by $|.|$ operator.

To measure the outperformance of a MAP decoder in average, *PER improvement* is calculated as

$$PER\ improvement\ = \frac{PER_{ML} - PER_{MAP}}{PER_{ML}}, \tag{5-28}$$

where PER_{MAP} and PER_{ML} are PER of the MAP and ML decoders. *Corruption ratio* and *PER improvement* are measured for different videos and represented in Table 5.2.

There is a trade-off between *corruption ratio* and *PER improvement* that can be controlled by the value of entropy threshold. By decreasing the threshold, the corruption ratio gets better, although performance of the system gets closer to ML algorithm in average. The threshold 37.5% is chosen for other simulations, because in this threshold the proposed method not only outperforms the PER for all videos, but also it has a reasonable corruption ratio comparing to the MAP-Ad method. In this threshold, PER of less than %8 of frames are increased and in average PER is decreased about 13%. Depending on decoding design criteria, this threshold can be changed.

Table 5.2. *Corruption ratio* and *improvement percentag*e of MAP-Ad and MAP-Ad-Entp-th% with respect to ML algorithm in various values of the entropy thresholds for five different videos

Corruption Ratio % / PER Improvement %		Method							
		MAP-Ad-Entp-12.5%	MAP-Ad-Entp-25%	MAP-Ad-Entp-37.5%	MAP-Ad-Entp-50%	MAP-Ad-Entp-62.5%	MAP-Ad-Entp-75%	MAP-Ad-Entp-87.5%	MAP-Ad
video	Foreman	0.00% / 0.95%	6.67% / 3.25%	6.67% / 6.02%	20.00% / 9.66%	13.33% / 17.24%	13.33% / 24.08%	16.67% / 26.59%	10.00% / 30.49%
	Car-phone	0.00% / 8.33%	6.67% / 12.63%	3.33% / 16.22%	0.00% / 18.87%	0.00% / 21.75%	0.00% / 24.99%	0.00% / 26.54%	0.00% / 29.33%
	Crew	0.00% / 2.58%	3.33% / 5.07%	10.00% / 6.06%	23.33% / 7.74%	43.33% / 9.00%	36.67% / 8.73%	33.33% / 9.14%	36.67% / 8.96%
	Table-tennis	3.33% / 2.40%	20.00% / 4.33%	13.33% / 5.35%	23.33% / 5.85%	30.00% / 2.90%	60.00% / -2.40%	56.67% / -1.11%	50.00% / 0.00%
	Football	3.85% / 0.22%	7.69% / 1.24%	34.62% / 2.30%	65.38% / 1.15%	88.46% / -4.12%	80.77% / -5.76%	61.54% / -3.81%	65.38% / -2.39%
	Average	1.37% / 2.97%	8.90% / 5.42%	13.01% / 7.34%	25.34% / 8.90%	33.56% / 9.83%	36.99% / 10.55%	32.88% / 12.09%	31.51% / 13.93%

Error rates are calculated separately by simulation for each video, and for each of the methods ML, MAP and MAP-Entp-37%. The results for different SNRs are represented in Table 5.3-5.7. As expected MAP-Entp-37% has better performance than ML in all SNRs. Albeit moderate, this improvement can be enhanced by an iterative mechanism similar to [23]. It can be seen in these tables, MAP-Entp-37% has better performance for almost all videos, while MAP works well just for some particular videos, because MAP-Entp-37% avoids unreliable *a priori* information by thresholding mechanism on local entropies. Videos with high spatial activities, *e.g., football*, have more unreliable MBs. As expected in such cases, MAP method has higher error rates than ML, because *intra* mode estimation misleads the decoder, while the performance of MAP-Entp is improved or is not noticeably changed with respect to ML performance.

As expected the computational complexity of MAP-Entp-37% is lower than MAP-Ad. because MAP-Entp-37% saves some calculations of *a priori* probabilities in unreliable MBs, while MAP decoder calculates *a priori* probability of all MBs. The efficiency of MAP-Ad-Entp mechanism in terms of error rates computational complexity is borne out by these results.

Table 5.3. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 4.32dB

SNR	Video	Method	PER	PER improvement % ML	SER	BER	SEER	T (sec)
4.32 dB	Foreman	ML	9.94E-01	-	3.91E-01	1.09E-02	6.25E-01	1.25
		MAP-Ad-Entp-37.5%	9.90E-01	0.37%	3.73E-01	1.05E-02	5.93E-01	2.10
		MAP-Ad	9.54E-01	4.05%	3.16E-01	1.01E-02	4.90E-01	2.77
	Car-phone	ML	9.91E-01	-	3.88E-01	1.09E-02	5.96E-01	1.20
		MAP-Ad-Entp-37.5%	9.74E-01	1.73%	3.45E-01	1.02E-02	5.24E-01	2.04
		MAP-Ad	9.55E-01	3.65%	3.17E-01	1.04E-02	4.69E-01	2.55
	Crew	ML	9.90E-01	-	3.89E-01	1.09E-02	6.06E-01	1.21
		MAP-Ad-Entp-37.5%	9.77E-01	1.37%	3.72E-01	1.06E-02	5.77E-01	1.97
		MAP-Ad	9.63E-01	2.80%	3.80E-01	1.17E-02	5.77E-01	2.48
	Table-tennis	ML	9.90E-01	-	3.92E-01	1.12E-02	5.87E-01	1.11
		MAP-Ad-Entp-37.5%	9.85E-01	0.51%	3.89E-01	1.12E-02	5.82E-01	1.87
		MAP-Ad	9.91E-01	-0.11%	4.01E-01	1.28E-02	5.89E-01	2.68
	Football	ML	9.86E-01	-	3.93E-01	1.10E-02	5.93E-01	1.30
		MAP-Ad-Entp-37.5%	9.78E-01	0.81%	3.88E-01	1.10E-02	5.83E-01	2.23
		MAP-Ad	9.64E-01	2.21%	3.80E-01	1.20E-02	5.64E-01	2.88
	Average	ML	9.90E-01	-	3.91E-01	1.10E-02	6.01E-01	1.21
		MAP-Ad-Entp-37.5%	9.81E-01	0.96%	3.73E-01	1.07E-02	5.72E-01	2.04
		MAP-Ad	9.65E-01	2.52%	3.58E-01	1.14E-02	5.38E-01	2.67

Table 5.4. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 5.20 dB

SNR	Video	Method	PER	PER improvement % ML	SER	BER	SEER	T (sec)
5.20 dB	Foreman	ML	7.76E-01	-	2.60E-01	5.29E-03	4.12E-01	1.41
		MAP-Ad-Entp-37.5%	7.34E-01	5.44%	2.30E-01	4.84E-03	3.64E-01	2.36
		MAP-Ad	5.77E-01	25.66%	1.57E-01	3.82E-03	2.46E-01	2.93
	Car-phone	ML	7.41E-01	-	2.42E-01	5.17E-03	3.68E-01	1.19
		MAP-Ad-Entp-37.5%	6.15E-01	17.10%	1.69E-01	3.98E-03	2.54E-01	2.44
		MAP-Ad	5.45E-01	26.47%	1.41E-01	3.64E-03	2.07E-01	3.12
	Crew	ML	7.37E-01	-	2.49E-01	5.18E-03	3.85E-01	1.20
		MAP-Ad-Entp-37.5%	6.95E-01	5.73%	2.29E-01	4.81E-03	3.53E-01	1.97
		MAP-Ad	6.83E-01	7.39%	2.34E-01	5.21E-03	3.54E-01	2.42
	Table-tennis	ML	7.23E-01	-	2.43E-01	5.30E-03	3.63E-01	1.07
		MAP-Ad-Entp-37.5%	6.86E-01	5.16%	2.31E-01	5.08E-03	3.45E-01	1.75
		MAP-Ad	7.41E-01	-2.49%	2.67E-01	6.17E-03	3.93E-01	2.21
	Football	ML	7.45E-01	-	2.52E-01	5.30E-03	3.81E-01	1.27
		MAP-Ad-Entp-37.5%	7.37E-01	1.16%	2.51E-01	5.28E-03	3.78E-01	2.16
		MAP-Ad	7.70E-01	-3.27%	2.75E-01	6.06E-03	4.08E-01	2.78
	Average	ML	7.45E-01	-	2.49E-01	5.25E-03	3.82E-01	1.23
		MAP-Ad-Entp-37.5%	6.93E-01	6.91%	2.22E-01	4.80E-03	3.39E-01	2.14
		MAP-Ad	6.63E-01	10.95%	2.14E-01	4.97E-03	3.21E-01	2.69

Table 5.5. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 6.78dB

SNR	Video	Method	PER	PER improvement % ML	SER	BER	SEER	T (sec)
6.78 dB	Foreman	ML	2.36E-02	-	5.58E-03	9.97E-05	8.79E-03	1.20
		MAP-Ad-Entp-37.5%	2.05E-02	12.87%	4.46E-03	8.41E-05	7.06E-03	1.92
		MAP-Ad	1.23E-02	47.89%	2.41E-03	5.55E-05	3.87E-03	2.46
	Car-phone	ML	2.21E-02	-	5.26E-03	1.01E-04	7.93E-03	1.10
		MAP-Ad-Entp-37.5%	1.50E-02	31.98%	3.01E-03	6.63E-05	4.44E-03	1.98
		MAP-Ad	1.05E-02	52.25%	1.94E-03	4.75E-05	2.84E-03	2.54
	Crew	ML	2.35E-02	-	5.55E-03	1.04E-04	8.66E-03	1.23
		MAP-Ad-Entp-37.5%	2.10E-02	10.78%	4.81E-03	9.21E-05	7.49E-03	2.07
		MAP-Ad	2.41E-02	-2.33%	6.15E-03	1.09E-04	9.34E-03	2.56
	Table-tennis	ML	2.00E-02	-	5.02E-03	9.70E-05	7.52E-03	1.09
		MAP-Ad-Entp-37.5%	1.77E-02	11.91%	4.56E-03	8.74E-05	6.81E-03	1.83
		MAP-Ad	2.19E-02	-9.43%	6.95E-03	1.22E-04	1.02E-02	2.58
	Football	ML	1.94E-02	-	4.62E-03	9.14E-05	7.02E-03	1.37
		MAP-Ad-Entp-37.5%	1.98E-02	-2.06%	4.78E-03	9.41E-05	7.26E-03	2.30
		MAP-Ad	3.54E-02	-82.26%	1.11E-02	1.78E-04	1.64E-02	2.96
	Average	ML	2.17E-02	-	5.21E-03	9.88E-05	7.98E-03	1.20
		MAP-Ad-Entp-37.5%	1.88E-02	13.47%	4.33E-03	8.49E-05	6.61E-03	2.02
		MAP-Ad	2.08E-02	4.12%	5.67E-03	1.02E-04	8.52E-03	2.62

Table 5.6. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 7.34dB

SNR	Video	Method	PER	PER improvement % ML	SER	BER	SEER	T (sec)
7.34 dB	Foreman	ML	3.62E-03	-	6.51E-04	1.35E-05	1.02E-03	1.17
		MAP-Ad-Entp-37.5%	3.51E-03	3.00%	5.94E-04	1.30E-05	9.35E-04	1.90
		MAP-Ad	2.41E-03	33.50%	3.77E-04	1.02E-05	5.87E-04	2.32
	Car-phone	ML	3.41E-03	-	6.20E-04	1.45E-05	9.29E-04	1.24
		MAP-Ad-Entp-37.5%	2.70E-03	20.74%	3.97E-04	1.18E-05	5.87E-04	2.15
		MAP-Ad	1.97E-03	42.02%	3.30E-04	1.06E-05	4.87E-04	2.59
	Crew	ML	3.39E-03	-	6.76E-04	1.47E-05	1.03E-03	1.29
		MAP-Ad-Entp-37.5%	3.13E-03	7.49%	5.95E-04	1.35E-05	9.02E-04	2.19
		MAP-Ad	3.39E-03	0.00%	6.79E-04	1.53E-05	1.03E-03	2.89
	Table-tennis	ML	3.04E-03	-	5.39E-04	1.41E-05	8.12E-04	1.14
		MAP-Ad-Entp-37.5%	2.75E-03	9.52%	4.76E-04	1.32E-05	7.17E-04	1.97
		MAP-Ad	2.92E-03	4.17%	7.05E-04	1.62E-05	1.04E-03	2.49
	Football	ML	2.94E-03	-	5.33E-04	1.37E-05	8.12E-04	1.29
		MAP-Ad-Entp-37.5%	2.94E-03	0.00%	5.41E-04	1.37E-05	8.24E-04	2.13
		MAP-Ad	4.51E-03	-53.37%	1.27E-03	2.29E-05	1.89E-03	2.68
	Average	ML	3.28E-03	-	6.05E-04	1.41E-05	9.21E-04	1.23
		MAP-Ad-Entp-37.5%	3.01E-03	8.28%	5.22E-04	1.31E-05	7.93E-04	2.07
		MAP-Ad	3.04E-03	7.28%	6.70E-04	1.50E-05	1.01E-03	2.59

Table 5.7. PER, SER, BER, SEER, average running time per packet and PER improvement with respect to ML for the MAP, MAP-Ad-Entp-37.5% and ML decoders, and different videos in channel SNR= 8.39dB

SNR	Video	Method	PER	PER improvement % ML	SER	BER	SEER	T (sec)
8.39 dB	Foreman	ML	3.65E-04	-	3.97E-05	1.13E-06	6.00E-05	1.18
		MAP-Ad-Entp-37.5%	3.55E-04	2.74%	3.72E-05	1.12E-06	5.61E-05	2.06
		MAP-Ad	2.20E-04	39.73%	1.97E-05	9.64E-07	2.78E-05	2.52
	Car-phone	ML	3.15E-04	-	3.07E-05	1.36E-06	4.50E-05	1.04
		MAP-Ad-Entp-37.5%	2.95E-04	6.35%	2.52E-05	1.31E-06	3.65E-05	1.89
		MAP-Ad	1.79E-04	43.00%	1.89E-05	1.24E-06	2.79E-05	2.20
	Crew	ML	2.15E-04	-	1.96E-05	1.34E-06	3.14E-05	1.19
		MAP-Ad-Entp-37.5%	2.20E-04	-2.33%	1.99E-05	1.34E-06	3.21E-05	2.00
		MAP-Ad	2.30E-04	-6.98%	2.32E-05	1.37E-06	3.82E-05	2.53
	Table-tennis	ML	2.45E-04	-	1.78E-05	1.19E-06	2.73E-05	1.05
		MAP-Ad-Entp-37.5%	2.05E-04	16.33%	1.33E-05	1.16E-06	2.02E-05	1.80
		MAP-Ad	1.48E-04	39.37%	9.19E-06	1.16E-06	1.34E-05	2.29
	Football	ML	2.10E-04	-	2.33E-05	1.29E-06	3.70E-05	1.14
		MAP-Ad-Entp-37.5%	2.10E-04	0.00%	2.35E-05	1.29E-06	3.71E-05	1.86
		MAP-Ad	1.22E-04	42.03%	1.39E-05	1.15E-06	2.17E-05	2.17
	Average	ML	2.58E-04	-	2.27E-05	1.30E-06	3.46E-05	1.09
		MAP-Ad-Entp-37.5%	2.40E-04	7.10%	1.95E-05	1.27E-06	2.96E-05	1.90
		MAP-Ad	1.82E-04	29.52%	1.65E-05	1.25E-06	2.52E-05	2.33

5.7. Summary

In this chapter, a new adaptive method (MAP-Ad-Entp) has been proposed for MAP arithmetic decoding to exploit the residual redundancy in H.264 video streams. In this method, *a priori* probabilities of *intra* modes are estimated adaptively using the syntax elements decoded earlier. The decoder categorizes every MB to reliable and unreliable based on the entropy of estimated statistics. If the MB is unreliable all *intra* modes are considered to be equiprobable, otherwise the estimated *a priori* probabilities are used in the decoding metric. Unlike previous work [44], the estimated PMF from the spatially adjacent syntax elements is corrected using a transition probability function between every

MB and its neighbors. The simulation results have shown improvement in terms of error rates.

CHAPTER 6. Design of the quantizer on the channel output for joint source channel arithmetic decoding

6.1. Introduction

In high speed telecommunication architectures based on digital signal processing, quantizing the data received from a channel is inevitable. A channel quantizer discretizes the continuous values received from the physical channel using decision levels, and then maps them to discrete values called reconstructions levels which are represented by bits [45]. In high speed applications, low precision quantizers are used due to limitations in analog to digital convertors and to reduce the processing power of the decoder. For example, no more than three bits of precision are used in NAND flash memory, due to read latency [69], [70]. Also, in high speed optical telecommunication (32 GSamples/second) only two bits of precision are used to quantize data [71]. However, limiting the quantization precision causes error resilience to be degraded. Therefore, designing an optimum channel quantizers is important to achieve the lowest error rates. In the channel quantizer design, the performance of the successive modules including channel decoder and source decoder should be considered.

In joint source channel arithmetic coding (JSCAC), the decoder uses the quantized channel data for both decompression and for error correction. The objective of channel quantizer optimization is to find quantization parameters to minimize the error rates.

To optimize the channel quantizer, the calculation of the error rates for various parameters of quantization and JSCAC is impractical. To address this problem, a figure of merit as a proxy for quantizer performance is needed. Two main approaches have been used in the past for the quantization design. One of them is minimum mean square error (MMSE) based approach [46] and the other is an information theoretic approach [47]-[50], [72] and [73]. The former uses mean square error and the later uses information theoretic criteria including channel cutoff rate, channel capacity and mutual information [7] as a proxy for evaluating the performance of the channel quantizer.

MMSE design introduced by Lloyd [74] and Max [75] is the most well-known approach for designing quantizers [46]. In this method, for a given quantizer input PDF, the decision and reconstruction levels are optimized to minimize the mean square error (MSE) between the quantizer input and output, but the effect of error correction codes is not considered in the design.

In [49] and [50], the quantizer is designed to maximize the channel cutoff rate which is the upper limit of code rates over which the average computational operations per bit is finite if sequential decoding [38] is used. A particular probability distribution of quantizer input results in the channel cutoff rate. Therefore, the input of the quantizer does not respect this condition would not achieves this rate, and in turn this might not be a proper criterion of goodness.

In [48], the capacity of an AWGN quantized channel is calculated under an average power constraint. The optimum input distribution to achieve the capacity is calculated. However, this approach is not applicable when there is no constraint on the quantizer input

distribution or power. In [47], a 4-level symmetric quantizer is optimized to maximize the mutual information (MMI) between its output and input. The performance of this quantizer with a low-density parity check (LDPC) coding is evaluated. The LDPC code is in a family of capacity-achieving codes whose coding rate is equal to the channel capacity. In other words, the output distribution of these codes achieves the maximum value of the mutual information between input and output of the channel. However, mutual information might not be a proper figure of merit for the coding algorithms whose rate is considerably less than the channel capacity.

In this chapter, new proxy measures for performance of the channel quantizer are proposed. In calculation of these measures, the decoding tree is developed using the suboptimum breadth first search algorithm, MA, as explained below.

The output of the channel quantizer is a sequence of symbols fed to a joint source channel arithmetic decoder. The decoder uses a decoding tree to generate candidates of the channel bit sequence. In each stage of the decoding tree, the *a posteriori* probability of each candidate is calculated as the decoding metric. Afterward, the candidates are sorted based on their decoding metric and only the M best candidates are saved for the next stage and the others are eliminated. The probability of eliminating the true candidate, *i.e.,* the bit sequence that was the input to the channel, is directly related to its rank among the sorted candidates. The rank of true candidate (RTC) is a random variable whose statistics are used to define two proxy measures.

The first measure is dispersion of the true candidate (DTC). This is defined as the second statistical moment [39] of RTC. In other words, DTC measures how far the true

candidate is from the top of the ranked list of candidates. The higher the DTC, the more probable the event that the true candidate is not in the top M candidates and it is dropped from the list. However, to calculate DTC, the probability mass function (PMF) of RTC must be calculated completely which is a computationally complex procedure especially for long candidates.

The second measure is the probability that the true candidate is among the best two candidates, *i.e.,* probability that RTC is less than or equal to 1 (PRTC1). It is shown that the probability that the true candidate is the best (RTC=0) is independent from the quantization parameters. Therefore, to optimize PRTC1 value of PMF of RTC at only one point (RTC=1) is required, unlike DTC calculation in which PMF of RTC is required in its entire domain. Afterward, PRTC1 is calculated and approximated for various channel SNRs. Quantization parameters are chosen to maximize PRTC1.

Simulations are carried out for different channel noise levels and various FS probabilities. Simulation results show the effectiveness of these proposed measures, *i.e.,* DTC and PRTC1 for JSCAC using the breadth first decoder comparing to the two other approaches, *i.e.,* MMSE and MMI.

This chapter is organized as follows. In Section 6.2, the channel quantization problem is stated followed by explanation on two available methods for optimizing the channel quantizer, *i.e.,* MMSE and MMI. Section 6.3 explains the proposed figure of merits for quantizer optimization based on the statistics of RTC. Simulation results are provided in Section 6.4. Finally, the chapter is summarized in Section 6.5.

6.2. Available channel quantization designs

In order to moderate the complexity of the arithmetic decoder a quantizer is used to quantize the soft information coming from the channel. The block diagram of the transmission system using the quantized channel is depicted in Figure 6-1. Similar to Section 3.2, the binarizer maps the sequence of syntax elements $\boldsymbol{x} = [x_1, \dots, x_{\ell_x}]$ to bin sequence $\boldsymbol{s} = [s_1, \dots, s_{\ell_s}]$, where s_i is either 0 or 1. The binary arithmetic coder (BAC) with FS compresses the bin sequence to a bit sequence, $\boldsymbol{u} = [u_1, \dots, u_{\ell_u}]$ of length ℓ_u, where again u_i is either 0 or 1. The BPSK modulator maps $u_i = 0$ and $u_i = 1$ respectively to $v_i = -\sqrt{E_b}$ and $v_i = +\sqrt{E_b}$, where E_b is the energy per bit of the transmitted signal through a communication channel modeled by the addition of white Gaussian noise (AWGN), $\boldsymbol{n}$. Throughout this chapter, E_b remains fixed to 1 and the energy of the noise is changed to have various channel SNRs. Using a demodulator and quantizer the received vector from the channel $\boldsymbol{y}$ is mapped to $\boldsymbol{z}$ whose element are the quantization levels as shown in Figure 6-2, *i.e.,* $r_1, \dots, r_4$. If the number of the reconstruction levels is 2 (1-bit precision), the decoding turns to the hard input decoder (see Chapter 3). Here the 2-bit quantizer (see Figure 6-2) is discussed. It is shown later in this Section that the 2-bit quantizer design is a uni-parametric optimization problem for JSCAC.

Since the U_is are the output of the entropy coder (BAC), its output bits are assumed to be equally likely [67], *i.e.,*

$$\mathrm{P}(U_i = 0) = \mathrm{P}(U_i = 1) = \frac{1}{2}. \tag{6-1}$$

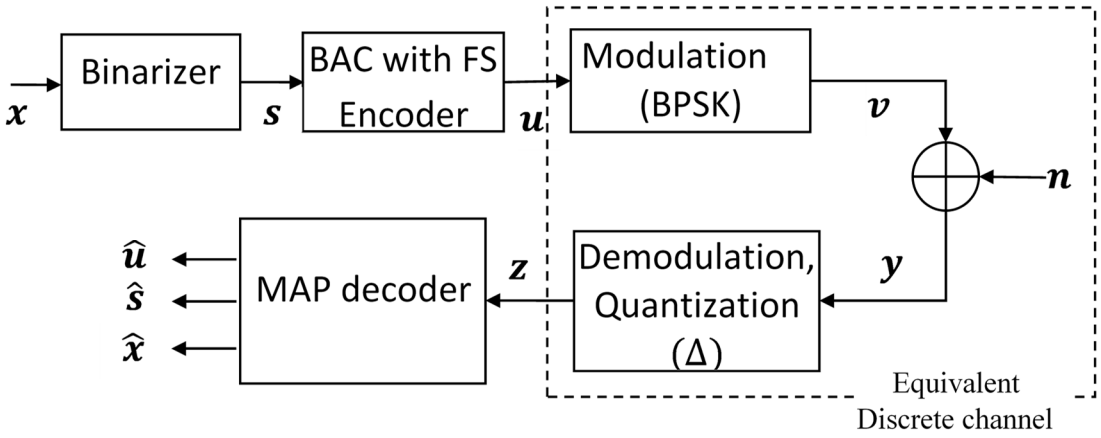


Figure 6-1. Block diagram of the transmission system using the quantized channel.

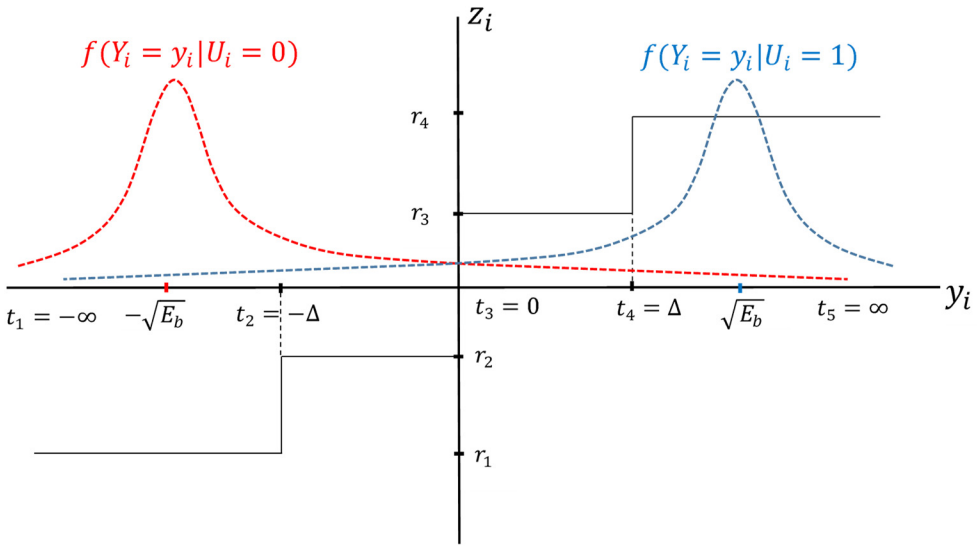


Figure 6-2. Quantizer function with 4 decision levels, and conditional PDF of channel output given the sent bit for BPSK modulation.

6.2.1. MMSE method

The first approach for optimizing the quantizer is the MMSE approach also known as the Lloyd-Max method [46]. In this approach, the optimization measure is the MSE between y_i and z_i as

$$\mathcal{E} = E[(Y_i - Z_i)^2] = \int_{-\infty}^{+\infty} E[(Y_i - Z_i)^2 | Y_i = y_i].f(Y_i = y_i)\, dy_i$$

$$= \int_{-\infty}^{+\infty} (y_i - z_i)^2.f(Y_i = y_i)\, dy_i. \tag{6-2}$$

where $f(Y_i = y_i)$ is PDF of the channel output and z_i is quantized version of y_i (see Figure 6-2). Using the assumption provided in Equation (6-1), $f(Y_i = y_i)$ is calculated as

$$f(Y_i = y_i) = \sum_{\tilde{u}_i = 0,1} f(Y_i = y_i | U_i = \tilde{u}_i) P(U_i = \tilde{u}_i) =$$

$$= \frac{1}{2}[f(Y_i = y_i | U_i = 0) + f(Y_i = y_i | U_i = 1)], \tag{6-3}$$

where

$$f(Y_i = y_i | U_i = \tilde{u}_i) = \begin{cases} \dfrac{1}{\sqrt{2\sigma^2\pi}} \exp\left(-\dfrac{\left(y_i + \sqrt{E_b}\right)^2}{2\sigma^2} \right) & : \quad \tilde{u}_i = 0 \\ \dfrac{1}{\sqrt{2\sigma^2\pi}} \exp\left(-\dfrac{\left(y_i - \sqrt{E_b}\right)^2}{2\sigma^2} \right) & : \quad \tilde{u}_i = 1. \end{cases} \tag{6-4}$$

here σ^2 is the variance of the noise. These two conditional normal PDFs are shown in Figure 6-2. For a 4-level quantizer, Equation (6-2) can be written as

$$\mathcal{E} = \sum_{k=1}^{4} \int_{t_k}^{t_{k+1}} (y_i - r_k)^2 . f(Y_i = y_i)\, dy_i \,, \tag{6-5}$$

where t_k and r_k are decision and reconstruction levels, respectively as shown Figure 6-2. It can be seen in Equations (6-3) and (6-4) that $f(Y_i = y_i)$ is an even function. Therefore, the optimum quantizer decision and reconstruction levels are symmetric. In other words, the optimum decision levels and reconstruction levels of the quantizer are $t_5 = -t_1 = \infty$, $t_4 = -t_2 = \Delta$, $t_3 = 0$, $r_4 = -r_1$ and $r_3 = -r_2$. By taking the partial derivative of $\mathcal{E}$ with respect to t_k and r_k and equating it to zero, the optimum decision levels and reconstruction levels respectively are

$$t_k = \frac{r_k + r_{k+1}}{2}, \tag{6-6}$$

and

$$r_k = \frac{\int_{t_k}^{t_{k+1}} y_i\, f(Y_i = y_i)\, dy_i}{\int_{t_k}^{t_{k+1}} f(Y_i = y_i)\, dy_i} \,, \tag{6-7}$$

as expected [46]. Combining these two equations and using the symmetric property of the quantizer leads to

$$\Delta = \frac{\int_0^{\Delta} y_i \, f(Y_i = y_i)\, dy_i}{2\int_0^{\Delta} f(Y_i = y_i)\, dy_i} + \frac{\int_{\Delta}^{\infty} y_i \, f(Y_i = y_i)\, dy_i}{2\int_{\Delta}^{\infty} f(Y_i = y_i)\, dy_i}. \tag{6-8}$$

Noting $E_b = 1$, the denominators can be written as

$$\mathcal{F}(\Delta) \triangleq 2\int_0^{\Delta} f(Y_i = y_i)\, dy_i = Q\left(\frac{1-\Delta}{\sigma}\right) - Q\left(\frac{1+\Delta}{\sigma}\right), \tag{6-9}$$

and

$$\mathcal{G}(\Delta) \triangleq 2\int_{\Delta}^{\infty} f(Y_i = y_i)\, dy_i = 1 - \mathcal{F}(\Delta) = 1 - Q\left(\frac{1-\Delta}{\sigma}\right) + Q\left(\frac{1+\Delta}{\sigma}\right), \tag{6-10}$$

where $Q(.)$ is defined as Equation (3-14). Using integration by parts the numerators in Equation (6-8) are written as

$$\begin{aligned} \mathcal{J}(\Delta) &= \int_0^{\Delta} y_i f(Y_i = y_i)\, dy_i \\ &= \frac{\sigma}{2\sqrt{2\pi}}\left[2e^{\frac{-1}{2\sigma^2}} - e^{\frac{-(\Delta-1)^2}{2\sigma^2}} - e^{\frac{-(\Delta+1)^2}{2\sigma^2}}\right] \\ &\quad + \frac{1}{2}\left[-Q\left(\frac{1}{\sigma}\right) + Q\left(\frac{1-\Delta}{\sigma}\right) + Q\left(\frac{1+\Delta}{\sigma}\right)\right], \end{aligned} \tag{6-11}$$

and

$$\mathcal{K}(\Delta) = \int_{\Delta}^{\infty} y_i f(Y_i = y_i)\, dy_i$$

$$= \frac{\sigma}{2\sqrt{2\pi}}\left[e^{\frac{-(\Delta-1)^2}{2\sigma^2}} + e^{\frac{-(\Delta+1)^2}{2\sigma^2}}\right] + \frac{1}{2}\left[Q\left(\frac{1-\Delta}{\sigma}\right) - Q\left(\frac{1+\Delta}{\sigma}\right)\right]. \quad (6\text{-}12)$$

Therefore, Equation (6-8) turns into

$$\Delta = \frac{\mathcal{J}(\Delta)}{\mathcal{F}(\Delta)} + \frac{\mathcal{K}(\Delta)}{\mathcal{G}(\Delta)} \quad , \quad (6\text{-}13)$$

which can be solved using an iterative method. As this equation is a nonlinear equation in form of $\Delta = g(\Delta)$ and we do not have derivative of $g(\Delta)$, it is solved using *fixed-point iteration* method [76]. In this method the equation is iteratively solved using initial point $\Delta^{(0)}$ and recursive relation

$$\Delta^{(i+1)} = g\left(\Delta^{(i)}\right). \quad (6\text{-}14)$$

The recursion continues until the difference between two consecutive points is less than a predefined threshold e, *i.e.,* $|\Delta^{(i+1)} - \Delta^{(i)}| < e$. The result is shown in Figure 6-3. It can be seen that the optimum Δ is about 1 for the operational SNRs (4-8 dB).

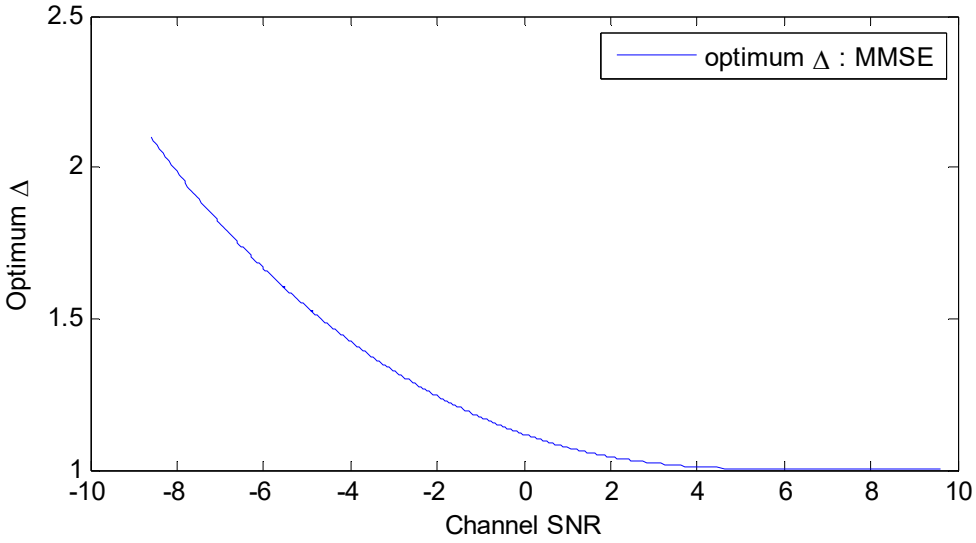


Figure 6-3. Optimum Δ versus channel SNR for MMSE quantizer.

6.2.2. MMI method

Another approach for optimum quantizer design is the maximum mutual information (MMI) method [47], [73]. In this method, the combination of modulator, AWGN channel, demodulator and quantizer is considered as an equivalent discrete memoryless channel, and the mutual information between its input and output is maximized. As the quantizer has 4 levels, this is equivalent to a binary input-quaternary output channel, as shown in Figure 6-4. The input is a bit $U_i \in \{0,1\}$ and output is a symbol $Z_i \in \{r_1, r_2, r_3, r_4\}$. The transition probabilities ρ_1, ρ_2, ρ_3 and ρ_4 are calculated as

$$\rho_1 = P(Z_i = r_1 | U_i = 0) = P(Z_i = r_4 | U_i = 1) = Q\left(\frac{\Delta - \sqrt{\mathrm{E_b}}}{\sigma}\right), \tag{6-15}$$

$$\rho_2 = P(Z_i = r_2|U_i = 0) = P(Z_i = r_3|U_i = 1) = Q\left(\frac{\sqrt{\mathrm{E_b}} - \Delta}{\sigma}\right) - Q\left(\frac{\sqrt{\mathrm{E_b}}}{\sigma}\right), \quad (6\text{-}16)$$

$$\rho_3 = P(Z_i = r_3|U_i = 0) = P(Z_i = r_2|U_i = 1) = Q\left(\frac{\sqrt{\mathrm{E_b}}}{\sigma}\right) - Q\left(\frac{\sqrt{\mathrm{E_b}} + \Delta}{\sigma}\right), \quad (6\text{-}17)$$

and

$$\rho_4 = P(Z_i = r_4|U_i = 0) = P(Z_i = r_1|U_i = 1) = Q\left(\frac{\sqrt{\mathrm{E_b}} + \Delta}{\sigma}\right), \quad (6\text{-}18)$$

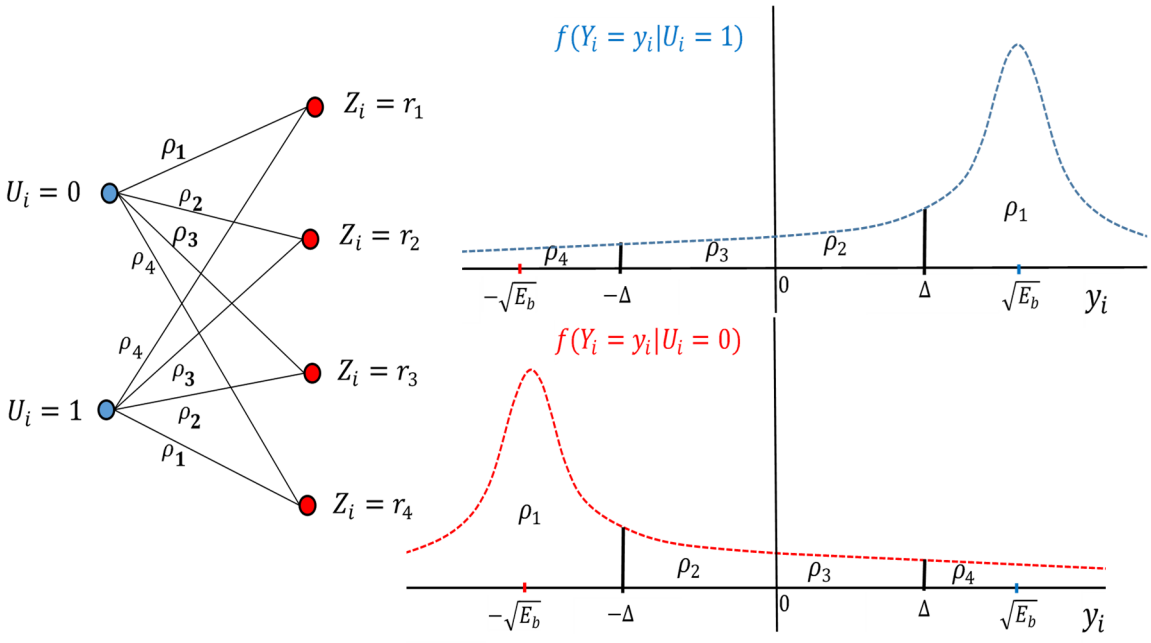


Figure 6-4. Transition of the equivalent channel of a quantized channel.

These probabilities are functions of Δ and the chanel SNR. The mutual information [7] between the input and the output of the equivalent discrete channel (U_i and Z_i) is

$$I(Z_i; U_i) = H(Z_i) - H(Z_i|U_i), \tag{6-19}$$

where $H(.)$ is the entropy function [7]. By using total probability rule, the PMF of the output symbols $P(Z_i = z_i)$ can be written as

$$P(Z_i = z_i) = \sum_{u_i=0,1} P(U_i = u_i)P(Z_i = z_i|U_i = u_i). \tag{6-20}$$

Using that the assumption in Equation (6-1), and conditional probabilities in Equations (6-15)-(6-18), $P(Z_i = z_i)$ turns into

$$P(Z_i = z_i) = \begin{cases} \frac{1}{2}(\rho_1 + \rho_4) : \ z_i \in \{r_1, r_4\} \\ \frac{1}{2}(\rho_2 + \rho_3) : \ z_i \in \{r_2, r_3\}. \end{cases} \tag{6-21}$$

Thus, the entropy $H(Z_i)$ is calculated using the definition as

$$H(Z_i) = -(\rho_1 + \rho_4)\log\left(\frac{\rho_1 + \rho_4}{2}\right) - (\rho_2 + \rho_3)\log\left(\frac{\rho_2 + \rho_3}{2}\right) \tag{6-22}$$

The definition of the conditional entropy and transition probabilities in Equations (6-15)-(6-18) imply

$$H(Z_i|U_i) = \sum_{u_i=0,1} P(U_i = u_i)H(Z_i|U_i = u_i)$$

$$= -\sum_{u_i=0,1} P(U_i = u_i)\sum_{k=1}^{4} \rho_k \log(\rho_k) = -\sum_{k=1}^{4} \rho_k \log(\rho_k), \qquad (6\text{-}23)$$

By plugging Equations (6-22) and (6-23) into (6-19), the mutual information is written as

$$I(Z_i; U_i) = \sum_{k=1}^{4} \rho_k \log(\rho_k) - (\rho_1 + \rho_4)\log\left(\frac{\rho_1 + \rho_4}{2}\right) - (\rho_2 + \rho_3)\log\left(\frac{\rho_2 + \rho_3}{2}\right), \quad (6\text{-}24)$$

which ρ_ks are the transition probabilities shown in Figure 6-4 and calculated in Equations (6-15)-(6-18) [69]. $I(Z_i; U_i)$ is only a function of the ρ_ks whose values depend on Δ and the channel SNR. The optimum Δ for the MMI quantizer is calculated using a brute-force method with Δ step size of 0.001. In Figure 6-5, the optimum Δ for MMI quantizer is plotted as a function of SNR. It can be seen that the optimum Δ varies from 0.2 to 0.4 in the operational SNRs which is between 4dB and 8dB.

Comparing Figure 6-4 and Figure 6-5, the optimum Δ obtained in MMI method is much smaller than the optimum Δ obtained in MMSE method.

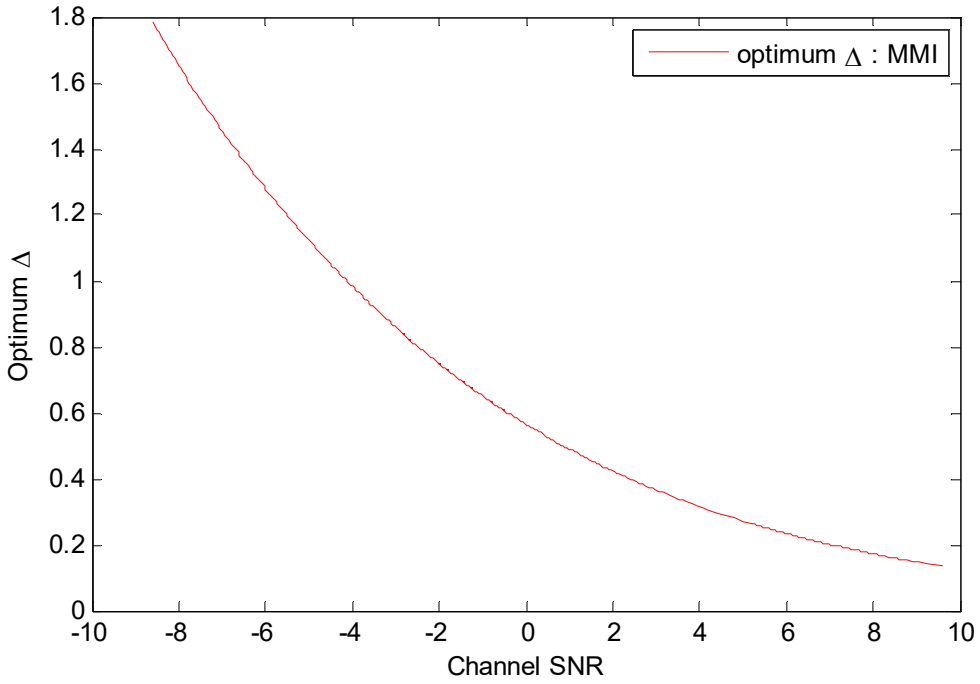


Figure 6-5. Optimum Δ versus channel SNR for MMI quantizer.

6.3. Channel quantizer designs based on rank of true candidate

Neither MMI method nor MMSE method described in Section 6.2 consider influence of different values of Δ on efficiency of the decoding method [77]. In this section, two new proxy measures are proposed to calculate the effect of Δ on the rank of true candidate among sorted candidates in the decoding tree.

Due to the large number of candidates in the decoding process, keeping all of them is not feasible. As explained in Section 3.2, the breadth-first algorithm MA is used to moderate this complexity by pruning the decoding tree. This is done by sorting the candidates in each stage, keeping the best M candidates and then eliminating the others. The sort and elimination processes are based on the decoding metric of candidates provided

later in Equation (6-25). As explained in Section 6.2, the value of Δ determines the transition probabilities of the equivalent discrete channel (see Equations (6-15)-(6-18)), and these probabilities are part of the decoding metric. Lower the RTC, lower the probability of elimination of the true candidate (the candidate corresponds to the input of the channel). In this section, PMF of RTC is calculated for various values of Δ.

As explained in Section 6.2, the channel followed by a quantizer is equivalent to a discrete memoryless channel whose input and output are bit sequence $\boldsymbol{u} = [u_1, \dots, u_{\ell_u}]$ and symbol sequence $\boldsymbol{z} = [z_1, \dots, z_{\ell_u}]$ respectively (see Figure 6-1), where $u_i \in \{0,1\}$ and $z_i \in \{r_1, r_2, r_3, r_4\}$. $\boldsymbol{z}$ is the input to the MAP decoder. By applying logarithm of the decoding metric in Equation (3-2), the decoding metric at stage j assigned to each candidate $\tilde{\boldsymbol{u}}^j$ with length of j channel bits is

$$m_{\tilde{\boldsymbol{u}}^j} = \log\left(P\left(\boldsymbol{z}^j \middle| \tilde{\boldsymbol{u}}^j\right)\right) = \sum_{k=1}^{j} \log\left(P\left(z_k \middle| \tilde{u}_k^j\right)\right). \tag{6-25}$$

where $\boldsymbol{z}^j$ and $\tilde{\boldsymbol{u}}^j$ are the first j symbols of $\boldsymbol{z}$ and $\tilde{\boldsymbol{u}}$ respectively. To derive this equation, it is assumed that U_is are independent and equiprobable (see Equation (6-1)) and the additive noise is white.

If the decoder keeps all candidates between stage 1 and j, at stage j there are 2^j candidates which are sorted based on their metric. Only one of these candidates is the same as the bit sequence sent, *i.e.,* the true candidate. The rank of true candidate (RTC) among the sorted candidates is a random variable whose range is between 0 and $2^j - 1$.

To calculate the PMF of the RTC, the elimination process is assumed to be inactive up to stage j. Since there are 4 levels of quantization, the number of possibilities for the received symbol sequence $\boldsymbol{z}^j$ is 4^j. The equivalent channel is symmetric as represented in Figure 6-4, and all $\widetilde{\boldsymbol{u}}^j$s are equiprobable (see Equation (6-1)). Thus, the PMF of RTC given that the full zero sequence ($\widetilde{\boldsymbol{u}}^j = \mathbf{0} = [0, \dots, 0]$) is transmitted is identical to the PMF of RTC given that any other bit sequence is transmitted. Therefore, PMF of RTC at stage j is written as

$$\begin{aligned} \mathrm{P}(RTC = \eta) &= \mathrm{P}(RTC = \eta|, \boldsymbol{U} = \mathbf{0}) \\ &= \sum_{\tilde{\boldsymbol{z}}^j \in \mathcal{D}^j} P(RTC = \eta|\boldsymbol{Z} = \tilde{\boldsymbol{z}}^j, \boldsymbol{U} = \mathbf{0}) \times P\left(\boldsymbol{Z} = \tilde{\boldsymbol{z}}^j|\boldsymbol{U} = \mathbf{0}\right) \end{aligned} \tag{6-26}$$

where $\mathcal{D}^j$ is set of all possible symbol sequences $\boldsymbol{z}^j$ (see Figure 6-1). Using that the channel is memoryless, the second term is written as

$$P\left(\boldsymbol{Z} = \tilde{\boldsymbol{z}}^j|\boldsymbol{U} = \mathbf{0}\right) = \prod_{i=1}^{j} P\left(Z_i = \tilde{z}_i^j|U_i = 0\right) \tag{6-27}$$

where $P\left(Z_i = \tilde{z}_i^j|U_i = 0\right)$ is the transition probability calculated in Equations (6-15)-(6-18). To calculate the first term, $P(RTC = \eta|\boldsymbol{Z} = \tilde{\boldsymbol{z}}^j, \boldsymbol{U} = \mathbf{0})$, the metric of all 2^j candidates need to be calculated based on Equation (6-25). The position of the true candidate (full zero bit sequence $\widetilde{\boldsymbol{u}} = \mathbf{0}$) among the sorted candidates represents RTC. If

there is a tie between the metric of the true candidate and the metric of other candidates which their rank is between η_l and η_h, this probability is written as

$$P\left(RTC = \eta \middle| \boldsymbol{Z} = \tilde{\boldsymbol{z}}^j, \boldsymbol{U} = \boldsymbol{0}\right) = \begin{cases} \frac{1}{\eta_h - \eta_l + 1} & : \quad \eta_l \leq \eta \leq \eta_h \\ 0 & : \quad Otherwise. \end{cases} \tag{6-28}$$

6.3.1. Dispersion of true candidate (DTC)

If the PMF of RTC is concentrated near 0, the probability of eliminating the true candidate in MA would be decreased. DTC is defined as the 2nd moment of RTC as

$$\gamma(\Delta) \triangleq \sum_{\eta=0}^{2^j-1} \eta^2 . P(RTC = \eta), \tag{6-29}$$

i.e., the dispersion of RTC away from zero. This is a function of SNR, Δ and j. DTC is plotted as a function of Δ for SNR=5.208 dB and various js in Figure 6-6.

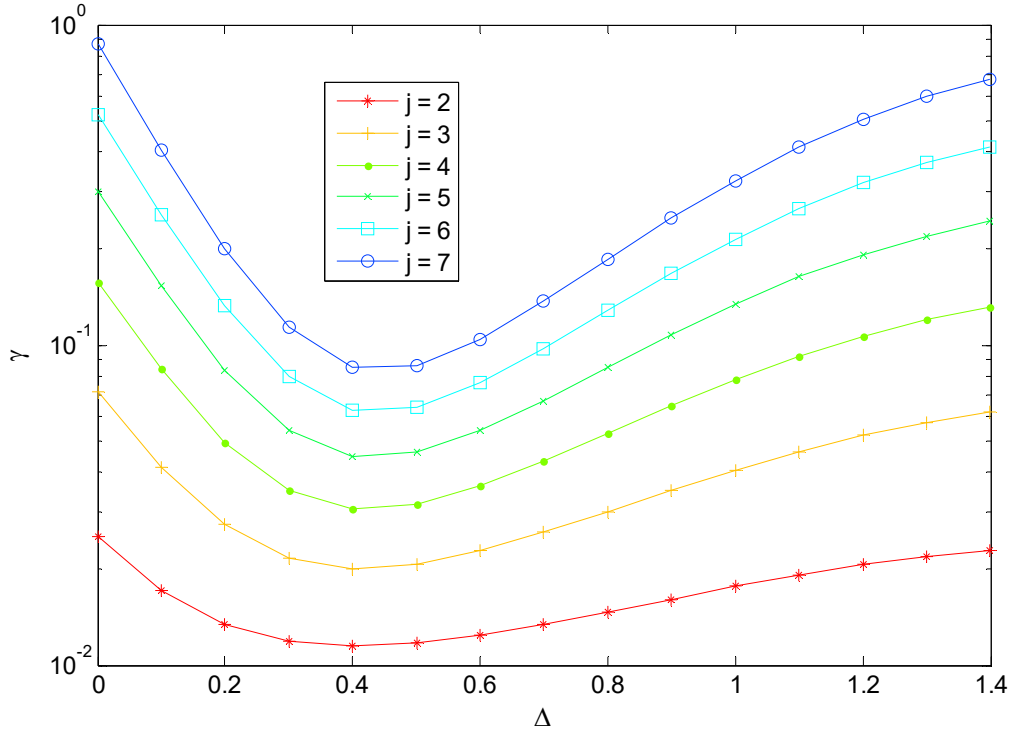


Figure 6-6. DTC versus Δ for SNR = 5.208 dB and $j = 2,3,\ldots,7$.

It can be seen that DTC is a unimodal function of Δ in the represented interval, *i.e.,* it has only one local minimum. DTC is used as the cost function for the quantizer optimization problem

$$\Delta_{opt} = \underset{\Delta}{\operatorname{argmin}}\, \gamma(\Delta), \tag{6-30}$$

and this method is called minimum DTC (MDTC).

Since a closed form for DTC and its gradient is not available and calculating DTC for all Δs is computationally complex, a sequential technique is used to minimize DTC. As

DTC is a unimodal function of Δ, the golden section search (GSS) is used to determine optimum Δ [78]. GSS sequentially brackets the minimum of the cost function. The minimum is bracketed by the triple of points $(\Delta_1, \Delta_2, \Delta_3)$ if $\gamma(\Delta_2) < \min[\gamma(\Delta_1), \gamma(\Delta_3)]$, and $\Delta_1 < \Delta_2 < \Delta_3$. In each iteration, GSS selects two points Δ_2 and Δ_3 inside the searching interval (Δ_1, Δ_4) to make two tuples $(\Delta_1, \Delta_2, \Delta_3)$ and $(\Delta_2, \Delta_3, \Delta_4)$ where $\Delta_1 < \Delta_2 < \Delta_3 < \Delta_4$ and then limits its searching spaces to the tuple which brackets the minimum. Length of the searching interval is decreased by factor of 0.618 in every iteration, and consequently length of the interval in k^{th} iteration is,

$$\Delta_4^{(k)} - \Delta_1^{(k)} = 0.618^k(\Delta_4^{(0)} - \Delta_1^{(0)}). \tag{6-31}$$

Thus, given the initial interval $(\Delta_1^{(0)}, \Delta_4^{(0)})$, the minimum number of steps needed for the uncertainty interval of e is equal to $\left\lceil \log_{0.618}\left(\frac{e}{\Delta_4^{(0)} - \Delta_1^{(0)}}\right)\right\rceil$.

To find Δ_{opt} using GSS, initial values of searching interval is set to $(\Delta_1^{(0)} = 0.1, \Delta_4^{(0)} = 1.0)$ and $e = 0.01$. in Figure 6-7, Δ_{opt} is drawn as a function of SNR for method MDTC for various number of decoding stages (j). In the operational SNRs (between 4dB and 8dB) Δ_{opt} obtained in MDTC is around 0.4 and 0.6. It can be seen that Δ_{opt} obtained in MDTC (when j is large) is about 0.2 greater than Δ_{opt} obtained in MMI.

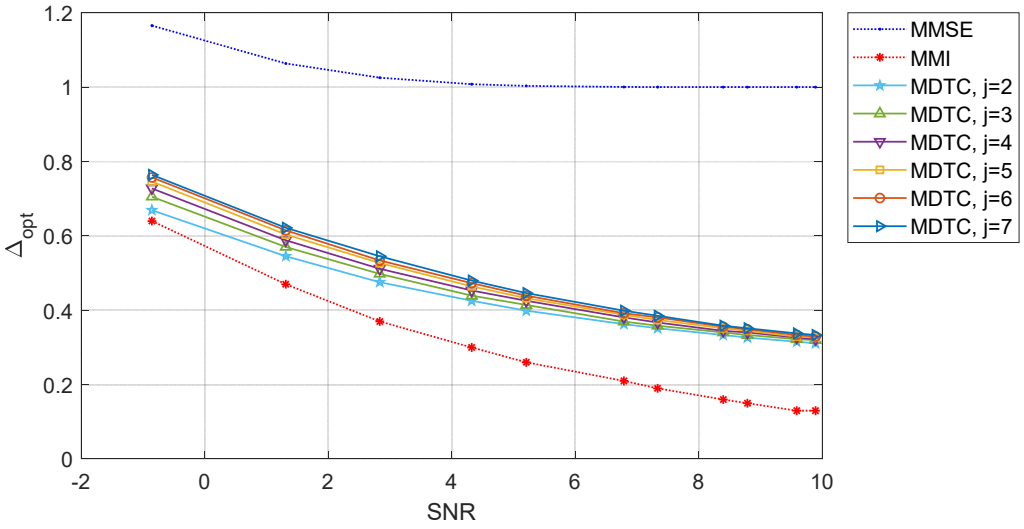


Figure 6-7. Optimum Δ versus channel SNR for MDTC (j=2,3, …, 7), MMI and MMSE quantizers.

6.3.2. Probability of true candidate ranked zero or one

In the MDTC method, the PMF of RTC must be calculated to calculate DTC. However, the calculation of the PMF is computationally complex especially for large decoding trees. As in MA, only a small portion of candidates survive; the probability that the true candidate is the first or second ranked candidate is more important than the probability of the other ranks. In other words, the probability that the true candidate is among the best two candidates (with the highest scores) is more important than when it is among the best three or more candidates.

. In this section, an alternative approach is proposed based on RTC.

In this approach, the probability of RTC being ranked the first or second, *i.e.,* $RTC = 0\ or\ 1$ is maximized. Similar to Section 6.3.1, these probabilities are calculated given that

the bit sequence $\boldsymbol{u} = \mathbf{0}$ is transmitted, *i.e.,* the true candidate is the full zero candidate. It is shown that, to calculate this, the value of the PMF only at one point is required.

6.3.2.1. Calculating P(RTC = 0)

Given $\boldsymbol{u} = \mathbf{0}$, the only condition that causes the true candidate $\tilde{\boldsymbol{u}} = \mathbf{0}$ be top ranked (RTC=0), is that every input (y_i) to the quantizer, is either in $(-\infty, -\Delta]$ or in $(-\Delta, 0]$, and consequently $\forall z_i \in \{r_1, r_2\}$ as explained below.

The decoding metric is the summation of the logarithms of the transition probabilities, *i.e.,* $\log \rho_1$, $\log \rho_2$, $\log \rho_3$, and $\log \rho_4$ (see Equations (6-15)-(6-18)). The candidate whose bits are generated as

$$\tilde{u}_i = \begin{cases} 0 \ : & z_i \in \{r_1, r_2\} \\ 1 \ : & z_i \in \{r_3, r_4\} \end{cases}, \tag{6-32}$$

has the largest decoding metric which is equal to $n_1 \log(\rho_1) + n_2 \log(\rho_2)$, where n_1 is the number of z_i's is either r_1 or r_4 and n_2 is the number of z_i's is either r_2 or r_3. Consequently, when $\forall z_i \in \{r_1, r_2\}$, the true candidate, *i.e.,* $\tilde{\boldsymbol{u}} = \mathbf{0}$, has the highest metric and in turn it is top ranked. Contrarily, if $\exists z_i \in \{r_3, r_4\}$, the top ranked candidate has at least one bit other than zero which implies that the true candidate ($\tilde{\boldsymbol{u}} = \mathbf{0}$) is not top ranked.

Since the only condition for $RTC = 0$ is $\forall z_i \in \{r_1, r_2\}$, using Equations (6-15) and (6-16) the probability that the true candidate is top ranked is

$$P(RTC=0)=P(RTC=0|\boldsymbol{U}=\mathbf{0})=\prod_{i=1}^{j}P(\boldsymbol{Z}_i\in\{r_1,r_2\}|U_i=0)$$

$$=(\rho_1+\rho_2)^j=\left[1-Q\left(\frac{\sqrt{\mathrm{E_b}}}{\sigma}\right)\right]^j. \tag{6-33}$$

which is independent of the value of Δ.

6.3.2.2. Calculating P(RTC=1) for limited length of decoding tree

In this section, the probability that the true candidate is the 2nd top ranked candidate, *i.e.,* $P(RTC=1)$ is calculated as a function of Δ.

In Table 6.1, RTC is calculated for various outputs of the quantizer ($\boldsymbol{z}$) when $j=2$ and $\boldsymbol{u}=\mathbf{0}$. The last column of Table 6.1, shows the probability of observing $\boldsymbol{z}$. To calculate RTC in each row of this table, the decoding metric of every candidate $\tilde{\boldsymbol{u}}$ is calculated using a decoding tree. In Figure 6-8, this process is represented for $\boldsymbol{z}=[r_1,r_4]$. After generating and sorting the metrics of all candidates, a tie between the metrics of the true candidate $\tilde{\boldsymbol{u}}=00$ and $\tilde{\boldsymbol{u}}=11$ is observed which is handled using Equation (6-28). Thus, RTC is either 1 or 2 with the same probability 0.5 in this example. To calculate $P(RTC=1)$, the last two columns of Table 6.1 are multiplied element by element and summed up (see Equation (6-26)). This results in

$$P(RTC=1;j=2)=2\rho_1\rho_3+2\times\frac{1}{2}\rho_1\rho_4+2\times\frac{1}{2}\rho_2\rho_3. \tag{6-34}$$

Coefficient $\frac{1}{2}$ behind the 2nd and the 3rd terms are due to the tie between the metric of the true candidate and the metrics of other candidates.

Table 6.1. Conditions on $\boldsymbol{z}$ when RTC=1 for $j = 2$ and their probability

$\boldsymbol{z}$				
z_1	z_2	RTC	$P(RTC = 1 \mid \boldsymbol{Z} = \boldsymbol{z}, \boldsymbol{U} = \boldsymbol{0})$	$P(\boldsymbol{Z} = \boldsymbol{z} \mid \boldsymbol{U} = \boldsymbol{0})$
r_1	r_1	0	0	ρ_1^2
r_1	r_2	0	0	$\rho_1\rho_2$
r_1	r_3	1	1	$\rho_1\rho_3$
r_1	r_4	1 or 2	0.5	$\rho_1\rho_4$
r_2	r_1	0	0	$\rho_1\rho_2$
r_2	r_2	0	0	ρ_2^2
r_2	r_3	1 or 2	0.5	$\rho_2\rho_3$
r_2	r_4	3	0	$\rho_2\rho_4$
r_3	r_1	1	1	$\rho_1\rho_3$
r_3	r_2	1 or 2	0.5	$\rho_2\rho_3$
r_3	r_3	4	0	ρ_3^2
r_3	r_4	4	0	$\rho_3\rho_4$
r_4	r_1	1 or 2	0.5	$\rho_1\rho_4$
r_4	r_2	3	0	$\rho_2\rho_4$
r_4	r_3	4	0	$\rho_3\rho_4$
r_4	r_4	4	0	ρ_4^2

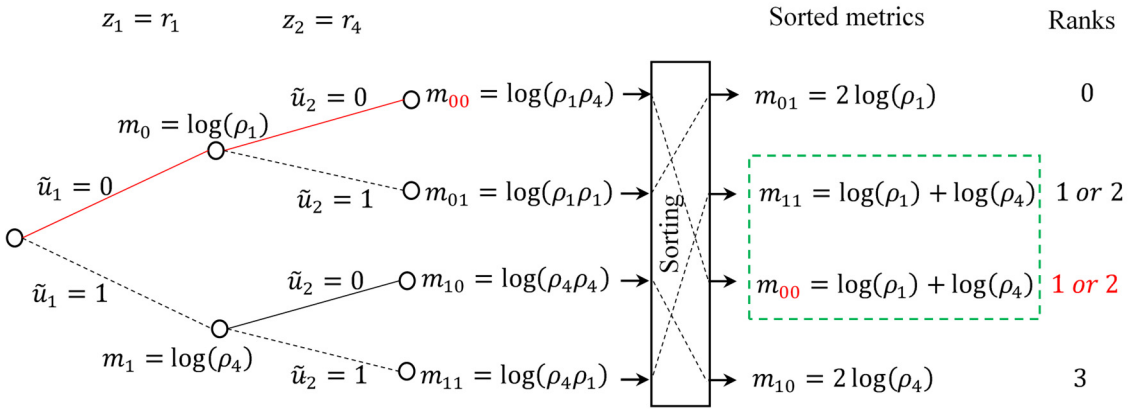


Figure 6-8. Calculation of RTC using decoding tree for $\mathbf{z}$=[r_1, r_4] given $\boldsymbol{u} = \boldsymbol{0}$.

To calculate $P(RTC = 1)$ for $j = 3$, the result of the ranking process is provided in Table 6.2. For reasons of space, in contrast to Table 6.1 in this table only $\boldsymbol{z}$'s resulting in $RTC = 1$ are listed. The rows of the table are grouped based on the probability of $\boldsymbol{z}$ (the last column). For example, the first three rows of this table are in the same group with the corresponding probability $\rho_1^2\rho_3$. The number of elements in each group is equal to the number of permutations of symbols $r_1, \ldots, r_4$ in $\boldsymbol{z}$. Thus, the number of elements in the 1st, 2nd and 4th group is $\frac{3!}{2!1!} = 3$ and the number of elements in the 3rd is $\frac{3!}{1!1!1!} = 6$.

Table 6.2. $\boldsymbol{z}$'s results in RTC=1 for $j = 3$ and their probability given $\boldsymbol{u} = \boldsymbol{0}$

Group	$\boldsymbol{z}$			RTC	$P(\boldsymbol{Z} = \boldsymbol{z}\|\boldsymbol{U} = \boldsymbol{0})$
	z_1	z_2	z_3		
1	r_1	r_1	r_3	1	$\rho_1^2\rho_3$
	r_1	r_3	r_1	1	$\rho_1^2\rho_3$
	r_3	r_1	r_1	1	$\rho_1^2\rho_3$
2	r_1	r_1	r_4	1 or 2 or 3	$\rho_1^2\rho_4$
	r_1	r_4	r_1	1 or 2 or 3	$\rho_1^2\rho_4$
	r_4	r_1	r_1	1 or 2 or 3	$\rho_1^2\rho_4$
3	r_1	r_2	r_3	1 or 2	$\rho_1\rho_2\rho_3$
	r_1	r_3	r_2	1 or 2	$\rho_1\rho_2\rho_3$
	r_2	r_1	r_3	1 or 2	$\rho_1\rho_2\rho_3$
	r_2	r_3	r_1	1 or 2	$\rho_1\rho_2\rho_3$
	r_3	r_1	r_2	1 or 2	$\rho_1\rho_2\rho_3$
	r_3	r_2	r_1	1 or 2	$\rho_1\rho_2\rho_3$
4	r_2	r_2	r_3	1 or 2 or 3	$\rho_2^2\rho_3$
	r_2	r_3	r_2	1 or 2 or 3	$\rho_2^2\rho_3$
	r_3	r_2	r_2	1 or 2 or 3	$\rho_2^2\rho_3$

Based on this table $P(RTC = 1; j = 3)$ is calculated as

$$P(RTC=1;j=3)=3\rho_1^2\rho_3+3\times\frac{1}{3}\rho_1^2\rho_4+6\times\frac{1}{2}\rho_1\rho_2\rho_3+3\times\frac{1}{3}\rho_2^2\rho_3= \\ =3\rho_1^2\rho_3+\rho_1^2\rho_4+3\rho_1\rho_2\rho_3+\rho_2^2\rho_3. \tag{6-35}$$

Noting $\rho_1>\rho_2>\rho_3>\rho_4$, the dominant terms are the terms which contain ρ_1. Thus, this probability is approximated using the first three dominant terms as

$$P(RTC=1;j=3)\cong 3\rho_1^2\rho_3+\rho_1^2\rho_4+3\rho_1\rho_2\rho_3\,. \tag{6-36}$$

Similarly, probability $P(RTC=1;j=4)$ is calculated and approximated as

$$P(RTC=1;j=4) \\ =\frac{4!}{3!}\rho_1^3\rho_3+\frac{1}{4}\frac{4!}{3!}\rho_1^3\rho_4+\frac{1}{2}\frac{4!}{2!}\rho_1^2\rho_2\rho_3+\frac{1}{3}\frac{4!}{2!}\rho_1\rho_2^2\rho_3+\frac{1}{4}\frac{4!}{3!}\rho_2^3\rho_3 \\ \cong 4\rho_1^3\rho_3+\rho_1^3\rho_4+\frac{1}{2}\frac{4!}{2!}\rho_1^2\rho_2\rho_3. \tag{6-37}$$

The approximation is done by keeping only the dominant terms containing ρ_1^3 and ρ_1^2, *i.e.,* the terms with two highest power of ρ_1.

6.3.2.3. Approximating P(RTC=1) for arbitrary length of decoding tree

Since in the operating SNRs, the optimum value of Δ is less than 1, ρ_1 is significantly greater than the other transition probabilities ρ_2, ρ_3, and ρ_4. By increasing j, the dominant terms in the calculation of $P(RTC = 1)$ are the terms with higher powers of ρ_1. To approximate $P(RTC = 1)$ in general for a decoding tree with length j, the two highest powers of ρ_1, *i.e.,* $\rho_1^{j-1}$and ρ_1^{j-2} are kept as dominant terms. It is shown that keeping only these two highest power of ρ_1 results in the approximation as $P(RTP = 1) = P(RTP = 1|\boldsymbol{U} = \mathbf{0}) \cong \psi(\Delta, j)$ where

$$\psi(\Delta, j) = j\rho_1^{(j-1)}\rho_3 + \rho_1^{(j-1)}\rho_4 + \frac{1}{2}j(j-1)\rho_1^{(j-2)}\rho_2\rho_3. \tag{6-38}$$

$P(RTP = 1|\boldsymbol{U} = \mathbf{0})$ in Equation (6-26) consists of two terms $P\left(RTC = 1\middle|\boldsymbol{Z} = \tilde{\boldsymbol{z}}^j, \boldsymbol{U} = \mathbf{0}\right)$ and $P\left(\boldsymbol{Z} = \tilde{\boldsymbol{z}}^j|\boldsymbol{U} = \mathbf{0}\right)$. Using the memorylessness of the channel (see Equations (6-27) and (6-25)), both of these terms are only dependent on the number of symbols $r_1, \dots, r_4$ in $\tilde{\boldsymbol{z}}^j$ which are denoted by random variables Ψ_1,…, Ψ_4, respectively. These two probabilities are written as

$$\begin{aligned} P\left(RTC = 1\middle|\boldsymbol{Z} = \tilde{\boldsymbol{z}}^j, \boldsymbol{U} = \mathbf{0}\right) &= P(RTC = 1|\Psi_1, \Psi_2, \Psi_3, \Psi_4, \boldsymbol{U} = \mathbf{0}), \\ P\left(\boldsymbol{Z} = \tilde{\boldsymbol{z}}^j|\boldsymbol{U} = \mathbf{0}\right) &= \rho_1^{\Psi_1}\rho_2^{\Psi_2}\rho_3^{\Psi_3}\rho_4^{\Psi_4}. \end{aligned} \tag{6-39}$$

Consequently Equation (6-26) is converted to

$$P(RTP = 1|\boldsymbol{U} = \mathbf{0}) =$$

$$= \sum_{\substack{0 \le a,b,c,d \le j \\ a+b+c+d=j}} \Big\{ P(RTC = 1|\Psi_1 = a, \Psi_2 = b,\ \Psi_3 = c, \Psi_4 = d, \boldsymbol{U} = \mathbf{0})$$

$$\times\ \rho_1^a \rho_2^b \rho_3^c \rho_4^d \times \frac{j!}{a!\, b!\, c!\, d!} \Big\}, \tag{6-40}$$

where $\frac{j!}{a!b!c!d!}$ is the number of $\tilde{\boldsymbol{z}}^j$s with a, b, c and d number of symbols r_1, r_2, r_3 and r_4, respectively.

Different values of $0 \le a \le j$ result in different powers of ρ_1 in Equation (6-40). If a is less than $j-2$, the corresponding terms in the summation is negligible in the approximation. If a is equal to j, all symbols of $\boldsymbol{Z}$ are equal to r_1 (note $a+b+c+d=j$) and in turn the true candidate is top ranked ($RTC = 0$) as explained in Section 6.3.2.1. The other two cases $a = j-1$ and $a = j-2$ are explained in the Sections 6.3.2.3.16.3.2.3.1 and 6.3.2.2 followed by Section 6.3.2.3.3 which studies the accuracy of the approximation.

6.3.2.3.1. Calculation of $P(RTC = 1|\Psi_1 = \mathrm{j}-1)$

If $\boldsymbol{z}$ contains $j-1$ number of symbol r_1, there are j possibilities for the position of the element which is not r_1. For the sake of simplicity, suppose the first $j-1$ symbols of $\boldsymbol{z}$ are r_1, *i.e.,* $\boldsymbol{z} = [r_1, r_1, \dots, r_1, z_j]$. The RTC for three different values of $z_j \in \{r_2, r_3, r_4\}$ are provided as follows.

i) $z_j = r_2$: The true candidate is top ranked ($RTC = 0$) as explained in Section 6.3.2.1.

ii) $z_j = r_3$: The metric of each candidate $\tilde{\boldsymbol{u}}$ can be written as

$$m_{\tilde{\boldsymbol{u}}} = \begin{cases} (j-1-\zeta)\log\rho_1 + \zeta\log\rho_4 + \log\rho_2 & : \ \tilde{u}_j = 1 \\ (j-1-\zeta)\log\rho_1 + \zeta\log\rho_4 + \log\rho_3 & : \ \tilde{u}_j = 0 \end{cases} \tag{6-41}$$

where $\zeta \geq 0$ is the number of bit 1 in the first $j-1$ elements of $\tilde{\boldsymbol{u}}$. If $\zeta = 0$ and $\tilde{u}_j = 1$, $m_{\tilde{\boldsymbol{u}}}$ is maximum and equal to $(j-1)\log\rho_1 + \log\rho_2$. If $\zeta = 0$ and $\tilde{u}_j = 0$ ($\tilde{\boldsymbol{u}}$ is true candidate), $m_{\tilde{\boldsymbol{u}}}$ is equal to $m_{\mathbf{0}} = (j-1)\log\rho_1 + \log\rho_3$ which is the 2nd largest metric. In other cases, *i.e.,* $\zeta > 0$, $m_{\tilde{\boldsymbol{u}}}$ is less than the metric of the true candidate $m_{\mathbf{0}}$. Therefore, $\boldsymbol{z} = [r_1, r_1, \dots, r_1, r_3]$ and all of its j permutations result in $RTC = 1$ with probability 1. The probability of receiving such $\boldsymbol{z}$'s is $j\rho_1^{(j-1)}\rho_3$ which is the first term in Equation (6-38).

iii) $z_j = r_4$: The metric of each candidate $\tilde{\boldsymbol{u}}$ can be written as

$$\begin{aligned} m_{\tilde{\boldsymbol{u}}} &= \begin{cases} (j-1-\zeta)\log\rho_1 + \zeta\log\rho_4 + \log\rho_1 & : \ \tilde{u}_j = 1 \\ (j-1-\zeta)\log\rho_1 + \zeta\log\rho_4 + \log\rho_4 & : \ \tilde{u}_j = 0 \end{cases} \\ &= \begin{cases} (j-\zeta)\log\rho_1 + \zeta\log\rho_4 & : \ \tilde{u}_j = 1 \\ (j-1-\zeta)\log\rho_1 + (\zeta+1)\log\rho_4 & : \ \tilde{u}_j = 0 \end{cases} \end{aligned} \tag{6-42}$$

where $\zeta \geq 0$ is the number of bit 1 in the first $j-1$ elements of $\tilde{\boldsymbol{u}}$. If $\zeta = 0$ and $\tilde{u}_j = 1$, $m_{\tilde{\boldsymbol{u}}}$ is maximum and equal to $j\log\rho_1$. If $\zeta = 0$ and $\tilde{u}_j = 0$ ($\tilde{\boldsymbol{u}}$ is true candidate), $m_{\tilde{\boldsymbol{u}}}$ is equal to $m_{\mathbf{0}} = (j-1)\log\rho_1 + \log\rho_4$ which is the 2nd largest metric. However, there is a tie between the metric of the true candidate and the candidates with $\zeta = 1$ and $\tilde{u}_j = 1$.

There are $j-1$ number of such candidates whose metric is equal to the metric of the true candidate. It can be easily shown that the metric of the other candidates is less than $m_{\mathbf{0}}$. $\boldsymbol{z}=[r_1, r_1, \dots, r_1, r_4]$ and all of its j permutations result in $RTC=1$ with probability of $\frac{1}{j}$ due to a tie between the metrics. By multiplying this probability ($\frac{1}{j}$) and the probability of receiving such $\boldsymbol{z}$'s the 2nd term of Equation (6-38) is obtained ($\rho_1^{(j-1)}\rho_4$,).

6.3.2.3.2. Calculation of $P(RTC=1|\Psi_1=\mathrm{j}-2)$

In this section like Section 6.3.2.3.1, only $\boldsymbol{z}$ whose first $j-2$ elements are r_1 is analyzed and the result is then extrapolated to other permutations of $\boldsymbol{z}$. Assuming $\boldsymbol{z}=[r_1, r_1, \dots, r_1, z_{j-1}, z_j]$, where $z_{j-1}, z_j \in \{r_2, r_3, r_4\}$, different values of z_{j-1}, z_j results in various RTC which are provided as follows.

i) $z_{j-1}=z_j=r_2$: The true candidate is top ranked ($RTC=0$) as explained in Section 6.3.2.1.

ii) One of z_{j-1} and z_j is r_2 and the other is r_3: For sake of simplicity suppose $z_{j-1}=r_2$ and $z_j=r_3$. By separating the first $j-2$ bits of $\tilde{\boldsymbol{u}}$ the decoding metric is written as

$$m_{\tilde{\boldsymbol{u}}}=\begin{cases}(j-2-\varsigma)\log\rho_1+\varsigma\log\rho_4+\log\rho_2+\log\rho_2 & : \quad [\tilde{u}_{j-1},\tilde{u}_j]=01\\ (j-2-\varsigma)\log\rho_1+\varsigma\log\rho_4+\log\rho_2+\log\rho_3 & : [\tilde{u}_{j-1},\tilde{u}_j]=00\ or\ 11\\ (j-2-\varsigma)\log\rho_1+\varsigma\log\rho_4+\log\rho_3+\log\rho_3 & : \quad [\tilde{u}_{j-1},\tilde{u}_j]=10\,,\end{cases} \tag{6-43}$$

where $\varsigma \geq 0$ is number of bit 1 in the first $j-2$ elements of $\widetilde{\boldsymbol{u}}$. If $\varsigma = 0$ and $[\tilde{u}_{j-1}, \tilde{u}_j] = 01$ the metric is maximum and equal to $(j-2)\log\rho_1 + \log\rho_4 + 2\log\rho_2$. The metric of the true candidate is equal to $m_{\mathbf{0}} = (j-2)\log\rho_1 + \log\rho_2 + \log\rho_3$ and it is the 2nd largest metric. There is a tie between the metric of the true candidate, $\widetilde{\boldsymbol{u}} = \mathbf{0}$, and the metric of candidate $\widetilde{\boldsymbol{u}} = [0, \dots, 0, 1, 1]$ in Equation (6-43). The metric of the other candidates is obviously smaller than $m_{\mathbf{0}}$. Therefore, if $\boldsymbol{z}$ is in form of any permutation of $[r_1, \dots, r_1, r_2, r_3]$, RTC is 1 with probability ½ due to the tie. Probability of observing such $\boldsymbol{z}$'s (given $\boldsymbol{u} = \mathbf{0}$) is equal to $j(j-2)\rho_1^{j-1}\rho_2\rho_3$. Multiplication of these two probabilities makes the last term in Equation (6-38).

iii) $z_{j-1} = z_j = r_3$: Metric of true candidate is $m_{\mathbf{0}} = (j-2)\log\rho_1 + 2\log\rho_3$. There are at least two candidates whose metric is greater than $m_{\mathbf{0}}$. Two of them are $\widetilde{\boldsymbol{u}} = [0, \dots, 0, 1, 0]$ and $\widetilde{\boldsymbol{u}} = [0, \dots, 0, 0, 1]$ whose metrics are equal to $(j-2)\log\rho_1 + \log\rho_2 + \log\rho_3$. Therefore, RTC is greater than 1.

iv) At least One of z_{j-1} and z_j is r_4: For sake of simplicity suppose $z_{j-1} = r_4$ and $z_j \in \{r_2, r_3, r_4\}$. Metric of the true candidate is

$$m_{\mathbf{0}} = \begin{cases} (j-2)\log\rho_1 + \log\rho_4 + \log\rho_2 & : \quad z_j = r_2 \\ (j-2)\log\rho_1 + \log\rho_4 + \log\rho_3 & : \quad z_j = r_3 \\ (j-2)\log\rho_1 + \log\rho_4 + \log\rho_4 & : \quad z_j = r_4\,. \end{cases} \tag{6-44}$$

Thus, maximum value of $m_{\mathbf{0}}$ is $(j-2)\log\rho_1 + \log\rho_4 + \log\rho_2$. Similarly, metric of candidates $\widetilde{\boldsymbol{u}} = [0, \dots, 0, 1, 0]$ and $\widetilde{\boldsymbol{u}} = [0, \dots, 0, 1, 1]$ is summation of $(j-1)\log\rho_1$ with one

of $\log \rho_2$, $\log \rho_3$ and $\log \rho_4$ depending on the value of z_j. Minimum value of metric of these two candidates, $(j-1)\log \rho_1 + \log \rho_4$, is greater than maximum value of $\boldsymbol{m_0}$. Thus, if at least one of z_{j-1} and z_j is r_4, RTC is greater than 1.

6.3.2.3.3. Comparing $P(RTP = 1)$ and its approximation

In Figure 6-9 the exact value of $P(RTP = 1)$ and its approximation, *i.e.,* $\psi(\Delta, j)$ in Equation (6-38), are drawn as functions of Δ for a typical channel SNR= 6.7895 dB and various j's. It is seen that by increasing j, the accuracy of the approximation is reduced, especially in larger values of Δ. This is because of increasing the number of discarded terms in the approximation process by growing the length of the decoding tree, j. These neglected terms are the terms in $P(RTC = 1)$ whose power of ρ_1 is less than $j-2$. By increasing Δ, values of ρ_2, ρ_3 raises and in turn value of the discarded terms grows. This results in inaccuracy of the approximation for larger Δ and larger j. However, the approximation is accurate enough for Δ's in which $P(RTC = 1)$ is maximum.

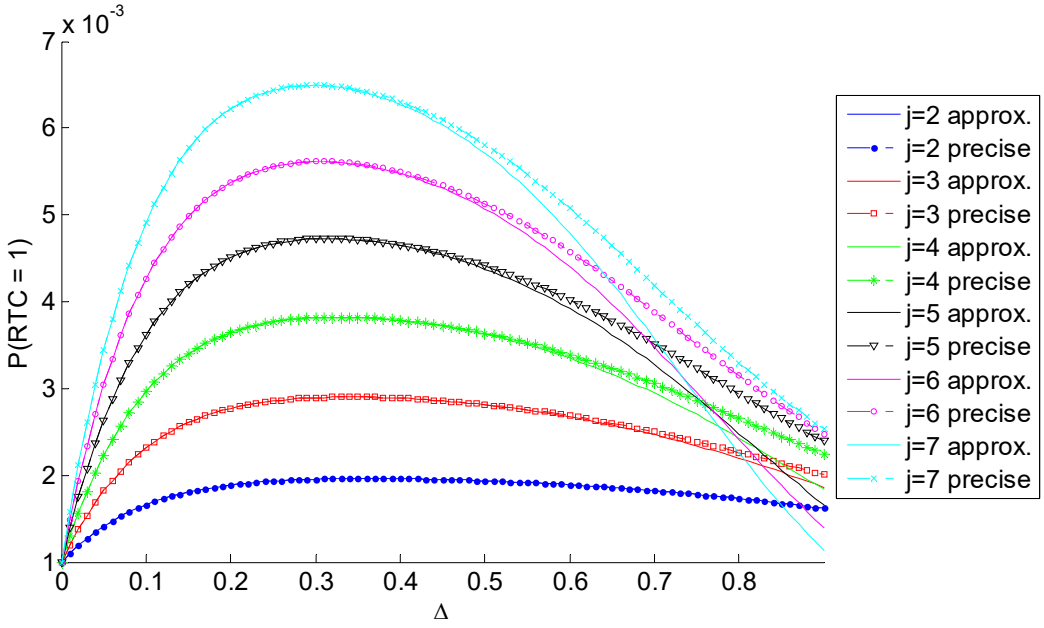


Figure 6-9. Precise $P(RTC = 1)$ and its approximation, $\psi(\Delta, j)$, for channel SNR= 6.7895 dB.

6.3.2.4.Maximizing $P(RTC = 1)$

As shown in Figure 6-9, $\psi(\Delta, \mathrm{j})$ is a unimodal function. Therefore, to find the optimum Δ satisfies $\frac{d\psi(\Delta,\mathrm{j})}{d\Delta} = 0$. Using the chain rule the derivative of Equation (6-38) is calculated as

$$\frac{d\psi(\Delta,\mathrm{j})}{d\Delta} = \frac{(j-1)\rho_1^{j-3} e^{\frac{-(\Delta^2+E_b)}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} \left[e^{\frac{-\Delta\sqrt{\mathrm{E_b}}}{\sigma^2}} \left(\rho_1^2 + \frac{1}{2} j\rho_1\rho_2 \right) \right.$$
$$\left. - e^{\frac{\Delta\sqrt{\mathrm{E_b}}}{\sigma^2}} \left(\frac{1}{2} j\rho_1\rho_3 + \rho_1\rho_4 + \frac{1}{2} j(j-2)\rho_2\rho_3 \right) \right]. \quad (6\text{-}45)$$

In this calculation, derivatives of ρ_1, ρ_2, ρ_3 and ρ_4 with respective to Δ are obtained using $\frac{dQ}{dx} = -\frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ as

$$\frac{d\rho_1}{d\Delta} = -\frac{d\rho_2}{d\Delta} = \frac{-1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{\Delta-\sqrt{E_b}}{\sigma\sqrt{2}}\right)^2}, \tag{6-46}$$

$$\frac{d\rho_3}{d\Delta} = -\frac{d\rho_4}{d\Delta} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{\Delta+\sqrt{E_b}}{\sigma\sqrt{2}}\right)^2}. \tag{6-47}$$

As shown in Figure 6-4, $\rho_1 + \rho_2$ and $\rho_3 + \rho_4$ are constant and independent of Δ. Therefore, negative relations $\frac{d\rho_1}{d\Delta} = -\frac{d\rho_2}{d\Delta}$ and $\frac{d\rho_3}{d\Delta} = -\frac{d\rho_4}{d\Delta}$ was expected. Noting $j > 1$ and $\rho_1 \neq 0$, $\frac{d\psi(\Delta,j)}{d\Delta} = 0$ results in

$$\frac{\rho_1^2 + \frac{1}{2} j\rho_1\rho_2}{\frac{1}{2} j\rho_1\rho_3 + \rho_1\rho_4 + \frac{1}{2} j(j-2)\rho_2\rho_3} = e^{\frac{2\Delta\sqrt{E_b}}{\sigma^2}}. \tag{6-48}$$

Taking logarithm from both sides, Δ is obtained as

$$\Delta = \frac{\sigma^2}{2\sqrt{E_b}} \ln\left[\frac{2\rho_1^2 + j\rho_1\rho_2}{j\rho_1\rho_3 + 2\rho_1\rho_4 + j(j-2)\rho_2\rho_3}\right]. \tag{6-49}$$

Noting for the operational SNRs (4-8 dB) ρ_1 is the significantly higher than the other transition probabilities. Thus, the dominant terms in the numerator and the denominator inside logarithm function in this equation are $2\rho_1^2$ and $j\rho_1\rho_3$, respectively. Keeping only these dominant terms, it turns into

$$\Delta = \frac{\sigma^2}{2\sqrt{E_b}} \ln\left[\frac{2\rho_1}{j\rho_3}\right] = \frac{\sigma^2}{2\sqrt{E_b}} \ln\left[\frac{2Q\left(\frac{\Delta - \sqrt{\mathrm{E_b}}}{\sigma}\right)}{jQ\left(\frac{\sqrt{\mathrm{E_b}}}{\sigma}\right) - jQ\left(\frac{\sqrt{\mathrm{E_b}} + \Delta}{\sigma}\right)}\right]. \tag{6-50}$$

This is a nonlinear equation in form of $\Delta = g(\Delta)$, and it is solved using *fixed-point iteration* method [76] as explained in Equation (6-14). This quantizer design is named *maximum probability of RTC=1* (MPRTC1) method.

In Figure 6-10, Δ_{opt} is drawn as a function of SNR for MPRTC1 method along with the other methods (MSE, MMI and MDTC). It can be seen that Δ_{opt} in MPRTC1 method follows the same trend of the other MDTC. MPRTC1 method is computationally faster than MDTC, because in MDTC all candidates and their metric must be generated and sorted while in MPRTC1 only Equation (6-50) is solved.

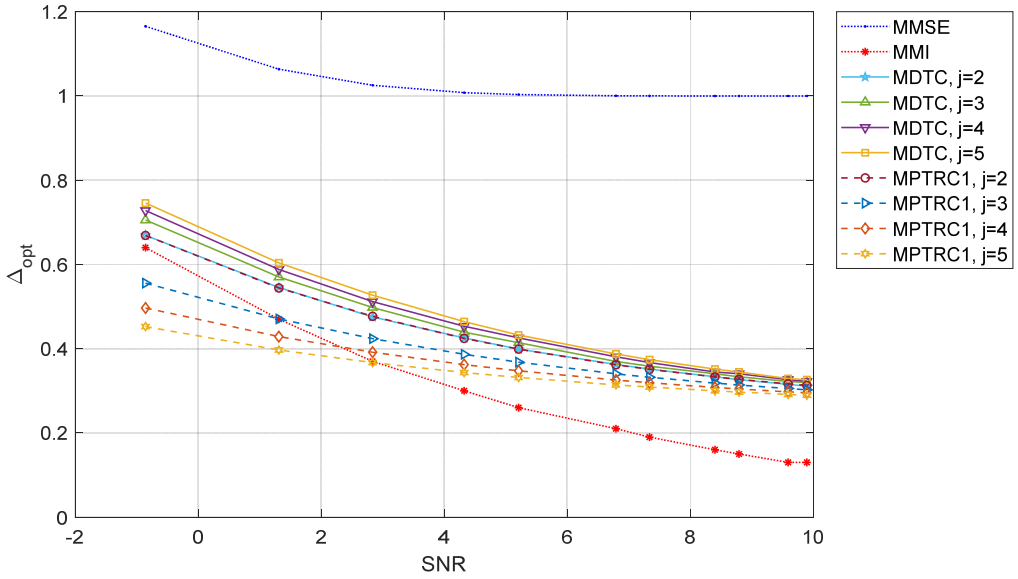


Figure 6-10. Optimum Δ versus channel SNR for MDTC (j=2, 3,4,5), MPRTC1 (j=2, 3,4,5), MMI and MMSE quantizer designs.

6.4. Simulation results

In the simulations, bin sequences are randomly generated and packetized. Then they are encoded, BPSK modulated and transmitted through an AWGN channel whose error probability is p which is a function of SNR, *i.e.,* $p = Q(\sqrt{2 \times SNR})$. The used channel SNR's 8.3982 dB 7.335 dB, 6.7895 dB, 5.208 dB and 4.3232 dB correspond to the channel bit error rates $p = 1\times10^{-4}$, 5×10^{-4}, 10^{-3}, 5×10^{-3} and 10^{-2}, respectively. The energy per bit of the transmitted signal, *i.e.,* E_b, is always fixed to1 and energy of noise is varied. The received noisy signal is quantized with quantization parameter Δ which is spanned from 0.2 to 0.8 with steps of 0.1. The tests are done for FS probabilities $\epsilon = 0.05, 0.1$ and EOPS probability $\delta = 0.01$. Parameter M that controls the performance and complexity tradeoff

of the MAP decoder, is set to 32, 64 and 128. The performance of the transmission is evaluated in terms of PER for various values of quantization parameter Δ.

PER is plotted as a function of Δ in different channel SNRs and ε's in Figure 6-11, Figure 6-12 and Figure 6-13 respectively for $M = 128$, $M = 64$ and $M = 32$. To find the Δ minimizing PER, curves are fitted to the data points $\left(\Delta_i, f(\Delta_i)\right)$ using smoothing cubic spline interpolation technique [79], [80], where $[\Delta_1, \dots, \Delta_7] = [0.2, 0.3, \dots, 0.8]$ and $f(\Delta_i)$ is the measured PER at Δ_i. A cubic spline is a continuous piecewise polynomial function of degree 3 whose first and second derivatives are continuous at the data points except the first and last points, *i.e.,* Δ_1 and Δ_7. In other words, slope and curvature of curves do not vary in in the transition at the data points. The spline function $\hat{f}(\Delta)$ is found out to minimize

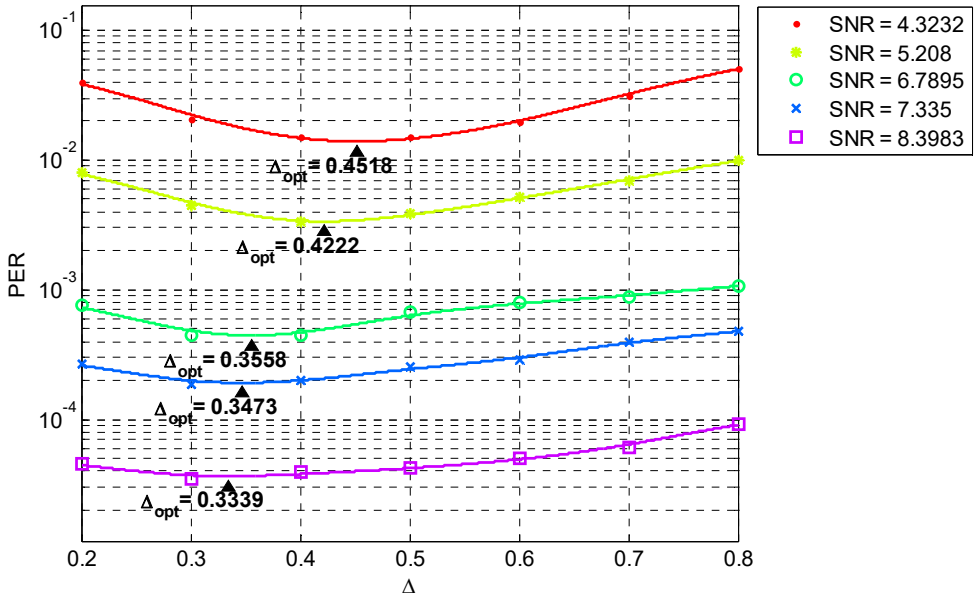
$$\tau \sum_{i=1}^{7} \left(f(\Delta_i) - \hat{f}(\Delta_i)\right)^2 + (1-\tau) \int \left(\frac{d^2 \hat{f}(\Delta)}{d\Delta^2}\right)^2 d\Delta, \tag{6-51}$$

where $\tau \geq 0$ is smoothing parameter. This criterion trades off the least square error (the first term) and regularization term (the second term). The regularization term is dominant when the spline function is too wiggly and in turn its second derivate is too high. To control this tradeoff, value of τ is set to $\tau = \left(1 + \frac{\kappa^3}{6}\right)^{-1}$ as recommended in [81], where κ is the average spacing of the data points. Here, as the distance between consecutive Δ_is is 0.1, κ is equal to 0.1 and in turn $\tau = 0.999833$. Minimum values of PER are determined using the interpolation functions. The minimum points are shown by solid triangles in Figure 6-11, Figure 6-12 and Figure 6-13. This method of finding Δ_{opt} based on simulations is called PER-Sim. It can be seen that the minimum points move to the left

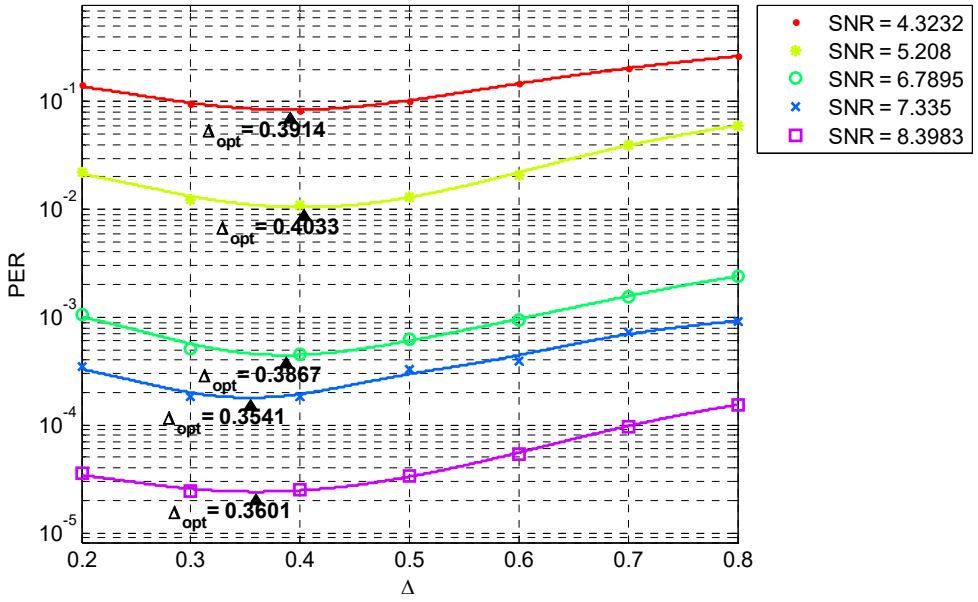
by increasing the channel SNR which is consistent with the calculated trends depicted in Figure 6-10.

It can be seen that, the minimum value of PER is lower than its maximum with a factor between 0.1 and 0.3, *i.e.*, the range of PER variation (Depth of the valley-shaped curves) is 0.5 to 1 decade. It shows that optimizing the channel quantizer may result in PER reduction with an order of magnitude. Comparing the curves with the same values of M and channel SNR, PER range is broader apparently when ε is lower, since improving the probability of surviving the true candidate in decoding tree done increasingly by quantizer optimization instead of the FS. Asymptotically, when ε merges toward 1 the true candidate is certainly top ranked no matter what quantization parameter Δ is.

Keeping ε and channel SNR constant and changing value of M do not have a noticeable effect on the PER range. Since PMF of RTC is mostly tended towards RTC=0, effect of changing Δ on the PMF is important for smaller RTCs. Therefore, considering more candidates by increasing value of M does not affect the PER improvement using the quantization optimization.

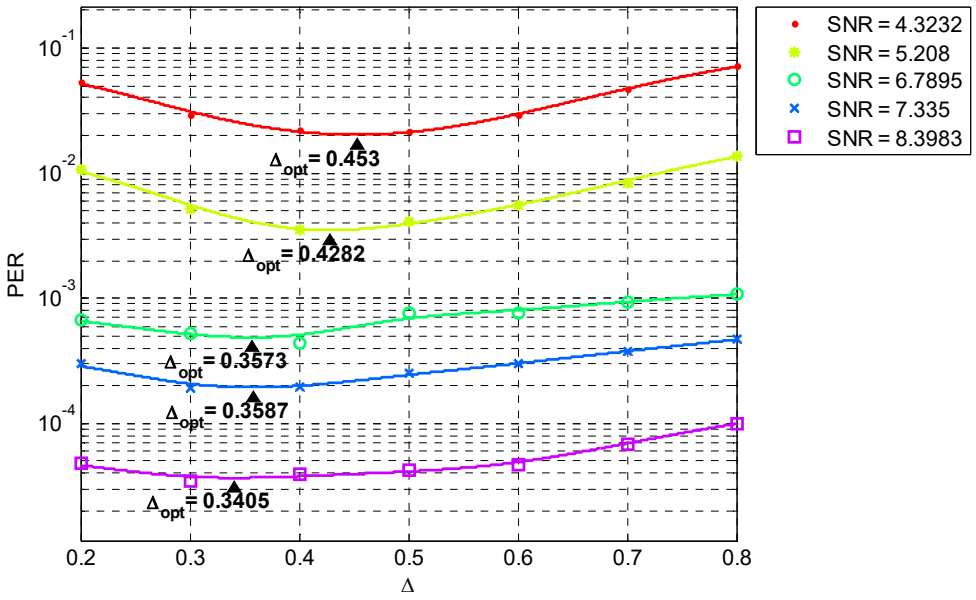


(a)

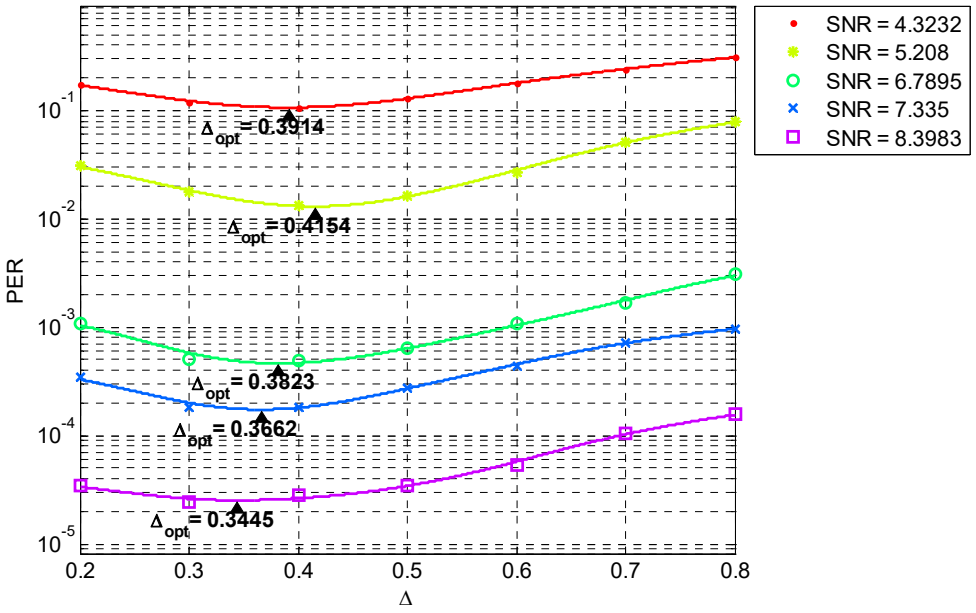


(b)

Figure 6-11. Plot of PER versus Δ for various channel SNRs, M=128 and FS probability (a) $\epsilon = 0.1$ (b) $\epsilon = 0.05$.

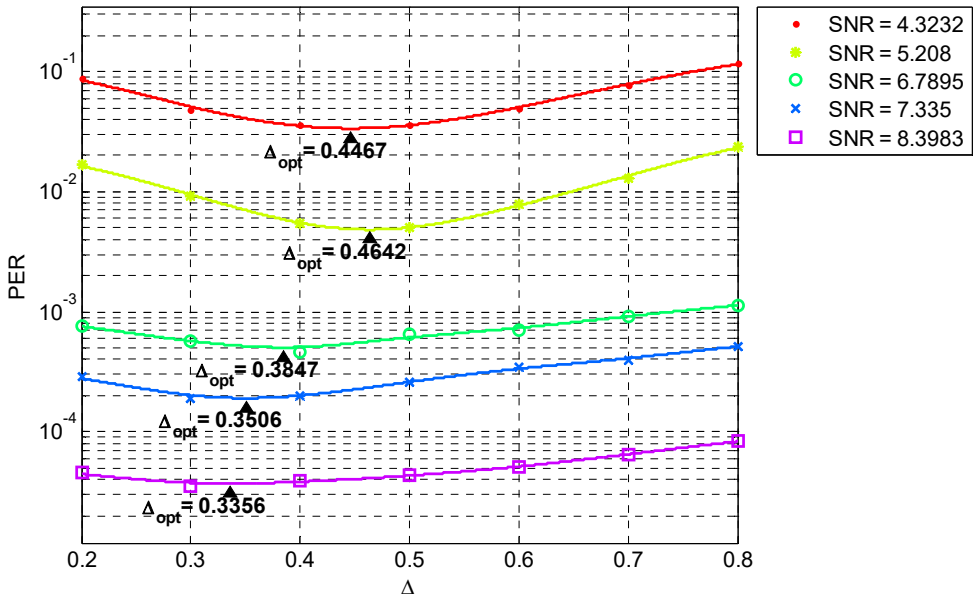


(a)

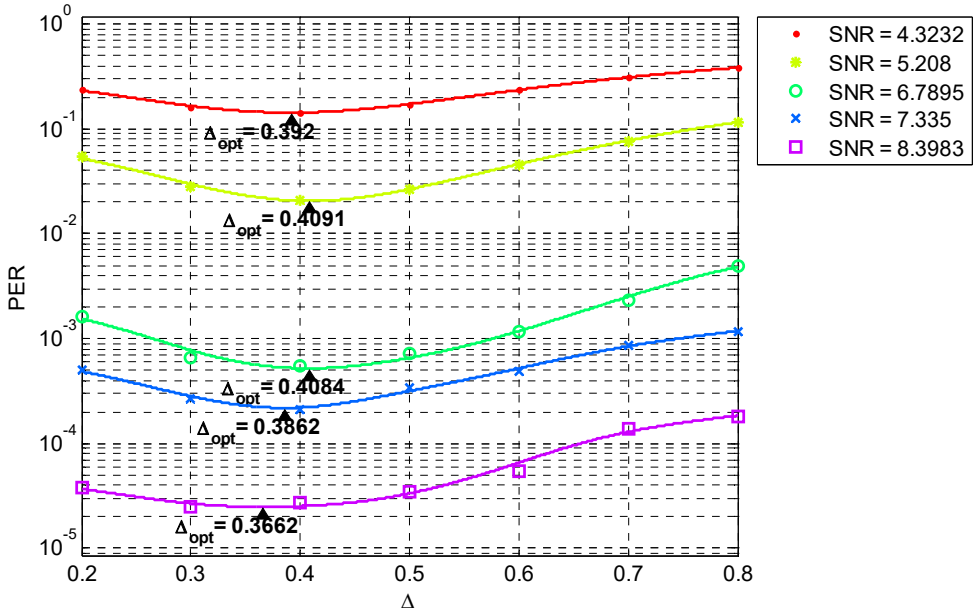


(b)

Figure 6-12. Plot of PER versus Δ for various channel SNRs, M=64 and FS probability (a) $\epsilon = 0.1$ (b) $\epsilon = 0.05$.



(a)

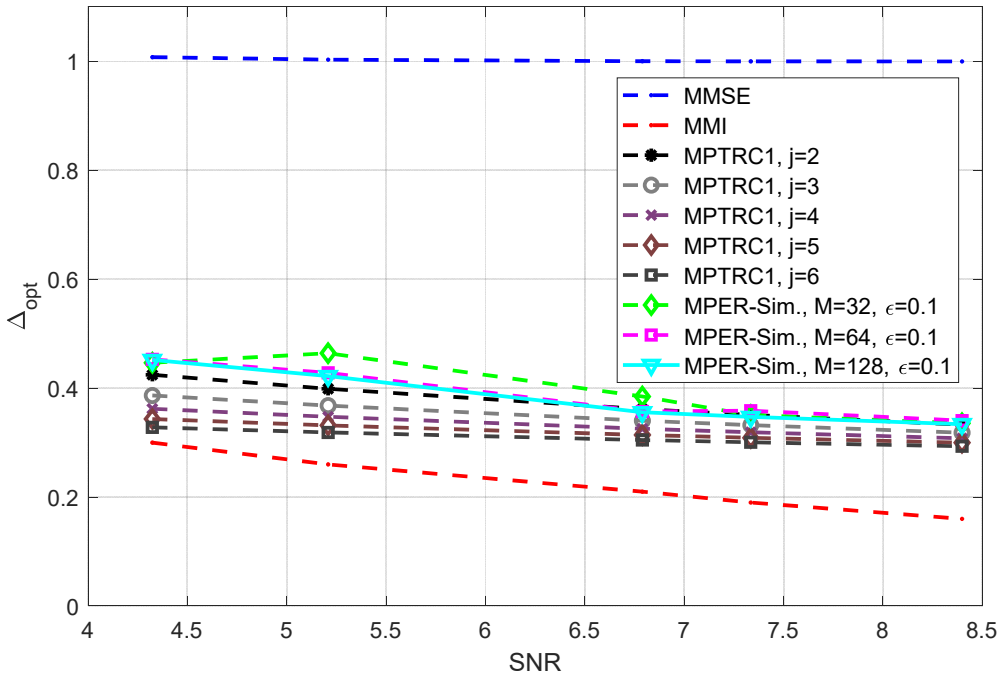


(b)

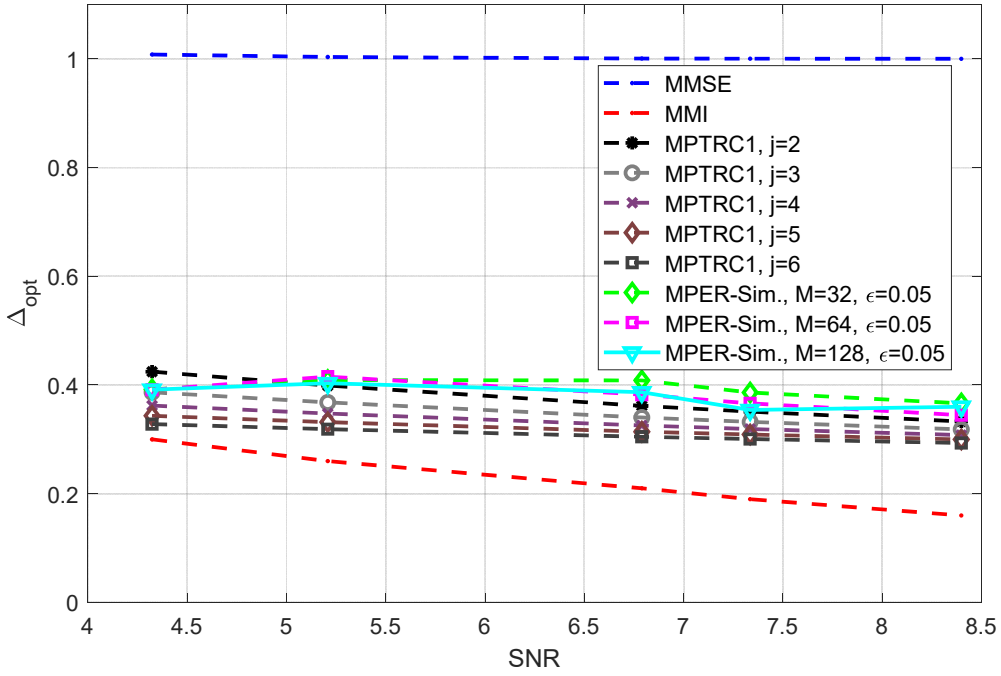
Figure 6-13. Plot of PER versus Δ for various channel SNRs, M=32 and FS probability (a) $\epsilon = 0.1$ (b) $\epsilon = 0.05$.

In Figure 6-14, Δ_{opt} is plotted as a function of SNR based on PER-Sim and for the methods mentioned in Section 6.3, *i.e.,* MMSE, MMI and MPRTC1 (j=2,…,6). In PER-Sim, Δ_{opt} is found for different values of ε and M ($\varepsilon = 0.05, 0.1$ and M=32, 64, 128). MPRTC1 provides the closest results to the simulations among three analytical methods in different values of SNR (4.3-8.4 dB). By increasing value of M, the PER-Sim gets closer to MPRTC1 especially in higher SNRs. In MPRTC1, the probability that the true candidate is ranked "1" or second best, among all possible candidates is maximized. However, in simulation, MA keeps only limited number of candidates which increases with the value of M. Therefore, using larger value of M in the simulations results in the closer results to MPRTC1. This is more noticeable in higher SNR due to the fact that in MPRTC1, the approximations used are more accurate in higher SNRs (ρ_1 is more dominate over ρ_2, ρ_3 and ρ_4 when the SNR is higher).

The simulations are closer to MPRTC1 for small values of j (about $j = 2, 3$). Because in MA structure of the decoding tree at each stage is not a complete binary tree, *i.e.* a tree in which every node is connected to two other nodes in its next stage. As shown in Figure 3-8 to Figure 3-11, the decoding tree is made up of several short length complete subtrees connected to each other by long roots. A root is set of consecutive nodes each of which is connected to only one node in its next stage. In other words, no branching to two different paths happen in a root. MPTRC1 is designed for complete decoding trees. Since length of these complete subtrees is short, this method provides better results in small values of j.



(a)

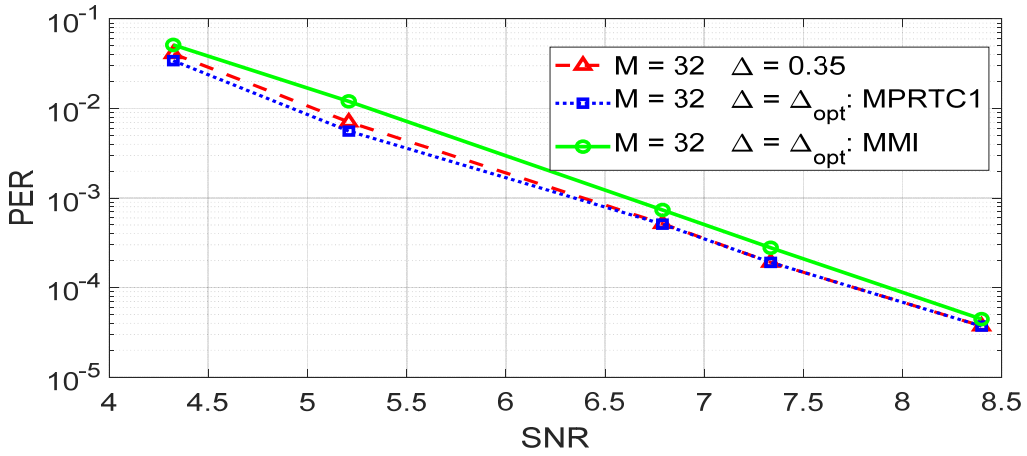


(b)

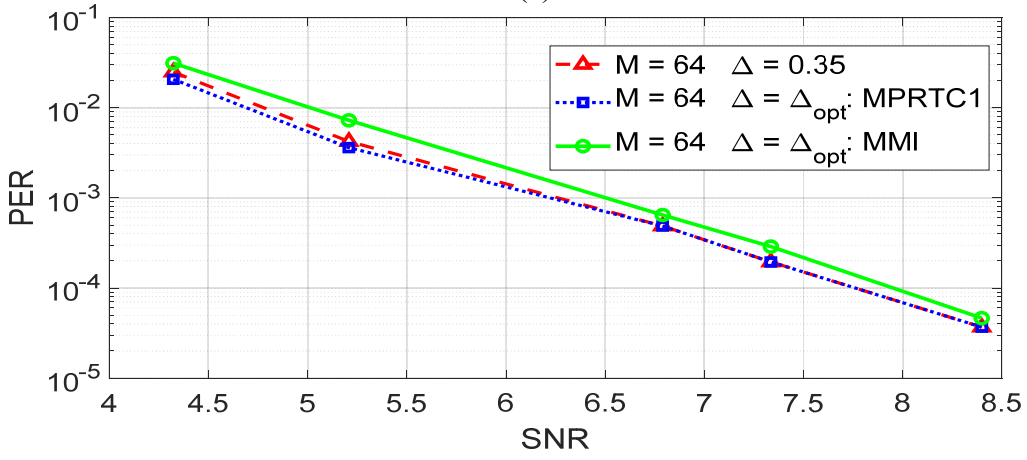
Figure 6-14. Optimum Δ versus channel SNR for MPRTC1 (j=3, 4), MMI and MMSE methods and PER-Sim, for various M and, (a) $\epsilon = 0.1$ and (b) $\epsilon = 0.05$.

The quantizer parameter can be adaptively changed with varying channel conditions. In other words, in an adaptive design, Δ changes for different values of SNR adaptively. In Figure 6-15, two adaptive quantizers based on MMI and MPRTC1 (j=3) are compared with a quantizer with constant Δ in terms of PER in different SNRs. It can be seen that the MMI method degrades PER comparing to constant quantizer. The MPRTC1 adaptive method improves PER most noticeably in lower SNRs, as the optimum delta as a function of SNR varies more in lower SNRs as depicted in Figure 6-10. The only cost of the adaptive method is solving Equation (6-50) for every value of SNR. To solve this equation using *fixed-point iteration* method [76], the initial value of Δ is set to 0.5 and the iterative proceeds until the difference between Δs in two consecutive iterations is less than 10^{-4}. In these simulations, the average number of iterations is 3.2 and the average time for each point is 2.1×10^{-4} on MATLAB using a personal computer (PC) with a core i7 CPU 2.67GHz. Comparing to the rest of decoding process this complexity overhead is negligible.

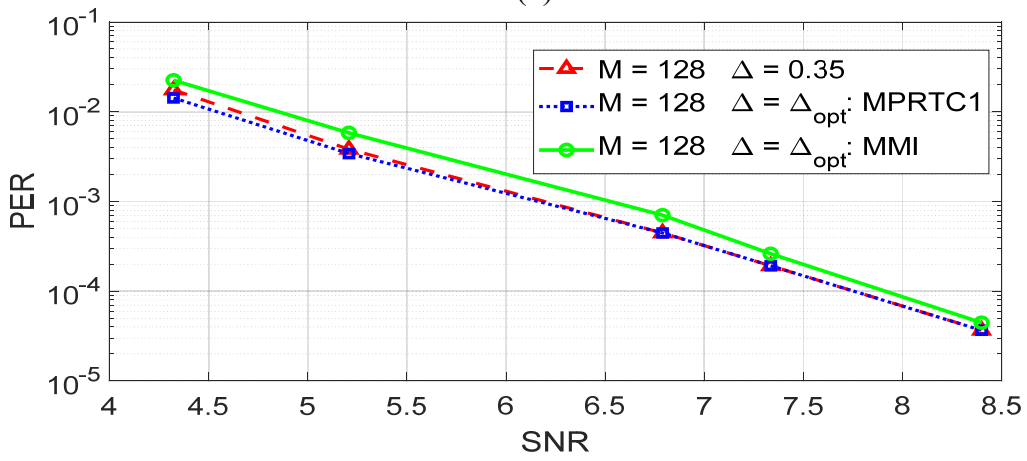
In PER-Sim, measuring PER is based on a Monte-Carlo method which make it computationally complex especially for large SNRs [82]. In contrast to PER-Sim, the proposed method MPRTC1 determines the quantization parameter analytically using Equation (6-50). In MPRTC1, it is assumed that the output bits of the encoder are independent and uniformly distributed. In general, this method can be extended to be used in ML decoders which use MA as candidate generator.



(a)



(b)



(c)

Figure 6-15. PER versus channel SNR for three methods fixed quantizer $\Delta = 0.3468$, adaptive quantizers $\Delta=\Delta_{opt}$:MMI and $\Delta=\Delta_{opt}$:MPRTC1(j=3) for (a) M=32, (b) M=64 and (c) M=128.

As explained in Chapter 3, SC can be used to eliminate the candidates violating the syntax of the video compression standard. Here, the channel quantizer is evaluated when the MAP decoder uses SC along with FS. Similar to Chapter 3, the video *Foreman* in QCIF format is compressed using the reference software H.264, *i.e.,* JM [63]. All frames are divided into slices each of which consists of rows of MBs, and then *intra* encoded with QP of 28. *Intra* modes, after the signaling process in the H.264 standard and binarization, are encoded with $\epsilon = 0.05$. Syntax checking rate (SCR) is defined as the ratio between the number of decoding tree stages in which the SC is active and the total number of stages. In the simulations SCR is chosen as 0, ¼ and 1 which correspond to SC is not active, SC is active every 4 stages and SC is active in all stages, respectively.

In Table 6.3 the optimum Δ minimizes the PER, *i.e.,* Δ_{opt}(MPER-Sim), and the corresponding PER is provided for various channel SNRs and SCRs. In this table, Δ_{opt} is provided for the other methods using the other methods MMI MMSE and MPTRC1. Comparing Δ_{opt}(MPER-Sim) and the calculated Δ_{opt} for analytical methods, the effectiveness of MPTRC1 method is shown in this table. Using SC improves PER with the cost of computational complexity. However, this complexity can be moderated by choosing smaller value of SCR. For example, changing SCR from 1 to ¼ almost halves the decoding time while it increases PER slightly. The decoder can change value of SCR adaptively based on the available resources and the channel SNR.

Table 6.3. Optimum Δ and for various channel SNRs for MPRTC1(j=2,…,5), MMI and MMSE methods and PER-Sim, for M=32, M=64 and M=128 and $\epsilon = 0.05$

	SCR	SNR (dB)	4.323	5.208	6.790	7.335	8.398	T (sec/packet)
M=32	1	PER	3.69E-02	4.25E-03	1.59E-04	4.45E-05	6.95E-06	1.130
		Δ_{opt} (MPER-Sim)	0.457	0.464	0.454	0.398	0.370	
	1/4	PER	3.69E-02	4.93E-03	1.63E-04	4.45E-05	6.95E-06	0.585
		Δ_{opt} (MPER-Sim)	0.446	0.473	0.457	0.401	0.370	
	0	PER	1.43E-01	2.06E-02	5.21E-04	2.19E-04	2.48E-05	0.079
		Δ_{opt} (MPER-Sim)	0.392	0.4091	0.4084	0.3862	0.3662	
M=64	1	PER	1.96E-02	1.93E-03	1.42E-04	4.41E-05	7.15E-06	2.252
		Δ_{opt} (MPER-Sim)	0.470	0.461	0.412	0.380	0.366	
	1/4	PER	2.01E-02	1.93E-03	1.45E-04	4.48E-05	7.22E-06	1.187
		Δ_{opt} (MPER-Sim)	0.469	0.470	0.415	0.377	0.365	
	0	PER	1.07E-01	1.30E-02	4.65E-04	1.74E-04	2.53E-05	0.163
		Δ_{opt} (MPER-Sim)	0.3914	0.4154	0.3823	0.3662	0.3445	
M=128	1	PER	1.15E-02	1.27E-03	1.45E-04	4.29E-05	6.95E-06	4.488
		Δ_{opt} (MPER-Sim)	0.461	0.453	0.404	0.382	0.368	
	1/4	PER	1.17E-02	1.29E-03	1.45E-04	4.28E-05	7.22E-06	2.380
		Δ_{opt} (MPER-Sim)	0.477	0.453	0.409	0.379	0.363	
	0	PER	8.38E-02	1.05E-02	4.45E-04	1.79E-04	2.40E-05	0.337
		Δ_{opt} (MPER-Sim)	0.3914	0.4033	0.3867	0.3541	0.3601	
		Δ_{opt} (MMSE)	1.008	1.003	1.000	1.000	1.000	
		Δ_{opt} (MMI)	0.300	0.260	0.210	0.190	0.160	
		Δopt (MPTRC1, j=2)	0.425	0.399	0.362	0.351	0.333	
		Δopt (MPTRC1, j=3)	0.387	0.368	0.340	0.332	0.318	
		Δopt (MPTRC1, j=4)	0.362	0.347	0.326	0.319	0.308	
		Δopt (MPTRC1, j=5)	0.343	0.332	0.314	0.309	0.300	

6.5. Summary

In this chapter, a new method, MPRTC1, has been proposed for optimizing the quantizer used on the channel output. The quantized channel outputs are used for joint source channel arithmetic decoding by MA. In MA, parts of the generated candidates are pruned due to limitations on memory and computational complexity. In each stage of decoding, the candidates are generated and ranked based on their metrics and only the best

M candidates are retained. The PMF of the rank of true candidate has been calculated. The probability that the true candidate is among the two candidates with the highest metrics is used as proxy measure for error rate to design the channel quantizer. This probability is calculated as a function of Δ in two steps. Firstly, it has been proven the probability that the true candidate is the best, *i.e.,* $P(RTC = 0)$, is independent of quantization parameter. Secondly, $P(RTC = 1)$ is calculated and used for quantizer optimization in various channel SNRs. In calculation of $P(RTC = 1)$, some approximations are used which are valid in the operational channel SNRs, *i.e.,* SNR>4dB. In MPRTC1, quantization parameter maximizes $P(RTC = 1)$. The simulation results have demonstrated the effectiveness of the proposed method comparing to the previous methods, MMSE and MMI. It has been shown that the result of MPRTC1 is closer to the simulation for larger SNRs due to the approximations used in MPRTC1 which are more accurate for higher SNRs.

CHAPTER 7. Conclusion

7.1. Summary

This thesis has been concerned with enhancing the error resilience of compressed videos using joint source channel decoding techniques. Residual redundancies have been exploited using JSCAD with FS to detect and correct errors. The error resilience improvement has been carried out in four phases as follow.

In the first phase, a MAP arithmetic decoder has been proposed to exploit the residual redundancy using a syntax checker for *intra* prediction modes in compressed video. Unlike previous work [24], the decoded syntax elements have been checked through development of the decoding tree in all stages. Simulation results have shown a considerable error rate reduction with the cost of higher computational complexity owing to the syntax checking process. Accordingly, a tradeoff between error rate and computational complexity has been shown. Whereas depending on the available computing resources, the decoder could check the syntax through development of the decoding tree or at the end of the decoding tree.

In the second phase, the probability of missed error detection has been proposed as a figure of merit to evaluate the FS configuration, *i.e.,* FS position in the binary arithmetic coding symbol alphabet. It has been shown that placing the FS between information carrying symbols (0 and 1) minimizes the proposed figure of merit. Simulation results have shown that optimizing the FS configuration results in the error rate reduction without imposing additional redundancy.

In the third phase, a new statistical model has been proposed to estimate *a priori* probabilities in MAP decoder to exploit residual redundancy in compressed video. In the introduced method the PMF of *intra* modes in a MB has been estimated adaptively using the syntax elements located in the spatially adjacent to MBs generated earlier in the decoding tree. the estimation has been categorized as either reliable or unreliable based on the local entropy of *intra* modes in adjacent MBs, and the decoder switches adaptively to ML in unreliable cases. The simulation results have shown that the proposed categorization prevents unreliable estimated data misleads the decoder. Furthermore, improvement in error rates comparing to ML decoder have been shown.

In the last phase of the thesis, the effect of quantizing the channel output on the performance of the MAP arithmetic decoder has been studied. Two new measures have been proposed as proxies of error rates to optimize the channel quantizer. The first measure is the second moment of rank of true candidate in the sorted candidate list in the decoding tree. Computing this measure is only feasible for short bit sequences. Accordingly, the second measure has been proposed as probability that the true candidate ranked 0 or 1. This measure has been calculated and approximated for the operating SNRs more than 4 dB. Minimizing this measure, an adaptive channel quantizer has been introduced for various channel SNRs. Simulation results have shown the effectiveness of the proposed method comparing to the other methods based on mutual information and mean square error.

7.2. Concluding remarks

Video compressors cannot entirely remove the redundancies due to a number of issues, for example: the lack of available resources like processing power, and constraints in compression standards like limited number of *intra* modes. It has been shown that the video decompressor with JSCD can use the residual redundancy for error detection and correction at the cost of higher computational complexity. The JSCD thus provides a tradeoff between four elements namely complexity of the encoder, overhead in terms of data bit rate, complexity of the decoder and error rates. For example, by discarding some *intra* prediction modes (due to some constraints in video compression standards), the encoder saves complexity in *intra* mode selection process and increases the overhead in terms of data bit rate by increasing the amount of residual redundancy. while the decoder uses this redundancy in MAP decoding to improve the error rate by performing complex checking processes. The more frequent the syntax checking, the better error correction and in turn the more computational complexity. In Chapters 3 and 6 the tradeoff has been shown by lowering the error rate of a MAP arithmetic decoder with the cost of higher computational complexity.

The MAP decoder uses two kinds of information. The first one is prior information and the second one is the data that come from channel as either soft or quantized bits. The prior information in video may be in form of interdependencies between spatially and temporally adjacent syntax elements. These interdependencies are extracted at the cost of computational complexity. The tradeoff between the computational complexity and error

rate has been shown in *a priori* probabilities estimation of *intra* modes. The more sophisticated *a priori* probability estimator the more error rate reduction (see section 5.6).

Due to practical limitations, the channel output typically passes through a quantizer. The parameters of an optimum quantizer are dependent on the channel SNR and the encoding/decoding algorithms. Since calculating the relation between the quantizer parameters and the error rate is infeasible, a measure is needed as a proxy for error rate. This proxy measure is used to design the quantizer in different channel SNRs. This measure should consider the structure of the decoding algorithm. The computational complexity of calculating this measure should be moderated especially for long packets. Therefore, this measure should be carefully chosen based on the coding type and available computational resources. It has been shown that MSE and mutual information are not good proxy measures for error rates in the JSCAD with MA algorithm (see Section 6.4). The quantizer based on the proposed measure PRTC1 provides the best performance in terms of decoder bit error rate. Furthermore, this measure can be used in suboptimum decoding algorithms where only a limited number of bit sequence candidates are preserved in developing the decoding tree due to constraints on memory and in light of the enormous number of decoding states.

In this thesis, MA has been used. It keeps the best M nodes of the decoding tree at each stage. In Chapter 3 different structures of decoding tree are represented (see Figure 3-8, Figure 3-9, Figure 3-10 and Figure 3-11). DTTL has been proposed as a measure of the decoding tree structure. High value of DTTL represents that the decoder cannot eliminate wrong path as timely fashion as required. This condition occurs either in small

amount of redundancy (small probability of FS), or in high level of noise. In addition to decoding metric, DTTL can be used to change the encoding and decoding adaptively.

7.3. Future works

This thesis has focused on outperforming the error resilience of MAP joint source channel arithmetic decoder by exploiting the residual redundancies and optimizing the channel quantizer. Furthermore, there are other studies along with the ideas developed in this thesis provided below.

- To add redundancy into the video stream intentionally without using FS, some syntax elements can be considered as forbidden ones and they would not be encoded. This strategy saves computational complexity at the encoder side. Furthermore, the structure of the arithmetic coding in the video codecs will not be changed in this method which makes it compatible with the standard. Although it imposes overhead in terms of bit rate and computational complexity at the decoder side, it improves the error resilience without changing the structure of arithmetic decoder.
- The proposed MAP decoder in Chapter 5 can use probabilistic models of other syntax elements such as motion vectors. These models provide *a priori* probability of different syntax elements as accurately as possible. The tradeoff between computational complexity of the models and error rate reduction can be studied.

- The AC with FS can be implemented more efficiently. This can be done by using lookup tables as used in the H.264 and HEVC standards. The other way is to use the standard BAC in the video standards in two steps. In the first step, an information carrying bin (0 or 1) is encoded. In the second step, one of the bins is consider as FS and the other is encoded. Using this strategy, the efficient standard BAC in video compressions can be converted to BAC with FS.
- In this thesis, the measure PRTC1 has been proposed to optimize a 4-level channel quantizer. The channel has been modeled by AWGN and the search algorithm has been MA for MAP arithmetic coding. As future work, this measure can be calculated and applied in other forms of channel and decoding metrics. The number of quantizer levels can be extended to more than 4 which results in more than 1 quantization parameters. Multivariate optimization techniques should be used to optimize such quantizers.
- DTTL can be calculated through the construction of the decoding tree, and it can be used to make the coding algorithm adaptive in two ways. First, the decoder can increase the value of M adaptively when DTTL is greater than a threshold. Second, the syntax checker can be switched on and off adaptively depending on value of DTTL. This adaptive algorithm can be implemented by slightly modifying the current MAP decoder based on MA.

References

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2016–2021," 15 September 2017. [Online]. Available: https://www.cisco.com/c/en/us/ solutions/ collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html#_Toc484813971.

[2] J. Joskowicz and R. Sotelo, "A model for video quality assessment considering packet loss for broadcast digital television coded in H.264," *International journal of digital multimedia broadcasting,* vol. 2014, no. 242531, pp. 1-11, 2014.

[3] I. E. Richardson, The H.264 advanced video compression standard, John Wiley, 2010.

[4] Y. Wang, J. Ostermann and Y.-Q. Zhang, Video processing and communications, Prentice Hall, 2001.

[5] G. J. Sullivan, J. Ohm, W. J. Han and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, no. 12, pp. 1649-1668, 2012.

[6] I. H. Witten, R. M. Neal and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM,* vol. 30, pp. 520-540, June 1987.

[7] T. M. Cover and J. A. Thomas, Elements of information theory, Hoboken, N.J. : Wiley-Interscience, 2nd ed, 2006.

[8] D. Levine, W. E. Lynch and T. Le-Ngoc, "Observations on error detection in H.264," in *proceedings 50th Midwest Symposium on Circuits and Systems*, Montreal, Canada, 5-8 Aug. 2007.

[9] D. Levin, Error Resilient Transmission of H.264 Compressed Video Using CABAC Entropy Coding, Montreal: Concordia University, Thesis, 2007.

[10] J. Chou and K. Ramchandran, "Arithmetic coding-based continuous error detection for efficient ARQ-based image transmission," *IEEE Journal on Selected Areas in Communications,* vol. 18, no. 6, pp. 861-867, June 2000.

[11] D. Xuewen, Y. Zhaoxuan, W. Jiapeng, C. Yang and Y. Yuan, "Detection and localization of transmission error in MPEG-4 video," in *International Conference on Wireless Communications, Networking and Mobile Computing*, Vancouver, Canada, Sept. 2006.

[12] P. Bansal, M. R. Narendran and M. N. K. Murali, "Improved error detection and localization techniques for MPEG-4 video," in *Proceedings international conference on image processing*, 2002.

[13] W. Kwok and H. Sun, "Multi-directional interpolation for spatial error concealment," *IEEE Transactions on Consumer Electronics,* vol. 39, no. 3, pp. 455-460, June 1993.

[14] W. E. Ryan and S. Lin, Channel codes : classical and modern, Cambridge, UK; New York: Cambridge University Press, 2009.

[15] H. Wang, L. Kondi, A. Luthra and S. Ci, 4G wireless video communications, Chichester, West Sussex, U.K.: Wiley, 2009.

[16] ITU-T, "H.264 : Advanced video coding for generic audiovisual services," Telecommunication standardization sector of ITU, April 2017.

[17] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang and T. Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Last Call)," 12th Meeting: Geneva, CH, 14–23 Jan. 2013.

[18] T. Wiegand, G. J. Sullivan, G. Bjøntegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 13, no. 7, pp. 560-576, 2003.

[19] M. Pourazad, C. Doutre, M. Azimi and P. Nasiopoulos, "HEVC: the new gold standard for video compression: how does HEVC compare with H.264/AVC?," *Consumer Electronics Magazine, IEEE,* vol. 1, no. 3, pp. 36-46, July 2012.

[20] P. Decker, "An adaptive type-II hybrid ARQ/FEC protocol suitable for GSM," in *in Proceedings of IEEE vehicular technology conference (VTC)*, Stockholm, 1994.

[21] D. W. Redmill, D. R. Bull, J. T. Chung-How and N. G. Kingsbury, "Error-resilient image and video coding for wireless communication systems," *Electronics and Communication Engineering Journal,* vol. 10, no. 4, pp. 181-190, Aug. 1998.

[22] J. R. Casas and L. Torres, "Coding of details in very low bit-rate video systems," *IEEE Transactions on circuits and systems for video technology,* vol. 4, no. 3, pp. 317-327, June 1994.

[23] S. Ben Jamaa, M. Kieffer and P. Duhamel, "Improved sequential MAP estimation of CABAC encoded data with objective adjustment of the complexity/efficiency tradeoff," *IEEE Transactions on Communications,* vol. 57, no. 7, pp. 2014-2023, July 2009.

[24] G. Sabeva, S. Ben Jamaa, S. Kieffer and P. Duhamel, "Robust decoding of H.264 encoded video transmitted over wireless channels," in *proceedings of IEEE 8th Workshop on Multimedia Signal Processing*, Victoria, Canada, 2006.

[25] P. Duhamel and M. Kieffer, Joint source-channel decoding, a cross-layer perspective with applications in video broadcasting, Academic Press, 2010.

[26] F. Caron and S. Coulombe, "A maximum likelihood approach to video error correction applied to H.264 decoding," in *proceedings 6th International Conference*

on Next Generation Mobile Applications, Services and Technologies (NGMAST), 12-14 Sep. 2012.

[27] M. Grangetto, P. Cosman and G. Olmo, "Joint source/channel coding and MAP decoding of arithmetic codes," *IEEE Transactions on Communications,* vol. 53, no. 6, pp. 1007-1016, June 2005.

[28] M. Grangetto, B. Scanavino and G. Olmo, "Joint source-channel iterative decoding of codes," in *proceeding IEEE International Conference on Communications*, 20-24 June 2004.

[29] M. Grangetto, B. Scanavino, G. Olmo and S. Benedetto, "Iterative decoding of serially concatenated arithmetic and channel codes with JPEG 2000 applications," *IEEE Transactions on Image Processing,* vol. 16, no. 6, pp. 1557-1567, June 2007.

[30] D. Levine, W. E. Lynch and T. Le-Ngoc, "Iterative joint source-channel decoding of H.264 compressed video," in *proceeding IEEE International Symposium on Circuits and Systems*, New Orleans, LA, USA, May 2007.

[31] N. Q. Nguyen, W. E. Lynch and T. Le-Ngoc., "Iterative Joint Source-Channel Decoding for H.264 video transmission using virtual checking method at source decoder," in *proceeding 23rd Canadian Conference on Electrical and Computer Engineering*, Calgary, AB, May 2010.

[32] C. Boyd, J. Cleary, I. Irvine, I. Rinsma-Melchert and I. Witten, "Integrating error detection into arithmetic coding," *IEEE Transaction on Communications,* vol. 45, no. 1, pp. 1-3, January 1997.

[33] T. Guionnet and C. Guillemot, "Soft decoding and synchronization of arithmetic codes: application to image transmission over noisy channels," *IEEE Transactions on Image Processing,* vol. 12, no. 12, pp. 1599-1609, Dec. 2003.

[34] S. J. Johnson, Iterative error correction : turbo, low-density parity-check and repeat-accumulate codes, Cambridge, UK ; New York: Cambridge University Press, 2010.

[35] M. R. Soleymani and U. V. Yingzi Gao, Turbo coding for satellite and wireless communications, Boston: Kluwer Academic Publishers, 2002.

[36] D. Bi, M. W. Hoffman and K. Sayood, Joint Source Channel Coding Using Arithmetic Codes, Morgan & Claypool, 2010.

[37] M. Grangetto and P. Cosman, "MAP decoding of arithmetic codes with a forbidden symbol," in *proceedings Advanced Concepts for Intelligent Vision Systems*, Ghent, Belgium, Sep., 2002.

[38] J. B. Anderson and S. Mohan, Source and Channel Coding: an algorithmic aproach, Norwell, MA: Kluwer Academic Publisher, 1991.

[39] A. Papoulis and S. U. Pillai, Probability, random variables, and stochastic processes, Boston: McGraw-Hill, 2002.

[40] W. Xuan, L. Jianhua, Y. Chengwei and C. Changwen, "Improved error detection ability of binary arithmetic codes," in *Proceedings of IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, August 2005.

[41] T. Spiteri and V. Buttigieg, "Arithmetic coding for joint source-channel coding," in *Proceedings of International Conference on Signal Processing and Multimedia Applications*, Athens, Greece, 2010.

[42] S. Ben Jamaa, C. Weidmann and M. Kieffer, "Analytical tools for optimizing the error correction performance of arithmetic codes," *IEEE Transactions on in Communications,* vol. 56, no. 9, pp. 1458-1468, September 2008.

[43] A. H. Murad and T. E. Fuja, "Exploiting the residual redundancy in motion estimation vectors to improve the quality of compressed video transmitted over noisy channels," in *Proceedings 1998 International Conference on Image Processing*, Chicago, IL, 1998.

[44] F. Caron and S. Coulombe, "Video error correction using soft-output and hard-output maximum likelihood decoding applied to an H.264 baseline profile," *IEEE*

Transactions on Circuits and Systems for Video Technology, vol. 7, no. 25, pp. 1161-1174, July 2015.

[45] J. G. Proakis, Digital communications, New York, NY: McGraw Hill, 2001.

[46] A. K. Jain, "Fundamentals of digital image processing," 1989.

[47] A. D. Liveris and C. N. Georghiades, "On quantization of low-density parity-check coded channel measurements," in *Proceeding global telecommunications conference*, 2003.

[48] J. Singh, O. Dabeer and U. Madhow, "Capacity of the discrete-time AWGN channel under output quantization," in *IEEE International symposium on information theory*, Toronto, ON, 2008.

[49] J. L. Massey, "Coding and modulation in digital communications," in *Procedding International Zurich Seminar Digital Communication*, Zurich, Switzerland, 1974.

[50] L. N. Lee, "On optimal soft-decision demodulation," *IEEE transaction on Information Theory,* vol. 22, no. 4, pp. 437-444, 1976.

[51] ITU-T, "Recommendation ITU-T H.265: High efficiency video coding," Telecommunication standardization sector of ITU, December 2016.

[52] V. Sze, M. Budagavi and G. J. Sullivan, High efficiency video coding (HEVC) algorithms and architectures, New York: Springer, 2014.

[53] A. Tamhankar and K. R. Rao, "An overview of H.264/MPEG-4 part 10," in *proceeding 4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications*, Zagreb, Croatia, 2003.

[54] A. Bovik, The essential guide to video processing, Academic Press, 2009, pp. 246-248.

[55] S. Milani, "Fast H.264/AVC FRExt intra coding using belief propagation," *IEEE Transactions on Image Processing,* vol. 20, no. 1, pp. 121-131, Jan. 2011.

[56] http://trace.eas.asu.edu/yuv/index.html.

[57] D. Marpe, H. Schwarz and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 7, no. 13, pp. 620-636, July 2003.

[58] C. E. Shannon, "A mathematical theory of communication," *Bell SystemTechnical Journal,* vol. 27, p. 279–423, 1948.

[59] H. Kourkchi, W. E. Lynch and M. O. Ahmad, "A measure for the missed error detection probability for optimizing the forbidden symbol configuration in joint

source-channel arithmetic codes," in *proceeding IEEE International Symposium on Circuits and Systems*, Montreal, QC, May 2016.

[60] N. Phamdo and N. Farvardin, "Optimal detection of discrete Markov sources over discrete memoryless channels—Applications to combined-source channel coding," *IEEE Transaction in Information Theory,* vol. 40, p. 186–193, 1994.

[61] D. J. Miller and M. Park, "A sequence-based approximate mmse decoder for source coding over noisy channels using discrete hidden markov models," *IEEE Transactions on Communications,* vol. 46, no. 2, pp. 222-231, Feb 1998.

[62] H. Kourkchi, W. E. Lynch and M. O. Ahmad, "A joint source channel arithmetic MAP decoder using probabilistic relations among intra modes in predictive video compression," in *proceeding IEEE International Conference on Acoustics, Speech, and Signal Processing*, Calgary, AB, April 2018.

[63] "Fraunhofer Heinrich Hertz Institute," [Online]. Available: http://iphome.hhi.de/suehring/tml/.

[64] H. Kourkchi, W. E. Lynch and M. O. Ahmad, "Improving MAP arithmetic decoding of H.264 intra modes using residual redundancy," in *Proceedings of International Conference on Digital Image Processing*, Los Angeles, USA, April 2015.

[65] Q. Lin and K. W. Wong, "Improving the error correction capability of arithmetic coding by forecasting forbidden symbols," in *IEEE International Symposium on Circuits and Systems*, Beijing, 2013.

[66] T. Guionnet and C. Guillemot, "Soft and joint source-channel decoding of quasi-arithmetic codes," *EURASIP Journal on Applied Signal Processing,* vol. 2004, pp. 393-411, March 2004.

[67] F. Lahouti and A. K. Khandani, "Efficient source decoding over memoryless noisy channels using higher order Markov models," *IEEE Transactions on Information Theory,* vol. 50, no. 9, pp. 2103-2118, September 2004.

[68] L. Z. B. Luo, "Fast intra-prediction mode selection method for h.264 video coding," in *International conference on intelligent system design and engineering application*, Changsha, 2010.

[69] J. Wang, T. Courtade, H. Shankar and R. D. Wesel, "Soft information for LDPC decoding in flash: mutual-information optimized quantization," in *IEEE Global Telecommunications Conference*, Houston, TX, USA, 2011.

[70] G. Dong, N. Xie and T. Zhang, "On the use of softdecision errorcorrection codes in NAND flash memory," *IEEE Transaction in Circuits and Systems,* vol. 5, no. 2, pp. 429-439, Feb. 2011.

[71] T. Kobayashi, S. Kametan, K. Shimizu, K. Onohara, H. Tagami and T. Mizuochi, "Soft decision LSI operating at 32 Gsample/s for LDPC FEC-based optical transmission systems," in *Conference on Optical Fiber Fommunication - Incudes Post Deadline Papers*, San Diego, CA, Mar. 2009.

[72] X. Ma, X. Zhang, H. Yu and A. Kavcic, "Optimal quantization for soft-decision decoding revisited," in *Proc. Int. Symp. Inform. Theory Appl.*, Xian, China, October 2002.

[73] B. M. Kurkoski and H. Yagi, "Quantization of binary-input discrete memoryless channels," *IEEE transactions on information theory,* vol. 60, no. 8, pp. 4544-4552, August 2014.

[74] S. Lloyd, "Least squares quantization in PCM," *IEEE Transaction on Information Theory,* vol. 28, pp. 129-137, 1982.

[75] J. Max, "Quantizing for minimum distortion," *IEEE Transaction on Information Theory,* vol. 6, pp. 7-12, 1960.

[76] J. D. Hoffman, Numerical methods for engineers and scientists, 2 ed., New York: McGraw-Hill, 1992.

[77] H. Kourkchi, W. E. Lynch and M. O. Ahmad, "Arithmetic decoding with limited-precision information," *To be submitted for publication in IET Image Processing.*

[78] R. E. Miller, Optimization: foundations and applications, New York: John Wiley and Sons, 2000.

[79] A. Gilat and V. Subramaniam, Numerical methods for engineers and scientists: an introduction with applications using MATLAB, John wiley and sons, 2011.

[80] C. De Boor, A practical guide to splines, New York: Springer-Verlag, 2001.

[81] Mathworks, 13 July 2017. [Online]. Available: https://www.mathworks. com/help/curvefit/csaps.html. [Accessed 13 July 2017].

[82] J. Dong, "Estimation of bit error rate of any digital communication system," Telecom Bretagne, Universite de Bretagne Occidentale, 2013.