University of Alberta

Improved forward error control decoding and hybrid ARQ techniques for wireless systems

by

Chunlong Bai ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Electrical and Computer Engineering

Edmonton, Alberta
Spring 2008

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-45400-8
Our file Notre référence
ISBN: 978-0-494-45400-8

Abstract

This research work addresses problems related to the application of hybrid automatic repeat request (ARQ) protocols to wireless communication systems to provide reliable data transmission. In particular, techniques that facilitate ARQ to be used in conjunction with forward error correcting (FEC) coding are investigated and proposed.

Hybrid ARQ uses a concatenated error control coding scheme that consists of an outer error detection code and an inner error correction code. These two codes are traditionally decoded separately. In this research work, the theoretical potential of, and practical methods to, improve the decoding performance of the error correction code that is used in hybrid ARQ have been investigated. One possible method is to apply the list decoding algorithm to the error correction code to generate a list of more than one codewords, and use the outer error detection code to select the correct codeword. An improved analysis for list decoding is proposed and applied to terminated convolutional codes and turbo codes. An efficient list decoding algorithm for turbo codes is proposed, and it is shown that this approach achieves better performance than the list decoding algorithms reported in the literature. A sub-block recovery scheme is proposed for turbo coded systems that use a sub-block structure, where a data block for a turbo code contains several sub-blocks, each protected by an error detection code. Analysis shows that this scheme results in improvement to sub-optimal iterative decoding. A type II hybrid ARQ scheme with incremental redundancy for systems that use turbo codes and the sub-block structure is developed, and several techniques, including sub-block recovery, built-in CRC and a frequent terminating of the turbo code are used to improve the throughput performance.

In this research work, hybrid ARQ for multiple antenna systems has also been investigated. The discussion is limited to the layered space time (LST) approach with the channel state information assumed to be available only at the receiver. A system model that is applicable to multiple ARQ processes is proposed. Based on this system model, symbol level joint detection algorithms for multiple ARQ transmission are developed. Computer simulations are performed to compare the performance of single hybrid ARQ and multiple hybrid ARQ transmissions. It is found that with linear detection, single hybrid ARQ outperforms multiple hybrid ARQ in the high signal to noise ratio (SNR) region. With the vertical Bell Labs space time (V-BLAST) architecture, multiple hybrid ARQ outperforms single hybrid ARQ.

Acknowledgements

I take this opportunity to express deep gratitude to my supervisors, Dr. Witold A. Krzymień and Dr. Ivan J. Fair, for their professional guidance and contributions in the development of my research. Their constant encouragement and constructive critique greatly helped me at every stage.

I would also like to thank my examiners, Dr. Chintha Tellambura, Dr. Vincent Gaudet, Dr. Hai Jiang and Dr. Janelle Harms for taking the time to review this work and their insightful comments and constructive suggestions on my thesis.

I would like to thank Dr. Bartosz Mielczarek for the valuable inputs in our regular technical discussions.

I would like to extend my sincere appreciation to my fellow members in the group, Dr. Yan Xin, Dr. Fengqing Zhai, Dr. Ge Li, Dr. Jia Liu, Dr. Robert Novak, Dr. David Mazzarese, Yongguang Zhu, Shuying Shen, Robert Elliott, Shreeram Sigdel and Aaron Hughes for their support.

I would like to thank the Alberta Ingenuity Fund, Informatics Circle of Research Excellence (iCORE) and TRLabs for their financial support.

Finally, I thank my family for their support.

Contents

List of Tables

List of Figures

List of Symbols

Symbol	Definition
$\mathrm{A}_{\mathcal{C}}(W,H)$	The input-redundancy weight enumerating function
$\mathrm{A}_z(H)$	the conditional weight enumerating function of the parity check bits
$\mathrm{B}_{\mathcal{C}}^{(L)}(H)$	the effective weight enumerating function (EWEF) of code $\mathcal{C}$ for L-candidate list decoding
$B_j^{(L)}$	the multiplicity of the codewords with effective weight $d_j^{(L)}$
$\mathcal{C}$	a linear block code
$\mathbf{C}_i$	code word i
$\mathbf{d}$	the vector of all possible weights of codewords in ascending order
$\mathbf{d}^{(L)}\}$	the vector of all possible effective weights of codewords for L-candidate list decoding in ascending order
$d_{\min}$	the minimum codeword weight
$d_{j,\min}^{(L)}$	the minimum effective weight for a codeword of weight d_j for L-candidate list decoding
$d_{j,\mathrm{LB}}^{(L)}$	the lower bound on the effective weight for codewords of weight d_j for L-candidate list decoding
$\Delta w_i^{(L)}$	the effective weight increment of codeword i for L-candidate list decoding
η_{SAW}	throughput
E_{0i}	pairwise error event
$\mathrm{E}_{0i}^{(L)}$	the generalized pairwise error event concerning codeword $\mathbf{C}_i$ for L-candidate list decoding
E_s	the energy per modulated symbol
Γ	the number of frames that can be transmitted during the
J	the number of distinct non-zero weights that codewords may
$J^{(L)}$	the number of all distinct effective weights that a codeword may

	assume for L-candidate list decoding
K	number of information bits in a codeword of a binary block code
L	the number of candidates in list decoding
$\mathbf{L}_{(y)}$	the log likelihood rations of the channel observations
$\mathbf{L}_{(a)}$	the *a priori* information for the systematic bits
$\mathbf{L}_{(e)}$	the extrinsic information for the systematic bits
ν	the size of memory in a binary recursive systematic convolutional code
N	number of bits in a codeword of a binary block code
$N_0/2$	two sided noise power spectral density
r	code rate of a binary block code
P_f	frame error rate
	round trip delay
$\mathrm{P}\,(\mathrm{CE})$	probability of codeword error
$\mathrm{P}(\mathrm{E}_{0i}^{(L)})$	the probability of the generalized pairwise error event $\mathrm{E}_{0i}^{(L)}$ concerning codeword $\mathbf{C}_i$ for L-candidate list decoding
$\mathrm{P}(\mathrm{E}_{0i})$	probability of the pairwise error event E_{0i}
R_{0i}	the decision error region for the pairwise error event E_{0i} in ML decoding
$\mathrm{R}_{0i}^{(L)}$	the decision error region for the generalized pairwise error event $\mathrm{E}_{0i}^{(L)}$ in list decoding
^{T}	transposition
$\mathbf{u}$	the binary input sequence to the encoder
$\hat{\mathbf{u}}$	the decoded binary sequence
$\mathbf{uv}$	the binary output sequence from the encoder
w_i	Hamming weight of code word $\mathbf{C}_i$
$w_i^{(L)}$	the effective weight of $\mathbf{C}_i$ for L-candidate list decoding
$\mathbf{x}$	the BPSK modulated symbols
$\mathbf{y}$	the received symbols
^{H}	conjugate transposition
$(\cdot)_k$	the k-th row of the matrix "$\cdot$"
$[\cdot]_k$	the k-th column of the matrix "$\cdot$"
$(\cdot)_{k,h}$	the element at the k-th row and the h-th column of the matrix "$\cdot$"
$\mathbf{I}_n$	the $n \times n$ identity matrix

Chapter 1

Introduction

1.1 Hybrid ARQ and its application in wireless communication systems

In a communication system, if there only exists a one-way channel, forward error correcting (FEC) codes can be used for error control. At the transmitter, the information bits are encoded into codewords that are transmitted through the channel. At the receiver, a decoder estimates the value of the information bits based on the received words. If appropriately designed, the amount of redundancy introduced by the encoder can ensure successful recovery of the information bits in the majority of cases. However, if the decoder fails to recover the information bits, a remedy for this erroneous transmission is not possible because of the one-way nature of the communication which results in the transmitter being unaware of the decoding failure at the receiver.

If a feedback channel is available in the communication system, automatic repeat request (ARQ) procedures can be used to increase system reliability. At the transmitter, the information bits are divided into packets, and each packet is encoded by an error detection code before transmission. The receiver can then check the integrity of the received frames, and acknowledges the latest accepted packets by sending acknowledgements (ACKs) back to the transmitter using the feedback channel. By establishing an appropriate protocol, retransmissions can be arranged upon the occurrence of a transmission error, and therefore reliable communication is possible although there is no guarantee that each individual transmission is successful.

Pure ARQ schemes

In the following discussion, it is assumed that the transmitter fetches packets from a full source queue, and that it is the objective of the receiver to deliver packets free of error in their original order to the destination.

In a pure ARQ protocol, each packet received from the queue is encoded at the transmitter by an error detection code; a cyclic redundancy check (CRC) code is commonly used for this purpose. A frame is formed with a sequence number and the coded packet. "Pure" in this instance means that only error detection is involved and no error correcting code is used.

Depending on the availability of buffers at the transmitter and the receiver, a pure ARQ scheme can use one of three basic retransmission protocols, namely, stop-and-wait (SAW), go-back-$\mathcal{N}$ (GBN) and selective-repeat (SR) [1].

If buffers are not available at either the transmitter or the receiver, the stop-and-wait protocol can be used. In stop-and-wait, the transmitter and the receiver communicate one frame at a time. The transmitter sends out a frame, starts a timer, and waits for an acknowledgement. If the transmitter does not receive an acknowledgement for this frame before the time-out expires, it retransmits the frame. Upon receiving a valid acknowledgement for this frame, the transmitter sends the next frame. Any acknowledgement that contains detected errors or any acknowledgement for a frame other than the one most recently transmitted is ignored by the transmitter. At the receiver, a frame is checked upon arrival. If an error is detected in the frame, the receiver discards the frame and does not send anything back to the transmitter. If the frame passes the error check and if it contains the expected sequence number, the receiver sends back an acknowledgement for the current frame. The acknowledgement contains the sequence number of the next frame that the receiver expects. However, if the received frame does not have the expected sequence number, the receiver discards the frame, and resends an acknowledgement for the most recently accepted frame. The stop-and-wait protocol is illustrated in Fig. 1.1, where the first transmission of frame 2 and the acknowledgement for the first transmission of frame 3 are lost in the channel.

The fact that the transmitter is idle when awaiting for the acknowledgement is a major source of inefficiency in the stop-and-wait protocol, especially when the delay-bandwidth product of the channel is high.

To improve the efficiency, an ARQ protocol can be designed to utilize the forward channel when the transmitter is waiting for the acknowledgement for a frame. To enable the transmission of subsequent frames while there are frames not yet acknowledged by the receiver, a buffer has to be made available to the transmitter to save the transmitted but not yet acknowledged frames in case they need to be retransmitted in the future.

If a buffer is only available at the transmitter, but not at the receiver, the go-back-$\mathcal{N}$

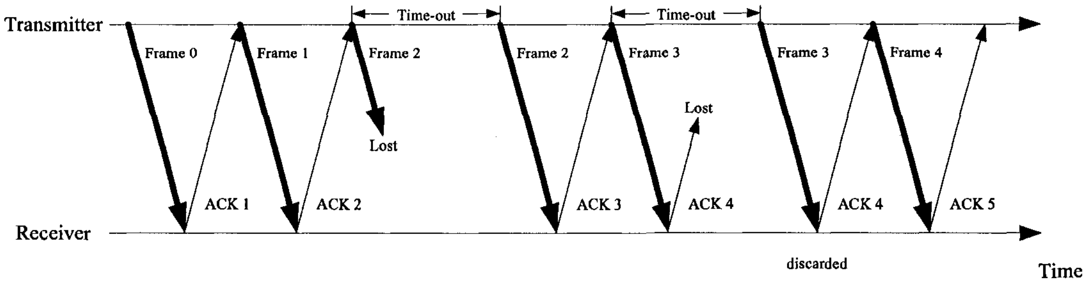


Figure 1.1: The stop-and-wait protocol.

protocol can be employed to achieve more efficient transmission compared to the stop-and-wait protocol. In the go-back-$\mathcal{N}$ protocol, a buffer that can store exactly $\mathcal{N}$ frames is assumed to be available at the transmitter. As long as there is room in the buffer for a new frame, the transmitter fetches a packet from the source queue, forms and sends out a frame, starts a timer, and saves the frame in the buffer. Therefore, the processing of multiple frames is pipelined to keep the forward channel busy. However, if there is no room in the buffer for a new frame, the transmitter does not fetch a new packet from the queue. If an error-free acknowledgement for a frame stored in the buffer is received, that frame together with any frames transmitted before that frame are cleared from the buffer. Any error-free acknowledgement for a frame not stored in the buffer is ignored. If the time-out for the oldest frame in the buffer expires, the frames in the buffer are retransmitted in their original order. In the go-back-$\mathcal{N}$ protocol, the processing at the receiver is exactly the same as that described previously for the stop-and-wait protocol.

The go-back-$\mathcal{N}$ protocol is illustrated in Fig. 1.2, where the first transmission of frame 4 and the acknowledgement for the first transmission of frame 8 are lost in the channel.

It is worth noting that, in the go-back-$\mathcal{N}$ protocol, after the receiver discards a frame that contains errors, it will also discard all the subsequently received frames that have higher sequence numbers until it receives retransmission of the frame that was corrupted in the channel. In the case of a transmission error, the loss of the transmission time in the go-back-$\mathcal{N}$ protocol is similar to that in the stop-and-wait protocol. Therefore, compared to the stop-and-wait protocol, although go-back-$\mathcal{N}$ protocol improves the efficiency when there is no error during the transmission by keeping the forward channel busy, it does not improve the efficiency in the case of transmission errors.

To further improve the efficiency, an ARQ protocol can be designed to make use of the transmission time between the original transmission and the retransmission of an unacknowledged frame. To enable the acceptance of the correctly received out-of-order frames

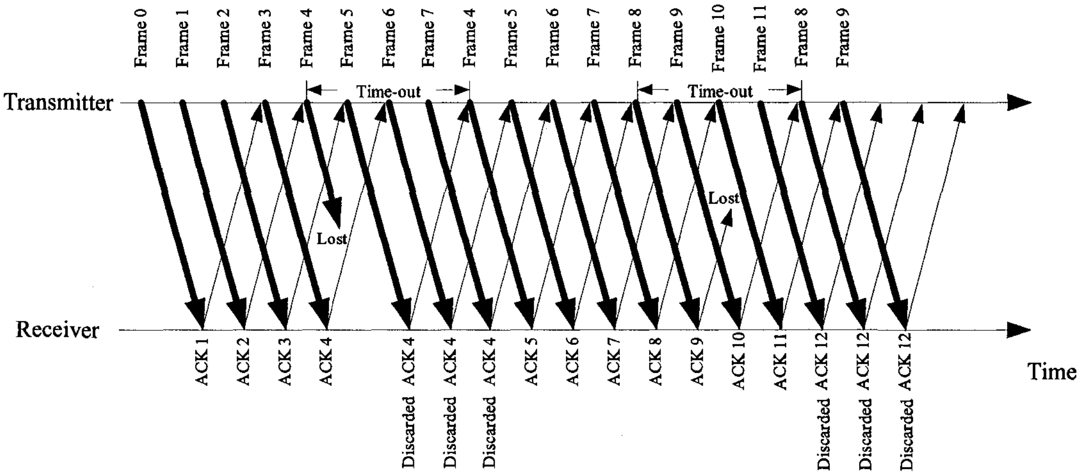


Figure 1.2: The go-back-$\mathcal{N}$ protocol. "Discarded" means that the received frame is discarded by the receiver.

transmitted following an unacknowledged frame and before the retransmission of that unacknowledged frame, a buffer has to be made available to the receiver to store those frames until the previously unacknowledged frame is correctly received.

If buffers are available at both the transmitter and the receiver, the selective-repeat protocol can be used to achieve higher efficiency compared to the go-back-$\mathcal{N}$ protocol. In the selective-repeat protocol, buffers that can store exactly $\mathcal{N}$ and $\mathcal{M}$ frames are assumed to be available at the transmitter and the receiver, respectively. As long as there is room in the transmit buffer for a new frame, the transmitter fetches a packet from the source queue, forms and sends out a frame, starts a timer, and saves the frame in the buffer. However, if there is no room in the buffer for a new frame, the transmitter does not fetch a new packet from the queue. If an error-free acknowledgement for a frame stored in the buffer is received, that frame together with any frames transmitted before that frame are cleared from the buffer. An error-free acknowledgement for a frame not stored in the buffer is ignored. If a time-out for the least recently transmitted or retransmitted frame in the buffer expires, the frame is retransmitted and the timer for that frame is reset.

The receiver can buffer up to $\mathcal{M}$ frames, and therefore expects to receive and is willing to buffer packets with sequence numbers at most $\mathcal{M}-1$ greater than that of the packet with the smallest packet number that has not yet been delivered to the destination. A frame is checked upon arrival. If error is detected in the frame, the receiver discards the frame and does not send anything back to the transmitter. If the frame passes the error check, and the frame is one of the expected frames, it is accepted and buffered. If this frame has

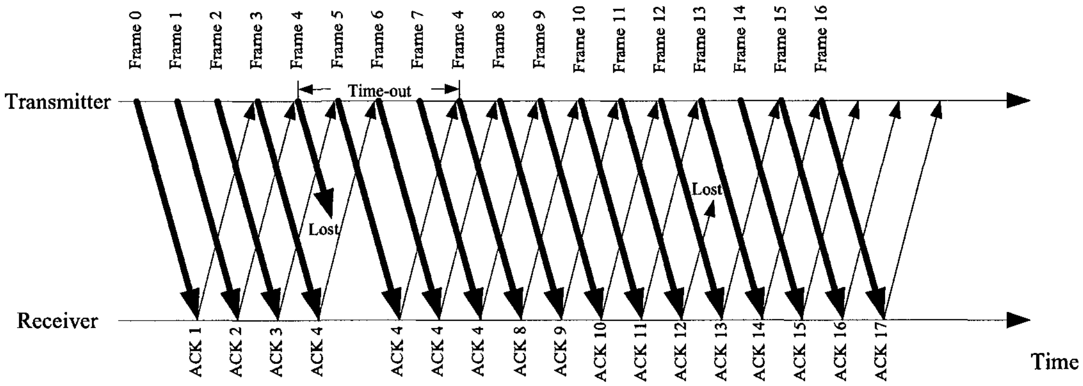


Figure 1.3: The selective-repeat protocol.

the smallest sequence number of the expected frames, an acknowledgement is transmitted for that frame, and the frame together with any accepted subsequent in-order frames are delivered to the destination and cleared from the receive buffer. Otherwise, if the frame does not have the least sequence number of the expected frames, an acknowledgement is sent with the least sequence number of the expected frames. If the frame passes the error check, but is not one of the expected frames, the frame is discarded and an acknowledgement is sent with the least sequence number of the expected frames.

The selective-repeat protocol is illustrated in Fig. 1.3, where the first transmission of frame 4 and the acknowledgement for the first transmission of frame 8 are lost in the channel.

One essential performance metric for an ARQ protocol is the throughput, η, which can be defined as the reciprocal of the average number of encoded data bits the transmitter could have sent continuously during the time required for a single user data bit to be accepted by the receiver. Suppose each packet contains K bits of user data and is encoded into a codeword of N bits by a binary error detection code of rate $r = K/N$. Let P_f denote the frame error rate; $(1 - P_f)$ is then the probability that a transmitted frame is received correctly. Let Γ be the number of frames that can be transmitted during the round trip delay of the channel. Assuming that Γ is an integer, that both $\mathcal{N}$ and $\mathcal{M}$ equal Γ for the go-back-$\mathcal{N}$ and the selective-repeat protocols, ignoring the overhead introduced by the sequence number and the acknowledgements, the throughput for the stop-and-wait, go-back-$\mathcal{N}$, and selective-repeat ARQ protocols, denoted by η_{SAW}, η_{GBN}, η_{SR}, respectively, can be expressed as [2]

$$\eta_{SAW} = \frac{(1 - P_f)r}{1 + \Gamma}, \tag{1.1}$$

$$\eta_{GBN} = \frac{(1 - P_f)r}{1 + \Gamma P_f}, \tag{1.2}$$

$$\eta_{SR} = (1 - P_f)r. \tag{1.3}$$

Equations (1.1)- (1.3) indicate that, among the three ARQ protocols, the selective-repeat protocol achieves the highest throughput, and that this throughput is determined by the frame error rate and the rate of the error detection code. The denominators in equations (1.2) and (1.3) clearly explain the sources of the inefficiency of the stop-and-wait and the go-back-$\mathcal{N}$ protocols. In the stop-and-wait protocol, the transmission of each frame is accompanied by an idle period on the forward channel equal to the duration of a round trip delay, which could have been used to transmit Γ frames. Equivalently, the transmission of each encoded bit, either correctly received or not, occupies the duration that could have been used to transmit $(1 + \Gamma)$ encoded bits. In the go-back-$\mathcal{N}$ protocol, each frame that does not encounter errors during transmission does not incur any idle period on the forward channel, however, the transmission of each frame that encounters errors, as in the stop-and-wait protocol, is still acompanied by an idle period on the forward channel with a duration equal to the round trip delay. Equivalently, the transmission of each encoded bit, on average, occupies the duration that could have been used to transmit $(1 + \Gamma P_f)$ encoded bits.

Hybrid ARQ schemes

From (1.1 - 1.3), no matter which protocol is used, an underlying problem associated with the pure ARQ scheme is that if the channel quality deteriorates, the increased frequency of retransmission requests has a severe impact on the throughput.

Hybrid ARQ, first introduced by Wozencraft and Horstein in [3], uses FEC in conjunction with error detection to improve throughput performance over the noisy channel. In a hybrid ARQ system, at the transmitter, packets are first encoded by an error detection code and then encoded by an FEC code. At the receiver, the FEC code is decoded first, and the decoded results are fed into an error detector. FEC reduces the frequency of retransmission by correcting the most frequently occurring error patterns. When FEC fails to correct an error pattern and the errors are detected by the error detection code, the receiver requests a retransmission. The hybrid ARQ scheme can provide throughput similar to that of FEC while offering reliability performance typical of ARQ schemes.

Hybrid ARQ can employ one of the previously discussed retransmission protocols: stop-and-wait, go-back-$\mathcal{N}$ or selective-repeat. Furthermore, depending on whether the failed transmission attempts are rejected or saved for further combining, hybrid ARQ can be classified as type-I hybrid ARQ and type-II hybrid ARQ [2]. [1]

In a type-I hybrid ARQ system, the noisy packet that causes the retransmission request is discarded by the receiver. During retransmission, the transmitter sends the same coded packet as during the first transmission attempt. The retransmitted packet is decoded at the receiver without combining with the previously transmitted packet.

In a type-II hybrid ARQ system, the noisy packet that caused retransmission is stored by the receiver for combining with retransmitted packets. Two basic approaches can be used to perform this combining:

a: Diversity combining: the transmitter retransmits the same packet as in the original transmission and the receiver combines the replicas before decoding.

b: Incremental redundancy (IR): upon receipt of the retransmission request, the transmitter transmits additional symbols instead of another copy of the packet it sent in the previous transmission attempt. In this approach, the transmitter usually employs a rate compatible code, e.g., a rate compatible punctured convolutional (RCPC) code [5] or a rate compatible punctured turbo (RCPT) code [6]. Rate compatible codes are from families of codes of distinct rates that satisfy the rate compatibility constraint [5]. The rate compatible constraint for a family of codes requires that all the symbols in a code are used by all codes of a lower rate. In other words, a lower rate code is constructed from all the symbols of the next higher rate code with extra parity bits. In the first transmission attempt, a codeword with the highest rate is transmitted. Upon each retransmission request, the transmitter transmits extra redundancy so that, at the receiver, a code of lower and lower rate can be constructed after combining all the packets that represent the same data.

ARQ schemes used in wireless communication systems

The stop-and-wait protocol requires less resources for signalling and memory at both the transmitter and the receiver than the go-back-$\mathcal{N}$ and the selective-repeat protocols, therefore it was adopted by the third generation (3G) wireless communication systems such as the wideband code division multiple access (WCDMA) by the 3rd Generation Partnership Project (3GPP) and the CDMA2000 by 3GPP2. However, since in the stop-and-wait protocol feedback is not instantaneous after every transmission, the transmitter must wait for an acknowledgement before transmitting the next frame. In the interim, the channel remains idle and system capacity is wasted. In a slotted system, this feedback delay will waste at

[1] In the literature, there are further classifications. For example, in [4], a type-III hybrid ARQ is defined as a special case of type-II hybrid ARQ in which the packets used for retransmission are self-decodable.

least a half of the system capacity since at least every other time slot must be idle even when the channel is error free.

To overcome this drawback, multi-channel stop-and-wait ARQ was proposed in [7,8] as an ARQ scheme that achieves the low complexity of stop-and-wait as well as the throughput efficiency of selective-repeat. In a slotted system, this approach offers a solution by paralleling multiple stop-and-wait protocols and in effect running a separate instantiation of the ARQ protocol when the other channels are idle. As a result, no system capacity is wasted because there is always one channel communicating a frame from the transmitter to the receiver. The hardware cost will not increase significantly if the multi-channel receiver is implemented as a logical representation of a single hardware receiver. In a Γ channel stop-and-wait ARQ system, the sum of the processing time of the frame at the receiver and the processing time of the acknowledgements at the transmitter must be less than $(\Gamma - 1)$ time slots in order to achieve the maximum throughput. The multiplicity Γ of parallel channels can be adjusted according to the relationship between the processing time and the length of the time slot.

ARQ schemes in MIMO systems

To satisfy the high data rate requirement of wireless communication systems under bandwidth constraints, technologies that achieve high spectrum efficiency should be employed. Multiple-input multiple-output (MIMO) systems in which multiple antennas are deployed at both the transmitter and the receiver offer a promising solution. As shown by Foschini [9] and Telatar [10], when the channel state information (CSI) is known to the receiver, MIMO systems exploit the spatial domain in order to provide considerably higher capacity than single-input single-output (SISO) systems with the same bandwidth. Denote the number of transmit and receive antennas as M_T and M_R, respectively. For a single user system, the achievable capacity of a MIMO channel increases linearly with $\min(M_T, M_R)$ when the channel exhibits rich scattering and its variations can be accurately tracked.

Two basic approaches to exploit the high degrees of freedom provided by the MIMO channel have been reported in the literature. One approach is space time coding (STC), proposed by Tarokh [11], where the MIMO channel is primarily used to provide extra diversity. Carefully designed STC can achieve full antenna diversity of order $M_T M_R$ but cannot increase the spectral efficiency. The complexity of maximum-likelihood (ML) decoding of STC is, in the worst case, exponential in $M_T M_R$, and for the special case of orthogonal space time block code (OSTBC), linear in M_T [12]. where the MIMO channel is decomposed into parallel transmission layers and separate data streams are transmitted on each layer [13]. A low complexity ordered successive interference cancellation technique can be used for decoding in LST [13]. The increase in spectral efficiency is proportional to M_T for $M_R \geq M_T$, however, the antenna diversity order is bounded by $M_R - M_T + 1$. STC and LST exemplify two extremes of MIMO techniques that can achieve full antenna diversity gain and full spectral efficiency gain, respectively. Other techniques exist to achieve tradeoffs between these two extremes [14].

When packet data is transmitted in a MIMO system, an appropriate ARQ scheme should be invoked to ensure reliable transmission. The simplest approach is to completely isolate the modulation from the error control coding so that the ARQ schemes discussed previously can be directly applied to MIMO systems. However, it is of interest to investigate how an ARQ scheme can be integrated into a MIMO system in order to achieve better performance in terms of throughput and average delay.

Several authors have addressed this problem. When a layered space time scheme is used in a MIMO system, independent ARQ loops can be designed for each layer so that failure on one layer will not affect correct transmissions on other layers. In [15], Zheng et al. demonstrated that by employing independent ARQ loops for each layer, an improvement in throughput performance can be achieved in a frequency flat fading channel. Alternatively, an increase in diversity can be created by appropriate design of the retransmission scheme, e.g., by code reassignment and antenna permutation [16] or basis hopping [17]. These techniques are especially useful when the fading is slow or correlated.

1.2 Turbo codes and their application in wireless communication systems

Shannon's noisy channel coding theorem established the fundamental limit that error control coding can achieve in additive white Gaussian noise (AWGN) channels [18]. The proof of the theorem suggests three characteristics of codes that might approach the theoretical limit: a large code block length, random codebook encoding, and ML decoding. However, with a limited computational power, these characteristics are somewhat conflicting. For example, if ML decoding is implemented by exhaustive search, then decoding complexity increases exponentially as the code block length increases. On the other hand, the complexity of ML decoding can be reduced by dividing the ML decoding of an entire codeword into a series of individual and preferably independent decoding steps that involve only a portion of the codeword with reasonable computational complexity. This usually requires structure to be imposed on the code construction. However, the more structure that is applied, the less likely it is that the codebook looks random.

The turbo code is a solution that achieves a good trade-off among the desired code characteristics. The encoder of a turbo code contains two or more constituent encoders that have a relatively simple structure and are concatenated in parallel or in serial through one or more interleavers. This encoder structure enables a significantly large block length with a high level of codebook randomness with low encoding complexity which is difficult to achieve with other encoding structures [19].

Instead of ML decoding, a decoder for turbo codes usually uses sub-optimal iterative decoding with constituent decoders connected by interleavers and deinterleavers. The constituent decoders use soft-input soft-output decoding algorithms that take in channel outputs and a priori information, and output extrinsic information. During each iteration, the constituent decoders decode the constituent codes locally and exchange extrinsic information converging on a decoded sequence.

The weight spectrum of a turbo code is usually characterized by low multiplicity of low weight codewords, because of the interleaver in the turbo encoder which introduces a phenomenon called "spectral thinning" [20]. The bit error rate (BER) and frame error rate (FER) performance of a turbo code is characterized by a steep drop in the low signal to noise ratio (SNR) region, referred to as the "turbo cliff" region, and a flat performance as SNR further increases, referred to as the "error floor" region. Although the error floor causes the performance of some turbo codes to be inferior to other codes at high SNR, a well designed turbo code has an error floor that is low enough to satisfy the performance requirement for the targeted application, therefore, turbo codes are attractive for applications in power limited environments [21].

The concatenated structure of turbo codes indicates that the code block length is largely independent of the structure of the constituent codes and is determined by the length of the interleaver(s). Therefore, without changing the constituent encoders and decoders, a flexible block length can be achieved by carefully designing the interleaving algorithm so that interleaveres of the desired lengths all result in random codebooks. The flexibility to support a variable input data block length at a fixed code rate with the same encoder, among other features, makes turbo code an attractive candidate for wireless communication systems that provide a variety of services at different bit rates.

Only a few years after its invention, the turbo code with parallel concatenation was adopted by 3GPP and 3GPP2 for the WCDMA and CDMA2000 systems, respectively.

Throughout this thesis, the turbo code that is specified by 3GPP for the WCDMA system is considered, although the results obtained are in general applicable to other turbo codes. In the rest of the thesis, therefore, the term *the* turbo code refers to the WCDMA turbo code whose encoder consists of two identical $(1, 15/13)_8$ 8-state recursive systematic convolutional (RSC) constituent encoders concatenated by the prime interleaver [22]. The input sequence is encoded by one constituent encoder and its interleaved version is encoded in parallel by the other constituent encoder. Both constituent RSC codes are terminated to the all-zero state in the manner specified in [22]. The systematic bits, the parity output of the first component encoder, the parity output of the second component encoder and the terminating bits form the turbo codeword.

1.3 Objectives and motivations

The objectives of this research work are:

- Firstly, to address problems related to the application of hybrid ARQ to wireless communication systems to provide reliable data transmission, and
- Secondly, to investigate and propose techniques that facilitate ARQ to be used in conjunction with forward error correcting coding for packet data transmission.

In order to achieve these objectives, the following approaches are taken:

- Techniques are developed to improve the decoding performance of forward error control codes in a packet data transmission context;
- The use of multiple ARQ processes and a single ARQ process for spatial multiplexing (SM) MIMO systems is studied.

The research work on improving the decoding performance of forward error control codes is motivated by the following facts:

- With hybrid-ARQ, a concatenated error control coding scheme is inherently used, as shown in Fig. 1.4. At the transmitter, the packet is first encoded by an error detection code and then by an error correcting code. At the receiver, traditionally, these two codes are decoded separately. The inner error correcting code is decoded without utilizing the redundancy introduced by the outer code.
- A sub-block structure has been proposed for packet data transmission where turbo codes are involved. This structure is based on the fact that turbo codes generally

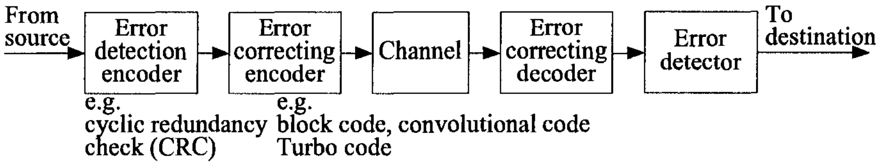


Figure 1.4: Concatenated error control coding scheme.

achieve better performance with large data blocks and in ARQ, a small retransmission block size helps to achieve good throughput performance. Therefore, in a sub-block structure, a data block for turbo coding consists of several sub-blocks, each sub-block corresponds to a packet encoded by an error detection code. In this case, to make each sub-block self-error-detectable, a significant number of parity bits are introduced for error detection.

- One of the features of turbo codes is their low multiplicity of low weight codewords.

To improve the decoding performance of the forward error control correction codes used in hybrid-ARQ schemes for packet data transmission systems, several methods to perform sub-optimal joint decoding of this concatenated code are investigated. The key idea is to utilize the redundancy introduced by the error detection code in decoding the error correcting code.

In high data rate cellular packet transmission systems, packets are transmitted in short bursts. The duration of the bursts is usually designed to be shorter than the coherence time of the channel, therefore, each packet experiences block fading in that the fading coefficient does not change within the transmission of each packet but changes from packet to packet [23–25]. The appropriate channel model during reception of any one packet is therefore an AWGN channel with a certain SNR. For example, if carrier frequency in our system is 2 GHz and packet time slot duration is 1 ms, then packets transmitted to and from mobiles moving at speeds up to about 27 m/s = 96 km/h will experience a constant gain channel within any given slot, and hence the transmission errors within a packet will be caused only by noise and interference, jointly modelled as AWGN at a constant SNR specific for the particular slot. Based on this observation, when investigating the techniques that improve the decoding performance of forward error control codes, our analysis and simulations assume an AWGN channel. In practical systems, scheduling algorithms are used to determine to which users packets should be transmitted. These are usually the users with the best channel conditions (the largest SNR within the current packet time slot).

1.4 Organization of this thesis

In Chapter 2, an improved analysis of list decoding for linear block codes is presented. The error control scheme used in communication systems that employ hybrid ARQ can be modelled as a concatenation of an inner error correcting code and an outer error detecting code. Traditionally, the inner code and the outer code are decoded separately where the inner code is decoded without utilizing the redundancy introduced by the outer code and the outer code is used purely for error detection. One approach to improve the decoding performance of this error control scheme is to use a list decoder for the inner code to generate a list of more than one codeword candidates, from which the outer code chooses the correct codeword. Assuming that the outer error detection code does not give any false detection, nor missed detection, decoding is erroneous only if the transmitted codeword is not included in the list generated by the list decoder for the inner code, where the false detection refers to the situation that the decoder indicates erroneously that the received codeword contains an error and the missed detection refers to the situation that the decoder indicates erroneously that the received codeword is correct.

Probability of codeword error analysis for a linear block code with list decoding is typically based on a "worst case" lower bound on the effective weights of codewords for list decoding evaluated from the weight enumerating function of the code. In Chapter 2, the concepts of generalized pairwise error event and effective weight enumerating function are proposed for evaluation of the probability of codeword error in list decoding of linear block codes. Geometrical analysis shows that the effective Euclidean distances are not necessarily as low as those predicted by the lower bound. An approach to evaluate the effective weight enumerating function of a particular code with list decoding is proposed. The effective Euclidean distances for decisions in each pairwise error event are evaluated taking into consideration the actual Hamming distance relationships between codewords, which relaxes the pessimistic assumptions upon which the traditional lower bound analysis is based. Using the effective weight enumerating function, a more accurate approximation is developed for the probability of codeword error with list decoding. The proposed approach is applied to codes of practical interest, including terminated convolutional codes and turbo codes with parallel concatenation.

The focus of Chapter 3 is sub-optimal list decoding algorithms for turbo codes. First, error events and weight spectra for convolutional codes and turbo codes are analyzed with emphasis on their effects on list decoding. The result of this analysis is used to explain the inefficiency of list decoding algorithms that do not generate the list directly for the turbo codes but instead generate lists for the component codes. It is also shown that these list decoding algorithms face a trade-off between complexity and performance in iterative decoding for turbo codes because better performance can be achieved with symbol-wise constituent decoders than with sequence-wise decoders, but optimal list decoding for constituent codes can only be efficiently incorporated with the sequence-wise decoding algorithm. In this chapter, a new sub-optimal list decoding algorithm is introduced for turbo codes that generates the list directly for the turbo code and can be efficiently incorporated into symbol-wise decoding algorithms for the component codes. The additional complexity of the new algorithm is low and does not depend on the complexity of the component code. Simulation results show that the new algorithm can lower the frame error floor by more than one order of magnitude compared to the algorithms reported in the literature.

In Chapter 4, a sub-block recovery scheme is proposed for a turbo coded system that uses the sub-block structure. When the sub-block structure is used together with concatenated error control coding, each data block input to the inner encoder consists of several sub-blocks, and each of these sub-blocks is protected with the error detection code. The sub-block structure is used in the WCDMA system specified by the 3GPP. In this chapter, a sub-block recovery scheme is proposed for this concatenated error control coding scheme in order to utilize the error detection capability introduced by the outer code in the decoding of the inner code. It is demonstrated that if the inner code is a turbo code with a highly structured interleaver and iterative sub-optimal decoding is used, the sub-block recovery scheme is helpful in correcting typical error patterns, therefore it helps to improve the block error rate performance. It is also shown that if the maximum likelihood decoding algorithm is used for a systematic code, as in the case when a convolutional code is decoded using the Viterbi algorithm, the sub-block recovery scheme does not introduce performance improvement. These results are confirmed by computer simulations.

In Chapter 5, a type-II hybrid ARQ scheme with incremental redundancy is proposed for a turbo coded system that uses the sub-block structure. Sub-block recovery is applied to the decoder and a method to dynamically select the bits for retransmission according to the knowledge of the decoding status is proposed. The new scheme benefits from both frequent termination and the built-in CRC of the constituent recursive systematic convolutional codes of the turbo encoder. Simulations show that for a turbo coded system that employs the sub-block structure, the new type-II hybrid ARQ scheme outperforms both type-I hybrid ARQ with sub-block recovery and the traditional type-II hybrid ARQ scheme without sub-block recovery on both the AWGN channel and the flat Rayleigh fading channel.

In Chapter 6, the hybrid ARQ schemes for spatial multiplexing MIMO systems with only channel state information at the receiver (CSIR) are investigated. In particular, the multiple hybrid ARQ scheme and the single hybrid ARQ scheme with repetition are compared. A system model for symbol detection with multiple hybrid ARQ processes is proposed, and joint detection algorithms and separate detection algorithms for both multiple hybrid ARQ and single hybrid ARQ are discussed. Simulation results show that with linear detection, single hybrid ARQ outperforms multiple hybrid ARQ in the high SNR region. With the vertical Bell Labs space time (V-BLAST) architecture, multiple hybrid ARQ always outperforms single hybrid ARQ, and joint detection always outperforms separate detection.

In Chapter 7, the contributions of this research work are summarized and the directions for future works are outlined.

Chapter 2

Improved Analysis of List Decoding and Its Application to Convolutional Codes and Turbo Codes [1]

List decoding, introduced independently by Elias in 1957 [26] and Wozencraft in 1958 [27], signifies a class of decoding algorithms used in error control systems employing block codes. Based on the received noisy sequence, the decoder outputs a list of L possible transmitted messages. A decoding error occurs if the transmitted message is not included in the list. In other words, instead of outputting an estimate of the transmitted message, the decoder determines a range in which the transmitted codeword lies. The decoder reduces the ambiguity about the transmitted message, but does not completely eliminate this ambiguity. List decoding was originally used as a conceptual decoding algorithm to explore the capacity of communication channels.

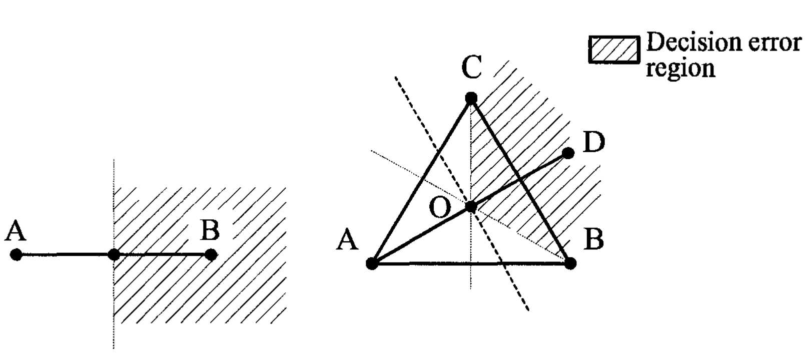
In 1994, Seshadri and Sundberg [28, 29] first proposed the use of list decoding in a concatenated error control system, where the outer code is an error detecting code and the inner code is an error correcting code. This form of code concatenation is used in a wide range of digital communication systems, especially in wireless communication systems. In this concatenated error control system, if the inner decoder can produce a list of the most likely transmitted messages, the outer error detection code can be used to determine the correct codeword from the candidate codeword list. In particular, Seshadri and Sundberg considered the case, in which the terminated convolutional code is used as the inner code. In [28, 29], Seshadri and Sundberg generalized the classical Viterbi algorithm (VA) to the

[1] A version of this chapter was presented in part in "Improved analysis of list decoding and its application to convolutional codes," at the *IEEE Globecom 2005*, St. Louis, USA, November/December 2005, and in *IEEE Transactions on Information Theory* vol. 53, no. 2, pp. 615-627, February 2007.

list Viterbi algorithm (LVA), which produces a rank ordered list of more than one globally best candidates. Using an efficient implementation of the list Viterbi algorithm, such as the parallel list Viterbi algorithm (PLVA) or the serial list Viterbi algorithm (SLVA), a list of any size can be generated with linearly increasing complexity [28,29]. In [28], Seshadri and Sundberg proposed a geometrical framework to analyze the asymptotic performance of a terminated convolutional code with list decoding on the AWGN and Rayleigh fading channels; their approach applies, in general, to all linear block codes. On the AWGN channel, the probability of codeword error is approximately determined by the minimum Euclidean distance from the decision error region to the transmitted codeword in the signal space. In [28], Seshadri and Sundberg gave a lower bound on the minimum Euclidean distance of a linear block code with list decoding. This corresponds to the case where there exist $L+1$ codewords with minimum Hamming weight and minimum Hamming distances from each other, forming a simplex in the signal space. Since then, research work regarding the performance evaluation for list decoding applied to concatenated error control systems has focused on the lower bound on the so-called effective Euclidean distance. In [30], Narayanan and Stüber generalized the lower bound on effective Euclidean distance given in [28] to cases where codewords have non-equal Euclidean distances from each other. In [31, 32], Leanderson and Sundberg further extended the lower bound to take into consideration even more distant codewords.

The general problem with the approach based on the lower bound on effective Euclidean distance is that the results are derived based on the most pessimistic situation of Hamming distance relationships among codewords that might not even exist in a given code. In this chapter, a method is proposed to evaluate the actual effective Euclidean distances for codewords in a given code when list decoding is applied, and based on analysis, a more accurate approximation to the performance of the code with list decoding is presented. The concept of overlapping for low weight codewords of a code is introduced and explain its influence on the effective weight for list decoding.

In Section 2.1, the concept of the generalized pairwise error event is introduced. Although similar concepts were implicitly used in [28,30,32], in this chapter, an explicit definition for this concept will be given. In Section 2.2, lower bound analysis on effective weight is extended to gain a better understanding of the function of list decoding. In Section 2.3, a novel method to evaluate the actual effective weight of a code is proposed. In Section 2.4, the concept of overlapping for low weight codewords is introduced. In Section 2.5 and 2.6, the proposed analysis method is applied to convolutional codes and turbo codes, followed



(a) ML decoding

(b) $L = 2$ candidate list decoding

Figure 2.1: Decision error region. A is the transmitted codeword C_0, B is the codeword C_i, C is an arbitrary codeword C_C, and D is the virtual codeword.

by analysis examples and simulation results in Section 2.7. The chapter is summarized in Section 2.8.

2.1 Improved analysis of list decoding

Consider an (N, K) binary linear block code $\mathcal{C}$ with its codewords ordered by their Hamming weights in a non-decreasing manner. Denote a codeword and its Hamming weight respectively by $\mathbf{C}_i$ and w_i, $i = 0, 1, \cdots, (2^K - 1)$, where $\mathbf{C}_0$ is the all-zero codeword. Let $\mathbf{d} = \{d_0, d_1, \cdots, d_J\}$ denote the vector of all possible weights of codewords in ascending order, where J is the number of distinct non-zero weights that codewords may assume. Moreover, $d_0 = 0$ and $d_1 = d_{\min}$ is the minimum codeword weight of the code $\mathcal{C}^2$. Let B_j denote the multiplicity of codewords with weight d_j. Without loss of generality, transmission of the all-zero codeword is assumed throughout this chapter.

Let E_{0i}, $i \neq 0$, denote the *pairwise error event* that occurs when a decision is made at the decoder in favor of $\mathbf{C}_i$ when $\mathbf{C}_0$ was transmitted. A simple example for this event under ML decoding is shown geometrically in Fig. 2.1(a), where a pairwise error event occurs when the received sequence falls in the shaded region. Assuming binary phase shift keying (BPSK) modulation, an AWGN channel with two sided noise power spectral density $N_0/2$, and ML soft decision decoding, the probability of the pairwise error event E_{0i}, where $\mathbf{C}_i$ has weight d_j, is $\mathrm{P}(\mathrm{E}_{0i}) = \mathrm{P}_{d_j} = \mathrm{Q}\left(\sqrt{2d_jE_s/N_0}\right)$, where E_s is the energy per modulated symbol and the $\mathrm{Q}(x)$ function is defined as $\mathrm{Q}(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}}\exp(-\frac{x^2}{2})dx$ [33].

In this chapter, the list decoding algorithm that generates the L most likely transmitted

[2] Here both w_i and d_j are defined because in the actual effective codeword weight analysis presented later in this chapter, codewords of the same weight are considered individually and their actual effective codeword weights are distinguished.

codewords in the ML sense is referred to as L-candidate list decoding. Define the *generalized pairwise error event* $\mathrm{E}_{0i}^{(L)}$, $i \neq 0$, for L-candidate list decoding as follows: when, based on the received sequence, the decoder outputs the L most likely (in the ML sense) transmitted codewords, $\mathbf{C}_0$ is not in the output codeword list, and $\mathbf{C}_i$ is the highest ranked solution. An example of a generalized pairwise error event is shown in Fig. 2.1(b), and occurs when the received sequence falls in the shaded region.

By definition, any element ζ in the event $\mathrm{E}_{0i}^{(L+1)}$ is an element in the event $\mathrm{E}_{0i}^{(L)}$, i.e., $\forall\zeta \in \mathrm{E}_{0i}^{(L+1)}, \zeta \in \mathrm{E}_{0i}^{(L)} \Leftrightarrow \mathrm{E}_{0i}^{(L+1)} \subseteq \mathrm{E}_{0i}^{(L)} \Rightarrow \mathrm{P}(\mathrm{E}_{0i}^{(L+1)}) \leq \mathrm{P}(\mathrm{E}_{0i}^{(L)})$. The decrease of the generalized pairwise error probability associated with a certain codeword as L increases can be interpreted as an increase in the effective Euclidean distance in decision region or, correspondingly, as an increase in the effective weight of the codeword $\mathbf{C}_i$.

Define the *effective Euclidean distance* of $\mathbf{C}_i$ for L-candidate list decoding as twice the minimum Euclidean distance from $\mathbf{C}_0$ to the decision error region for the generalized pairwise error event $\mathrm{E}_{0i}^{(L)}$. The weight of a virtual codeword[3] that has Euclidean distance that equals the effective Euclidean distance from $\mathbf{C}_0$ in the signal space is called the *effective weight* of $\mathbf{C}_i$ for L-candidate list decoding, denoted as $w_i^{(L)}$, and equals the square of the effective Euclidean distance normalized by $4E_s$. Define the *effective weight increment* $\Delta w_i^{(L)}$ of a codeword for L-candidate list decoding as the difference between its effective weight for L-candidate list decoding and its weight, i.e., $\Delta w_i^{(L)} = w_i^{(L)} - w_i$. Let $\mathbf{d}^{(L)} = \{d_0^{(L)}, d_1^{(L)}, \cdots, d_{J^{(L)}}^{(L)}\}$ denote the vector of all possible effective weights of codewords for L-candidate list decoding in an ascending order, where $J^{(L)}$ is the number of all distinct effective weights that a codeword may assume for L-candidate list decoding. Note that $d_0^{(L)} = 0$, the effective weights are no longer confined to integer values, and the codewords of the same weight may have different effective weights for L-candidate list decoding.

Let $B_j^{(L)}$ denote the multiplicity of the codewords with effective weight $d_j^{(L)}$. Define the effective weight enumerating function (EWEF) as

$$\mathrm{B}_{\mathcal{C}}^{(L)}(H) = 1 + \sum_{j=1}^{J^{(L)}} B_j^{(L)} H^{d_j^{(L)}} \tag{2.1}$$

where H is an indeterminate. In the AWGN channel, the probability of the generalized pairwise error event $\mathrm{E}_{0i}^{(L)}$ concerning codeword $\mathbf{C}_i$ with weight d_j for L-candidate list decoding can be approximately expressed as $\mathrm{P}(\mathrm{E}_{0i}^{(L)}) \simeq \mathrm{P}_{d_j^{(L)}} = \mathrm{Q}\left(\sqrt{2d_j^{(L)}E_s/N_0}\right)$. With L-candidate list decoding, the event of a decoding error is the union of these generalized

[3] Because the sequence that satisfies the weight property as specified below might not be a codeword, it is called a virtual codeword.

pairwise error events. Using the union bound, the probability of codeword error for a code with L-candidate list decoding can be approximated as

$$\mathrm{P}\,(\mathrm{CE}) = \mathrm{P}\left(\bigcup_{i=1}^{2^K-1} \mathrm{E}_{0i}^{(L)}\right) \leq \sum_{i=1}^{2^K-1} \mathrm{P}(\mathrm{E}_{0i}^{(L)}) = \sum_{j=1}^{J^{(L)}} B_j^{(L)} \mathrm{P}_{d_j^{(L)}} \tag{2.2}$$

where "CE" in P(CE) denotes a codeword error.

Let $d_{j,\min}^{(L)}$ denote the minimum effective weight for a codeword of weight d_j,

$$d_{j,\min}^{(L)} = \min_{w_i = d_j} w_i^{(L)}. \tag{2.3}$$

This quantity can serve as a lower bound on effective weight for codewords of weight d_j for L-candidate list decoding. Therefore, (2.1), (2.2) can be approximated by

$$\mathrm{B}_C^{(L)}(H) \;\approx\; 1 + \sum_{j=1}^{J} B_j H^{d_{j,\min}^{(L)}}, \tag{2.4}$$

$$\mathrm{P(CE)} \;\leq\; \sum_{j=1}^{J} B_j \mathrm{P}_{d_{j,\min}^{(L)}} \tag{2.5}$$

where B_j is the multiplicity of the codewords with weight d_j.

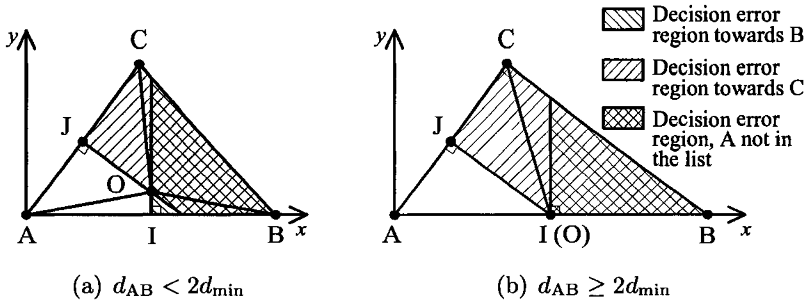
Let $d_{j,\mathrm{LB}}^{(L)}$ denote a lower bound on the effective weight for codewords of weight d_j for L-candidate list decoding as given by (30)-(31) in [32]. If $d_{j,\min}^{(L)}$ in (2.5) is replaced by $d_{j,\mathrm{LB}}^{(L)}$, the bound (2.5) becomes identical to that given in [32]. However, it will be shown later in this chapter that $d_{j,\min}^{(L)}$ can be significantly larger than $d_{j,\mathrm{LB}}^{(L)}$.

2.2 Extended lower bound analysis

In this section, the analysis based on the lower bound on effective Euclidean distance given in [32] is extended. Consider the generalized pairwise error event $\mathrm{E}_{0i}^{(L)}$, $i \neq 0$.

- Let A[4] denote the transmitted all-zero codeword $\mathbf{C}_0$;
- Let B denote the codeword $\mathbf{C}_i$, and let $d_j = w_i$;
- Let C, D, and E denote arbitrary codewords, $\mathbf{C}_\mathrm{C}$, $\mathbf{C}_\mathrm{D}$ and $\mathbf{C}_\mathrm{E}$, other than $\mathbf{C}_0$ and $\mathbf{C}_i$;
- Let d_AB and $D_\mathrm{AB} = \sqrt{4 d_\mathrm{AB} E_s}$ denote the Hamming distance and the Euclidean distance between A and B, respectively.

[4] The bold capital letters are used to denote codewords and regular capital letters to denote their corresponding signal points in Euclidean space.


y
C
J
O
A
I
B
x
y
C
J
A
I (O)
B
x
▧ Decision error region towards B
▨ Decision error region towards C
▩ Decision error region, A not in the list

(a) $d_{\mathrm{AB}} < 2d_{\min}$

(b) $d_{\mathrm{AB}} \geq 2d_{\min}$

Figure 2.2: Effective weight evaluation for $L = 2$ candidate list decoding.

$L = 2$

For $L = 2$ candidate list decoding, three codewords, $\mathbf{C}_0$, $\mathbf{C}_i$, and $\mathbf{C}_{\mathrm{C}}$, denoted by A, B, and C, respectively, are involved in effective weight evaluations. The Hamming distance relationships of these codewords in the most pessimistic situation, where the lower bound is achieved, are given by (30)-(31) in [32] as: $d_{\mathrm{AB}} = d_j \geq d_{\min}$; $d_{\mathrm{AC}} = d_{\min}$; $d_{\mathrm{BC}} = \max(d_j - d_{\min}, d_{\min})$.

If $d_{\mathrm{AB}} < 2d_{\min}$, ABC forms an acute triangle in the signal space, as shown in Fig. 2.2(a). The effective weight can be evaluated geometrically and is given by (14) in [28]. If $d_{\mathrm{AB}} \geq 2d_{\min}$, $D^2_{\mathrm{AB}} = D^2_{\mathrm{AC}} + D^2_{\mathrm{BC}}$, and ACB forms a right triangle as shown in Fig. 2.2(b), where the midpoint of the line section AB has the same distances from A, B, and C. Based on this analysis, the lower bound on effective weight for $L = 2$ candidate list decoding can be expressed as

$$w_i^{(2)} = d_{j,\mathrm{LB}}^{(2)} = \begin{cases} d_j & \text{if } d_j \geq 2d_{\min} \\ (\frac{4d_{\min}}{4d_{\min} - d_j})d_{\min} & \text{if } d_{\min} \leq d_j < 2d_{\min} \end{cases}. \tag{2.6}$$

$L = 3$

For $L = 3$ candidate list decoding, four codewords, $\mathbf{C}_0$, $\mathbf{C}_i$, $\mathbf{C}_{\mathrm{C}}$, and $\mathbf{C}_{\mathrm{D}}$, denoted by A, B, C, and D, respectively, are involved in effective weight evaluations. The Hamming distance relationships of these codewords in the most pessimistic situation are given by (30)-(31) in [32] as: $d_{\mathrm{AB}} = d_j \geq d_{\min}$; $d_{\mathrm{AC}} = d_{\mathrm{AD}} = d_{\mathrm{CD}} = d_{\min}$; $d_{\mathrm{BC}} = d_{\mathrm{BD}} = \max(d_j - d_{\min}, d_{\min})$.

If $d_{\mathrm{AB}} \leq 2d_{\min}$, all Hamming distances between A, B, C, and D except $d_{\mathrm{AB}} = d_j$ equal $d_{\min}$. The Euclidean distance relationships of A, B, C, and D in the signal space can be visualized by a tetrahedron in 3D Cartesian coordinates. The effective Euclidean distance evaluation for this special case is discussed in Appendix A. If $d_{\mathrm{AB}} > 2d_{\min}$, ACB and ADB form right triangles in the signal space, respectively, and the midpoint of line section AB has equal distances from A, B, C, and D. Based on this analysis, the lower bound on effective weight for $L = 3$ candidate list decoding can be expressed as

$$w_i^{(3)} = d_{j,\mathrm{LB}}^{(3)} = \begin{cases} d_j & \text{if } d_j > 2d_{\min} \\ (\frac{4d_{\min}-d_j}{3d_{\min}-d_j})d_{\min} & \text{if } d_{\min} \le d_j \le 2d_{\min} \end{cases} . \tag{2.7}$$

$L \geq 4$

For $L \geq 4$ candidate list decoding, more than four codewords, $\mathbf{C}_0$, $\mathbf{C}_i$, $\mathbf{C}_\mathrm{C}$, $\mathbf{C}_\mathrm{D}$, $\mathbf{C}_\mathrm{E}$, $\cdots$, denoted by A, B, C, D, E, $\cdots$, respectively, are involved in effective weight evaluations. The Hamming distance relationships of these codewords in the most pessimistic situation is given by (30)-(31) in [32] as: $d_\mathrm{AB} = d_j \geq d_{\min}$; $d_\mathrm{AC} = d_\mathrm{AD} = d_\mathrm{CD} = d_\mathrm{AE} = d_\mathrm{CE} = d_\mathrm{DE} = \cdots = d_{\min}$; $d_\mathrm{BC} = d_\mathrm{BD} = d_\mathrm{BE} = \cdots = \max(d_j - d_{\min}, d_{\min})$.

If $d_\mathrm{AB} > 2d_{\min}$, ABC, ABD, ABE $\cdots$ form right triangles in the signal space, respectively. The midpoint of line section AB has equal distances from A, B, C, D, E, $\cdots$. This means that also in this case, $d_{j,\mathrm{LB}}^{(L)} = d_j$. Unfortunately, the expression for the lower bound on effective weight for $L \geq 4$ candidate list decoding for the cases where $d_\mathrm{AB} < 2d_{\min}$ is difficult to obtain.

Summary

From the above analysis, the following results based on the assumptions in the lower bound analysis are obtained:

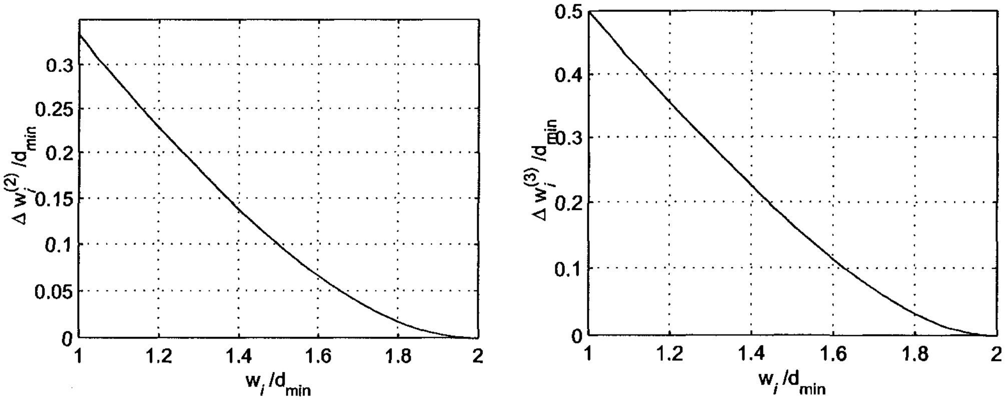
- For codewords with weight $d_j \geq 2d_{\min}$, the effective weight increment $\Delta w_i^{(L)} = w_i^{(L)} - w_i = d_{j,\mathrm{LB}}^{(L)} - d_j = 0$. Based on the lower bound on effective Euclidean distance, list decoding does not introduce improvement on the lower bound on effective weight compared to d_j;

- For $L = 2$ and $L = 3$ candidate list decoding, the effective weight increment $\Delta w_i^{(L)} = w_i^{(L)} - w_i = d_{j,\mathrm{LB}}^{(L)} - d_j$ decreases with increasing Hamming weight of the codeword in the range of $[d_{\min}, 2d_{\min}]$, as shown in Fig. 2.3;

- It is conjectured that for $L \geq 4$ candidate list decoding, $\Delta w_i^{(L)}$ should be a decreasing function of w_i in the range of $[d_{\min}, 2d_{\min}]$.

2.3 Actual effective codeword weight evaluation

In this section, a method is proposed to evaluate $w_i^{(L)}$. Denote the decision error region for the pairwise error event E_{0i} in ML decoding as R_{0i}, and the generalized pairwise error



(a) $L = 2$

(b) $L = 3$

Figure 2.3: Normalized effective weight increment as a function of normalized codeword weight obtained from lower bound on effective weight.

event $\mathrm{E}_{0i}^{(L)}$ in list decoding as $\mathrm{R}_{0i}^{(L)}$. In the decision error region $\mathrm{R}_{0i}^{(L)}$, the Euclidean distance from the signal point to codeword $\mathbf{C}_0$ is greater than the Euclidean distances to at least L other codewords including the highest ranked codeword $\mathbf{C}_i$. Therefore, $\mathrm{R}_{0i}^{(L)}$ is the portion, in which codeword $\mathbf{C}_i$ is the highest ranked solution, of the intersection of the decision error regions for pairwise error events in ML decoding associated with these L codewords. The effective Euclidean distance for decision is twice the minimum Euclidean distance from codeword $\mathbf{C}_0$ to the decision error region. The weight of the virtual codeword that has Euclidean distance that equals the effective Euclidean distance from codeword $\mathbf{C}_0$ in the signal space is the effective weight of codeword $\mathbf{C}_i$ for L-candidate list decoding. This effective weight is denoted as $w_i^{(L)}$. In order to determine this effective weight, the concept of partial decision error region, the corresponding partial effective Euclidean distance, and partial effective weight is introduced as follows.

Let $\mathbf{I}_a = \{\mathrm{I}_a(1), \mathrm{I}_a(2), \cdots, \mathrm{I}_a(L-1)\}$ denote the indices of any $(L-1)$ codewords other than $\mathbf{C}_0$ and $\mathbf{C}_i$, $\mathbf{I}_a \subset \{1, 2, \cdots, 2^K - 1\}$, $i \notin \mathbf{I}_a$. Define $\mathrm{R}_{0i,\mathbf{I}_a}^{(L)}$, the partial decision error region for $\mathrm{E}_{0i}^{(L)}$ determined by codewords $\mathbf{C}_{\mathrm{I}_a(1)}$, $\mathbf{C}_{\mathrm{I}_a(2)}$, $\cdots$, $\mathbf{C}_{\mathrm{I}_a(L-1)}$, as the portion, in which codeword $\mathbf{C}_i$ is the highest ranked solution, of the intersection, if it exists, of the decision error regions of the pairwise error events for ML decoding associated with codewords $\mathbf{C}_i$ and $\mathbf{C}_{\mathrm{I}_a(1)}$, $\mathbf{C}_{\mathrm{I}_a(2)}$, $\cdots$, $\mathbf{C}_{\mathrm{I}_a(L-1)}$

$$\mathrm{R}_{0i,\mathbf{I}_a}^{(L)} \subset \mathrm{R}_{0i} \cap \left(\bigcap_{k=1}^{L-1} \mathrm{R}_{0\mathrm{I}_a(k)} \right). \tag{2.8}$$

Define the *partial effective Euclidean distance* of $\mathbf{C}_i$ for L-candidate list decoding determined by these $(L-1)$ codewords as twice the minimum Euclidean distance from $\mathbf{C}_0$ to this partial decision error region $\mathrm{R}^{(L)}_{0i,\mathbf{I}_a}$. The weight of the virtual codeword that has Euclidean distance that equals the partial effective Euclidean distance from $\mathbf{C}_i$ in the signal space is the *partial effective weight* of $\mathbf{C}_i$ determined by these $(L-1)$ codewords and denoted as $w^{(L)}_{i,\mathbf{I}_a}$.

With the above definitions, the decision error region $\mathrm{R}^{(L)}_{0i}$ associated with $\mathbf{C}_i$ for L-candidate list decoding is the union of all the partial decision error regions for $\mathrm{E}^{(L)}_{0i}$ determined by any $(L-1)$ codewords other than $\mathbf{C}_0$ and $\mathbf{C}_i$

$$\mathrm{R}^{(L)}_{0i} = \bigcup_{\substack{\mathbf{I}_a \subset \{1,2,\cdots,2^K-1\} \\ i \notin \mathbf{I}_a}} \mathrm{R}^{(L)}_{0i,\mathbf{I}_a}. \tag{2.9}$$

The union in (2.9) is over all sets of indices $\mathbf{I}_a$, where $|\mathbf{I}_a| = L-1$ (as noted above). Therefore the effective weight of $\mathbf{C}_i$ for L-candidate list decoding is the minimum partial effective weight

$$w^{(L)}_i = \min_{\substack{\mathbf{I}_a \subset \{1,2,\cdots,2^K-1\} \\ i \notin \mathbf{I}_a}} w^{(L)}_{i,\mathbf{I}_a}. \tag{2.10}$$

The minimum in (2.10) is also over all sets of indices $\mathbf{I}_a$. The actual effective weight of a codeword in a given code for L-candidate list decoding can be found by enumerating all the partial effective weights of the codeword for L-candidate list decoding and taking the minimum.

Consider for example the $L=2$ candidate list decoding, the Euclidean distance relationships of signal points A, B, C, D, and E, except D_{CD}, D_{CE}, D_{DE}, can be illustrated in 2D Cartesian coordinates as shown in Fig. 2.4. As introduced in the previous section, let A denotes the all-zero codeword and let B denote codeword $\mathbf{C}_i$, and for arbitrary codewords $\mathbf{C}_{\mathrm{C}}$, $\mathbf{C}_{\mathrm{D}}$, and $\mathbf{C}_{\mathrm{E}}$, let D_{AX}, D_{AY}, and D_{AZ} be the minimum Euclidean distances from $\mathbf{C}_0$ to the partial decision error regions $\mathrm{R}^{(2)}_{0i,\{\mathrm{C}\}}$, $\mathrm{R}^{(2)}_{0i,\{\mathrm{D}\}}$, and $\mathrm{R}^{(2)}_{0i,\{\mathrm{E}\}}$, respectively. The corresponding partial effective weights are: $w^{(2)}_{i,\{\mathrm{C}\}} = \frac{(2D_{\mathrm{AX}})^2}{4E_s}$; $w^{(2)}_{i,\{\mathrm{D}\}} = \frac{(2D_{\mathrm{AY}})^2}{4E_s}$; $w^{(2)}_{i,\{\mathrm{E}\}} = \frac{(2D_{\mathrm{AZ}})^2}{4E_s}$. For any codeword $\mathbf{C}_{\mathrm{I}_a(1)}$, $\mathrm{I}_a(1) \neq 0$ or i, if $\mathrm{R}_{0i} \cap \mathrm{R}_{0\mathrm{I}_a(1)} \neq \emptyset$, the partial effective weight, $w^{(2)}_{i,\{\mathrm{I}_a\}}$, can be evaluated geometrically. The effective weight of codeword $\mathbf{C}_i$ for $L=2$ candidate list decoding will equal the minimum partial effective weight.

2.4 Overlapping for low weight codewords

The effective weight of a codeword for L-candidate list decoding depends on the amount of *overlapping* of the codeword with other low weight codewords of the code.

For a codeword $\mathbf{C}_i$,

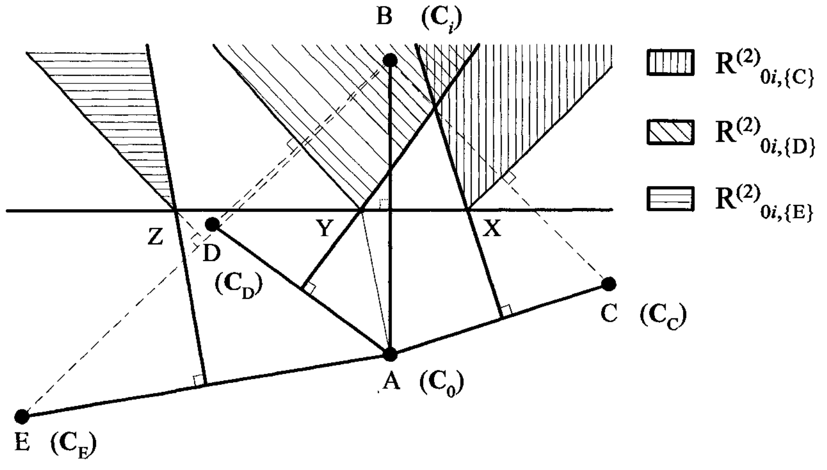


Figure 2.4: Partial effective weights. All the shaded regions shown in the figure are part of the decision error region $R_{0i}^{(2)}$.

- It is said that *sufficient overlapping* exists for this codeword for L-candidate list decoding if there exists a group of $(L-1)$ codewords, indexed by $C_1, \cdots, C_{L-1}$ (codewords $\mathbf{C}_{C_1}, \cdots, \mathbf{C}_{C_{L-1}}$), with the following properties:

  - Each codeword has weight $d_{\min}$, i.e., $\forall l \in \{1, 2, \cdots, L-1\}$,

$$w_{C_l} = d_{\min}; \tag{2.11}$$

  - The Hamming distance between any two of these $(L-1)$ codewords equals $d_{\min}$, i.e., $\forall l, h \in \{1, 2, \cdots, L-1\}$ and $l \neq h$,

$$d_{C_l C_h} = d_{\min}; \tag{2.12}$$

  - The Hamming distance between codeword $\mathbf{C}_i$ and any one of these $(L-1)$ codewords takes the minimum possible value, i.e., $\forall l \in \{1, 2, \cdots, L-1\}$,

$$d_{BC_l} = \max(d_{\min}, w_i - d_{\min}); \tag{2.13}$$

- It is said that *non-overlapping* exists for this codeword for L-candidate list decoding, if any group of L codewords, including $\mathbf{C}_i$, that results in the effective codeword weight $w_i^{(L)}$ contains a subset of $\lceil \log_2(L+1) \rceil$ codewords such that any two codewords from that subset do not have common positions of ones;

- Otherwise, It is said that *insufficient overlapping* exists for this codeword for L-candidate list decoding.

In the following example, the effective weight for a codeword of weight $w_i = d_{\min}$ is evaluated for three linear block codes, $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$, with $L = 2$ candidate list decoding, and the influence of the overlapping property of low weight codewords of a code is illustrated. As in Section 2.2.A, for a list size of $L = 2$, three codewords $\mathbf{C}_0$, $\mathbf{C}_i$, and $\mathbf{C}_\mathrm{C}$, denoted by A, B, and C in the signal space, respectively, are considered. Assume $N = 8$, $d_{\min} = 4$ and for all three codes, the minimum effective weights for $L = 2$ candidate list decoding are obtained from the scenario that involve codewords listed in Table 2.1. The Euclidean distance relationship between codewords can be represented by triangles in the signal space, as illustrated in Fig. 2.5. In this figure, point O, the intersection of the perpendicular bisectors of the edges AB and AC, is the point within the $L = 2$ candidate list decoding decision error region that has the smallest Euclidean distance from A. The effective Euclidean distance is the length of the diameter of the circumcircle of triangle ABC.

1. Sufficient overlapping. In code $\mathcal{C}_1$, sufficient overlapping exists for codeword $\mathbf{C}_i$ for $L = 2$ candidate list decoding. The effective weight in this scenario is $w_i^{(2)} = \frac{4}{3}d_{\min} < w_i + w_\mathrm{C}$ and equals the lower bound $d^{(2)}_{1,\mathrm{LB}}$.
2. Insufficient overlapping. In code $\mathcal{C}_2$, insufficient overlapping exists for codeword $\mathbf{C}_i$, for $L = 2$ candidate list decoding. The effective weight in this scenario is $w_i^{(2)} = \frac{8}{5}d_{\min} < w_i + w_\mathrm{C}$ and is greater than the lower bound $d^{(2)}_{1,\mathrm{LB}}$.
3. Non-overlapping. In code $\mathcal{C}_3$, non-overlapping exists for codeword $\mathbf{C}_i$, for $L = 2$ candidate list decoding. The effective weight in this scenario is $w_i^{(2)} = 2d_{\min} = w_i + w_\mathrm{C}$ and is the greatest of the three cases.

The actual minimum effective weights of codes $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ for $L = 2$ candidate list decoding differ and depend on the overlapping property of low weight codewords.

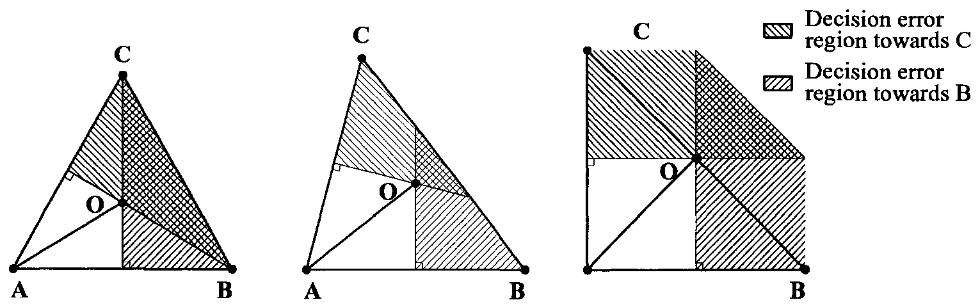
In the following two sections, the analysis method introduced in Sections 2.1 and 2.2 will be applied to convolutional codes and turbo codes. As will be shown, these two types of codes have quite different overlapping properties of low weight codewords, which leads to differences in effective weight increments.

2.5 Analysis of list decoding for convolutional codes

In this section, the terminated binary feed-forward convolutional code (FFCC) with input sequence length K, number of states 2^ν, and rate r is considered. The input data sequence

Table 2.1: Codewords A, B and C and the Hamming distance relationship for code $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$.

	Code $\mathcal{C}_1$	Code $\mathcal{C}_2$	Code $\mathcal{C}_3$
A	0000 0000	0000 0000	0000 0000
B	1111 0000	1111 0000	1111 0000
C	0011 1100	0001 1110	0000 1111
d_{AB}	4	4	4
d_{AC}	4	4	4
d_{BC}	4	6	8


C
O
A
B
C
O
A
B
C
O
B
▧ Decision error region towards C
▨ Decision error region towards B

(a) Code $\mathcal{C}_1$ (b) Code $\mathcal{C}_2$ (c) Code $\mathcal{C}_3$

Figure 2.5: The influence of the overlapping property of low weight codewords of a code on the effective weight.

is followed by ν terminating bits to reset the encoder memory. This terminated finite length FFCC is a binary (N, K) linear block code, where $N = (K + \nu)/r$.

In [34], Takeshita and Costello introduced the concept of *signature* for binary sequences. In the following, a slightly different definition of this concept is used.

For a length-K binary sequence $\mathbf{u}$, define:

- Delay $\mathtt{del}(\mathbf{u})$ and degree $\mathbf{deg}(\mathbf{u})$ as the degrees of the terms with the smallest and the largest degrees in $\mathbf{u}(D) = u_0 + u_1 D + u_2 D^2 + \cdots + u_{K-1} D^{K-1}$, respectively;
- Signature $\mathtt{sig}(\mathbf{u}) = (u_{\mathtt{del}(\mathbf{u})}, u_{\mathtt{del}(\mathbf{u})+1}, \cdots, u_{\mathbf{deg}(\mathbf{u})-1}, u_{\mathbf{deg}(\mathbf{u})})$;
- Span $\mathbf{sp}(\mathbf{u})$ as $\mathbf{deg}(\mathbf{u}) - \mathtt{del}(\mathbf{u}) + 1$.

A binary sequence $\mathbf{u}$ can be uniquely identified by its signature $\mathtt{sig}(\mathbf{u})$ and its delay $\mathtt{del}(\mathbf{u})$. Each set of sequences of length K sharing the same signature $\mathtt{sig}(\mathbf{u})$, called the *common signature set*, has $K - \mathbf{sp}(\mathbf{u}) + 1$ elements.

For a terminated convolutional code, the weight enumerating function (WEF) can be expressed as

$$\begin{aligned}
\mathrm{B}_{\mathcal{C}}(H) &= 1+\sum_{j=1}^{J} B_j H^{d_j} \\
&= 1+\sum_{j:d_{\min}\le d_j<2d_{\min}} B_j H^{d_j} \\
&\quad + \sum_{j:2d_{\min}\le d_j<3d_{\min}} \left(B_j^{\mathrm{SD}}+B_j^{\mathrm{MD}}\right) H^{d_j} \\
&\quad + \sum_{j:d_j\ge 3d_{\min}} B_j H^{d_j}. \qquad (2.14)
\end{aligned}$$

If the path of a codeword on the trellis departs from and later merges into the all-zero state once or more than once, the codeword is called a single detour or a multi-detour codeword, respectively. The superscripts "SD" and "MD" are used to denote the multiplicity of these two kinds of codewords of the same weight as in the second summation term in (2.14).

In the following, the effect of codewords represented by each summation term on the right hand side of (2.14) on the asymptotic performance evaluation for L-candidate list decoding is discussed.

- The codewords in the first summation have weights in $[d_{\min}, 2d_{\min})$, and are all single detour codewords [35]. For a terminated FFCC, encoder input sequences from the same common signature set result in codewords of the same weight and effective weight. The effective weights of these codewords for L-candidate list decoding are evaluated.

- The codewords in the second summation have weights in $[2d_{\min}, 3d_{\min})$. For $L = 2$ and $L = 3$ candidate list decoding, the actual effective weight for these codewords are the same as their codeword weights. For $L \ge 4$ candidate list decoding, the effective weight increments equal zero based on the lower bound on effective weight, as discussed in Section 2.2. Therefore, for these codewords, their weights are used as the effective weights for L-candidate list decoding.

- For codewords of weight larger than or equal to $3d_{\min}$, represented in the third summation, their effect on the asymptotic performance is ignored.

From the above discussion, the probability of codeword error of a terminated FFCC with L-candidate list decoding can be approximated by

$$\mathrm{P(CE)} \approx \sum_{j:d_{\min}\le d_j<2d_{\min}} \sum_{s=0}^{S_j-1} [K-\mathbf{sp}(\mathbf{u}_{j,s})+1]\,\mathrm{P}_{d_j+\Delta d_{j,s}^{(L)}}$$

$$B_j^{\mathrm{MD}} = \sum_{\substack{h\,:\,d_{\min} \leq d_h < \frac{3}{2}d_{\min} \\ d_g = d_j - d_h \in \mathbf{d}}} \sum_{s=0}^{S_h-1} \sum_{t=0}^{S_g-1} \binom{K - \mathbf{sp}(\mathbf{u}_{h,s}) - \mathbf{sp}(\mathbf{u}_{g,t}) - \nu}{2} \tag{2.17}$$

$$+ \sum_{j:2d_{\min} \leq d_j < 3d_{\min}} \sum_{s=0}^{S_j-1} [K - \mathbf{sp}(\mathbf{u}_{j,s}) + 1]\, \mathrm{P}_{d_j}$$

$$+ \sum_{j:2d_{\min} \leq d_j < 3d_{\min}} B_j^{\mathrm{MD}} \mathrm{P}_{d_j} \tag{2.15}$$

where S_j is the multiplicity of the distinct signatures $\mathbf{u}_{j,s} = \mathtt{sig}(\mathbf{u})$ of input sequences resulting in single detour codewords of weight d_j. $\Delta d_{j,s}^{(L)}$ is the effective weight increment of codewords with signature $\mathbf{u}_{j,s}$. $S_j^{(L)}$ is the multiplicity of the distinct signatures $\mathbf{u}_{j,s}^{(L)} = \mathtt{sig}(\mathbf{u})$ of input sequences resulting in codewords of effective weight $d_j^{(L)}$. On the right hand side of (2.15), the first summation term corresponds to codewords of weight within $[d_{\min}, 2d_{\min})$, and the second and the third summation terms correspond to the single detour and the multi-detour codewords of weight within $[2d_{\min}, 3d_{\min})$, respectively. For $K \gg \nu$, $K - \mathbf{sp}(\mathbf{u}_{j,s}) + 1 \approx K$.

$$\mathrm{P(CE)} \approx K \sum_{j:d_{\min} \leq d_j < 2d_{\min}} S_j^{(L)} \mathrm{P}_{d_j^{(L)}}$$

$$+ \sum_{j:2d_{\min} \leq d_j < 3d_{\min}} \left(KS_j + B_j^{\mathrm{MD}}\right) \mathrm{P}_{d_j} \tag{2.16}$$

where B_j^{MD} is the multiplicity of codewords of weight within $[2d_{\min}, 3d_{\min})$ as a result of multiple detours (exactly two detours in this case) which can be expressed as in (2.17), where the first summation is over all h such that d_h is in the range $d_{\min} \leq d_h < \frac{3}{2}d_{\min}$, and $d_j - d_h$ equals the weight of a codeword.

From Sections 2.1 and 2.3, by enumerating the partial effective weights and taking the minimum, the effective weight of a codeword can be obtained. For a terminated FFCC with $L = 2$ and $L = 3$ candidate list decoding, the following procedure to analyze the performance is proposed.

1. Enumerate the signatures of the encoder input sequences causing single detour codewords with weight smaller than $2d_{\min}$;

2. Obtain effective weight for $L = 2$ and $L = 3$ candidate list decoding for the codewords corresponding to each signature enumerated in the first step via exhaustive search based on the set of the enumerated signatures;

3. Enumerate the signatures of the encoder input sequences causing single detour codewords of weight within $[2d_{\min}, 3d_{\min})$;

4. Evaluate B_j^{MD}, the multiplicity of codewords that have weights in $[2d_{\min}, 3d_{\min})$ according to (2.17);

5. Approximate the performance of the FFCC with L-candidate list decoding using (2.16).

Ignoring the tail effect, for convolutional codes, a time shifted version of an input sequence with a signature that results in a low weight codeword results in another low weight codeword. With the multiplicity of low weight codewords proportional to the length of the input data block, although sufficient overlapping does not always exist for a low weight codeword, as will be shown in the analysis examples in Section 2.7, the possibility that non-overlapping exists for a low weight codeword is extremely low. Therefore, for convolutional codes, since the actual effective weight of a codeword is not significantly larger than the lower bound prediction on the effective weight, the latter can be used to produce a good approximation for performance evaluation.

2.6 Analysis of list decoding for turbo codes

The proposed method is also applicable to the analysis of list decoding for turbo codes. Unlike the analysis of convolutional codes discussed in the previous section, due to the existence of the semi-random interleaver, the multiplicity of low weight codewords in turbo codes is low and independent of the length of the data block. This results in an extremely low probability of sufficient overlapping and a high probability of non-overlapping for low weight codewords. Because of this unique overlapping property of low weight codewords, for turbo codes, the effective weight of a codeword for L-candidate list decoding in these codes is significantly higher than that predicted by the lower bound.

As different turbo interleavers are associated with different codeword weight spectra, their choice may influence the analysis. In this chapter, performance evaluations are performed for codes with specified interleavers, i.e., codes with determined weight spectra, as opposed to the uniform interleaver approach as presented in [31, 32].

To obtain the weight spectrum of the code under consideration, the following concepts, originally introduced by Benedetto and Montorsi in [36], are used. The input-redundancy weight enumerating function (IRWEF) of a code is defined as

$$\mathrm{A}_{\mathcal{C}}(W,H)=\sum_{z,y}A_{z,y}W^{z}H^{y} \tag{2.18}$$

where W and H are indeterminates and $A_{z,y}$ denotes the number of codewords generated by an input information sequence of Hamming weight z with parity bits of Hamming weight y (the overall Hamming weight is $z+y$).

The conditional weight enumerating function (CWEF) of the parity check bits $\mathrm{A}_z(H)$ generated by the code $\mathcal{C}$ corresponding to the input words of weight z is implicitly given by

$$\mathrm{A}_{\mathcal{C}}(W,H)=\sum_{z}W^{z}\sum_{y}A_{z,y}H^{y}=\sum_{z}W^{z}\cdot\mathrm{A}_z(H). \tag{2.19}$$

The CWEF of a turbo code can be approximated by a limited number of terms, as

$$\mathrm{A}_{\mathcal{C}}(W,H)\approx\sum_{z=0}^{Z}W^{z}\cdot\mathrm{A}_z(H) \tag{2.20}$$

where Z is the maximum input weight under consideration.

Using (2.20), the low weight portion of the IRWEF of a turbo code can be approximated by enumerating CWEFs with low input sequence weights, and can be used for evaluation of the performance for L-candidate list decoding. In the following analysis, only the low weight codewords of turbo codes caused by input sequences of weight up to $z=4$ are enumerated. These enumerated low weight codewords are then used to approximate the effective weights of codewords for $L=2$ and $L=3$ candidate list decoding, which are cases that can be solved geometrically.

2.7 Analysis examples and simulation results

Convolutional codes

The proposed analytical method is used to evaluate the performance of FFCCs with $L=2$ and $L=3$ candidate list decoding. The convolutional codes under evaluation have maximum free distance, as discussed by Odenwalder in [37] and Larsen in [38].

The lower bound on the minimum effective weight, $d_{1,\mathrm{LB}}^{(2)}$, $d_{1,\mathrm{LB}}^{(3)}$, evaluated by (2.6), (2.7), and the actual minimum effective weight, $d_{1,\min}^{(2)}$, $d_{1,\min}^{(3)}$, evaluated by (2.3), (2.10), are listed in Table 2.2. This table shows that the actual minimum effective weight does not always equal the lower bound predictions and that codes with the same minimum weights may have different minimum effective weights for L-candidate list decoding.

Tables 2.3 and 2.4 show the detailed analysis results for the G $= (5,7)_8$ code and the G $= (53,75)_8$ code, respectively. For both Tables 2.3 and 2.4, sub-table (a) compares $d_{j,\mathrm{LB}}^{(L)}$ and $d_{j,\min}^{(L)}$ for $d_j < 2d_{\min}$. As the codeword weight d_j increases, both the actual effective weight increment and its lower bound decrease. In sub-tables (b) and (c)[5], the effective weights for codewords of weight less than $2d_{\min}$ are listed together with the multiplicity of distinctive signatures of encoder input associated with these codewords. For both codes, the number of terms contained in the EWEF for L-candidate list decoding is different from that of the original WEF, because codewords of the same weight may result in different effective weights for L-candidate list decoding. Sub-tables (d) show S_j and B_j^{MD} for codewords of weight within $[2d_{\min}, 3d_{\min})$.

Figs. 2.6 through 2.9 show simulation results for the FFCCs with $K = 1024$ on the AWGN channel in terms of codeword error rate (CER) vs. SNR. In the simulations, serial list Viterbi algorithm as described in [28] is employed as the list decoding algorithm and the resulting candidate list is compared with the transmitted codeword to determine whether or not a decoding error has occurred. (In a real system, error checking could be accomplished by an outer error detection code.) In these figures, the analytical approximation based on the actual effective codeword weights is denoted as "Approx. based on actual EWEF", and the analytical approximation based on the lower bound on effective weight is denoted as "Approx. based on lower bound EWEF". The SNR is defined as E_b/N_0, where E_b is the energy per information bit. As a reference, the performance of the codes with ML decoding and its union bound approximation are also shown in the figures. The improvement of the analytical approximation based on the actual effective weight as compared to the one based on the lower bound on effective weight is clear.

[5] To conserve space, in Table 2.4(c), only the first 11 terms are listed.

Table 2.2: Maximum free distance convolutional codes.

ν	G	$d_{\min}$	$d^{(2)}_{1,\text{LB}}$	$d^{(2)}_{1,\min}$	$d^{(3)}_{1,\text{LB}}$	$d^{(3)}_{1,\min}$
2	$(5,7)_8$	5	6.7	7.1	7.5	8
3	$(15,17)_8$	6	8	8.9	9	10
4	$(23,35)_8$	7	9.3	9.8	10.5	11.0
5	$(53,75)_8$	8	10.7	11.6	12	13.0
6	$(133,171)_8$	10	13.3	13.3	15	15
7	$(247,371)_8$	10	13.3	14.2	15	16
2	$(5,7,7)_8$	8	10.7	10.7	12	12
3	$(13,15,17)_8$	10	13.3	13.3	15	15
4	$(25,33,37)_8$	12	16	16	18	18
5	$(47,53,75)_8$	13	17.3	17.8	19.5	20
6	$(133,145,175)_8$	15	20	20.5	22.5	23
7	$(225,331,367)_8$	16	21.3	22.3	24	25.0
2	$(5,7,7,7)_8$	10	13.3	14.2	15	16
3	$(13,15,15,17)_8$	13	17.3	17.8	19.5	20
4	$(25,27,33,37)_8$	16	21.3	21.3	24	24
5	$(53,67,71,75)_8$	18	24	24	27	27
6	$(135,135,147,163)_8$	20	26.7	26.7	30	30
7	$(235,275,313,357)_8$	22	29.3	30.2	33	34

Table 2.3: Analysis results for 4-state FFCC with G $= (5,7)_8$, $K = 1024$

(a)

d_j	S_j	$d^{(2)}_{j,\text{LB}}$	$d^{(2)}_{j,\min}$	$d^{(3)}_{j,\text{LB}}$	$d^{(3)}_{j,\min}$
5	1	6.6667	7.1429	7.5000	8
6	2	7.1429	7.1429	7.7778	8
7	4	7.6923	8.0769	8.1250	8.6000
8	8	8.3333	9	8.5714	9.4717
9	16	9.0909	9.9474	9.1667	10.3333

(b)

$d^{(2)}_j$	$S^{(2)}_j$
7.1429	3
8.0769	4
9	8
9.9474	16

(c)

$d^{(3)}_j$	$S^{(3)}_j$
8	3
8.6000	4
9.4717	8
10.3333	16

(d)

d_j	S_j	B^{MD}_j
10	32	517653
11	64	2067562
12	128	5161287
13	256	10307352
14	512	18011252

Table 2.4: Analysis result for 32-state FFCC with G = $(53, 75)_8$, $K = 1024$

(a)

d_j	S_j	$d^{(2)}_{j,\mathrm{LB}}$	$d^{(2)}_{j,\min}$	$d^{(3)}_{j,\mathrm{LB}}$	$d^{(3)}_{j,\min}$
8	1	10.6667	11.5714	12	13
9	8	11.1304	11.5714	12.2667	13
10	7	11.6364	11.6364	12.5714	13
11	12	12.1905	12.5714	12.9231	13.5714
12	48	12.8000	12.8000	13.3333	13.9655
13	95	13.4737	13.7647	13.8182	14.3333
14	281	14.2222	14.2222	14.4000	14.8776
15	604	15.0588	15.2113	15.1111	15.4000

(b)

$d^{(2)}_j$	$S^{(2)}_j$	$d^{(2)}_j$	$S^{(2)}_j$
11.5714	9	14.4444	40
11.6364	1	14.4810	1
12.4615	6	14.7273	201
12.5714	11	14.7368	21
12.8000	3	15.2113	41
13.3784	1	15.3846	13
13.5000	43	15.4000	43
13.5211	1	15.6977	318
13.7647	54	15.7143	48
14.2222	3	16.3366	61
14.2857	1	16.3636	136

(c)

$d^{(3)}_j$	$S^{(3)}_j$
13	10
13.5556	6
13.5714	6
13.9264	3
13.9655	2
14	3
14.2500	17
14.3333	1
14.3846	1
14.4135	10
14.4648	1
⋮	⋮

(d)

d_j	S_j	B^{MD}_j
16	1272	505515
17	3334	8063174
18	7615	39209578
19	18131	63322844
20	43195	94818828
21	99206	175101234
22	236116	254296278
23	556537	375008762

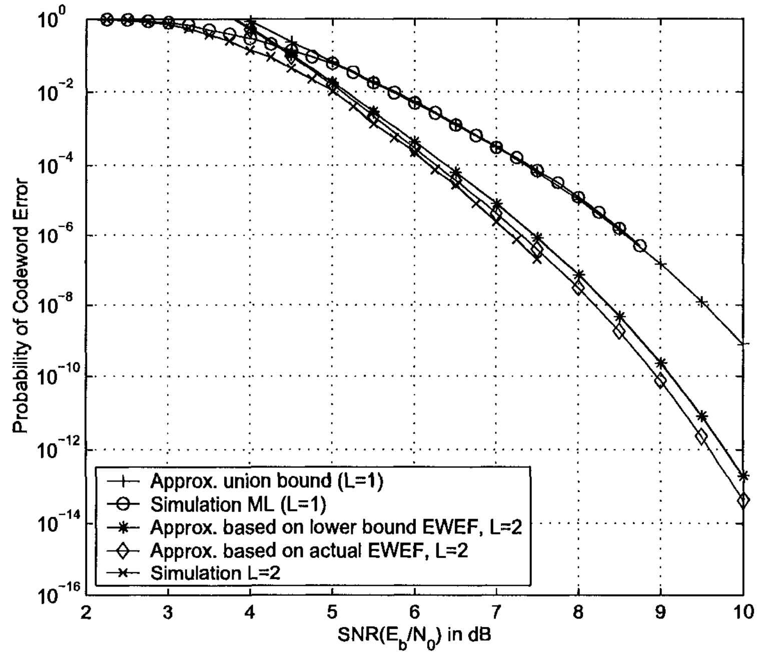


Figure 2.6: CER of terminated convolutional code, 4-state, G $= (5,7)_8$, $K = 1024$ with $L = 2$ candidate list decoding in the AWGN channel.

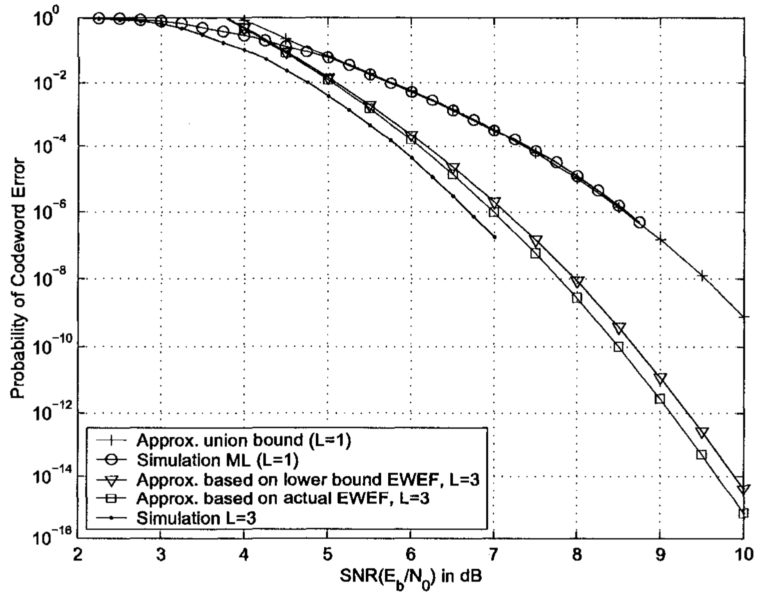


Figure 2.7: CER of terminated convolutional code, 4-state, G $= (5,7)_8$, $K = 1024$ with $L = 3$ candidate list decoding in the AWGN channel.

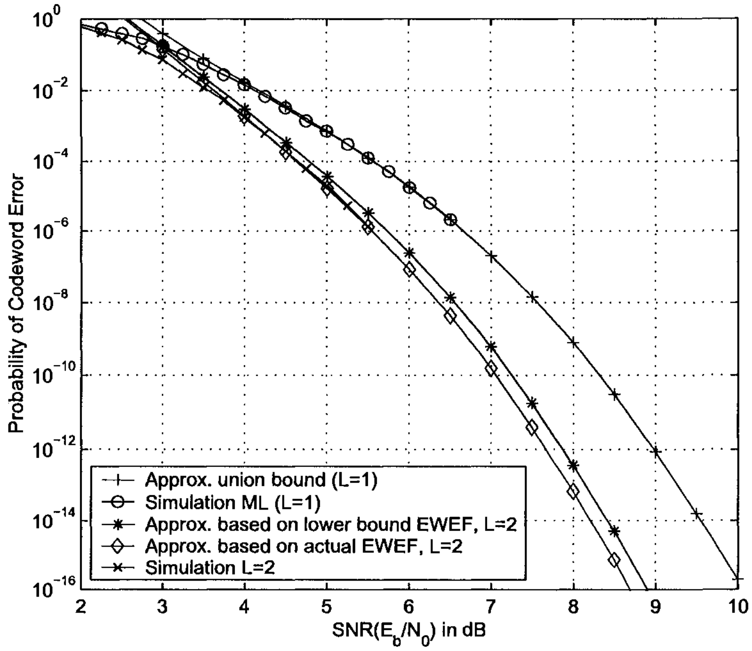


Figure 2.8: CER of terminated convolutional code, 32-state, G = $(53, 75)_8$, $K = 1024$ with $L = 2$ candidate list decoding in AWGN channel.

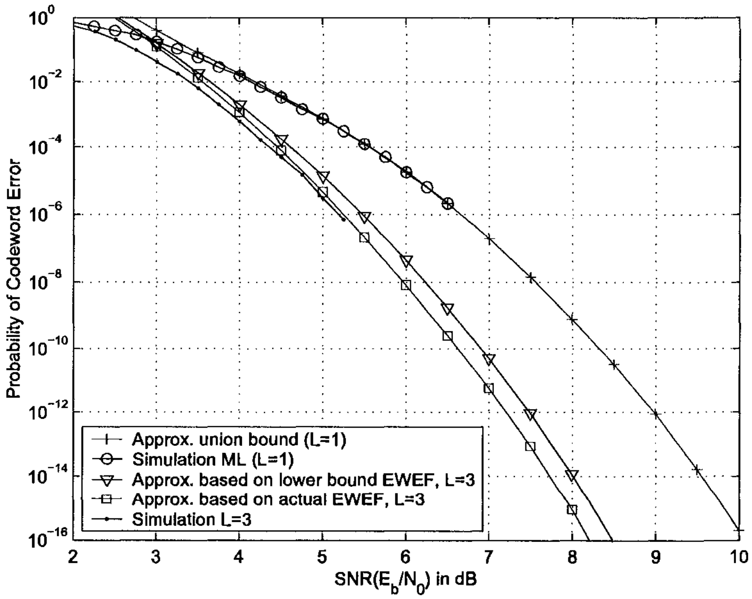


Figure 2.9: CER of terminated convolutional code, 32-state, G = $(53, 75)_8$, $K = 1024$ with $L = 3$ candidate list decoding in AWGN channel.

Table 2.5: Turbo code with random interleaver, $K = 1760$.

d_j	b_j	$d^{(2)}_{j,\text{LB}}$	$d^{(2)}_{j,\text{min}}$	$d^{(3)}_{j,\text{LB}}$	$d^{(3)}_{j,\text{min}}$
14	2	18.6667	28	21	28
17	1	20.1026	31	21.8400	31
18	1	20.6316	32	22.1667	32
20	2	21.7778	34	22.9091	34
21	3	22.4000	35	23.3333	35
22	6	23.0588	33.3474	23.8000	36
24	3	24.5000	38	24.8889	38
25	3	25.2903	37.4784	25.5294	39
26	11	26.1333	33.4713	26.2500	36
27	1	27.0345	41	27.0667	41
28	12	28	28	28	28

Turbo codes

Table 2.5 shows the analysis results for the turbo code with a random interleaver. The length of the input data block is $K = 1760$. To conserve space, only the results for codewords of weight less than or equal to $2d_{\text{min}}$ are tabulated. The analysis results show that the effective weight increments for turbo codes can be significantly higher than those predicted by the lower bound.

For every codeword, the approximations obtained for effective weights are optimistic because the low weight codewords corresponding to input sequences of weight larger than $z = 4$ are not considered in the evaluation, and there is the possibility that the actual effective weight is lower. Nevertheless, these results give us an indication that when the multiplicity of low weight codewords is low and sufficient overlapping rarely exists for low weight codewords, the effective weight increments are significant.

List decoding for turbo codes was studied by Narayanan and Stüber in [30], Leanderson and Sundberg in [39]. Motivated by the theoretical results, an efficient list decoding algorithm for turbo codes is proposed in this research project and will be presented in detail in the next chapter. However, since the list decoding algorithms described in this thesis are sub-optimal, there is a significant gap between their performance and that achievable by optimal algorithms. Because of this, simulation results for turbo codes with L-candidate list decoding are not included here.

2.8 Chapter summary

The performance improvement introduced by list decoding is the result of an increase of the effective weights as compared to the original codeword weights. As a rule, the lower weight codewords achieve higher effective weight increments, and in the most pessimistic situation, the effective weight increments of a codeword vanish as its weight exceeds $2d_{\text{min}}$.

For a given code, the actual effective weight for a low weight codeword depends on its overlapping property. Effective weight larger than the lower bound prediction can be achieved when sufficient overlapping does not exist for the codeword for L-candidate list decoding.

In this chapter, it has been shown that for a linear block code, the actual effective weight of every codeword can be obtained by evaluating all the partial effective weights and taking the minimum. A more accurate approximation of the performance of a given code with list decoding can be obtained based on the actual effective weights compared to the one based on the lower bound on effective weight.

For feed-forward convolutional codes, because of the time invariant nature of the encoder, the evaluation of the actual effective weight can be simplified and carried out on a signature basis as opposed to a codeword basis. Analysis for convolutional codes shows that the actual effective weight does not always equal the lower bound predictions and codes with the same minimum weight may have different minimum effective weights. For turbo codes, the actual effective weights can be approximately evaluated considering the low weight codewords. Furthermore, analysis for turbo codes shows that the approximate effective weight is significantly higher than that predicted by the lower bound due to the fact that the multiplicity of the low weight codewords of a turbo code is low, and sufficient overlapping rarely exists for a low weight codeword.

To fully exploit the potential in performance improvement offered by list decoding, the results presented in this chapter should motivate the investigation of better sub-optimal list decoding algorithms for turbo codes. Moreover, to achieve better performance of turbo codes with list decoding, a new criterion for the design of turbo interleavers could be formulated. Such a criterion should aim to increase the possibility of non-overlapping for low weight codewords in order to increase the effective weight of the low weight codewords for list decoding, and, as a result, reduce the probability of a codeword error with list decoding.

Chapter 3

Efficient List Decoding for Turbo Codes [1]

In [28], Seshadri and Sundberg introduced the list Viterbi algorithm. They also proposed the parallel list Viterbi algorithm and the serial list Viterbi algorithm for efficient implementation.

For turbo codes, two iterative list decoding algorithms can be found in the literature:

- Log-MAP + SLVA: In [30], Narayanan and Stüber modified the serial list Viterbi algorithm (SLVA) so that it can take into account the extrinsic information in addition to the log-likelihood ratios (LLRs) of the channel observations. The modified serial list Viterbi algorithm is then applied to one of the component decoders at the last iteration of a typical iterative turbo decoder to generate a list of candidate codewords.
- MLLA: In [39], Leanderson and Sundberg addressed the problem of list decoding of turbo codes by introducing the Max-log list algorithm (MLLA) to simultaneously produce the soft symbol decisions and the soft sequence list for decoding convolutional codes. When used for the component codes of a turbo code, the Max-log list algorithm can be viewed as an efficient integration of the list Viterbi algorithm into the process of the Max-Log-MAP algorithm. To alleviate the error fluctuation phenomenon that typically occurs in iterative decoding, as described in [40], list decoding is invoked repeatedly after every iteration instead of only after the last iteration [31]. Max-log list algorithm exists in both optimal and low complexity suboptimal versions when $L > 3$.

These two list decoding algorithms for turbo codes belong to the same general category

[1] A version of this chapter was presented in part in "Efficient list decoding for parallel concatenated convolutional codes," at the *IEEE PIMRC 2004*, Barcelona, Spain, September 2004.

that avoids generating the list for the turbo code directly by utilizing the list generated for the RSC component code by the list Viterbi algorithm as the list for the turbo code. Although it was mentioned in [30] that the modified serial list Viterbi algorithm can be applied to either one or both of the component codes of the turbo code, the operation of the serial list Viterbi algorithm concerns only one component code at a time, therefore these alternatives all fall in the same generalization.

In this chapter, a different approach is taken and the list is constructed directly for the turbo code rather than for its component codes. In Section 3.1, analysis of the error events of convolutional codes and turbo codes is given to demonstrate the differences when list decoding algorithms are invoked. In Section 3.2, a new algorithm is introduced for list decoding for turbo codes. Simulation results are shown in Section 3.3, which is followed by the chapter summary in Section 3.4.

3.1 Analysis of list decoding algorithms for convolutional codes and turbo codes

In this section, both binary RSC codes with memory of size ν and turbo codes with a prime interleaver, as specified in [22], and a random interleaver are considered. Let $\mathbf{u} = (u_0, u_1, \cdots, u_{K-1})$ and $\mathbf{v} = (v_0, v_1, \cdots, v_{N-1})$ denote the binary input sequence and the output codeword of the encoder, respectively. For rate $r = 1/2$ RSC codes, $N = 2(K + \nu)$. For rate $r = 1/3$ turbo codes, $N = 3K + 4\nu$. Let $\mathbf{x} = (2\mathbf{v} - 1)$ denote the BPSK modulated symbols, and $\mathbf{y} = \mathbf{x} + \mathbf{n}$ the received symbols, where $\mathbf{n} = (n_0, n_1, \cdots, n_{N-1})$, n_i are i.i.d. $\sim N(0, N_0/2)$ assuming the AWGN channel.

Recall the definitions of *signature* for binary sequences introduced in Section 2.5. For RSC codes, some of the possible signatures lead the encoder to the all-zero state before termination. Let $\mathtt{SIG}^{\mathrm{st}}$ denote the set of signatures with this *self termination* property. The majority of contributions to the low weight portion of the WEF are made by input sequences that have signatures belonging to $\mathtt{SIG}^{\mathrm{st}}$. A useful approximation of the low weight portion of the WEF for RSC codes is:

$$\mathrm{B}_C(H) \cong \sum_{j=d_{\mathrm{free}}}^{J} \sum_{s=0}^{S} (K - \mathrm{sp}(\mathbf{u}_{j,s}) + 1) H^j \tag{3.1}$$

where H is a dummy variable, d_{free} is the free distance of the code, J is the maximum codeword weight considered, and S is the number of the distinct signatures $\mathbf{u}_{j,s} = \mathbf{sig}_{j,s}(\mathbf{u}) \in \mathtt{SIG}^{\mathrm{st}}$ that result in weight j output sequences. Eq.(3.1) shows the strong influence of common signature sets on the error events for RSC codes: the multiplicity of the terms in the low weight portion of the WEF of the RSC codes is on the order of K. When two RSC codes are concatenated by an interleaver forming a turbo code, the low weight codewords of the turbo code occur only when these two RSC codes produce low weight codewords simultaneously. Because of the existence of the interleaver, the influence of the common signature sets on the error events vanishes.

Using the concept of the uniform interleaver introduced in [36], when $K \gg \nu$, the WEF of a turbo code with RSC component encoders can be approximated as [41]:

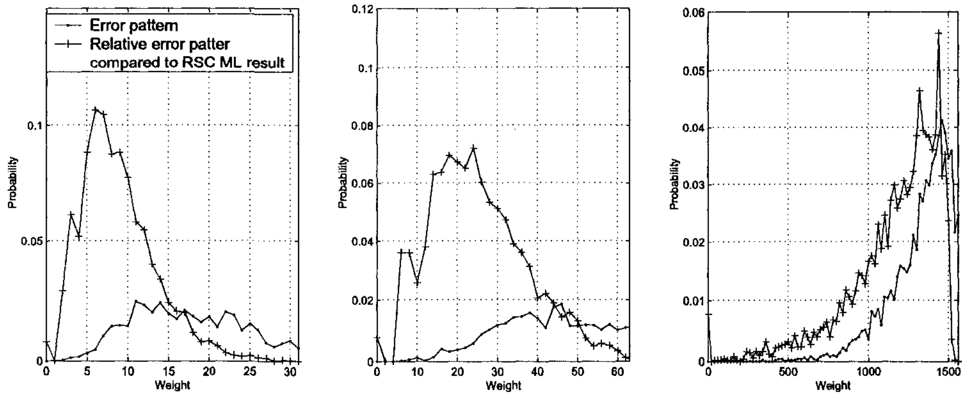
$$\begin{aligned} \mathrm{B}_{\mathcal{C}}(H) &\cong \sum_{j=1}^{\lfloor K/2 \rfloor} \binom{2j}{j} \left(\frac{H^{d_{2\mathrm{p}}+1}}{1 - H^{d_{2\mathrm{p}}-2}} \right)^{2j} \\ &= 2H^{2d_{2\mathrm{p}}+2} + 4H^{3d_{2\mathrm{p}}} + 2H^{4d_{2\mathrm{p}}-2} \dots \end{aligned} \tag{3.2}$$

where $d_{2\mathrm{p}}$ is the minimum weight of parity bits in the component RSC codewords generated by input sequences of weight 2. From (3.2), it can be seen that in the low weight region of the turbo code WEF, the coefficients of each term are independent of K and usually far lower than K. List decoding allows for correction of the situation when ML decoding results in the wrong codeword by enlarging the radius of the *decision hypersphere* so that it includes L codewords [28].

Let an *error pattern* refer to the bitwise XOR of the decoded binary sequence $\hat{\mathbf{u}}$ and the original binary input sequence $\mathbf{u}$. Let the *relative error pattern* refer to the bitwise XOR of the decoded binary sequence $\hat{\mathbf{u}}$ and the ML solution $\hat{\mathbf{u}}^{\mathrm{ML}}$. A list decoding algorithm searches for valid candidate codewords based on relative error patterns associated with low weight codewords.

In the list decoding algorithms for turbo codes described in [30] and [39], the relative error patterns are those of the component RSC code rather than of the whole turbo code. If relative error patterns that correspond to low weight codewords for the RSC component code do not result in low weight codewords for the turbo code, which is one of the primary objectives of the interleaver design for turbo codes and therefore is the case for a well designed interleaver, a large fraction of the search results will have high turbo code codeword weights. Since only a small number of generated codewords are potential candidates for turbo code list decoding, such a search is inefficient and the incremental gains from the increases of the list size diminish quickly, as noticed in [30]. This is due to the fact that if the correct codewords do not show up in the front of the list, the codewords with relative error patterns that result in low weight codewords for the RSC code but not for the turbo code will quickly fill up the list.

Note that even though iterative decoding for a turbo code does not guarantee the ML



(a) Error patterns, relative error patterns.

(b) RSC error codewords.

(c) Turbo code error codewords.

Figure 3.1: Weight statistics of *error patterns*, *relative error patterns* and the associated corresponding codewords with serial list Viterbi algorithm (SLVA), when SLVA fails, $L = 128$, 39 cases.

solution, the above analysis is still valid because the serial list Viterbi algorithm works on the component RSC code instead of on the turbo code.

Fig. 3.1 shows the statistics of the error patterns and the relative error patterns compared to the ML solution found by the serial list Viterbi algorithm for the component code using a list of size $L = 128$ constructed by a modified serial list Viterbi algorithm at SNR $= E_b/N_0 = 0.6$ dB where FER $= 7.0 \times 10^{-4}$. Details of the simulation setup are described later in Section 3.4. The statistics show the general fact that the error patterns, relative error patterns, and the corresponding RSC component codewords are constrained to low weight, but the weights of the corresponding turbo code codewords spread out over a large spectrum.

Based on previously presented analysis, a list generated for the component RSC with list Viterbi algorithm differs from the optimal list for the turbo code, which makes the traditional list decoding algorithm less efficient for large list sizes. A list decoding algorithm that generates the list for the turbo code, rather than for its component codes, is expected to result in better performance compared to the traditional algorithms when the list size is large.

Another issue in implementing list decoding is the trade-off between performance and efficiency. The most commonly used algorithms for iterative decoding of the turbo codes are the logarithm-maximum a posteriori (Log-MAP) and the MAX-Log-MAP algorithms, the latter being a simplified version of the former at a cost of about 0.5 dB degradation [42]. The fact that the Log-MAP algorithm is a symbol-wise decoding algorithm and the list

Viterbi algorithm is a sequence-wise algorithm makes their integration difficult. Combining the list Viterbi algorithm with the Max-Log-MAP algorithm, as in Max-log list algorithm, results in efficient implementation, while performing list Viterbi algorithm and the Log-MAP algorithm separately results in better performance with higher decoding complexity.

It is therefore desirable to develop an algorithm that efficiently generates lists based on the Log-MAP algorithm to achieve good performance with low complexity.

3.2 Proposed algorithms

Since soft decisions generated by the Log-MAP algorithm provide an accurate indication of the decision reliability, it is of interest to consider use of the Log-MAP algorithm in conjunction with the 2^M method (originally used by Nill and Sundberg in [43] to generate the list based on the symbol-wise soft decisions generated by the soft output Viterbi algorithm (SOVA)). Moreover, the 2^M method is used in this work together with the sub-block structure, where a data block before turbo encoding consists of several sub-blocks, each protected by an independent error detection code, as first proposed for turbo coded packet data transmissions in [44] [2].

List decoding with the 2^M method benefits from the sub-block structure. When errors occur in turbo codes, they spread across the entire frame rather than concentrating on a specific region as in convolutional codes. It means that for turbo codes, the number of erroneous bits within each sub-block will likely be smaller than that within the entire code block. Because the complexity of the 2^M method grows exponentially with the number of unreliable bits flipped, the complexity of list decoding with the 2^M method will be significantly reduced by flipping a small number of bits within each sub-block.

The proposed algorithm can be formalized as follows.

Let the information block be divided into E sub-blocks. For simplicity, assuming that K is divisible by E, each sub-block contains $\frac{K}{E}$ bits and the information binary sequence can be further expressed as $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{E-1})$, where $\mathbf{u}_i = (u_{i,0}, u_{i,1}, \cdots, u_{i,\frac{K}{E}-1})$, $i = 0, 1, \cdots, (E-1)$. Denote the soft LLR output for the whole code block from the iterative decoder as $\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_{E-1})$ where $\mathbf{s}_i = (s_{i,0}, s_{i,1}, \cdots, s_{i,\frac{K}{E}-1})$, and

$$s_{i,j} = \ln \frac{P(u_{i,j} = 1|\mathbf{y})}{P(u_{i,j} = 0|\mathbf{y})} \tag{3.3}$$

where $j = 0, 1, \cdots, (\frac{K}{E} - 1)$. Let $q_{i,m}$ denote the indexes of the m-th smallest magnitude of

[2] The sub-block structure will be discussed in detail in Chapter 4

the LLR outputs within the i-th sub-block

$$q_{i,m} = \mathop{\mathtt{argmin}}_{m=0, m \notin \{q_{i,0}, \cdots, q_{i,(m-1)}\}}^{\frac{K}{E}-1} \left(|s_{i,j}|\right). \tag{3.4}$$

The candidate list is constructed by flipping the $M = \log_2 L$ most unreliable bits within the sub-block as illustrated in Fig. 3.2. Let $\mathbf{z} = \mathtt{sign}(\mathbf{s})$ denote the hard decision and $(l)_{\text{bin},m}$ denote the m-th least significant bit (LSB) of the M bit binary representation of l, where $l = 0, 1, \cdots, L-1$. The l-th candidate for the decoded sequence in the list for the i-th sub-block can be represented as: $\hat{\mathbf{z}}_i^l = (\hat{z}_{i,0}^l, \hat{z}_{i,1}^l, \cdots, \hat{z}_{i,\frac{K}{E}-1}^l)$ and

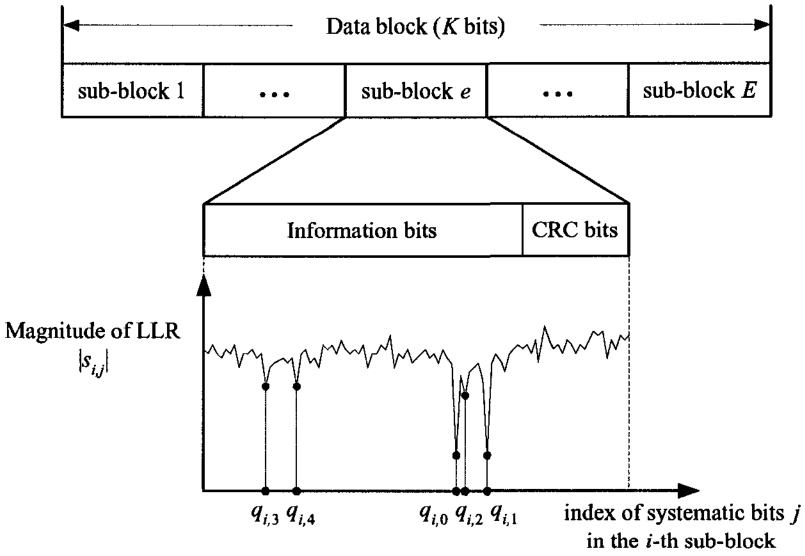
$$\hat{z}_{i,h}^l = \begin{cases} z_{i,h}, & \text{if } h \notin \{q_{i,0}, \cdots, q_{i,M-1}\} \\ (-1)^{(l)_{\text{bin},m}} \cdot z_{i,h}, & \text{if } h = q_{i,m}, m \in \{0, 1, \cdots, M-1\}. \end{cases} \tag{3.5}$$

In list decoding algorithms for turbo codes where the list is generated for one component code, the list is constructed based on the LLRs of the corresponding parity bits as well as the LLRs and *a priori* information on systematic bits, resulting in biased candidate codewords. In the new algorithm, the list is constructed based solely on the soft output of the turbo code, which is the LLR of the systematic bits. The bias introduced by parity codewords is avoided.

The arithmetic complexity of different algorithms for iterative decoding of turbo codes can be compared based on the number of the addition/subtraction and comparison (max) operations required to decode each information bit in one component decoder. The complexity required for the CRC check is not considered because it is common for all list decoding algorithms and can be implemented very efficiently.

The extra complexity introduced by the proposed new algorithm compared to the Log-MAP algorithm is the ranking of the magnitudes of the LLR outputs within each sub-block. For each information bit, this is an insertion operation in an ordered list with M elements, which requires, in the worst case, $(L-1) \cdot \lceil \log_2(L+1) \rceil$ comparison operations for a binary search algorithm [45], where "$\lceil \cdot \rceil$" denotes the minimum integer no less than "$\cdot$". This list generating process operates once per iteration.

Table 3.1 shows the arithmetic complexity of different algorithms for iterative decoding of a turbo code, where, as in [31] and [42], it is assumed that addition/subtraction has the same implementation cost as a comparison operation. Fig. 3.3 shows the relative increase of arithmetic complexity of different list decoding algorithms compared to the Max-log-MAP algorithm as a function of ν for list sizes of $L = 2, 8$, and 32. In Fig. 3.3 and other simulation results presented later, the new algorithm is denoted as "Log-MAP + Sub-block 2^M".



(a) Identify the most unreliable decisions within each sub-block.

$$
\begin{aligned}
\hat{\mathbf{z}}_i^0 &= (\ldots, z_{i,q_{i,3}}, \ldots, z_{i,q_{i,4}}, \ldots, z_{i,q_{i,0}}, \ldots, z_{i,q_{i,2}}, \ldots, z_{i,q_{i,1}}, \ldots) \\
\hat{\mathbf{z}}_i^1 &= (\ldots, z_{i,q_{i,3}}, \ldots, z_{i,q_{i,4}}, \ldots, -z_{i,q_{i,0}}, \ldots, z_{i,q_{i,2}}, \ldots, z_{i,q_{i,1}}, \ldots) \\
\hat{\mathbf{z}}_i^2 &= (\ldots, z_{i,q_{i,3}}, \ldots, z_{i,q_{i,4}}, \ldots, z_{i,q_{i,0}}, \ldots, z_{i,q_{i,2}}, \ldots, -z_{i,q_{i,1}}, \ldots) \\
\hat{\mathbf{z}}_i^3 &= (\ldots, z_{i,q_{i,3}}, \ldots, z_{i,q_{i,4}}, \ldots, -z_{i,q_{i,0}}, \ldots, z_{i,q_{i,2}}, \ldots, -z_{i,q_{i,1}}, \ldots) \\
\hat{\mathbf{z}}_i^4 &= (\ldots, z_{i,q_{i,3}}, \ldots, z_{i,q_{i,4}}, \ldots, z_{i,q_{i,0}}, \ldots, -z_{i,q_{i,2}}, \ldots, z_{i,q_{i,1}}, \ldots) \\
\hat{\mathbf{z}}_i^5 &= (\ldots, z_{i,q_{i,3}}, \ldots, z_{i,q_{i,4}}, \ldots, -z_{i,q_{i,0}}, \ldots, -z_{i,q_{i,2}}, \ldots, z_{i,q_{i,1}}, \ldots) \\
&\vdots \\
\hat{\mathbf{z}}_i^{31} &= (\ldots, -z_{i,q_{i,3}}, \ldots, -z_{i,q_{i,4}}, \ldots, -z_{i,q_{i,0}}, \ldots, -z_{i,q_{i,2}}, \ldots, -z_{i,q_{i,1}}, \ldots)
\end{aligned}
$$

(b) Construct the candidate codeword list by flipping these bits.

Figure 3.2: The proposed list decoding algorithm.

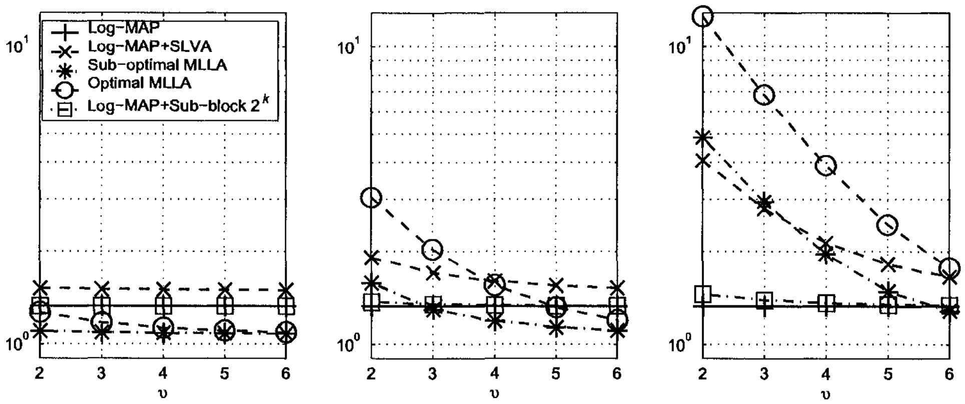
In actual list decoding systems, where the error detection code is not perfect, the probability of an accepted block error increases as the error detection redundancy decreases or the input BER to the error detector increases [46]. To reduce the probability of the accepted block error, a number of preliminary iterations could be performed to ensure a certain level of accuracy before the list decoding algorithm is activated.

3.3 Simulations

The list decoding algorithms under evaluation include Log-MAP + SLVA, Max-Log-MAP + SLVA, with performance equivalent to the optimal Max-log list algorithm, and the proposed new list decoding algorithm for turbo codes, together with the classical iterative decoding

Table 3.1: Arithmetic complexity for different decoding algorithms by means of arithmetic operations per bit per component decoder

Algorithm	Add/Sub	Max
Max-Log-MAP	$8 \times 2^\nu$	$4 \times 2^\nu$
Log-MAP	$12 \times 2^\nu + (L-2)$	$4 \times 2^\nu$
Log-MAP + SLVA	$13.5 \times 2^\nu + (L-2)/2$	$4.5 \times 2^\nu + (L-1) \cdot \lceil \log_2(L+1) \rceil / 2$
Sub-optimal MLLA	$8 \times 2^\nu$	$4 \times 2^\nu + (L-1) \cdot \lceil \log_2(L+1) \rceil + \max(2^v - L + 1, 0)$
Optimal MLLA	$8 \times 2^\nu + 3(L-1)$	$4 \times 2^\nu + (L-1) \cdot (2\lceil \log_2(L+1) \rceil + 3) + \max(2^v - L + 1, 0)$
Log-MAP + Sub-block 2^M	$12 \times 2^\nu$	$4 \times 2^\nu + (M-1) \cdot \lceil \log_2(M+1) \rceil / 2$



(a) L=2.

(b) L=8.

(c) L=32.

Figure 3.3: Relative increase of arithmetic complexity of list decoding algorithms compared to the Max-log-MAP algorithm vs. memory ν.

with the Log-MAP algorithm and the MAX-Log-MAP algorithm as references.

The turbo codes with a prime interleaver [22] and a random interleaver are considered. The 16-bit CRC-CCITT code with generator polynomial $G(D) = D^{16} + D^{12} + D^5 + 1$ is used for error detection [2].

The code block structure considered in the simulation is as follows:

- When the sub-block structure is used, one code block of length $K = 1760$ contains $E = 4$ sub-blocks, each with $K_{\text{info-s}} = 424$ information bits, corresponding to one asynchronous transfer mode (ATM) cell as in [30], and $K_{\text{CRC}} = 16$ CRC bits [3]. The code rate is $r = 0.3205$.

- When the sub-block structure is not used, a similar code block of length $K = 1760$ is considered to avoid performance differences inherently introduced by a different

[3] The length of an ATM cell is 53 bytes (424 bits), consisting of a 5-byte cell header and 48 bytes of payload. Strictly speaking, only the 48 payload bytes contain user data. However, in the context of the discussion, from the encoder point of view, all 53 bytes are treated as information bits.

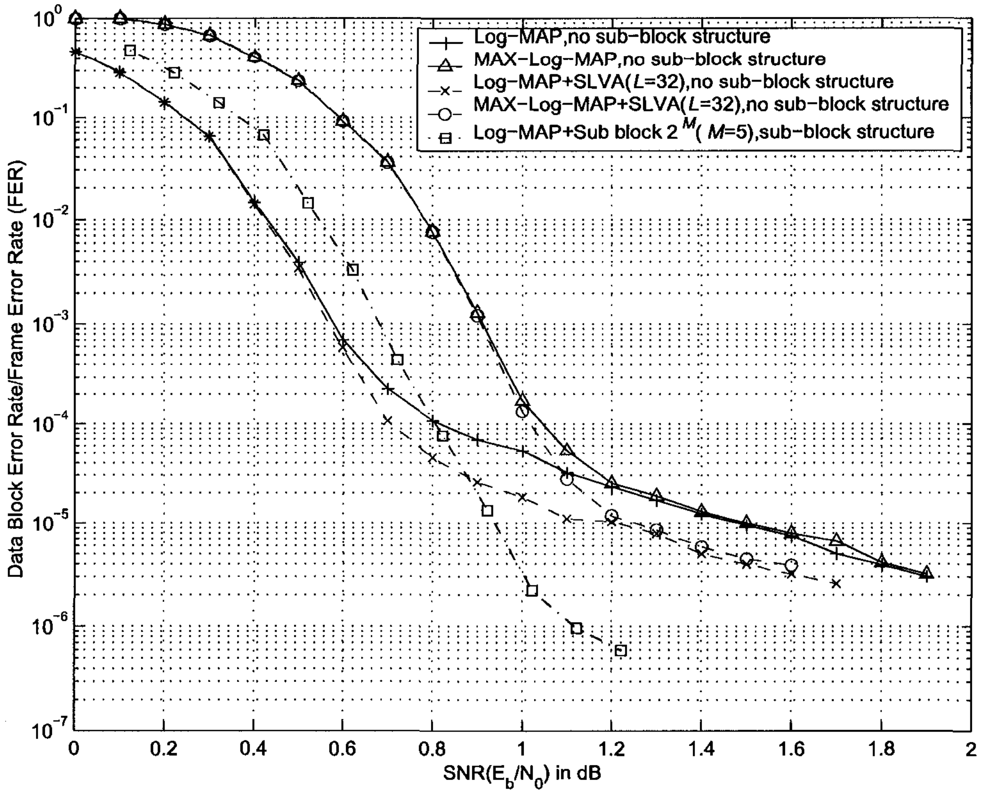


Figure 3.4: FER performance for the turbo code with prime interleaver.

turbo code interleaver. The code block consists of $K_{\text{info}} = 1744$ information bits and $K_{\text{CRC}} = 16$ CRC bits. The code rate is $r = 0.3296$.

In the simulations, a maximum of 32 iterations are allowed. The number of preliminary iterations is arbitrarily set to 9. After the preliminary iterations, list decoding is invoked repeatedly. For each simulation point, the simulation is stopped when 50 frame errors (code block errors) are accumulated.

With the sub-block structure, extra redundancy is introduced to make each sub-block self-error-detectable. Assuming that, without the sub-block structure, each code block has to include only one CRC, the SNR loss introduced by the CRC bits for the sub-block structure can be calculated as:

$$\text{SNR}_{\text{loss}} = 10 \cdot \log_{10} \frac{K_{\text{info}}}{K_{\text{info-s}} \cdot E} \text{ (dB)} \tag{3.6}$$

which, for the system under evaluation, equals an SNR loss of 0.1212 dB.

Figs. 3.4 and 3.5 show the FER comparisons for the prime and random interleavers. For both interleavers, the proposed algorithm lowers the error floor by more than one order of magnitude compared to Log-MAP + SLVA with the same list size. With a random

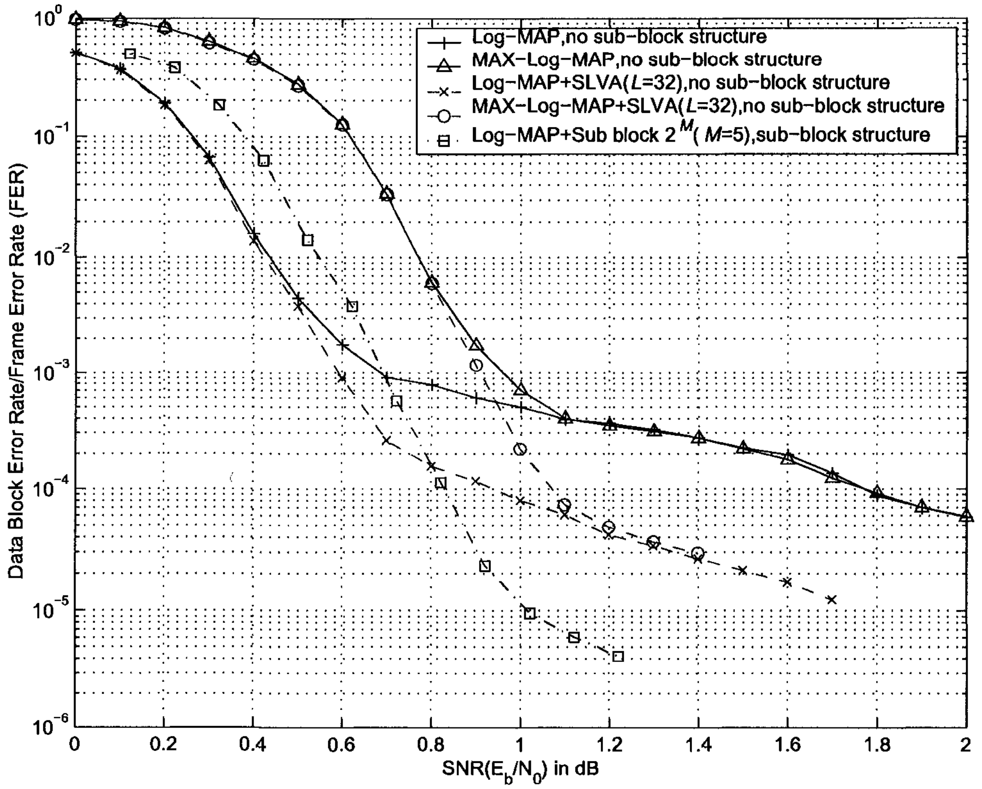


Figure 3.5: FER performance for the turbo code with random interleaver.

interleaver, relative error patterns that result in low weight RSC component codewords might also result in low weight turbo code codewords so that the traditional component list decoding algorithm is efficient in lowering the error floor. However, with a prime interleaver, where the relative error patterns that result in low weight codewords for the RSC component code do not result in low weight codewords for the turbo code, the efficiency of the traditional list decoding algorithm is reduced.

3.4 Chapter summary

Various properties of error events and weight spectra between turbo codes and their component RSCs explain the inefficiency of list decoding algorithms that use lists generated for the component code. A new algorithm which combines the 2^M method with the Log-MAP algorithm and the sub-block structure is proposed as a method to generate the list directly for the turbo code. The new algorithm introduces low extra complexity compared with the traditional Log-MAP algorithm and achieves better performance compared with previously proposed list decoding algorithms. Simulation results show that the new algorithm can lower the frame error floor by more than one order of magnitude compared to the algorithms reported in the literature. To reduce the probability of the accepted block error in list decoding for turbo codes, a suitable amount of redundancy for error detection should be included, the list size should be carefully chosen, and a preliminary number of iterations should be performed before the list decoding algorithm takes effect to reduce the BER before error detection.

Chapter 4

Sub-Block Recovery for Iterative Decoding of Turbo Codes with the Sub-Block Structure [1]

Turbo codes are characterized by a quickly falling BER and FER which form the so called turbo cliff in the low SNR region. Due to the fact that turbo codes usually contain a few low weight codewords, the BER and FER in the medium and high SNR regions decrease slower, usually exhibiting an error floor phenomenon. Generally speaking, as the data block length increases, turbo codes achieve better performance in both the turbo cliff region and the error floor region. Therefore, from the FEC point of view, it is desirable to employ large data blocks in a turbo encoded system.

In data transmission, delay is tolerable to some extent. To ensure reliable transmission in these systems, ARQ can be invoked when FEC fails to recover the transmitted data. For efficient retransmission, it is desirable to locate the positions of the errors in the data stream as accurately as possible and ask for retransmission of only the portion of the data that is of greatest importance to recovery of the erroneously decoded data. When the retransmission request concerns packets, shorter packets are preferable in order to avoid unnecessary retransmission of the portion of data that is already correctly decoded.

In a system where hybrid ARQ incorporating turbo codes is used for error control, a straightforward solution to the differing requirements of long data blocks for turbo codes and short packets for ARQ is the sub-block structure [22,44]. With this approach, one data block for turbo encoding contains several sub-blocks, each protected by an independent

[1] The material in this chapter was presented in part in "Analysis of a sub-block recovery scheme for decoding a concatenated error control code," at the *IEEE VTC 2005 Spring*, Stockholm, Sweden, May 2005 and "Sub-block recovery scheme for iterative decoding of turbo codes," at the *IEEE VTC 2005 Fall*, Dallas, USA, September 2005, and constitutes a paper accepted for publication in the *IEICE Transactions on Communications.*

error detection code. At the receiver, retransmissions are requested only for the sub-blocks in which errors are detected.

In order to make the sub-blocks self-error-detectable, a non-negligible amount of redundancy is introduced in the sub-block structure. It is therefore of interest to investigate how this extra redundancy can be utilized by the decoder to improve its performance. One scheme to improve the decoding performance was proposed in [47]. The scheme is based on the error fluctuation phenomenon observed in iterative decoding of turbo codes, where the number and locations of the errors vary occasionally from iteration to iteration. In that scheme, sub-block error detection is performed at every iteration. The hard decisions of the bits are recorded immediately after a sub-block has been determined error-free so that the decoder can catch the "best status" for each sub-block in the iterative decoding process. This leads to a performance improvement compared to conventional turbo decoding [47].

In this chapter, a novel sub-block recovery scheme is proposed that not only catches the "best decoding status" for each sub-block but also utilizes the correctly decoded sub-blocks to assist in the decoding of those not yet correctly decoded.

In Sections 4.1 - 4.3, the system model, the proposed sub-block recovery scheme and notations and definitions are introduced, respectively. In Sections 4.4 - 4.7, analysis is presented to show that the sub-block recovery scheme helps improve the performance of iterative decoding of turbo codes. In Section 4.8, simulation results are presented to demonstrate the performance improvement introduced by the sub-block recovery scheme, and the chapter is summarized in Section 4.9.

4.1 System model

The transmitter studied in this chapter consists of a CRC encoder, a turbo encoder, and a modulator. The information bits are first divided into transport blocks, each transport block is CRC encoded to form a sub-block, and several sub-blocks are combined into one data block and encoded by a turbo encoder [22, 44]. Encoding is followed by pulse shaping and modulation before transmission. To simplify the discussion, BPSK modulation with unit symbol energy $E_s = 1$ is assumed. Assuming that time slots for individual packet transmissions are short in comparison to the coherence time of the fading channel, the packet radio link can be modelled as an AWGN channel with noise variance per dimension equal to $N_0/2$.

At the receiver, the received signal is demodulated and normalized to unit energy per symbol, during which perfect channel estimation and synchronization are assumed. A soft

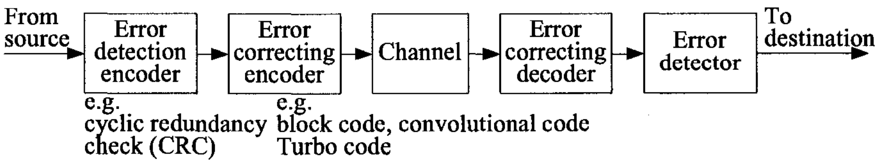


Figure 4.1: Concatenated error control coding scheme.

input hard output iterative decoder is used to decode the turbo code. After that, a CRC check is performed on the hard decisions to determine the correctness of each sub-block. Fig. 4.1 presents the block diagram of this concatenated error control scheme. Traditionally, the inner code and the outer code are decoded separately, and therefore the redundancy introduced by the error detection code is not utilized by the inner decoder for the error correction code.

4.2 New scheme

In this section, a sub-block recovery scheme is proposed for the system described in the previous section in order to improve the decoding performance of the turbo code. After a few initial iterations, a CRC check is performed on each sub-block after each iteration of turbo decoding. If at least one but not all sub-blocks satisfy the CRC check, the hard decisions on the data bits of these sub-blocks are recorded, and the LLRs corresponding to these hard decisions are constructed and forwarded to the next stage in the iterative decoder. The constructed LLRs assume the sign of the corresponding hard decisions and a pre-defined high magnitude, which represents very high reliability of these decoder inputs. As a result, after sub-block recovery, the recovered part of the decoder input looks noise-free in the following iterations. This sub-block recovery process is invoked repeatedly in the following iterations when there are new sub-blocks that satisfy the CRC check and there is still at least one sub-block that does not. During the initial iterations, sub-block recovery is not invoked in order to avoid the potentially high probability of undetected error events of the CRC code.

In the following analysis, the decoding scheme involving sub-block recovery is referred to as the *new scheme*. In comparison, in a *conventional scheme*, as described in [47], after the same number of initial iterations as that in the new scheme, the hard decisions for the sub-blocks that satisfy the CRC check are recorded in subsequent iterations to catch the "best status", but no sub-block recovery is performed. Note that, assuming the probability of the undetected packet error is zero, the conventional scheme will never perform worse than the iterative decoding scheme without recording the hard decisions in the iterative decoding process.

The flow charts for both the new scheme and the conventional scheme are shown in Fig. 4.2. Compared to the conventional scheme, in the new scheme, after the 'Record the hard decisions' procedure, a new procedure called "Recover the systematic inputs" is performed.

4.3 Notation and definitions

Let ENC1 and ENC2 denote the RSC component encoders of the turbo code, respectively, and let Π denote the interleaver of length K.

A turbo code can be treated as an (N, K) linear block code, where N is the length of each codeword. Let $\mathbf{u} = (u_0, u_1, \cdots, u_{K-1})$ and $\mathbf{v} = (v_0, v_1, \cdots, v_{N-1})$ denote the binary input sequence (data block) and output codeword of the encoder, respectively. For notational simplicity, in this chapter it is assumed that $v_k = u_k$ for $k \in \{0, 1, \cdots, K-1\}$. Let $\mathbf{x} = (2\mathbf{v} - 1)$ denote the BPSK modulated symbols, and $\mathbf{y} = \mathbf{x} + \mathbf{n}$ the received symbols, where $\mathbf{n} = (n_0, n_1, \cdots, n_{N-1})$, and n_i are i.i.d. $\sim N(0, N_0/2)$.

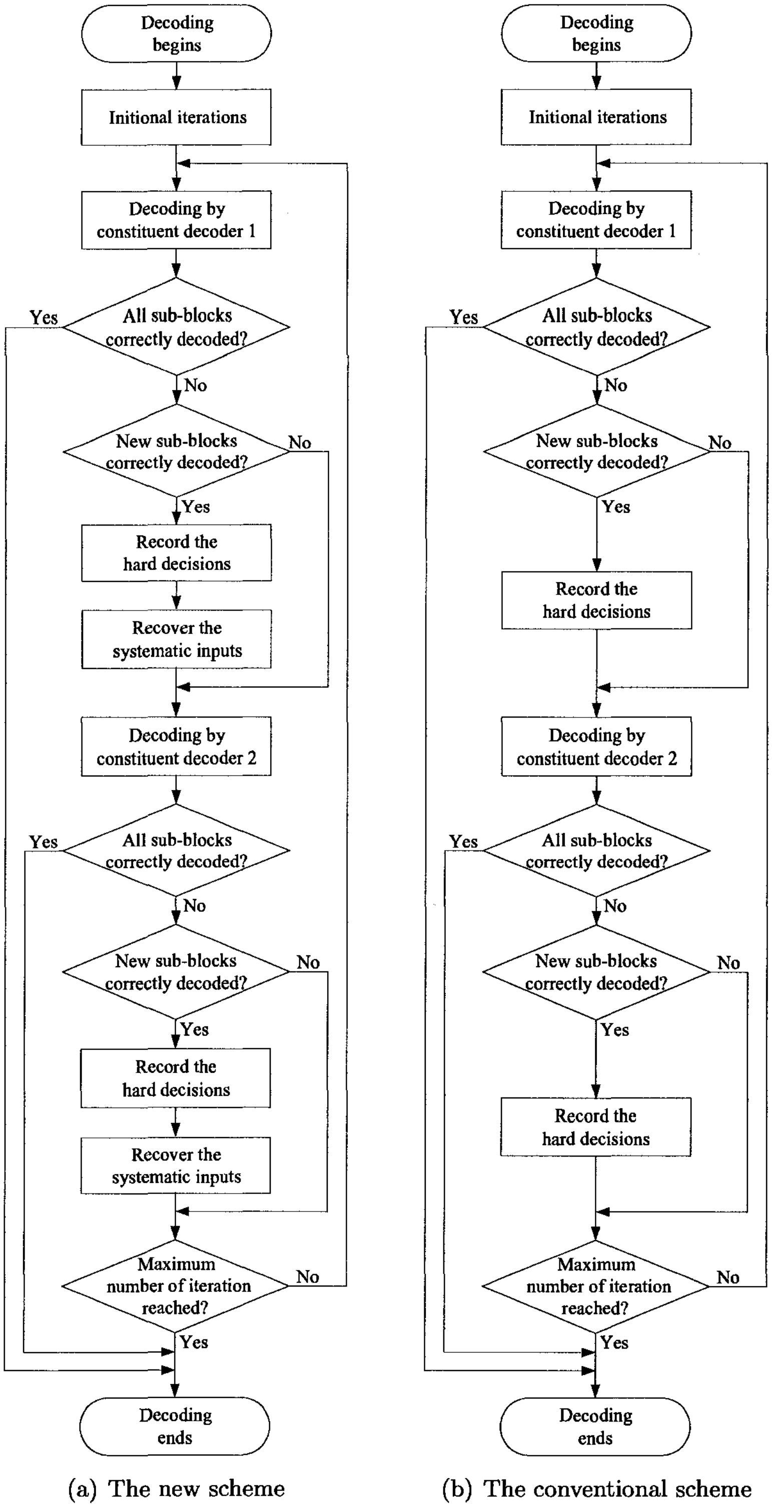
Denote the binary representation of a sequence of length K as $\mathbf{u}^p = (u_0^p, u_1^p, \cdots, u_{K-1}^p) = (p)_{\text{bin}}$, where p is an integer $p \in \{0, 1, \cdots, 2^K - 1\}$. If $\mathbf{u}^p$ is fed into the encoder, the corresponding codeword and the antipodal modulated symbol sequence are denoted as $\mathbf{v}^p$ and $\mathbf{x}^p$, respectively. Without loss of generality, it is assumed that the all-zero codeword, $\mathbf{v}^0$, corresponding to the all-zero data block $\mathbf{u}^0$, is transmitted. Finally, the interleaved data block associated with $\mathbf{u}$ is denoted as $\tilde{\mathbf{u}}$ and the interleaving and deinterleaving functions are denoted as $\Pi(\mathbf{u}) = \tilde{\mathbf{u}}$ and $\Pi^{-1}(\tilde{\mathbf{u}}) = \mathbf{u}$, respectively.

In an iterative turbo decoder, two SISO constituent decoders, DEC1 and DEC2, work cooperatively via exchange of extrinsic information [19]. One full iteration is characterized by one decoding operation of each constituent decoder with DEC2 following DEC1. The input to the iterative turbo decoder consists of LLRs of the channel observations: $\mathbf{L}_{(y)} = \{L_{(y)_0}, L_{(y)_1}, \cdots, L_{(y)_{N-1}}\}$, where

$$L_{(y)_n} = \ln \frac{P(v_n = 1|y_n)}{P(v_n = 0|y_n)} = \frac{4}{N_0} y_n, \text{ for } n \in \{0, 1, \cdots, N-1\} \tag{4.1}$$

for the AWGN channel. Each constituent decoder takes in the LLRs of the channel observation for the corresponding systematic and parity bits, as well as the *a priori* information $\mathbf{L}_{(a)} = \{L_{(a)_0}, L_{(a)_1}, \cdots, L_{(a)_{K-1}}\}$, where

$$L_{(a)_k} = \ln \frac{P(x_k = 1)}{P(x_k = -1)}, \text{ for } k \in \{0, 1, \cdots, K-1\}. \tag{4.2}$$


Decoding begins
Initional iterations
Decoding by constituent decoder 1
All sub-blocks correctly decoded?
Yes
No
New sub-blocks correctly decoded?
No
Yes
Record the hard decisions
Recover the systematic inputs
Decoding by constituent decoder 2
All sub-blocks correctly decoded?
Yes
No
New sub-blocks correctly decoded?
No
Yes
Record the hard decisions
Recover the systematic inputs
Maximum number of iteration reached?
No
Yes
Decoding ends
Decoding begins
Initional iterations
Decoding by constituent decoder 1
All sub-blocks correctly decoded?
Yes
No
New sub-blocks correctly decoded?
No
Yes
Record the hard decisions
Decoding by constituent decoder 2
All sub-blocks correctly decoded?
Yes
No
New sub-blocks correctly decoded?
No
Yes
Record the hard decisions
Maximum number of iteration reached?
No
Yes
Decoding ends

(a) The new scheme

(b) The conventional scheme

Figure 4.2: Comparison of the flow charts for the new scheme and the conventional one.

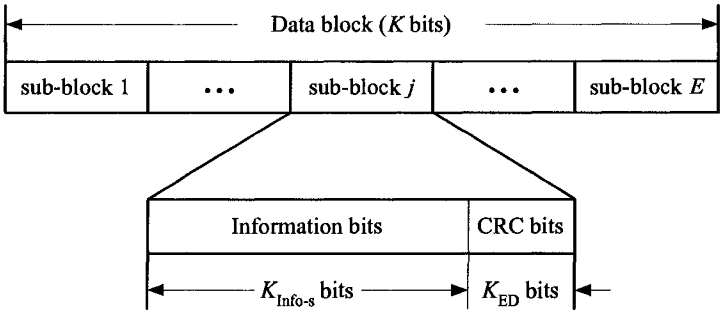


Figure 4.3: The sub-block structure.

The *a priori* information is the extrinsic information generated by the other constituent decoder, which at its output is denoted $\mathbf{L}_{(e)} = \{L_{(e)_0}, L_{(e)_1}, \cdots, L_{(e)_{K-1}}\}$. Based on the LLRs of the channel observation and extrinsic information, the Log-MAP algorithm or the SOVA is used to estimate the transmitted codeword [19, 48, 49].

Let $\mathbf{L}_{(e^{1,i})}$ and $\mathbf{L}_{(e^{2,i})}$ denote the extrinsic information from DEC1 and DEC2 at the i-th iteration, respectively. Similarly let $\mathbf{L}_{(a^{1,i})}$ and $\mathbf{L}_{(a^{2,i})}$ denote the *a priori* information to DEC1 and DEC2. Moreover, let $\hat{\mathbf{u}}^i = (\hat{u}_0^i, \hat{u}_1^i, \cdots, \hat{u}_{K-1}^i)$ denote the hard decisions made by the turbo decoder at the i-th iteration, and let $\mathbf{L}_{(\hat{u}^i)} = \{L_{(\hat{u}^i)_0}, L_{(\hat{u}^i)_1}, \cdots, L_{(\hat{u}^i)_{K-1}}\}$ be the corresponding LLRs. Let $\hat{\mathbf{u}}^{1,i}$ and $\hat{\mathbf{u}}^{2,i}$ denote the hard decisions made by DEC1 and DEC2, respectively, and let $\mathbf{L}_{(\hat{u}^{1,i})}$, $\mathbf{L}_{(\hat{u}^{2,i})}$ be the corresponding LLRs. With the sub-block structure, each data block contains E sub-blocks. Assuming, for simplicity, that K is divisible by E, each sub-block contains $\frac{K}{E}$ bits, within which K_{ED} bits are parity checks for error detection, and the other $K_{\text{info-s}} = (\frac{K}{E} - K_{\text{ED}})$ bits are information bits, as shown in Fig. 4.3. Finally, let SB_e denote the set of indices of the symbols belonging to the e-th sub-block in the input code block $\mathbf{u}$.

4.4 Analysis of the sub-block recovery scheme

The analysis of the sub-block recovery scheme is presented in two parts. In this section, a method is proposed to transform the performance comparison of iterative decoding with and without sub-block recovery to the performance comparison of the decoding of two simple codes. In Sections 4.5 and 4.6 it is shown that the sub-block recovery scheme results in improved performance with the iterative decoding of turbo codes using either the SOVA or the Log-MAP algorithms. In Section 4.7 it is shown that if ML decoding is used, sub-block recovery does not improve decoding performance.

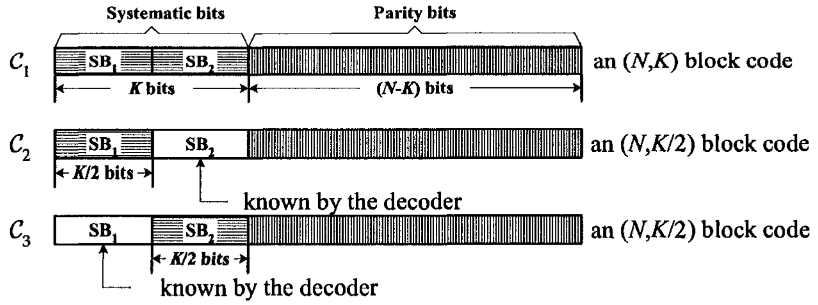


Figure 4.4: Three codes for analysis of the sub-block recovery scheme.

Basic analysis

In the following, the case where there are two sub-blocks within each data block is considered. The sub-blocks are denoted as SB_1 and SB_2. Let $\mathrm{P}_{\mathrm{SB}_j}(\mathrm{BE})$ denote the BER of the bits within sub-block SB_j ($j = 1$ or 2), and let $\mathrm{P}_{\mathrm{SB}_j}(\mathrm{PE})$ denote the sub block error rate of SB_j. Let A_j^i and $\overline{A_j^i}$ denote the events that SB_j is or is not correctly decoded at the i-th iteration during the iterative decoding process, where $i \in [0, (I-1)]$ and I is the maximum number of iterations allowed in the decoding process. Events A_j^i and $\overline{A_j^i}$ are mutually exclusive.

To facilitate the analysis, three codes are defined, as shown in Fig. 4.4. Let $\mathcal{C}_1$ denote the original inner code in Fig. 4.1, which is an (N, K) block code. When the inner decoder decodes a received sequence, in which the first sub-block contains information bits and the second sub-block is known to the decoder, this is equivalent to decoding an $(N, \frac{K}{2})$ code, denoted as $\mathcal{C}_2$. Similarly, when the second sub-block contains information bits and the first sub-block is known to the decoder, this is equivalent to decoding a different rate $(N, \frac{K}{2})$ code, denoted as $\mathcal{C}_3$.

In the following, the decoding process in the $(i+1)$-th iteration is considered. In the conventional scheme, the decoder decodes code $\mathcal{C}_1$ in every iteration. In the new scheme, if after the i-th iteration, SB_2 is detected to be error-free and SB_1 is determined to contain errors, a sub-block recovery process will be triggered. In the $(i+1)$-th iteration, the decoder knows the information bits contained in SB_2 and only wants to estimate the information bits in SB_1. The decoder is equivalently decoding code $\mathcal{C}_2$ rather than $\mathcal{C}_1$. Similarly, if after the i-th iteration SB_1 is detected to be error-free and SB_2 is determined to contain errors, during the $(i+1)$-th iteration the decoder is equivalently decoding code $\mathcal{C}_3$ rather than $\mathcal{C}_1$.

As an example, consider the mutually exclusive events A_2^i and $\overline{A_2^i}$. From the total probability theorem, the BER of the bits within SB_1 in the $(i+1)$-th iteration is

$$\mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}) = \mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}|\overline{A_2^i}) \cdot \mathrm{P}(\overline{A_2^i}) + \mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}|A_2^i) \cdot \mathrm{P}(A_2^i) \tag{4.3}$$

where $\mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}|\overline{A_2^i})$ is the BER of the bits within sub-block SB_1 in the $(i+1)$-th iteration given that sub-block SB_2 is not correctly decoded in the i-th iteration.

For the conventional scheme,

$$\mathrm{P}_{\mathrm{SB}_1}^{\mathrm{Conventional}}(\mathrm{BE}) = \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|\overline{A_2^i}) \cdot \mathrm{P}(\overline{A_2^i}) + \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|A_2^i) \cdot \mathrm{P}(A_2^i) \tag{4.4}$$

where the superscript "$\mathcal{C}_1$" indicates that the probability refers to the decoding of $\mathcal{C}_1$. The same notation rule applies in the following analysis.

With sub-block recovery, if SB_2 is correctly decoded and SB_1 has errors after the i-th iteration, then during the $(i+1)$-th iteration

$$\mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}|A_2^i) = \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_2}(\mathrm{BE}). \tag{4.5}$$

For the new scheme it then follows,

$$\mathrm{P}_{\mathrm{SB}_1}^{\mathrm{New}}(\mathrm{BE}) = \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|\overline{A_2^i}) \cdot \mathrm{P}(\overline{A_2^i}) + \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_2}(\mathrm{BE}) \cdot \mathrm{P}(A_2^i). \tag{4.6}$$

Comparing the performance of the new scheme with that of the conventional scheme reduces to comparing the error probability $\mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_2}(\mathrm{BE})$ with $\mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|A_2^i)$. When $\mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_2}(\mathrm{BE}) < \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|A_2^i)$,

$$\mathrm{P}_{\mathrm{SB}_1}^{\mathrm{New}}(\mathrm{BE}) < \mathrm{P}_{\mathrm{SB}_1}^{\mathrm{Conventional}}(\mathrm{BE}), \tag{4.7}$$

and the new scheme will result in improvement in the BER performance compared to the conventional scheme.

If instead of A_2^i and $\overline{A_2^i}$, events A_1^i and $\overline{A_1^i}$ are considered, the BER analysis of the bits within SB_2 in the $(i+1)$-th iteration is similar to the discussion above with the conclusion that the improvement of the new scheme depends on whether or not $\mathrm{P}_{\mathrm{SB}_2}^{\mathcal{C}_3}(\mathrm{BE}) < \mathrm{P}_{\mathrm{SB}_2}^{\mathcal{C}_1}(\mathrm{BE}|A_1^i)$.

Extended analysis

In the simple example discussed above, all the possible events regarding whether SB_1 or SB_2 is correctly decoded in the i-th iteration, $A_1^iA_2^i$, $A_1^i\overline{A_2^i}$, $\overline{A_1^i}A_2^i$, $\overline{A_1^iA_2^i}$, are mutually exclusive, with $\mathrm{P}(A_1^iA_2^i \cup A_1^i\overline{A_2^i} \cup \overline{A_1^i}A_2^i \cup \overline{A_1^iA_2^i}) = 1$. From the total probability theorem, the BER of the bits within SB_2 in the $(i+1)$-th iteration is

$$\mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}) \;=\; \mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}|A_1^iA_2^i)\mathrm{P}(A_1^iA_2^i) + \mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}|A_1^i\overline{A_2^i})\mathrm{P}(A_1^i\overline{A_2^i})$$

$$+\mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}|\overline{A_1^i}A_2^i)\mathrm{P}(\overline{A_1^i}A_2^i) + \mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}|\overline{A_1^i A_2^i})\mathrm{P}(\overline{A_1^i A_2^i}). \tag{4.8}$$

For the conventional scheme,

$$\begin{aligned}\mathrm{P}_{\mathrm{SB}_1}^{\mathrm{Conventional}}(\mathrm{BE}) &= \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|A_1^i A_2^i)\mathrm{P}(A_1^i A_2^i) + \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|A_1^i\overline{A_2^i})\mathrm{P}(A_1^i\overline{A_2^i}) \\ &\quad +\mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|\overline{A_1^i}A_2^i)\mathrm{P}(\overline{A_1^i}A_2^i) + \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|\overline{A_1^i A_2^i})\mathrm{P}(\overline{A_1^i A_2^i}). \end{aligned} \tag{4.9}$$

With sub-block recovery, if SB$_2$ is correctly decoded and SB$_1$ has errors after the i-th iteration, then during the $(i+1)$-th iteration

$$\mathrm{P}_{\mathrm{SB}_1}(\mathrm{BE}|\overline{A_1^i}A_2^i) = \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_2}(\mathrm{BE}|\overline{A_1^i}). \tag{4.10}$$

For the new scheme, it then follows that

$$\begin{aligned}\mathrm{P}_{\mathrm{SB}_1}^{\mathrm{New}}(\mathrm{BE}) &= \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|A_1^i A_2^i)\mathrm{P}(A_1^i A_2^i) + \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|A_1^i\overline{A_2^i})\mathrm{P}(A_1^i\overline{A_2^i}) \\ &\quad +\mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_2}(\mathrm{BE}|\overline{A_1^i})\mathrm{P}(\overline{A_1^i}A_2^i) + \mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|\overline{A_1^i A_2^i})\mathrm{P}(\overline{A_1^i A_2^i}). \end{aligned} \tag{4.11}$$

Note that (4.9) and (4.11) differ in the third term on the right hand side. Similarly, one can write down the counterparts of (4.8) - (4.11) for the BER of the bits within SB$_2$ in the $(i+1)$-th iteration for both the conventional scheme and the new scheme. The only difference would be the calculation of the BER for the event that SB$_1$ is correctly decoded and SB$_2$ has errors after the i-th iteration, i.e.,

$$\mathrm{P}_{\mathrm{SB}_2}(\mathrm{BE}|A_1^i\overline{A_2^i}) = \mathrm{P}_{\mathrm{SB}_2}^{\mathcal{C}_1}(\mathrm{BE}|A_1^i\overline{A_2^i}) \tag{4.12}$$

for the conventional scheme, and

$$\mathrm{P}_{\mathrm{SB}_2}(\mathrm{BE}|A_1^i\overline{A_2^i}) = \mathrm{P}_{\mathrm{SB}_2}^{\mathcal{C}_3}(\mathrm{BE}|\overline{A_2^i}) \tag{4.13}$$

for the new scheme.

Comparing the performance of the new scheme with that of the conventional scheme reduces to comparing the following two pairs of conditional BERs:

- $\mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_2}(\mathrm{BE}|\overline{A_1^i})$ vs. $\mathrm{P}_{\mathrm{SB}_1}^{\mathcal{C}_1}(\mathrm{BE}|\overline{A_1^i}A_2^i)$, and
- $\mathrm{P}_{\mathrm{SB}_2}^{\mathcal{C}_3}(\mathrm{BE}|\overline{A_2^i})$ vs. $\mathrm{P}_{\mathrm{SB}_2}^{\mathcal{C}_1}(\mathrm{BE}|A_1^i\overline{A_2^i})$.

This analysis can be extended to cases where one data block contains more than two sub-blocks. It also applies to cases where the sub-block recovery process is invoked more than once during the iterative decoding process. Note that (4.3) - (4.13) are still valid after substituting all the $\mathrm{P}_{\mathrm{SB}_j}(\mathrm{BE})$ by the $\mathrm{P}_{\mathrm{SB}_j}(\mathrm{PE})$. Therefore, the above analysis is also applicable to sub-block error rate analysis.

4.5 Analysis of the sub-block recovery scheme for iterative turbo decoding with modified SOVA

In this section, the modified SOVA is first reviewed and it is then demonstrated that sub-block recovery helps correct a typical error pattern that is inherent in turbo codes with highly structured interleavers such as those used in 3GPP and 3GPP2 systems [22].

Iterative turbo decoding with modified SOVA

As described in [48, 49], during the i-th iteration of the modified SOVA, DEC1 yields the estimation of the transmitted codeword that maximizes the sequence-wise *a posteriori* function

$$\mathrm{P}^{1,i}(\mathbf{u}|\mathbf{y}_{\mathrm{sys,par1}}) = \mathrm{P}(\mathbf{x}_{\mathrm{sys,par1}}|\mathbf{y}_{\mathrm{sys,par1}}) = \frac{\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}}|\mathbf{x}_{\mathrm{sys,par1}})\mathrm{P}^{1,i}(\mathbf{x}_{\mathrm{sys}})}{\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}})} \tag{4.14}$$

where $\mathbf{y}_{\mathrm{sys,par1}}$ denotes the portion of received symbol sequence corresponding to the systematic bits and the parity bits of ENC1, $\mathbf{x}_{\mathrm{sys,par1}}$ denotes the transmitted symbols corresponding to the systematic bits and the parity bits of ENC1, and $\mathbf{x}_{\mathrm{sys}}$ denotes the transmitted symbols corresponding to the systematic bits.

The superscript "$1, i$" is used to denote variables in DEC1 in the i-th iteration[2]. The metric between $\mathbf{x}^p$ and $\mathbf{y}$ is defined as

$$\begin{aligned} \mathtt{M}^{p}_{\mathrm{dec1},i} &= \sum_{n\in\mathbf{n}_s} x^p_n \cdot (L_{(a^{1,i})_n} + L_{(y)_n}) + \sum_{n\in\mathbf{n}_{p1}} x^p_n \cdot L_{(y)_n} \\ &= 2\ln\frac{1}{\kappa_1}\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}}|\mathbf{x}^p_{\mathrm{sys,par1}})\mathrm{P}^{1,i}(\mathbf{x}^p_{\mathrm{sys}}) \\ &= 2\ln\frac{1}{\kappa_1}\mathrm{P}^{1,i}(\mathbf{u}|\mathbf{y}_{\mathrm{sys,par1}})\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}}) \end{aligned} \tag{4.15}$$

where $\mathbf{n}_s$ and $\mathbf{n}_{p^1}$ denote the set of indices of the systematic bits and the parity bits of ENC1 in codeword $\mathbf{v}$, respectively. For a given *a priori* information sequence and a given received symbol sequence, κ_1 is a constant independent of $\mathbf{x}_{\mathrm{sys,par1}}$. Moreover, for a given $\mathbf{y}_{\mathrm{sys,par1}}$, $\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}})$ is a constant independent of $\mathbf{x}^p_{\mathrm{sys,par1}}$ and $\exp(\cdot)$ is a monotonically increasing function, therefore the decision rule of DEC1 in the i-th iteration when signal corruption is due to the AWGN channel is

$$\hat{\mathbf{u}}^{1,i} = \mathbf{u}^{p^{1,i}_{\mathrm{APP}}} \tag{4.16}$$

where

$$p^{1,i}_{\mathrm{APP}} = \operatorname*{argmax}_{p=0}^{2^K-1} \mathrm{P}^{1,i}(\mathbf{u}^p|\mathbf{y}_{\mathrm{sys,par1}})$$

[2] Correspondingly, in the following, "$2, i$" denote variables in DEC2 in the i-th iteration.

$$
\begin{aligned}
&= \operatorname*{argmax}_{p=0}^{2^K-1} \mathrm{P}(\mathbf{y}_{\text{sys,par1}}|\mathbf{x}^p_{\text{sys,par1}})\mathrm{P}^{1,i}(\mathbf{x}^p_{\text{sys}}) \\
&= \operatorname*{argmax}_{p=0}^{2^K-1} \mathtt{M}^p_{\text{dec1},i}. \qquad (4.17)
\end{aligned}
$$

The *metric difference* associated with two binary sequences, $\mathbf{u}^{p_a}$ and $\mathbf{u}^{p_b}$, for DEC1 in the i-th iteration is defined as

$$
\Delta^{(p_a,p_b)}_{\text{dec1},i} = \mathtt{M}^{p_a}_{\text{dec1},i} - \mathtt{M}^{p_b}_{\text{dec1},i}. \qquad (4.18)
$$

Eq. (4.17) can be alternatively expressed as

$$
p^{1,i}_{\text{APP}} = p_a|\forall p_b \in \{0, 1, \cdots, 2^K - 1\},\ p_a \neq p_b;\ \Delta^{(p_a,p_b)}_{\text{dec1},i} \geq 0. \qquad (4.19)
$$

As a SISO component decoder, DEC1 in the i-th iteration produces the symbol-wise LLR of its decisions, $\hat{u}^{1,i}_k$, as

$$
\begin{aligned}
L_{(\hat{u}^{1,i})_k} &= 2\ln \frac{\max_{\mathbf{u}^{p_a}:u^{p_a}_k=1} \mathrm{P}^{1,i}(\mathbf{u}^{p_a}|\mathbf{y}_{\text{sys,par1}})}{\max_{\mathbf{u}^{p_a}:u^{p_a}_k=0} \mathrm{P}^{1,i}(\mathbf{u}^{p_a}|\mathbf{y}_{\text{sys,par1}})} \\
&= (2u_k^{p^{1,i}_{\text{APP}}} - 1) \min_{\substack{u^{p_a}_k = 1;\, u^{p_b}_k = 0 \\ p_a = p^{1,i}_{\text{APP}} \text{ or } p_b = p^{1,i}_{\text{APP}}}} \left|\Delta^{(p_a,p_b)}_{\text{dec1},i}\right| \qquad (4.20)
\end{aligned}
$$

where the coefficient 2 is introduced for consistency with the metric evaluations.

The extrinsic information produced by DEC1 and DEC2 in the i-th iteration can be evaluated by

$$
\begin{aligned}
L_{(e^{1,i})_k} &= L_{(\hat{u}^{1,i})_k} - L_{(y)_k} - L_{(a^{1,i})_k} \qquad (4.21) \\
L_{(e^{2,i})_k} &= L_{(\hat{u}^{2,i})_k} - L_{(y)_k} - L_{(a^{2,i})_k} \qquad (4.22)
\end{aligned}
$$

where the *a priori* information input to DEC1 and DEC2 in the i-th iteration is the extrinsic information generated by the other constituent decoder in the previous decoding step, i.e., $L_{(a^{1,i})_k} = L_{(e^{2,i-1})_{\Pi(k)}}$, $L_{(a^{2,i})_{\Pi(k)}} = L_{(e^{1,i})_k}$.

Analysis of the sub-block recovery scheme

In the following, it is shown how the sub-block recovery scheme helps to improve the decoding performance by analyzing the decoding of a typical *error pattern*, denoted by (22:22) in [50], that a turbo decoder has difficulty correcting. In this chapter, an *error pattern* is defined as the symbol-wise modulo-2 sum of the encoder input sequence $\mathbf{u}$ and the erroneously decoded sequence $\hat{\mathbf{u}}$.

The (22:22) error pattern is inherent in turbo codes with structured interleavers which are often used in real systems which use different block lengths for different services. Structured interleavers have the advantage that the interleaving pattern can be generated on-the-fly rather than stored in memory for all possible data block lengths. Interleaver design for turbo codes has been extensively investigated in [34,51,52]. However, as pointed out in [50], (22:22) error patterns are inherent in highly structured interleavers and the multiplicity of such error patterns cannot be reduced. As will be shown later, the sub-block recovery scheme helps to correct this error pattern, therefore, improvements in decoder performance is expected when sub-block recovery is used.

The typical (22:22) error pattern contains four erroneous bits. Assuming the source sequence is all-zero, these errors have value one. Their indices in the data block $\mathbf{u}$ are denoted as P1 - P4, and those in the interleaved data block, $\tilde{\mathbf{u}}$, as Π(P1)-Π(P4). Let SB_1 and SB_2 denote the sub-blocks that contain errors that the turbo decoder has difficulty correcting. It is assumed that

- positions P1, P2 are in SB_1, and positions P3, P4 are in SB_2;
- sequence $\mathbf{u}^{\text{EP1}}$ with errors in P1 and P2 results in a low weight codeword when fed into ENC1;
- sequence $\mathbf{u}^{\text{EP2}}$ with errors in P3 and P4 results in a low weight codeword when fed into ENC1;
- sequence $\mathbf{u}^{\text{EP3}}$ with errors in Π(P2) and Π(P3) results in a low weight codeword when fed into ENC2;
- sequence $\mathbf{u}^{\text{EP4}}$ with errors in Π(P1) and Π(P4) results in a low weight codeword when fed into ENC2.

Fig. 4.5 illustrates the positions of the bits in the typical error pattern in both the original input sequence $\mathbf{u}$ and the interleaved input sequence $\tilde{\mathbf{u}}$. The input sequence with zeros in all but these four positions results in a low weight turbo codeword. Consider the situation when the transmitted symbols x_{P2} and x_{P3} experience adverse noise and the corresponding LLR reliability inputs to the decoder, $L_{(y)_{\text{P2}}}$ and $L_{(y)_{\text{P3}}}$, satisfy

$$-\left(-L_{(y)_{\max}} - L_{(y)_{\text{P2}}}\right) \gg 0, \tag{4.23}$$

$$-\left(-L_{(y)_{\max}} - L_{(y)_{\text{P3}}}\right) \gg 0, \tag{4.24}$$

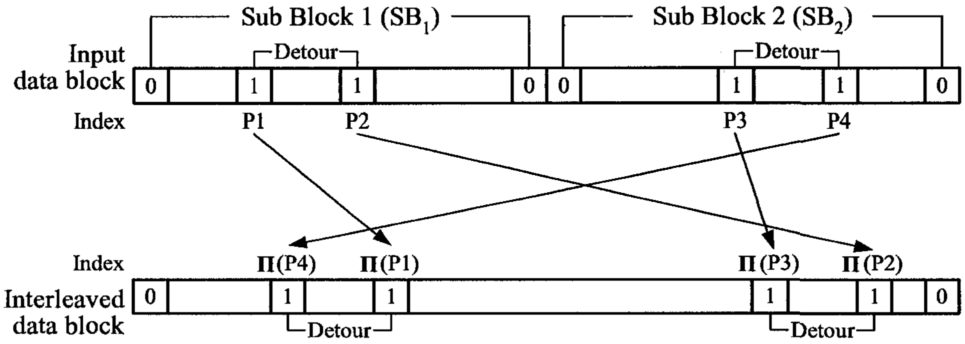


Figure 4.5: The typical error pattern under consideration.

where $(-L_{(y)_{\max}})$ is the maximum LLR value for y_{P2} and y_{P3} when those two symbols are received free of noise. It is assumed that, with the conventional scheme, DEC1 corrects the errors in positions P3 and P4, but not those in P1 and P2 during the i-th iteration, and that DEC2 corrects the errors in positions Π(P1) and Π(P4) but not those in Π(P2) and Π(P3), all with small margins[3], which means

$$\begin{cases} \Delta_{\text{dec1},i}^{(\text{EP1},p_a)} > 0 & \forall p_a \neq \text{EP1} \\ \Delta_{\text{dec1},i}^{(0,p_a)} > 0 & \forall p_a \neq 0,\ \text{EP1} \end{cases} \tag{4.25}$$

$$\begin{cases} \Delta_{\text{dec2},i}^{(\text{EP3},p_a)} > 0 & \forall p_a \neq \text{EP3} \\ \Delta_{\text{dec2},i}^{(0,p_a)} > 0 & \forall p_a \neq 0,\ \text{EP3} \end{cases} \tag{4.26}$$

and $\Delta_{\text{dec1},i}^{(\text{EP1},0)}$, $\Delta_{\text{dec2},i}^{(\text{EP3},0)}$ are small values. The hard decisions of DEC1 and DEC2 at the i-th iteration are $\hat{\mathbf{u}}^{1,i} = \mathbf{u}^{\text{EP1}}$ and $\hat{\mathbf{u}}^{2,i} = \mathbf{u}^{\text{EP3}}$, respectively. It is further assumed that with the conventional scheme, this situation does not change in the $(i+1)$-th iteration, i.e., (4.25) and (4.26) are also valid when the "i" in the subscript is substituted by "$i+1$".

Impact of sub-block recovery on the hard decisions made by DEC2 in the i-th iteration

In the new scheme, after the decoding of DEC1 in the i-th iteration and under the assumption that SB$_2$ is correctly decoded, sub-block recovery is performed. The systematic LLR inputs to the iterative decoder corresponding to SB$_2$ are set to values representing the hard decisions. The superscript " $'$ " is used to denote the variables evaluated in the new scheme and to distinguish them from the ones that are evaluated in the conventional scheme. For DEC2, the LLR input for $y_{\Pi(\text{P3})}$ that is changed to $L_{(y)_{\Pi(\text{P3})}}{}' = -L_{(y)_{\max}}$ after sub-block

[3] Here it is assumed that DEC2 corrects the errors in positions Π(P1) and Π(P4) to simplify the analysis. However, if DEC2 does not correct the errors in positions Π(P1) and Π(P4), the following analysis is also valid, as discussed later.

recovery is particularly important since this change will impact the immediate subsequent decoding.

With the new scheme, the metric difference associated with $\mathbf{u}^{\mathrm{EP3}}$ and $\mathbf{u}^0$ evaluated by DEC2 in the i-th iteration is

$$\Delta^{(\mathrm{EP3},0)'}_{\mathrm{dec2},i} = \Delta^{(\mathrm{EP3},0)}_{\mathrm{dec2},i} + 2(L_{(y)\Pi(\mathrm{P3})}{}' - L_{(y)\Pi(\mathrm{P3})}) = \Delta^{(\mathrm{EP3},0)}_{\mathrm{dec2},i} + 2(-L_{(y)\max} - L_{(y)\Pi(\mathrm{P3})}). \quad (4.27)$$

The difference introduced by sub-block recovery is $2(-L_{(y)\max} - L_{(y)\Pi(\mathrm{P3})})$. Based on the previous assumption in (4.24) that $y_{\Pi(\mathrm{P3})}$ is significantly distorted by noise,

$$\Delta^{(\mathrm{EP3},0)}_{\mathrm{dec2},i} + 2(-L_{(y)\max} - L_{(y)\Pi(\mathrm{P3})}) < 0 \quad (4.28)$$

from which it follows that

$$\Delta^{(0,\mathrm{EP3})'}_{\mathrm{dec2},i} = -\Delta^{(\mathrm{EP3},0)'}_{\mathrm{dec2},i} > 0. \quad (4.29)$$

Also, from (4.24) and (4.26),

$$\Delta^{(0,p_a)'}_{\mathrm{dec2},i} = \Delta^{(0,p_a)}_{\mathrm{dec2},i} - \sum_{n \in \mathbf{n}_{\mathrm{SB}_2,p_a}} 2(-L_{(y)\max} - L_{(y)\Pi(n)}) > 0 \ \ \forall p_a \neq 0, \ \mathrm{EP3} \quad (4.30)$$

where $\mathbf{n}_{\mathrm{SB}_2,p_a}$ is the set of indices of the non-zero systematic bits that are in sequence $\mathbf{u}^{p_a}$ and belong to SB_2. From (4.19), (4.29) and (4.30), it follows that the hard decision made by DEC2 will be $\mathbf{u}^0$ rather than $\mathbf{u}^{\mathrm{EP3}}$, i.e.,

$$\hat{\mathbf{u}}^{2,i}{}' = \mathbf{u}^0. \quad (4.31)$$

Impact of sub-block recovery on the extrinsic information produced by DEC2 in the i-th iteration

To examine the extrinsic information for y_{P2} in the new scheme, sequences with the maximum metric for $u^{p_a}_{\Pi(\mathrm{P2})} = 1$ and $u^{p_b}_{\Pi(\mathrm{P2})} = 0$ for DEC2 in the i-th iteration with both the conventional scheme and the new scheme are considered. With the conventional scheme, the all-zero sequence $\mathbf{u}^0$ is the sequence with the maximum metric for $u^{p_b}_{\Pi(\mathrm{P2})} = 0$, and $\mathbf{u}^{\mathrm{EP3}}$ is the sequence with the maximum metric for $u^{p_a}_{\Pi(\mathrm{P2})} = 1$.

$$\begin{aligned} L_{(e^{2,i})\Pi(\mathrm{P2})} &= L_{(\hat{u}^{2,i})\Pi(\mathrm{P2})} - L_{(y)\Pi(\mathrm{P2})} - L_{(a^{2,i})\Pi(\mathrm{P2})} \\ &= \Delta^{(\mathrm{EP3},0)}_{\mathrm{dec2},i} - L_{(y)\Pi(\mathrm{P2})} - L_{(a^{2,i})\Pi(\mathrm{P2})}. \end{aligned} \quad (4.32)$$

First, the improvement introduced by the new scheme to the metric for different sequences is examined. For the all-zero sequence,

$$\mathtt{M}^0_{\mathrm{dec2},i}{}' = \mathtt{M}^0_{\mathrm{dec2},i} + \sum_{n \in \mathbf{n}_{\mathrm{SB}_2}} (-1)\left(-L_{(y)\max} - L_{(y)\Pi(n)}\right) \quad (4.33)$$

where the second term is the sum of LLR improvement on systematic bits introduced by sub-block recovery. This term has a positive value and is denoted by κ_2 in the following.

For any other sequences $\forall p^b \neq 0$,

$$\mathrm{M}^{p^b}_{\mathrm{dec2},i}{}' = \mathrm{M}^{p^b}_{\mathrm{dec2},i} + \kappa_2 + 2 \sum_{n \in \mathbf{n}_{\mathrm{SB}_2,p^b}} \left(-L_{(y)\max} - L_{(y)\Pi(n)}\right) \tag{4.34}$$

where the last term on the right hand side is negative. It is clear from (4.33) and (4.34) that the improvement introduced by sub-block recovery on the metric associated with a sequence is proportional to the number of positions in which that sequence agrees with the transmitted sequence in the recovered part. Specifically, in the case under consideration, the all zero sequence receives the most significant improvement on its metric.

Therefore, for any sequence, $\mathbf{u}^{\mathrm{EPx}} : u^{\mathrm{EPx}}_{\Pi(\mathrm{P2})} = 0$, other than the all-zero sequence,

$$\mathrm{M}^{0}_{\mathrm{dec2},i}{}' = \mathrm{M}^{0}_{\mathrm{dec2},i} + \kappa_2 > \mathrm{M}^{\mathrm{EPx}}_{\mathrm{dec2},i} + \kappa_2 > \mathrm{M}^{\mathrm{EPx}}_{\mathrm{dec2},i}{}' \tag{4.35}$$

which indicates that the all-zero sequence $\mathbf{u}^0$ is the one with the maximum metric for $u^{p^b}_{\Pi(\mathrm{P2})} = 0$ for DEC2 in the i-th iteration. However, $\mathbf{u}^{\mathrm{EP3}}$ is not necessarily the sequence with the maximum metric for $u^{p^a}_{\Pi(\mathrm{P2})} = 1$.

- If $\mathbf{u}^{\mathrm{EP3}}$ is still the sequence with the maximum metric for $u^{p^a}_{\Pi(\mathrm{P2})} = 1$, then the improvement introduced by the new scheme on the extrinsic information for $y_{\Pi(\mathrm{P2})}$ produced by DEC2 in the i-th iteration is

$$L_{(e^{2,i})\Pi(\mathrm{P2})}{}' - L_{(e^{2,i})\Pi(\mathrm{P2})} = 2(L_{(y)\Pi(\mathrm{P3})}{}' - L_{(y)\Pi(\mathrm{P3})}) = 2(-L_{(y)\max} - L_{(y)\Pi(\mathrm{P3})}) < 0. \tag{4.36}$$

- Assume that after sub-block recovery, another sequence, denoted as $\mathbf{u}^{\mathrm{EP5}}$, rather than $\mathbf{u}^{\mathrm{EP3}}$ has the maximum metric for $u^{p^a}_{\Pi(\mathrm{P2})} = 1$. From (4.26), before sub-block recovery, the metric associated with $\mathbf{u}^{\mathrm{EP5}}$ from the received sequence is smaller than that of $\mathbf{u}^{\mathrm{EP1}}$ as well as that of $\mathbf{u}^{\mathrm{EP3}}$, i.e., $\mathrm{M}^{\mathrm{EP5}}_{\mathrm{dec2},i} < \mathrm{M}^{\mathrm{EP1}}_{\mathrm{dec2},i} < \mathrm{M}^{\mathrm{EP3}}_{\mathrm{dec2},i}$. It can be shown that

$$L_{(e^{2,i})\Pi(\mathrm{P2})}{}' - L_{(e^{2,i})\Pi(\mathrm{P2})} = (\mathrm{M}^{\mathrm{EP5}}_{\mathrm{dec2},i} - \mathrm{M}^{\mathrm{EP3}}_{\mathrm{dec2},i}) + \sum_{n \in \mathbf{n}_{\mathrm{SB}_2,\mathrm{EP5}}} 2(-L_{(y)\max} - L_{(y)\Pi(n)}) < 0 \tag{4.37}$$

where $\mathbf{n}_{\mathrm{SB}_2,\mathrm{EP5}}$ is the set of indices of the non-zero systematic bits that are in the sequence $\mathbf{u}^{\mathrm{EP5}}$ and belong to SB_2.

From (4.36) and (4.37), the extrinsic information for y_{P2} produced by DEC2 brings the decoder closer toward the correct decision, independent of whether DEC2 makes a favorable decision regarding $\mathbf{u}^0$ or not.

Following a similar procedure, it can be shown that the improvement introduced by the new scheme on the extrinsic information for $y_{\Pi(\mathrm{P1})}$ produced by DEC2 in the i-th iteration is also always toward the correct decision, i.e.,

$$L_{(e^{2,i})_{\Pi(\mathrm{P1})}}' - L_{(e^{2,i})_{\Pi(\mathrm{P1})}} < 0. \tag{4.38}$$

Impact of sub-block recovery on the hard decisions made by DEC1 in the $i+1$-th iteration

The improvement introduced by the new scheme on the metric difference associated with $\mathbf{u}^{\mathrm{EP1}}$ and $\mathbf{u}^0$ evaluated by DEC1 in the $(i+1)$-th iteration is

$$\begin{aligned} \Delta_{\mathrm{dec1},i+1}^{(\mathrm{EP1},0)\;\prime} - \Delta_{\mathrm{dec1},i+1}^{(\mathrm{EP1},0)} &= 2(L_{(a^{1,i+1})_{\mathrm{P1}}}' - L_{(a^{1,i+1})_{\mathrm{P1}}}) + 2(L_{(a^{1,i+1})_{\mathrm{P2}}}' - L_{(a^{1,i+1})_{\mathrm{P2}}}) \\ &= 2(L_{(e^{2,i})_{\Pi(\mathrm{P1})}}' - L_{(e^{2,i})_{\Pi(\mathrm{P1})}}) + 2(L_{(e^{2,i})_{\Pi(\mathrm{P2})}}' - L_{(e^{2,i})_{\Pi(\mathrm{P2})}}) \\ &< 0. \end{aligned} \tag{4.39}$$

The second term in the second last row on the right hand side of (4.39) is negative, as indicated in (4.37); following a procedure similar to that presented above, it can be shown that the first term is also negative.

From the previous assumption in (4.25), if

$$-[2(L_{(e^{2,i})_{\Pi(\mathrm{P1})}}' - L_{(e^{2,i})_{\Pi(\mathrm{P1})}}) + 2(L_{(e^{2,i})_{\Pi(\mathrm{P2})}}' - L_{(e^{2,i})_{\Pi(\mathrm{P2})}})] > \Delta_{\mathrm{dec1},i+1}^{(\mathrm{EP1},0)}, \tag{4.40}$$

it follows that

$$\Delta_{\mathrm{dec1},i+1}^{(0,\mathrm{EP1})\;\prime} = -\Delta_{\mathrm{dec1},i+1}^{(\mathrm{EP1},0)} > 0. \tag{4.41}$$

In this case, DEC1 will make a favorable decision toward $\mathbf{u}^0$ rather than $\mathbf{u}^{\mathrm{EP1}}$. Thus, the influence of the sub-block recovery scheme can propagate to the unrecovered sub-blocks and will help correct the remaining errors in the unrecovered sub-blocks.

In the above analysis, it is assumed that DEC2 can correct the errors in positions $\Pi(\mathrm{P1})$ and $\Pi(\mathrm{P4})$. If this is not true, the change of LLR input for y_{P4} will cause changes similar to those expressed in (4.27-4.39). If the decoder is experiencing difficulties correcting more than one (22:22) error pattern that satisfies the assumptions used in the above analysis, sub-block recovery will introduce similar effects toward the correction of the corresponding erroneous decisions.

4.6 Analysis of the sub-block recovery scheme for iterative decoding with the Log-MAP algorithm

In this section, it is shown that when the systematic bits are recovered with high probability, then with the Log-MAP algorithm, sub-block recovery results in virtually pruned trellis connections which results in improvement to the decoding process.

The Log-MAP algorithm

In each component decoder, the Log-MAP algorithm evaluates the LLR for each decision according to [19] as

$$\begin{aligned} L(\hat{u})_k &= \ln \frac{\sum_{\mathbf{S}_1} \alpha_{h,k} \gamma(s_k, s_{k+1}) \beta_{h',k+1}}{\sum_{\mathbf{S}_0} \alpha_{h,k} \gamma(s_k, s_{k+1}) \beta_{h',k+1}} \\ &= \max_{\mathbf{S}_1}{}^{*}[\bar{\alpha}_{h,k} + \bar{\gamma}(s_k, s_{k+1}) + \bar{\beta}_{h',k+1}] \\ &\quad - \max_{\mathbf{S}_0}{}^{*}[\bar{\alpha}_{h,k} + \bar{\gamma}(s_k, s_{k+1}) + \bar{\beta}_{h',k+1}] \qquad (4.42) \end{aligned}$$

where $\mathbf{S}_1 : \{(s_k = S_h) \rightarrow (s_{k+1} = S_{h'}) : u_k = 1\}$ and $\mathbf{S}_0 : \{(s_k = S_h) \rightarrow (s_{k+1} = S_{h'}) : u_k = 0\}$ are the sets of state transitions caused by input information bits "1" and "0", respectively, and $\bar{\alpha}_{h,k} = \ln \alpha_{h,k}$, $\bar{\beta}_{h',k} = \ln \beta_{h',k}$ and $\bar{\gamma}(s_k, s_{k+1}) = \ln \gamma(s_k, s_{k+1})$. Function $\max^*$ is defined as [19]

$$\max{}^*(x, y) = \log(e^x + e^y) = \max(x, y) + f_c(|y - x|) \qquad (4.43)$$

where $f_c(|y-x|) = \log[1+\exp(-|x-y|)]$ is the correction function. This correction function can be implemented by a look-up table with limited number of entries to reduce complexity while achieving a good performance [53]. $\bar{\alpha}_{h,k}$ and $\bar{\beta}_{h,k}$ for every state S_h at instance k are evaluated as

$$\begin{aligned} \bar{\alpha}_{h,k} &= \max_{s_{k-1} \in \mathbf{A}}{}^{*}[\bar{\alpha}_{h',k-1} + \bar{\gamma}(s_{k-1}, s_k)] \qquad (4.44) \\ \bar{\beta}_{h,k} &= \max_{s_{k+1} \in \mathbf{B}}{}^{*}[\bar{\beta}_{h',k+1} + \bar{\gamma}(s_k, s_{k+1})] \qquad (4.45) \end{aligned}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the sets of states $s_{k-1} = S_{h'}$ and $s_{k+1} = S_{h'}$ that are connected to state s_k, respectively.

For the rate 1/2 binary RSC component code,

$$\bar{\gamma}(s_k, s_{k+1}) = \frac{1}{2}[(L_{(a)_k} + L_{(y)_{k,0}})x_{k,0} + L_{(y)_{k,1}} x_{k,1}] \qquad (4.46)$$

where $x_{k,0}$ and $x_{k,1}$ are the modulated symbols corresponding to the systematic bit and the parity bit output from the encoder in the k-th state transition interval, respectively. $L_{(y)_{k,0}}$ and $L_{(y)_{k,1}}$ are the corresponding LLR inputs to the decoder.

The impact of the sub-block recovery scheme

Based on the trellis structure of a convolutional code and assuming that the LLRs and the extrinsic information input to the decoder have the same dynamic range, it is straightforward to show that in the $\bar{\alpha}$ updating process, as well as the $\bar{\beta}$ updating process, the difference between the largest and the smallest $\bar{\alpha}_{h,k}$ at any instance k is bounded by

$$\max_h \bar{\alpha}_{h,k} - \min_h \bar{\alpha}_{h,k} \leq 2T[(\eta + 1)R_{\text{iv}} + f_{c,\text{MAX}}]. \tag{4.47}$$

In (4.47), T is the minimum number of state transitions required for any state at instance k and any state at instance $k - T$ to have a path connecting each other, R_{iv} is the dynamic range of the internal variables, including $\bar{\alpha}$, $L_{(a)}$ and $L_{(y)}$, η is the number of symbols output at each state transition interval and $f_{c,\text{MAX}}$ is the maximum value the correction function in (4.43) can assume. By appropriately choosing the value for the recovered input LLRs corresponding to the systematic bits in the state transition intervals where the sub-block recovery occurs, the recovered input LLRs for the systematic bits will, with a high probability, dominate the max* function in (4.44) and (4.45). As a result, the state transitions corresponding to the systematic bits that differ from the correct decisions are virtually pruned from the trellis, and the trellis structure is simplified. After the sub-block recovery process, the negative contributions made by the paths through the trellis, which contain systematic bits that do not match the correct decisions, will be reduced for the LLRs evaluated for the so far incorrectly decoded bits. At the same time, the contributions made by paths containing systematic bits that match the correct decisions will be increased.

In the following, a simple four-state RSC encoder is used as an example to illustrate the trellis prunning effect. The trellis of the encoder with generator polynomial $\text{G} = (1, 7/5)_8$ with the input information bit and the output encoded bits for each branch is shown in Fig. 4.6(a). If the systematic LLR input for a time instant k is recovered, the trellis will be simplified to one of the trellises shown in Fig. 4.6(b) or (c), depending on whether the hard decision is "0" or "1". In Fig. 4.6(b) or (c), the branches indicated by dotted lines will be pruned from the trellis if the recovered input LLRs for the systematic bits dominate the max* function in (4.44) and (4.45). In this case, only the branches indicated by solid lines, representing the possible state transition corresponding to the correct decisions, remain valid.

4.7 ML decoding

The ML decoder yields the estimate of the transmitted codeword that maximizes the likelihood function, $\text{P}(\mathbf{y}|\mathbf{u})$.

When signal corruption is due to the AWGN, the metric between $\mathbf{x}^p$ and $\mathbf{y}$ can be defined as

$$\mathtt{M}^p = \sum_{n=0}^{N-1} x_n^p \cdot y_n = \sigma^2 \ln \frac{1}{\kappa_3}\text{P}(\mathbf{y}|\mathbf{u}), \tag{4.48}$$

where κ_3, for a given received symbol sequence, is a constant independent of $\mathbf{x}$.

Because $\exp(\cdot)$ is a monotonically increasing function, the decision rule of an ML decoder for this signal corruption is

$$\hat{\mathbf{u}} = \mathbf{u}^{p_{\mathrm{ML}}} \tag{4.49}$$

where

$$p_{\mathrm{ML}} = \operatorname*{argmax}_{p=0}^{2^K-1} \mathrm{P}(\mathbf{y}|\mathbf{u}^p) = \operatorname*{argmax}_{p=0}^{2^K-1} \mathrm{P}(\mathbf{y}|\mathbf{x}^p) = \operatorname*{argmax}_{p=0}^{2^K-1} \mathtt{M}^p. \tag{4.50}$$

The metric difference associated with two binary sequences $\mathbf{u}^{p_a}$ and $\mathbf{u}^{p_b}$ is:

$$\Delta^{(p_a,p_b)} = \mathtt{M}^{p_a} - \mathtt{M}^{p_b}. \tag{4.51}$$

Eq. (4.50) can be further expressed as

$$p_{\mathrm{ML}} = p_a \mid \forall p_b \in \{0, 1, \cdots, 2^K - 1\},\ p_a \neq p_b;\ \Delta^{(p_a,p_b)} = \mathtt{M}^{p_a} - \mathtt{M}^{p_b} \geq 0. \tag{4.52}$$

When the additive noise in the channel is so significant that it makes the received sequence closer to a valid codeword other than the transmitted all-zero sequence, the ML decoder will make a wrong decision. This event can be expressed as

$$\Delta^{(p_{\mathrm{ML}},0)} = \mathtt{M}^{p_{\mathrm{ML}}} - \mathtt{M}^0 > 0. \tag{4.53}$$

If several sub-blocks are correctly decoded in the ML decision sequence, sub-block recovery can be performed in the sense that the channel observations for the correctly decoded sub-blocks can be replaced by the hard decisions with the maximum allowable magnitude, and then the decoding can be performed one more time. The metric difference between the previous ML decision sequence and the all-zero sequence in the first decoding attempt can be expressed as

$$\begin{aligned}\Delta^{(p_{\mathrm{ML}},0)} &= \sum_{n=1}^{N} (x_n^{P_{\mathrm{ML}}} - x_n^0) \cdot y_n' \\ &= \sum_{n \in \mathbf{n}_{s,\mathrm{R}}} (x_n^{P_{\mathrm{ML}}} - x_n^0) \cdot y_n + \sum_{n \in \mathbf{n}_{s,\mathrm{NR}}} (x_n^{P_{\mathrm{ML}}} - x_n^0) \cdot y_n \\ &\quad + \sum_{n \in \mathbf{n}_p} (x_n^{P_{\mathrm{ML}}} - x_n^0) \cdot y_n \end{aligned} \tag{4.54}$$

where,

- $\mathbf{n}_{s,\mathrm{R}}$ are the positions of the systematic bits in the recovered sub-blocks;
- $\mathbf{n}_{s,\mathrm{NR}}$ are the positions of the systematic bits in the unrecovered sub-blocks;

- $\mathbf{n}_p$ are the positions of the parity bits.

Only the inputs corresponding to the bits with indices in $\mathbf{n}_{s,\mathrm{R}}$ are recovered by the sub-block recovery process. They are contained in the first summation term on the right hand side of (4.54). Therefore, the improvement introduced by sub-block recovery can be expressed as

$$\Delta^{(P_{\mathrm{ML}},0)'} - \Delta^{(P_{\mathrm{ML}},0)} = \sum_{n=1}^{N} (x_n^{P_{\mathrm{ML}}} - x_n^0)(y_n' - y_n) = 0, \qquad (4.55)$$

since for the correctly decoded sub-blocks, $x_n^{P_{\mathrm{ML}}} = x_n^0$. Therefore, the sub-block recovery process will not introduce changes to the decisions made by the decoder, and will have no influence in correcting the remaining errors in the other sub-blocks.

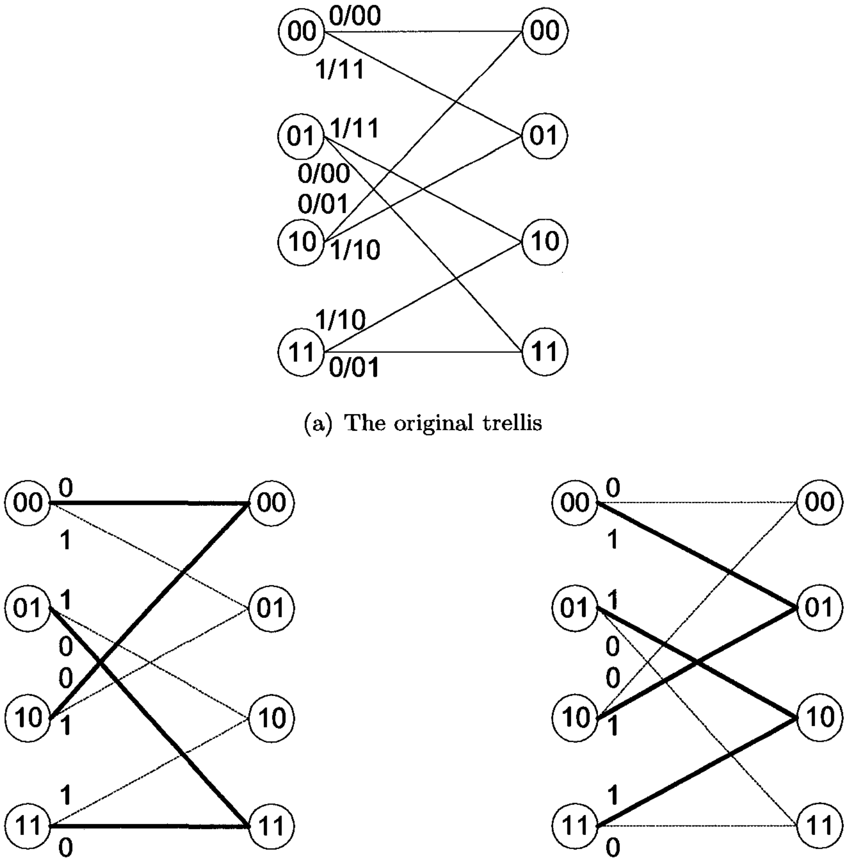
From the above analysis, it is evident that for a systematic code with ML decoding such as a convolutional code decoded with VA, sub-block recovery will not result in improvement in performance. This analysis of sub-block recovery for ML decoding is confirmed by simulations for convolutional codes with the VA.

4.8 Simulations

The performance of the conventional scheme and the new scheme for iterative turbo decoding with both the SOVA and the Log-MAP algorithm is compared for the AWGN channel.

The turbo code and the 16-bit CRC-CCITT that were described in Section 3.3 are used in the simulations. A data block length of $K = 1760$ is used for all simulations to exclude the effect of different interleaver lengths on performance.

Cases are considered where one data block contains $E = 4$ and 32 sub-blocks. Although dividing the data block of length $K = 1760$ into so many sub-blocks is impractical in a real system, these results suggest potential improvements in cases where larger data blocks are used and can be divided into many sub-blocks. As has been mentioned in Section 3.2, a number of preliminary iterations should be performed before invoking sub-block recovery, because the CRC checks are relied on for determining the correctness of each sub-block and during the first few iterations, the hard decisions might be so unreliable that the probability of undetected error of the CRC check becomes an issue. The number of preliminary iterations should be determined taking into consideration the length of the CRC check, the undetected error rate, and the block error rate. This problem is beyond the scope of this chapter although it deserves careful investigation in the future. Instead, in the simulations, a conservative approach is taken to arbitrarily choose a long CRC check code, as specified above and the relatively large number of preliminary iterations of 9. Therefore,



(a) The original trellis

(b) After sub-block recovery, hard decision"0"

(c) After sub-block recovery, hard decision "1"

Figure 4.6: Trellis for a G = $(1, 7/5)_8$ RSC encoder.

both the hard decision recording in the conventional scheme and the sub-block recovery process in the new scheme are active from the 10-th iteration. In the simulations, constant improvements introduced by sub-block recovery are observed as the number of maximum iterations increases. The simulation results are for the case where a maximum of $I = 100$ iterations is allowed. Although this number is very high compared to those used in current systems, it is chosen for this case to show the potential improvements achievable by using sub-block recovery. For each test point, the simulation is stopped when 50 data block errors are accumulated.

The data block error rate vs. E_b/N_0 achieved with the Log-MAP algorithm for both the $E = 4$ and $E = 32$ cases are shown in Fig. 4.7 and 4.8, respectively, where E_b is the energy corresponding to an input data bit to the turbo encoder. From Fig. 4.7 and 4.8, it is observed that with the new scheme, a lower error floor is achieved compared to the conventional scheme for both the $E = 4$ and the $E = 32$ cases, where the improvement in the latter case is more significant. The improvement in coding gain is about 0.2 dB and 0.4 dB at a data block error rate of 10^{-5}, respectively. Fig. 4.8 also shows that sub-block recovery introduces improvement in the turbo cliff region when the number of sub-blocks is large. The improvement is about 0.15 dB at a data block error rate of 10^{-3}.

Fig. 4.9 depicts the sub-block error rate vs. E_b/N_0 for $E = 32$ with both the SOVA and the Log-MAP algorithms. The results for a maximum number of iterations of both 25 (dashed lines) and 100 (solid lines) iterations are presented. From Fig. 4.9, it is observed that as the maximum number of iterations increases from 25 to 100, the performance in the error floor region for the new scheme does not change, whereas the performance in the turbo cliff region improves significantly. Fig. 4.9 also shows that when the new scheme is used, the gap between the performance for the SOVA and the Log-MAP algorithms is smaller compared to that when the conventional scheme is used, especially when a large number of iterations is allowed.

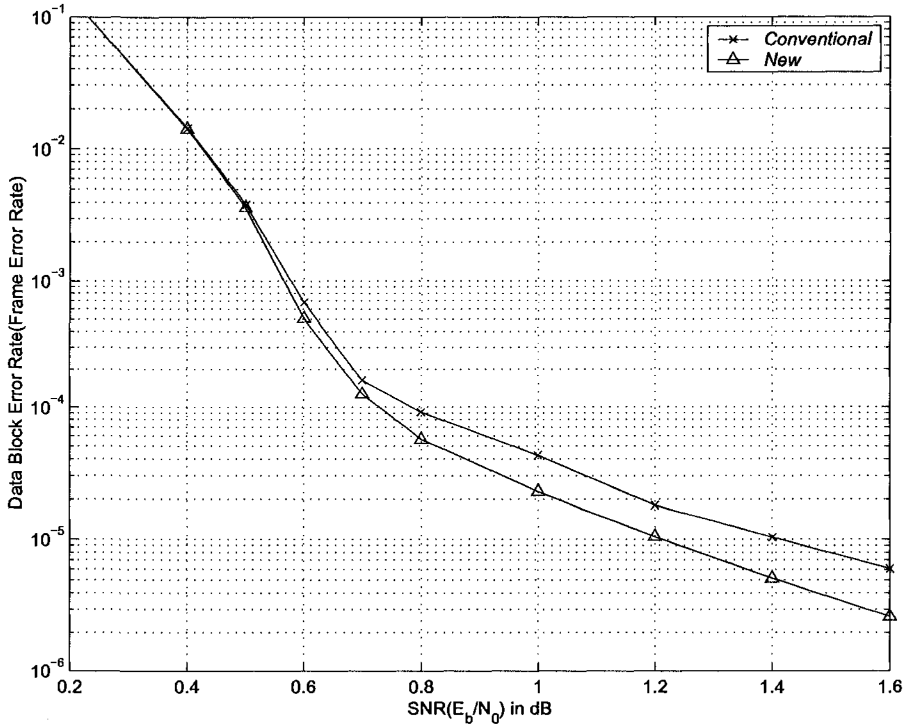


Figure 4.7: Data block error rate in AWGN channel, Log-MAP algorithm, 9 preliminary iterations, maximum 100 iterations, E=4 sub-blocks per data block.

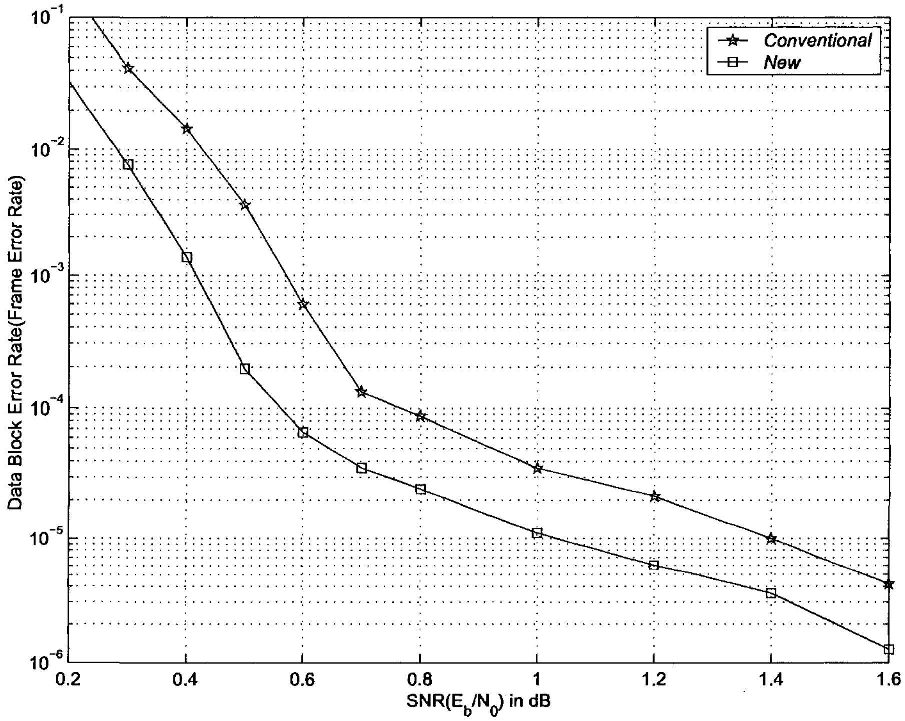


Figure 4.8: Data block error rate in AWGN channel, Log-MAP algorithm, 9 preliminary iterations, maximum 100 iterations, E=32 sub-blocks per data block.

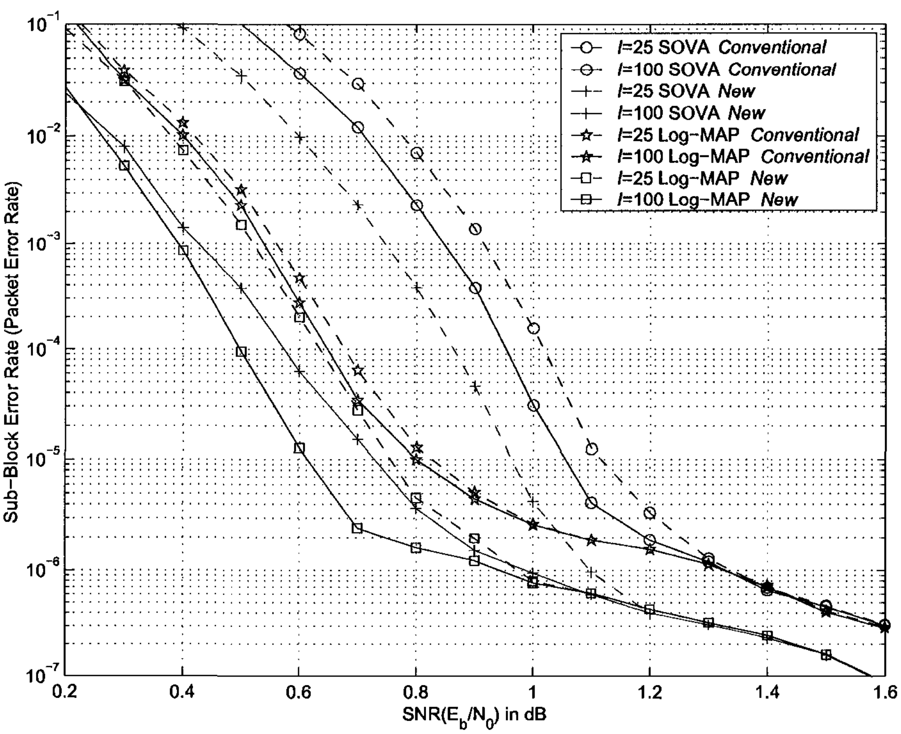


Figure 4.9: Sub-block error rate in AWGN channel, SOVA and Log-MAP, 9 preliminary iterations.

4.9 Chapter summary

In this chapter, a sub-block recovery scheme is proposed for the decoding of turbo codes with a sub-block structure. The analysis presented in this chapter shows that the proposed scheme helps to correct the (22:22) error pattern in turbo decoders. Since this kind of error pattern is inherent in turbo codes with highly structured interleavers, which are popular in practical systems, the proposed scheme improves the decoding performance of an iterative decoder for such a system. Simulation results show that the proposed scheme improves the sub-block error rate performance in the turbo cliff region and leads to a lowered error floor. It is also shown in this chapter that if the decoder uses an ML decoding algorithm, sub-block recovery does not result in improved performance.

Chapter 5

Turbo Coded Type-II Hybrid ARQ with Incremental Redundancy Using Sub-Block Recovery [1]

Depending on whether the receiver discards or retains the previously received noisy symbols after failing to correctly decode them, a hybrid ARQ protocol can be categorized as either type-I or type-II, respectively [2]. Type-II hybrid ARQ protocols offer significantly better throughput performance in the low SNR region compared to type-I protocols since they make use of the previously received noisy symbols. In a type-II hybrid ARQ protocol with incremental redundancy, additional parity bits are transmitted upon a retransmission request. At the receiver, the previously received and newly received symbols are combined to construct a sequence corresponding to a codeword of a lower rate.

A key component of type-II hybrid ARQ protocols with incremental redundancy is a rate compatible code. In [5], Hagenauer introduced rate-compatible punctured convolutional (RCPC) codes, which are constructed from a low rate convolutional code by successively puncturing the code to obtain a family of rate compatible codes with increasing rate. These codes can be decoded by the same decoder as the original code, which facilitates the application of hybrid ARQ with incremental redundancy. In [6], Rowitch and Milstein applied this concept to turbo codes and introduced rate-compatible punctured turbo (RCPT) codes.

In the previous chapter, the sub-block recovery scheme was introduced for a system that inherently uses the sub-block structure and uses a turbo code as the error correction code to improve the decoding performance.

Hybrid ARQ schemes based on the sub-block structure have been discussed in [47,54].

[1] A version of this chapter was presented in part at the *IEEE Globecom 2006*, San Francisco, USA, November/December 2006.

In these previous works, a type-I hybrid ARQ protocol was implicitly assumed. The ARQ protocols are separated from the FEC in the sense that ARQ operates on a higher layer of a packet (sub-block) basis that is transparent to the lower layer where FEC processes operate on a data-block basis. Although the sub-block structure and the related decoding algorithms are designed taking into consideration both the FEC code and ARQ procedures, from an operational point of view these two processes do not interact with each other.

As opposed to the previous work on type-I hybrid ARQ protocols for turbo coded systems that use the sub-block structure, in this chapter a type-II hybrid ARQ protocol with incremental redundancy using RCPT codes with the sub-block recovery is proposed. In Section 5.1 and 5.2, the system model and the proposed type-II hybrid ARQ scheme are introduced. In Section 5.3, the mechanisms by which sub-block recovery improves throughput performance are explained. Furthermore, in Section 5.4, a dynamic selection approach is proposed along with a scheme that makes use of both the concept of frequent termination introduced by Mielczarek and Svensson in [55] and the built-in CRC introduced by Zhai and Fair in [56]. In Section 5.5, a simple RCPT code that provides rate flexibility is introduced, and in Section 5.6, it is demonstrated through simulations that, for a turbo coded system using the sub-block structure, the proposed type-II hybrid ARQ scheme will achieve better throughput performance compared to either type-I hybrid ARQ schemes or type-II hybrid ARQ schemes with incremental redundancy but without the sub-block recovery scheme.

5.1 System model

In this part of the research work, both the AWGN channel and the Rayleigh fading channel are considered. In both cases, the additive noise has variance equal to $N_0/2$. Besides the channel and the turbo code, the system model is the same as the one that was introduced in Section 4.1.

The system employs the type-II hybrid ARQ with incremental redundancy using an RCPT code. A rate 1/3 codeword is generated for each data block by a parallel turbo encoder. During the first transmission, the systematic data bits and a small portion of the parity bits are transmitted. If all the sub-blocks cannot be successfully decoded, the receiver will ask for a retransmission. Bits for each retransmission are selected according to a predetermined rule that specifies the RCPT code. After a retransmission, the decoder uses the newly received symbols as well as all those received in previous transmissions and attempts to decode the lower rate code. If the decoder fails to correctly decode the entire data block after all the parity bits has been transmitted, the transmission and retransmission process is repeated a predetermined number of times while using maximum ratio combining to form the decoder input.

5.2 Sub-block recovery scheme

In this chapter, the sub-block recovery scheme is applied to the type-II hybrid ARQ system. Sub-block recovery is invoked repeatedly whenever new sub-blocks are correctly decoded after every transmission. As shown in the previous chapter, when sub-block recovery is used with a large number of sub-blocks in a data block, improvement in BER performance is observed in the form of a left-shifted turbo cliff. This directly translates to throughput improvement when applied to a type-II hybrid ARQ system as discussed in the next section.

5.3 The mechanism whereby sub-block recovery improves throughput performance

In order to efficiently incorporate sub-block recovery into the type-II hybrid ARQ scheme, it is important to investigate how sub-block recovery affects the throughput performance of the hybrid ARQ scheme.

For ease of presentation, in the following, the retransmissions that occur before the first sub-block is correctly decoded are called the *first stage* of retransmissions, and retransmissions following the detection of at least one correct sub-block are called the *second stage* of retransmissions. In the first stage of retransmissions, all the sub-blocks contain errors and sub-block recovery is not triggered. The number of retransmissions in the first stage should be unchanged whether sub-block recovery is used or not. During the second stage of retransmissions, since there already exist correctly decoded sub-blocks, sub-block recovery will be triggered until all the remaining sub-blocks are correctly decoded. With sub-block recovery, the number of retransmissions in the second stage will be fewer than or equal to that in the scheme without sub-block recovery. The reduced average number of retransmissions in the second stage is the source of the throughput improvement introduced by the sub-block recovery scheme.

5.4 Method to improve the performance of type-II hybrid ARQ with sub-block recovery

As indicated in the previous section, attempts to improve the throughput performance using sub-block recovery should aim at reducing the number of retransmissions in either the first stage or the second stage of retransmissions.

Reducing the number of retransmissions in the second stage

One approach to improve throughput performance is to dynamically select the bits for retransmission in accordance with the decoding status of the sub-blocks. In the following, this approach is referred to as "dynamic selection" and it is based on the fact that if there are correctly decoded sub-blocks and further retransmissions are required, bits corresponding to the correctly decoded sub-blocks do not need to be retransmitted. To keep the size of the retransmission block unchanged, more bits can be devoted to erroneous sub-blocks. In the second stage of retransmissions, when more than one retransmission is required, dynamic selection may result in fewer retransmissions, leading to throughput improvement.

However, the benefit of this method might be limited due to the following two issues. Firstly, in the SNR range where no more than one retransmission is usually necessary in the second stage of retransmissions, the improvement introduced by dynamic selection will be minor since for these dominant cases the number of retransmissions cannot be reduced. Secondly, to incorporate this dynamic selection in a practical system, the receiver needs to reliably transmit the decoding status of each sub-block back to the transmitter, which will unavoidably increase the signaling load. This problem might be more prominent when one data block contains a large number of sub-blocks. One method to reduce the load on signaling is to use one bit to indicate whether multiple sub-blocks have been correctly decoded or not, instead of using a bit to indicate the decoding status of each sub-block. However, as the decoding status feedback becomes less precise, the retransmitted block may also contain bits from previously correctly decoded sub-blocks, and as a result, the improvement introduced by the dynamic selection will decrease.

Reducing the number of retransmissions in both stages

Another approach that reduces the number of retransmissions in both the first stage and the second stage is motivated by the following two results.

In [55], Mielczarek and Svensson introduced the concept of frequently terminated turbo codes (FT-TC), where, as illustrated in Fig. 5.1, a certain number of bits are inserted

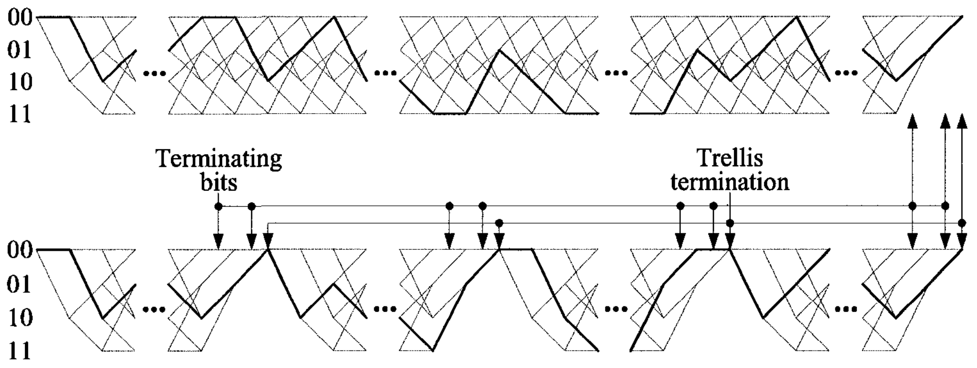


Figure 5.1: Trellises for G $= (1, \frac{7}{5})_8$ RSC codes without frequent termination (upper trellis) and with frequent termination (lower trellis). The solid line shows an example of possible paths through the trellis. Terminating bits are inserted to force periodic termination to the all-zero state.

into the input information sequence to the turbo encoder to terminate the trellis of the first constituent encoder more than once within one data-block encoding. One of the many advantages of this frequent terminating approach is improved decoding performance due to the awareness of the positions of the trellis terminations in the iterative decoding process. The price paid for this improved decoding performance is the rate loss due to the introduction of additional terminating bits. A related approach called slice turbo codes, is designed to enable parallel decoding and therefore is attractive for high throughput applications [57].

The feedback branch in a RSC code implements division over the ring of polynomials in GF(2). In [56], Zhai and Fair pointed out that when the trellis is terminated in the all-zero state, this circuit actually implements a CRC check, with the feedback polynomial of the RSC code as the generator polynomial and the terminating bits as the parity check bits. This is called the built-in CRC of the RSC encoder. In [56], it is proposed that the built-in CRC of the RSC constituent code be cascaded with the CRC commonly used in a data transmission system to construct an overall CRC with length equal to the sum of the length of both CRCs in order to improve the error detection performance.

Based on the above two techniques, a scheme is proposed to improve the throughput performance of a turbo coded hybrid ARQ system using the sub-block structure that benefits from both frequent termination and the built-in CRC. A CRC code whose generator polynomial contains as a factor the feedback polynomial of the RSC constituent encoder of the turbo code is selected. Since each sub-block corresponds to a packet protected by a CRC, by choosing such a CRC code for error protection, the trellis of the first RSC constituent encoder will return to the all-zero state after encoding each sub-block, where no

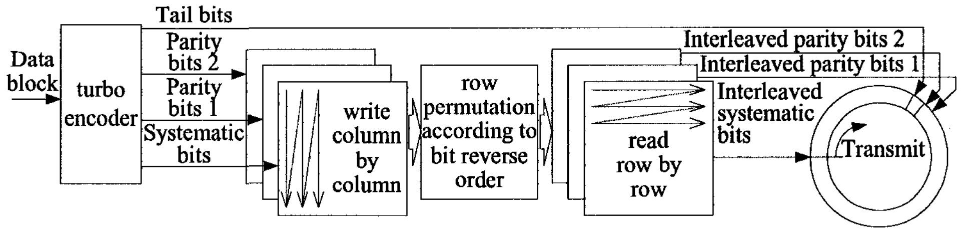

Tail bits
Data block
turbo encoder
Parity bits 2
Parity bits 1
Systematic bits
write column by column
row permutation according to bit reverse order
read row by row
Interleaved parity bits 2
Interleaved parity bits 1
Interleaved systematic bits
Transmit

Figure 5.2: Construction of the rate compatible turbo code.

extra terminating bits are required. In this way, an FT-TC is actually constructed and the extra terminations will result in fewer retransmissions in both stages.

Note that utilizing the built-in CRC and frequent termination without employing sub-block recovery will also result in improvement. However, applying sub-block recovery will result in additional throughput gains.

5.5 The RCPT code

To construct a flexible rate RCPT code, instead of optimizing puncturing patterns for a group of predetermined rates as in [6], a simple RCPT code is designed from a rate 1/3 parallel concatenated turbo code. The primary purpose for this RCPT code design is that after any number of retransmissions of any size, the transmitted bits are uniformly distributed within the entire codeword, therefore the received symbols could make balanced contributions to the decoding of the entire data block for both constituent codes.

The proposed RCPT encoding process is illustrated in Fig. 5.2. The systematic bits and the parity bits from both constituent codes are separately permuted, and then combined together with the tail bits into a queue, from which the bits for retransmissions are selected in a loop. Permuted systematic bits and the tail bits are inserted into the front of the queue, followed by permuted parity bits from both constituent codes that are interlaced to ensure a balanced number of symbols for both constituent decoders. The systematic bits and the parity bits from both constituent encoders are uniformly interleaved by writing column-wise into a matrix of 64 rows and reading out row-wise after a row permutation according to the bit reverse order [2]. For example, the 5-th row will be moved to the 10-th row because the binary representation of 10, $(1010)_2$, is the bit reversed version of that of 5, $(0101)_2$.

[2] The dimension parameter 64 of the matrix is chosen in an ad-hoc fashion.

5.6 Simulations

The throughput performance of the proposed type-II hybrid ARQ scheme and several reference hybrid ARQ schemes are compared on the AWGN channel and the perfectly interleaved uncorrelated flat Rayleigh fading channel. In the simulations, the turbo code that was described in Section 3.3 is considered. The Log-MAP algorithm for iterative decoding with a maximum of I=10 iterations is used. The 16-bit CRC code with generator polynomial $(200433)_8 = (27161)_8 \cdot (13)_8$ for error detection [58] is used (note that the second factor defines the feedback polynomial of the RSC constituent encoder).

The data block length is arbitrarily set to N=1760 for all simulations to exclude the effect of different interleaver lengths on performance. Each data block contains $E = 16$ sub-blocks each consisting of a data packet of 94 information bits and 16 parity bits for the CRC, which results in a rate loss of 0.68 dB. The rate loss is inherent in the sub-block structure and would be lower for longer data blocks and sub-blocks as would be the case in practical systems.

In the simulated type-II hybrid ARQ scheme, the first transmission contains all the systematic bits, the tail bits and 220 parity bits selected in the manner introduced in the previous section corresponding to an initial code rate of 0.89. Note that in a practical system, the initial rate should be selected in accordance with the channel condition to achieve the desired delay performance. Each retransmission contains 220 bits and a maximum of 63 retransmissions are allowed, corresponding to the lowest code rate of approximately 1/9. In a practical system, when the size of the retransmissions larger and the maximum number of retransmissions is smaller, the throughput performance curve will exhibit more coarse granularity compared to the curves shown here.

In the type-I hybrid ARQ scheme used for comparison, the full rate (approximately 1/3) turbo code is always applied. The sub-blocks that are detected in error in a data block are included in the data block to be transmitted in the next time slot. The turbo codeword for the data block containing the retransmitted sub-blocks is decoded independently of any information obtained from previous transmissions.

SNR is defined as E_s/N_0, where E_s is the energy for each transmitted symbol and throughput is defined as the reciprocal of the average number of encoded bits transmitted for the successful reception of one information bit, where CRC bits are not counted as information. The performance statistics are based on 10000 data block transmissions on each test point with a 0.1 dB separation between adjacent test points to capture the detailed variation as SNR changes.

Fig. 5.3 compares the throughput of the proposed type-II hybrid ARQ scheme with the BPSK capacity and the type-I hybrid ARQ scheme using the turbo code with the sub-block structure. "SBR" stands for sub-block recovery; "DS" stands for dynamic selection. For the dynamic selection case, it is assumed that the transmitter has complete knowledge of the decoding status for each sub-block. "FT" stands for frequent termination. The "non-SBR" scheme refers to the scheme proposed in [47], where, after each iteration, the hard decisions for each correctly decoded sub-block are recorded but are not fed back to the decoder. This algorithm improves the decoding performance by taking advantage of the error fluctuation phenomenon observed in iterative decoding where, occasionally, the number and locations of the errors vary from iteration to iteration.

From Fig. 5.3, it is clear that the type-II hybrid ARQ scheme outperforms the type-I hybrid ARQ schemes due to its capability to adapt the code rate according to the channel condition. By utilizing sub-block recovery, dynamic selection and frequent termination, a steady improvement in throughput over the entire SNR range is achieved. Specifically, the throughput is improved by about 0.3 dB in the low throughput region and 0.6 dB in the high throughput region.

Fig. 5.4 shows a comparison of throughput loss with reference to the capacity of different type-II hybrid ARQ schemes discussed in this chapter. As shown in the figure, the sub-block recovery scheme with frequent termination provides a visible improvement compared to schemes that do not use these techniques. Dynamic selection introduces improvement only in the low SNR region, where usually more than one retransmission is required in the second stage.

Fig. 5.5 shows a comparison of the average number of retransmissions in the second stage. It shows that the sub-block recovery scheme helps reduce retransmissions in the second stage, and fewer retransmissions are required in the low SNR region when using dynamic selection. At $E_s/N_0 = -4$ dB, where the throughput is about 0.3, sub-block recovery with dynamic selection saves on average one retransmission.

As shown in Figs. 5.6 - 5.8, similar throughput improvement is observed in the perfectly interleaved uncorrelated flat Rayleigh fading channel, where throughput improvement in the high SNR region reaches over 1.3 dB.

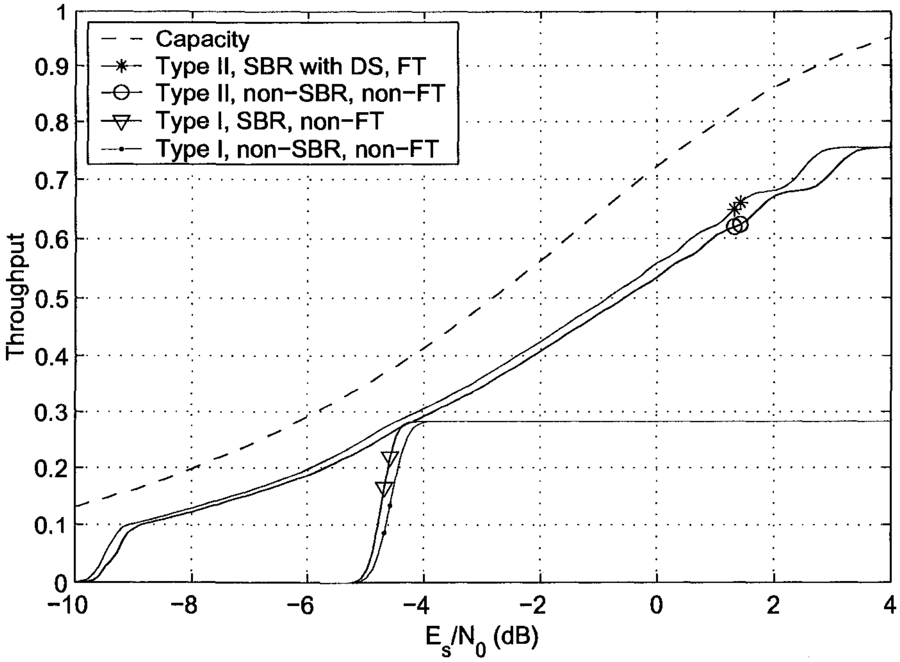


Figure 5.3: Throughput on AWGN channel for N=1760, 16 sub-blocks, 10 iterations. FT - frequent termination, DS - dynamic selection.

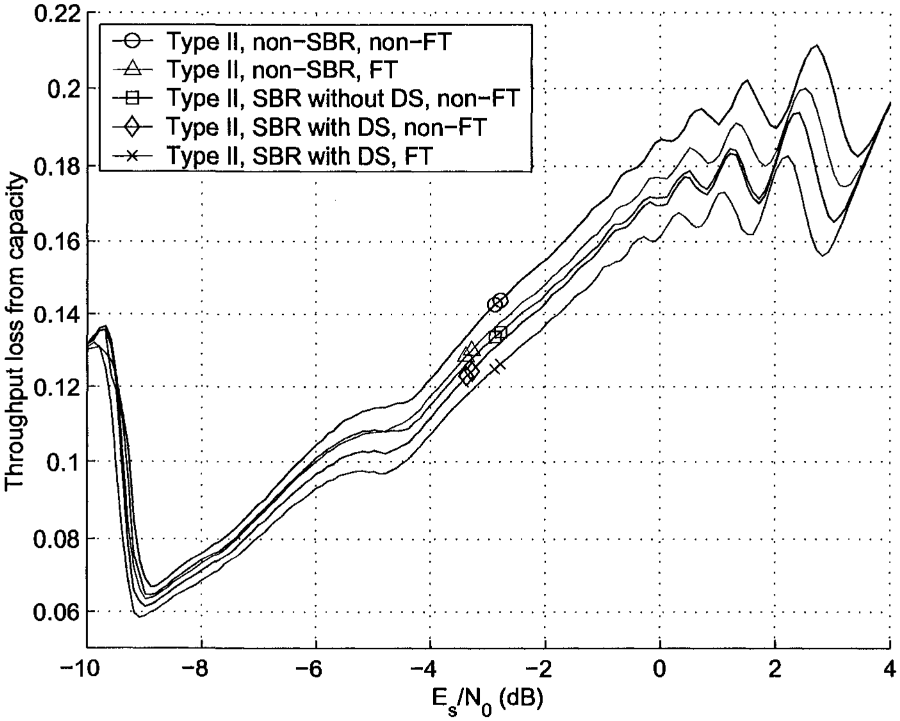


Figure 5.4: Throughput loss on AWGN channel for N=1760, 16 sub-blocks, 10 iterations.

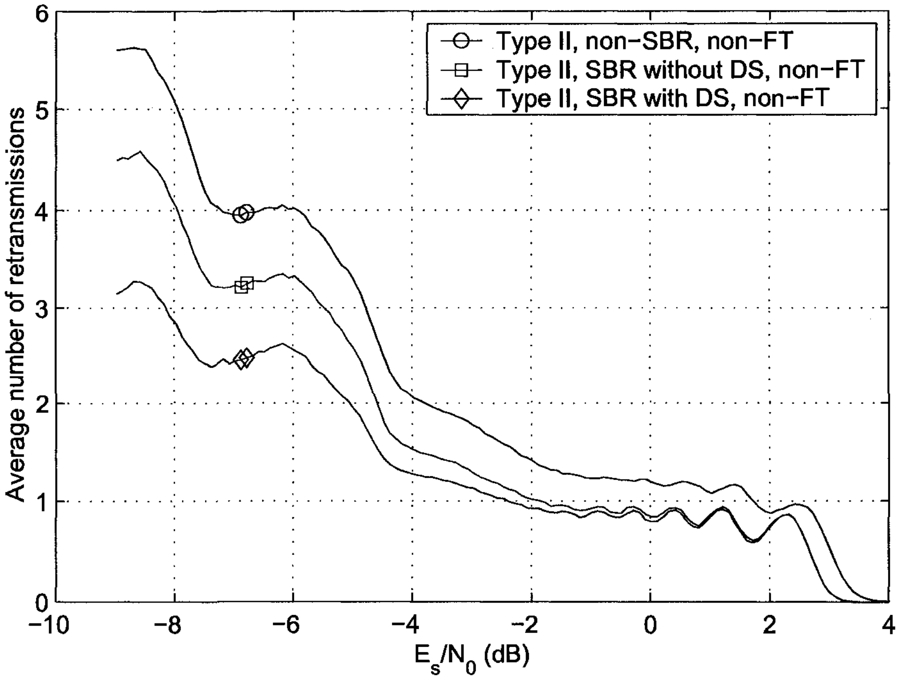


Figure 5.5: Average number of retransmissions in the second stage on AWGN channel for N=1760, 16 sub-blocks 10 iteration.

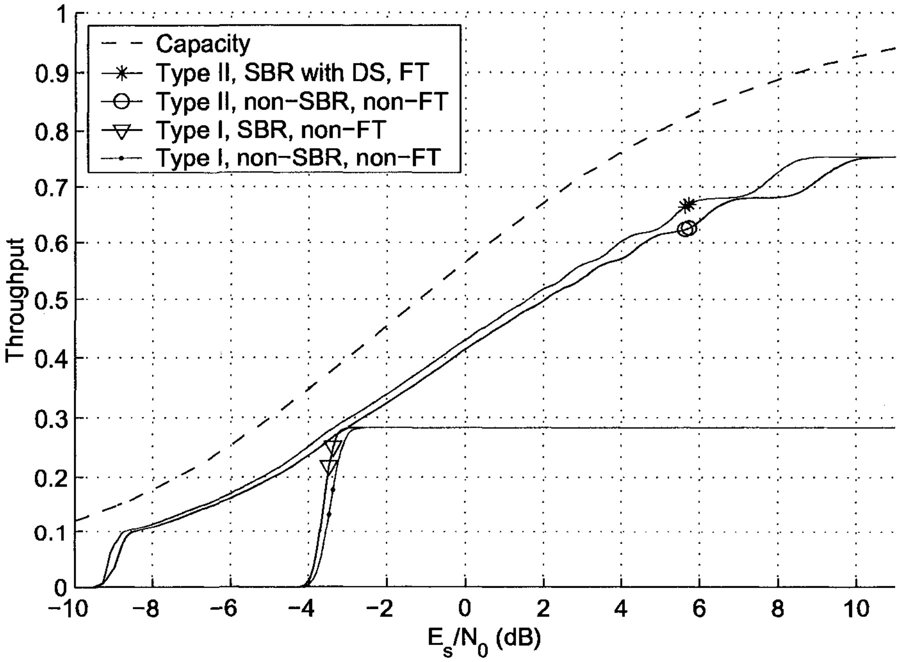


Figure 5.6: Throughput on flat Rayleigh fading channel for N=1760, 16 sub-blocks, 10 iterations.

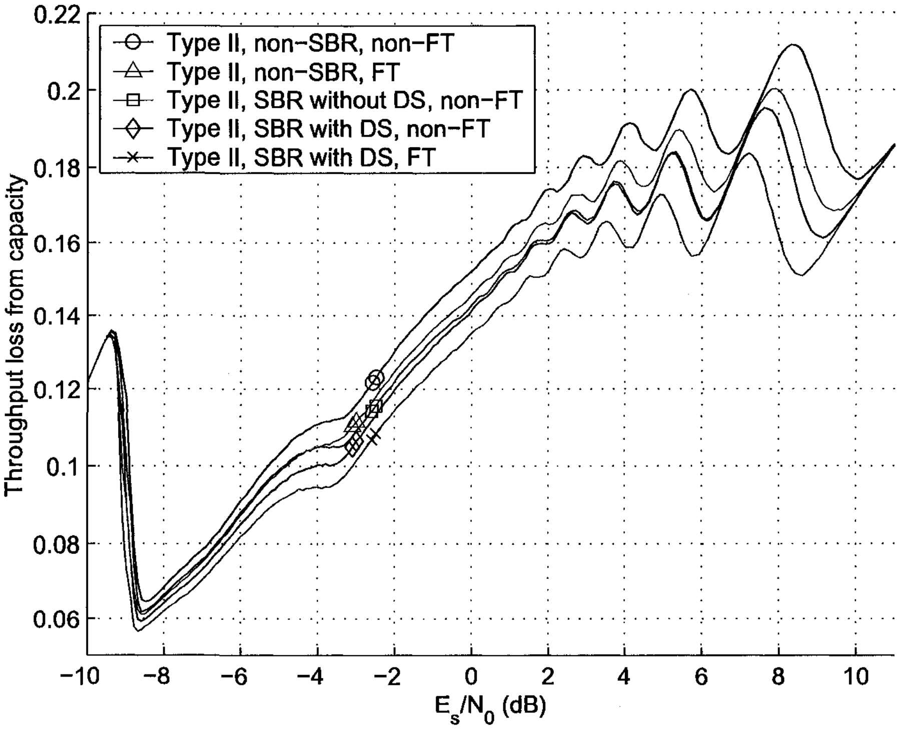


Figure 5.7: Throughput loss on flat Rayleigh fading channel for N=1760, 16 sub-blocks, 10 iterations.

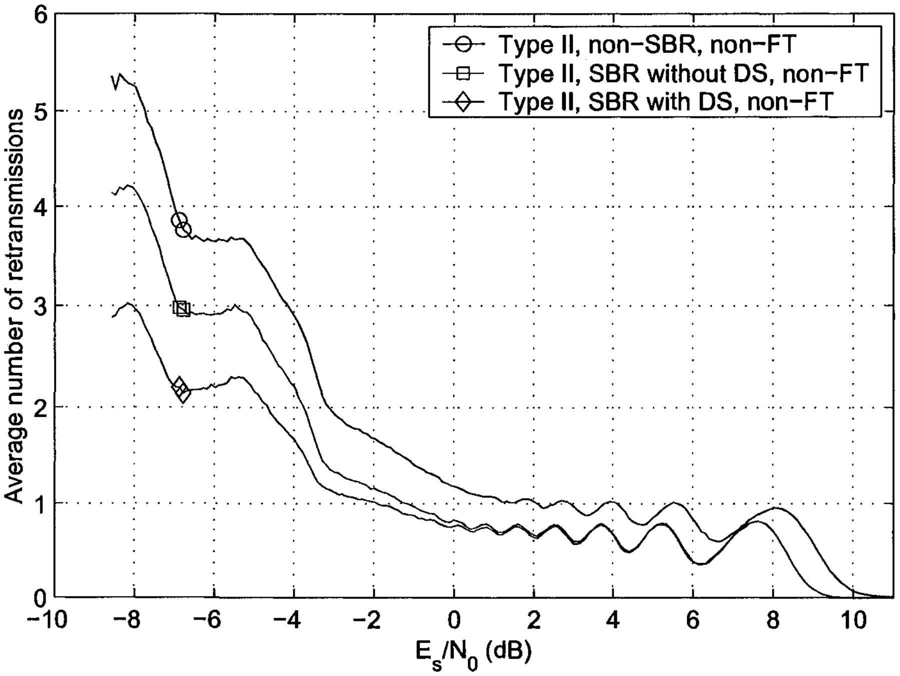


Figure 5.8: Average number of retransmissions in the second stage on flat Rayleigh fading channel for N=1760, 16 sub-blocks 10 iteration.

5.7 Chapter summary

In this chapter, a type-II hybrid ARQ scheme with incremental redundancy and soft packet combining is proposed for turbo coded systems that use the sub-block structure. Sub-block recovery is applied to rate compatible turbo codes, and as a result, fewer retransmissions are required after the first sub-block is correctly decoded which leads to throughput improvement. Given knowledge of the decoding status, throughput is further improved by dynamically selecting only the bits corresponding to the erroneous sub-blocks for retransmission. For a turbo coded hybrid ARQ scheme with sub-block structure, by selecting the CRC code to have a generator polynomial that contains as a factor the feedback polynomial of the RSC constituent code, a frequently terminated turbo code can be constructed without rate loss. This can result in further improvement in terms of throughput.

Simulations show that the proposed type-II hybrid ARQ scheme outperforms both the type-I hybrid ARQ with sub-block recovery and the traditional type-II hybrid ARQ scheme without sub-block recovery on both the AWGN and flat Rayleigh fading channels.

Chapter 6

Hybrid ARQ for Layered Space Time MIMO Systems with Channel State Information Only at the Receiver [1]

It has been shown through theory and practice that by deploying multiple antennas at both the transmitter and receiver, an spatial multiplexing MIMO system can achieve high spectral efficiency in a rich scattering environment, and therefore enables high data rate packet transmission [10, 59]. In this chapter, we investigate hybrid ARQ schemes for an spatial multiplexing MIMO system with CSIR.

In such a system, equal power is allocated to each layer, and at the receiver the raw BER varies for different layers. Recognizing this phenomenon, Zheng et al. proposed the use of multiple ARQ processes instead of a single ARQ process to achieve better throughput performance [60]. In the system setup described in [60], the block size for retransmission in the multiple hybrid ARQ is smaller than that in the single hybrid ARQ. In the multiple hybrid ARQ scheme, a retransmission for a packet can be carried out in the unit of $\frac{1}{M_T}$ (where M_T is the number of transmit antennas) of the data in a packet, while for the single hybrid ARQ scheme, a packet has to be repeated in full if retransmission is required. Although the simulation results show that the multiple hybrid ARQ as described in [60] outperforms the single hybrid ARQ, a comparison of these two schemes based on the same granularity for hybrid ARQ retransmission is necessary in order to show the superiority of the multiple hybrid ARQ scheme. In [17], Onggosanusi et al. developed a joint detection algorithm for ARQ transmission with repetition and proved that it is superior to the separate

[1] A version of this chapter was presented in part at the *IEEE WCNC 2007*, Hong Kong, China, March 2007.

detection algorithm. The latter is equivalent to the combining method used in [60]. The joint detection algorithm developed in [17] is designed for a single ARQ process, and requires that the symbols transmitted on all antennas be retransmitted simultaneously to facilitate joint detection.

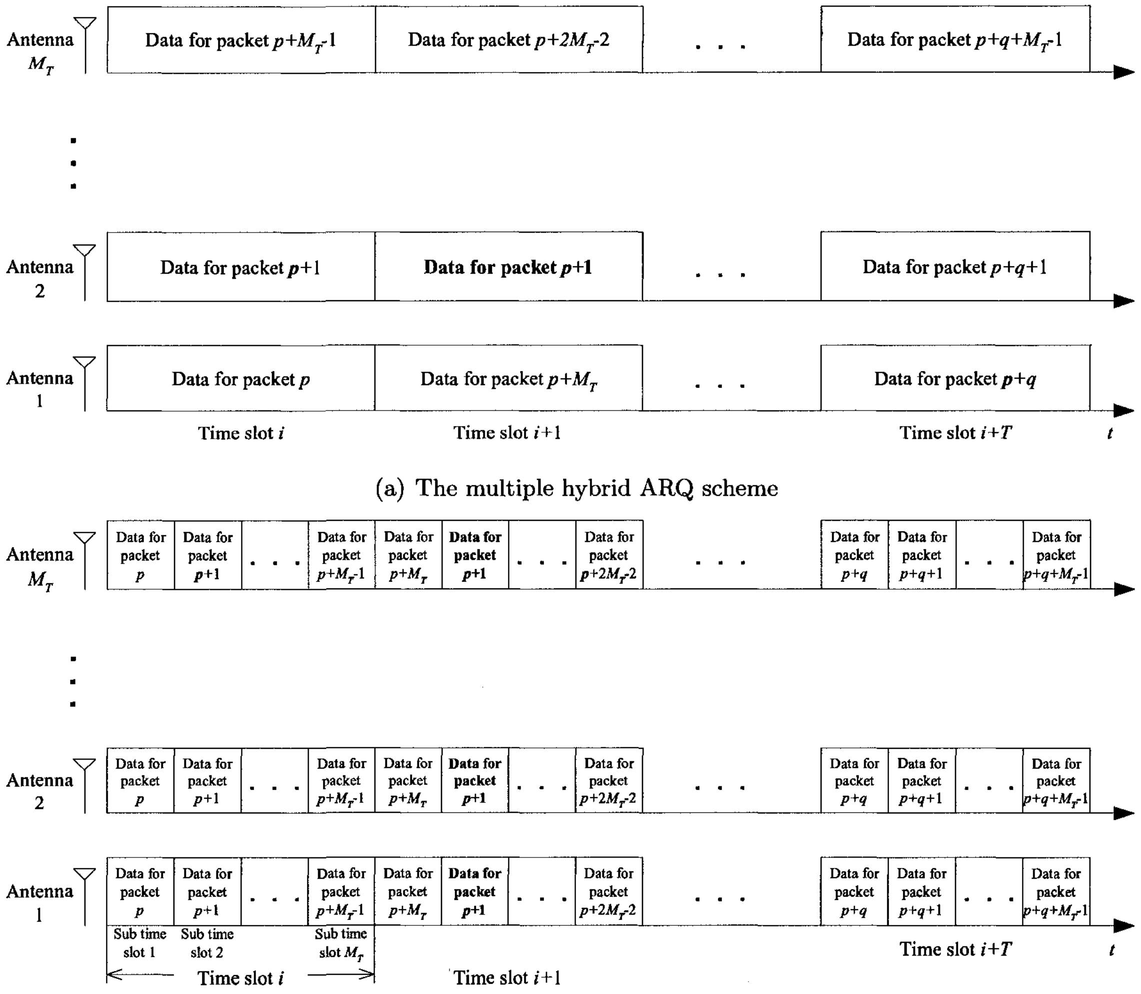
Motivated by the above work, in this chapter we first introduce a system model for the spatial multiplexing MIMO transmission that incorporates the multiple ARQ processes, and then develop a joint detection algorithm for the multiple hybrid ARQ processes. An improved combining algorithm is also presented for the separate detection algorithm. The performance of the single hybrid ARQ and the multiple hybrid ARQ processes based on the same block size for retransmission with joint detection and separate detection is compared via simulation.

In Section 6.1, a system model for LST MIMO transmission that incorporates the multiple ARQ process is introduced. In Section 6.2, the joint detection schemes for multiple ARQ transmissions is developed. In Section 6.3, we compare the performance of different hybrid ARQ schemes with different combining algorithms and illustrate the advantage of using joint detection together with multiple ARQ processes for the system of interest. The simulation results are presented in Section 6.4, followed by a brief summary in Section 6.5.

In this chapter, superscripts T and H denote the transpose and the conjugate transpose respectively; $(\cdot)_k$ and $[\cdot]_k$ denote the k-th row and the k-th column of the matrix "$\cdot$" respectively; $(\cdot)_{k,h}$ denotes the element at the k-th row and the h-th column of the matrix "$\cdot$"; and $\mathbf{I}_n$ denotes the $n \times n$ identity matrix.

6.1 System Model

In this chapter, a packet-based point-to-point layered MIMO system with M_T transmit antennas and M_R receive antennas is considered, where $M_T \leq M_R$. Each packet has equal length, and is first encoded by an error detection code and then encoded by an error correction code into a single codeword. Physical layer transmissions are organized in time slots, and 2^b-ary quadrature amplitude modulation (QAM) is used. It is assumed that each time slot equals the length of N symbol intervals. The parameters are chosen so that N is divisible by M_T, and Nb equals the length of a codeword. Therefore in each time slot, from M_T antennas, M_T packets are transmitted. It is assumed that the channel is complex Gaussian, and is quasi-static in that the channel realization does not change during each time slot, but changes independently from one time slot to the next. The receiver requests retransmission of a packet if the parity check indicates that it is erroneous, upon which



(a) The multiple hybrid ARQ scheme

(b) The single hybrid ARQ scheme

Figure 6.1: Hybrid ARQ schemes under consideration, assuming that retransmission of packet $p+1$ is requested.

the entire codeword is retransmitted. A failure is declared if a packet cannot be correctly detected after a maximum of T_{Max} transmissions.

In this chapter, two types of hybrid ARQ schemes are considered. In a multiple hybrid ARQ scheme, the transmission/retransmission of data on each antenna is managed by individual ARQ loops. Each packet is preassigned to an antenna and is transmitted only from that antenna. The transmission/retransmission on each layer is independent from that on other layers. In a single hybrid ARQ scheme, the transmission/retransmission of data on all layers is managed by a single ARQ loop, therefore, if retransmission is requested, symbols transmitted on all antennas are retransmitted simultaneously. To avoid unnecessary retransmission of correctly received packets, one time slot is divided into M_T sub time slots. M_T packets are transmitted in each time slot in a time division multiplexing (TDM) fashion.

The transmission/retransmission in each sub time slot is independent from that in other sub time slots. Each packet is pre-assigned to a sub time slot and the codeword is transmitted only in that sub time slot.

It is assumed that the two hybrid ARQ schemes described above provide a premise for a fair comparison of the multiple and single ARQ schemes since they require the same amount of feedback per slot, and since no packet will be retransmitted when it is correctly decoded at the receiver.

For simplicity, a stop and wait ARQ protocol is considered and it is assumed that the feedback channel is free of errors and that the feedback is available immediately after transmission so that the ARQ transmission can be continuous. In a practical system, multiple stop and wait ARQ processes can be used to accommodate delays encountered in the transmission and processing of the ACK/NACK, thereby improving the aggregate throughput and spectral efficiency.

Fig. 6.1 illustrates how packet transmissions are organized in both the multiple hybrid ARQ scheme and the single hybrid ARQ scheme. It is assumed that no retransmission occurs during time slot i and packets of index p through $p + M_T - 1$ are transmitted. For the multiple hybrid ARQ scheme, each packet occupies a time slot and data for one packet is transmitted from one antenna. For the single hybrid ARQ scheme, each packet occupies a sub time slot and data for one packet is transmitted from all the antennas.

It is assumed that only packet $p+1$ needs to be retransmitted in time slot $i+1$. For the multiple hybrid ARQ scheme, data for packet $p + 1$, shown in bold face in Fig. 6.1(a), is retransmitted on antenna 2, which is the same antenna from which the original data for the packet was transmitted. For the single hybrid ARQ scheme, data for packet $p + 1$, shown in bold face in Fig. 6.1(b), is retransmitted during sub time slot 2, which is the same sub time slot in which the original data for the packet was transmitted.

Single transmission (no retransmission)

Let $\mathbf{x}^{(i,h)}$, $\mathbf{r}^{(i,h)}$ and $\mathbf{w}^{(i,h)}$ denote the $M_T \times 1$ transmitted symbol vector, the $M_R \times 1$ received symbol vector and the $M_R \times 1$ independent white Gaussian noise vector, respectively, in the h-th symbol of the i-th time slot. Let $\mathbf{H}^{(i)}$ be the matrix of channel coefficients for the i-th time slot. To simplify notation, the second superscript h is dropped from the notation to give:

$$\mathbf{r}^{(i)} = \mathbf{H}^{(i)}\mathbf{x}^{(i)} + \mathbf{w}^{(i)}. \tag{6.1}$$

The received symbol vector is passed through a matched filter $\mathbf{H}^{(i)^H}$, yielding output $\mathbf{y}^{(i)}$:

$$\mathbf{y}^{(i)} = \mathbf{H}^{(i)^H}\mathbf{r}^{(i)}. \tag{6.2}$$

It is assumed that the symbols transmitted on different layers are i.i.d. with zero mean, therefore, their covariance matrix is

$$\mathbf{R}_{\mathbf{x}^{(i)}} = E\{\mathbf{x}^{(i)}\mathbf{x}^{(i)^H}\} = \sigma_x^2\mathbf{I}_{M_T} = E_s\mathbf{I}_{M_T}. \tag{6.3}$$

It is also assumed that the noise samples $\mathbf{w}^{(i)}$ are i.i.d., with zero mean, and the covariance matrix of the noise samples is

$$\mathbf{R}_{\mathbf{w}^{(i)}\mathbf{w}^{(j)}} = E\{\mathbf{w}^{(i)}\mathbf{w}^{(j)^H}\} = \begin{cases} \mathbf{R}_{\mathbf{w}^{(i)}} = \sigma_w^2\mathbf{I}_{M_R} & \text{for } i = j \\ \mathbf{0}_{(M_R\times M_R)} & \text{for } i \neq j \end{cases} \tag{6.4}$$

where $\mathbf{0}_{(M_R\times M_R)}$ is an $M_R \times M_R$ all zero matrix.

Single hybrid ARQ

Without loss of generality, the system model for transmission/retransmission in the first sub time slot of each time slot is considered. It is assumed that the packet transmitted in (the first sub time slot of) time slot i was first transmitted in time slot $i - T + 1$, where $1 \leq T \leq T_{\text{Max}}$. At time slot i, let $S = \{i-T+1, i-T+2, \cdots, i\}$ denote the set of indices of the time slots in which the same packet as that transmitted in time slot i have been transmitted. Noticing that $\mathbf{x}^{(i)} = \mathbf{x}^{(s)}$ for every $s \in S$, and with $\mathbf{r} = \left[\mathbf{r}^{(i-T+1)}, \cdots, \mathbf{r}^{(i)}\right]^T$, it follows from (6.1) and (6.2)that :

$$\begin{aligned}
\mathbf{r} &= \begin{bmatrix} \mathbf{H}^{(i-T+1)}\mathbf{x}^{(i-T+1)} \\ \vdots \\ \mathbf{H}^{(i)}\mathbf{x}^{(i)} \end{bmatrix} + \begin{bmatrix} \mathbf{w}^{(i-T+1)} \\ \vdots \\ \mathbf{w}^{(i)} \end{bmatrix} && (6.5)\\
&= \underbrace{\begin{bmatrix} \mathbf{H}^{(i-T+1)} \\ \vdots \\ \mathbf{H}^{(i)} \end{bmatrix}}_{\mathbb{H}} \mathbf{x}^{(i)} + \underbrace{\begin{bmatrix} \mathbf{w}^{(i-T+1)} \\ \vdots \\ \mathbf{w}^{(i)} \end{bmatrix}}_{\mathbf{w}} && (6.6)\\
&= \sum_{n=1}^{M_T} [\mathbb{H}]_n x_n^{(i)} + \mathbf{w} && (6.7)
\end{aligned}$$

where $[\mathbb{H}]_n = \left[[\mathbf{H}]_n^{(i-T+1)^T}, \cdots, [\mathbf{H}]_n^{(i)^T}\right]^T$ is the channel vector for the symbols transmitted on the n-th antenna. After the received symbol vector is passed through the matched filter $\mathbb{H}^H$,

$$\mathbf{y} = \mathbb{H}^H\mathbf{r} = \mathbf{C}\mathbf{x}^{(i)} + \sum_{s\in S}\mathbf{H}^{(s)^H}\mathbf{w}^{(s)} \tag{6.8}$$

where $\mathbf{C}$ is defined as $\mathbf{C} = \sum_{s \in S} \mathbf{H}^{(s)^H} \mathbf{H}^{(s)}$. The system model expressed in (6.8) is the same as the joint detection ("pre-combining") approach described in [17].

Multiple hybrid ARQ

For a multiple hybrid ARQ scheme, at time slot i, let $S = \{i - T + 1, i - T + 2, \cdots, i\}$ denote the set of indices of the time slots in which on at least one of the layers, the same packet as that transmitted in time slot i was transmitted, where $T \leq T_{\mathrm{Max}}$. With multiple hybrid ARQ, (6.5) still applies, but (6.6) is no longer valid because $\mathbf{x}^{(i)} = \mathbf{x}^{(s)}$ does not hold for every $s \in S$. In order to construct a compact model that contains all the information relevant to the detection of the symbols transmitted in time slot i, an $M_R \times M_T$ modified channel matrix $\tilde{\mathbf{H}}^{(s)}$ and an $M_R \times 1$ modified received symbol vector $\tilde{\mathbf{r}}^{(s)}$ for every $s \in S$ are introduced. The definition of the n-th column of the modified channel matrix depends on whether the same packet as that transmitted in time slot i was transmitted in time slot s on layer n:

$$[\tilde{\mathbf{H}}^{(s)}]_n = \begin{cases} [\mathbf{H}^{(s)}]_n & \begin{array}{l}\text{if the same packet was transmitted in} \\ \text{time slot } i \text{ and } s \text{ on layer } n\end{array} \\ 0_{(M_R \times 1)} & \begin{array}{l}\text{if different packet was transmitted in} \\ \text{time slot } i \text{ and } s \text{ on layer } n\end{array} \end{cases} \tag{6.9}$$

and the modified received symbol vector can be expressed as

$$\tilde{\mathbf{r}}^{(s)} = \mathbf{r}^{(s)} - \sum_{n \in N} [\mathbf{H}^{(s)}]_n \mathbf{x}_n^{(s)} \tag{6.10}$$

where N is the set of the indices of the layers on which the packet transmitted in time slot s has been correctly decoded. In order to facilitate detection in the next time slot, the modified channel matrix and the modified received symbol vector, as well as the set S, should be updated according to the following procedure. At time slot i:

- Record $\tilde{\mathbf{H}}^{(i)} = \mathbf{H}^{(i)}$, and $\tilde{\mathbf{r}}^{(i)} = \mathbf{r}^{(i)}$;

- For every $s \in S$, update $\tilde{\mathbf{H}}^{(s)}$ by substituting the columns corresponding to the layers that are correctly detected in the current time slot by the $(M_R \times 1)$ all-zeros column vector;

- For every $s \in S$, update $\tilde{\mathbf{r}}^{(s)}$ by cancelling the interference caused by the symbols transmitted on the layers that are correctly detected in the current time slot;

- If the element $i - T_{\mathrm{Max}} + 2 \in S$, remove it from S;

- Remove any $s \in S$ for which $\tilde{\mathbf{H}}^{(s)}$ is an all-zero matrix, as this indicates that all the packets transmitted in time slot s have been correctly detected, therefore the symbols received in that time slot are irrelevant to detection in future time slots.

This update procedure essentially removes the components corresponding to the packets correctly detected in time slot i. The updated $\tilde{\mathbf{H}}^{(s)}$ and $\tilde{\mathbf{r}}^{(s)}$ for $s \in S$ involve only the previous (re)transmissions for the packets that will be retransmitted in time slot $i+1$.

Assuming that no packet has been declared failed in any time slot $s \in S$, all the interference not related to the current detection can be completely removed from the received symbol vectors. For n such that the packet transmitted on layer n in time slot i was not transmitted in time slot s, $s \in S$, $[\tilde{\mathbf{H}}^{(s)}]_n \mathbf{x}_n^{(s)} = 0_{(M_R \times 1)} = [\tilde{\mathbf{H}}^{(s)}]_n \mathbf{x}_n^{(i)}$, therefore

$$\tilde{\mathbf{r}}^{(s)} = \tilde{\mathbf{H}}^{(s)} \mathbf{x}^{(i)} + \mathbf{w}^{(s)}. \tag{6.11}$$

Let $\tilde{\mathbf{r}} = \left[\tilde{\mathbf{r}}^{(i-T+1)}, \cdots, \tilde{\mathbf{r}}^{(i)}\right]^T$. Similarly to (6.7),

$$\tilde{\mathbf{r}} = \sum_{n=1}^{M_T} [\tilde{\mathbb{H}}]_n x_n^{(i)} + \mathbf{w} \tag{6.12}$$

where $[\tilde{\mathbb{H}}]_n = \left[[\tilde{\mathbf{H}}]_n^{(i-T+1)^T}, \cdots, [\tilde{\mathbf{H}}]_n^{(i)^T}\right]^T$ is the channel vector for the symbol transmitted on the n-th antenna. Once the modified received symbol vector is passed through the matched filter $\tilde{\mathbb{H}}^H$, the output $\tilde{\mathbf{y}}$ can be expressed in a manner similar to (6.8).

6.2 Joint detection algorithms

In joint detection algorithms, symbols received in the first transmission and any subsequent retransmissions are detected jointly.

Linear detection algorithms

Let $\mathbf{G}$ denote the linear post processing matrix and let $\hat{\mathbf{x}}^{(i)}$ denote the decision statistics in time slot i. It then follows that

$$\hat{\mathbf{x}}^{(i)} = \mathbf{G}\mathbf{y} = \mathbf{G}\mathbf{C}\mathbf{x}^{(i)} + \mathbf{G} \sum_{s \in S} \mathbf{H}^{(s)^H} \mathbf{w}^{(s)} \tag{6.13}$$

Zero forcing (ZF) criterion

When the ZF criterion is used, as given by (5) in [17]:

$$\mathbf{G}^{\text{ZF-JD}} = \mathbf{C}^{-1} = \left(\sum_{s \in S} \mathbf{H}^{(s)^H} \mathbf{H}^{(s)}\right)^{-1}. \tag{6.14}$$

where "ZF-JD" stands for "zero forcing - joint detection". The decision statistics for the symbols transmitted on layer n are

$$\hat{x}_n^{(i)^{\text{ZF-JD}}} = (\mathbf{C}^{-1})_n \mathbf{y} = x_n^{(i)} + \left(\mathbf{C}^{-1}\right)_n \sum_{s \in S} \mathbf{H}^{(s)^H} \mathbf{w}^{(s)} \tag{6.15}$$

The SNR for the symbols transmitted on layer n, as given by (11) in [17], is

$$\text{SNR}_n = \frac{E_s}{\sigma_w^2} \left[\left(\mathbf{C}^{-1}\right)_{n,n} \right]^{-1}. \tag{6.16}$$

Minimum mean square error (MMSE) criterion

By definition, the mean square error (MSE) for the linear detector is

$$\text{MSE} = E\{\|\hat{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}\|^2\} = E\{\|\mathbf{G}\mathbf{y} - \mathbf{x}^{(i)}\|^2\}. \tag{6.17}$$

The linear post processing matrix that results in the minimum MSE, as given by (7) in [17], is

$$\mathbf{G}^{\text{MMSE-JD}} = \left(\mathbf{C} + \frac{\sigma_w^2}{E_s} \mathbf{I}_{M_T} \right)^{-1}, \tag{6.18}$$

where "MMSE-JD" stands for "minimum mean square error - joint detection". The decision statistics for the symbols transmitted on layer n are

$$\hat{x}_n^{(i)^{\text{MMSE-JD}}} = (\mathbf{G}^{\text{MMSE-JD}})_n \mathbf{y} = \left((\mathbf{C} + \frac{\sigma_w^2}{E_s} \mathbf{I}_{M_T})^{-1} \right)_n \mathbf{y}. \tag{6.19}$$

The minimum MSE for the symbols transmitted on layer n is

$$\text{MSE}_n = \sigma_w^2 (\mathbf{G}^{\text{MMSE-JD}})_{n,n}. \tag{6.20}$$

Nonlinear detection algorithms

In this section, the V-BLAST receiver architecture, which is essentially a decision feedback detection (DFD) algorithm with best-first ordering, is considered [59]. This nonlinear detection algorithm can be viewed as a series of M_T linear detection steps, one for each layer. The detection is performed symbol-wise.

For time slot i, the nonlinear detection algorithm for the multiple hybrid ARQ scheme can be described as follows.

1. Initialization

1.1) $m = 0$, $K = \emptyset$

1.2) $S = S \cup \{i\}$ (if $i = 0$, $S = \{i\}$)

1.3) $\tilde{\mathbf{H}}^{(i)} = \mathbf{H}^{(i)}$

```
1.4) $\mathbf{H}_m^{(s)} = \tilde{\mathbf{H}}^{(s)} \;\forall\; s \in S$
1.5) $\tilde{\mathbf{r}}^{(i)} = \mathbf{r}^{(i)}$
1.6) $\mathbf{r}_m^{(s)} = \tilde{\mathbf{r}}^{(s)} \;\forall\; s \in S$
 2. Iteration
2.1) If $m$ equals $M_T - 1$, stop
2.2) $\mathbf{C}_m = \sum_{s \in S} \mathbf{H}_m^{(s)^H} \mathbf{H}_m^{(s)} \;\forall\; s \in S$
2.3) $\mathbf{y}_m^{(s)} = \mathbf{H}_m^{(s)^H} \mathbf{r}_m^{(s)} \;\forall\; s \in S$
2.4) $k_m = \text{argmin}_k (\mathbf{C}_m^{-1})_{k,k}$ for the ZF criterion, or
     $k_m = \text{argmin}_k \left( (\mathbf{C}_m + \frac{\sigma_w^2}{E_s} \mathbf{I}_{M_T - |K|})^{-1} \right)_{k,k}$ for the MMSE criterion
2.5) $\hat{x}_{k_m} = (\mathbf{C}_m^{-1})_{k_m} \sum_{s \in S} \mathbf{y}_m^{(s)}$ for the ZF criterion, or
     $\hat{x}_{k_m} = \left( (\mathbf{C}_m + \frac{\sigma_w^2}{E_s} \mathbf{I}_{M_T - |K|})^{-1} \right)_{k_m} \sum_{s \in S} \mathbf{y}_m^{(s)}$ for the MMSE criterion
2.6) $K = K \cup \{k_m\}$
2.7) $\mathbf{r}_{m+1}^{(s)} = \mathbf{r}_m^{(s)} - [\mathbf{H}^{(s)}]_{k_m} \text{Slice}(\hat{x}_{k_m}) \;\forall\; s \in S$
2.8) $\mathbf{H}_{m+1}^{(s)} = \mathbf{H}_{\overline{K}}^{(s)} \;\forall\; s \in S$
2.9) $m \Leftarrow m + 1$;
```

In step 2.7), "Slice($\cdot$)" is the slicing function, and in step 2.8), $\mathbf{H}_{\overline{K}}^{(s)}$ is the matrix $\mathbf{H}^{(s)}$ with the columns whose indices are in the set K deleted. Therefore, $\mathbf{H}_{m+1}^{(s)}$ is reduced by one column compared to $\mathbf{H}_m^{(s)}$.

After detection in time slot i, the modified channel matrices, the modified received symbols, and the set S should be updated according to the procedure described in Section 6.1.

6.3 Combining algorithms for separate detection

As an alternative to joint detection, symbols can be detected separately after each transmission/retransmission, and in each time slot, the soft symbol values can be combined with other soft copies detected in the previous time slots.

In this section, the combining algorithms for separate detection based on the ZF and MMSE criteria are discussed. Here only the linear detection algorithms are discussed; the corresponding application of these algorithms to the VBLAST structure is straightforward.

Define $\mathbf{C}^{(i)} = \mathbf{H}^{(i)^H} \mathbf{H}^{(i)}$, and let $\mathbf{G}^{(i)}$ denote the post processing matrix. The decision statistics for the symbols detected in time slot i can then be expressed as:

$$\hat{\mathbf{x}}^{(i)} = \mathbf{G}^{(i)} \mathbf{y}^{(i)} = \mathbf{G}^{(i)} \mathbf{C}^{(i)} \mathbf{x}^{(i)} + \mathbf{G}^{(i)} \mathbf{H}^{(i)^H} \mathbf{w}^{(i)}. \tag{6.21}$$

ZF criterion

The processing matrix based on the ZF criterion is $\mathbf{G}^{(i)^{\mathrm{ZF}}} = \mathbf{C}^{(i)^{-1}}$. Define the coefficient $a_n^{(i)} = \left(\mathbf{C}^{(i)^{-1}}\right)_{n,n}^{-\frac{1}{2}}$; this is a real number since $\mathbf{C}$ is a hermitian matrix. Multiplying the decision statistics $\hat{x}_n^{(i)^{\mathrm{ZF}}}$ for the symbols transmitted on layer n by $a_n^{(i)}$ gives

$$a_n^{(i)}\hat{x}_n^{(i)^{\mathrm{ZF}}} = a_n^{(i)}(\mathbf{C}^{(i)^{-1}})_n\mathbf{y}^{(i)} = a_n^{(i)}x_n^{(i)} + \dot{w}_n^{(i)} \tag{6.22}$$

where the noise term is

$$\dot{w}_n^{(i)} = a_n^{(i)}\left(\mathbf{C}^{(i)^{-1}}\right)_n \mathbf{H}^{(i)^H}\mathbf{w}^{(i)} \tag{6.23}$$

and with (6.4), its variance is $E\{\dot{w}_n^{(i)}\dot{w}_n^{(i)^H}\} = \sigma_w^2$.

Let S_n denote the set of indices of the time slots in which the same packet is transmitted on layer n. That is, $x_n^{(s)} = x_n^{(i)}$ for $s \in S_n$. Let $c_n^{(s)}$ denote the post-combining coefficient for the copy detected in time slot s, $s \in S_n$. Then

$$\hat{x}_n^{(i)^{\text{ZF-SD}}} = \sum_{s\in S_n} c_n^{(s)}\left(a_n^{(s)}x_n^{(i)} + \dot{w}_n^{(s)}\right) \tag{6.24}$$

where the superscript "ZF-SD" stands for "zero forcing - separate detection".

Equal gain combining

This is the "post-combining" algorithm in [17]. In this approach, the decision statistics obtained for the same layer from different time slots are summed and normalized by $|S_n|$, which is equivalent to choosing the combining coefficients as $c_n^{(s)^{\text{ZF-EGC-SD}}} = |S_n|^{-1}a_n^{(s)^{-1}} = |S_n|^{-1}\left(\mathbf{C}^{(s)^{-1}}\right)_{n,n}^{\frac{1}{2}}$. It then follows that

$$\hat{x}_n^{(i)^{\text{ZF-EGC-SD}}} = x_n^{(i)} + \frac{1}{|S_n|}\sum_{s\in S_n}\left(\mathbf{C}^{(s)^{-1}}\right)_n \mathbf{H}^{(s)^H}\mathbf{w}^{(s)} \tag{6.25}$$

where the superscript "ZF-EGC-SD" stands for "zero forcing - equal gain combining - separate detection".

With equal gain combining, the SNR for the symbols on layer n, as given by (12) in [17], is

$$\mathrm{SNR}_n^{(i)^{\text{ZF-EGC-SD}}} = \frac{E_s}{\sigma_w^2}\frac{|S_n|^2}{\sum_{s\in S_n}\left(\mathbf{C}^{(s)^{-1}}\right)_{n,n}}. \tag{6.26}$$

Maximal ratio combining

Alternatively, the coefficients can be chosen to maximize the SNR. The coefficients are

$$c_n^{(s)^{\text{ZF-MRC-SD}}} = \frac{a_n^{(s)}}{\sum_{s'\in S_n} a_n^{(s')^2}} \tag{6.27}$$

so that

$$\hat{x}_n^{(i)^{\text{ZF-MRC-SD}}} = \frac{\sum_{s\in S_n}\left(\mathbf{C}^{(s)^{-1}}\right)_{n,n}^{-1}\hat{x}_n^{(s)^{\text{ZF}}}}{\sum_{s'\in S_n}\left(\mathbf{C}^{(s')^{-1}}\right)_{n,n}^{-1}} \tag{6.28}$$

where the superscript "ZF-MRC-SD" stands for "zero forcing - maximum ratio combining - separate detection".

The SNR achieved by MRC is the sum of the SNR achieved in each time slot:

$$\text{SNR}_n^{(i)^{\text{ZF-MRC-SD}}} = \frac{E_s}{\sigma_w^2}\sum_{s\in S_n}\left(\mathbf{C}^{(s)^{-1}}\right)_{n,n}^{-1}. \tag{6.29}$$

MMSE criterion

With the MMSE criterion, the linear post processing matrix that results in the minimum MSE is

$$\mathbf{G}^{(i)^{\text{MMSE}}} = \left(\mathbf{C}^{(i)} + \frac{\sigma_w^2}{E_s}\mathbf{I}_{M_T}\right)^{-1}. \tag{6.30}$$

The decision statistics for the symbols transmitted on layer n in time slot i can be expressed as

$$\begin{aligned}\hat{x}_n^{(i)^{\text{MMSE}}} &= \left(\mathbf{G}^{(i)^{\text{MMSE}}}\right)_n [\mathbf{C}^{(i)}]_n x_n^{(i)} \\ &+ \left(\mathbf{G}^{(i)^{\text{MMSE}}}\right)_n \sum_{u\neq n}[\mathbf{C}^{(i)}]_u x_u^{(i)} \\ &+ \left(\mathbf{G}^{(i)^{\text{MMSE}}}\right)_n \mathbf{H}^{(i)^H}\mathbf{w}^{(i)}.\end{aligned} \tag{6.31}$$

Let $c_n^{(s)}$ denote the post combining coefficient for the copy detected in time slot s, $s \in S_n$. It then follows that

$$\hat{x}_n^{(i)} = \frac{1}{|S_n|}\sum_{s\in S_n} c_n^{(s)}\hat{x}_n^{(i)^{\text{MMSE}}}. \tag{6.32}$$

where $|\cdot|$ denotes the dimension of set "$\cdot$". Instead of equal gain combing, the coefficients can be chosen to minimize the MSE $E\{\|\hat{x}_n^{(i)} - x_n^{(i)}\|^2\}$. It can be shown that these coefficients are

$$\mathbf{c}_n^{(i)} = \mathbf{A}_{n,n}^H\left(\mathbf{A}_{n,n}\mathbf{A}_{n,n}^H + \frac{1}{E_s}E\{\mathbf{B}\mathbf{B}^H\}\right)^{-1} \tag{6.33}$$

where

$$\mathbf{A}_{n,n} = \begin{bmatrix} \left(\mathbf{G}^{(i-T+1)^{\text{MMSE}}}\right)_n \left[\mathbf{C}^{(i-T+1)}\right]_n \\ \vdots \\ \left(\mathbf{G}^{(i)^{\text{MMSE}}}\right)_n \left[\mathbf{C}^{(i)}\right]_n \end{bmatrix} \tag{6.34}$$

and

$$\begin{aligned} \mathbf{B} &= \begin{bmatrix} \sum_{u\neq n}\left(\mathbf{G}^{(i-T+1)^{\mathrm{MMSE}}}\right)_n \left[\mathbf{C}^{(i-T+1)}\right]_u x_u^{(i-T+1)} \\ \vdots \\ \sum_{u\neq n}\left(\mathbf{G}^{(i)^{\mathrm{MMSE}}}\right)_n \left[\mathbf{C}^{(i)}\right]_u x_u^{(i)} \end{bmatrix} \\ &+ \begin{bmatrix} \left(\mathbf{G}^{(i-T+1)^{\mathrm{MMSE}}}\right)_n \mathbf{H}^{(i-T+1)^H}\mathbf{w}^{(i-T+1)} \\ \vdots \\ \left(\mathbf{G}^{(i)^{\mathrm{MMSE}}}\right)_n \mathbf{H}^{(i)^H}\mathbf{w}^{(i)} \end{bmatrix}. \end{aligned} \tag{6.35}$$

The (j,k)-th entry of the covariance of the interference plus noise term $E\{\mathbf{BB}^H\}$ can be expressed as

$$\begin{aligned} \left(E\{\mathbf{BB}^H\}\right)_{j,k} &= E_s\left(\mathbf{G}^{(j')^{\mathrm{MMSE}}}\right)_n \sum_{u\neq n}\left[\mathbf{C}^{(j')}\right]_u \left(\mathrm{I}_{n,u}^{(i)}\right)_{j,k} \\ &\quad \left[\mathbf{C}^{(k')}\right]_u^H \left(\mathbf{G}^{(j')^{\mathrm{MMSE}}}\right)_n^H \\ &+ \sigma_n^2\delta(j-k)\left(\mathbf{G}^{(j')^{\mathrm{MMSE}}}\right)_n \\ &\quad \mathbf{C}^{(j')}\left(\mathbf{G}^{(j')^{\mathrm{MMSE}}}\right)_n^H \end{aligned} \tag{6.36}$$

where, for notational simplicity, $j' = i - T + j$ and $k' = i - T + k$ are defined, and $\mathrm{I}_{n,u}^{(i)}$ is a $T \times T$ identifier matrix such that its (j,k)-th entry indicates whether or not the packets transmitted on layer u during time slots $i - T + j$ and $i - T + k$ are the same, i.e.,

$$\left(\mathrm{I}_{n,u}^{(i)}\right)_{j,k} = \frac{1}{E_s}E\{x_u^{(j')}x_u^{(k')^H}\}. \tag{6.37}$$

The evaluation of this identifier matrix requires a record of transmissions on all the layers.

The MSE achieved using this MMSE combining is

$$\mathrm{MSE} = \mathbf{c}_n^{(i)}E\{\mathbf{BB}^H\}\mathbf{A}_{n,n}^{-H}. \tag{6.38}$$

6.4 Simulations

In the simulations, an $M_T = M_R = 4$ system with QPSK modulation is considered. Each packet contains 112 information bits, and a genie code for error detection is used. For error correction, a 64 state, rate 1/2 convolutional code with generator polynomials $(133, 171)_8$ is used [33]. Six zero bits are appended to the end of each packet to force the encoder back to the all-zero state. Therefore, the codeword for each packet contains 256 bits, and each time slot contains $N = 128$ symbols. In the simulations, SNR is defined as $SNR = \frac{EsM_T}{b\sigma_w^2}$, where b is the number of bits per QAM symbol.

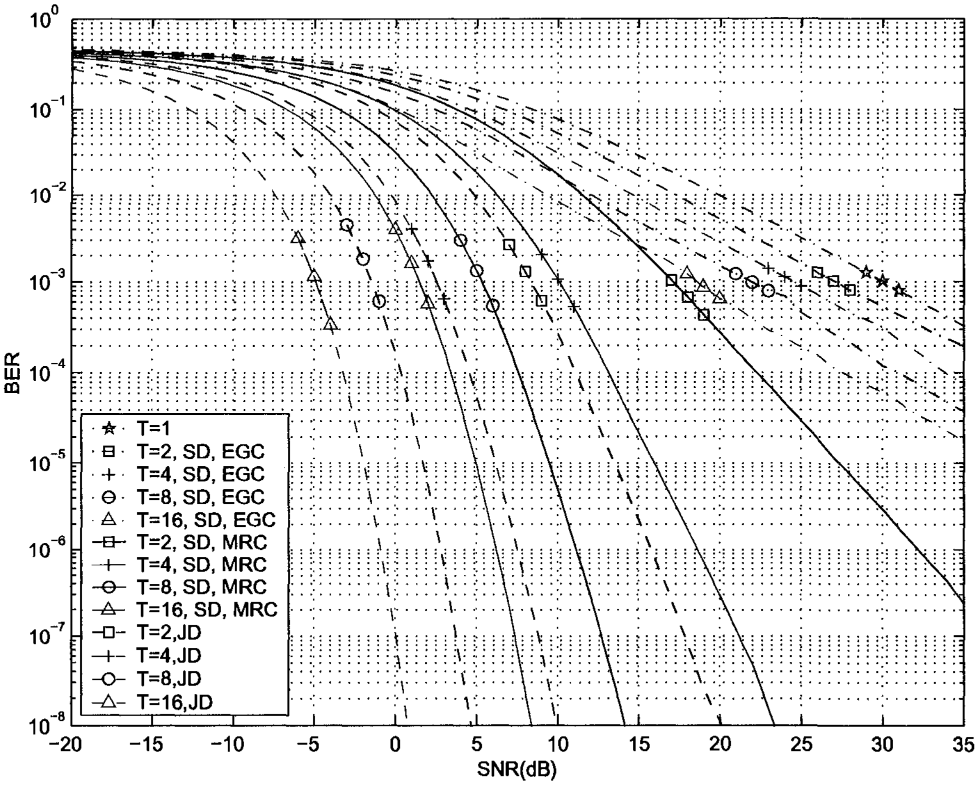


Figure 6.2: BER for linear detection with ZF criterion.

BER performance

First the raw BER (without hybrid ARQ) for the linear detection algorithms discussed in this chapter for different numbers of retransmissions is compared. $T = 1, 2, 4, 8$, and 16 transmissions are considered in the simulations. Figs. 6.2 and 6.3 show the simulation results for the ZF criterion and the MMSE criterion, respectively. In the legend, "SD" stands for separate detection and "JD" stands for joint detection, "EGC" stands for equal gain combining, "MRC" stands for maximal ratio combining and "MMSE C" stands for MMSE combining. A dashed line denotes joint detection, a dash dotted line denotes separate detection with equal gain combining, and a solid line denotes separate detection with maximal ratio combining for the ZF criterion or MMSE combining for the MMSE criterion.

The simulation results show that for separate detection the combining algorithm proposed in this chapter significantly outperforms the equal gain combining algorithm presented previously in the literature. However, the performance is still inferior to that of the joint detection algorithm.

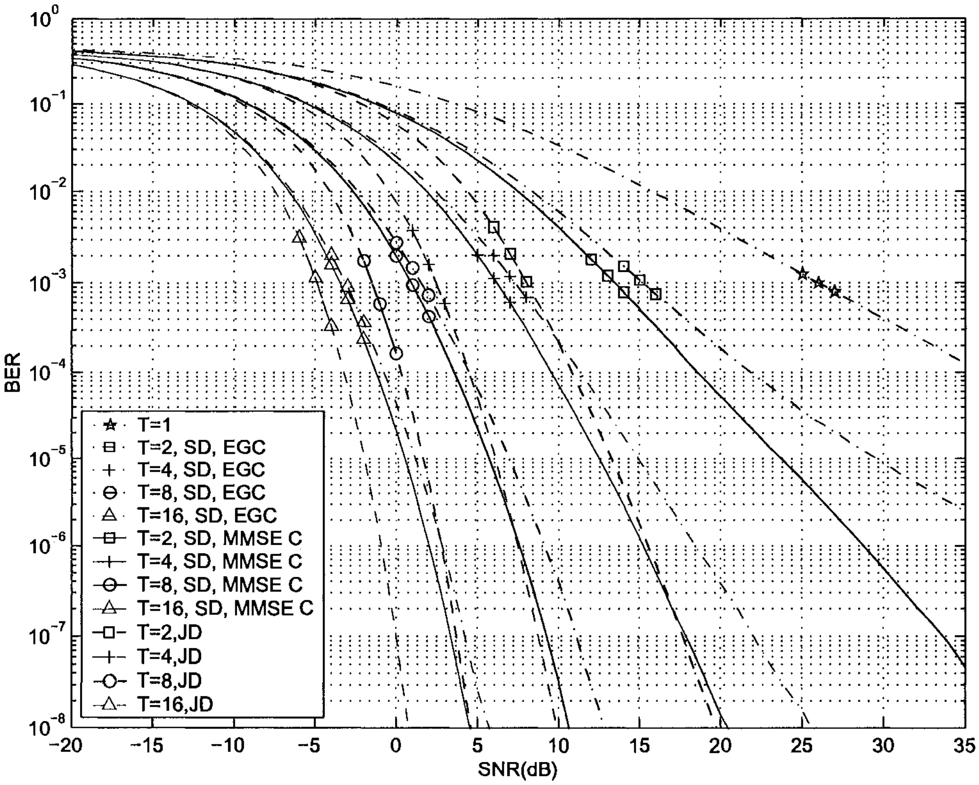


Figure 6.3: BER for linear detection with MMSE criterion.

Throughput

Linear detection

In this section, the throughput performance of multiple hybrid ARQ and single hybrid ARQ with both joint and separate linear detection is compared. For single hybrid ARQ, each codeword is randomly-interleaved before being divided into M_T blocks and transmitted on all antennas in the same sub time slot. A new random interleaving pattern is generated for each packet. The simulations show that this random interleaving scheme performs better than the block interleaving scheme or a scheme without interleaving. At the receiver, the soft decisions made on symbols detected from different antennas are weighted by the SNR before being forwarded to the decoder. For the separate detection scheme, the receiver also performs detection and decoding based only on the symbols received in the current time slot before combining them with previously received copies. The simulations show that this "local check" can bring minor improvement in throughput.

Figs. 6.4 and 6.5 show the simulation results for the ZF and the MMSE criteria, respectively. The results show that single hybrid ARQ provides greater throughput than the multiple hybrid ARQ scheme when SNR > 1 dB for the ZF criterion, and when SNR >

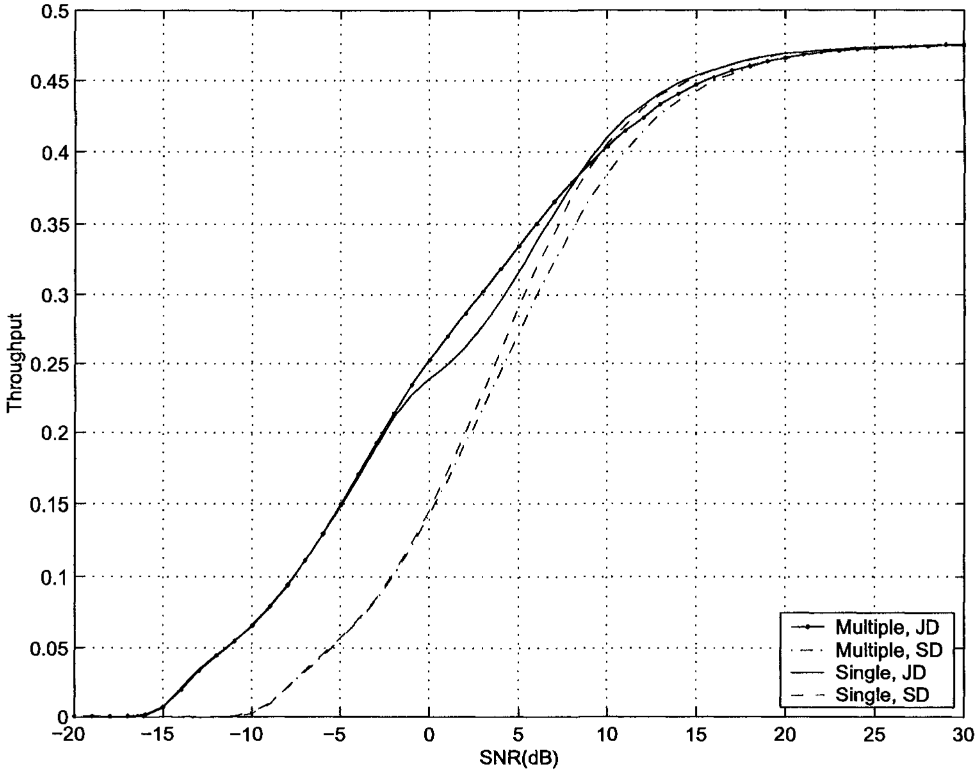


Figure 6.4: Throughput performance for linear detection with ZF criterion.

9.5 dB for the MMSE criterion. This is due to the random interleaving which allows the single hybrid ARQ scheme to exploit the diversity across layers. Joint detection always outperforms separate detection with the exception of the single hybrid ARQ in the high SNR region, where the local check results in some improvement in throughput.

V-BLAST detection

When the V-BLAST structure is used for single hybrid ARQ, the codewords are not interleaved before being divided into M_T blocks. Each codeword is transmitted on one antenna in a sub time slot so as to facilitate interlayer interference cancellation in V-BLAST detection. Simulations show that this scheme results in better performance than the interleaved scheme in which there is no straightforward way to cancel interlayer interference.

For both joint and separate detection, after the symbols on a layer are detected, channel decoding is performed using the (combined) soft decisions, and interlayer interference cancellation is performed using the reencoded symbols. The simulations show that this approach outperforms the scheme where interference cancellation is based on hard decisions on the detected symbols (without decoding and reencoding). Figs. 6.6 and 6.7 show simulation results for the ZF criterion and the MMSE criterion, respectively. These results show

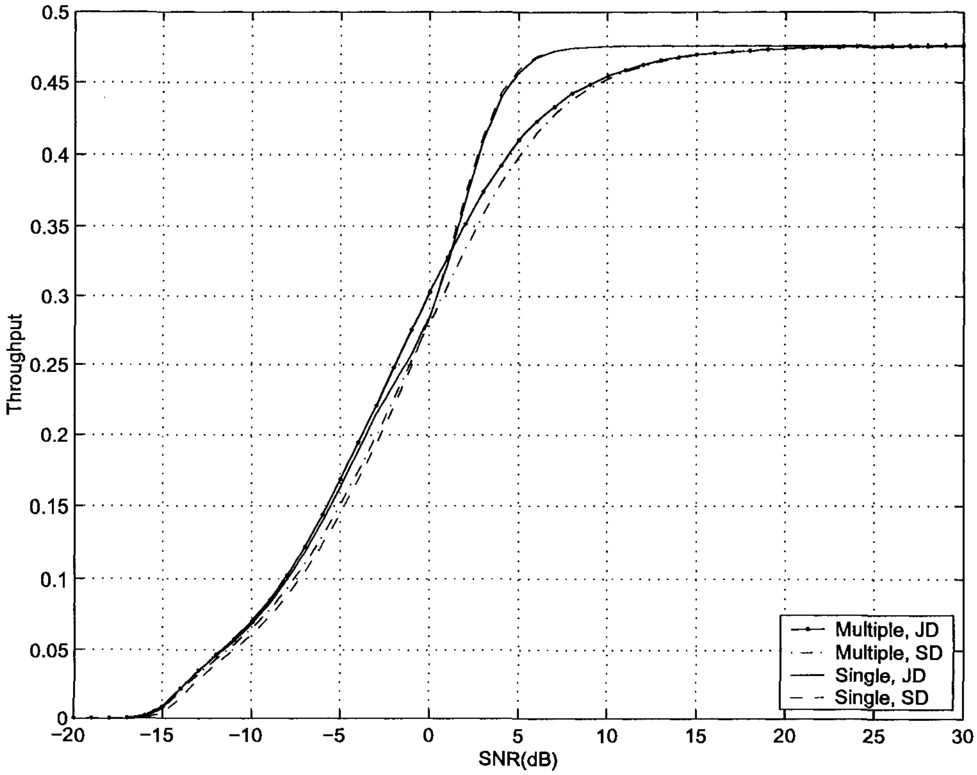


Figure 6.5: Throughput performance for linear detection with MMSE criterion.

that multiple hybrid ARQ always outperforms the single hybrid ARQ scheme and that the joint detection scheme always outperforms separate detection.

6.5 Chapter summary

In this chapter, the multiple and single hybrid ARQ schemes with repetition for spatial multiplexing MIMO systems with channel state information available only at the receiver are compared. A system model for symbol detection for the multiple hybrid ARQ processes is proposed, and the joint detection algorithms and the separate detection algorithms for both multiple hybrid ARQ and single hybrid ARQ are discussed.

The simulation results show that for separate detection the combining algorithm proposed in this chapter significantly outperforms the equal gain combining algorithm presented previously in the literature. However, the performance is still inferior to that of the joint detection algorithm.

When linear detection is employed, because single hybrid ARQ enables interleaving across layers to exploit diversity, single hybrid ARQ outperforms the multiple hybrid ARQ scheme in the high SNR region. However, when the V-BLAST architecture is used, the

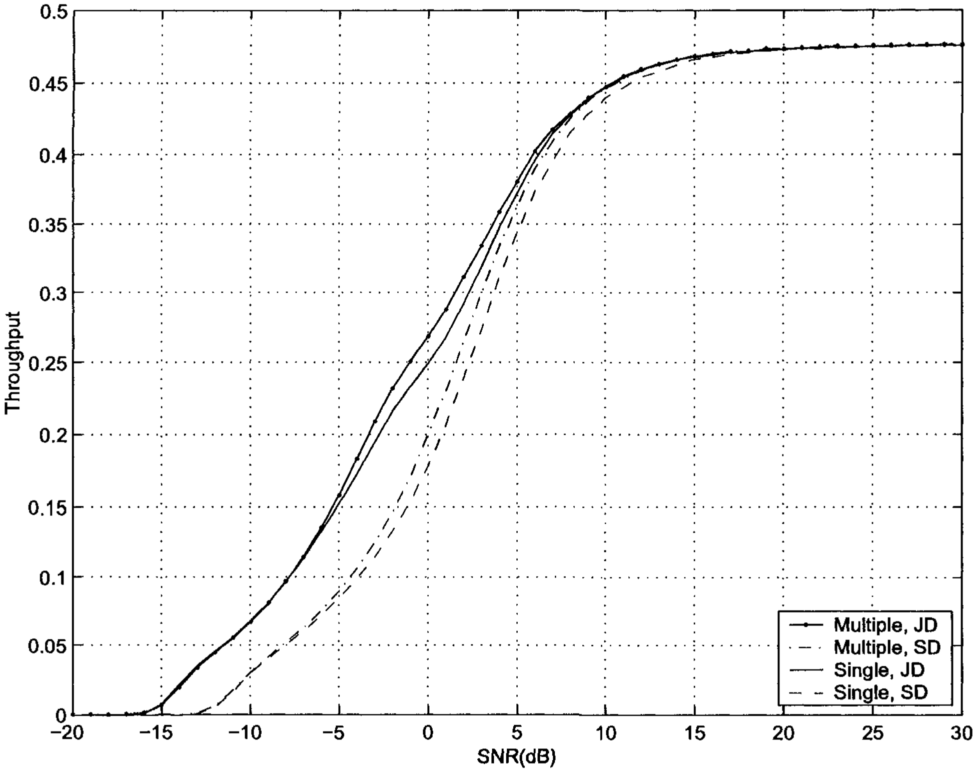


Figure 6.6: Throughput for V-BLAST detection with ZF criterion.

advantage of interleaving is not comparable to the improvement introduced by interlayer interference cancellation, which is possible only in the absence of interleaving across layers. Therefore, with the V-BLAST architecture, the performance of single hybrid ARQ is always inferior to that of the multiple hybrid ARQ scheme.

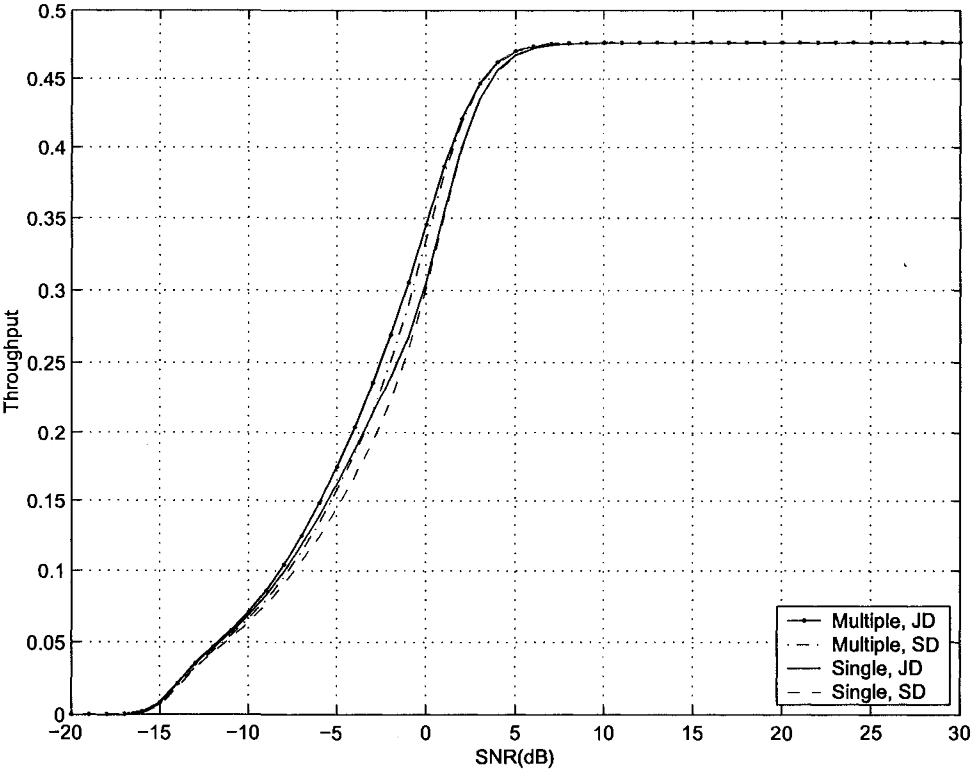


Figure 6.7: Throughput for V-BLAST detection with MMSE criterion.

Chapter 7

Conclusions

In this thesis, techniques that can be used in hybrid ARQ schemes that employ powerful FEC codes such as turbo codes are proposed and investigated.

7.1 Summary of contributions

Improved analysis of list decoding and an efficient list decoding algorithm for turbo codes

The first major contribution reported in this thesis results in more thorough understanding and more accurate analysis of list decoding for linear block codes in general and convolutional codes and turbo codes in particular than treatments reported to date in the literature. By introducing the concept of generalized pairwise error event and effective weight enumerating function, a platform was constructed for more accurate evaluation of the performance of list decoding compared to the traditional "worst case" lower bound analysis. An approach to evaluate the effective weight enumerating function of a particular code with list decoding was then proposed. The effective Euclidean distances for decisions in each pairwise error event are evaluated by taking into consideration the actual Hamming distance relationships between codewords. This approach relaxes the pessimistic assumptions upon which traditional lower bound analysis is based. Geometrical analysis shows that the effective Euclidean distances are not necessarily as low as those predicted by the lower bound.

Analysis of list decoding for feed-forward convolutional codes was then considered. Owing to the time invariant nature of the encoder in feed-forward convolutional codes, the evaluation of their actual effective weight can be simplified and carried out on a signature basis as opposed to a codeword basis. Analysis for feed-forward convolutional codes shows that the actual effective weight does not always equal the lower bound predictions and codes with the same minimum weight may have different minimum effective weights.

It was shown that the actual effective weights for turbo codes can be approximately evaluated by considering only the low weight codewords. Furthermore, analysis of turbo codes showed that the approximate effective weight can be significantly higher than that predicted by the lower bound due to the fact that the multiplicity of the low weight codewords in turbo codes is low. Although a full exploration of this characteristic is not possible since a low complexity optimal list decoding algorithm for turbo codes is not available, the results of this analysis may encourage further research efforts toward improved sub-optimal list decoding algorithms for turbo codes in order to achieve better performance.

Guided by the theoretical analysis, a novel and practical list decoding algorithm is proposed for turbo codes. The error events and weight spectra for convolutional codes and turbo codes are analyzed in order to gain a better understanding of the inefficiencies of list decoding algorithms that avoid generating the list directly for the turbo codes but instead use lists generated for the component codes. Based on this analysis, a novel list decoding algorithm that generates the list directly for the turbo code is proposed. The new algorithm results in a significantly reduced error floor with less computational complexity than list decoding algorithms reported in the literature.

Sub-block recovery for turbo coded systems with the sub-block structure and type II hybrid ARQ

The second major contribution of the research reported in this thesis includes the development of a practical method to achieve better performance based on the sub-block structure and the development of a type II hybrid ARQ scheme based on this method.

A sub-block structure is used in the WCDMA 3G wireless communication system. In the 3GPP physical layer specifications for coding and multiplexing for the WCDMA cellular wireless communication system, a hierarchical data structure is used. The physical layer receives data from the higher layer in the form of transport blocks, to which CRC check bits are added. The transport blocks are then concatenated and divided into data blocks for error correction coding. One data block for error correction coding usually contains several transport blocks, each protected by a CRC check.

In this part of the research, a method was proposed to utilize this structure to improve the decoding performance when a turbo code is used as the error correction code. At almost the same time as our original work on this topic was published [61], Chen, Cao and Chen independently proposed a very similar algorithm [54], which they called "constrained decoding for turbo codes" for reducing the computational complexity of decoding turbo codes with the sub-block structure.

During iterative decoding of turbo codes, it is intuitive that if the correct values of bits in a sub-block are known, then decoding of the rest of the data block will benefit. With the physical layer coding and multiplexing procedure specified by 3GPP, this sub-block recovery scheme is a natural choice for decoding the turbo code in the WCDMA system.

The research work in this area contributes to a more complete understanding of the intuitive reasoning of this result from an analytical approach. Comparison of the decoding performance of a turbo code with the sub-block structure with and without the sub-block recovery scheme was conducted by comparing the performance of several simple codes. The decoding of a typical error pattern was analyzed for both the SOVA and Log-MAP algorithm. Assuming that some of the bits of the error pattern in question are recovered in the sub-block recovery process, the analysis has shown that the sub-block recovery scheme contributes to the correction of a typical error patter in both the SOVA and Log-MAP algorithms. The analysis also shows that the sub-block recovery scheme will not be of benefit to ML decoding. Therefore, when a convolutional code is used as the error correction code and the Viterbi algorithm is used for decoding, the sub-block recovery approach cannot be used to improve the decoding performance.

A Type-II hybrid ARQ with incremental redundancy is then developed for a turbo coded system that uses the sub-block structure and uses various techniques to improve throughput performance. A CRC code is proposed to be selected such that its generator polynomial contains as a factor the feedback polynomial of the RSC constituent encoder of the turbo code. By choosing such a CRC code for error protection, the trellis of the first RSC constituent encoder will return to the all-zero state after each sub-block is encoded. Therefore, the trellis of the first RSC constituent encoder regularly returns to the all-zero state, which results in a frequently terminated turbo code without adding extra terminating bits for each sub-block. One of the many advantages of this frequent termination feature is improved decoding performance due to the awareness of the positions within the trellis which are terminated in the all-zero state. This improved decoding performance will result in fewer retransmissions and lead to increased throughput. The sub-block recovery scheme is then applied to the iterative decoding of turbo codes in order to improve their decoding performance. Simulation results show improved throughput with the proposed scheme when compared to conventional schemes in both the AWGN channel and the uncorrelated flat Rayleigh fading channel.

Hybrid ARQ for MIMO systems

The third major contribution reported in this thesis concerns the use of hybrid ARQ in MIMO systems. In this part of the research, discussion is limited to the layered space time MIMO system and it is assumed that the channel state information is available only at the receiver, and that this channel state information at the receiver is perfect.

A system model for layered space time MIMO transmission that is applicable for multiple ARQ processes is proposed. Symbol level joint detection algorithms were then developed. The performance of these detection algorithms with both the multiple H-ARQ scheme, and the single H-ARQ scheme with repetition were compared via computer simulations. These simulations showed that with linear detection the single H-ARQ outperforms multiple H-ARQ in the high SNR region, while with the V-BLAST architecture, multiple H-ARQ always outperforms single H-ARQ and joint detection always outperforms separate detection.

7.2 Future work

There are a number of different potential areas of future work based on the original research reported in this thesis. In Chapter 2, the analysis for list decoding of turbo codes predicts that the approximate effective weight is significantly higher than that predicted by the lower bound. Although the list decoding algorithm introduced in Chapter 3 generates the list directly for the turbo code and achieves better performance than the list decoding algorithms that use the list generated for the component codes, there is still a significant gap between the performance it achieves and that predicted for the optimal list decoding algorithm. Therefore, it is still of interest to develop other sub-optimal list decoding algorithms in order to achieve performance that approaches the theoretical limit.

By deploying multiple antennas at both the transmitter and the receiver, MIMO wireless systems provide extra degrees of freedom compared to SISO wireless systems. The design of hybrid ARQ transmission schemes for MIMO systems is a broad research area, and only a single topic is touched upon in this thesis, namely, a performance comparison of the multiple ARQ and single ARQ processes assuming that the channel state information is available only at the receiver and the channel state information at the receiver is perfect. It is of interest to investigate further improvements that might be possible if perfect channel state information is also available at the transmitter. It is also of interest to investigate how this information may be used in a progressive linear precoder to organize transmissions and retransmissions as described by Sun, Manton and Ding in [62], and whether in this instance it is advantageous to use the multiple ARQ process or the single ARQ process. Finally, it is of practical interest to investigate what performance is achievable when the channel state information at the receiver and/or the transmitter is not perfect, because this is the situation for which practical ARQ schemes must be designed.

Bibliography

[1] S. Lin, D.J. Costello Jr., and M. Miller, "Automatic-repeat-request error-control schemes," *IEEE Commun. Mag.*, vol. 22, no. 12, pp. 5–17, Dec 1984.

[2] S. B. Wicker, *Error Control Systems for Digital Communication and Storage.* Prentice-Hall, Upper Saddle River, NJ, USA, 1995.

[3] J. Wozencraft and M.Horstein, "Coding for two way channels," Research Laboratory of Electronics, MIT, Tech. Rep., Jan. 1961.

[4] S. Kallel, "Complementary punctured convolutional (CPC) codes and their applications," *IEEE Trans. Commun.*, vol. 43, no. 6, pp. 2005–2009, June 1995.

[5] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389–400, Apr. 1988.

[6] D. Rowitch and L. Milstein, "On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes," *IEEE Trans. Commun.*, vol. 48, no. 6, pp. 948–959, June 2000.

[7] 3GPP, "UTRA high-speed downlink packet access," *TS 25.950 V4.0.0*, Mar. 2001.

[8] ——, "Dual-channel stop and wait H-ARQ," *TSG-R WG2, R2-A010016*, Jan. 2001.

[9] G. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Commun.*, vol. 6, no. 3, pp. 311–335, 1998.

[10] E. Telatar, "Capacity of multi-antenna Gaussian channels," *Eur. Trans. Telecomm.*, vol. 10, no. 6, pp. 585–596, Nov. 1999.

[11] V. Tarokh, N. Seshadri, and A. R. Calderbank, "Space-time codes for high data rate wireless communication: Performance criterion and code construction," *IEEE Trans. Inform. Theory*, vol. 44, pp. 744–765, Mar 1998.

[12] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications.* Cambridge University Press, 2003.

[13] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multiple antennas," *Bell Labs. Tech. J.*, vol. 1, pp. 41–59, Nov. 1996.

[14] L. Zheng and D. Tse, "Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels," *IEEE Trans. Inform. Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.

[15] H. Zheng, A. Lozano, and M. Haleem, "Multiple ARQ processes for MIMO systems," *in Proc. IEEE PIMRC 2002*, vol. 3, pp. 1023–1026, Sept. 2002.

[16] T. Koike, H. Murata, and S. Yoshida, "Hybrid ARQ scheme suitable for coded MIMO transmission," *in Proc. IEEE ICC'04*, vol. 5, pp. 2919–2923, June 2004.

[17] E. Onggosanusi, A. Dabak, Y. Hui, and G. Jeong, "Hybrid ARQ transmission and combining for MIMO systems," *in Proc. IEEE ICC '03*, vol. 5, pp. 3205–3209, USA, May 2003.

[18] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, 1948.

[19] C.B.Schlegel and L. Pérez, *Trellis and Turbo Coding.* IEEE/Wiley, 2004.

[20] L. C. Perez, J. Seghers, and J. Daniel J. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1698–1709, Nov. 1996.

[21] M. C. Valenti, "Iterative detection and decoding for wireless communications," Ph.D. dissertation, Bradley Dept. of Elect. and Comp. Eng., Virginia Tech, Blacksburg, Virginia, USA, July 1999.

[22] 3GPP, "Multiplexing and channel coding," *TS 25.212 V6.2.0*, June 2004.

[23] ——, "High speed downlink packet access (HSDPA); overall description; stage 2," *TS 25.308 V8.0.0*, Dec. 2007.

[24] 3GPP2, "CDMA2000 high rate packet data air interface specification," *C.S0024 V2.0*, Oct. 2000.

[25] T. Rappaport, *Wireless Communications: Principles & Practice*, 2nd ed. Prentice-Hall, 2002.

[26] P. Elias, "List decoding for noisy channels," Research Laboratory of Electronics, MIT, Tech. Rep., Sept. 20, 1957.

[27] J.M.Wozencraft, "List decoding," Research Laboratory of Electronics, MIT, Tech. Rep., Jan. 15, 1958.

[28] N. Seshadri and C.-E. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp. 313–323, Feb./Mar./Apr. 1994.

[29] ——, "Generalized Viterbi algorithms for error detection with convolutional codes," *in Proc. IEEE GLOBECOM89*, pp. 1534–1538, Dallas, Texas, Nov. 1989.

[30] K. R. Narayanan and G. L. Stüber, "List decoding of turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 6, pp. 754–762, June 1998.

[31] C. F. Leanderson and C.-E. W. Sundberg, "On list sequence turbo decoding," *IEEE Trans. Commun.*, vol. 53, no. 5, pp. 760–763, May 2005.

[32] ——, "Performance evaluation of list sequence MAP decoding," *IEEE Trans. Commun.*, vol. 53, no. 3, pp. 422–432, Mar. 2005.

[33] J. Proakis, *Digital Communications*, 4th ed. McGraw-Hill, 2001.

[34] O. Takeshita and D. Costello, Jr., "New deterministic interleaver designs for turbo codes," *IEEE Trans. Inform. Theory*, vol. 46, no. 6, pp. 1988–2006, Sept. 2000.

[35] M. P. C. Fossorier, S. Lin, and D. J. Costello, Jr., "On the weight distribution of terminated convolutional codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1646–1648, July 1999.

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

[37] J. P. Odenwalder, "Optimal decoding of convolutional codes," Ph.D. dissertation, University of California, Los Angeles, 1970.

[38] K. Larsen, "Short convolutional codes with maximal free distance for rates 1/2, 1/3, and 1/4," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 371–372, May 1973.

[39] C. F. Leanderson and C.-E. W. Sundberg, "The Max-Log list algorithm (MLLA): a list sequence decoding algorithm that provides soft symbol output," *IEEE Trans. Commun.*, vol. 53, no. 3, pp. 433–444, Mar 2005.

[40] O. Y. Takeshita, O. M. Collins, P. C. Massey, and D. J. Costello, Jr., "On the frame-error rate of concatenated turbo codes," *IEEE Trans. Commun.*, vol. 49, no. 4, pp. 602–608, Apr. 2001.

[41] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, no. 5, pp. 591–600, May 1996.

[42] P. Robertson and P. Hoher, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *European Trans. on Telecommun.*, vol. 8, no. 2, pp. 119–125, Mar./Apr. 1997.

[43] C. Nill and C.-E. W. Sundberg, "List and soft symbol output Viterbi algorithms: extensions and comparisons," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 277–287, Feb./Mar./Apr. 1995.

[44] N. Guo, F. Khaleghi, A. Gutierrez, J. Li, and M.-H. Fong, "Transmission of high speed data in cdma2000," *in Proc. IEEE WCNC '99*, vol. 3, pp. 1442–1445, New Orleans, USA, Sept. 1999.

[45] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms.* Addison-Wesley, 1983.

[46] S. Sun and W. A. Krzymień, "Hybrid ARQ and optimal signal-to-interference ratio assignment for high-quality data transmission in DS-CDMA," *European Trans. Telecoms*, vol. 12, no. 1, pp. 19–29, Jan./Feb. 2001.

[47] Y. Q. Zhou and J. Wang, "Optimum sub-packet transmission for turbo-coded hybrid ARQ systems," *in Proc. IEEE ICC '03*, vol. 5, pp. 3080–3084, Anchorage, AK, USA, May 2003.

[48] L. Lin and R. S. Cheng, "Improvements in SOVA-based decoding for turbo codes," *in Proc. IEEE ICC'97*, pp. 1473–1478, June 1997.

[49] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and Max-Log-MAP decodings," *IEEE Comm. Lett.*, vol. 2, no. 5, pp. 137–139, May 1998.

[50] S. Crozier and P. Guinand, "Distance bounds and the design of high-distance interleavers for turbo-codes," *in Proc. the 21st Biennial Symposium on Communications,*, pp. 10–14, Kingston, ON, Canada, June 2002.

[51] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *JPL TDA progress report 42-122*, pp. 56–65, Aug. 1995.

[52] A. Shibutani, H. Suda, and F. Adachi, "Multistage recursive interleaver for turbo codes in DS-CDMA mobile radio," *IEEE Trans. Veh. Technol.*, vol. 51, no. 1, pp. 88–100, Jan. 2002.

[53] W. J. Gross and P. G. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *IEE Elect. Lett.*, vol. 34, pp. 1577–1578, Aug. 1998.

[54] H. Chen, L. Cao, and C. W. Chen, "Constrained decoding for turbo-CRC code with high spectral efficient modulation," *in Proc. IEEE WCNC'05*, pp. 1050–1054, New Orleans, LA, USA, Mar. 2005.

[55] B. Mielczarek and A. Svensson, "Joint adaptive rate turbo decoding and synchronization on Rayleigh fading channels," *in Proc. IEEE VTC'01-Spring*, pp. 1342–1346, Rhodes, Greece, May 2001.

[56] F. Zhai and I. J. Fair, "Techniques for early stopping and error detection in turbo decoding," *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1617 –1623, Oct. 2003.

[57] D. Gnaedig, E. Boutillon, M. Jezequel, V. Gaudet, and G. Gulak, "On multiple slice turbo codes," *Annals of telecommunications*, vol. 60, no. 1-2, pp. 79–102, Jan./Feb. 2005.

[58] G. Castagnoli, J. Ganz, and P. Graber, "Optimum cyclic redundancy-check codes with 16-bit redundancy," *IEEE Trans. Commun.*, vol. 38, no. 1, pp. 111–114, Jan. 1990.

[59] P. Wolniansky, G. Foschini, G. Golden, and R. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," *in Proc. URSI ISSSE'98*, pp. 295–300, Pisa, Italy Sept./Oct. 1998.

[60] H. Zheng, A. Lozano, and M. Haleem, "Multiple ARQ processes for MIMO systems," *EURASIP Journal on Applied Signal Processing, Special Issue on MIMO Systems and Signal Processing,*, vol. 2004, no. 5, pp. 772–782, May 2004.

[61] C. Bai, B. Mielczarek, W. A. Krzymień, and I. J. Fair, "Analysis of a sub-block recovery scheme for decoding a concatenated error control code," *in Proc. IEEE VTC'05-Spring*, Stockholm, Sweden, May 2005.

[62] H. Sun, J. H. Manton, and Z. Ding, "Progressive linear precoder optimization for MIMO packet retransmissions," *IEEE Trans. Commun.*, vol. 24, pp. 448–456, Mar. 2006.

Appendix A

Effective weight evaluation for $L = 3$ candidate list decoding

If the distance relations of four signal points can be represented by a tetrahedron in 3D Cartesian coordinates, the evaluation of the effective weight increment can be transformed to the problem of finding the circumradius of the tetrahedron, given $\begin{pmatrix} 4 \\ 2 \end{pmatrix} = 6$ distances between any two out of the four points.

Denote the signal points as A, B, C, and D. In 3D Cartesian coordinates, as shown in Fig. A.1:

- Put A on the origin: $x_{\mathrm{A}} = 0$, $y_{\mathrm{A}} = 0$, $z_{\mathrm{A}} = 0$;
- Put B on the x-axis: $x_{\mathrm{B}} = D_{\mathrm{AB}}$, $y_{\mathrm{B}} = 0$, $z_{\mathrm{B}} = 0$;
- Put C in the x-o-y plane with y index positive: $y_{\mathrm{C}} > 0$, $z_{\mathrm{C}} = 0$;
- Put D in the space upper to the x-o-y plane: $z_{\mathrm{D}} > 0$.

C is the intersection of two circles in x-o-y plane with center at A and B, with radii D_{AC}

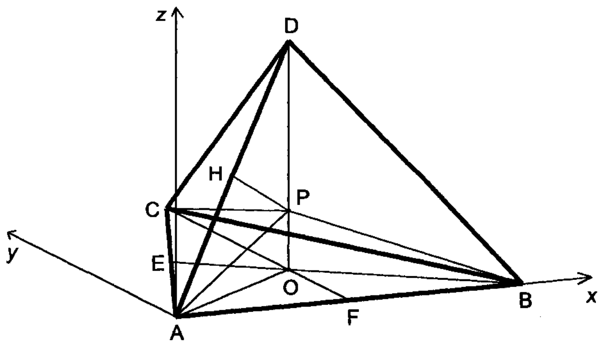


Figure A.1: Effective weight evaluation for $L = 3$ candidate list decoding.

and D_{BC}, respectively. Solving

$$\left\{\begin{array}{rl} x_{\mathrm{C}}^2 + y_{\mathrm{C}}^2 & = D_{\mathrm{AC}}^2 \\ (x_{\mathrm{C}} - x_{\mathrm{B}})^2 + (y_{\mathrm{C}} - y_{\mathrm{B}})^2 & = D_{\mathrm{BC}}^2 \\ z_{\mathrm{C}} & = 0 \\ y_{\mathrm{C}} & > 0 \end{array}\right. \tag{A.1}$$

yields $x_{\mathrm{C}} = \frac{D_{\mathrm{AC}}^2 + D_{\mathrm{AB}}^2 - D_{\mathrm{BC}}^2}{2D_{\mathrm{AB}}}$, $y_{\mathrm{C}} = \sqrt{D_{\mathrm{AC}}^2 - x_{\mathrm{C}}^2}$, $z_{\mathrm{C}} = 0$.

D is the intersection of three spheres with centers at A, B, and C, with radii D_{AD}, D_{BD} and D_{CD}, respectively. Solving

$$\left\{\begin{array}{rl} (x_{\mathrm{D}} - x_{\mathrm{A}})^2 + (y_{\mathrm{D}} - y_{\mathrm{A}})^2 + (z_{\mathrm{D}} - z_{\mathrm{A}})^2 & = D_{\mathrm{AD}}^2 \\ (x_{\mathrm{D}} - x_{\mathrm{B}})^2 + (y_{\mathrm{D}} - y_{\mathrm{B}})^2 + (z_{\mathrm{D}} - z_{\mathrm{B}})^2 & = D_{\mathrm{BD}}^2 \\ (x_{\mathrm{D}} - x_{\mathrm{C}})^2 + (y_{\mathrm{D}} - y_{\mathrm{C}})^2 + (z_{\mathrm{D}} - z_{\mathrm{C}})^2 & = D_{\mathrm{CD}}^2 \\ z_{\mathrm{D}} & > 0 \end{array}\right. \tag{A.2}$$

yields $x_{\mathrm{D}} = \frac{D_{\mathrm{AD}}^2 + D_{\mathrm{AB}}^2 - D_{\mathrm{BD}}^2}{2D_{\mathrm{AB}}}$, $y_{\mathrm{D}} = \frac{D_{\mathrm{AC}}^2 + D_{\mathrm{AD}}^2 - D_{\mathrm{CD}}^2 - 2x_{\mathrm{C}}x_{\mathrm{D}}}{2y_{\mathrm{C}}}$, $z_{\mathrm{D}} = \sqrt{D_{\mathrm{AD}}^2 - x_{\mathrm{D}}^2 - y_{\mathrm{D}}^2}$.

Let P be the center of the circumsphere of the tetrahedron. Then $D_{\mathrm{AP}} = D_{\mathrm{BP}} = D_{\mathrm{CP}} = D_{\mathrm{DP}}$ equals half the effective Euclidean distance.

The line passing P and perpendicular to the x-o-y plane intersects the x-o-y plane at O. The line OP is

$$x = x_{\mathrm{P}}, \quad y = y_{\mathrm{P}}. \tag{A.3}$$

The midpoints of the edges AB and AC, denoted by F and E, have $x_{\mathrm{F}} = \frac{x_{\mathrm{A}}+x_{\mathrm{B}}}{2}$, $y_{\mathrm{F}} = 0$, $z_{\mathrm{F}} = 0$ and $x_{\mathrm{E}} = \frac{x_{\mathrm{A}}+x_{\mathrm{C}}}{2}$, $y_{\mathrm{E}} = \frac{y_{\mathrm{A}}+y_{\mathrm{C}}}{2}$, $z_{\mathrm{E}} = \frac{z_{\mathrm{A}}+z_{\mathrm{C}}}{2}$, respectively. Triangle APO, BPO, and CPO are equal triangles, where $D_{\mathrm{AO}} = D_{\mathrm{BO}} = D_{\mathrm{CO}}$. O is the center of the circumsphere of triangle ABC which is also the intersection of the lines OF and OE, the perpendicular bisectors of the edges AB and AC, respectively. The lines OF and OE are, respectively

$$x = \frac{x_{\mathrm{F}}}{2}, \quad z = 0, \tag{A.4}$$

$$(x_{\mathrm{C}} - x_{\mathrm{A}})(x - x_{\mathrm{E}}) + (y_{\mathrm{C}} - y_{\mathrm{A}})(y - y_{\mathrm{E}}) = 0, \quad z = 0. \tag{A.5}$$

Solving (A.4) and (A.5) yields the indices of point O as $x_{\mathrm{O}} = \frac{D_{\mathrm{AB}}}{2}$, $y_{\mathrm{O}} = \frac{D_{\mathrm{AC}}^2 - 2x_{\mathrm{C}}x_{\mathrm{O}}}{2y_{\mathrm{C}}}$, $z_{\mathrm{O}} = 0$.

Point P is the intersection of the line PO and the plane perpendicular to line AD and passing through its midpoint H, which is described by the equation

$$(x_{\mathrm{D}} - x_{\mathrm{A}})(x - x_{\mathrm{H}}) + (y_{\mathrm{D}} - y_{\mathrm{A}})(y - y_{\mathrm{H}}) + (z_{\mathrm{D}} - z_{\mathrm{A}})(z - z_{\mathrm{H}}) = 0 \tag{A.6}$$

where H has $x_{\mathrm{H}} = \frac{x_{\mathrm{A}}+x_{\mathrm{D}}}{2}$, $y_{\mathrm{H}} = \frac{y_{\mathrm{A}}+y_{\mathrm{D}}}{2}$, $z_{\mathrm{H}} = \frac{z_{\mathrm{A}}+z_{\mathrm{D}}}{2}$. Jointly solving (A.3) and (A.6) yields $x_{\mathrm{P}} = x_{\mathrm{O}}$, $y_{\mathrm{P}} = y_{\mathrm{O}}$, $z_{\mathrm{P}} = \frac{D_{\mathrm{AD}}^2 - 2x_{\mathrm{P}}x_{\mathrm{D}} - 2y_{\mathrm{P}}y_{\mathrm{D}}}{2z_{\mathrm{D}}}$.

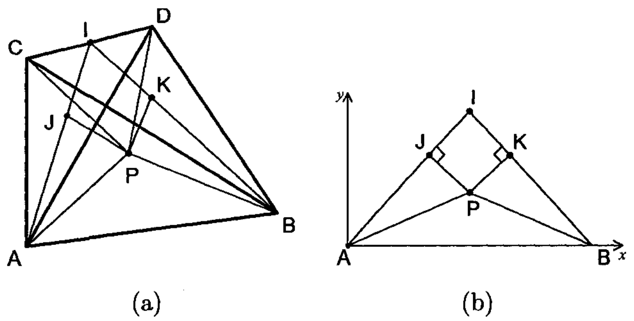


Figure A.2: Effective weight evaluation for $L = 3$ candidate list decoding, where $D_{\mathrm{AC}} = D_{\mathrm{BC}} = D_{\mathrm{AD}} = D_{\mathrm{BD}} = D_{\mathrm{CD}}$.

The circumradius of the tetrahedron ABCD, D_{AP} is

$$D_{\mathrm{AP}} = \sqrt{x_{\mathrm{P}}^2 + y_{\mathrm{P}}^2 + z_{\mathrm{P}}^2}. \tag{A.7}$$

The effective Euclidean distance is the diameter of the circumsphere of the tetrahedron ABCD. The effective weight is

$$\Delta d_j^{(3)} = \frac{(2D_{\mathrm{AP}})^2}{4E_s} = \frac{x_{\mathrm{P}}^2 + y_{\mathrm{P}}^2 + z_{\mathrm{P}}^2}{E_s} \tag{A.8}$$

which, with the previously derived expressions for the coordinates of point C, D, O and P, can be represented as a function of D_{AC} through D_{CD}.

Alternatively, for the special case where $D_{\mathrm{AC}} = D_{\mathrm{BC}} = D_{\mathrm{AD}} = D_{\mathrm{BD}} = D_{\mathrm{CD}}$ and $d_{\mathrm{AB}} = d_j \geq d_{\mathrm{min}}$, a less complex derivation is as follows.

Denote $D_1 = \sqrt{4d_{\mathrm{min}}E_s}$, $D_2 = \sqrt{4d_jE_s}$. As shown in Fig. A.2(a), let P be the center of the circumsphere of the tetrahedron. $D_{\mathrm{AP}} = D_{\mathrm{BP}} = D_{\mathrm{CP}} = D_{\mathrm{DP}}$ equals half the effective Euclidean distance. Line PJ is perpendicular to the plane ACD at J. Line PK is perpendicular to the plane BCD at K. It is easy to show that J and K are the centers of equilateral triangles ACD and BCD, respectively. Line AJ and BK intersects at I, which is the midpoint of edge CD. Based on the equilateral triangle ACD, $D_{\mathrm{AI}} = \frac{\sqrt{3}}{2}D_1$, $D_{\mathrm{AJ}} = \frac{2}{3}D_{\mathrm{AI}}$.

As shown in Fig. A.2(b), with triangle ABI in the x-o-y plane

$$x_{\mathrm{I}} = \frac{D_2}{2}, y_{\mathrm{I}} = \sqrt{D_{\mathrm{AI}}^2 - (\frac{D_{\mathrm{AB}}}{2})^2} = \frac{1}{2}\sqrt{3D_1^2 - D_2^2} \tag{A.9}$$

$$x_J = \frac{2}{3}x_{\mathrm{I}} = \frac{1}{3}D_2, y_J = \frac{2}{3}y_{\mathrm{I}} = \frac{1}{3}\sqrt{3D_1^2 - D_2^2}. \tag{A.10}$$

Line PJ is $(x - x_{\mathrm{J}})(x_{\mathrm{I}} - x_{\mathrm{A}}) + (y - y_{\mathrm{J}})(y_{\mathrm{I}} - y_{\mathrm{A}}) = 0$. Due to $D_{\mathrm{AP}} = D_{\mathrm{BP}}$, $x_{\mathrm{P}} = \frac{1}{2}x_{\mathrm{B}} = \frac{D_2}{2}$,

$y_{\mathrm{P}} = \frac{2D_1^2 - D_2^2}{2\sqrt{3D_1^2 - D_2^2}}$,

$$D_{\mathrm{AP}} = \sqrt{(x_{\mathrm{P}} - x_{\mathrm{A}})^2 + (y_{\mathrm{P}} - y_{\mathrm{A}})^2} = \frac{1}{2}\sqrt{\frac{4D_1^2 - D_2^2}{3D_1^2 - D_2^2}} D_1, \tag{A.11}$$

and the effective weight is

$$d_{j,\mathrm{LB}}^{(3)} = \frac{(2D_{\mathrm{AP}})^2}{4E_s} = \frac{4d_{\min} - d_j}{3d_{\min} - d_j} d_{\min}. \tag{A.12}$$

Appendix B

Derivation of several equations for Chapter 4

Equation 4.15

From (4.1), for $n \in \mathbf{n}_s$

$$\mathrm{P}^{1,i}(x_n) = \begin{cases} \frac{\exp(L_{(a^{1,i})_n})}{1+\exp(L_{(a^{1,i})_n})}, & x_n = 1 \\ \frac{1}{1+\exp(L_{(a^{1,i})_n})}, & x_n = -1 \end{cases} \tag{B.1}$$

$$\mathrm{P}^{1,i}(x_n) = \frac{1}{1+\exp\left(L_{(a^{1,i})_n}\right)} \exp\left[\frac{1}{2} L_{(a^{1,i})_n}(1+x_n)\right]. \tag{B.2}$$

Assuming that the *a priori* information for each symbol is independent,

$$\begin{aligned} & \mathrm{P}^{1,i}(\mathbf{x}_{\mathrm{sys}}) \\ = & \prod_{n\in\mathbf{n}_s} \mathrm{P}^{1,i}(x_n) \\ = & \left[\prod_{n\in\mathbf{n}_s} \frac{1}{1+\exp\left(L_{(a^{1,i})_n}\right)}\right] \exp\left[\frac{1}{2}\sum_{n\in\mathbf{n}_s} L_{(a^{1,i})_n}(1+x_n)\right] \\ = & \left[\prod_{n\in\mathbf{n}_s} \frac{1}{1+\exp\left(L_{(a^{1,i})_n}\right)}\right] \exp\left(\frac{1}{2}\sum_{n\in\mathbf{n}_s} L_{(a^{1,i})_n}\right) \exp\left(\frac{1}{2}\sum_{n\in\mathbf{n}_s} L_{(a^{1,i})_n} x_n\right) \\ = & C_2 \exp\left(\frac{1}{2}\sum_{n\in\mathbf{n}_s} L_{(a^{1,i})_n} x_n\right) \end{aligned} \tag{B.3}$$

where $C_2 = \left[\prod_{n\in\mathbf{n}_s} \frac{1}{1+\exp(L_{(a^{1,i})_n})}\right] \exp\left(\frac{1}{2}\sum_{n\in\mathbf{n}_s} L_{(a^{1,i})_n}\right)$ is a constant independent of $\mathbf{x}_{\mathrm{sys}}$ for a given *a priori* information sequence.

Let $\sigma = \sqrt{\frac{N_0}{2}}$. Based on the assumption of AWGN channel,

$$\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}} | \mathbf{x}_{\mathrm{sys,par1}})$$

$$
\begin{aligned}
&= \frac{1}{(\sigma\sqrt{2\pi})^{2K}} \exp\left(-\frac{1}{2\sigma^2}||\mathbf{y}_{\mathrm{sys,par1}} - \mathbf{x}_{\mathrm{sys,par1}}||^2\right) \\
&= \frac{1}{(\sigma\sqrt{2\pi})^{2K}} \exp\left(-\frac{1}{2\sigma^2}\sum_{n\in\mathbf{n}_s\cup\mathbf{n}_{p1}} |y_n - x_n|^2\right) \\
&= \frac{1}{(\sigma\sqrt{2\pi})^{2K}} \exp\left[-\frac{1}{2\sigma^2}\sum_{n\in\mathbf{n}_s\cup\mathbf{n}_{p1}} (x_n^2 + y_n^2 - 2x_n y_n)\right] \\
&= \frac{1}{(\sigma\sqrt{2\pi})^{2K}} \exp\left[-\frac{1}{2\sigma^2}\sum_{n\in\mathbf{n}_s\cup\mathbf{n}_{p1}} (x_n^2 + y_n^2)\right] \exp\left[\frac{1}{2}\sum_{n\in\mathbf{n}_s\cup\mathbf{n}_{p1}} x_n(\frac{2}{\sigma^2}y_n)\right] \\
&= C_3 \exp\left[\frac{1}{2}\sum_{n\in\mathbf{n}_s\cup\mathbf{n}_{p1}} x_n(\frac{2}{\sigma^2}y_n)\right] \\
&= C_3 \exp\left(\frac{1}{2}\sum_{n\in\mathbf{n}_s\cup\mathbf{n}_{p1}} x_n L_{(y)_n}\right) \qquad \text{(B.4)}
\end{aligned}
$$

where $C_3 = \frac{1}{(\sigma\sqrt{2\pi})^{2K}} \exp\left[-\frac{1}{2\sigma^2}\sum_{n\in\mathbf{n}_s\cup\mathbf{n}_{p1}} (x_n^2 + y_n^2)\right]$ is a constant independent of $\mathbf{x}_{\mathrm{sys,par1}}$ for a given received symbol sequence.

From (B.3) and (B.4),

$$
\begin{aligned}
&\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}}|\mathbf{x}^p_{\mathrm{sys,par1}})\mathrm{P}^{1,i}(\mathbf{x}^p_{\mathrm{sys}}) \\
&= C_2 C_3 \exp\left(\frac{1}{2}\sum_{n\in\mathbf{n}_s} L_{(a^{1,i})_n} x_n^p\right) \exp\left(\frac{1}{2}\sum_{n\in\mathbf{n}_s\cup\mathbf{n}_{p1}} x_n^p L_{(y)_n}\right) \\
&= C_2 C_3 \exp\left[\frac{1}{2}\sum_{n\in\mathbf{n}_s} x_n^p (L_{(a^{1,i})_n} + L_{(y)_n}) + \frac{1}{2}\sum_{n\in\mathbf{n}_{p1}} x_n^p \cdot L_{(y)_n}\right]. \qquad \text{(B.5)}
\end{aligned}
$$

For DEC1 in the i-th iteration, define the metric between $\mathbf{x}^p$ and $\mathbf{y}$ as

$$
\mathrm{M}^p_{\mathrm{DEC1},i} = \sum_{m\in\mathbf{m}_s} x_m^p \cdot (L_{(a^{1,i})_m} + L_{(y)_m}) + \sum_{m\in\mathbf{m}_{p1}} x_m^p \cdot L_{(y)_m}. \qquad \text{(B.6)}
$$

From (B.5),

$$
\mathrm{M}^p_{\mathrm{DEC1},i} = 2\ln\frac{1}{\kappa_1}\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}}|\mathbf{x}^p_{\mathrm{sys,par1}}). \qquad \text{(B.7)}
$$

Where κ_1 is defined as $\kappa_1 = C_2 C_3$.

Equation 4.20

By definition

$$
\begin{aligned}
L_{(\hat{u}^{1,i})_k} &= 2\ln\frac{\max_{\mathbf{u}^{p_a}:u_k^{p_a}=1}\mathrm{P}^{1,i}(\mathbf{u}^{p_a}|\mathbf{y}_{\mathrm{sys,par1}})}{\max_{\mathbf{u}^{p_a}:u_k^{p_a}=0}\mathrm{P}^{1,i}(\mathbf{u}^{p_a}|\mathbf{y}_{\mathrm{sys,par1}})}\\
&= 2\ln\frac{\max_{\mathbf{u}^{p_a}:u_k^{p_a}=1}\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}}|\mathbf{x}^{p_a}_{\mathrm{sys,par1}})\mathrm{P}^{1,i}(\mathbf{x}^{p_a}_{\mathrm{sys}})}{\max_{\mathbf{u}^{p_a}:u_k^{p_a}=0}\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}}|\mathbf{x}^{p_a}_{\mathrm{sys,par1}})\mathrm{P}^{1,i}(\mathbf{x}^{p_a}_{\mathrm{sys}})}\\
&= \max_{\mathbf{u}^{p_a}:u_k^{p_a}=1} 2\ln\left[\frac{1}{C_2C_3}\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}}|\mathbf{x}^{p_a}_{\mathrm{sys,par1}})\mathrm{P}^{1,i}(\mathbf{x}^{p_a}_{\mathrm{sys}})\right]\\
&\quad - \max_{\mathbf{u}^{p_a}:u_k^{p_a}=0} 2\ln\left[\frac{1}{C_2C_3}\mathrm{P}(\mathbf{y}_{\mathrm{sys,par1}}|\mathbf{x}^{p_a}_{\mathrm{sys,par1}})\mathrm{P}^{1,i}(\mathbf{x}^{p_a}_{\mathrm{sys}})\right]\\
&= \max_{\mathbf{u}^{p_a}:u_k^{p_a}=1}\mathrm{M}^{p_a}_{\mathrm{DEC1},i} - \max_{\mathbf{u}^{p_b}:u_k^{p_b}=0}\mathrm{M}^{p_b}_{\mathrm{DEC1},i} && \text{(B.8)}\\
&= (2u_k^{p^{1,i}_{\mathrm{APP}}}-1)\min_{\substack{u_k^{p_a}=1;\,u_k^{p_b}=0\\ p_a=p^{1,i}_{\mathrm{APP}}\text{ or }p_b=p^{1,i}_{\mathrm{APP}}}}\left|\Delta^{(p_a,p_b)}_{\mathrm{DEC1},i}\right|. && \text{(B.9)}
\end{aligned}
$$

One of the max operations in (B.8) results in the metric of the APP solution, as can be translated to the constrain $p_a = p^{1,i}_{\mathrm{APP}}$ or $p_b = p^{1,i}_{\mathrm{APP}}$ of the min operation in (B.9), therefore, the equality between (B.8) and (B.9) can be justified by considering the two possibilities as follows:

- Suppose $u_k^{p^{1,i}_{\mathrm{APP}}} = 1$, (B.8) can be rewritten as

$$
\begin{aligned}
&\max_{\mathbf{u}^{p_a}:u_k^{p_a}=1}\mathrm{M}^{p_a}_{\mathrm{DEC1},i} - \max_{\mathbf{u}^{p_b}:u_k^{p_b}=0}\mathrm{M}^{p_b}_{\mathrm{DEC1},i}\\
=\;& \mathrm{M}^{p^{1,i}_{\mathrm{APP}}}_{\mathrm{DEC1},i} - \max_{\mathbf{u}^{p_b}:u_k^{p_b}=0}\mathrm{M}^{p_b}_{\mathrm{DEC1},i}\\
=\;& \min_{\mathbf{u}^{p_b}:u_k^{p_b}=0}(\mathrm{M}^{p^{1,i}_{\mathrm{APP}}}_{\mathrm{DEC1},i} - \mathrm{M}^{p_b}_{\mathrm{DEC1},i})\\
=\;& \min_{\mathbf{u}^{p_b}:u_k^{p_b}=0}\Delta^{(p^{1,i}_{\mathrm{APP}},p_b)}_{\mathrm{DEC1},i}\\
=\;& \min_{\substack{u_k^{p_a}=1;\,u_k^{p_b}=0\\ p_a=p^{1,i}_{\mathrm{APP}}\text{ or }p_b=p^{1,i}_{\mathrm{APP}}}}\Delta^{(p_a,p_b)}_{\mathrm{DEC1},i}\\
=\;& (2u_k^{p^{1,i}_{\mathrm{APP}}}-1)\min_{\substack{u_k^{p_a}=1;\,u_k^{p_b}=0\\ p_a=p^{1,i}_{\mathrm{APP}}\text{ or }p_b=p^{1,i}_{\mathrm{APP}}}}\left|\Delta^{(p_a,p_b)}_{\mathrm{DEC1},i}\right|. && \text{(B.10)}
\end{aligned}
$$

- Suppose $u_k^{p^{1,i}_{\mathrm{APP}}} = 0$, (B.8) can be rewritten as

$$
\max_{\mathbf{u}^{p_a}:u_k^{p_a}=1}\mathrm{M}^{p_a}_{\mathrm{DEC1},i} - \max_{\mathbf{u}^{p_b}:u_k^{p_b}=0}\mathrm{M}^{p_b}_{\mathrm{DEC1},i}
$$

$$\begin{aligned}
&= \max_{\mathbf{u}^{p_a}:u_k^{p_a}=1} \mathsf{M}^{p_a}_{\mathrm{DEC1},i} - \mathsf{M}^{p^{1,i}_{\mathrm{APP}}}_{\mathrm{DEC1},i} \\
&= \max_{\mathbf{u}^{p_a}:u_k^{p_a}=1} \left(\mathsf{M}^{p_a}_{\mathrm{DEC1},i} - \mathsf{M}^{p^{1,i}_{\mathrm{APP}}}_{\mathrm{DEC1},i}\right) \\
&= \max_{\mathbf{u}^{p_a}:u_k^{p_a}=1} \Delta^{(p_a,p^{1,i}_{\mathrm{APP}})}_{\mathrm{DEC1},i} \\
&= (-1) \min_{\mathbf{u}^{p_a}:u_k^{p_a}=1} \Delta^{(p^{1,i}_{\mathrm{APP}},p_a)}_{\mathrm{DEC1},i} \\
&= (-1) \min_{\substack{u_k^{p_a}=1;\, u_k^{p_b}=0 \\ p_a = p^{1,i}_{\mathrm{APP}} \text{ or } p_b = p^{1,i}_{\mathrm{APP}}}} \Delta^{(p_b,p_a)}_{\mathrm{DEC1},i} \\
&= (2u_k^{p^{1,i}_{\mathrm{APP}}} - 1) \min_{\substack{u_k^{p_a}=1;\, u_k^{p_b}=0 \\ p_a = p^{1,i}_{\mathrm{APP}} \text{ or } p_b = p^{1,i}_{\mathrm{APP}}}} \left|\Delta^{(p_a,p_b)}_{\mathrm{DEC1},i}\right| \qquad \text{(B.11)}
\end{aligned}$$

Equations B.10 and B.11 show the equivalence of (B.8) and (B.9), therefore, (4.20) follows.