Iterative Decoding

Thesis by

Jung-Fu Cheng

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1997
(Submitted March 7, 1997)

Abstract

Though coding theory suggests long error correcting codes chosen at random perform close to the optimum, the problem of designing good codes has traditionally been attacked by developing codes with a lot of structure, which lends itself to feasible decoders. The challenge to find practical decoders for long random codes has not been seriously considered until the recent introduction of turbo codes in 1993. This methodology of multi-stage iterative decoding with exchange of soft information, applied to codes with pseudo-random structure, has provided a whole new approach to construct good codes and to decode them with low complexity. This thesis examines the theoretical ground as well as the design and implementation details of these iterative decoding techniques. The methodology is first applied to parallel concatenated unit-memory convolutional codes and generalized concatenated convolutional codes to demonstrate its power and the general design principle. We then show that, by representing these coding systems with appropriate Bayesian belief networks, all the *ad hoc* algorithms can be derived from a general statistical inference belief propagation algorithm. A class of new binary codes based on low-density generator matrices is proposed to eliminate the arbitrariness and unnecessary constraints in turbo coding we have recognized from this Bayesian network viewpoint. Contrary to the turbo decoding paradigm where sequential processing is accomplished by very powerful central units, the decoding algorithm for the new code is highly parallel and distributive. We also apply these codes to M-ary modulations using multilevel coding techniques to achieve higher spectral efficiency. In all cases, we have constructed systems with flexible error protection capability and performance within 1 dB of the channel capacity.

Acknowledgements

It is impossible to express in words of any length my gratefulness and appreciation to my advisor Professor Robert J. McEliece, who offered me an unparalleled opportunity to study at Caltech and spent numerous hours with me for my research. Working with him is a true privilege, and I have benefited tremendously from his knowledge of science, both in depth and broadness, and his enthusiasm of doing research. While being a constant source of valuable insights and advice, he has always allowed me high degree of freedom and independence to pursue my own research and study. It was, however, his unlimited encouragement, patience, and availability that kept me focused and made all the accomplishments in this thesis possible.

My special thanks go to Prof. Dariush Divsalar, who has been given me invaluable suggestions and comments on my research. I am also grateful to Prof. Andrea Goldsmith, Prof. Aron Kiely, Prof. P. P. Vaidyanathan, Prof. Marvin Simon, and Prof. Mani Chandy for their interest in my work, their helpful comments, and their precious time serving on my candidacy/defense examinations committees. I have also learned immensely from many state-of-the-art classes given by them.

I would like to thank my friends, especially James Tong and Masayuki Hattori, for their friendship and encouragement for me. Without them, life at Caltech would have been very difficult. I am grateful to my colleagues, Zhong Yu, Mohamed-Slim Alouini, Wei Lin, Gavin Horn, and Srinivas Aji for many fruitful discussions I had with them. I thank Lilian Porter, our secretary, for her professional help for my administrative obligations and for her care and kindness toward me. Robert Freeman, our system administrator, who devoted great amount of time to maintain the computers, has been an indispensable resource for my work.

Above all, I would like to express my greatest gratitude to my family: my mother Yumei, my brother Chung-Hong, my cousin Mulan, and my newly-wedded wife Kay. (I wish I could share this delightful moment with my father, who had passed months before I came to Caltech.) They supported me in every aspect of my life and encouraged me every step of the way. They enriched both my life and my dreams.

Contents

List of Figures

Chapter 1
Introduction

With the advance of digital logic design, the last decade has observed wide application and deployment of digital communication and error protection techniques. These systems have enabled, and induced explosive demands for, high-quality and high-speed voice-band modems, digital subscriber loops, personal wireless communications, mobile and direct-broadcast satellite communications. To achieve efficient use of bandwidth and power and, at the same time, combat against adverse channel conditions, new engineering challenges have arisen. For example, the systems should have low physical and computational complexity to increase portability and reachability, allow seamless data rate changes to cope with time-varying channel conditions and higher level network protocols, and provide unequal error protection to accommodate different service rates and to differentiate bits of nonuniform importance from advanced source encoders. In this thesis, new high-performance error correcting techniques with low system complexity are developed to address these new challenges.

1.1 Error Correcting Codes for Digital Communications

The basic structure of a digital communication system [63,70] is illustrated in Fig. 1.1. The function of the modulator is to convert a digital sequence into signals that are compatible with the characteristics of the channel. The channel, however, is unreliable and, hence, the demodulator is usually unable to reproduce the input to the modulator exactly. One way to increase the reliability of the system is to introduce some structure into the information sequence by adding redundant bits [46, 51, 52]. This process (encoding) is furnished by the encoder and the new digital sequence is termed a codeword. The collection of all the digital sequences is called an error correcting code. Because of the structure introduced, the decoder is able to infer the original information from the demodulator output with high probability, despite the corruption from the channel. In fact, it was shown by Shannon that, by operating on very long sequences of data with rates below the "capacity" of the channel, which is defined as the theoretical limit performance limit, the encoder and decoder are able to achieve error-free communication [23, 52, 68, 75].

These ideas are best illustrated by an example: the classic (7,4) Hamming code. The encoder takes four bits of input, (u_1, u_2, u_3, u_4), and outputs three redundant parity bits (p_1, p_2, p_3) according to:

$$p_1 = u_1 \oplus u_2 \oplus u_4,$$

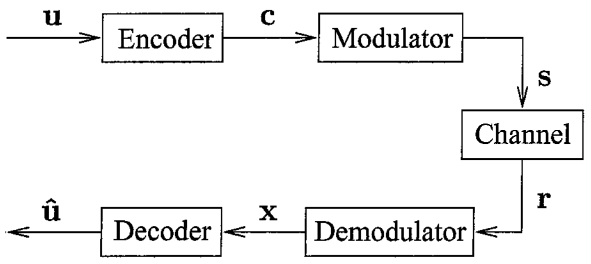


Figure 1.1 Digital communication system.

$$p_2 = u_1 \oplus u_3 \oplus u_4,$$

$$p_3 = u_2 \oplus u_3 \oplus u_4.$$

The structure thus imposed on the codeword can be illustrated as the diagram in Fig. 1.2: a valid codeword will have even number of ones (even parity) in each circle. As claimed before, this will enable us to correct some errors that occurred during the transmission of these bits. For example, suppose there is only one bit whose value was changed because of the noisy channel. There are three cases to be considered:

1. If only one circle has odd parity, then it is the parity bit of the circle that was changed. The information bits are intact.

2. If two circles have odd parity, then it is the information bit of the intersection of the two circles that was damaged. The error can corrected by flipping the value of that information bit.

3. If none of the circles has even parity, then it is the value of u_4 that was changed. The correct information bits can be obtained by flipping the value of this bit.

Therefore, we can correct transmission errors so long as there is only one error bit within a block of seven bits (a codeword). The above procedure (decoding algorithm), however, does not produce correct answers if there are more than one error within a codeword and thus results in decoding errors. The purpose of this thesis is to construct

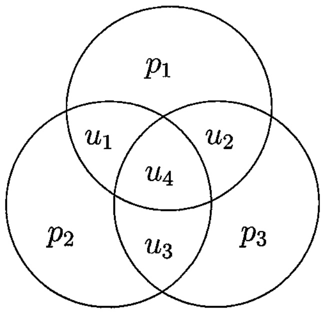


Figure 1.2 The structure of the (7,4) Hamming Code.

codes that are able to correct large number of errors with low error probabilities and require low complexity in the decoding algorithms.

1.2 Thesis Outline

Though coding theory suggests that error correcting codes chosen *at random* perform close to the optimum if their block lengths are large enough [52,68,75], the problem of designing good codes has traditionally been attacked by developing codes with *a lot of structure*, which lends itself to feasible decoders [51,52]. The challenge to find practical decoders for long random codes was not seriously considered until the recent introduction of turbo codes in 1993 [7]. This methodology of multi-stage iterative decoding with exchange of soft information, applied to codes with *pseudo-random structure*, provides a whole new approach to construct good codes and to decode them with low complexity. The thesis will first apply this decoding method to a wide range of codes and will clarify some ambiguities and problems in the decoding algorithms that have appeared in the literature. The connection between these decoding algorithms and a statistical inference framework [58] from the artificial intelligence community will then be made so that, when any of the codes is represented by a "belief network," the application of this framework leads directly to the corresponding decoding algorithm without any *ad hoc* arguments used in the literature. A class of generalized turbo codes based on low-density generator matrices are then proposed and shown to achieve near-capacity performance with this belief propagation algorithm. These low-density generator matrix (LDGM) codes can be applied to binary or M-ary signaling schemes with performance approaching the channel capacity. The content of subsequent chapters is briefed as follows:

✎ **Unit Memory Hamming Turbo Codes (Chapter 2)**

The coding scheme of parallel concatenation of two simple recursive systematic convolutional codes (turbo codes) was recently proposed to achieve near Shannon-limit error correction performance with reasonable decoding complexity [7]. The underlying component code adopted in these results is a single-input convolutional code, or, more specifically, a $(2,1,4,7)$ code which is of memory $\nu = 4$ and free distance $d_f = 7$. On the other hand, in many cases of interest, multi-input unit-memory codes have been demonstrated to have larger free distances than the single-input codes with the same rate and the same number of memory elements [1]. In this chapter, new turbo codes based on the $(8,4,3,8)$ unit-memory Hamming code are proposed and BCJR's optimal bit decision algorithm [2] is modified for this multiple input recursive convolutional code. Our results show that the new codes achieve marginally better performance than conventional turbo codes. Better performance can be obtained by using two different component codes in the design of turbo codes.

Generalized Concatenated Convolutional Codes (Chapter 3)

Since the decoders of turbo codes usually require maximum *a posteriori* (MAP) algorithm [2], which is of relatively high complexity if implemented straightforwardly. Modified MAP algorithms [3,60,65] and soft-output Viterbi algorithm (SOVA) [38,40] have thus been proposed in place of MAP decoders to reduce system complexity. Alternative high-performance coding systems of low complexity are proposed in this chapter via the generalized concatenation of convolutional codes. Two classes of generalized concatenated (GC) codes with convolutional outer codes are studied. The first class is based on the classical Plotkin $|a \oplus b|b|$ construction. A new suboptimal multi-stage soft decision algorithm is proposed and the corresponding performance bounds are obtained. These codes are shown to achieve better performance than conventional convolutional codes with equal or less decoding complexity, and are capable of unequal error protection. The Plotkin construction is then generalized using an inner differential encoding structure to obtain a second class of GC codes. A low-complexity two-iteration decoding algorithm using traditional hard-output Viterbi decoders is proposed. Numerical results show that the new coding systems can achieve comparable and sometimes superior performance to low-complexity turbo codes with similar computational complexity.

✎ **Turbo Decoding and Belief Propagation (Chapter 4)**

Though the turbo decoding method has been applied to a wide range of codes with remarkable success, a satisfactory theoretical explanation as to why the turbo decoding algorithm performs so well has been lacking. In addition, some pathological cases where the algorithm never converges or converges to the wrong answer have been identified [53]. In this chapter, we establish a close connection between the turbo decoding mechanism and the "belief propagation" algorithm [58], well-known in the artificial intelligence community. We show that, by representing a turbo code with a "belief network", the application of the belief propagation algorithm directly leads to the turbo decoding algorithm. This framework also prescribes explicitly how the decoders for multiple turbo codes should proceed without any *ad hoc* arguments. In fact, it is also argued that the decoding algorithms for a wide range of turbo code constructions as well as low-density parity check codes [33] can be explained by this Bayesian network and belief propagation principle.

✎ **Low-Density Generator Matrix Codes (Chapter 5)**

It is the purpose of this chapter to further examine the applicability of the belief propagation algorithm to coding theory. It is observed that the turbo codes separate parity bits into subsets in their Bayesian network representation, which is unnecessary for the belief propagation algorithm. A class of codes based on low-density generator matrix, where the parity bits are not differentiated, are proposed as a generalization of classical turbo codes. Contrary to the turbo decoding paradigms where sequential processing is accomplished by very powerful central units, the decoding algorithm proposed here is formulated in a distributed parallel form. The decoders can thus enjoy modular pipeline design and the systems therefore seem more suitable for practical applications. For high-rate applications, numerical results show that these codes achieve performance within 1 dB of the channel capacity.

✎ **Efficient Multilevel Coded Modulations (Chapter 6)**

In this chapter, the low-density generator matrix codes are applied to M-ary signaling schemes by multilevel coding methods [12, 42, 62, 66, 72]. The equivalent channel capacities [41] for individual partition levels used in the multilevel coding systems are first computed. Partition rules other than Ungerboeck's maximum intra-set distance criterion are examined. LDGM codes for individual levels are then selected according to the corresponding equivalent capacities. We show this approach can be used to devise systems that achieve near channel capacity performance and are also able to provide unequal error protection.

Chapter 2
Unit Memory Hamming Turbo Codes

The coding scheme of parallel concatenation of two simple recursive systematic convolutional codes—the turbo code—was recently proposed to achieve near Shannon-limit error correction performance with reasonable decoding complexity [7,24,25,64]. The underlying component code used in these constructions is a single-input (SI) convolutional code or, more specifically, a $(2,1,4,7)$ code, which is of rate $1/2$, memory $\nu = 4$, and free distance $d_f = 7$. On the other hand, in many cases of interest, multi-input unit-memory (UM) codes have been demonstrated to have larger free distances than SI codes with the same rate and the same number of memory elements [1,45]. In this chapter, new turbo codes based on the $(8,4,3,8)$ UM Hamming code are developed.

2.1 Encoder

The generator matrix of the $(8, 4, 3, 8)$ unit-memory Hamming code is [1]

$$G'' = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1+D & 1+D & 1 & D & 1+D & 0 & 0 & 0 \\ 1+D & 0 & 1+D & 1 & D & 1+D & 0 & 0 \\ 1+D & 0 & 0 & 1+D & 1 & D & 1+D & 0 \end{array}\right] = [A|B].$$

To convert this into a systematic recursive code with a generator matrix of the form $G = [I|A^{-1}B]$, one needs to find a column permutation such that (1) the matrix A is nonsingular and (2) the matrix $A^{-1}B$ is in its simplest form. There are $\binom{8}{4} = 70$ permutations to be considered, since those that group columns into the same submatrices are all equivalent for our purpose. The following matrix was found to fulfill the requirements through computer search:

$$G' = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1+D & 1+D & 0 & 1 & D & 0 & 1+D \\ 0 & 1+D & 0 & 0 & 1+D & 1 & 1+D & D \\ 0 & 1+D & 0 & 1+D & 0 & 1+D & D & 1 \end{array}\right] = [A'|B'].$$

Therefore, the generator matrix of the equivalent systematic recursive code is

$$G = [I|A'^{-1}B'] = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & \frac{1}{1+D} & 1 & \frac{D}{1+D} & 1 \\ 0 & 1 & 0 & 0 & 1 & \frac{1}{1+D} & 1 & \frac{D}{1+D} \\ 0 & 0 & 1 & 0 & \frac{D}{1+D} & 1 & 1 & \frac{1}{1+D} \\ 0 & 0 & 0 & 1 & 1 & \frac{D}{1+D} & \frac{1}{1+D} & 1 \end{array}\right].$$

Because the McMillan degree of the generator is three [55, 73], it can be implemented with three memory elements as shown in Fig. 2.1. If we use the average edge complexity of the trellis per information bit $2^{k+\nu}/k$ as the decoding complexity $\mathcal{D}$ of a convolutional code [54], both the SI $(2, 1, 4, 7)$ code and the UM $(8, 4, 3, 8)$ code have the same complexity, $\mathcal{D} = 32$, and hence are fair competitors.

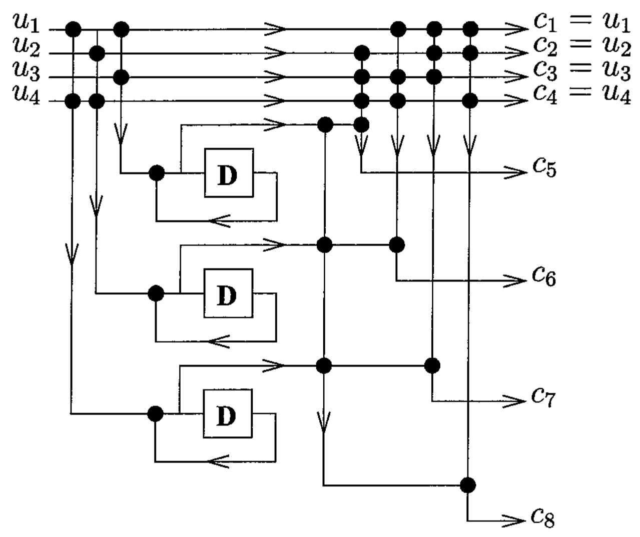
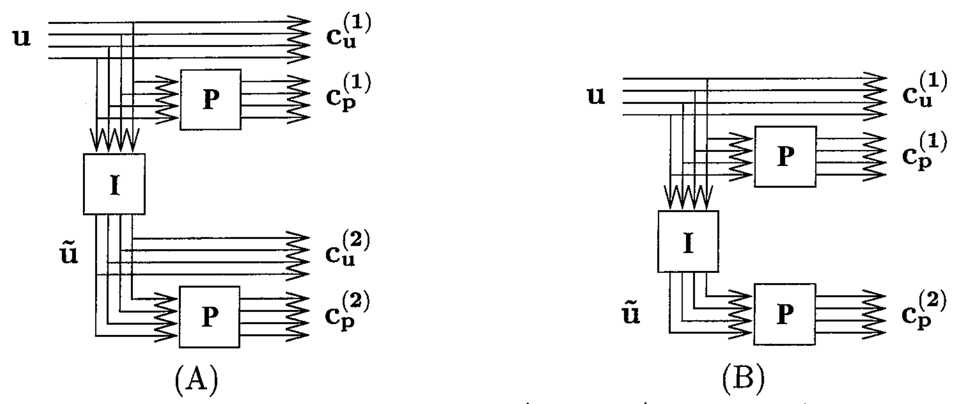


Figure 2.1 Encoder of the recursive partial unit-memory Hamming code (additions are performed in the vertical direction only.)

The encoders of parallel concatenated codes are illustrated in Fig. 2.2, where P represents the shift-register circuits for implementing the parity-check part $A'^{-1}B'$ of the generator P and I is a pseudo-random bit interleaver. The first encoder operates on the input information bits directly and outputs $\mathbf{c}_\mathbf{u}^{(1)} = \mathbf{u}$ and $\mathbf{c}_\mathbf{p}^{(1)}$. The second encoder operates on the interleaved information sequence $\tilde{\mathbf{u}}$ and outputs $\mathbf{c}_\mathbf{u}^{(2)} = \tilde{\mathbf{u}}$ and $\mathbf{c}_\mathbf{p}^{(2)}$. After encoding a block of K information bits, the trellises of both codes can terminated by selecting two four-bit sequences for each of the encoders. These bits can be generated by the circuit in Fig. 2.3. This results in a $(4(K+4), K)$ block



u
$\mathbf{c}_\mathbf{u}^{(1)}$
P
$\mathbf{c}_\mathbf{p}^{(1)}$
I
ũ
$\mathbf{c}_\mathbf{u}^{(2)}$
P
$\mathbf{c}_\mathbf{p}^{(2)}$
(A)
u
$\mathbf{c}_\mathbf{u}^{(1)}$
P
$\mathbf{c}_\mathbf{p}^{(1)}$
I
ũ
P
$\mathbf{c}_\mathbf{p}^{(2)}$
(B)

Figure 2.2 The encoders of rate 1/4 and 1/3 turbo codes.

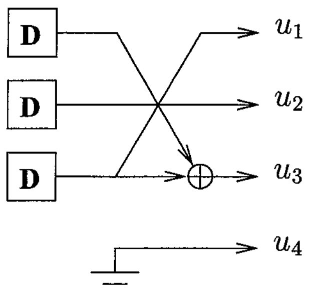


Figure 2.3 Trellis termination circuit.

code. The interleaved information sequence $\tilde{\mathbf{u}}$ can be discarded to increase spectral efficiency. This will give us a $(3(K+4)+4, K)$ block code. Note that the 4 terminating bits for the second encoder should be kept and transmitted.

Since the minimum distance of the component UM code is larger than that of the SI code, the new unit-memory turbo (UMT) code is expected to have better performance than the SI turbo (SIT) codes. The estimate of the performance of a code requires information about its weight distribution. However, obtaining this weight distribution is a particularly challenging problem for turbo codes because of the pseudo-random interleaver. Though some bit error rate (BER) bounding techniques have been developed [26], they are not useful here. As to be shown later, the operating

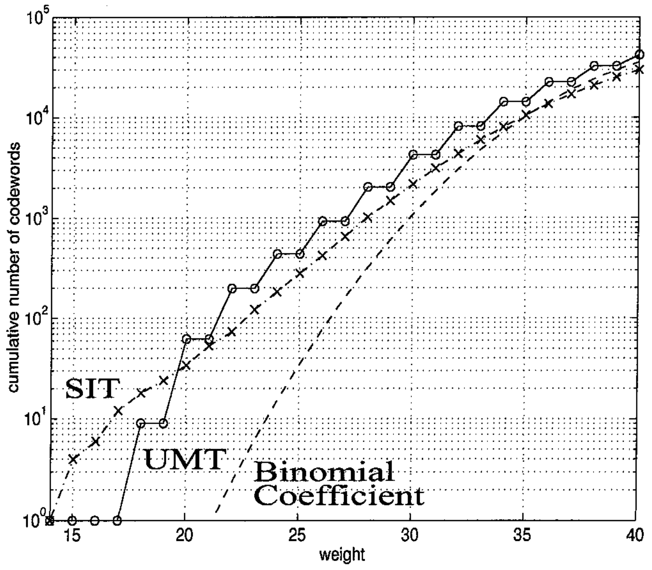


Figure 2.4 The weight distributions of $(80, 16)$ SIT and UMT codes.

E_b/N_0 range of interest is below 2 dB, but the bounds diverge above that. Before plunging into full-scale simulations, short-block ($K = 16$) cases of both SIT and UMT codes are compared. While the best found interleaver for the $(80, 16)$ SIT code gives a minimum distance of 15 [24], the new $(80, 16)$ UMT code enjoys a larger minimum distance of 18, achieved using the interleaver {13, 5, 10, 1, 11, 14, 0, 12, 6, 4, 15, 7, 3, 9, 2, 8}. The weight distributions of both codes and a "random code," which is computed from the binomial coefficients $A_w = 2^{k-n}\binom{n}{w}$, are plotted in Fig. 2.4.

2.2 Decoding Algorithm

In this section, the classical maximum a posteriori (MAP) algorithm for computing the *a posteriori* probabilities [2] will be modified to deal with multiple input recursive trellis codes. Turbo decoding based on this algorithm will then be described.

2.2.1 MAP Algorithm for Multi-Input Recursive Trellis Codes

Let the state of the encoder for the (n, k, ν) code at time t be $S_t \in \{0, 1, \dots, 2^\nu - 1\}$ for $t = 0, \dots, K$, where the initial and final states, S_0 and S_K, are known. As shown in Fig. 2.5, the input symbol $\mathbf{u}_t = (u_{t,1}, \dots, u_{t,k})$ causes a transition from S_{t-1} to S_t, and the corresponding output codeword $\mathbf{c}_t = (c_{t,1}, \dots, c_{t,n})$ is observed over an AWGN channel as $\mathbf{y}_t = (y_{t,1}, \dots, y_{t,n})$, for $t = 1, \dots, K$. Note that $\mathbf{c}_t = (u_{t,1}, \dots, u_{t,k}, c_{t,k+1}, \dots, c_{t,n})$ since the code is systematic. The log likelihood ratios of the *a posteriori* probabilities, given all the received signals $\mathbf{y}_1^K \triangleq (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K)$, are computed as

$$\Lambda(u_{t,j}) \triangleq \log \frac{\Pr\{u_{t,j} = +1 \,|\, \mathbf{y}_1^K\}}{\Pr\{u_{t,j} = -1 \,|\, \mathbf{y}_1^K\}} = \log \frac{\sum_s \Pr\{S_t = s, u_{t,j} = +1 \,|\, \mathbf{y}_1^K\}}{\sum_s \Pr\{S_t = s, u_{t,j} = -1 \,|\, \mathbf{y}_1^K\}}, \tag{2.1}$$

for $t = 1, \dots, K$, and $j = 1, \dots, k$. The summations are over all the possible states at time t. In order to compute (2.1) recursively, the following quantities are introduced:

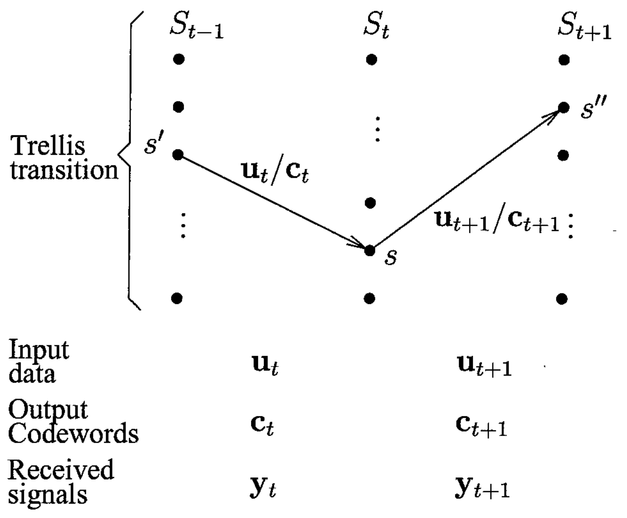


Figure 2.5 The variables involved in the MAP algorithm.

- The probability of $S_t = s$ given the past signals $\mathbf{y}_1^t \triangleq (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t)$:

$$\alpha_t(s) \triangleq \Pr\left\{S_t = s \,|\, \mathbf{y}_1^t\right\}.$$

- The normalized probability of the future signals $\mathbf{y}_{t+1}^K \triangleq (\mathbf{y}_{t+1}, \mathbf{y}_{t+2}, \ldots, \mathbf{y}_K)$ given $S_t = s$:

$$\beta_t(s) \triangleq \frac{\Pr\left\{\mathbf{y}_{t+1}^K \,|\, S_t = s, \mathbf{y}_1^t\right\}}{\Pr\left\{\mathbf{y}_{t+1}^K \,|\, \mathbf{y}_1^t\right\}} = \frac{\Pr\left\{\mathbf{y}_{t+1}^K \,|\, S_t = s\right\}}{\Pr\left\{\mathbf{y}_{t+1}^K \,|\, \mathbf{y}_1^t\right\}},$$

which follows from the Markov property of the trellis.

- The branch transition probability from $S_{t-1} = s'$ to $S_t = s$ with the present signal $\mathbf{y}_t$:

$$\Gamma_t(s', s) \triangleq \Pr\left\{S_t = s, \mathbf{y}_t \,|\, S_{t-1} = s'\right\}.$$

- The joint probability of the transition from s' to s and a the jth input of i:

$$\gamma_{t,j}^i(s', s) \triangleq \Pr\left\{S_t = s, u_{t,j} = i, \mathbf{y}_t \,|\, S_{t-1} = s'\right\}.$$

Consider the terms in the summands of (2.1):

$$\begin{aligned}
&\Pr\left\{S_t = s, u_{t,j} = i \,|\, \mathbf{y}_1^K\right\} \\
&\quad= \frac{\Pr\left\{S_t = s, u_{t,j} = i, \mathbf{y}_1^t, \mathbf{y}_{t+1}^K\right\}}{\Pr\left\{\mathbf{y}_1^t, \mathbf{y}_{t+1}^K\right\}} \\
&\quad= \frac{\Pr\left\{S_t = s, u_{t,j} = i, \mathbf{y}_1^t\right\}}{\Pr\left\{\mathbf{y}_1^t\right\}} \frac{\Pr\left\{\mathbf{y}_{t+1}^K \,|\, S_t = s, u_{t,j} = i, \mathbf{y}_1^t\right\}}{\Pr\left\{\mathbf{y}_{t+1}^K \,|\, \mathbf{y}_1^t\right\}} \\
&\quad= \frac{\sum_{s'} \Pr\left\{S_t = s, u_{t,j} = i, \mathbf{y}_t \,|\, S_{t-1} = s', \mathbf{y}_1^{t-1}\right\} \Pr\left\{S_{t-1} = s', \mathbf{y}_1^{t-1}\right\}}{\Pr\left\{\mathbf{y}_t \,|\, \mathbf{y}_1^{t-1}\right\} \Pr\left\{\mathbf{y}_1^{t-1}\right\}} \beta_t(s) \\
&\quad= \frac{\sum_{s'} \gamma_{t,j}^i(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)}{\Pr\left\{\mathbf{y}_t \,|\, \mathbf{y}_1^{t-1}\right\}},
\end{aligned}$$

where

$$\Pr\left\{\mathbf{y}_{t+1}^K \,|\, S_t = s, u_{t,j} = i, \mathbf{y}_1^t\right\} = \Pr\left\{\mathbf{y}_{t+1}^K \,|\, S_t = s\right\}$$

$$\Pr\left\{S_t = s, u_{t,j} = i, \mathbf{y}_t \,|\, S_{t-1} = s', \mathbf{y}_1^{t-1}\right\} = \Pr\left\{S_t = s, u_{t,j} = i, \mathbf{y}_t \,|\, S_{t-1} = s'\right\}$$

because of the Markov property. Therefore,

$$\Lambda(u_{t,j}) = \log \frac{\sum_s \sum_{s'} \gamma_{t,j}^{+1}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)}{\sum_s \sum_{s'} \gamma_{t,j}^{-1}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)}. \tag{2.2}$$

Similarly,

$$\begin{aligned}
\alpha_t(s) &= \frac{\Pr\left\{S_t = s, \mathbf{y}_t \,|\, \mathbf{y}_1^{t-1}\right\}}{\Pr\left\{\mathbf{y}_t \,|\, \mathbf{y}_1^{t-1}\right\}} \\
&= \frac{\sum_{s'} \Pr\left\{S_t = s, \mathbf{y}_t, S_{t-1} = s' \,|\, \mathbf{y}_1^{t-1}\right\}}{\Pr\left\{\mathbf{y}_t \,|\, \mathbf{y}_1^{t-1}\right\}} \\
&= \frac{\sum_{s'} \Pr\left\{S_t = s, \mathbf{y}_t \,|\, S_{t-1} = s', \mathbf{y}_1^{t-1}\right\} \Pr\left\{S_{t-1} = s' \,|\, \mathbf{y}_1^{t-1}\right\}}{\Pr\left\{\mathbf{y}_t \,|\, \mathbf{y}_1^{t-1}\right\}} \\
&= \frac{\sum_{s'} \Gamma_t(s', s)\, \alpha_{t-1}(s')}{\Pr\left\{\mathbf{y}_t \,|\, \mathbf{y}_1^{t-1}\right\}},
\end{aligned}$$

and

$$\beta_t(s) = \frac{\sum_{s''} \Pr\left\{S_{t+1} = s'', \mathbf{y}_{t+1}^K \,|\, S_t = s\right\}}{\Pr\left\{\mathbf{y}_{t+2}^K \,|\, \mathbf{y}_1^{t+1}\right\} \Pr\left\{\mathbf{y}_{t+1} \,|\, \mathbf{y}_1^t\right\}}$$

$$= \frac{\sum_{s''} \Pr\left\{S_{t+1} = s'', \mathbf{y}_{t+1} \,|\, S_t = s\right\} \Pr\left\{\mathbf{y}_{t+2}^K \,|\, S_{t+1} = s'', S_t = s, \mathbf{y}_{t+1}\right\}}{\Pr\left\{\mathbf{y}_{t+2}^K \,|\, \mathbf{y}_1^{t+1}\right\} \Pr\left\{\mathbf{y}_{t+1} \,|\, \mathbf{y}_1^t\right\}}$$

$$= \frac{\sum_{s''} \Gamma_{t+1}(s, s'')\, \beta_{t+1}(s'')}{\Pr\left\{\mathbf{y}_{t+1} \,|\, \mathbf{y}_1^t\right\}}.$$

Since

$$\begin{aligned}
\Pr\left\{\mathbf{y}_t \,|\, \mathbf{y}_1^{t-1}\right\} &= \sum_s \sum_{s'} \Pr\left\{S_t = s, \mathbf{y}_t, S_{t-1} = s' \,|\, \mathbf{y}_1^{t-1}\right\} \\
&= \sum_s \sum_{s'} \Pr\left\{S_t = s, \mathbf{y}_t \,|\, S_{t-1} = s', \mathbf{y}_1^{t-1}\right\} \Pr\left\{S_{t-1} = s' \,|\, \mathbf{y}_1^{t-1}\right\} \\
&= \sum_s \sum_{s'} \Gamma_t(s', s)\, \alpha_{t-1}(s'),
\end{aligned}$$

we have

$$\alpha_t(s) = \frac{\sum_{s'} \Gamma_t(s', s)\, \alpha_{t-1}(s')}{\sum_s \sum_{s'} \Gamma_t(s', s)\, \alpha_{t-1}(s')}, \tag{2.3}$$

$$\beta_t(s) = \frac{\sum_{s''} \Gamma_{t+1}(s, s'')\, \beta_{t+1}(s'')}{\sum_{s''} \sum_s \Gamma_{t+1}(s, s'')\, \alpha_t(s)}. \tag{2.4}$$

If the trellis is terminated, then these two recursions are initialized as:

$$\alpha_0(0) = 1 \quad \text{and} \quad \alpha_0(s \neq 0) = 0,$$

$$\beta_N(0) = 1 \quad \text{and} \quad \beta_N(s \neq 0) = 0.$$

The branch transition probabilities are given by

$$\begin{aligned}
\gamma_{t,j}^i(s', s) &= \Pr\left\{u_{t,j} = i \,|\, S_{t-1} = s'\right\} \Pr\left\{\mathbf{y}_t \,|\, S_t = s, u_{t,j} = i, S_{t-1} = s'\right\} \\
&\quad \cdot \Pr\left\{S_t = s \,|\, u_{t,j} = i, S_{t-1} = s'\right\} \\
&= \begin{cases} \Pr\left\{u_{t,j} = i\right\} \Pr\left\{\mathbf{y}_t \,|\, S_t = s, u_{t,j} = i, S_{t-1} = s'\right\}, & \\ & \text{if } s' \to s \text{ is allowed by } u_{t,j} = i \\ 0, & \text{otherwise} \end{cases}
\end{aligned}$$

and similarly,

$$\Gamma_t(s', s) = \sum_{\mathbf{i}\,:\,s' \to s} \Pr\{\mathbf{u}_t = \mathbf{i}\} \Pr\{\mathbf{y}_t \,|\, S_t = s, \mathbf{u}_t = \mathbf{i}, S_{t-1} = s'\},$$

where the summation is over all possible inputs $\mathbf{i}$ that allow the transition from s' to s. If the *a priori* likelihood ratios $\mathrm{V}(u_{t,j}) \triangleq \log \frac{\Pr\{u_{t,j}=+1\}}{\Pr\{u_{t,j}=-1\}}$ are provided, then

$$\begin{aligned}
\Pr\{u_{t,j} = +1\} &= \frac{e^{\mathrm{V}(u_{t,j})}}{1 + e^{\mathrm{V}(u_{t,j})}}, \\
\Pr\{u_{t,j} = -1\} &= 1 - \frac{e^{\mathrm{V}(u_{t,j})}}{1 + e^{\mathrm{V}(u_{t,j})}},
\end{aligned}$$

and $\Pr\{\mathbf{u}_t = \mathbf{i}\} = \prod_j \Pr\{u_{t,j} = i_j\}$.

2.2.2 Extrinsic Information

Let $q(y|c)$ be the channel transition function for receiving y if c is transmitted, and $\mathbf{c}$ be the codeword specified by $\{S_t = s, \mathbf{u}_t, S_{t-1} = s'\}$. Then

$$\begin{aligned}
&\Pr\{\mathbf{y}_t \,|\, S_t = s, u_{t,j} = i, S_{t-1} = s'\} \\
&= \sum_{\substack{u_{t,m} \\ m \neq j}} \Pr\{\mathbf{y}_t \,|\, S_t = s, u_{t,j} = i, \{u_{t,m}\}_{m \neq j}, S_{t-1} = s'\} \prod_{m \neq j} \Pr\{u_{t,m}\} \\
&= \sum_{\substack{\mathbf{c} \\ u_{t,j}=i}} q(\mathbf{y}_t \,|\, \mathbf{c}) \prod_{m \neq j} \Pr\{u_{t,m}\} \\
&= \sum_{\substack{\mathbf{c} \\ u_{t,j}=i}} \prod_{l=1}^{n} q(y_{t,l} \,|\, c_l) \prod_{m \neq j} \Pr\{u_{t,m}\} \\
&= q(y_{t,j} \,|\, i) \left[\sum_{\substack{\mathbf{c} \\ u_{t,j}=i}} \prod_{l \neq j} q(y_{t,l} \,|\, c_l) \prod_{m \neq j} \Pr\{u_{t,m}\} \right] \\
&= q(y_{t,j} \,|\, i)\, \delta_{t,j}^{i}(s', s),
\end{aligned}$$

where the last equation follows because the code is systematic. Therefore, if the transition from state s' into state s is allowed by $u_{t,j} = i$, we have

$$\gamma^i_{t,j}(s', s) = \Pr\{u_{t,j} = i\}\, q(y_{t,j}\,|\,i)\, \delta^i_{t,j}(s', s).$$

Substituting this into (2.2), we can then decompose the *a posteriori* log likelihood ratios into three terms:

$$\begin{aligned}
\Lambda(u_{t,j}) &= \log \frac{\sum_s \sum_{s'} \gamma^{+1}_{t,j}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)}{\sum_s \sum_{s'} \gamma^{-1}_{t,j}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)} \\
&= \log \frac{\Pr\{u_{t,j} = +1\}}{\Pr\{u_{t,j} = -1\}} + \log \frac{q(y_{t,j}\,|+1)}{q(y_{t,j}\,|-1)} + \log \frac{\sum_s \sum_{s'} \delta^{+1}_{t,j}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)}{\sum_s \sum_{s'} \delta^{-1}_{t,j}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)} \\
&= \mathrm{V}(u_{t,j}) + \log \frac{q(y_{t,j}\,|+1)}{q(y_{t,j}\,|-1)} + \mathrm{W}(u_{t,j}).
\end{aligned}$$

For an additive white Gaussian noise channel, $q(y|c) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y-c)^2\right]$, where $\sigma^2 = N_0/2E_s = N_0/2RE_b$. E_s is the average energy per symbol, E_b is the average energy per bit, R is the overall code rate, and N_0 is the single-sided noise power spectral density. The above equation can then be reduced to

$$\Lambda(u_{t,j}) = \mathrm{V}(u_{t,j}) + \frac{2}{\sigma^2} y_{t,j} + \mathrm{W}(u_{t,j}). \tag{2.5}$$

There three terms are the *a priori* information, systematic information, and extrinsic information, respectively.

2.2.3 Turbo Decoding

The presentation in this section follows historical accounts. A more solid theoretical ground is given in Chapter 4. Given the received signals $\mathbf{y}^{(1)} = (\mathbf{y}^{(1)}_{\mathbf{u}}, \mathbf{y}^{(1)}_{\mathbf{p}})$ and $\mathbf{y}^{(2)} = (\mathbf{y}^{(2)}_{\mathbf{u}}, \mathbf{y}^{(2)}_{\mathbf{p}})$, the optimal bit decision is based on $\log \frac{\Pr\{u_{t,j}=+1\,|\,\mathbf{y}^{(1)},\mathbf{y}^{(2)}\}}{\Pr\{u_{t,j}=-1\,|\,\mathbf{y}^{(1)},\mathbf{y}^{(2)}\}}$ and the optimal sequence decision is based on $\log \frac{\Pr\{\mathbf{u}\,|\,\mathbf{y}^{(1)},\mathbf{y}^{(2)}\}}{\Pr\{\mathbf{u}\,|\,\mathbf{y}^{(1)},\mathbf{y}^{(2)}\}}$. Both are, however, difficult to compute because of the presence of the pseudo-random interleaver, which is usually of large size. The wisdom of turbo decoding is to compute these optimal decision

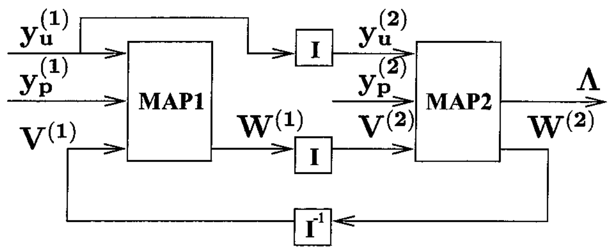


Figure 2.6 Turbo decoder for $(3(K+4)+4, K)$ codes.

variables separately for each of the component codes, which is feasible using a Viterbi or a MAP decoder, and then combine them "intelligently" to obtain good suboptimal decisions. The innovation in turbo decoding is the recognition that the extrinsic information $\mathrm{W}(u_{t,j})$ produced by the current MAP decoder is rid of the *a priori* and systematic information and thus is suitable to be used as an independent estimate by the next MAP decoder [7,64].

For example, the decoder structure for rate 1/3 codes, generated by the encoder shown in Fig. 2.1 (b), is illustrated in Fig. 2.6. The parity parts of the received signals for the two component codes are used by the corresponding MAP decoders. The systematic part is used by the first decoder and its interleaved version is used by the second. The decoding process begins by setting the *a priori* input to MAP1 decoder to zero. The extrinsic part computed by MAP1 is passed through an interleaver to MAP2 decoder as its *a priori* input. The likelihood output from this decoder is taken as the decision variables for this iteration. The process is repeated by setting the interleaved version of the extrinsic output from MAP2 decoder as the *a priori* input to the MAP1 decoder. As will be seen in the next section, performance can be improved by executing more iterations but with diminishing gains.

2.3 Simulation Results

In this section, the performance of rate $K/(3(K+4)+4)$ codes, obtained by discarding the systematic bits from the second branch but retaining the termination bits as shown in Fig. 2.1 (b), are studied by computer simulations. For $K = 512$, the bit error probabilities for different numbers of iterations are shown in Fig. 2.7. Performance improvements from the first few iterations are quite substantial but saturate after about 20 iterations. For BER $= 10^{-5}$, the coding gain over a rate 1/3, constraint length $\nu = 6$ convolutional code [80] is about 3 dB. The performance is also better than the complicated rate 1/4, constraint length $\nu = 14$ convolutional code used by JPL for the Galileo mission [24]. Performance for larger block lengths and comparison to those for the conventional single input turbo (SIT) codes are shown in Fig. 2.8. For $K = 16384$, the performance of the code is within only 0.7 dB of the Shannon capacity for the corresponding rate [10, 52]. It is also observed that the proposed UMT codes achieve marginally better performance than SIT codes. The performance is improved further by an unbalanced constructions where one UM code and one SI are used as the two component codes. The performance of these unbalanced codes are also shown in Fig. 2.7, where we observe only marginal improvements which seem to diminish for very low BER.

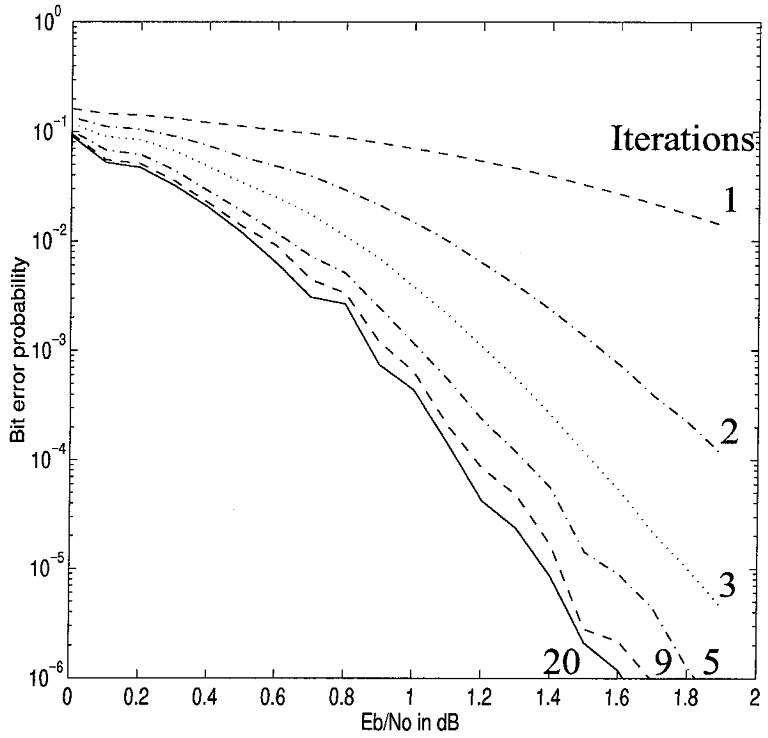


Figure 2.7 Performance of the (1552, 512) unit-memory turbo code.

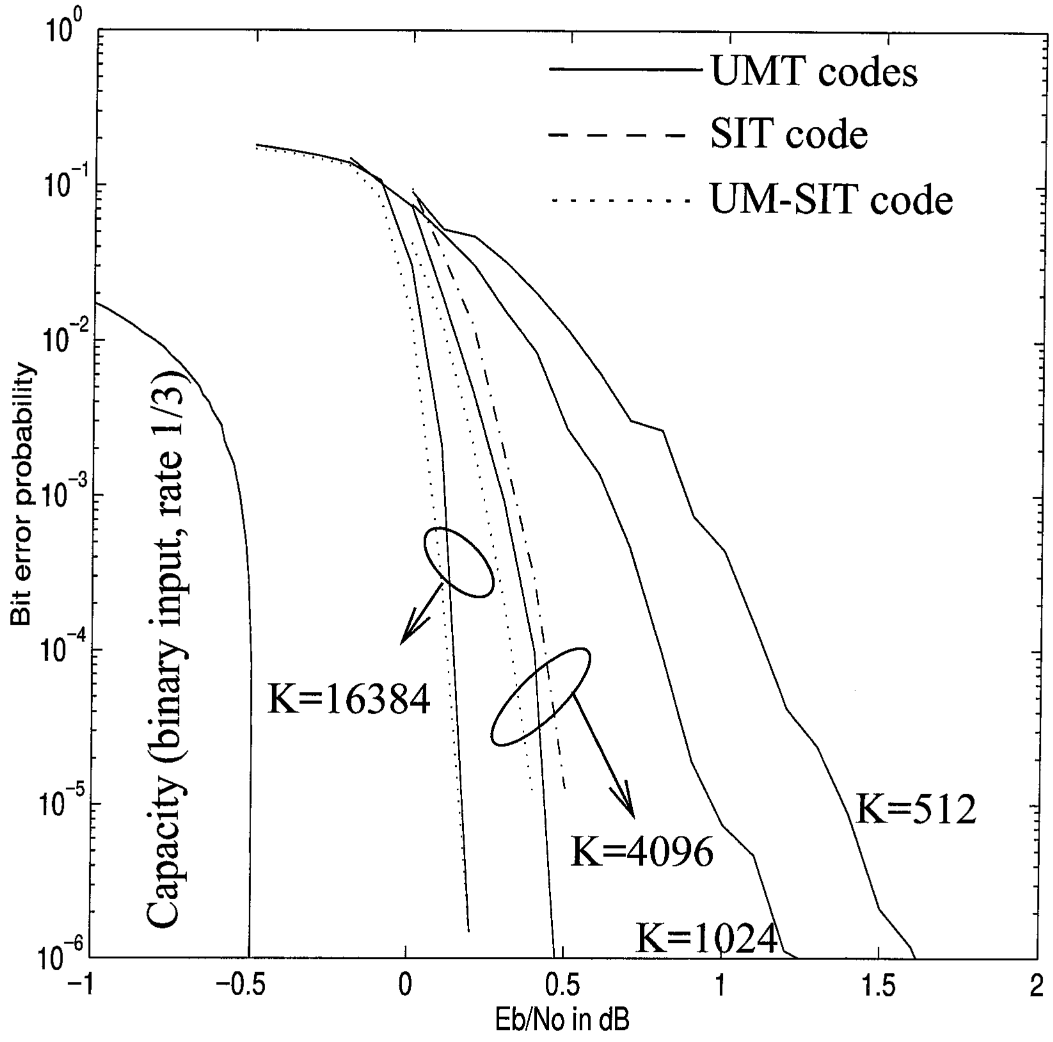


Figure 2.8 Performance of several turbo codes, code rates $\approx 1/3$.

2.4 Conclusions

We have presented an extensive introduction to the methodology of turbo coding by constructing new turbo codes and a new iterative decoding algorithm for multiple input trellis codes. By iterative decoding with exchange of soft information between the decoders for two simple component codes, these codes achieve near-capacity performance. As will be presented in the next chapter, substantial coding gains are also attainable even if the information exchanged between the decoders is not so "soft." Empirically, the methodology provides ways to achieve low bit error probabilities at low SNR that were not possible in the past using even very complicated codes. However, a theoretical explanation as to the choice of extrinsic information and why the decoding algorithm performs so well are not available. We will return to this problem in Chapter 4 from a more general probabilistic inference framework.

2.A Symbol MAP Algorithm for Multi-Input Recursive Trellis Codes

Though only the bit-by-bit MAP algorithm was used in this chapter, the symbol MAP algorithm is included here for completeness. Let $\mathbf{0}$ be the k-tuple $(-1, -1, \ldots, -1)$. For $\mathbf{i} = 1, 2, \ldots, 2^k - 1$ and $t = 1, 2, \ldots, K$, define

$$\Lambda(\mathbf{u}_t = \mathbf{i}) = \log \frac{\Pr\left\{\mathbf{u}_t = \mathbf{i} \,|\, \mathbf{y}_1^K\right\}}{\Pr\left\{\mathbf{u}_t = \mathbf{0} \,|\, \mathbf{y}_1^K\right\}} = \log \frac{\sum_s \Pr\left\{S_t = s, \mathbf{u}_t = \mathbf{i} \,|\, \mathbf{y}_1^K\right\}}{\sum_s \Pr\left\{S_t = s, \mathbf{u}_t = \mathbf{0} \,|\, \mathbf{y}_1^K\right\}},$$

and

$$\begin{cases} \alpha_t(s) & = & \Pr\left\{S_t = s \,|\, \mathbf{y}_1^t\right\} \\ \beta_t(s) & = & \frac{\Pr\left\{\mathbf{y}_{t+1}^K \,|\, S_t = s\right\}}{\Pr\left\{\mathbf{y}_{t+1}^K \,|\, \mathbf{y}_1^t\right\}} \\ \gamma_t^{\mathbf{i}}(s', s) & = & \Pr\left\{S_t = s, \mathbf{u}_t = \mathbf{i}, \mathbf{y}_t \,|\, S_{t-1} = s'\right\} \\ \Gamma_t(s', s) & = & \Pr\left\{S_t = s, \mathbf{y}_t \,|\, S_{t-1} = s'\right\} = \sum_{\mathbf{i}} \gamma_t^{\mathbf{i}}(s', s) \end{cases}$$

Then the log likelihood ratios can computed iteratively as:

$$\begin{aligned} \Lambda(\mathbf{u}_t = \mathbf{i}) &= \log \frac{\sum_s \sum_{s'} \gamma_t^{\mathbf{i}}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)}{\sum_s \sum_{s'} \gamma_t^{\mathbf{0}}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)}, \\ \alpha_t(s) &= \frac{\sum_{s'} \Gamma_t(s', s)\, \alpha_{t-1}(s')}{\sum_s \sum_{s'} \Gamma_t(s', s)\, \alpha_{t-1}(s')}, \\ \beta_t(s) &= \frac{\sum_{s'} \Gamma_{t+1}(s, s')\, \beta_{t+1}(s')}{\sum_s \sum_{s'} \Gamma_{t+1}(s', s)\, \alpha_t(s')}. \end{aligned}$$

Let the codeword specified by $\{S_t = s, \mathbf{u}_t = \mathbf{i}, S_{t-1} = s'\}$ be $\mathbf{c} = (\mathbf{i}, \mathbf{p})$, where $\mathbf{i}$ are systematic bits and $\mathbf{p}$ are the parity-check bits, and the corresponding received signal $\mathbf{y}_t = (\mathbf{y}_{\mathbf{i}t}, \mathbf{y}_{\mathbf{p}t})$. As shown in the previous subsection, if the transition from state s' into state s is allowed by the input $\mathbf{u}_t = \mathbf{i}$, then $\gamma_t^{\mathbf{i}}(s', s)$ can be expressed as a product of three terms, namely,

$$\gamma_t^{\mathbf{i}}(s', s) = \Pr\{\mathbf{u}_t = \mathbf{i}\}\, q(\mathbf{y}_{\mathbf{i}t} \,|\, \mathbf{i})\, q(\mathbf{y}_{\mathbf{p}t} \,|\, \mathbf{p}) = \Pr\{\mathbf{u}_t = \mathbf{i}\}\, q(\mathbf{y}_{\mathbf{i}t} \,|\, \mathbf{i})\, \delta_t^{\mathbf{i}}(s', s).$$

Therefore,

$$\begin{aligned} \Lambda(\mathbf{u}_t = \mathbf{i}) &= \log \frac{\Pr\{\mathbf{u}_t = \mathbf{i}\}}{\Pr\{\mathbf{u}_t = \mathbf{0}\}} + \log \frac{q(\mathbf{y}_{\mathbf{i}t} \,|\, \mathbf{i})}{q(\mathbf{y}_{\mathbf{i}t} \,|\, \mathbf{0})} + \log \frac{\sum_s \sum_{s'} \delta_t^{\mathbf{i}}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)}{\sum_s \sum_{s'} \delta_t^{\mathbf{0}}(s', s)\, \alpha_{t-1}(s')\, \beta_t(s)} \\ &= \mathrm{V}(\mathbf{u}_t = \mathbf{i}) + \frac{2}{\sigma^2} \sum_{j \le k,\, j:\, i_j = +1} y_{t,j} + \mathrm{W}(\mathbf{u}_t = \mathbf{i}). \end{aligned}$$

Chapter 3 Generalized Concatenated Convolutional Codes

Two classes of generalized concatenated (GC) codes with convolutional outer codes are studied. The first class is based on the classical Plotkin $|a \oplus b|b|$ construction. A new suboptimal multi-stage soft decision algorithm is proposed and the corresponding performance bounds are obtained. These codes are shown to achieve better performance than conventional convolutional codes with equal or less decoding complexity, and are capable of unequal error protection. The Plotkin construction is then generalized using an inner differential encoding structure to obtain a second class of GC codes. A low-complexity two-iteration decoding algorithm using traditional hard-output Viterbi decoders is proposed. Numerical results show that the new coding systems can achieve comparable and sometimes superior performance to low-complexity turbo codes with similar computational complexity.

3.1 Introduction

Let $C = (n, k, \nu, d_f)$ be a k input and n output binary convolutional code with memory ν and free distance d_f. The asymptotic coding gain of the code is characterized by $\frac{k}{n}d_f$, but the maximum likelihood Viterbi decoding complexity for this code, defined as the average number of operations per information bit $\mathcal{D} \triangleq \frac{n}{k}2^{k+\nu}$, grows exponentially with the number of inputs and the size of memory. Though the introduction of (rate-compatible) punctured convolutional codes [11, 37] reduces this complexity to $\frac{n}{k}2^{1+\nu}$, the tradeoff between performance and complexity using convolutional codes still exhibits an exponential relationship. The recent introduction of multi-stage iterative decoding with exchange of soft information [7, 47, 50, 53], applied to codes with pseudo-random structure, has provided a whole new approach to construct good codes and to decode them with low complexity. This methodology has been applied to parallel concatenated convolutional (turbo) codes [7, 13, 25, 64], and serial concatenated convolutional codes [6] with remarkable success. (Similar results are also obtained for product codes [40] and long block codes based on low-density parity-check matrices [49] or generator matrices [16, 18, 19].) These approaches have the effect of reducing the exponential dependence on ν of the complexity at the cost of the linear dependence on the number of decoding iterations. To achieve best performance, however, these decoders usually require maximum *a posteriori* (MAP) algorithm [2], which is of relatively high complexity if implemented straightforwardly. Modified MAP algorithms [3, 60, 65] and soft-output Viterbi algorithm (SOVA) [38, 40] have thus been proposed in place of MAP decoders to reduce system complexity.

Alternative high-performance coding systems of low complexity are proposed in this chapter via the generalized concatenation of convolutional codes. Of central focus is the classical Plotkin $|a \oplus b|b|$ construction [61] and its generalizations to be presented later.

Suppose $\mathbf{a}$ and $\mathbf{b}$ are codewords of two codes, C_1 and C_2, of the same length N, the Plotkin construction leads to a new codeword $(\mathbf{a} \oplus \mathbf{b}, \mathbf{b})$ of length $2N$, i.e., the first half of the new codeword is the superimposing of those of the component codes and second half is the same as that of one of the component codes. This construction can be interpreted as the concatenation of $C_1 \times C_2$ with a block inner code of the generator matrix

$$G'_{\mathbb{P}} = \begin{bmatrix} I_N & \mathbf{0}_{N,N} \\ I_N & I_N \end{bmatrix}, \tag{3.1}$$

where I_N is an $N \times N$ identity matrix, $\mathbf{0}_{N,N}$ is an $N \times N$ matrix with all zero elements, and $\mathbb{P}$ stands for the Plotkin structure. If the minimum distances of the two component codes are d_1 and d_2, respectively, then the new code has minimum distance $d_{min} = \min(d_1, 2d_2)$ [51]. This construction has given good block codes, mostly notably Reed-Muller codes [51], and can be equally applied to convolutional codes [14, 15, 29, 36]. Refinements of the encoding and the hard-decision cascaded decoding procedures [8, 31] for this class of generalized concatenated convolutional codes of the Plotkin type (GCC-PT codes) are proposed in Section 3.2 to address the issue of error propagation. A new suboptimal multi-stage soft-decision decoding algorithm of the same structure as the hard-decision one is proposed in Section 3.3. Probability distribution functions of decision variables and the corresponding bit error rate (BER) bounds are obtained. Numerical results show that these codes achieve better performance than conventional convolutional codes with equal or less decoding complexity, and are capable of unequal error protection. An additional advantage of our approach is that the new decoding systems are based on standard hard-output Viterbi decoders with minimal modification in the input instead of custom designed devices as the case for SOVA.

In Section 3.4, memory is introduced into the Plotkin structure by replacing the inner block encoder $G'_{\mathbb{P}}$ with a differential encoder:

$$G'_{\mathbb{C}} = I_N + D^N I_N, \tag{3.2}$$

where D represents delay and $\mathbb{C}$ indicates the convolutionalness of the inner differential encoder. Namely, if $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(M)}$ is a sequence of M codewords of C, this construction results in a new sequence of $M+1$ codewords:

$$\mathbf{a}^{(1)}, \mathbf{a}^{(1)} \oplus \mathbf{a}^{(2)}, \mathbf{a}^{(2)} \oplus \mathbf{a}^{(3)}, \ldots, \mathbf{a}^{(M-1)} \oplus \mathbf{a}^{(M)}, \mathbf{a}^{(M)}.$$

With a two-iteration decoding algorithm based on the clipped soft decision algorithm from Section 3.3, this class of generalized concatenated convolutional codes of the convolutional type (GCC-CT codes) achieves significant gains over conventional convolutional coding systems with comparable complexity. If the underlying component code is (recursive) systematic, an alternative construction arises by excluding nonsystematic bits from the superimposing operation. Compared to nonsystematic GCC-CT codes, this class of codes, denoted by GCC-SCT codes, give smoother performance curves. Coding systems based on both classes of GCC-CT codes achieve performance that is comparable and sometimes superior to that of low-complexity turbo codes with comparable computational complexity.

3.2 Generalized Concatenated Convolutional Codes of the Plotkin Type

Suppose the code rates of the two component codes (CC1 and CC2) are R_1 and R_2, respectively. Find positive integers J, K, and N such that $\frac{J}{N} = R_1$ and $\frac{K}{N} = R_2$. Let $\mathbf{A}$ and $\mathbf{B}$ be independent random vectors of lengths J and K with equiprobable values from a binary field $\mathcal{F} = \{+1, -1\}$. $+1$ represents the additive identity of the group, i.e., $+1 \oplus \pm 1 = \pm 1$ and $-1 \oplus -1 = +1$. An encoder for the GCC-PT code and the channel model is illustrated in Fig. 3.1. The two information blocks $\mathbf{A}$ and $\mathbf{B}$ are first encoded by the encoders of the two component codes, $\mathcal{E}_1$ and $\mathcal{E}_2$, to obtain the codewords $\mathbf{a}$ and $\mathbf{b}$, respectively. $\mathbf{b}$ is then scrambled by an interleaver Π_N of size N and let $\tilde{\mathbf{b}}$ denote this interleaved codeword. The superimposed code block $\mathbf{c} = \mathbf{a} \oplus \tilde{\mathbf{b}}$ and the scrambled code block $\tilde{\mathbf{b}}$ are then transmitted. Notice that we have replaced

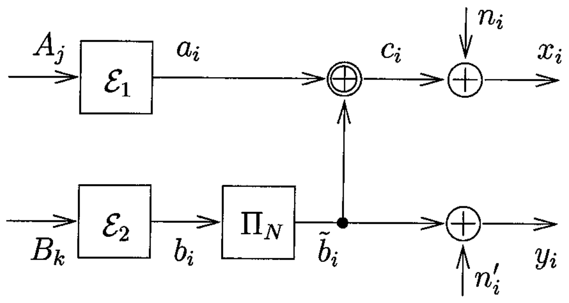


Figure 3.1 Superimposed code encoder and channel model.

the Plotkin structure of (3.1) by

$$G_{\mathbb{P}} = \begin{bmatrix} I_N & \mathbf{0}_{N,N} \\ \Pi_N & \Pi_N \end{bmatrix}, \tag{3.3}$$

based on reasons to presented in the next subsection. An additive white Gaussian noise (AWGN) channel is assumed, so the received signals x_i and y_i are the sum of the transmitted signals and independent white Gaussian noises n_i and n'_i with zero mean and variance $\sigma^2 = N_0/2E_s = N_0/2RE_b$, where E_s is the average energy per symbol, E_b is the average energy per bit, R is the overall code rate, and N_0 is the single-sided noise power spectral density. An example of this construction is given below.

Example Code 1 Let the two component codes CC1 and CC2 be rate-compatible punctured convolutional codes [37] of memory $\nu = 6$ and with rates $R_1 = 1/3$, $R_2 = 2/3$, and free distances $d_{f_1} = 14$, $d_{f_2} = 6$, respectively. Let Π_N be a 32×32 regular rectangular interleaver. The resulting GCC-PT code, denoted by EC1, is of overall code rate $R = 1/2$ and free distance $d_f = 12$.

3.2.1 Hard-Decision Decoding

The basic idea of decoding codes of the Plotkin structure with multiple stages [8,31] is to decode CC1 first and then subtract its effect from the received signal. The modified received signal is then used to decode CC2. In this multi-stage decoding scheme,

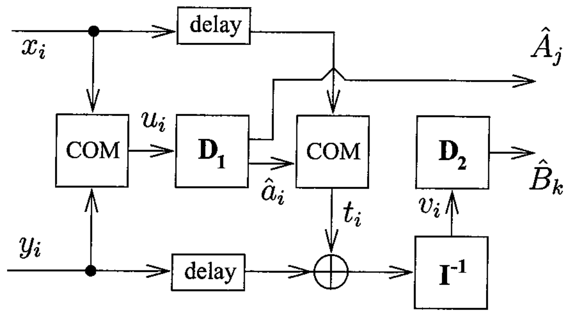


Figure 3.2 Block diagram of the generic decoder.

the quality of the first decoder output plays an essential role in the performance of the second decoder. For Viterbi decoding of convolutional codes, the errors in the output tend to be correlated and appear in bursts, which seriously impairs the performance of the subsequent decoder. To mitigate this problem, an interleaver Π_N and a deinterleaver Π_N^{-1} are introduced in the encoder and decoder, as shown in Fig. 3.1 and Fig. 3.2.

For the case of hard decision, the combining operation (COM) in Fig. 3.2 will be the addition of the field $\mathcal{F}$ and the two decoders will be Viterbi decoders for the two component codes. That is, the combiner output $\mathbf{u} \triangleq \mathbf{x} \oplus \mathbf{y}$ is fed into Viterbi decoder D_1, which then produces the estimate of the source sequence $\hat{\mathbf{A}}$ and the estimate of the codeword $\hat{\mathbf{a}}$. The latter is then combined with the received signal $\mathbf{x}$ to yield $\mathbf{t}$ as an estimate of the sequence $\tilde{\mathbf{b}}$. The two noisy copies, $\mathbf{y}$ and $\mathbf{t}$, of the interleaved codeword $\tilde{\mathbf{b}}$ comprise a repetition code of CC2 and hence can be decoded using a Viterbi decoder for the repetition of CC2. Alternatively, by defining a real addition '+' on $\mathcal{F}$, these signals can be combined into soft inputs for the Viterbi decoder of CC2 with values from $\{+2, 0, -2\}$. The algorithm is summarized as follows:

Hard-Decision Decoding for GCC-PT Codes	
HD1	$\mathbf{u} = \mathbf{x} \oplus \mathbf{y}$.
HD2	$(\hat{\mathbf{A}}, \hat{\mathbf{a}}) = \text{ViterbiDecoder}_1(\mathbf{u})$.
HD3	$\mathbf{t} = \mathbf{x} \oplus \hat{\mathbf{a}}$.
HD4	$\hat{\mathbf{A}} = \text{ViterbiDecoder}_2(\text{DeInterleaver}(\mathbf{t} + \mathbf{y}))$.

The complexity of this algorithm is therefore the average of those of the two component codes, i.e., $\mathcal{D} = r_1\mathcal{D}_1 + r_2\mathcal{D}_2$. Note that if the two component codes are punctured codes from the same parent code (as in the case for EC1), they can share the same decoder and only one physical implementation is required.

3.2.2 Performance Bounds

The bit error probabilities of the individual decoder can be bounded by the transfer function method [52]:

$$P_b \leq \frac{1}{k} \sum_{d=d_f}^{\infty} c_d P_d, \tag{3.4}$$

where c_d is the number of information bit errors associated with all the incorrect paths with distance d from the correct path and P_d is pairwise error probability for the two paths. For binary symmetric channels,

$$P_d = \begin{cases} \sum_{e=\frac{d+1}{2}}^{d} \binom{d}{e} p^e (1-p)^{d-e}, & \text{if } d \text{ odd} \\ \sum_{e=\frac{d}{2}+1}^{d} \binom{d}{e} p^e (1-p)^{d-e} + \frac{1}{2}\binom{d}{d/2} p^{d/2}(1-p)^{d/2}, & \text{if } d \text{ even} \end{cases}, \tag{3.5}$$

where p is the channel transition probability. Since $u_i = x_i \oplus y_i$ and $\Pr\{x_i \neq c_i\} = \Pr\left\{y_i \neq \tilde{b}_i\right\} = Q\left(\sqrt{2E_s/N_0}\right)$, we have

$$p = 2Q\left(\sqrt{\frac{2E_s}{N_0}}\right)\left[1 - Q\left(\sqrt{\frac{2E_s}{N_0}}\right)\right], \quad \text{for CC1,}$$

where $Q(x) \triangleq \int_x^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}y^2\right] dy$. By assuming correct decoding for CC1, $\hat{a}_i = a_i$ and hence $\hat{a}_i$ is independent of x_i and y_i. Thus,

$$p \approx Q\left(\sqrt{\frac{2E_s}{N_0}}\right), \quad \text{for CC2.}$$

Note also that, since two copies of CC2 codewords are transmitted, every d on the right-hand side of (3.5) should be replaced by $2d$.

3.2.3 Numerical Results

Numerical evaluation of these bounds and computer simulation results for EC1 are shown in Fig. 3.3. Only the six leading terms in (3.4) were used for computing the curves. For BER around 10^{-6}, CC1 achieves a coding gain of 0.8 dB over the NASA standard convolutional code. The NASA standard code is of rate 1/2 and memory 6 with free distance 10 and generator polynomials 133 and 171 in octal form. The introduction of the interleaver/deinterleaver pair increases the coding gain of CC2 from 0.8 dB to 1.5 dB. The overall BER for EC1, though not shown in the figure, is the appropriately weighted sum of the BERs for CC1 and CC2 and, in the present case, is close to that for CC1. The discrepancy between the bound and the simulation results for CC2 at low signal-to-noise ratios is due to the many errors made

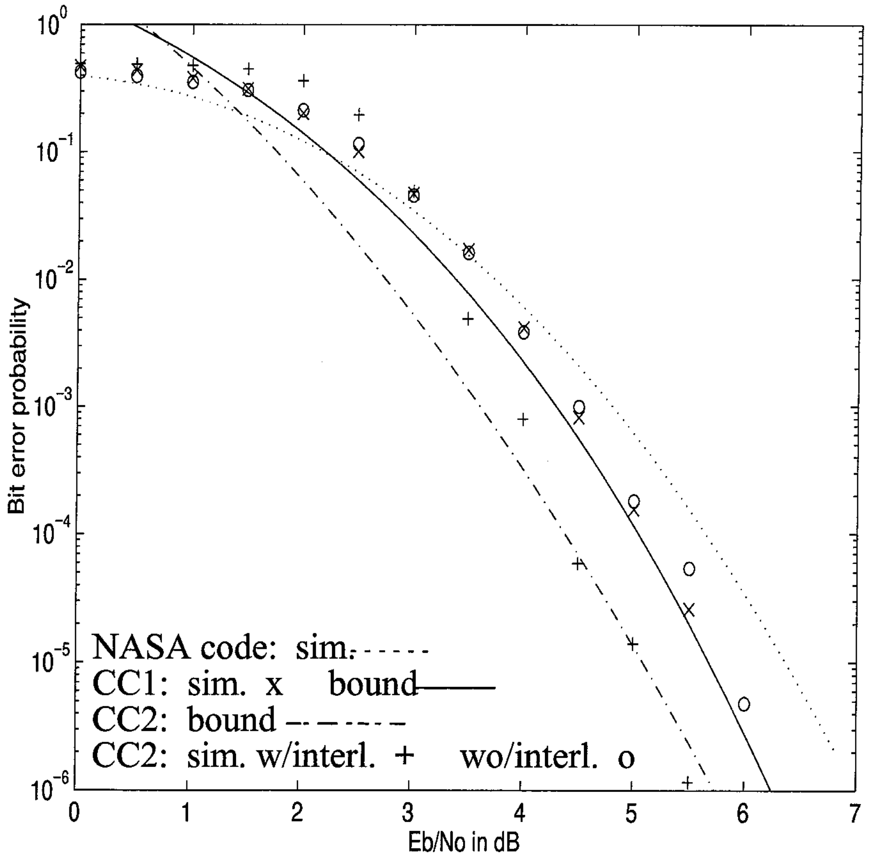


Figure 3.3 Bit error performance of EC1 with hard decision decoding (overall code rate is 0.5).

by the decoder of CC1, which invalidate the assumption that $p \approx Q\left(\sqrt{2E_s/N_0}\right)$. In addition, the length of the error bursts from the first decoder exceeds the interleaver depth and hence invalidates another assumption that the distribution of errors is random at the input of the second Viterbi decoder. For this example code and all those to be considered later here, random interleavers do not show any performance improvement over rectangular ones in our simulation, and hence only the results for rectangular interleavers are presented.

The unequal error protection capability of GCC-PT codes is a natural property of these codes since each of the two component codes yields a different level of error protection. This is illustrated by the numerical results here. Two thirds of the source bits enjoy a bit error rate more than ten times lower than for the rest. This can be exploited by some source coders (such as subband coders) to enhance the overall system performance. In spite of this excellent performance, the decoding complexity of EC1 is almost the same as the NASA code. Indeed, the average decoding complexity of the NASA code is $\mathcal{D}_{\text{NASA}} = 256$, and that of EC1 is $\mathcal{D}_{\text{EC1}} = r_1\mathcal{D}_1 + r_2\mathcal{D}_2 = \frac{1}{3}384 + \frac{2}{3}192 = 256$. (Note that C_2 is punctured from an $r = 1/3$ code).

3.3 Clipped Soft-Decision Decoding Algorithm

For soft channel outputs, the combining operation $\oplus$ is no longer applicable, but we would still like to use Viterbi decoders as the main decoding devices. The operation COM in Fig. 3.2 should be modified to provide some likelihood information of the received sequences for the Viterbi decoders. One such modification, clipped soft-decision decoding algorithm, is derived below.

Since a_i, $\tilde{b}_i$, n_i and n'_i are assumed to be independent, and a_i and $\tilde{b}_i$ take values from $\{+1, -1\}$ equiprobably, x_i and y_i are independent. Furthermore,

$$f(x_i, y_i \,|\, c_i) = f(x_i \,|\, c_i)\, f(y_i)$$
$$f(x_i, y_i \,|\, \tilde{b}_i) = f(x_i)\, f(y_i \,|\, \tilde{b}_i)$$

where $f(\cdot)$ and $f(\cdot|\cdot)$ are the probability density function (pdf) and the conditional pdf of the random variables. Hence, the log-likelihood ratio of c_i is

$$\Lambda(c_i) = \log \frac{\Pr\{x_i, y_i \,|\, c_i = +1\}}{\Pr\{x_i, y_i \,|\, c_i = -1\}} = \frac{2}{\sigma^2} x_i$$

and similarly $\Lambda(\tilde{b}_i) = \frac{2}{\sigma^2} y_i$. Because $a_i = c_i \oplus \tilde{b}_i$,

$$\begin{aligned}
\Lambda(a_i) &= \log \frac{\Pr\{x_i, y_i \,|\, a_i = +1\}}{\Pr\{x_i, y_i \,|\, a_i = -1\}} \\
&= \log \frac{e^{\frac{-1}{2\sigma^2}[(x+1)^2+(y+1)^2]} + e^{\frac{-1}{2\sigma^2}[(x-1)^2+(y-1)^2]}}{e^{\frac{-1}{2\sigma^2}[(x+1)^2+(y-1)^2]} + e^{\frac{-1}{2\sigma^2}[(x-1)^2+(y+1)^2]}} \\
&= \log \frac{1 + e^{\Lambda(\tilde{b}_i)+\Lambda(c_i)}}{e^{\Lambda(\tilde{b}_i)} + e^{\Lambda(c_i)}} \\
&\approx \operatorname{sgn}(\Lambda(\tilde{b}_i)\Lambda(c_i)) \, \min(|\Lambda(\tilde{b}_i)|, |\Lambda(c_i)|)
\end{aligned} \tag{3.6}$$

Therefore, for soft decision decoding, the COM operation in Fig. 3.2 will be defined as

$$\mathrm{COM}(x, y) \triangleq \operatorname{sgn}(xy) \min(|x|, |y|) \tag{3.7}$$

The decoding process defined in the previous section can then be applied to soft value inputs by simply modifying the first and third steps. More specifically, the algorithm proceeds as follows:

```
Clipped Soft-Decision Decoding for GCC-PT Codes

CSD1  $\mathbf{u} = \mathrm{COM}(\mathbf{x}, \mathbf{y})$.
CSD2  $(\hat{\mathbf{A}}, \hat{\mathbf{a}}) = \mathrm{ViterbiDecoder}_1(\mathbf{u})$.
CSD3  $\mathbf{t} = \mathrm{COM}(\mathbf{x}, \hat{\mathbf{a}})$.
CSD4  $\hat{\mathbf{A}} = \mathrm{ViterbiDecoder}_2(\mathrm{DeInterleaver}(\mathbf{t} + \mathbf{y}))$.
```

Since $\hat{a}_i$ takes value in $\{+1, -1\}$ only, $\mathrm{COM}(x_i, \hat{a}_i)$ functions like a "double-sided clipper" on x_i, as shown in Fig. 3.4. The above algorithm is similar to the generalized multiple concatenation decoding (GMCD) algorithm [67], which prescribes that $t_i =$

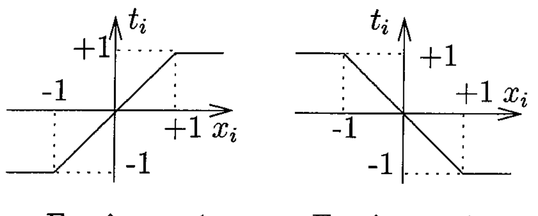


Figure 3.4 The input-output relationship of $t_i = \text{COM}(x_i, \hat{a}_i)$.

$\hat{a}_i x_i$, i.e., unclipped. Our simulation, however, shows that the proposed clipped soft-decision algorithm gives better performance than the GMCD does, especially for the class of codes to be presented in Section 3.4.

3.3.1 Performance Bounds

Without loss of generality, suppose $a_i = +1$, then $x_i = \tilde{b}_i + n_i$, $y_i = \tilde{b}_i + n'_i$. By the independence of n_i and n'_i, the joint conditional pdf of x_i and y_i is

$$f_{xy}(x_i, y_i \,|\, a_i = +1) = \frac{1}{4\pi\sigma^2}\left[e^{\frac{-1}{2\sigma^2}[(x_i-1)^2+(y_i-1)^2]} + e^{\frac{-1}{2\sigma^2}[(x_i+1)^2+(y_i+1)^2]}\right].$$

Since $u_i = \text{sgn}(x_i y_i)\min(|x_i||y_i|)$, the marginal cumulative conditional pdf of u_i can be obtained by integrating the above function over the regions shown in Fig. 3.5:

$$F_u(u_i \,|\, a_i = +1) = \begin{cases} 2Q\left(\frac{-u_i+1}{\sigma}\right) Q\left(\frac{-u_i-1}{\sigma}\right), & u_i < 0 \\ 1 - Q^2\left(\frac{u_i-1}{\sigma}\right) - Q^2\left(\frac{u_i+1}{\sigma}\right), & u_i \geq 0 \end{cases}.$$

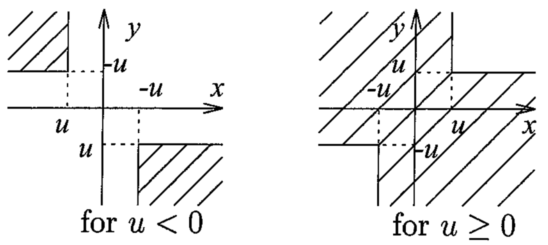


Figure 3.5 The integration regions for computing the cumulative conditional probability density function of u_i.

Differentiating this gives the conditional pdf of u_i:

$$f_u(u_i|a_i = +1) = 2\left[Q\left(\frac{|u_i|-1}{\sigma}\right) g_\sigma(u_i - 1) + Q\left(\frac{|u_i|+1}{\sigma}\right) g_\sigma(u_i + 1)\right], \tag{3.8}$$

where $g_\sigma(x) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{x^2}{2\sigma^2}\right]$. This function is plotted in Fig. 3.6 for several different signal-to-noise ratios. When compared to the input to the Viterbi decoders for conventional convolutional codes, which is a Gaussian random variable with unit mean and variance σ^2, here the conditional mean μ_u of u_i is smaller, the conditional variance σ_u^2 of u_i is also smaller. Since the bit error probability can be bounded by (3.4) with

$$P_d = \int_{-\infty}^{0} \underbrace{f_u(u_i|a_i = +1) * \cdots * f_u(u_i|a_i = +1)}_{d} \, du_i, \quad \text{for CC1},$$

if the free distance of CC1 is large (e.g. $d_{f_1} = 14$ for EC1), then by the central limit theorem argument, one would expect

$$P_d \approx Q\left(\sqrt{\frac{d\mu_u^2}{\sigma_u^2}}\right), \quad \text{for CC1}. \tag{3.9}$$

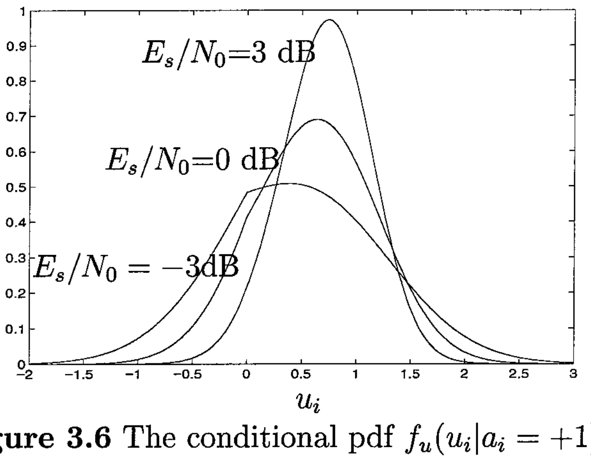


Figure 3.6 The conditional pdf $f_u(u_i|a_i = +1)$.

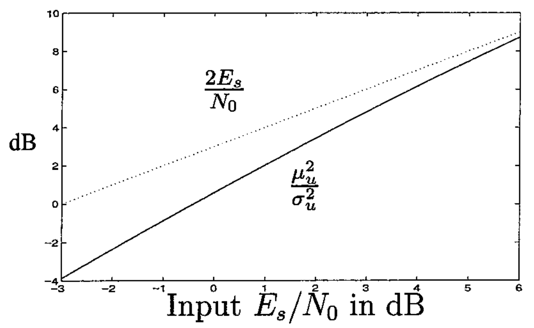


Figure 3.7 The SNR at the output of COM$(x_i, \hat{a}_i)$.

As shown in Fig. 3.7 by numerical integration of (3.8), the ratio μ_u^2/σ_u^2 approaches $2E_s/N_0$ at large input signal-to-noise ratios. That is, because $P_d = Q\left(\sqrt{2dE_s/N_0}\right)$ for maximum likelihood decoding, the performance of this suboptimal decoder approaches that of maximum likelihood decoding asymptotically.

If CC1 decodes perfectly, (e.g., at high signal-to-noise ratios,) $\hat{a}_i = a_i$ and hence $\hat{a}_i$ is independent of x_i and y_i. Thus the conditional pdf of t_i can be approximated by

$$f_t(t_i|\tilde{b}_i = +1) \approx \frac{1}{2}\delta(t_i - 1) + Q\left(\frac{2}{\sigma}\right)\delta(t_i + 1) + g_\sigma(t_i - 1)[s(t_i + 1) - s(t_i - 1)],$$

where $\delta(t)$ is an impulse function at $t = 0$ and $s(t)$ is a unit step function: $s(t) = 0$ for $t < 0$ and $s(t) = 1$ for $t \geq 0$. Since n_i and n_i' are independent,

$$f_v(v_i|b_i = +1) \approx \frac{1}{2}g_\sigma(v_i - 2) + Q\left(\frac{2}{\sigma}\right)g_\sigma(v_i) \\ + g_\sigma(v_i - 1) * \{g_\sigma(v_i - 1)[s(v_i + 1) - s(v_i - 1)]\}. \quad (3.10)$$

This function is plotted in Fig. 3.8 for two different signal-to-noise ratios. Also shown in the figure is a histogram obtained from computer simulation for $E_s/N_0 = 0$ dB, which agrees with the theoretical values so well that the two curves are hardly distinguishable. The bit error probability of CC2 can be bounded by (3.4) with

$$P_d \approx \int_{-\infty}^{0} \underbrace{f_v(v_i|b_i = +1) * \cdots * f_v(v_i|b_i = +1)}_{d}\, dv_i, \quad \text{for CC2.} \quad (3.11)$$

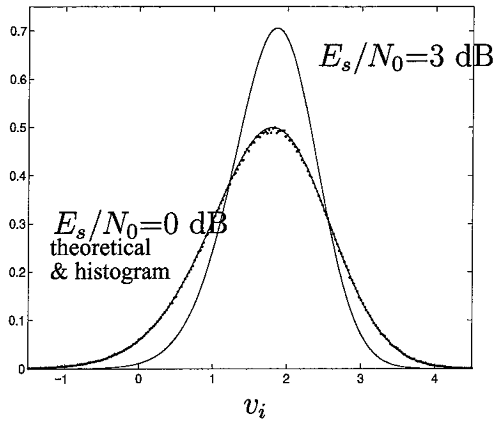


Figure 3.8 The conditional pdf $f_v(v_i|b_i = +1)$.

Unlike the case of CC1, however, d_{f_2} is usually too small (e.g. $d_{f_2} = 6$ for EC1) to justify the central limit theorem. The evaluation of P_d generally can not be closely approximated by the approach used in (3.9), and (3.11) should be applied instead.

3.3.2 Numerical Results

Our first concern for the clipped soft-decision algorithm is about how well the approximation in (3.6) performs. Fortunately, simulation results show that the approximation introduces a penalty of about 0.1 dB loss in E_b/N_0. This is a small price to pay, in view of the fact that the approximation not only gets rid of the log exponential sums but also allows the decoder to work without the knowledge of the channel signal-to-noise ratio.

Fig. 3.9 shows numerical results for EC1, where only the six leading terms in (3.4)

CC1 candidates			CC2 candidates		
R_1	d_{f_1}	E_s/N_0	R_2	$2d_{f_2}$	E_s/N_0
1/2	10	2.96	8/9	6	3.31
4/9	10	2.57	4/5	8	2.15
2/5	11	2.16	2/3	12	0.52
4/11	12	1.78	4/7	14	-0.24
1/3	14	1.42			

Table 3.1 The required E_s/N_0 in dB for BER = 10^{-6}.

were computed. With the same code rate and decoding complexity, EC1 outperforms the NASA code: while CC1 has comparable performance to the NASA code, CC2 provides a coding gain of 1.1 dB at BER=10^{-6}. The bit error probability bounds derived in the previous subsection prove to be tight and thus useful for performance prediction, especially at high signal-to-noise ratios. The discrepancy between the bound for CC2 and the simulation results at low signal-to-noise ratios is due to the many errors made by the first decoder, which invalidates several assumptions used in the derivation, as described before.

Table 3.1 lists the required signal-to-noise ratios for BER=10^{-6} for several rate-compatible punctured convolutional codes of memory $\nu = 6$ [37] computed by the bounds (3.4), (3.9) and (3.11). Using these values, it is possible to design a wide

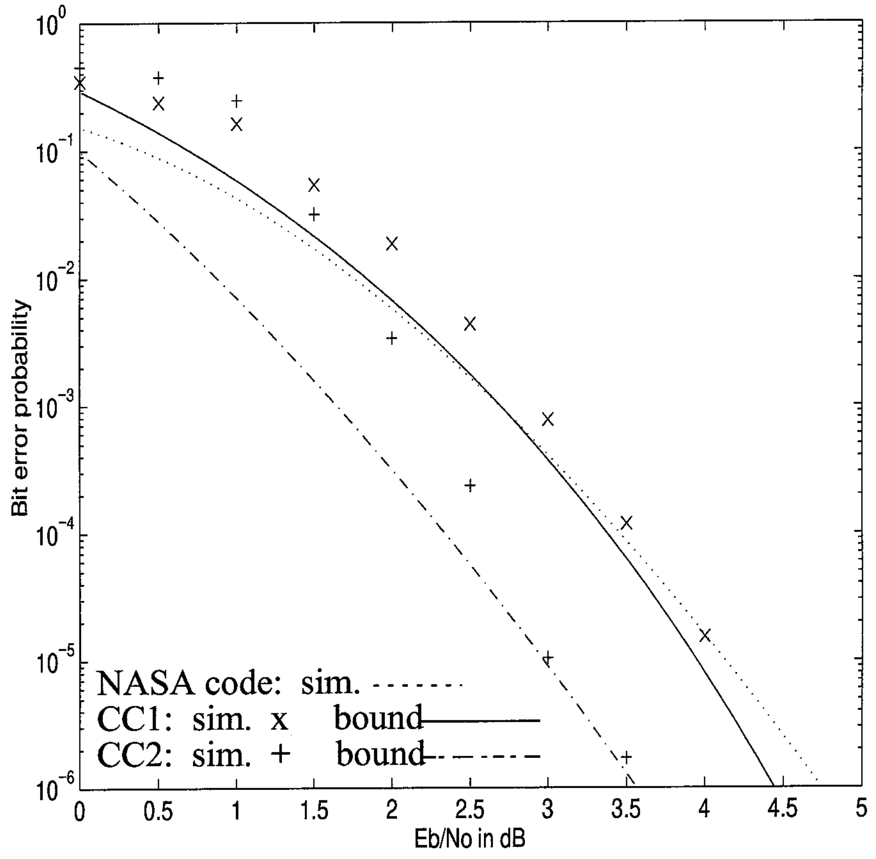


Figure 3.9 Bit error performance of EC1 with soft decision decoding (overall code rate is 0.5).

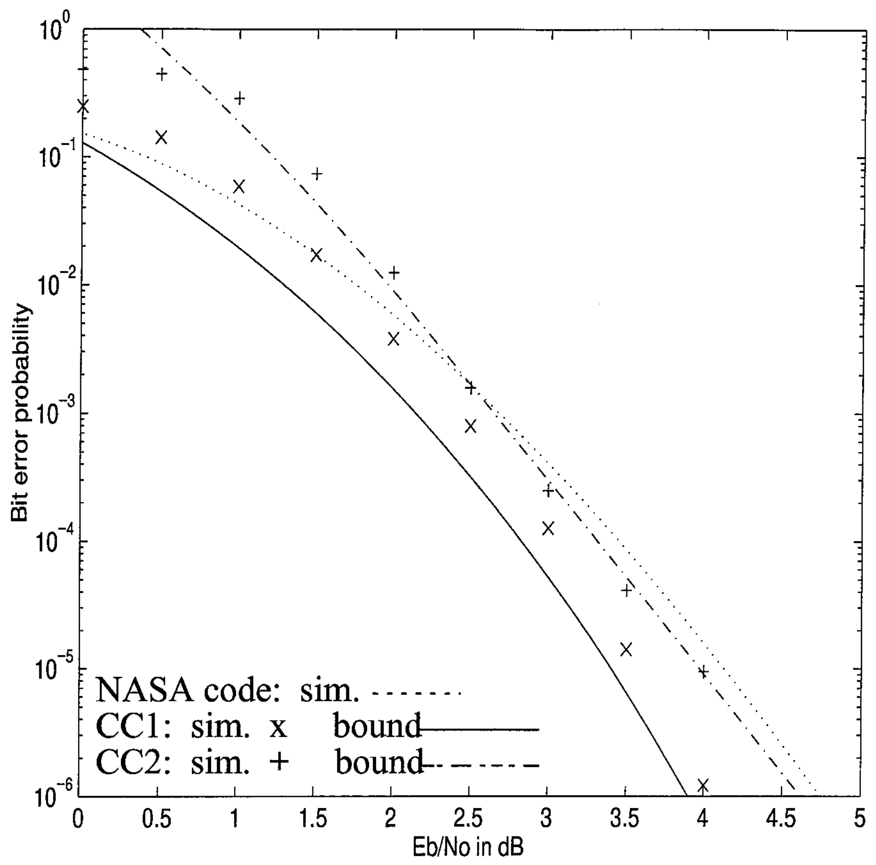


Figure 3.10 Bit error performance of EC2 with soft decision decoding (overall code rate is 0.567).

range of error control systems, from unequal error protection such as EC1, to codes that maximize channel throughput. To illustrate the latter idea, a second example of GCC-PT code is constructed as follows.

Example Code 2 Let the $R_1 = 1/3$ code from Table 3.1 be CC1 and the $R_2 = 4/5$ code from Table 3.1 be CC2. With the same interleaver as that for EC1, the new code EC2 has an overall code rate of 0.567 (=2.46 dB), which is about 13% higher than that of the NASA code. Table 1 predicts that the E_b/N_0 values at BER=10^{-6} will be about 3.9 (=1.42+2.46) dB for CC1 and 4.6 (=2.15+2.46) dB for CC2, which are equivalent to coding gains of about 0.7 dB and 0.1 dB over the NASA code. Simulation results shown in Fig. 3.10 confirm these predictions. Note that EC2 also has a lower average decoding complexity than the NASA code, viz., $\mathcal{D}_{\mathrm{EC2}} = \frac{5}{17}384 + \frac{12}{17}160 \approx 226 < 256$.

We also notice that the bound for CC2 agrees with the simulation points very well even at low signal-to-noise ratios. This is because of the relative small number of errors made by the first decoder.

3.3.3 Iterative Decoding

It should be noted that GCC-PT codes also allow iterative decoding to improve performance. For the example of EC1, Fazel [29] has proposed a scheme using soft-output Viterbi algorithm (SOVA) and "soft-re-encoding" to bring the performance of CC1 to be about the same as that of CC2, which is about 2.7 dB at BER=10^{-4}, as shown in Fig. 3.9. Because of the back-tracking operations in SOVA [38], the decoder is more complex than an ordinary Viterbi decoder. In terms of VLSI implementation, it is estimated that a SOVA decoder requires about 40–60% more chip area than a Viterbi decoder [44]. However, with the new constructions and iterative decoding algorithms of the next section, it is possible to achieve better performance using conventional hard-output Viterbi decoders.

3.4 Generalized Concatenated Convolutional Codes of the Convolutional Type

Memory can be incorporated into the Plotkin structure by replacing the inner block code with a convolutional code. Two such constructions based on a simple differential encoder are presented below for general convolutional outer codes and recursive systematic convolutional outer codes.

3.4.1 General Construction

For the same reasons presented in Section 3.2, the differential encoder (3.2) is modified as follows to address the issue of error propagation in a multistage decoder:

$$G_{\mathbb{C}} = I_N + D^N \Pi_N. \tag{3.12}$$

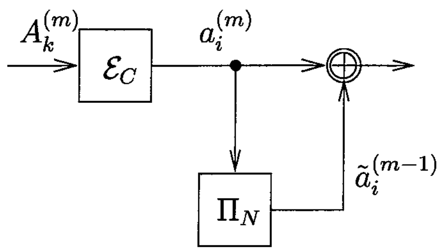


Figure 3.11 Hyperimposed code encoder.

The corresponding encoder block diagram is illustrated in Fig. 3.11, where $\mathcal{E}_C$ is the encoder for the underlying component code C and Π_N is an interleaver of size N. The m-th input information block $\mathbf{A}^{(m)}$ is first encoded by $\mathcal{E}_C$ to obtain a codeword $\mathbf{a}^{(m)}$, which is then fed into the interleaver. The current output from the interleaver is the scrambled version, $\tilde{\mathbf{a}}^{(m-1)}$, of the previous codeword $\mathbf{a}^{(m-1)}$, which corresponds to the information block $\mathbf{A}^{(m-1)}$. The m-th codeword of the GCC-CT code is then defined as the superimposing of the current output from both of the devices: $\mathbf{c}^{(m)} = \mathbf{a}^{(m)} \oplus \tilde{\mathbf{a}}^{(m-1)}$. For a sequence of M input information blocks $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(M)}$, let $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(M)}$ be the corresponding codewords encoded by $\mathcal{E}_C$ and $\tilde{\mathbf{a}}^{(1)}, \tilde{\mathbf{a}}^{(2)}, \ldots, \tilde{\mathbf{a}}^{(M)}$ be the interleaved versions of those. The construction results in a sequence of $M+1$ codewords:

$$\begin{aligned} \mathbf{c}^{(1)} &= \mathbf{a}^{(1)}, \\ \mathbf{c}^{(m)} &= \mathbf{a}^{(m)} \oplus \tilde{\mathbf{a}}^{(m-1)}, \qquad m = 2, 3, \ldots, M, \\ \mathbf{c}^{(M+1)} &= \tilde{\mathbf{a}}^{(M)}. \end{aligned}$$

Let $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(M+1)}$ denote the received blocks corresponding to those codewords over the AWGN channel defined in Section 3.2.

This construction embeds two copies of a codeword of the underlying component code C in the new code blocks: $\mathbf{a}^{(m)}$ in $\mathbf{c}^{(m)}$ and $\tilde{\mathbf{a}}^{(m)}$ in $\mathbf{c}^{(m+1)}$. Using the techniques from the previous sections, estimate signals of these two copies can be recovered and combined for better decoding performance. More specifically, begin with the first received block $\mathbf{x}^{(1)}$. Since $\mathbf{c}^{(1)} = \mathbf{a}^{(1)}$, this can be decoded by the Viterbi decoder of C to obtain the estimate $\hat{\mathbf{a}}^{(1)}$. Moreover, let $\mathbf{x}^{(1)} = \mathbf{t}^{(1)}$, which denotes the first recovered signal of the codeword $\mathbf{a}^{(1)}$. The next received block $\mathbf{x}^{(2)}$ is a noisy version of the superimposed code block $\mathbf{c}^{(2)} = \mathbf{a}^{(2)} \oplus \tilde{\mathbf{a}}^{(1)}$. Since an estimate of $\mathbf{a}^{(1)}$ is available from decoding the previous block, the combining operation from the clipped soft-decision algorithm can be used to obtain the first recovered signal of the codeword $\mathbf{a}^{(2)}$: $\mathbf{t}^{(2)} = \mathrm{COM}(\mathbf{x}^{(2)}, \mathrm{Interleaver}(\hat{\mathbf{a}}^{(1)}))$. Applying the Viterbi algorithm on this recovered signal gives the estimate $\hat{\mathbf{a}}^{(2)}$ of codeword $\mathbf{a}^{(2)}$. Combining this estimate $\hat{\mathbf{a}}^{(2)}$ with the received signal $\mathbf{x}^{(2)}$ again recovers the signal for $\tilde{\mathbf{a}}^{(1)}$. Hence, a second recovered signal for $\mathbf{a}^{(1)}$ can be obtained by passing this signal through the deinterleaver: $\mathbf{u}^{(1)} = \mathrm{DeInterleaver}(\mathrm{COM}(\mathbf{x}^{(2)}, \hat{\mathbf{a}}^{(2)}))$. At this point, signals for both of the copies of codeword $\mathbf{a}^{(1)}$ are recovered. The final estimate for the information block $\mathbf{A}^{(1)}$ can then be made based on both of these signals: $\hat{\mathbf{A}}^{(1)} = \mathrm{ViterbiDecoder}(\mathbf{t}^{(1)} + \mathbf{u}^{(1)})$. Since $\hat{\mathbf{a}}^{(2)}$ and $\mathbf{t}^{(2)}$ are available, these procedures can be applied recursively to the subsequent received signals. The complete decoding algorithm is summarized as follows:

```
Extended Decoding Algorithm for GCC-CT Codes

Step 1  $\mathbf{t}^{(1)} = \mathbf{x}^{(1)}$,
        $\hat{\mathbf{a}}^{(1)} = \mathrm{ViterbiDecoder}(\mathbf{x}^{(1)})$.
Step 2  For $m = 1, 2, \ldots, M-1$ do
            $\mathbf{t}^{(m+1)} = \mathrm{COM}(\mathbf{x}^{(m+1)}, \mathrm{Interleaver}(\hat{\mathbf{a}}^{(m)}))$,
            $\hat{\mathbf{a}}^{(m+1)} = \mathrm{ViterbiDecoder}(\mathbf{t}^{(m+1)})$,
            $\mathbf{u}^{(m)} = \mathrm{DeInterleaver}(\mathrm{COM}(\mathbf{x}^{(m+1)}, \hat{\mathbf{a}}^{(m+1)}))$,
            $\hat{\mathbf{A}}^{(m)} = \mathrm{ViterbiDecoder}(\mathbf{t}^{(m)} + \mathbf{u}^{(m)})$,
            Output $\hat{\mathbf{A}}^{(m)}$.
Step 3  $\mathbf{u}^{(M)} = \mathrm{DeInterleaver}(\mathbf{x}^{(M+1)})$,
        $\hat{\mathbf{A}}^{(M)} = \mathrm{ViterbiDecoder}(\mathbf{t}^{(M)} + \mathbf{u}^{(M)})$,
        Output $\hat{\mathbf{A}}^{(M)}$.
Stop
```

Because each of the code blocks is decoded twice, the complexity of this algorithm is about twice that of decoding the underlying component code.

Example Codes 3 and 4 Define EC3 as the GCC-CT code with the NASA standard code as its component code and a 64×64 rectangular interleaver in the encoder. $M = 100$ so the overall code rate is about 1/2. Simulation results of EC3 with extended decoding are shown in Fig. 3.12. At BER=10^{-6}, EC3 achieves a coding gain of 1.9 dB over the NASA code. Apart from the extra storage requirement for the interleavers, the arithmetic decoding complexity of EC3 is about twice that of the NASA code. Also shown in Fig. 3.12 is the performance of EC4, which is constructed with a 64×64 rectangular interleaver and a rate one half convolutional code with five memory elements. The generators of this code are 53 and 75 in octal form. The

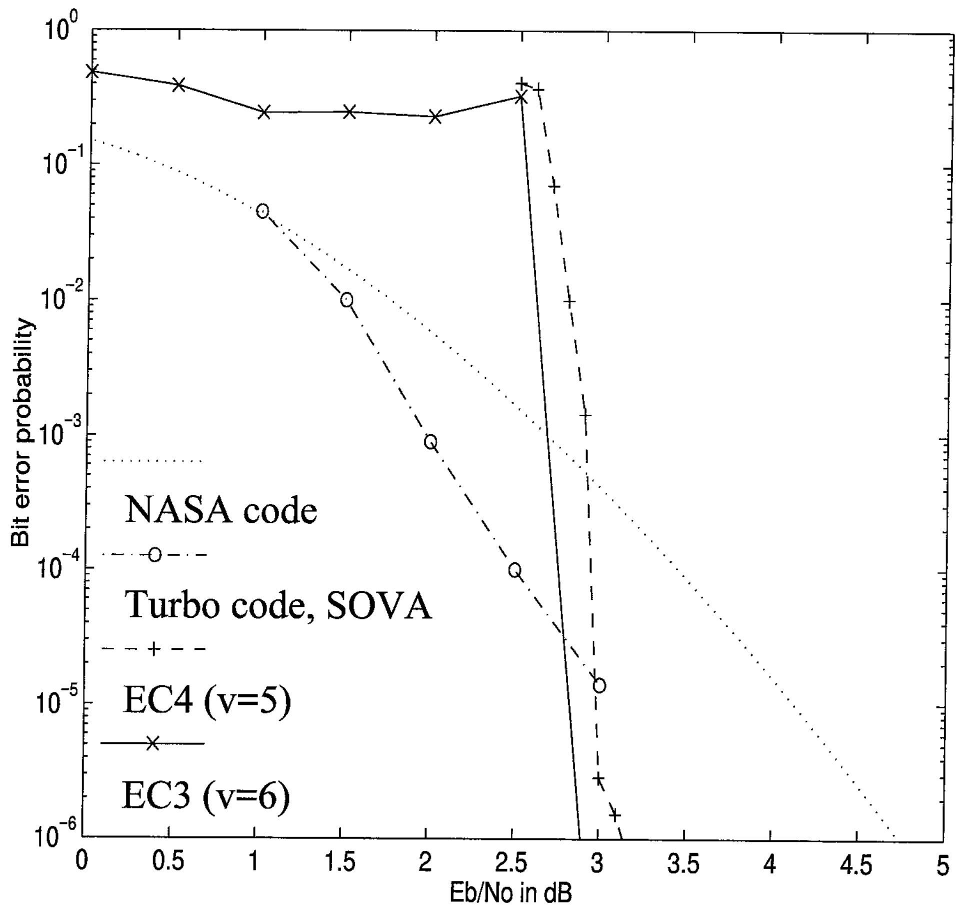


Figure 3.12 Bit error rates of hyperimposed codes and a turbo code with SOVA (overall code rate is 0.5).

decoding complexity of EC4 is roughly the same as the NASA code, but it achieves a coding gain of about 1.7 dB relative to the later at BER=10^{-6}.

It is observed that the bit error rate curves of GCC-CT codes exhibit a threshold effect: the bit error rates rise to a uselessly high level when the signal-to-noise ratio (SNR) drops below a certain point. We believe the explanation is as follows. If the decoding of the previous stages is perfect (e.g., $\hat{\mathbf{a}}^{(m-1)} = \mathbf{a}^{(m-1)}$ and $\hat{\mathbf{a}}^{(m+1)} = \mathbf{a}^{(m+1)}$ when $\mathbf{a}^{(m)}$ is to be re-decoded), then the effective minimum distance of the code is almost doubled, though the metric used in decoding algorithm is not optimal. This explains why the performance is so good for higher SNR. On the other hand, for every error decision made in the previous stages, there are at least d_f errors in the re-encoded sequence which are then spread by the interleaver. The effective distance of the current stage is reduced by two for every "hit" of such errors. Consequently, the performance of extended decoding deteriorates dramatically as the bit error rates of the decoding previous stages increase.

3.4.2 Systematic Recursive Construction

The re-encoding process that causes error propagation could be avoided by (1) using systematic codes (preferably recursive ones for larger free distances) and (2) excluding nonsystematic bits from the superimposing operation. Suppose the block length of the underlying component code C is N, of which the first K bits are systematic. The

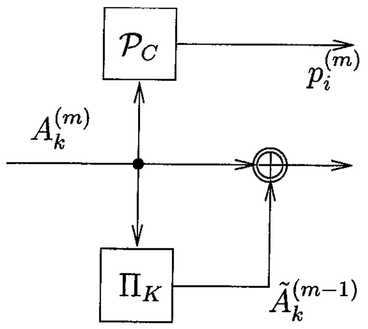


Figure 3.13 CC-SCT code encoder.

inner encoder is defined as

$$G_{\mathbb{SC}} = I_N + D^N \begin{bmatrix} \Pi_K & \mathbf{0}_{K,N-K} \\ \mathbf{0}_{N-K,K} & \mathbf{0}_{K,K} \end{bmatrix}, \tag{3.13}$$

where Π_K is permutation matrix of size K, $\mathbf{0}_{H,L}$ is a $H \times L$ matrix of all zero elements, and $\mathbb{SC}$ indicates the systematic convolutionalness of the inner code. The corresponding encoder is illustrated in Fig. 3.13, where $\mathcal{P}_C$ represents the parity-check generator of C. Namely, for a sequence of M input information blocks $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(M)}$, let $\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(M)}$ be the corresponding parity blocks encoded by $\mathcal{P}_C$ and $\tilde{\mathbf{A}}^{(1)}, \tilde{\mathbf{A}}^{(2)}, \ldots, \tilde{\mathbf{A}}^{(M)}$ be the interleaved versions of the information blocks. The construction results in a sequence of $M+1$ codewords:

$$\begin{aligned} \mathbf{c}^{(1)} &= (\mathbf{c_i}^{(1)}, \mathbf{c_p}^{(1)}) = (\mathbf{A}^{(1)}, \mathbf{p}^{(1)}), \\ \mathbf{c}^{(m)} &= (\mathbf{c_i}^{(m)}, \mathbf{c_p}^{(m)}) = (\mathbf{A}^{(m)} \oplus \tilde{\mathbf{A}}^{(m-1)}, \mathbf{p}^{(m)}), \qquad m = 2, 3, \ldots, M, \\ \mathbf{c}^{(M+1)} &= (\mathbf{c_i}^{(M+1)}, \mathbf{c_p}^{(M+1)}) = (\tilde{\mathbf{A}}^{(M)}, \mathbf{0}). \end{aligned}$$

The block of $N-K$ zeros in the last codeword do not need to be transmitted. Let $\mathbf{x_i}^{(m)}, m = 1, 2, \ldots, M+1$, denote the received blocks corresponding to the systematic blocks $\mathbf{c_i}^{(m)}$, and $\mathbf{x_p}^{(m)}, m = 1, 2, \ldots, M$, denote those corresponding to the parity-check blocks $\mathbf{c_p}^{(m)}$, respectively. The decoding algorithm for this GCC-SCT codes is similar to that for the GCC-CT codes except that now only the systematic bits of the codewords are to be recovered and combined. The complete algorithm is summarized as follows:

Extended Decoding Algorithm for GCC-SCT Codes

```
Step 1  $\mathbf{T}^{(1)} = \mathbf{x_i}^{(1)}$,
        $\hat{\mathbf{A}}^{(1)} = \text{ViterbiDecoder}((\mathbf{x_i}^{(1)}, \mathbf{x_p}^{(1)}))$.
Step 2  For $m = 1, 2, \ldots, M-1$ do
            $\mathbf{T}^{(m+1)} = \text{COM}(\mathbf{x_i}^{(m+1)}, \text{Interleaver}(\hat{\mathbf{A}}^{(m)}))$,
            $\hat{\mathbf{A}}^{(m+1)} = \text{ViterbiDecoder}((\mathbf{T}^{(m+1)}, \mathbf{x_p}^{(m+1)}))$,
            $\mathbf{U}^{(m)} = \text{DeInterleaver}(\text{COM}(\mathbf{x_i}^{(m+1)}, \hat{\mathbf{A}}^{(m+1)}))$,
            $\hat{\mathbf{A}}^{(m)} = \text{ViterbiDecoder}((\mathbf{T}^{(m)} + \mathbf{U}^{(m)}, \mathbf{x_p}^{(m)}))$,
            Output $\hat{\mathbf{A}}^{(m)}$.
Step 3  $\mathbf{U}^{(M)} = \text{DeInterleaver}(\mathbf{x_i}^{(M+1)})$,
        $\hat{\mathbf{A}}^{(M)} = \text{ViterbiDecoder}((\mathbf{T}^{(M)} + \mathbf{U}^{(M)}, \mathbf{x_p}^{(M)}))$,
        Output $\hat{\mathbf{A}}^{(M)}$.
Stop
```

Example Codes 5 and 6 The performance of two designs with 64×64 rectangular interleavers are shown in Fig. 3.14, where EC5 is based on the recursive systematic version of the NASA code and EC6 is based on the recursive version of the memory five code that was used in EC4 with half of the parity bits punctured to increase the rate to 2/3. As predicted, these codes exhibit less threshold effect and hence achieve better performance than the nonsystematic GCC-CT codes for low SNR. However, since less received signals (only those correspond to the systematic bits) are accumulated in the decoding algorithm, they are not as good as the nonsystematic codes for higher SNR. For example, the coding gain for EC3 is 0.5 dB higher than that of EC5 at BER=10^{-6}. The decoding complexity for EC5 is about twice that of the NASA code, and that of EC6 is about 80% of the NASA code. Also shown in Fig. 3.12 and 3.14 are the performance of a turbo code of memory 2 and block size 900 with six-iteration SOVA decoding for rates 1/2 and 2/3, taken from [40]. The complexity of both schemes is about twice of that of the NASA code. In the region of practical interests, however, the performance of GCC-SCT codes is superior to that of these low-complexity turbo codes.

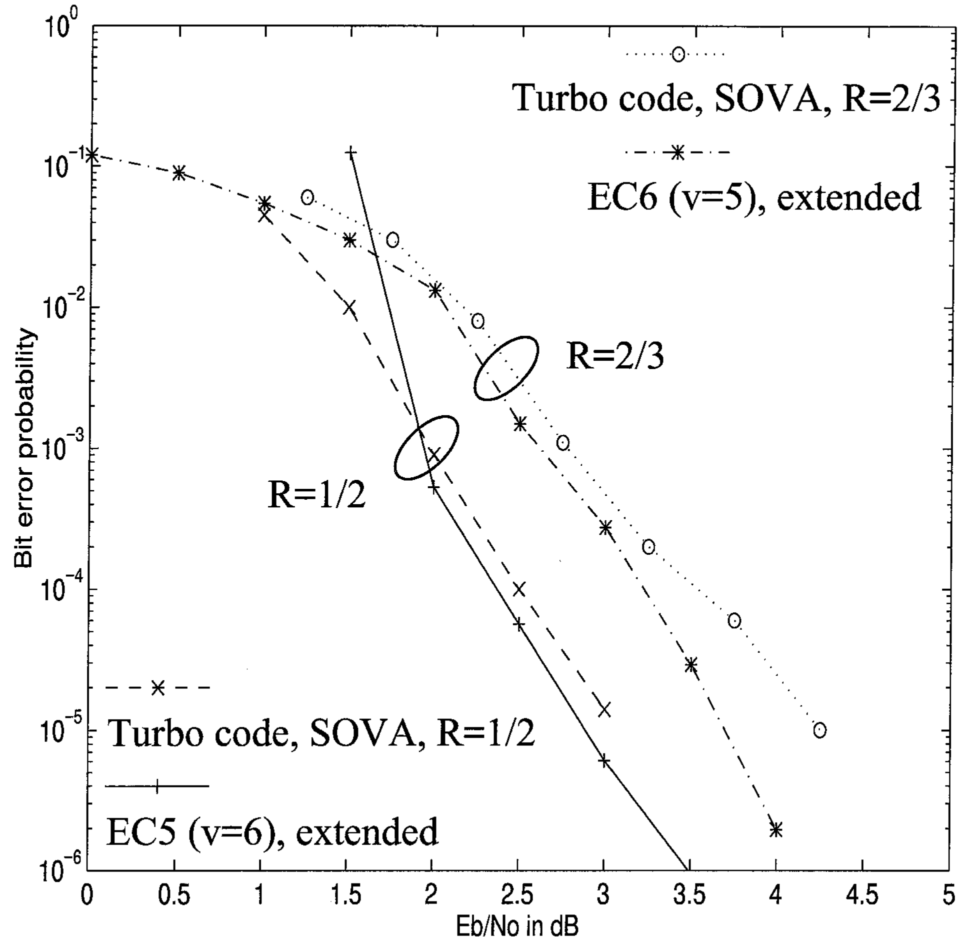


Figure 3.14 Bit error rates of systematic hyperimposed codes and turbo codes with SOVA.

3.5 Conclusions

Powerful coding systems are obtain with the generalized concatenation of convolutional outer code. We proposed new suboptimal multistage decoding algorithms for codes of the Plotkin structure, and obtained the corresponding upper bounds on bit error probabilities. Numerical results show that these codes provide good coding gains and unequal error protection. The classical Plotkin construction was then generalized to incorporate memory in the inner code. With a two-iteration decoding algorithm, these codes achieve comparable and sometimes superior performance to that of low-complexity turbo codes. Unlike turbo codes, random interleavers do not show any performance improvement over rectangular ones for all the example codes we considered. Our systems can be considered a very pragmatic approach to the iterative decoding paradigm since the new decoding systems can be easily constructed with off-the-shelf industry standard hard-output Viterbi decoders with minimal modification of the input.

Chapter 4
Turbo Decoding and Belief Propagation

Though the turbo decoding methods have been applied to a wide range of codes with remarkable success, a satisfactory theoretical explanation as to why the turbo decoding algorithm performs so well has been lacking. Some pathological cases where the algorithm does not converge or converge to the wrong answers have even been identified [53]. In this chapter, we establish a close connection between the turbo decoding mechanism and the "belief propagation" algorithm [58], well-known in the artificial intelligence community. An introduction to the Bayesian belief network and Pearl's belief propagation algorithm are given in Section 4.1. The optimal symbol decision rules for conventional systematic and general parallel concatenated (turbo) codes are discussed in Section 4.2. We show, in Section 4.3, that the turbo decoding algorithm is in fact a special case of the belief propagation algorithm applied to the Bayesian network of turbo codes. Section 4.4 concludes the chapter with a brief review of some other decoding algorithms that are related to belief propagation.

4.1 Introduction to Bayesian Networks and Belief Propagation

The general probabilistic inference problem and the Bayesian belief network approach to this problem are introduced. A detailed description of Pearl's belief propagation algorithm, which solves some special Bayesian networks efficiently, is given next.

4.1.1 Probabilistic Inference and Bayesian Networks

Consider a set of N discrete random variables $\mathbf{V} = \{V_1, V_2, \ldots, V_N\}$, where V_i assumes values in a finite alphabet $\mathcal{F}_i$. Let the marginal and joint probability density functions be:

$$\begin{aligned} q(v_i) &\triangleq \Pr\{V_i = v_i\}, \\ q(\mathbf{v}) &\triangleq \Pr\{V_1 = v_1, V_2 = v_2, \ldots, V_N = v_N\}. \end{aligned}$$

The marginal density function $q(v_i)$ represents our "*a priori* belief" about the random variable V_i. Now suppose realizations of some of the variables are observed: the random variable V_j is known to be v_j, for all j in $J \subseteq \{1, 2, \ldots, N\}$. Define the "evidence" to be this event $\mathcal{E} = \{V_j = v_j : j \in J\}$. The fundamental probabilistic inference problem is to compute the "*a posteriori* beliefs" given the evidence $\mathcal{E}$, i.e., to find the conditional probability densities

$$q(v_i \mid \mathcal{E}) \triangleq \Pr\{V_i = v_i \mid \mathcal{E}\},$$

for all $i \notin J$. A brute force approach to this problem is to sum over all of the random variables of $q(\mathbf{v})$ that do not involve either i or J. That is, without loss of generality, if $J = \{n+1, n+2, \ldots, N\}$, then

$$q(v_n \mid \mathcal{E}) = \frac{\sum_{v_1, v_2, \ldots, v_{n-1}} q(\mathbf{v})}{\sum_{v_1, v_2, \ldots, v_n} q(\mathbf{v})}.$$

By defining $\blacktriangle$ as a normalization operator on a collection of nonnegative numbers y_i such that $\sum_i x_i = 1$ if $x_i = \blacktriangle_i y_i$, we can rewrite the equation as

$$q(v_n \,|\, \mathcal{E}) = \blacktriangle_{v_n} \sum_{v_1, v_2, \cdots, v_{n-1}} q(\mathbf{v}). \tag{4.1}$$

If V_i can assume $|\mathcal{F}_i| = M_i$ values, then the above sum involves $M_1 M_2 \cdots M_{n-1}$ terms, which implies this approach is impractical unless n and the M_i's are small.

The idea behind the "Bayesian belief network" approach [43, 58] to this inference problem is to exploit the "partial independence" that may exist among the random variables to simply belief updating. For example, if all the variables are independent, then the computation can be completely avoided. More generally, the partial independence can be described systematically by a "directed acyclic graph" (DAG), which is a finite, directed graph without any directed cycles [9]. If there exists a directed edge from vertex U to vertex V, denoted by $U \to V$, then U is called a parent of V. Define $A(V)$ to be the set of all the parents of V. For example, for the DAG in Fig. 4.1, we have $A(V_1) = \varnothing, A(V_2) = \varnothing, A(V_3) = \{V_1\}, A(V_4) = \{V_1, V_2\}$, and $A(V_5) = \{V_4\}$. If $\mathbf{V}$ is a set of random variables in one-to-one correspondence with the vertices of a DAG G, the joint density function $q(\mathbf{V})$ factors according to G if

$$q(v_1, v_2, \ldots, v_N) = \prod_{i=1}^{N} q(v_i \,|\, A(v_i)),$$

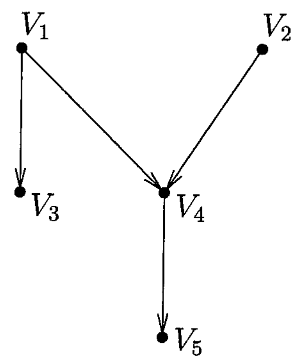


Figure 4.1 Example of a directed acyclic graph.

where $A(v_i)$ denotes a value assignment for the set $A(V_i)$ of the parents of V_i. For example, a five-variable density function factors according to the graph of Fig. 4.1 if

$$q(v_1, v_2, v_3, v_4, v_5) = q(v_1)q(v_2)q(v_3 \mid v_1)q(v_4 \mid v_1 v_2)q(v_5 \mid v_4).$$

Note also the Markovian property implied in the graph: If $V_i \to V_j \to V_k$, then

$$q(V_i V_k \mid V_j) = q(V_k \mid V_j)q(V_i \mid V_j).$$

This set of random variables $\mathbf{V}$ and the corresponding DAG G is called a Bayesian belief network [43, 58].

By systematically exploiting the partial independence described by the belief network, it is possible to simplify the computation considerably over the brute force approach (4.1). While the probabilistic inference problem for general belief networks may still be very hard, (NP-hard, in fact [21, 69],) Pearl [57, 58] has found an efficient distributed algorithm to solve this problem for the special case where the DAG is a tree, i.e., if there are no undirected loops. His "belief propagation" algorithm, to be described in the next section, solves the inference problem with $O(NM^\eta)$ computations, where $\eta = \max_i(|A(V_i)|)$ the largest number of parents of any vertex, rather than the exponential complexity $O(M_1 M_2 \cdots M_{n-1})$ required by (4.1).

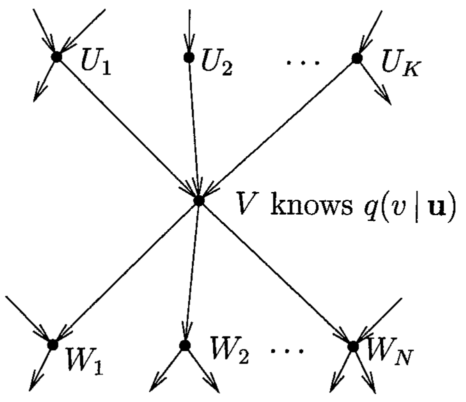


Figure 4.2 The local environment of a vertex V.

4.1.2 Pearl's Belief Propagation Algorithm

Pearl's belief propagation algorithm is a decentralized "message passing" algorithm, in which there is a processor associated with each vertex of the Bayesian Network G. Each processor can communicate only with its immediate neighbors, which are further divided into its parents and children. The processor associated with a variable V is assumed to know the conditional density function

$$q(v \mid \mathbf{u}) \triangleq \Pr\{V = v \mid U_1 = u_1, U_2 = u_2, \dots, U_K = u_K\},$$

where $U_1, U_2, \dots, U_K$ are the parents of V. If V has no parents, this knowledge degenerates to the variable's marginal density $q(v)$. Thus the local environment of a node V is illustrated as in Fig. 4.2.

When a processor is activated, it reads the messages sent from each of its parents and children, updates its belief based on these messages, and then sends new messages back to its parents and children. That is, each updating involves the following two phases:

✎ **Fusion Phase**

The message a node V receives from its parent U_k, denoted by $\pi_{U_k,V}(u_k)$, is a list of probabilities, one for each possible value $u_k \in \mathcal{F}_{U_k}$. Roughly speaking, $\pi_{U_k,V}(u_k)$ is the probability of the event $U_k = u_k$, given the evidence in the tree

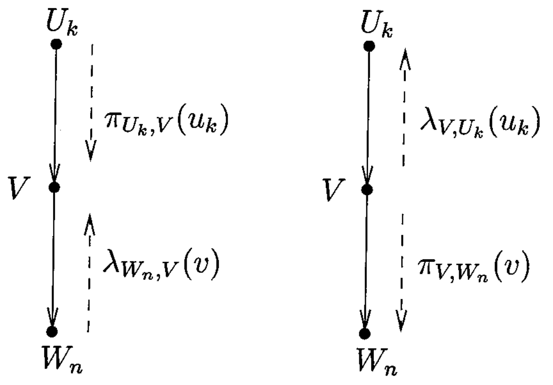


Figure 4.3 The messages in Pearl's belief propagation algorithm.

already "known" to U_k.

The message V receives from its child W_n, denoted by $\lambda_{W_n,V}(v)$, is a list of likelihoods, one for each possible value $v \in \mathcal{F}_V$. Roughly, speaking, $\lambda_{W_n,V}(v)$ is the probability of the evidence W_n "knows", given the event $V = v$.

The processor V then computes the probability of $V = v$ given the evidence from its parents, and the likelihood of $V = v$ given the evidence known to its children, for each value $v \in \mathcal{F}_V$:,

$$\pi_V(v) = \sum_{\mathbf{u}} q(v \mid \mathbf{u}) \prod_{k=1}^{K} \pi_{U_k,V}(u_k), \tag{4.2}$$

$$\lambda_V(v) = \prod_{n=1}^{N} \lambda_{W_n,V}(v). \tag{4.3}$$

For the special cases where (a) V has no parents then $\pi_V(v) = q(v)$; (b) V has no children, then $\lambda_V(v) = 1$. These two quantities can then be combined to obtain the updated belief about the variable V:

$$\forall v \in \mathcal{F}_V, \qquad \mathcal{B}_V(v) = \blacktriangle_v \, \pi_V(v)\lambda_V(v). \tag{4.4}$$

✎ **Fission Phase**

To keep the algorithm going, V should then pass to its parents messages similar to those it receives from its children, and to its children messages similar to those it receives from it parents. More specifically, the message V passes to its child W_n, denoted by $\pi_{V,W_n}(v)$, is a list of probabilities, one for each value $v \in \mathcal{F}_V$. Roughly speaking, $\pi_{V,W_n}(v)$ is the probability of the event $V = v$, given the evidence in the tree already known to V, which now includes any new evidence which may have been contained in the incoming messages. The updating rule is, $\forall n \in \{1, 2, \ldots, N\}$ and $v \in \mathcal{F}_V$,

$$\pi_{V,W_n}(v) = \blacktriangle_v \frac{\mathcal{B}_V(v)}{\lambda_{W_n,V}(v)}. \tag{4.5}$$

Similarly, the message V passes to its parent U_k, denoted by $\lambda_{V,U_k}(u_k)$, is the list of probabilities of the evidence it now knows about, given the event $U_k = u_k$ for each possible value $u_k \in \mathcal{F}_{U_k}$. The updating rule is, $\forall k \in \{1, 2, \dots, K\}$ and $b \in \mathcal{F}_{U_k}$,

$$\lambda_{V,U_k}(U_k = b) = \blacktriangle_b \sum_{\mathbf{u}:u_k=b} \left[\sum_{v} \lambda_V(v) q(v \,|\, \mathbf{u}) \right] \prod_{\substack{j=1 \\ j\neq k}}^{K} \pi_{U_j,V}(u_j). \tag{4.6}$$

The algorithm is initialized by setting all $\lambda_{V,U}(u)$ to 1 unless the corresponding variable V has been observed to be v_0 in which case $\lambda_{V,U}(u) = q(v_0 \,|\, u)$. If V has no parent, then $\pi_V(v)$ is initialized to $q(v)$ unless a V has been observed to be v_0 in which case $\pi_V(v) = \delta(v, v_0)$, where $\delta(x, w)$ is Dirac's delta function. A node can be activated only if all of its incoming messages exist. Otherwise the order of node activation is arbitrary. If the graph is a tree, the algorithm terminates after a number of iterations equal to the diameter of the tree when no change in the messages occurs if any of the nodes is activated. Each node then has correctly computed its belief, i.e., the probability of the associated random variable, conditioned on all of the evidence in the tree.

4.2 General Optimal Symbol Decision Rules

General optimal symbol decision rules for both conventional systematic codes and parallel concatenated (turbo) codes are defined in this section. The decision rule for conventional systematic codes is also shown to be consistent with the belief propagation algorithm.

4.2.1 Conventional Systematic Codes

Let $\mathbf{U} = (U_1, U_2, \dots, U_K)$ be a K-dimensional random vector of independent, but not necessarily equiprobable nor identically distributed, symbols from an alphabet $\mathcal{F}$ with *a priori* probabilities $\Pr\{U_i = u_i\} = \pi_i(u_i)$, for $u_i \in \mathcal{F}$. An (N, K) systematic

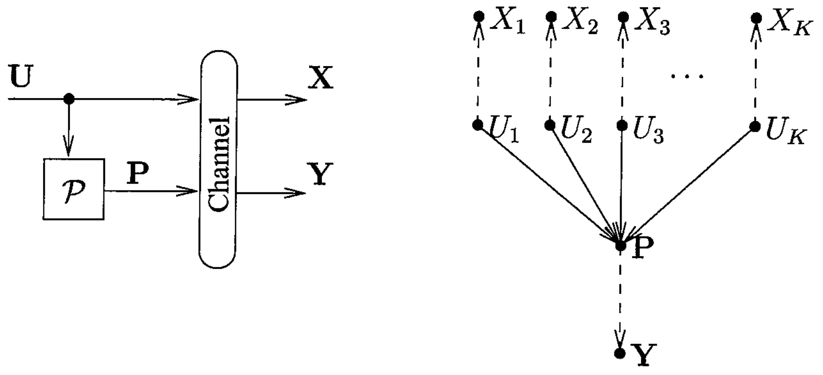


Figure 4.4 System model and the corresponding Bayesian network representation of a systematic code.

encoder $\mathcal{C}$ takes $\mathbf{U}$ as input and produces a codeword of the form

$$\mathbf{C} = \mathcal{C}(\mathbf{U}) = (\mathbf{U}, \mathbf{P}) = (\mathbf{U}, \mathcal{P}(\mathbf{U})),$$

where $\mathbf{U}$ is systematic part and $\mathbf{P}$, a $(N-K)$-dimensional vector, is the parity part of the codeword. The codeword is then transmitted over an unreliable memoryless channel with transition probability $q(r \mid c)$ that a symbol c is received as r. Suppose the codeword is received as $\mathbf{R} = (\mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ is the portion of $\mathbf{R}$ corresponding to the systematic symbols $\mathbf{U}$, and $\mathbf{Y}$ is portion corresponding to the parity symbols $\mathbf{P}$, as shown in Fig. 4.4. Because the encoding process is deterministic, the conditional density factors as

$$\begin{aligned} q(\mathbf{r} \mid \mathbf{u}) &= q(\mathbf{r} \mid \mathbf{c}) = q(\mathbf{x}, \mathbf{y} \mid \mathbf{u}, \mathbf{p}) = q(\mathbf{x} \mid \mathbf{u})\, q(\mathbf{y} \mid \mathbf{u}, \mathbf{p}) \\ &= \left[\prod_{j=1}^{K} q(x_j \mid u_j)\right] Q(\mathbf{y} \mid \mathbf{u}), \end{aligned} \tag{4.7}$$

where $\mathbf{r}, \mathbf{u}, \mathbf{c}, \mathbf{x}, \mathbf{y}, \mathbf{p}$ are particular realizations of the random vectors $\mathbf{R}, \mathbf{U}, \mathbf{C}, \mathbf{X}, \mathbf{Y}, \mathbf{P}$, and $Q(\mathbf{y} \mid \mathbf{u})$ is the overall conditional density for $\mathbf{Y} = \mathbf{y}$ given that the input to the encoder is $\mathbf{u}$.

The optimal symbol decision rule minimizes the probability of inferring an incorrect value for individual symbol U_i based on the observation of $\mathbf{R} = \mathbf{r}$ [74]. This rule concludes

$$U_i = b \quad \text{if} \quad \mathcal{B}_i(b) \geq \mathcal{B}_i(u_i),\ \forall u_i \in \mathcal{F}, \tag{4.8}$$

where the belief $\mathcal{B}_i(u_i) \triangleq \Pr\{U_i = u_i \,|\, \mathbf{R} = \mathbf{r}\}$. The following lemma shows that the belief $\mathcal{B}_i(u_i)$ factors in a very special way.

Lemma 4.1

$$\mathcal{B}_i(b) = \blacktriangle_b\, \pi_i(b) q(x_i \,|\, b) \sum_{\mathbf{u}:u_i=b} Q(\mathbf{y} \,|\, \mathbf{u}) \prod_{\substack{j=1\\ j\neq i}}^{K} \pi_j(u_j) q(x_j \,|\, u_j). \tag{4.9}$$

Proof

$$\begin{aligned}
\Pr\{U_i = b \,|\, \mathbf{R} = \mathbf{r}\} &= \blacktriangle_b \Pr\{\mathbf{R} = \mathbf{r}, U_i = b\} \\
&= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} \Pr\{\mathbf{R} = \mathbf{r}, \mathbf{U} = \mathbf{u}\} \\
&= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} q(\mathbf{r} \,|\, \mathbf{u}) \Pr\{\mathbf{U} = \mathbf{u}\} \\
&= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} Q(\mathbf{y} \,|\, \mathbf{u}) \prod_{j=1}^{K} q(x_i \,|\, u_i) \Pr\{\mathbf{U} = \mathbf{u}\} \qquad \text{by (4.7)} \\
&= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} Q(\mathbf{y} \,|\, \mathbf{u}) \prod_{j=1}^{K} \pi_j(u_j) q(x_i \,|\, u_i).
\end{aligned}$$

Q.E.D.

Just as the decomposition presented in Chapter 2, the belief $\mathcal{B}_i(b)$ is the product of three terms: the systematic evidence $q(x_i \,|\, b)$, the *a priori* belief $\pi_i(b)$, and the extrinsic term

$$\omega_i(b) \triangleq \sum_{\mathbf{u}:u_i=b} Q(\mathbf{y} \,|\, \mathbf{u}) \prod_{\substack{j=1\\ j\neq i}}^{K} q(x_j \,|\, u_j) \pi_j(u_j) \tag{4.10}$$

that contains information about the structure of the code and other evidence.

The corresponding Bayesian network representation for the system is also shown in Fig. 4.4. Since X_i and $\mathbf{Y}$ are terminal observation nodes (dummy nodes [58]), all meaningful message passing happens between X_i and $\mathbf{P}$ nodes. We have thus marked the edges from U_i to X_i and $\mathbf{P}$ to $\mathbf{Y}$ with dashed lines to indicate this.

Proposition 4.2 *For conventional systematic codes, where the corresponding Bayesian network representation forms a tree, the belief propagation algorithm and the optimal symbol decision (4.9) are consistent.*

Proof The following quantities are initialized and also fixed at the corresponding values:

$$\begin{aligned}\pi_{U_i}(u_i) &= \pi_i(u_i),\\ \lambda_{X_i,U_i}(u_i) &= q(x_i \,|\, u_i),\\ \lambda_{\mathbf{Y},\mathbf{P}}(\mathbf{p}) &= q(\mathbf{y} \,|\, \mathbf{p}).\end{aligned}$$

Since $\lambda_{\mathbf{P},U_i}(u_i)$ is initialized to 1, U_i nodes can be activated to perform the following computations:

$$\begin{aligned}\lambda_{U_i}(u_i) &= q(x_i \,|\, u_i),\\ \mathcal{B}_{U_i}(u_i) &= \blacktriangle_{u_i}\pi_i(u_i)q(x_i \,|\, u_i),\\ \pi_{U_i,\mathbf{P}}(u_i) &= \blacktriangle_{u_i}\pi_i(u_i)q(x_i \,|\, u_i).\end{aligned} \tag{4.11}$$

The $\mathbf{P}$ node then has all the necessary messages and is activated. Since $\lambda_{\mathbf{P}}(\mathbf{p}) = \lambda_{\mathbf{Y},\mathbf{P}}(\mathbf{p}) = q(\mathbf{y} \,|\, \mathbf{p})$ and the encoding process is deterministic, (4.6) becomes

$$\begin{aligned}\lambda_{\mathbf{P},U_i}(U_i = b) &= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} \lambda_{\mathbf{P}}(\mathbf{P} = \mathcal{P}(\mathbf{u})) \prod_{\substack{j=1\\ j\neq i}}^{K} \pi_{U_j,\mathbf{P}}(u_j)\\ &= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} Q(\mathbf{y} \,|\, \mathbf{u}) \prod_{\substack{j=1\\ j\neq i}}^{K} \pi_j(u_j)q(x_j \,|\, u_j).\end{aligned} \tag{4.12}$$

Therefore,

$$\mathcal{B}_{U_i}(u_i) = \blacktriangle_{u_i} \pi_i(u_i) q(x_i \mid u_i) \sum_{\mathbf{u}:u_i=b} Q(\mathbf{y} \mid \mathbf{u}) \prod_{\substack{j=1 \\ j \neq i}}^{K} \pi_j(u_j) q(x_j \mid u_j),$$

which is the same as (4.9). Note that the algorithm terminates at this point since the new $\pi_{U_i,\mathbf{P}}(u_i)$ remains unchanged. **Q.E.D.**

Some important insights about (4.9) can be learned from the proof. First, the effect of the systematic evidence $q(x_i \mid u_i)$ is to change the *a priori* distribution of U_i from $\pi_i(u_i)$ to $\pi_i(u_i)q(x_i \mid u_i)$, evidently shown in (4.11). Second, the mysterious extrinsic information, as defined in (4.10) and also in Chapter 2, is actually $\lambda_{\mathbf{P},U_i}(u_i = b)$, the likelihood of U_i computed by node $\mathbf{P}$ based on the evidence it observes from $\mathbf{Y}$ and other systematic nodes U_j, $j \neq i$.

4.2.2 General Turbo Codes

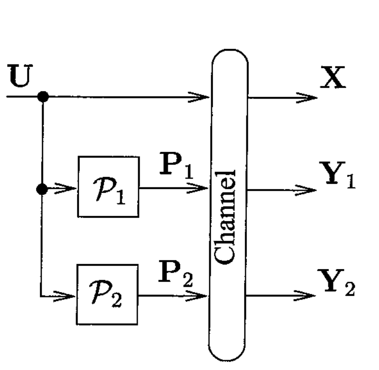
Suppose the input sequence has the same characteristics as defined in the previous section and two systematic codes with parameters (N_1, K) and (N_2, K) are available. The component codes are not restricted to be convolutional codes and no relationship between their code lengths is imposed. Let the encoding processes be defined as:

$$\mathbf{C}_1 = \mathcal{C}_1(\mathbf{U}) = (\mathbf{U}, \mathbf{P}_1) = (\mathbf{U}, \mathcal{P}_1(\mathbf{U})),$$
$$\mathbf{C}_2 = \mathcal{C}_2(\mathbf{U}) = (\mathbf{U}, \mathbf{P}_2) = (\mathbf{U}, \mathcal{P}_2(\mathbf{U})).$$

A turbo code based on C_1 and C_2 is the parallel concatenation of their codewords, i.e.,

$$\mathbf{C} = \mathcal{T}(\mathbf{U}) = (\mathbf{U}, \mathcal{P}_1(\mathbf{U}), \mathcal{P}_2(\mathbf{U})) = (\mathbf{U}, \mathbf{P}_1, \mathbf{P}_2),$$

as shown in Fig. 4.5. Suppose the three parts of the codeword are received as $\mathbf{R} = (\mathbf{X}, \mathbf{Y}_1, \mathbf{Y}_2)$, respectively. Since the channel is memoryless and the encoding process



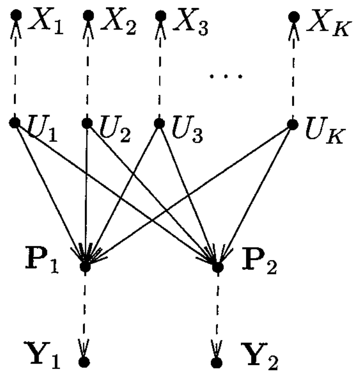


Figure 4.5 System model and the corresponding Bayesian network representation of a general turbo code.

is deterministic, we have

$$\begin{aligned} q(\mathbf{r} \,|\, \mathbf{u}) &= q(\mathbf{r} \,|\, \mathbf{c}) = q(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2 \,|\, \mathbf{u}, \mathbf{p}_1, \mathbf{p}_2) \\ &= q(\mathbf{x} \,|\, \mathbf{u})\, q(\mathbf{y}_1 \,|\, \mathbf{u}, \mathbf{p}_1)\, q(\mathbf{y}_2 \,|\, \mathbf{u}, \mathbf{p}_2) \\ &= \left[\prod_{j=1}^{K} q(x_j \,|\, u_j) \right] Q_1(\mathbf{y}_1 \,|\, \mathbf{u})\, Q_2(\mathbf{y}_2 \,|\, \mathbf{u}), \end{aligned} \tag{4.13}$$

where $Q_1(\mathbf{y}_1 \,|\, \mathbf{u})$ and $Q_2(\mathbf{y}_2 \,|\, \mathbf{u})$ are the overall conditional density for $\mathbf{Y}_1 = \mathbf{y}_1$ and $\mathbf{Y}_2 = \mathbf{y}_2$, respectively, given that the input to the encoder is $\mathbf{u}$. From Lemma 4.1 and (4.13), the optimal symbol decision for a turbo code is based on the beliefs

$$\mathcal{B}_i(b) = \blacktriangle_b q(x_i \,|\, b) \pi_i(b) \sum_{\mathbf{u}: u_i = b} Q_1(\mathbf{y}_1 \,|\, \mathbf{u})\, Q_2(\mathbf{y}_2 \,|\, \mathbf{u}) \prod_{\substack{j=1 \\ j \neq i}}^{K} \pi_j(u_j) q(x_j \,|\, u_j). \tag{4.14}$$

4.3 Turbo Decoding Is Belief Propagation

A general abstract version of the turbo decoding algorithm is defined in this section. The algorithm is then shown to be equivalent to belief propagation with a specific node activation order.

4.3.1 The Turbo Decoding Algorithm

To motivate turbo decoding, it suffices to observe the excessive complexity of (4.14) since the encoders are usually cascaded with pseudo-random interleavers of large size. The algorithm combines the output of "turbo decision modules" (TDM) to approximate the optimal decisions. Taking the same input as the optimal symbol decision, $Q(\mathbf{y} \mid \mathbf{u})$ and $\pi_i(u_i)q(x_i \mid u_i)$, and an additional input, the old extrinsic information $\omega_i^{(T)}(u_i)$, a TDM computes an updated extrinsic information as

$$\begin{aligned}\omega_i^{(T+1)}(u_i) &= \mathrm{TDM}\left(\left\{\omega_i^{(T)}(u_i)\right\}, Q(\mathbf{y} \mid \mathbf{u}), \{\pi_i(u_i)q(x_i \mid u_i)\}\right) \\ &\triangleq \blacktriangle_{u_i} \sum_{\mathbf{u}:u_i=b} Q(\mathbf{y} \mid \mathbf{u}) \prod_{\substack{j=1 \\ j\neq i}}^{K} \pi_i(u_i)q(x_i \mid u_i)\omega_j^{(T)}(u_j). \qquad (4.15)\end{aligned}$$

The algorithm is initialized by setting an unbiased extrinsic information:

$$\omega_i^{(0)}(u_i) = 1, \forall i \in \{1, 2, \ldots, K\}, u_i \in \mathcal{F}. \qquad (4.16)$$

The updating rules are

$$\omega_i^{(2T-1)}(u_i) = \mathrm{TDM}\left(\left\{\omega_i^{(2T-2)}(u_i)\right\}, Q_1(\mathbf{y}_1 \mid \mathbf{u}), \{\pi_i(u_i)q(x_i \mid u_i)\}\right), \qquad (4.17)$$

$$\omega_i^{(2T)}(u_i) = \mathrm{TDM}\left(\left\{\omega_i^{(2T-1)}(u_i)\right\}, Q_2(\mathbf{y}_2 \mid \mathbf{u}), \{\pi_i(u_i)q(x_i \mid u_i)\}\right). \qquad (4.18)$$

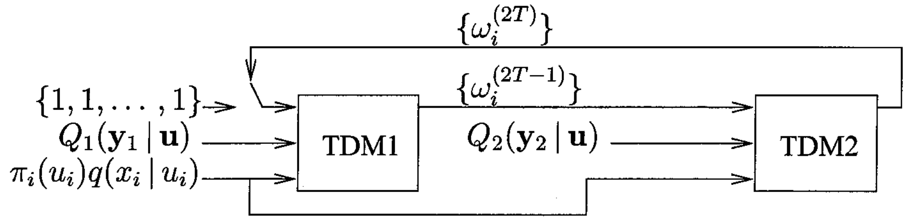

$\{\omega_i^{(2T)}\}$
$\{1,1,\ldots,1\}$
$Q_1(\mathbf{y}_1 \mid \mathbf{u})$
$\pi_i(u_i)q(x_i \mid u_i)$
TDM1
$\{\omega_i^{(2T-1)}\}$
$Q_2(\mathbf{y}_2 \mid \mathbf{u})$
TDM2

Figure 4.6 Turbo decoding modules.

The T-th decision is based on

$$\theta_i^{(T)}(u_i) \triangleq \blacktriangle_{u_i}\pi_i(u_i)q(x_i \,|\, u_i)\omega_i^{(2T-1)}(u_i)\omega_i^{(2T)}(u_i). \tag{4.19}$$

That is, it concludes $U_i = b$ if $\theta_i^{(T)}(b) \geq \theta_i^{(T)}(u_i), \forall u_i \in \mathcal{F}$.

4.3.2 Turbo Decoding as an Instance of Belief Propagation

In the following, we will show formally that if Pearl's belief propagation algorithm is applied to the belief network of a turbo code with a specific activation order, the results is the turbo decoding algorithm described in the previous section.

Theorem 4.3 *The turbo decoding algorithm for the turbo code of Fig. 4.5 is equivalent to the application of belief propagation on the corresponding belief network, also shown in the same figure, with the following node activation order:*

$$\{U_i\}, \mathbf{P}_1, \{U_i\}, \mathbf{P}_2, \{U_i\}, \mathbf{P}_1, \{U_i\}, \mathbf{P}_2, \dots , \tag{4.20}$$

and taking decisions at odd numbers of activation of the U_i nodes.

Proof The following quantities are initialized and also fixed at the corresponding values:

$$\begin{aligned}
\pi_{U_i}(u_i) &= \pi_i(u_i), \\
\lambda_{X_i,U_i}(u_i) &= q(x_i \,|\, u_i), \\
\lambda_{\mathbf{P}_1}(\mathbf{p}_1) &= \lambda_{\mathbf{Y}_1,\mathbf{P}_1}(\mathbf{p}_1) = q(\mathbf{y}_1 \,|\, \mathbf{p}_1), \\
\lambda_{\mathbf{P}_2}(\mathbf{p}_2) &= \lambda_{\mathbf{Y}_2,\mathbf{P}_2}(\mathbf{p}_2) = q(\mathbf{y}_2 \,|\, \mathbf{p}_2).
\end{aligned}$$

Since both $\lambda_{\mathbf{P}_1,U_i}(u_i)$ and $\lambda_{\mathbf{P}_1,U_i}(u_i)$ are initialized to 1, U_i nodes are activated and perform these computations:

$$\lambda_{U_i}(u_i) = q(x_i \,|\, u_i),$$

$$\mathcal{B}_{U_i}(u_i) = \blacktriangle_{u_i}\pi_i(u_i)q(x_i \mid u_i),$$
$$\pi_{U_i,\mathbf{P}_1}(u_i) = \blacktriangle_{u_i}\pi_i(u_i)q(x_i \mid u_i).$$

$\mathbf{P}_1$ node is then activated and gives

$$\begin{aligned}\lambda_{\mathbf{P}_1,U_i}(U_i = b) &= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} \lambda_{\mathbf{P}_1}(\mathbf{P}_1 = \mathcal{P}_1(\mathbf{u})) \prod_{\substack{j=1\\ j\neq i}}^{K} \pi_{U_j,\mathbf{P}_1}(u_j)\\ &= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} Q_1(\mathbf{y}_1 \mid \mathbf{u}) \prod_{\substack{j=1\\ j\neq i}}^{K} \pi_j(u_j)q(x_j \mid u_j)\\ &= \mathrm{TDM}\left(\left\{\omega_i^{(0)}(u_i)\right\}, Q_1(\mathbf{y}_1 \mid \mathbf{u}), \{\pi_i(u_i)q(x_i \mid u_i)\}\right) = \omega_i^{(1)}(u_i).\end{aligned}$$

U_i nodes are activated again and

$$\lambda_{U_i}(u_i) = q(x_i \mid u_i)\omega_i^{(1)}(u_i),$$
$$\mathcal{B}_{U_i}(u_i) = \blacktriangle_{u_i}\pi_i(u_i)q(x_i \mid u_i)\omega_i^{(1)}(u_i),$$
$$\pi_{U_i,\mathbf{P}_2}(u_i) = \blacktriangle_{u_i}\pi_i(u_i)q(x_i \mid u_i)\omega_i^{(1)}(u_i).$$

According to the order, $\mathbf{P}_2$ node is activated and outputs

$$\begin{aligned}\lambda_{\mathbf{P}_2,U_i}(U_i = b) &= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} \lambda_{\mathbf{P}_2}(\mathbf{P}_2 = \mathcal{P}_2(\mathbf{u})) \prod_{\substack{j=1\\ j\neq i}}^{K} \pi_{U_j,\mathbf{P}_2}(u_j)\\ &= \blacktriangle_b \sum_{\mathbf{u}:u_i=b} Q_2(\mathbf{y} \mid \mathbf{u}) \prod_{\substack{j=1\\ j\neq i}}^{K} \pi_j(u_j)q(x_j \mid u_j)\omega_i^{(1)}(u_i)\\ &= \mathrm{TDM}\left(\left\{\omega_i^{(1)}(u_i)\right\}, Q_2(\mathbf{y}_2 \mid \mathbf{u}), \{\pi_i(u_i)q(x_i \mid u_i)\}\right) = \omega_i^{(2)}(u_i).\end{aligned}$$

Now at the U_i nodes, we have

$$\lambda_{U_i}(u_i) = q(x_i \mid u_i)\omega_i^{(1)}(u_i)\omega_i^{(2)}(u_i),$$
$$\mathcal{B}_{U_i}(u_i) = \blacktriangle_{u_i}\pi_i(u_i)q(x_i \mid u_i)\omega_i^{(1)}(u_i)\omega_i^{(2)}(u_i) = \theta_i^{(1)}(u_i),$$

$$\pi_{U_i,\mathbf{P}_1}(u_i) = \blacktriangle_{u_i}\pi_i(u_i)q(x_i \mid u_i)\omega_i^{(2)}(u_i).$$

By repeating the procedure, one can conclude that

$$\lambda_{\mathbf{P}_1,U_i}(U_i = b) = \mathrm{TDM}\left(\left\{\omega_i^{(2T-2)}(u_i)\right\}, Q_1(\mathbf{y}_1 \mid \mathbf{u}), \{\pi_i(u_i)q(x_i \mid u_i)\}\right) = \omega_i^{(2T-1)}(u_i),$$
$$\lambda_{\mathbf{P}_2,U_i}(U_i = b) = \mathrm{TDM}\left(\left\{\omega_i^{(2T-1)}(u_i)\right\}, Q_2(\mathbf{y}_2 \mid \mathbf{u}), \{\pi_i(u_i)q(x_i \mid u_i)\}\right) = \omega_i^{(2T)}(u_i),$$

and at $(2T + 1)$-th activation of the U_i nodes,

$$\mathcal{B}_{U_i}(u_i) = \blacktriangle_{u_i}\pi_i(u_i)q(x_i \mid u_i)\omega_i^{(2T-1)}(u_i)\omega_i^{(2T)}(u_i) = \theta_i^{(T)}(u_i).$$

Q.E.D.

4.4 Concluding Remarks

We have shown that the belief propagation algorithm is consistent with both the optimal symbol decision rule for conventional systematic codes and the turbo decoding algorithm for parallel concatenated codes. Though the chapter discusses turbo codes using two component codes only, codes with three or more parallelly concatenated components have been proposed [25]. If there are M parallel components, the appropriate belief network representation is given in Fig. 4.7. By applying the belief propagation algorithm to this network with the activation order:

$$\{U_i\}, \{\mathbf{P}_j\}, \{U_i\}, \{\mathbf{P}_j\}, \ldots, \tag{4.21}$$

one would obtain the decoding algorithm proposed in [25]. This application is especially fruitful, since it convinces us of the "correctness" of the combining method for extrinsic information from different components proposed in [25]. More decoding algorithms, including those for serially concatenated codes [6] and low-density parity-check codes [33, 48, 49], have been identified as special cases of belief propagation in [50]. Pearl's algorithm thus provides a systematic method for devising suboptimal

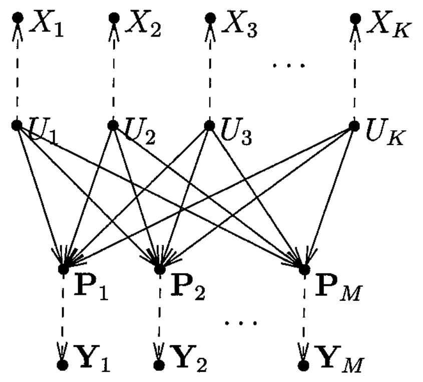


Figure 4.7 Belief network for a multiple turbo code with M parallel components.

iterative decoding algorithms for a wide variety of error-control systems. Since Pearl has proved the validity of his algorithm for loop-free network, there is no guarantee yet that these algorithms will give useful results for loopy networks. However, the great body of experimental work done in the "turbo code & variations" literature strongly suggests that the performance is very close to optimal. We thus conjecture that there are general undiscovered theorems about the performance of belief propagation algorithms on loopy DAG's. These theorems, which may have nothing directly to do with coding or decoding, will show that in some sense the computed beliefs converge with high probability to near-optimum values of the desired beliefs on a class of loopy DAG's that, however, includes the Bayesian network representations of the codes discussed in this chapter. If such theorems exist, they will no doubt find applications in realms far beyond information theory.

Chapter 5
Low-Density Generator Matrix Codes

It is the purpose of this chapter to further examine the applicability of the belief propagation algorithm to decoding error correcting codes. It seems unnecessary for the turbo codes to separate parity bits into subsets as shown in their belief network representations in Fig. 4.5. A class of codes based on low-density generator matrices, where the parity bits are not differentiated, are proposed as a generalization of classical turbo codes. Contrary to the turbo decoding paradigms where sequential processing is accomplished by very powerful central units, the decoding algorithm proposed here is formulated in a distributed parallel form. The decoders can thus enjoy modular pipeline design and the systems, therefore, seem more suitable for practical applications. These new codes can be encoded and decoded in linear time and, for high-rate applications, experimental results further show the performance of these new systems to approach the capacity of the Gaussian channel.

5.1 Introduction and Construction

We have shown in the last chapter that the turbo decoding algorithm for a turbo code is equivalent to the application of belief propagation to the corresponding Bayesian belief network with a specific sequential activation order. Two characteristics about this turbo decoding approach seem immediately unnecessary from this point of view. First, since the belief propagation algorithm allows nodes to be activated in any order, the special activation order (4.20) seems arbitrary and unlikely to be the most efficient. A parallel activation like (4.21) used in the decoding algorithm for multiple turbo codes [25] seems more attractive. Second, as the belief network illustrated in Fig. 4.5, or Fig. 5.1 with the parity check nodes expanded, indicates, the turbo coding approach separates the parity checks into subgroups. Compared to a network that does not differentiate the parity checks such as the one shown in Fig. 5.2, this approach imposes unnecessary restriction on the design of good codes, since belief propagation can be applied on both networks and allows arbitrary activation.

In this chapter, we consider a class of codes whose corresponding Bayesian belief networks have random regular bipartite structure as the network shown in Fig. 5.2.

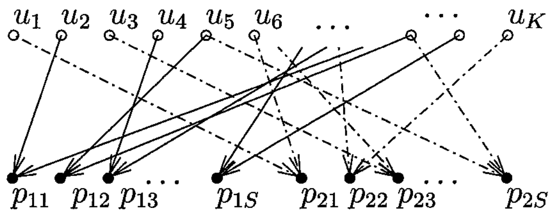

u_1 u_2 u_3 u_4 u_5 u_6 ... ... u_K
p_{11} p_{12} p_{13} ... p_{1S} p_{21} p_{22} p_{23} ... p_{2S}

Figure 5.1 Expanded belief network representation of a turbo code.

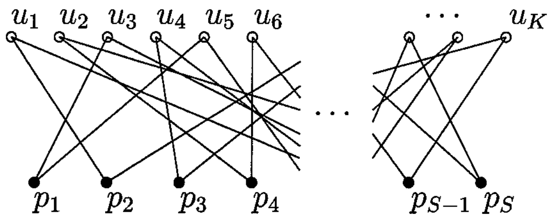

u_1 u_2 u_3 u_4 u_5 u_6 ... u_K
...
p_1 p_2 p_3 p_4 p_{S-1} p_S

Figure 5.2 A random regular bipartite belief network.

The regularity of the network is not essential for our decoding algorithm, but, in practice, it could lead to simplified modular hardware implementation. This representation allows two interpretations of the new codes. First, as already explained, they are generalized turbo codes which treat all the parity checks equally. Second, they can be viewed as multiple turbo codes with a large number of simple parity check component codes.

The first step is to construct a random regular bipartite graph $\mathcal{B}_{KS}$ with the size of the two partite sets as K and S. Call these two subsets the *systematic* nodes and *parity* nodes, and label the vertices in them by $u_1, u_2, \ldots, u_K$ and $p_1, p_2, \ldots, p_S$, respectively, as shown in Fig. 5.2. The degree of each of the systematic nodes is designed to be χ and hence the degree of each of the parity nodes is $\frac{K}{S}\chi$. The construction of such a graph can be accomplished either by trial-and-error with a computer program or by combinatorial methods [20]. Let $G = [I|P]$ be the corresponding generator matrix of the new code, where I is a $K \times K$ identity matrix and P is a $K \times S$ matrix. The entry P_{ij} of the matrix P will be 1 if and only if there is an edge connecting u_i and p_j, and 0 otherwise. This construction thus results in a K input, $N = K + S$ output systematic linear code with random parity bits. This class of codes is termed low-density generator matrix (LDGM) codes since G is sparse by this construction. We can also interpret the bipartite graph as a blueprint for the encoder in the following way:

1. Each of the systematic nodes is a buffer for an input information bit.

2. All the edges are directed from the systematic nodes towards the parity nodes.

3. Each of the parity nodes is in fact a modulo 2 adder.

Since χ is independent of K, the total encoding complexity $K\chi$ is, therefore, linear in the information length K.

The input information bits to the encoder are assumed to take values from $GF(2) = \{0, 1\}$ equiprobably. An encoded bit c is mapped into ± 1 by $(1 - 2c)$, i.e., $0 \to +1$ and $1 \to -1$, and then sent over an additive white Gaussian channel. That is,

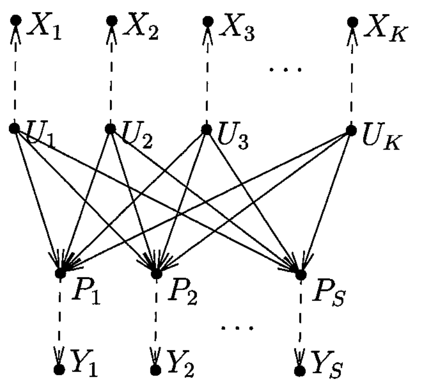


Figure 5.3 Bayesian network for the error-correcting system.

the received signals are assumed to be the sum of the transmitted signals and independent white Gaussian random variables with zero mean and variance $\sigma^2 = N_0/2E_s = N_0/2RE_b$, where E_s is the average energy per symbol, E_b the average energy per bit, R the code rate, and N_0 the single-sided noise power spectral density. Therefore, the channel transition probability $\mathsf{q}(r \mid c)$ for receiving r if c is sent is $\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(r - 1 + 2c)^2\right]$. Denote the received signals corresponding to the systematic bits by x_i, $(i = 1, 2, \ldots, K)$, and the received signals corresponding to the parity bits by y_j, $(j = 1, 2, \ldots, S)$. The complete Bayesian belief network representation of this system is shown in Fig. 5.3. As explained in Chapter 4, since X_i and Y_i are terminal observation nodes (dummy nodes [58]), all meaningful message passing happens between X_i and $\mathbf{P}$ nodes. We have thus marked the edges from U_i to X_i and $\mathbf{P}$ to $\mathbf{Y}$ with dash lines to indicate this.

5.2 Decoding Algorithm

Just as the encoding process is based on the belief network, the decoder is also matched to the network with the application of Pearl's belief propagation algorithm described in Section 4.1. Let $\mathcal{U} = \{1, 2, \ldots, K\}$ be the index set for the systematic nodes, and $\mathcal{P} = \{1, 2, \ldots, S\}$ the index set for the parity nodes. Furthermore, let $\mathcal{U}(j) \subset \mathcal{U}$ be the index set of the systematic nodes that are neighbors of the parity node j, and $\mathcal{P}(i) \subset \mathcal{P}$ the index set of the parity nodes that are neighbors of the systematic node i. The following quantities are initialized and also fixed at the corresponding values:

$$\begin{aligned}\pi_{U_i}(u_i) &= 1,\\ \lambda_{X_i,U_i}(u_i) &= \mathsf{q}(x_i \mid u_i),\\ \lambda_{P_j}(p_j) &= \lambda_{Y_j,P_j}(p_j) = \mathsf{q}(y_j \mid p_j).\end{aligned}$$

Since $\lambda_{P_j,U_i}(u_i)$ is initialized to 1, we have the initial beliefs $\mathcal{B}_{U_i}(u_i) = \blacktriangle_{u_i} \mathsf{q}(x_i \mid u_i)$ according to (4.2)–(4.6). Let $\mathbf{U}_j$ denote the set of variables U_k with $k \in \mathcal{U}(j)$, and $\mathbf{u}_j$ be a set of assignments to $\mathbf{U}_j$. Because the encoding process is deterministic, (4.6) becomes

$$\begin{aligned}\lambda_{P_j,U_i}(U_i = b) &= \blacktriangle_b \sum_{\mathbf{u}_j : u_i = b} \lambda_{P_j}\left(P_j = \bigoplus_{k\in\mathcal{U}(j)} u_k\right) \prod_{\substack{k\in\mathcal{U}(j)\\ k\neq i}} \pi_{U_k,P_j}(u_k)\\ &= \blacktriangle_b \sum_{\mathbf{u}_j : u_i = b} \mathsf{q}\left(y_j \,\middle|\, \bigoplus_{k\in\mathcal{U}(j)} u_k\right) \prod_{\substack{k\in\mathcal{U}(j)\\ k\neq i}} \pi_{U_k,P_j}(u_k).\end{aligned}$$

Therefore, the belief propagation updating rules (4.2)–(4.6) for our LDGM coded system are

$$\pi_{U_i,P_j}(u_i) = \blacktriangle_{u_i} \frac{\mathcal{B}_{U_i}(u_i)}{\lambda_{P_j,U_i}(u_i)}, \tag{5.1}$$

$$\lambda_{P_j,U_i}(U_i = b) = \blacktriangle_b \sum_{\mathbf{u}_j : u_i = b} \mathsf{q}\left(y_j \,\middle|\, \bigoplus_{k\in\mathcal{U}(j)} u_k\right) \prod_{\substack{k\in\mathcal{U}(j)\\ k\neq i}} \pi_{U_k,P_j}(u_k), \tag{5.2}$$

$$\mathcal{B}_{U_i}(u_i) = \blacktriangle_{u_i} \lambda_{U_i}(u_i) = \blacktriangle_{u_i} \mathsf{q}(x_i \mid u_i) \prod_{j\in\mathcal{P}(i)} \lambda_{P_j,U_i}(u_i). \tag{5.3}$$

The most computationally demanding part of these updating rules is obviously (5.2): a naive implementation would require $2^{\frac{K}{S}\chi}$ operations. The exponential complexity can be substantially reduced using the discrete Fourier transform techniques of the next section.

5.2.1 Fourier Transform Techniques for Belief Propagation

Consider a general M-ary case. Let $\mathbf{X} = (X_1, X_2, \ldots, X_M)$ be an M-dimensional random vector of independent, but not necessarily equiprobable nor identically distributed, symbols from a cyclic group $\mathcal{G} = \{0, 1, \ldots, Q-1\}$ with probability distributions $\Pr\{X_i = b\} = \pi_i(b)$, for $b \in \mathcal{G}$. Moreover, let $Y = \sum_{j=1}^{M} X_j$ and $Z_i = \sum_{j \neq i} X_j$. Our problem is to find the probabilities

$$\Lambda_i(a \,|\, b) \triangleq \Pr\{Y = a \,|\, X_i = b\} = \Pr\{Z_i = a - b\}, \tag{5.4}$$

for every $a, b \in \mathcal{F}$. Define the discrete Fourier transform (DFT) [56] of $\pi_i(b)$ by

$$\forall \beta \in \mathcal{G}, \qquad \xi_i(\beta) = \sum_{b=0}^{Q-1} \pi_i(b) W_Q^{\beta b}, \tag{5.5}$$

where $W_Q = \exp\left(j\frac{2\pi}{Q}\right)$, a Q-th root of unity. The inverse transform is given by

$$\pi_i(b) = \frac{1}{Q} \sum_{\beta=0}^{Q-1} \xi_i(\beta) W_Q^{-\beta b}. \tag{5.6}$$

By the convolutional theorem of DFT, the transform of the distribution of Z_i is

$$\zeta_i(\beta) = \prod_{\substack{j=1 \\ j \neq i}}^{M} \xi_j(\beta) = \frac{\eta(\beta)}{\xi_i(\beta)},$$

where $\eta(\beta) = \prod_{j=1}^{M} \xi_j(\beta)$, the transform of the distribution of Y. By the inverse transform, we have

$$\Lambda_i(a \,|\, b) = \frac{1}{Q} \sum_{\beta=0}^{Q-1} \zeta_i(\beta) W_Q^{-\beta(a-b)} = \frac{1}{Q} \sum_{\beta=0}^{Q-1} \frac{\eta(\beta)}{\xi_i(\beta)} W_Q^{-\beta(a-b)}. \tag{5.7}$$

For the special case $\mathcal{G} = GF(2)$, we have $W_Q = -1$ and

$$\xi_i(0) = \pi_i(0) + \pi_i(1) = 1, \quad \Rightarrow \quad \eta(0) = 1,$$

$$\xi_i(1) = \pi_i(0) - \pi_i(1).$$

Therefore, (5.7) becomes,

$$\Pr\left\{ \sum_{j=1}^{M} X_j = a \,\middle|\, X_i = b \right\} = \frac{1}{2}\left[1 + \frac{\eta(1)}{\xi_i(1)}(-1)^{a\oplus b}\right]. \tag{5.8}$$

5.2.2 Further Tricks to Speed Up the Algorithm

Define $\delta_{ji} = \pi_{U_k,P_k}(0) - \pi_{U_k,P_k}(1)$, and let

$$D_j = \prod_{i\in\mathcal{U}(j)} \delta_{ji}, \quad \text{and} \quad \triangle_{ji} = D_j/\delta_{ji}.$$

Using (5.8), (5.2) becomes

$$\lambda_{P_j,U_i}(b) = \blacktriangle_b \left\{ \mathsf{q}(y_j \,|\, 0)\left[1 + (-1)^b \triangle_{ji}\right] + \mathsf{q}(y_j \,|\, 1)\left[1 - (-1)^b \triangle_{ji}\right] \right\}. \tag{5.9}$$

The normalization can be avoided if we operate on likelihood ratios. Hence, let

$$\Theta_j \triangleq \frac{\mathsf{q}(y_j \,|\, 0)}{\mathsf{q}(y_j \,|\, 1)} = \exp\left(\frac{2}{\sigma^2} y_j\right), \tag{5.10}$$

then

$$L_{ji} \triangleq \log \frac{\lambda_{P_j,U_i}(0)}{\lambda_{P_j,U_i}(1)} = \log \frac{(1+\triangle_{ji})\Theta_j + (1-\triangle_{ji})}{(1-\triangle_{ji})\Theta_j + (1+\triangle_{ji})}. \tag{5.11}$$

(5.3) can also be converted into likelihood ratios:

$$B_i \triangleq \log \frac{\mathcal{B}_{U_i}(0)}{\mathcal{B}_{U_i}(1)} = \phi_i + \sum_{j\in\mathcal{P}(i)} L_{ji}, \tag{5.12}$$

where

$$\phi_i \triangleq \log \frac{\mathsf{q}(x_i \,|\, 0)}{\mathsf{q}(x_i \,|\, 1)} = \frac{2}{\sigma^2} x_i. \tag{5.13}$$

Because of (5.1),

$$\begin{aligned}\delta_{ji} &= \pi_{U_k,P_k}(0) - \pi_{U_k,P_k}(1) = \blacktriangle \left[\frac{\mathcal{B}_{U_i}(0)}{\lambda_{P_j,U_i}(0)} - \frac{\mathcal{B}_{U_i}(1)}{\lambda_{P_j,U_i}(1)} \right] \\ &= \frac{\frac{\mathcal{B}_{U_i}(0)}{\lambda_{P_j,U_i}(0)} - \frac{\mathcal{B}_{U_i}(1)}{\lambda_{P_j,U_i}(1)}}{\frac{\mathcal{B}_{U_i}(0)}{\lambda_{P_j,U_i}(0)} + \frac{\mathcal{B}_{U_i}(1)}{\lambda_{P_j,U_i}(1)}} = \frac{\frac{\mathcal{B}_{U_i}(0)}{\mathcal{B}_{U_i}(1)} \Big/ \frac{\lambda_{P_j,U_i}(0)}{\lambda_{P_j,U_i}(1)} - 1}{\frac{\mathcal{B}_{U_i}(0)}{\mathcal{B}_{U_i}(1)} \Big/ \frac{\lambda_{P_j,U_i}(0)}{\lambda_{P_j,U_i}(1)} + 1} = \frac{\exp(B_i - L_{ji}) - 1}{\exp(B_i - L_{ji}) + 1}.\end{aligned}$$

5.2.3 Decoding Algorithm for LDGM Codes

✎ **Initialization**

```
1   for $i \in \mathcal{U}$ do
2       $B_i := \phi_i$;
3       for $j \in \mathcal{P}$ do
4           $L_{ji} := 0$;
5       end
6   end
```

✎ **Updating Rules for Parity Nodes**[1]

```
7   for $j \in \mathcal{P}$ do
8       for $i \in \mathcal{U}(j)$ do
9           $\delta_{ji} := \frac{\exp(B_i - L_{ji}) - 1}{\exp(B_i - L_{ji}) + 1}$;
10      end
11      $D_j := \prod_{i \in \mathcal{U}(j)} \delta_{ji}$;
12      for $i \in \mathcal{U}(j)$ do
13          $\triangle_{ji} := D_j / \delta_{ji}$;
```

[1]The decision rule for parity bits is listed in Section 6.2.2.

```
14          $L_{ji} := \log \frac{(1+\triangle_{ji})\Theta_j + (1-\triangle_{ji})}{(1-\triangle_{ji})\Theta_j + (1+\triangle_{ji})}$;
15      end
16  end
```

✎ **Updating and Decision Rules for Systematic Nodes**

```
17  for $i \in \mathcal{U}$ do
18      $B_i := \phi_i + \sum_{j \in \mathcal{P}(i)} L_{ji}$;
19      $\hat{u}_i := \begin{cases} 0, & \text{if } B_i \geq 0 \\ 1, & \text{if } B_i < 0 \end{cases}$;
20  end
```

5.2.4 Comments about the Algorithm

The most significant characteristic of the proposed algorithm is its highly distributive parallelism, which contrasts to the turbo decoding paradigms where sequential processing is accomplished by very powerful central units [6,7,13,25,40]. More specifically,

- The operations in steps 8–15 can be executed locally at each of the parity nodes. No communication is necessary between them.
- Further parallel processing can be done within each parity node: steps 8–10 and steps 12–15 can each be executed in parallel.
- As with the parity nodes, the operations at each systematic node can be done locally without any exchange of information.

A direct implementation of the algorithm would require, for each iteration, about $5\frac{K}{S}\chi$ multiplications/divisions (MUL/DIV) and $3\frac{K}{S}\chi$ additions/subtractions (ADD/SUB) for each of the parity nodes, and $\frac{K}{S}\chi$ ADD/SUB for each of systematic nodes. The total decoding complexity is thus about $5K\chi$ MUL/DIV and $4K\chi$ ADD/SUB, which is linear in the information length K. The complexity, however, can be greatly reduced as follows:

- The complex operation in step 9 can be avoided by approximating the function with a piece-wise linear function.
- The number of computations in step 14 can be reduced by noticing that

$$L_{ji} \to \begin{cases} \pm\frac{2}{\sigma^2}y_j, & \text{if } \triangle_{ji} \to \pm 1 \\ \pm\log\frac{1+\triangle_{ji}}{1-\triangle_{ji}}, & \text{if } \pm\frac{2}{\sigma^2}y_j \gg 0 \end{cases}.$$

- Though many MUL/DIV are used in the above algorithm, for most cases, they can be replaced by ADD/SUB by converting the appropriate variables to the log domain. Since these considerations depend on specific hardware implementation, details will not be further explored here.

5.3 Simulation Results

The performance of two (1536, 1024) codes with $\chi = 6$ and $\chi = 7$ is shown in Fig. 5.4. It is observed that iterative decoding improves the performance by a large amount. For example, at $E_b/N_0 = 3.5$ dB, the bit error rate (BER) of the systems with $\chi = 7$ is 2×10^{-2} after the first iteration. It is decreased to 8×10^{-5} after four iterations and further reduced to 1.6×10^{-6} after 25 iterations. However, it is also observed from these plots that about ten iterations are enough to obtain most of the performance gains. At BER $= 10^{-5}$, these two coding systems offer gains of more that 6.5 dB over an uncoded system and, in fact, their performance is within 2 dB of the channel capacity with rate 2/3 and binary signaling. These impressive performance improvements are achieved with a relatively short block length of 1024. We expect larger coding gains can be achieved with longer block lengths.

An "error floor" phenomenon similar to those reported for turbo codes [26] can be observed in the performance curves for the $\chi = 6$ code. It has been argued that this

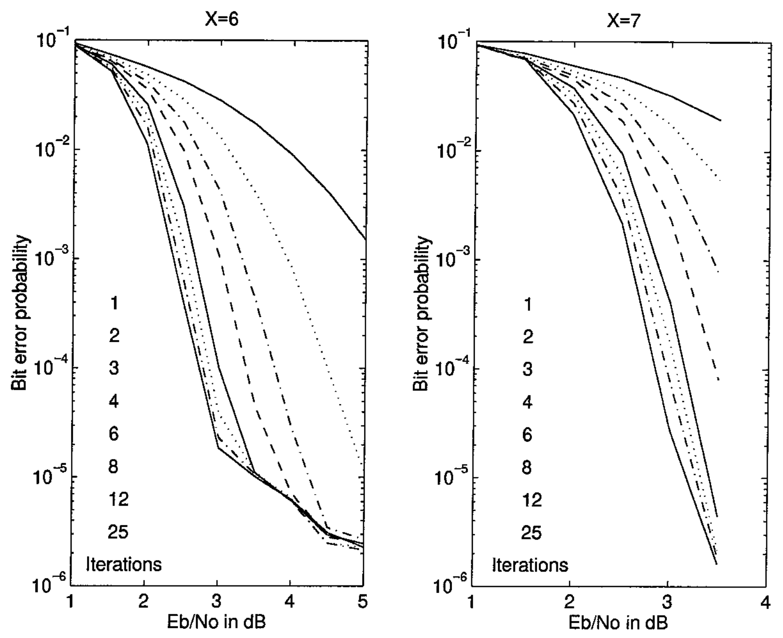


Figure 5.4 Performances of the (1536, 1024) systematic random linear codes with $\chi = 6$ and $\chi = 7$.

floor is caused by the true minimum distance of the code [4]. Since no such floor is observed for the $\chi = 7$ code for BERs down to 10^{-6}, it seems to be a better code than the $\chi = 6$ one in the sense that it has a larger minimum distance or a smaller number of nearest neighbor codewords. As shown in Fig. 5.5, however, the performance of the $\chi = 8$ code is inferior to that of the $\chi = 7$ code. Though there is no guarantee that the $\chi = 8$ code is better, it is suspected that the proposed algorithm might not be able to extract all the power of codes with small cycles such as the $\chi = 8$ code. This might be because of the smaller sizes of girths in the bipartite graphs [33, 34, 48, 49] with these high connection parameters.

K	S	Rate R	χ
1024	16	0.985	3
	32	0.970	4
	64	0.941	5
	128	0.889	4
	256	0.800	5
	512	0.667	7

K	S	Rate R	χ
16384	256	0.985	3
	512	0.970	4
	1024	0.941	5
	2048	0.889	4
	4096	0.800	6
	8192	0.667	8

Table 5.1 Parameters of some low-density generator matrix codes.

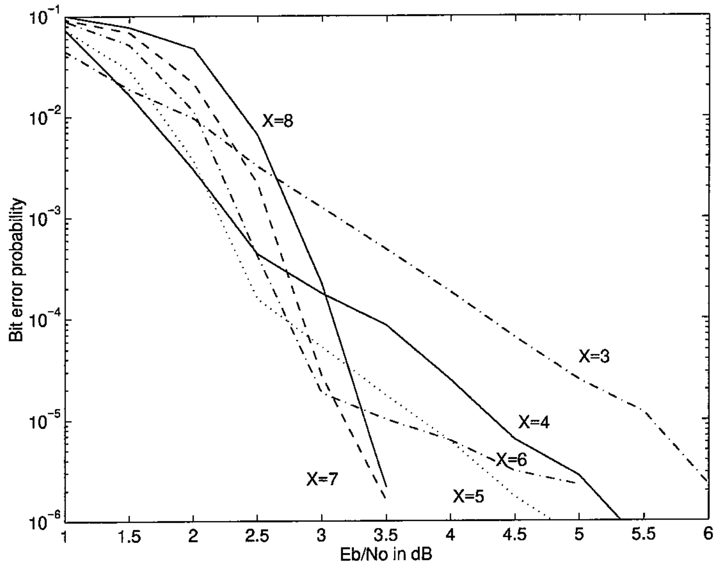


Figure 5.5 Performances of the $(1536, 1024)$ systematic random linear codes with $\chi = 3, 4, 5, 6, 7,$ and 8 after 16 iterations.

More high-rate codes with the parameters listed in Table 5.1 have been constructed and studied. The performance curves of the codes with $K = 1024$ after 16 iterations of decoding are shown in Fig. 5.6. With only 3% redundancy, the $(1056, 1024)$ code achieves an impressive coding gain of 3.5 dB over an uncoded system at BER $= 10^{-6}$. In addition, with a relatively short block length of 1024, the performance of these codes are all within about 2 dB of the channel capacities for their rates. Another distinguishing characteristic of the proposed decoding algorithm is its ability to base its decisions more on the systematic bits than on the parity bits when the signal-to-noise ratio is low. This is exhibited in the figure by the fact that the performance curves are roughly upper-enveloped by the curve for the uncoded systems, especially for very high rate codes.

To further explore the performance of these new codes, we have plotted in Fig. 5.7 the channel capacities of memoryless AWGN channels with equiprobable BPSK, QPSK, and 8PSK signaling given by [72]:

$$C(\Lambda) = \log_2 M - \frac{1}{M} \sum_{i=0}^{M-1} E\left[\log_2 \left(\sum_{j=0}^{M-1} \exp \frac{|\mathbf{s}_i + \mathbf{n} - \mathbf{s}_j|^2 - |\mathbf{n}|^2}{N_0} \right)\right], \tag{5.14}$$

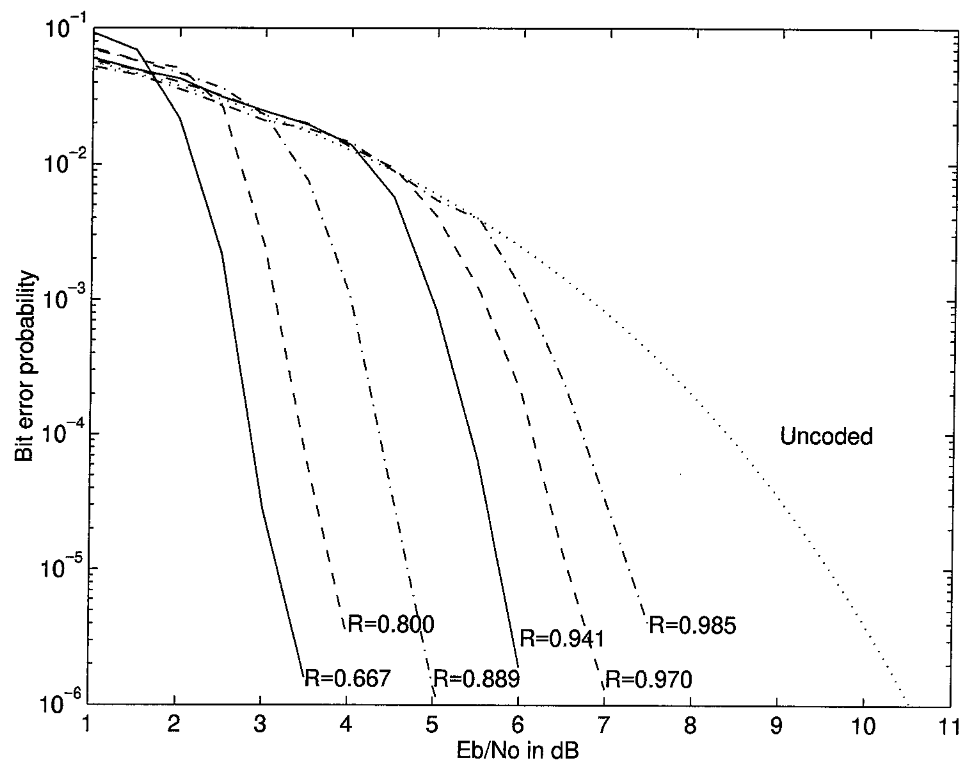


Figure 5.6 Performances of $(1024/R, 1024)$ systematic random linear codes after 16 iterations.

where M is the size of the signaling alphabet, $\mathbf{s}_i$ is a signaling symbol with $E[|\mathbf{s}_i|^2] = 1$, and $\mathbf{n}$ is a Gaussian random variable with zero mean and variance $N_0/2$. The performance of the codes listed in Table 5.1 with $K = 16384$ after 16 iterations of decoding is marked in Fig. 5.7 by cross-plus signs and labeled by "BPSK with LDGM Codes." Their performances is within about 1 dB of capacity, as shown in the figure. The longer length thus contributes about one more dB to the coding gains of the codes with $K = 1024$ listed in Table 5.1. The frequency efficiency can be easily doubled by using QPSK signaling instead of BPSK. These systems are marked in Fig. 5.7 by cross-plus signs and labeled by "QPSK with LDGM Codes." As shown in the figure, they achieve higher coding gains than those of Ungerboeck's trellis coded (TC) 8PSK [72] (marked by circle-plus signs) with marginally lower spectral efficiencies. The combination of QPSK and the $R = 0.941$ code achieves a coding gain higher than even the TC8PSK with a constraint length 16 code and sequential decoding [22], though with a slightly lower spectral efficiency.

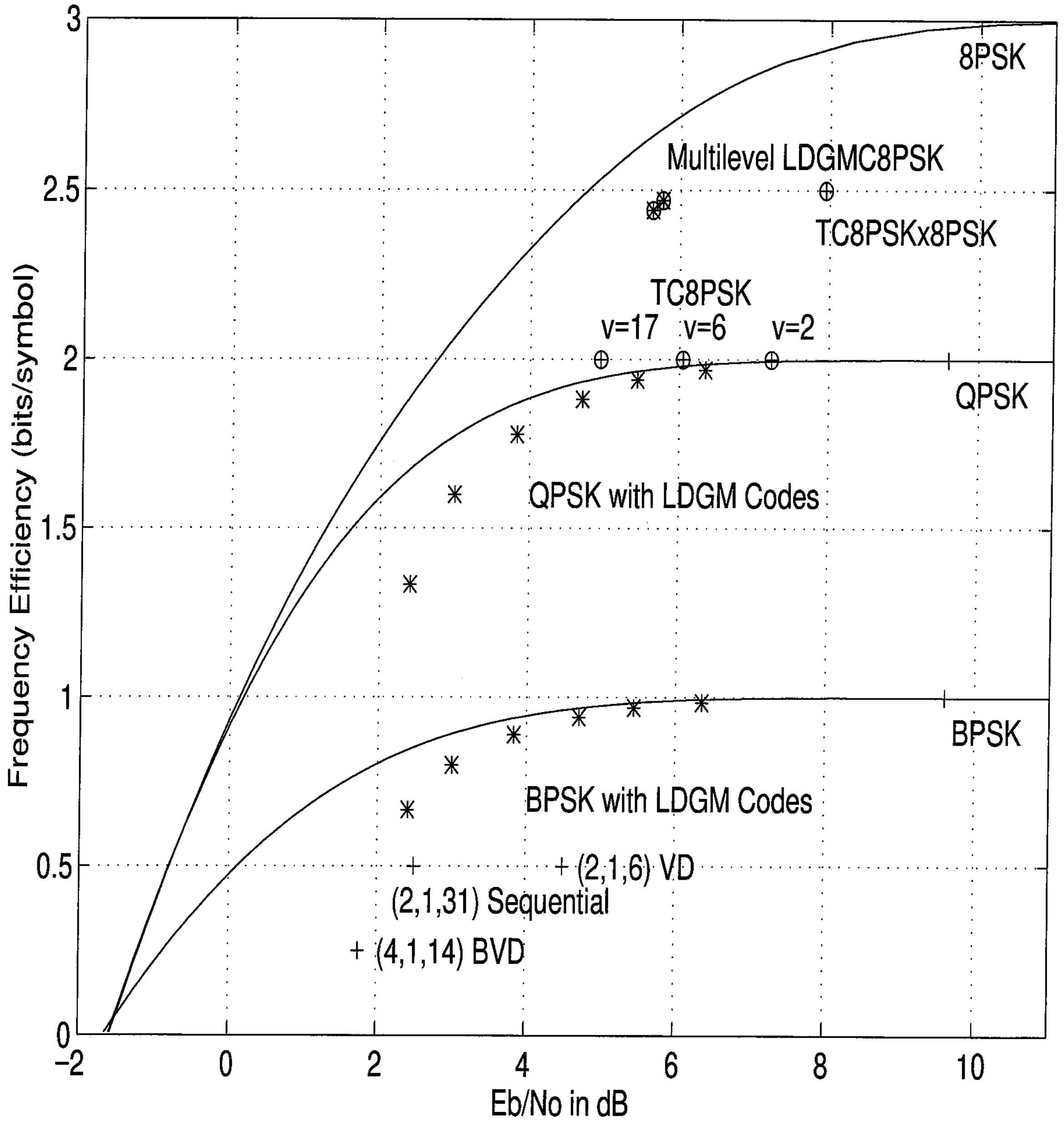


Figure 5.7 Spectral efficiency comparison of codes. (Marks for specific coding systems are based on the code rates and the required E_b/N_0 for BER $= 10^{-5}$.)

As to be discussed in the next chapter, it is also possible to apply these new codes to multilevel coded modulations [12, 42, 66] for even higher spectral efficiency. The performances of two such construction are marked in the figure by circle-cross-plus signs and labeled by "Multilevel LDGMC8PSK." Their frequency efficiencies are about 2.5 bits per symbol. They achieve coding gains that are within about 1 dB of the capacity and are 2 dB higher than that of a multidimensional TC8PSK [59] (labeled by "TC8PSKx8PSK" in Fig. 5.7). Please refer to the next chapter for details.

5.4 Conclusions

A class of high-rate linear binary codes based on low-density generator matrices was proposed. There codes are inherently linear-time encodable. A distributive parallel decoding algorithm of linear complexity based on the belief propagation algorithm was presented. Experimental results showed that performance approaches the channel capacities for binary and multiphase signalings. Compared with low-density parity-check codes [33, 34, 48, 49], the proposed systems achieve similar performance but require lower encoding complexity. This is because, since H is sparse, the generator matrix G is dense and, hence, the encoding process involves $O(N^2)$ additions [48]. For lower rates, the performance of these new codes seems to be not as good as that of the turbo codes [13,25,26,40]. Our systems, however, seem more suitable for physical implementation and practical application because of the distributive parallelism in the decoding algorithm.

Chapter 6

Efficient Multilevel Coded Modulations

To achieve higher spectral efficiency, low-density generator matrix (LDGM) codes introduced in the last chapter are applied to M-ary signaling schemes using multilevel coding methods. An introduction to this technique is given in Section 6.1. The equivalent channel capacities for individual partition levels used in the multilevel coding systems are discussed. Multilevel coded modulations based on LDGM codes and a multistage decoding algorithm are proposed in Section 6.2. Partition rules other than Ungerboeck's maximum intra-set distance criterion are examined in Section 6.3. LDGM codes for individual levels are then selected according to the corresponding equivalent capacities. We show that this approach can be used to devise systems that achieve near channel capacity performance and are also able to provide unequal error protection.

6.1 Multilevel Coding Based on Partitioning

The construction of many channel codes begins with a geometric partitioning of the signal sets [12, 22, 32, 42, 59, 62, 66, 72]. In Section 6.1.1, we present a definition of partitioning based on subsets, which is deliberately made more general than cosets [32] to allow some partitioning methods that have no direct algebraic meanings. The problems with the conventional design rules for multilevel codes are discussed in Section 6.1.2. The equivalent channel capacities for individual partition levels are defined in Section 6.1.2, and serve as new guidelines for designing multilevel codes.

6.1.1 Subset Partitioning

If Λ_1 is a subset of a signal set Λ_0, a *partition* Λ_0/Λ_1 of Λ_0 is defined as the collection of Λ_1 and its non-overlapping cosubsets. We restrict our attention to *regular* partitions where all the cosubsets have the same number of signals and the same channel capacity when regarded as individual modulation schemes. In most cases, this means that all the cosubsets are shifts of each other in the Euclidean space. The members of the partition Λ_0/Λ_1 can thus be denoted by $0, 1, \ldots, |\Lambda_0|/|\Lambda_1| - 1$, where $|\Lambda_l|$ denotes the size of the signal set. For example, in Fig. 6.1, Λ_0 (an 8PSK) is partitioned by

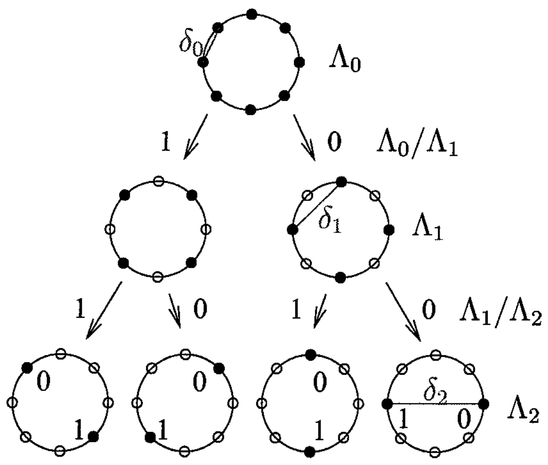


Figure 6.1 Partitioning of 8PSK by the maximum intra-set distance (Ungerboeck) criterion.

Λ_1 (a QPSK) and the partition Λ_0/Λ_1 contains 0 and 1. In this case, the partition is *binary* since it has just two members. The partitioning operation can be applied recursively to form a *partition chain* $\Lambda_0/\Lambda_1/\cdots/\Lambda_L$, where Λ_l is a subset of Λ_{l-1}, $l = 1, 2, \ldots, L$. This partition chain then induces a bijective mapping between Λ_0 and $\Lambda_0/\Lambda_1 \times \Lambda_1/\Lambda_2 \times \cdots \times \Lambda_{L-1}/\Lambda_L \times \Lambda_L$. That is, a symbol $\mathbf{s}_i$ in Λ_0 will be uniquely labeled by $(b_0, b_1, \ldots, b_L)$ iff $\mathbf{s}_i \in b_l$ for all $l = 0, 1, \ldots, L$, where $b_l \in \Lambda_l/\Lambda_{l+1}$ for $l = 0, 1, \ldots, L-1$ and $b_L \in \Lambda_L$. In this notation, we assume b_L is a singleton subset of $\Lambda_L = \{0, 1, \ldots, |\Lambda_L| - 1\}$. For example, the left-most symbol of the 8PSK constellation in Fig. 6.1 is labeled by $(0, 0, 1)$.

6.1.2 Multilevel Codes

Multilevel coding is a systematic method of combining general error correcting codes (binary or Q-ary, block or convolutional) and modulations to form efficient channel coding systems [12, 42, 62, 66]. It is constructed by selecting separate component codes, $C_0, C_1, \ldots, C_L$, to associate with each of the partition levels, Λ_0/Λ_1, Λ_1/Λ_2, $\ldots$, Λ_{L-1}/Λ_L, Λ_L, in the partition chain $\Lambda_0/\Lambda_1/\cdots/\Lambda_L$. A generic encoding structure for multilevel coded modulation is shown in Fig. 6.2.

Traditionally, the component codes in a multilevel coding scheme are selected according to the rule of balanced minimum Euclidean distance. Namely, if the minimum distance of code C_l is d_l and the minimum Euclidean distance within subset Λ_l is δ_l, for $l = 0, 1, \ldots, L$, then the codes are chosen such that the numbers $d_0\delta_0^2, d_1\delta_1^2, \ldots, d_L\delta_L^2$ are as close to each other as possible [12,42,62,66]. This rule is based on the argument

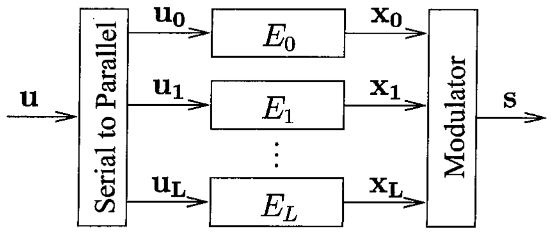


Figure 6.2 Encoding structure for multilevel coded modulation. (E_l is the encoder for code C_l, $l = 0, 1, \ldots, L$.)

that, for high signal-to-noise ratios, the codeword error probability is approximately

$$P_E \gtrsim N(D_m)\, Q(D_m/2\sigma),$$

where $D_m = \sqrt{d_m \delta_m^2}$ and $Q(x) \triangleq \int_x^\infty \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}y^2\right] dy$. Therefore, the asymptotic coding gain is determined by

$$D_{ACG}^2 = \min(d_0\delta_0^2, d_1\delta_1^2, \ldots, d_L\delta_L^2).$$

This rule, however, cannot capture the exponential increase in the numbers of nearest codewords in the higher coding levels[1] because of "multiple representation" of their letters [41, 78]. For example, in Fig. 6.1, the letter 0 of Λ_0/Λ_1 is represented by either one of the four symbols in Λ_1 equiprobably. Since this increase in the number of nearest neighbors degrades performance significantly, this rule cannot serve as a reliable design criterion or a performance indicator. In addition, there are times where this rule cannot even be properly used because the component codes do not have clearly defined minimum distances or cannot be adequately characterized by minimum distances. Turbo codes, product codes, and LDGM codes are examples of such codes. For these codes, the equivalent channel capacities to be introduced in the next section provide better guidelines for selecting component codes.

6.1.3 Equivalent Channel Capacity

Recall that the capacity of a discrete-time memoryless channel with equiprobable input from an M-ary signaling set Λ is defined as [35]

$$C(\Lambda) \triangleq \frac{1}{|\Lambda|} \sum_{\mathbf{s}_i \in \Lambda} \int q(\mathbf{r}|\mathbf{s}_i) \log_2 \frac{q(\mathbf{r}|\mathbf{s}_i)}{f(\mathbf{r})} d\mathbf{r},$$

where $\mathbf{s}_i \in \Lambda$ with $E[|\mathbf{s}_i|^2] = 1$ and $\mathbf{r}$ is the received signal. $q(\mathbf{r}|\mathbf{s}_i)$ is the channel transition probability density function for receiving $\mathbf{r}$ if $\mathbf{s}_i$ is sent, and the density function

[1]By higher coding levels, we refer to those partition levels Λ_l/Λ_{l+1} with smaller l.

for the received signal $\mathbf{r}$ is $f(\mathbf{r}) = \frac{1}{M}\sum_{i=0}^{M-1} q(\mathbf{r}|\mathbf{s}_i)$. For example, the capacity of the additive white Gaussian noise (AWGN) channel is given by (5.14). The equivalent channel capacity for a regular partition Λ_l/Λ_{l+1} is defined formally as [41]

$$C(\Lambda_l/\Lambda_{l+1}) \triangleq \frac{1}{|\Lambda_l/\Lambda_{l+1}|} \sum_{b_i \in \Lambda_l/\Lambda_{l+1}} \int g(\mathbf{r}|b_i) \log_2 \frac{g(\mathbf{r}|b_i)}{f(\mathbf{r})} d\mathbf{r},$$

where

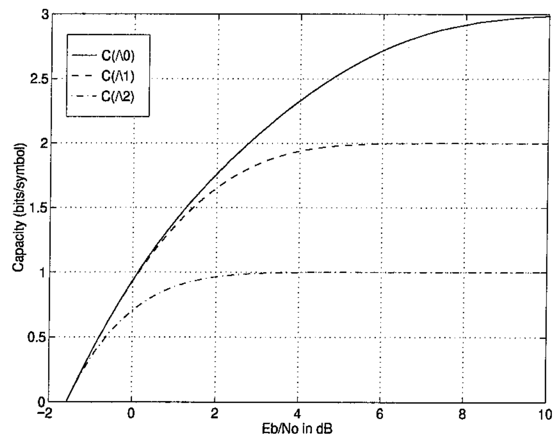
$$g(\mathbf{r}|b_i) \triangleq \frac{1}{|\Lambda_{l+1}|} \sum_{\mathbf{s}_i \in \Lambda_{l+1}+b_i} q(\mathbf{r}|\mathbf{s}_i).$$

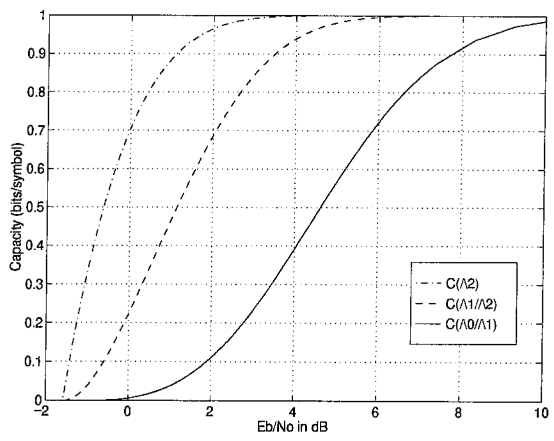
It is shown by Huber and Wachsmann that

$$C(\Lambda_l/\Lambda_{l+1}) = C(\Lambda_l) - C(\Lambda_{l+1}). \tag{6.1}$$

Moreover, for a regular partition chain $\Lambda_0/\Lambda_1/\cdots/\Lambda_L$,

$$C(\Lambda_0) = C(\Lambda_0/\Lambda_1) + C(\Lambda_1/\Lambda_2) + \cdots + C(\Lambda_{L-1}/\Lambda_L) + C(\Lambda_L). \tag{6.2}$$





(A) (B)

Figure 6.3 Capacities of 8PSK partitioned by the maximum intra-set distance (Ungerboeck) criterion. (A) Capacities of the subsets. (B) Capacities of individual partition levels.

That is, the capacity of the channel can be achieved by multilevel codes. As a numerical example, the capacities of the sets Λ_0, Λ_1, and Λ_2 that form the partition chain in Fig. 6.1 are computed by (5.14) with the Monte Carlo method [30] and plotted in Fig. 6.3(A). The corresponding equivalent capacities of the three partition levels $\Lambda_0/\Lambda_1, \Lambda_1/\Lambda_2$, and Λ_2 are computed according to (6.1) and plotted in Fig. 6.3(B).

6.2 Multilevel Coding and Multistage Decoding with LDGM Codes

The performance of low-density generator matrix codes introduced in the last chapter can be quite reliably characterized by the channel capacities at their rates. For example, the required E_b/N_0 to achieve a BER $= 10^{-5}$ for LDGM codes with $K =$ 1024 information bits is about 2 dB of the channel capacity, and about 1 dB for codes with $K = 16384$ information bits. The component codes for the multilevel coded modulation are thus chosen to match the equivalent channel capacities for the corresponding partition levels. Examples for explicit construction and numerical results will be given in the next section. In the remainder of this section, we discuss details about the decoding algorithm.

6.2.1 Multistage Decoding

Maximum likelihood decoding of a multilevel code would require a complexity that is the product of the complexities of its component codes. This formidable complexity can be substantially reduced to the sum of the MLD complexities of the component codes by a suboptimal multistage decoding scheme [12,42,62,66], as shown in Fig. 6.4. Namely, beginning from the highest level, the decoder for the component code C_0 tries to infer the information $\mathbf{u_0}$ based on the received signal $\mathbf{r}$. This decoder also provides the estimated codeword $\hat{\mathbf{x}}_\mathbf{0}$. The decoder for the component code of a lower level infers the information bits of that level using the received signal and all the estimated codewords produced by the higher level decoders. This approach is clearly suboptimal

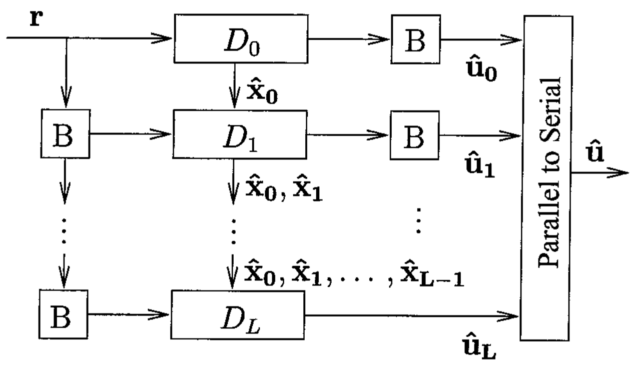


Figure 6.4 Multistage decoder for multilevel coded modulation. (D_l is the decoder for code C_l, $l = 0, 1, \ldots, L$, and B denotes for buffers.)

but not as dismal as some authors had predicted [62]. For practical applications, this algorithm might suffer from error propagation from higher levels to lower levels. However, because of (6.2), the channel capacity is achieved by multilevel coding and the multistage decoding algorithm asymptotically, so long as the partitioning is regular and the component codes are chosen such that their rates are matched to the equivalent channel capacities for the corresponding partition levels.

For codes of finite length, however, there are several variations to this multistage decoding scheme with better performance. For example, the estimates from a lower level could be fed back to the decoders of higher levels if the multistage decoding process is executed twice. It is shown in [12] that for multilevel coded modulation using punctured convolutional codes, this algorithm improves the performance by more than 1 dB. It is also possible for the decoders of higher levels to send soft estimates, instead of hard decisions, to low-level ones in the hope of improving performance. As shown in the next section, we have obtained performance that is so good that these modifications are unlikely to achieve an significant performance improvement.

6.2.2 Belief Propagation Decoding Revisited

Multistage decoding algorithm implies that estimates for the parity-check bits are required. Instead of reencoding the estimated information bit, which might lead to serious error propagation problems, estimates for parity bits can be made directly from their beliefs. Because the encoding process is deterministic, using the notations in Section 5.2, we have

$$\pi_{P_j}(p_j) = \sum_{\substack{\mathbf{u} \\ \bigoplus u_k = p_j}} \prod_{k \in \mathcal{U}(j)} \pi_{U_k,P_j}(u_k) \qquad \text{(from (4.2))} \tag{6.3}$$

$$= \frac{1}{2}\left[1 + (-1)^{p_j} D_j\right] \qquad \text{(by (5.8)).} \tag{6.4}$$

Therefore, we can append the following to the decoding algorithm for LDGM codes in Section 5.2.3:

✎ **Decision Rule for Parity Bits**

```
21  for $j \in \mathcal{P}$ do
22      $\mathbb{P}_j := \frac{1+D_j}{1-D_j}\Theta_j;$
23      $\hat{p}_j := \begin{cases} 0, & \text{if } \mathbb{P}_j \geq 1 \\ 1, & \text{if } \mathbb{P}_j < 1 \end{cases};$
24  end
```

Because of multiple representation of the alphabets, the input to the belief propagation decoding algorithm ϕ_i and Θ_j, defined by (5.13) and (5.10), should be modified as follows. For partition level l, the encoded bits $u_i, p_j \in \Lambda_l/\Lambda_{l+1}$. Hence,

$$\phi_i = \log \frac{\sum_{\mathbf{s}_k \in \Lambda_{l+1}} q(\mathbf{r}_i|\mathbf{s}_k)}{\sum_{\mathbf{s}_k \in \Lambda_{l+1}+1} q(\mathbf{r}_i|\mathbf{s}_k)}, \quad \text{and} \quad \Theta_j = \frac{\sum_{\mathbf{s}_k \in \Lambda_{l+1}} q(\mathbf{r}_j|\mathbf{s}_k)}{\sum_{\mathbf{s}_k \in \Lambda_{l+1}+1} q(\mathbf{r}_j|\mathbf{s}_k)}. \tag{6.5}$$

These metrics are usually approximated by taking the distance between the received signal and the signal point that is closest to it [12]. This approximation, however, is not used in the simulations to be presented in the next section.

6.3 Partitioning Rules for Multilevel Coding

In this section, we discuss the design and characteristics for several partitioning rules.

6.3.1 Partitioning by Ungerboeck's Criterion

Ungerboeck's maximum intra-set distance partitioning criterion is one of the most celebrated innovations in coding theory. The criterion prescribes that the selection of subsets should be made such that the minimum distance within the subsets is maximal [72]. The partitioning of 8PSK in Fig. 6.1 is an example of this criterion. In the following, we shall illustrate the idea of constructing multilevel codes using the equivalent channel capacities.

From Fig. 6.3(A), the capacity of 8PSK is about 2.5 bits per symbol for $E_b/N_0 =$ 4.7 dB. At this signal-to-noise ratio, the equivalent channel capacities of the three partition levels are about 0.5, 0.96 and 1 bits per symbol from Fig. 6.3(B). Two multilevel codes are then constructed accordingly. Both constructions use codes with rate $R_0 = 1/2$ at the 0th level and do not encode the 2nd level ($R_2 = 1$). The first one has length $N = 16896$ and uses a code with rate $R_1 = 32/33$ at the 1st level. The second construction uses length $N = 17408$ and a code with rate $R_1 = 16/17$ as the 1st component code. Sixteen iterations of belief propagation are applied for each of the encoded levels. Their performance is shown in Fig. 6.5 and is also plotted

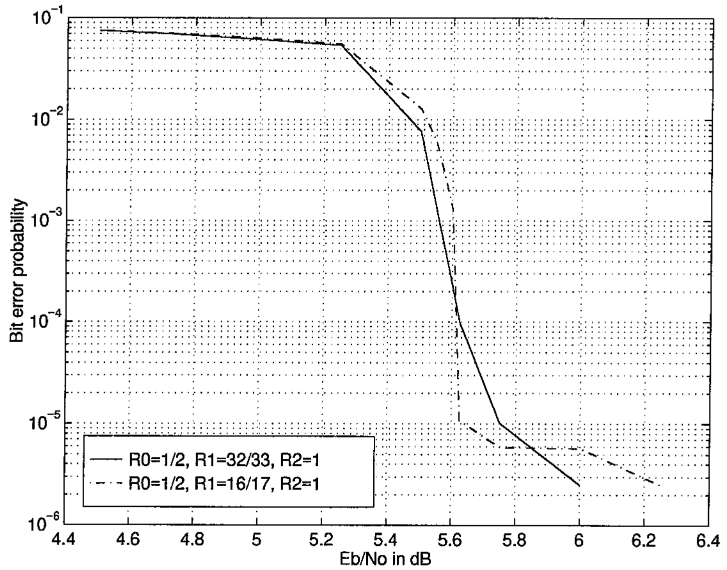


Figure 6.5 Performances of coded 8PSKs with frequency efficiencies $\doteq$ 2.5 bits/symbol. (Channel capacity is at $E_b/N_0 \doteq 4.7$ dB.)

in Fig. 5.7 where it is marked by circle-cross-plus signs and labeled by "Multilevel LDGMC8PSK." The overall performance is within about 1 dB of the channel capacity and is more than 2 dB better than a multidimensional trellis coded 8PSK [59] (labeled by "TC8PSKx8PSK" in Fig. 5.7).

6.3.2 Other Partitioning Criteria

Ungerboeck's partitioning criterion has the effect of maximizing the equivalent channel capacities of lower partition levels since minimum intra-set distances are maximized. This criterion, however, results in very low equivalent capacities for higher levels, as shown in Fig. 6.3(B), since subsets in the higher levels are highly mixed with their cosubsets as shown in Fig. 6.1. Multilevel coding based on this partition criterion might thus require significant decoding latency because most of the information bits are concentrated in the lower levels. Though Ungerboeck's criterion has long been deemed the only way to partition signal sets, this is not the case in light of (6.2). The total capacity is the sum of the equivalent capacities of the levels as long as the partitions are regular. Therefore, we have much more freedom in the ways we partition a signal set to cater to specific system requirements.

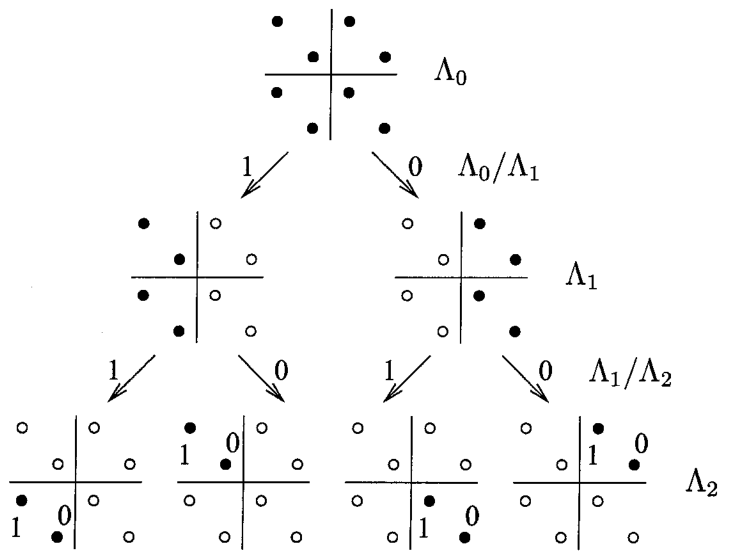


Figure 6.6 Partitioning of 8AMPM by the most separable criterion.

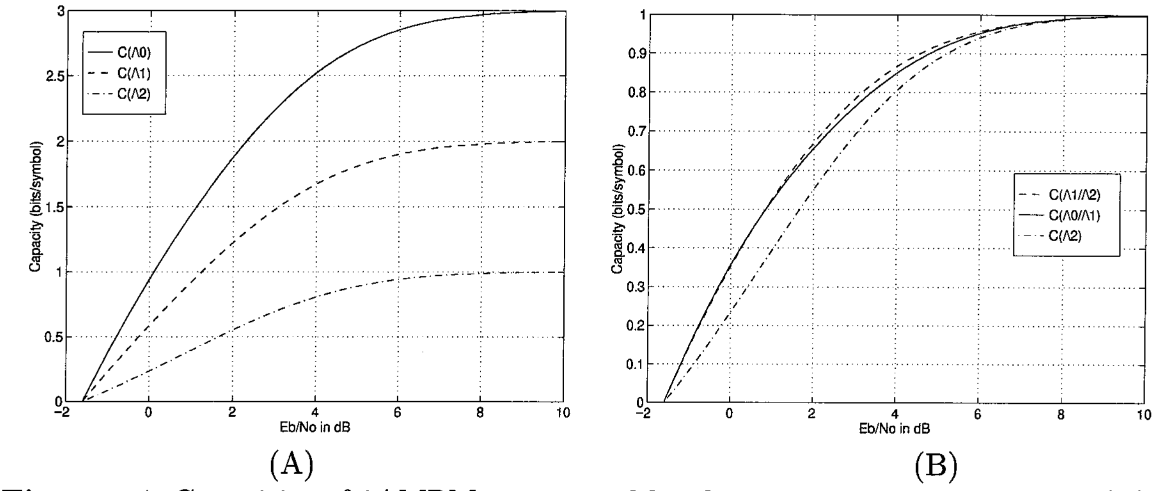


Figure 6.7 Capacities of 8AMPM partitioned by the most separable criterion. (A) Capacities of the subsets. (B) Capacities of individual partition levels.

In Fig. 6.6, the criterion of **most separable partitioning** is demonstrated for the 8AMPM constellation. The subsets are chosen so that the cosubsets are as "separable" as possible. Contrary to Ungerboeck's criterion, this one has the effect of maximizing the equivalent capacities of the higher partition levels at the expense of the capacities of the lower levels. The corresponding equivalent capacities for the partition of 8AMPM in Fig. 6.6 are plotted in Fig. 6.7. In this specific case, the

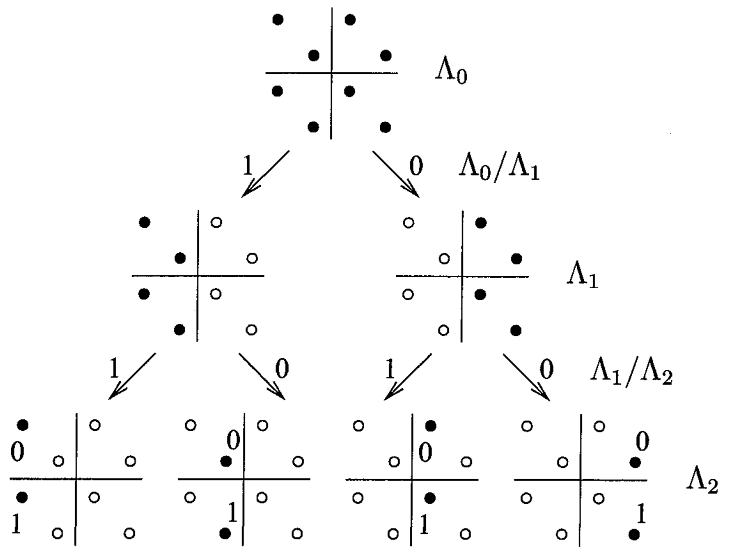


Figure 6.8 Partitioning of 8AMPM by the mixed criterion.

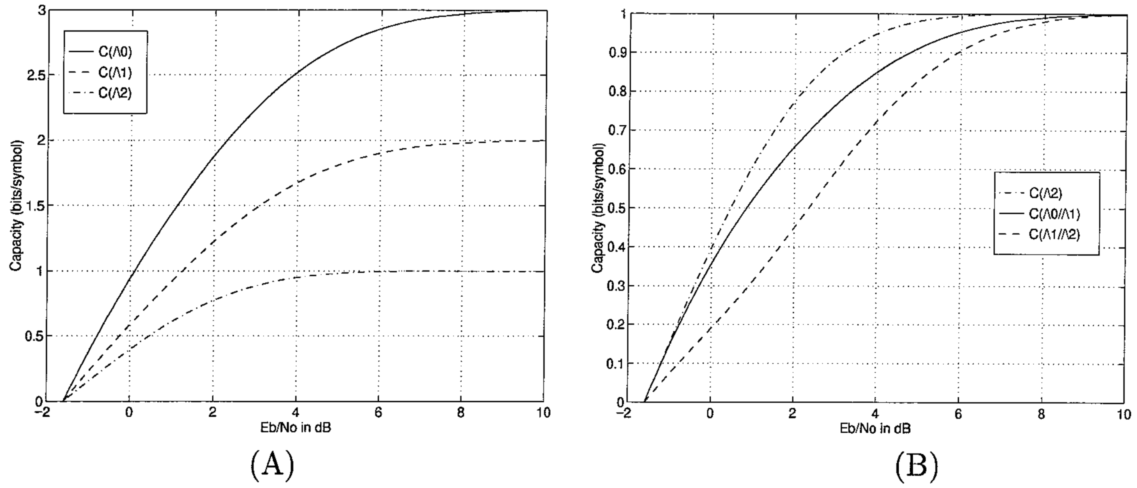


Figure 6.9 Capacities of 8AMPM partitioned by the mixed criterion. (A) Capacities of the subsets. (B) Capacities of individual partition levels.

proposed criterion has resulted in a relatively uniform distribution of capacity among the three partition levels.

The two partitioning criteria can also be mixed: the subsets are chosen to be most separable first and then to maximize intra-set distance. This idea is illustrated in Fig.6.8 and the corresponding equivalent capacities are plotted in Fig. 6.9.

6.3.3 Application: Unequal Error Protection Using One Code

We present here a very interesting application of the most separable partition criterion. It is found in Fig. 6.7, the capacity of 8AMPM is 2 bits/symbol at $E_b/N_0 \doteq 2.4$ dB. At this signal-to-noise ratio, the equivalent capacities $C(\Lambda_0/\Lambda_1)$ and $C(\Lambda_1/\Lambda_2)$ are about 0.7 bit/symbol, with $C(\Lambda_1/\Lambda_2)$ slightly higher, and $C(\Lambda_2) \doteq 0.6$ bit/symbol. A multilevel code can be constructed by using a length $N = 24576$ and rate $R = 2/3$ for all of the three partition levels. The decoding system of this design has significantly lower complexity since only one component decoder is needed to decode all the three coding levels. Comparing the rate and the equivalent capacities, we can expect that the performance in the first level will be the best, then the 0th level, and the second level will be the worst. Simulation results plotted in Fig. 6.10 confirm these expectations. Notice that, though the design is not completely matched to the

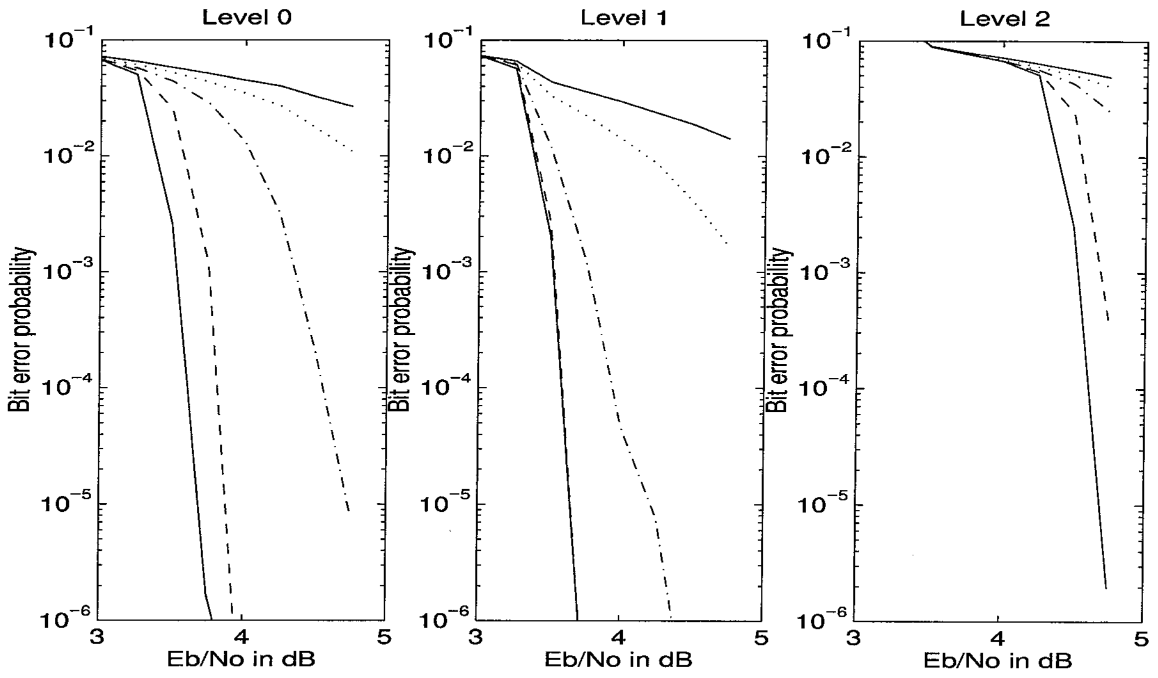


Figure 6.10 Performances of the three coding levels in a coded 8AMPM for unequal error protection. In each of the plots, the curves correspond to the BERs after 1st, 2nd, 4th, 8th, and 16th iterations of decoding. (Frequency efficiency is 2 bits/symbol and channel capacity is at $E_b/N_0 \doteq 2.4$ dB.)

equivalent channel capacities, the performances of the 0th and 1st levels are about $1 \sim 1.2$ dB of the channel capacity.

6.4 Conclusions

We presented a class of efficient multilevel coded modulations based on low-density generator matrix codes. In order to achieve good performance, the rates of the component codes should be selected to match the corresponding equivalent capacities for the partition levels. The component codes for each partition level are decoded by the belief propagation algorithm. Hard decisions are then forwarded to the lower levels. The design and characteristics for several partitioning rules are discussed. Simulation results show these systems achieve performance within 1 dB of the channel capacity and provide flexible error protection capability.

Chapter 7
Conclusions

This thesis examines the theory and application of multi-stage iterative decoding with exchange of soft information to codes with pseudo-random structure. On the practical side, the turbo decoding algorithm is generalized in Chapter 2 to handle multiple-input trellis codes. We showed that turbo codes based on these multiple-input codes achieve marginally better performance than those based on single-input codes. In Chapter 3, alternative high-performance coding systems of low complexity are proposed via the generalized concatenation of convolutional codes. Of central focus is the classical Plotkin $|a \oplus b|b|$ construction and its generalizations. A low-complexity two-iteration decoding algorithm using traditional hard-output Viterbi decoders is proposed. Numerical results show that the new coding systems can achieve comparable and sometimes superior performance to low-complexity turbo codes with similar computational complexity.

On the theory side, we showed in Chapter 4 that the turbo decoding algorithm is special case of Pearl's belief propagation algorithm applying to the appropriate Bayesian belief networks. More decoding algorithms, including those for serially concatenated codes and low-density parity-check codes, can also be shown to be special cases of the belief propagation algorithm. Pearl's algorithm thus provides a systematic method for devising suboptimal iterative decoding algorithms for a wide variety of error-control systems. Since Pearl has proved the validity of his algorithm for loop-free network, there is no guarantee yet that these algorithms will give useful results for loopy networks. However, the great body of experimental work done in the "turbo code & variations" literature strongly suggests that the performance is very close to optimal. We thus conjecture that there are general undiscovered theorems about the performance of belief propagation algorithms on loopy DAG's. These theorems, which may have nothing directly to do with coding or decoding, will show that in some sense the computed beliefs converge with high probability to near-optimum values of the desired beliefs on a class of loopy DAG's that, however, includes the Bayesian network representations of the codes discussed in this chapter. If such theorems exist, they will no doubt find applications in realms far beyond information theory.

In Chapter 5 and 6, we return to problem of constructing good error-correcting codes. Especially, we observed in Chapter 4 that the classical turbo codes make unnecessary differentiation of the parity-check bits and use a specific node activation order for the belief propagation algorithm. We constructed a new class of high-rate codes based on low-density generator matrices. The decoding algorithm based on belief propagation treats all the parity-check bits equally and use a uniform parallel activation order. These codes are then combined with M-ary modulations using multilevel coded modulation techniques to achieve higher spectral efficiency. In all cases, we have constructed systems with flexible error protection capability and performance within 1 dB of the channel capacity.

Bibliography

[1] K. Abdel-Ghaffar, R. J. McEliece and G. Solomon, "Some Partial-Unit-Memory Convolutional Codes," *JPL TDA Progress Report*, Nov. 1991.

[2] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, Mar. 1974.

[3] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Soft-Output Decoding Algorithms in Iterative Decoding of Turbo Codes," *JPL TDA Progress Report*, Feb. 1996.

[4] S. Benedetto and G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes," *IEEE Transactions on Information Theory*, Mar. 1996.

[5] S. Benedetto and G. Montorsi, "Design of Parallel Concatenated Convolutional Codes," *IEEE Transactions on Communications*, May 1996.

[6] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding," *JPL TDA Progress Report*, Aug. 1996.

[7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," *Proceedings of IEEE International Communications Conference '93.*

[8] E. L. Blokh and V. V. Zyablov, "Coding of Generalized Concatenated Codes," *Probl. Predachi Inform.*, no. 3 1974.

[9] B. Bollobas, *Graph Theory: An Introductory Course,* Springer-Verlag, 1979.

[10] S. A. Butman and R. J. McEliece, "The Ultimate Limits of Binary Coding for a Wideband Gaussian Channel," *JPL Deep Space Network Progress Report* 22, 1974.

[11] J. B. Cain, G. C. Clark, Jr., and J. M. Geist, "Punctured Convolutional Codes of Rate $(n-1)/n$ and Simplified Maximum Likelihood Decoding," *IEEE Transactions on Information Theory*, Jan. 1979.

[12] J.-F. Cheng, C.-H. Chuang, and L.-S. Lee, "Low-Complexity Multilevel Coding with Rate-Compatible Punctured Convolutional Codes," *Proceedings of IEEE Global Telecommunications Conference '93.*

[13] J.-F. Cheng and R. J. McEliece, "Unit-Memory Hamming Turbo Codes," *Proceedings of IEEE International Symposium on Information Theory '95.*

[14] J.-F. Cheng and R. J. McEliece, "Superimposed Convolutional Codes," *Proceedings of International Symposium on Communications '95.*

[15] J.-F. Cheng, "Hyperimposed Convolutional Codes," *Proceedings of IEEE International Communications Conference '96.*

[16] J.-F. Cheng and R. J. McEliece, "Some High-Rate Near Capacity Codecs for the Gaussian Channel," *Proc. 34th Allerton Conf. on Comm., Control, and Computing '96.*

[17] J.-F. Cheng, "On the Decoding of Certain Generalized Concatenated Convolutional Codes," submitted to *IEEE Journal on Selected Areas in Communications.*

[18] J.-F. Cheng, "On the Construction of Efficient Multilevel Coded Modulations," to appear in IEEE International Symposium on Information Theory '97.

[19] J.-F. Cheng and R. J. McEliece, "Frequency-Efficient Coding with Low-Density Generator Matrices," in preparation.

[20] F. R. K. Chung, "Constructing Random-Like Graphs," in *Probabilistic Combinatorics and Its Applications*, B. Bollobas, editor, 1991.

[21] G. Cooper, "The Computational Complexity of Probabilistic Inference Using Baysian Belief Networks," *Artificial Intelligence*, pp. 393–4-5, 1990.

[22] D. J. Costello, Jr., L. C. Perez, and F. Wang, "Bandwidth Efficient CCSDS Coding Standards Proposals," *Semi-Annual Status Report*, NASA Grant NAG 5-557, University of Notre Dame, Indiana, May 1992.

[23] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.

[24] D. Divsalar and F. Pollara, "Turbo Codes for Deep-Space Communications," *JPL TDA Progress Report*, Feb. 1995.

[25] D. Divsalar and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications," *JPL TDA Progress Report*, May 1995.

[26] D. Divsalar, S. Dolinar, R. J. McEliece, and F. Pollara, "Transfer Function Bounds on the Performance of Turbo Codes," *JPL TDA Progress Report*, July 1995.

[27] D. Divsalar and F. Pollara, "On the Design of Turbo Codes," *JPL TDA Progress Report*, Nov. 1995.

[28] S. Dolinar and D. Divsalar, "Weight Distributions for Turbo Codes Using Random and Non-Random Interleaving," *JPL TDA Progress Report*, July 1995.

[29] K. Fazel, "Iterative Decoding of Generalized Concatenated Blokh–Zyablov–Codes," *Proceedings of IEEE International Communications Conference '96.*

[30] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer-Verlag, 1996.

[31] G. D. Forney, Jr., *Concatenated Codes*, 1966.

[32] G. D. Forney, Jr., "Coset Codes–Part I: Introduction and Geometrical Classification," and "Coset Codes–Part II: Binary Lattices and Related Codes," *IEEE Transactions on Information Theory*, Sept. 1988.

[33] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, Jan. 1962.

[34] R. G. Gallager, *Low-Density Parity-Check Codes*, The MIT Press, 1963.

[35] R. G. Gallager, *Information Theory and Reliable Communication*, Wiley, 1968.

[36] M. Hattori and Y. Saitoh, "Superimposed Codes Based on Punctured Convolutional Codes," *IEE Electronic Letters*, June 1994.

[37] J. Hagenauer, "Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications," *IEEE Transactions on Communications*, Apr. 1988.

[38] J. Hagenauer and P. Hoeher, "A Viterbi Algorithm with Soft-Decision Outputs and its Applications," *Proceedings of IEEE Global Telecommunications Conference '89.*

[39] J. Hagenauer, N. Seshadri, and C.-E. W. Sundberg, "The Performance of Rate-Compatible Punctured Convolutional Codes for Digital Mobile Radio," *IEEE Transactions on Communications*, July 1990.

[40] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, Mar. 1996.

[41] J. Huber and U. Wachsmann, "Capacities of the Equivalent Channels in Multilevel Coding Schemes," *IEE Electronic Letters*, Mar. 1996.

[42] H. Imai and S. Hirakawa, "A New Multilevel Coding Method Using Error-Correcting Codes," *IEEE Transactions on Information Theory*, May 1997.

[43] F. V. Jensen, *An Introduction to Baysian Networks,* Springer-Verlag, 1996.

[44] O. J. Joeressen, M. Vaupel, and H. Meyr, "High-Speed VLSI Architectures for Soft-Output Viterbi Decoding," *Journal of VLSI Signal Processing*, Oct. 1994.

[45] L.-N. Lee, "Short Unit-Memory Byte-Oriented Binary Convlutional Codes Having Maximal Free Distance," *IEEE Transactions on Information Theory*, May 1976.

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

[47] J. H. Lodge, R. Young, and J. Hagenauer, "Separable MAP 'Filters' for the Decoding of Product and Concatenated Codes," *Proceedings of IEEE International Communications Conference '93.*

[48] D. J. C. MacKay and R. M. Neal, "Good Codes Based on Very Sparse Matrices," in C. Boyd, editor, *Cryptography and Coding, 5th IMA Conference*, Lecture Notes 1025 in Computer Science, Springer, 1995.

[49] D. J. C. MacKay and R. M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," *IEE Electronic Letters*, 1996.

[50] D. J. C. MacKay, R. J. McEliece, and J.-F. Cheng, "Turbo Decoding as an Instance of Peral's 'Belief Propagation' Algorithm," submitted to *IEEE Journal on Selected Areas in Communications.*

[51] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, Ch. 2, 1977.

[52] R. J. McEliece, *The Theory of Information and Coding*, Addison-Wesley, 1977.

[53] R. J. McEliece, E. Rodemich, and J.-F. Cheng, "The Turbo Decision Algorithm," *Proc. 33rd Allerton Conf. on Comm., Control, and Computing '95.*

[54] R. J. McEliece, "On the BCJR Trellis for Linear Block Codes," *IEEE Transactions on Information Theory*, July 1996.

[55] R. J. McEliece, "The Algegraic Theory of Convolutional Codes," in *The Handbook Coding Theory*, W. C. Huffman and V. Pless, editors, Elsevier Science Publishers, 1996.

[56] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing,* Prentice Hall.

[57] J. Pearl, "Fusion, Propagation, and Structuring in Belief Networks," *Artificial Intelligence*, pp. 241–288, 1986.

[58] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, 1988.

[59] S. S. Pietrobon, R. H. Deng, A. Lafanechere, G. Ungerboeck, and D. J. Costello, Jr., "Trellis-Coded Multidimensional Phase Modulation," *IEEE Transactions on Information Theory*, Jan. 1990.

[60] S. S. Pietrobon, "Implementation and Performance of a Serial MAP Decoder for Use in an Iterative Turbo Dedoder," *Proceedings of IEEE International Symposium on Information Theory '95.*

[61] M. Plotkin, "Binary Codes with Specified Minimum Distances," *IEEE Transactions on Information Theory*, Nov. 1960.

[62] G. J. Pottie and D. P. Taylor, "Multilevel Codes Based on Partitioning," *IEEE Transactions on Information Theory*, Jan. 1989.

[63] J. G. Proakis, *Digital Communications*, 2nd ed., McGraw-Hill, 1989.

[64] P. Robertson, "Illuminating the Structure of Code and Decoder of Parallel and Concatenated Recursive Systematic (Turbo) Codes," *Proceedings of IEEE Global Telecommunications Conference '94.*

[65] P. Robertson, E. Villebrun, and P. Hoeher, "A Comparison of Optimal and Sub-optimal MAP Decoding Algorithms Operating in the Log-Domain," *Proceedings of IEEE International Communications Conference '95.*

[66] S. Sayegh, "A Class of Optimum Block Codes in Signal Space," *IEEE Transactions on Communications*, Oct. 1986.

[67] G. Schnabl and M. Bossert, "Soft-Decision Decoding of Reed-Muller Codes as Generalized Multiple Concatenated Codes," *IEEE Transactions on Information Theory*, Jan. 1995.

[68] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technology Journal*, 1948.

[69] S. E. Shimony, "Finding MAPS for Belief Networks Is NP-Hard," *Artificial Intelligence*, pp.399–410, 1994.

[70] M. K. Simon, S. M. Hinedi, and W. C. Lindsey, *Digital Communication Techniques: Signal Design and Detection*, Prentice Hall, 1995.

[71] R. M. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Transactions on Information Theory*, Sep. 1981.

[72] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," *IEEE Transactions on Information Theory*, Jan. 1982.

[73] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Chapter 13, Prentice Hall, 1993.

[74] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Part I*, Wiley, 1968.

[75] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, 1979.

[76] U. Wachsmann and J. Huber, "Power and Bandwidth Efficient Digital Communication Using Turbo Codes in Multilevel Codes," *European Transactions on Telecommunications*, Sep.–Oct. 1995.

[77] N. Wilberg, H.-A. Loeliger and R. Kotter, "Codes and Iterative Decoding on General Graphs," *European Transactions on Telecommunications*, Sep. 1995.

[78] T. Woers and J. Hagenauer, "Multistage Coding and Decoding for a MPSK System," *Proceedings of IEEE Global Telecommunications Conference '90.*

[79] T. Woers and J. Hagenauer, "Iterative Decoding for Multilevel Codes Using Reliability Informaiton," *Proceedings of IEEE Global Telecommunications Conference '92.*

[80] W. W. Wu, D. Haccoun, R. Peile, and Y. Hirata, "Coding for Satellite Communication," *IEEE Journal on Selected Areas in Communications*, May 1987.

[81] V. A. Zinoviev, "Generalized Concatenated Codes," *Probl. Predachi Inform.*, no. 1 1976.