Fall 2001

On performance analysis and implementation issues of iterative decoding for graph based codes

Xuefei Wei
New Jersey Institute of Technology

ABSTRACT

ON PERFORMANCE ANALYSIS AND IMPLEMENTATION ISSUES OF ITERATIVE DECODING FOR GRAPH BASED CODES

by
Xuefei Wei

There is no doubt that long random-like code has the potential to achieve good performance because of its excellent distance spectrum. However, these codes remain useless in practical applications due to the lack of decoders rendering good performance at an acceptable complexity. The invention of turbo code marks a milestone progress in channel coding theory in that it achieves near Shannon limit performance by using an elegant iterative decoding algorithm. This great success stimulated intensive research on long compound codes sharing the same decoding mechanism. Among these long codes are low-density parity-check (LDPC) code and product code, which render brilliant performance. In this work, iterative decoding algorithms for LDPC code and product code are studied in the context of belief propagation.

A large part of this work concerns LDPC code. First the concept of iterative decoding capacity is established in the context of density evolution. Two simulation-based methods approximating decoding capacity are applied to LDPC code. Their effectiveness is evaluated. A suboptimal iterative decoder, Max-Log-MAP algorithm, is also investigated. It has been intensively studied in turbo code but seems to be neglected in LDPC code. The specific density evolution procedure for Max-Log-MAP decoding is developed. The performance of LDPC code with infinite block length is well-predicted using density evolution procedure.

Two implementation issues on iterative decoding of LDPC code are studied. One is the design of a quantized decoder. The other is the influence of mismatched signal-to-noise ratio (SNR) level on decoding performance. The theoretical capacities

of the quantized LDPC decoder, under Log-MAP and Max-Log-MAP algorithms, are derived through discretized density evolution. It is indicated that the key point in designing a quantized decoder is to pick a proper dynamic range. Quantization loss in terms of bit error rate (BER) performance could be kept remarkably low, provided that the dynamic range is chosen wisely. The decoding capacity under fixed SNR offset is obtained. The robustness of LDPC code with practical length is evaluated through simulations. It is found that the amount of SNR offset that can be tolerated depends on the code length.

The remaining part of this dissertation deals with iterative decoding of product code. Two issues on iterative decoding of product code are investigated. One is, improving BER performance by mitigating cycle effects. The other is, parallel decoding structure, which is conceptually better than serial decoding and yields lower decoding latency.

ON PERFORMANCE ANALYSIS AND IMPLEMENTATION ISSUES OF ITERATIVE DECODING FOR GRAPH BASED CODES

by
Xuefei Wei

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

January 2002

APPROVAL PAGE

ON PERFORMANCE ANALYSIS AND IMPLEMENTATION ISSUES OF ITERATIVE DECODING FOR GRAPH BASED CODES

Xuefei Wei

Dr. Ali N. Akansu, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. George Elmasry , Committee Member — Date
Principle Member of Technical Staff, General Dynamics-Communication Systems

---

Dr. Javier Garcia-Frias, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, Univ. of Delaware

---

Dr. Richard Haddad, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Alexander M. Haimovich, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Mahalingam Ramkumar, Committee Member — Date
Chief Technology Officer, AVWAY, IDT Corp. Newark, NJ

BIOGRAPHICAL SKETCH

Author: Xuefei Wei

Degree: Doctor of Philosophy

Date: January 2002

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2002
- Master of Science in Electrical Engineering,
  Xi'an Jiaotong University, Xi'an, P.R.China, 1996
- Bachelor of Science in Electrical Engineering,
  Xi'an Jiaotong University, Xi'an, P.R.China, 1993

Major: Electrical Engineering

Presentations and Publications:

Wei, X. and Akansu, A.N.,
"Quantized decoding for low-density parity-check codes", *to be submitted to IEEE Communications Letters.*

Wei, X. and Akansu, A.N.,
"Decoding capacity of low-density parity-check codes under SNR mismatch", *to be submitted to IEE Electronics Letters.*

Wei, X. and Akansu, A.N.,
"Density Evolution for Low-Density Parity-Check Codes under Max-Log-MAP Decoding", *IEE Electronics Letters*, vol. 37, pp 1225-1226, 2001.

Wei, X. and Akansu, A.N.,
"On parallel iterative decoding of product code", *Proceedings of IEEE Vehicular Technology Conference*, Atlantic City,NJ, 2001.

Wei, X. and Akansu, A.N.,
"Iterative decoding of product code: approaching ML performance", *Proceedings of 5th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando,FL, 2001.

Wei, X. and Akansu, A.N.,
"Iterative Decoding of Product Code: Correlated Extrinsic Information and Its Impact", *Proceedings of the 35th Annual Conference on Information Sciences and Systems*, Baltimore, MD, 2001

Wei, X. and Akansu, A.N.,
"An Improved Iterative Decoding Algorithm for Turbo Product Code", *Proceedings of the 38th Allerton Conference on Communications,Control,and Computing*, Univ. of Illinois at Urbana-Champaign,IL,2000.

In loving memory of my mother, Xiulan Zhu

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my advisor, Dr. Ali N. Akansu. His advice, guidance and insight helped me enormously throughout this research.

My gratitude is extended to Dr. Elmasry, Dr. Garcia-Frias, Dr. Haddad, Dr. Haimovich and Dr. Ramkumar for serving as members on the dissertation committee and for their comments.

I had the pleasure of working with my colleagues at the New Jersey Center for Multimedia Research at NJIT. Their help and suggestions are appreciated and acknowledged.

I sincerely thank my husband Zhifang, my father and my sister for their continuous support, encouragement and great love. This work could not have been possible if it were not for them.

Finally, I would like to thank Dr. Ronald S. Kane and Ms. Clarisa Gonzalez-Lenahan in the Office of Graduate Studies in NJIT for their time and assistance. I also wish to express my appreciation to Mr. Bob Lazar, who made efforts to revise this dissertation with respect to the language.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Motivations

A basic scheme for communicating over unreliable channels is illustrated in Figure 1.1. The message to be sent is encoded with a channel encoder before it is transmitted. At the receiving end, the output from the channel is decoded back into a message, hopefully the same as the original one. A fundamental property of such a system was established by Shannon [1] more than 50 years ago, which states that reliable communication can be achieved as long as the information rate does not exceed the capacity of the channel, provided that the encoder and the decoder are allowed to operate on long enough sequences of data.

The decoding problem can be solved, in principle, by searching through all possible messages and comparing their corresponding codewords with the channel output, selecting the message which is most likely to result in the observed channel output. In general, this is called maximum likelihood (ML) decoding and the performance obtained is called ML performance. A code set with large minimum Hamming distance has the potential to achieve excellent performance. There exist many good codes in this sense. However, ML decoding is generally impractical because the number of messages is too large. Therefore, the problem of constructing good code is actually constructing code that could be decoded with reasonable complexity and still give sufficiently good performance.

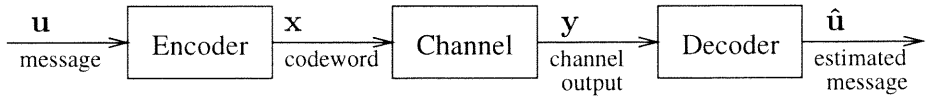


Figure 1.1 Shannon's model for reliable communication over unreliable channel

Decoding methods can be roughly divided into two classes: algebraic and "probabilistic". Algebraic decoders usually quantize the channel output to $\mathbf{y}$, which has the same alphabet as the transmitted codeword $\mathbf{x}$. $\mathbf{y}$ is interpreted as a copy of the codeword $\mathbf{x}$, except that some of the coordinates are flipped. Decoding is a matter of using linear algebra (in a finite field) to find the transmitted codeword $\mathbf{x}$ that is closest to $\mathbf{y}$ in Hamming distance. Many typical block codes such as Hamming, BCH, have built-in special structures to make algebraic decoding easier.

Probabilistic decoders make as much use as possible of the real-valued channel output. The goal of probabilistic decoding is either maximum likelihood (ML) information sequence detection or maximum a posteriori (MAP) information bit detection:

$$\begin{aligned} \mathbf{u}^{\mathrm{ML}} &= \arg\max_{\mathbf{u}} p(\mathbf{y}|\mathbf{u}) \\ u_k^{\mathrm{MAP}} &= \arg\max_{u_k} p(u_k|\mathbf{y}) \end{aligned} \tag{1.1}$$

A famous ML probabilistic decoder is the soft decision Viterbi algorithm for convolutional code, which takes advantage of code trellis to make ML decoding feasible.

By using the iterative MAP decoder, turbo code claims a performance approaching the Shannon limit [2]. The success should be attributed more to the elegant iterative MAP decoder than to the code construction. It is not surprising that good code spectrum could be achieved using random interleaver, for even a randomly selected code offers near Shannon limit performance under ML decoding [3]. Actually, with iterative decoding, many other compound codes such as low-density parity-check (LDPC) code and product code, also claim excellent performance. Being new and extremely powerful, iterative decoding algorithms surely deserve further research efforts. This dissertation deals with the iterative decoding of LDPC code and product code.

Interestingly enough, LDPC code was proposed as early as 1963 [4], together with the probability decoding algorithm. It had been largely forgotten due to the lack of computing power to demonstrate its excellent performance under longer code length. Stimulated by success of turbo code, it was rediscovered in 1997 [5, 6]. Soon after that, the iterative decoding algorithm for LDPC code, turbo code and some other compound codes were unified under the theoretical framework of Pearl's belief propagation (BP) on Bayesian network [7, 8]. Meanwhile, newly constructed LDPC codes were shown to approach the Shannon limit a step further [9, 10, 11], exceeding what had been achieved by the original turbo code.

A large part of this dissertation deals with the iterative decoding of LDPC code. The purpose of this study is to gain better understanding on several aspects of the decoding algorithm, namely, its decoding capacity, quantized implementations and robustness to channel estimation offset. Max-Log-MAP decoding, a suboptimal algorithm well known in turbo code research but neglected in LDPC code, is also examined.

The remaining part of this dissertation is devoted to iterative decoding of product code. Product code was proposed in as early as 1954 [12]. Its large minimum Hamming distance promises a good performance. However, due to lack of a powerful decoding algorithm, it had been largely forgotten until 1993 when soft-in soft-out iterative decoding was applied to it [13]. The purpose of this study is to gain better understanding on the decoder and seek some further improvement on error performance and decoding delay. The iterative decoding algorithm in the context of belief propagation is examined. Through this study, it is found that for product code a small modification on the original belief propagation algorithm would render some improvement on bit error rate (BER) performance. An alternative decoding structure, which has the potential to reduce decoding delays, is also investigated.

1.2 Outline of Dissertation

The dissertation is organized as follows:

Chapter 2 introduces two theoretical foundations for this work. One is Shannon's channel coding theorem. The other is Pearl's belief propagation (BP) on graphical models, which has been well known to artificial intelligence community but is relatively new to the coding theory community. All of the work in the following chapters are based on this newly established theoretical framework for iterative decoding. That is why the name graph based code is used to represent LDPC code and product code.

Chapter 3 is devoted to capacity evaluation of LDPC code. First the concept of iterative decoding capacity is introduced in the context of density evolution, a newly developed procedure based on certain idealized conditions. Then two simulation-based methods approximating decoding capacity are applied to LDPC codes. Their effectiveness is evaluated. A suboptimal decoder, Max-Log-MAP algorithm, which has been intensively studied in turbo code but seems to be neglected in LDPC code, is also investigated. The specific density evolution procedure for Max-Log-MAP decoding is developed and the result is included in a recent paper [14].

Chapters 4 and 5 deal with two implementation issues for LDPC code, design of quantized decoders and sensitivity to SNR mismatch. The capacity of quantized LDPC decoder, under Log-MAP and Max-Log-MAP algorithms, is derived through discretized density evolution. In addition, the influence of clipping limit on decoding performance, is studied. It is an important design parameter in general LDPC decoders but (to the best of the author's knowledge) has never been discussed in particular. It is indicated that the key point in designing a quantized decoder is to pick a proper dynamic range. Quantization loss in terms of bit error rate (BER) performance could be kept remarkably low, provided that the dynamic range is chosen wisely. In Chapter 5, the decoding capacity under some fixed SNR offset is

obtained. The robustness of LDPC codes with practical length is evaluated through simulations. It is found that the amount of SNR offset that can be tolerated depends on the code length. Two papers on this part of work have been completed [15, 16].

Chapter 6 investigates iterative decoding algorithms for product code. For the first time, linear correlation coefficient is used to measure the dependency among extrinsic information. Scaled factor decoding (SFD) is proposed, as a modified version of BP, to overcome the impact of uniform short cycles. Simulation results show that SFD offers a performance surprisingly close to optimum decoding. Parallel iterative decoding is also investigated. The results in this chapter have been published in several conference papers [17, 18, 19, 20].

Chapter 7 gives the conclusions for this research. The special contributions are also summarized.

CHAPTER 2

THEORETICAL FOUNDATIONS

In this chapter, necessary theoretical foundations for this work are introduced. The concept of channel capacity was proposed by Shannon in 1948. The recent wave of research on turbo code and other random compound code was stimulated by the fact that their performance approach the Shannon limit. Section 2.1 elaborates the concept of Shannon limit and gives some channel coding bounds.

McEliece et.al. [7, 8] established the theoretical framework for iterative decoding by connecting it with Pearl's belief propagation on Bayesian network [21], which has been well known in the artificial intelligence community but is relatively new to the coding theory community. Many significant new results on iterative decoding are derived in the context of graphical codes. The basic concepts and ideas are introduced in Section 2.2 and 2.3.

2.1 Channel Capacity

In his famous paper, "A Mathematical Theory of Communication" [1], Claude Shannon introduced the important concept of channel capacity C, which is the average mutual information between channel input X and output Y maximized over all channel input distributions:

$$C = \max_{p(x)} I(X;Y) \tag{2.1}$$

Average mutual information $I(X;Y)$ is defined as,

$$I(X;Y) = \begin{cases} \sum_x \sum_y P(x)P(y|x) \log_2 \frac{P(y|x)}{P(y)} & \text{for discrete channel} \\ \int\int p(y|x)p(x) \log_2 \frac{p(y|x)}{p(y)} dxdy & \text{for continuous channel} \end{cases} \tag{2.2}$$

where conditional probability $p(y|x)$ or $P(y|x)$ represents channel characteristics.

For binary symmetric channel (BSC) with transition probability $p(y|x)$, $I(X;Y)$ is maximized when the input probability $P(0) = P(1) = 0.5$. Thus,

the capacity for BSC is,

$$C = 1 - H_b(p) = 1 + p\log_2 p + (1-p)log_2(1-p) \tag{2.3}$$

For a waveform channel with $N(0, N_0/2)$ additive white Gaussian noise (AWGN) and continuous unconstrained input alphabet x (UC-AWGN), the maximum of $I(X;Y)$ over the input pdf $p(x)$ is obtained when x is zero-mean Gaussian random variable, i.e.,

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-x^2/2\sigma_x^2} \tag{2.4}$$

Then it follows that the channel capacity is,

$$\begin{aligned} C &= 0.5\log_2(1 + \frac{2\sigma_x^2}{N_0}) \\ &= 0.5\log_2(1 + \frac{2E_s}{N_0}) \end{aligned} \tag{2.5}$$

measured in bits per transmission. E_s is energy per channel symbol.

Shannon's **noisy channel coding theorem** states that: there exist channel codes (and decoders) that make it possible to achieve reliable communication, with as small an error probability as desired, if the transmission rate $r < C$, where C is the channel capacity. If $r > C$, it is not possible to make the probability of error tend toward zero with any code.

2.1.1 Channel Coding Bounds

Three types of channels are discussed. The first is the unconstrained AWGN (UC-AWGN), where the coding alphabet is unconstrained and can assume any value. The second is the binary input AWGN channel (BI-AWGN), where the input alphabet is constrained to $\{-1, +1\}$. The third is the binary-input binary-output channel (BIBO), where both the input and output are constrained to be binary. Since the channel noise is assumed to be AWGN, BIBO is equivalent to a BSC.

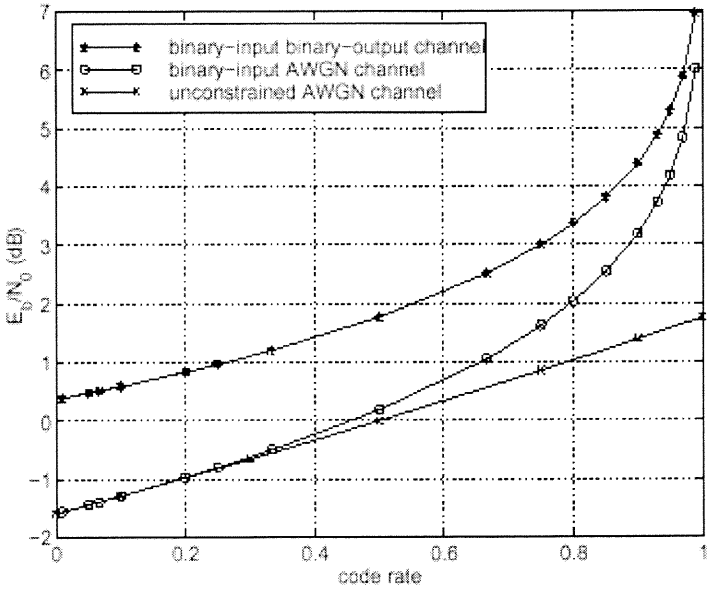


Figure 2.1 Minimum E_b/N_0 vs. coding rate r [22]

2.1.1.1 Minimum E_b/N_0. Let E_b denote the average transmitted symbol energy when channel coding is not used. Thus, when a rate r channel coding is applied, the average symbol energy becomes $E_s = rE_b$. Substituting into equation (2.5),

$$C = 0.5 \log_2(1 + \frac{2rE_b}{N_0}) \tag{2.6}$$

- For UC-AWGN channel, with the constraint $r \leq C$, equation (2.6) implies that,

$$r \leq 0.5 \log_2(1 + \frac{2rE_b}{N_0}) \tag{2.7}$$

- For BI-AWGN channel, the pdf of X is fixed to

$$p(x) = 0.5(\delta(x + \sqrt{rE_b}) + \delta(x - \sqrt{rE_b})) \tag{2.8}$$

so that the channel capacity is exactly

$$C = \frac{2rE_b}{\ln 2N_0} - \frac{1}{\sqrt{2\pi}} \int e^{-t^2/2} \log_2 \cosh(t\sqrt{\frac{2rE_b}{N_0}} + \frac{2rE_b}{N_0})dt \tag{2.9}$$

With $r \leq C$, it implies that

$$r \leq \frac{2rE_b}{\ln 2N_0} - \frac{1}{\sqrt{2\pi}} \int e^{-t^2/2} \log_2 \cosh(t\sqrt{\frac{2rE_b}{N_0}} + \frac{2rE_b}{N_0})dt \tag{2.10}$$

- For BIBO with AWGN, the binary channel symbol error probability is,

$$p = Q(\sqrt{\frac{2rE_b}{N_0}}) \tag{2.11}$$

and equation (2.3) gives,

$$r \leq C = 1 - H_b(p) = 1 + p\log_2 p + (1-p)log_2(1-p) \tag{2.12}$$

Taking equalities in equation (2.7), (2.10) and (2.12), the relationship between the minimum E_b/N_0 for error-free transmission and coding rate r can be found, as shown in Figure 2.1

It is clear that for $r < 1/3$, the power requirement for BI-AWGN is approximately the same as that for UC-AWGN. The difference increases significantly for greater values of r. The gap between BIBO and BI-AWGN can be interpreted as the difference of soft decision decoding and hard decision decoding. For most coding rates employed in practice, there is a 1.5-2 dB gain realized in going from hard decision decoding to soft decision decoding.

2.1.1.2 $P_b(e)$ **bounds.** The lower bound for achievable $P_b(e)$ under certain coding rate and E_b/N_0 is derived as follows. If the information bits are received with error probability $P_b(e)$, then the information capacity is $1 - H_b(e)$ [23], where

$$H_b(e) = -P_b(e)\log_2 P_b(e) - (1 - P_b(e))\log_2(1 - P_b(e)) \tag{2.13}$$

is the entropy of a binary source with error probability $P_b(e)$. Given the channel capacity C, it is bounded by ,

$$1 - H_b(e) \leq \frac{C}{r} \tag{2.14}$$

From this inequality, bounds for $P_b(e)$ can be derived.

For UC-AWGN, the capacity is obtained in equation (2.5). Substituting it into equation (2.14), the information capacity is obtained as,

$$1 - H_b(e) \leq \frac{\log_2(1 + \frac{2rE_b}{N_0})}{2r} \tag{2.15}$$

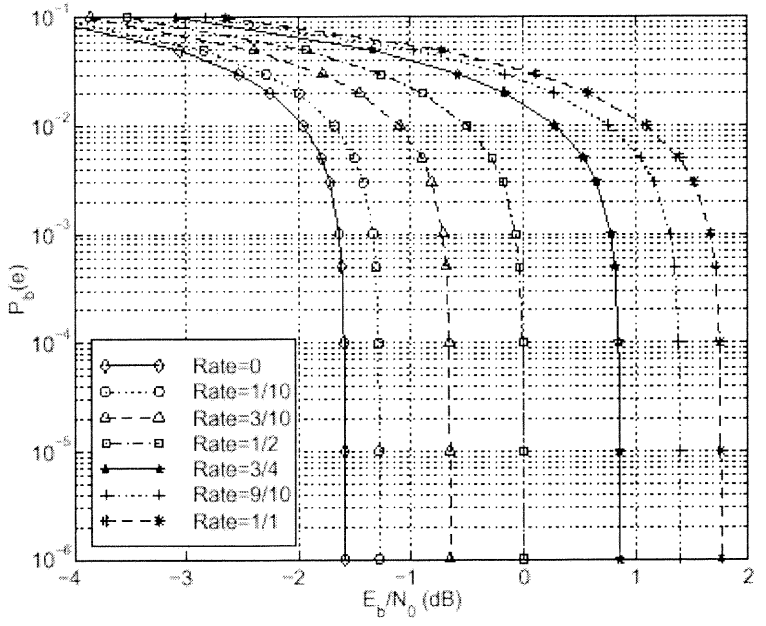


Figure 2.2 Bit error probability bounds for UC-AWGN channel [22]

$P_b(e)$ limits under some specific rate r are numerically calculated and plotted in Figure 2.2.

When $r \to 0$, the transmission bandwidth approaches infinity. For this case, the lowest possible SNR is reached if arbitrary low bit error rate is demanded.

$$\begin{aligned} \lim_{r\to 0}[1 - H_b(e)] &= \lim_{r\to 0} \frac{\log_2(1 + \frac{2rE_b}{N_0})}{2r} \\ &= \frac{E_b}{(\ln 2)N_0} \end{aligned} \tag{2.16}$$

When $P_b(e) \to 0$, $1 - H_b(e) \to 1$. This implies that to achieve reliable transmission, the minimum power for all coding schemes is: $E_b/N_0 = \ln 2$, which is -1.6 in dB.

Similarly, for BI-AWGN channel, substitute equation 2.9 into equation (2.14) to obtain,

$$1 - H_b(e) \leq \frac{2rE_b}{\ln 2 N_0} - \frac{1}{\sqrt{2\pi}} \int e^{-t^2/2} \log_2 \cosh(t\sqrt{\frac{2rE_b}{N_0}} + \frac{2rE_b}{N_0})dt \tag{2.17}$$

$P_b(e)$ limits under some specific rate r are numerically calculated and plotted in Figure 2.3.

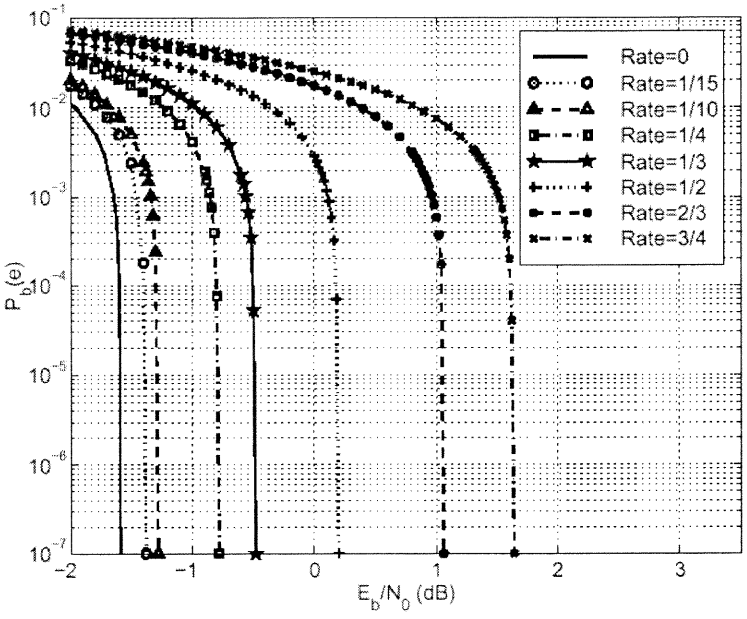


Figure 2.3 Bit error probability bounds for BI-AWGN channel [22]

This figure is occasionally referred to in papers trying to demonstrate codes with near Shannon limit performance. In this dissertation, rate 1/2 LDPC code over BI-AWGN channel is considered. Correspondingly, the bound on E_b/N_0 for bit error rate of 10^{-5} is about 0.18 dB.

Similarly, let $r \to 0$ and then $1 - H_b(e) \to 1$, the power limit of BI-AWGN is found to be: $E_b/N_0 = \ln 2$, identical to that of UC-AWGN.

2.2 Belief Propagation on Bayesian Network

2.2.1 Probability Inference Problem

Let $\mathbf{X} = \{X_1, X_2, \ldots X_N\}$ be a set of N discrete random variables. Assume that the marginal density $p(x_i)$ is also known, which represents a priori "belief" about the random variable (r.v.). Suppose some of the r.v.'s are "observed", which means that there is a subset $J \subseteq \{1, 2, \ldots, N\}$ (the evidence set) such that for all $j \in J$, the r.v. X_j is known to have a particular value a_j,

$$\mathcal{E} = \{X_j = a_j : j \in J\} \tag{2.18}$$

The fundamental probabilistic inference problem is to compute the updated beliefs, i.e. the a posteriori probabilities $\text{Bel}(x_i) = p(x_i|\mathcal{E})$ for all $i \notin J$. Under the circumstance of decoding, a MAP decision is made by

$$\hat{x}_i = \arg\max_{x_i} \text{Bel}(x_i) \tag{2.19}$$

A brute force approach to computing $p(x_i|\mathcal{E})$ is to sum over all the terms of $p(\mathbf{x})$ which do not involve either i or J. Unfortunately, the heavy computation burden makes this approach impractical.

The idea behind the "Bayesian belief network" approach to inference problem is to exploit any "partial independence" which may exist among X_i's to simplify belief propagation. The partial "independence" can be described by a directed acyclic graph (DAG). A DAG is a finite, directed graph, in which there are no directed cycles. If there is a directed edge $a \longrightarrow b$, then a is called a "parent" of b and b is called a "child" of a. Denote the set of parents of vertex v by $\text{pa}(v)$. If $\mathbf{X}$ is a set of r.v. in correspondence with the vertices of graph G, the joint density function $p(\mathbf{x})$ is factored according to G if

$$p(x_1, x_2, \ldots, x_N) = \prod_{i=1}^{N} p(x_i|\text{pa}(x_i)) \tag{2.20}$$

A DAG, together with the associated r.v. $\mathbf{X}$, is called a Bayesian belief network. In the 1980's, Kim and Pearl showed that if the DAG is loop-free, then there are efficient algorithms for solving the inference problem. The exact a posteriori probability could be obtained [21].

2.2.2 Pearl's Belief Propagation Algorithm

Pearl's belief propagation (BP) algorithm is a decentralized "message passing" algorithm, in which there is a processor associated with each vertex of the Bayesian network. Each processor can communicate only with its parents and children. The processor associated with a variable X knows the conditional density function

$p(x|\mathbf{u}) \triangleq \Pr\{X = x|U_1 = u_1, \ldots, U_M = u_M\}$, where U_1, ..., U_M are the parents of X. If $\text{pa}(x) = \varnothing$, this knowledge degenerates to the marginal density function $p(x) = \Pr\{X = x\}$. When a processor is activated, it "reads" the messages from its parents and children, updates its belief based on these messages, and then sends new messages back to its parents and children.

The message node X receives from its parents U_i, denoted $\pi_{U_i,X}(u_i)$, is in the form of a list of probabilities, one for each value $u_i \in \mathcal{F}_{U_i}$. Informally, $\pi_{U_i,X}(u_i)$ is the probability of the event $U_i = u_i$, conditioned on the evidence in the tree already "known" to U_i. Similarly, the message X receives from its child Y_j, denoted by $\lambda_{Y_j,X}(x)$ is in the form of a list of nonnegative real numbers, one for each value of $x \in \mathcal{F}_X$. Roughly speaking, $\lambda_{Y_j,X}(x)$ is the probability of the evidence Y_j "knows", given the event $X = x$.

Node X then computes the probability of $X = x$ given the evidence from its parents

$$\pi_X(x) = \sum_{\mathbf{u}} p(x|\mathbf{u}) \prod_{k=1}^{M} \pi_{U_k,X}(u_k) \tag{2.21}$$

If X is a root node, which means it has no parent, then $\pi_X(x) = p(x)$ (marginal probability). The likelihood of $X = x$ given the evidence known to its children is,

$$\lambda_X(x) = \prod_{j=1}^{N} \lambda_{Y_j,X}(x) \tag{2.22}$$

If X has no children then $\lambda_X(x) = 1$. These two quantities are then combined to obtain the updated belief about the variable X

$$\text{Bel}(x) = \alpha \pi_X(x) \lambda_X(x) \tag{2.23}$$

where α is a normalization factor.

After X has been activated, the message that X passes to its child Y_j, denoted $\pi_{X,Y_j}(x)$, is a list of probabilities, one for each value of x. Roughly speaking, $\pi_{X,Y_j}(x)$ is the probability of the event $X = x$, given the evidence in the network already

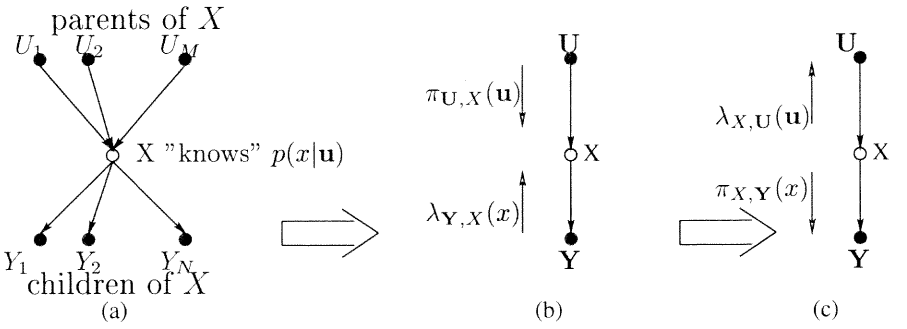


Figure 2.4 Summary of Pearl's belief propagation

known to X, which now indicates any new evidence which may have been contained in the incoming messages but excludes the information from Y_j to X

$$\pi_{X,Y_j}(x) = \alpha \frac{\text{Bel}(x)}{\lambda_{Y_j,X}(x)}. \tag{2.24}$$

Similarly, the message X passes to its parents U_k, denoted by $\lambda_{X,U_k}(u_k)$, is the list of probabilities of the evidence it knows about, given the event $U_k = u_k$ for each possible value u_k

$$\lambda_{X,U_k}(u_k = b) = \alpha \sum_{\mathbf{u}:u_k=b} [\sum_x \lambda_X(x) p(x|\mathbf{u})] \prod_{j=1,j\neq k}^{M} \pi_{U_j,X}(u_j). \tag{2.25}$$

A summary of Pearl's belief propagation algorithm is shown in Figure 2.4. The algorithm is initialized by setting all $\lambda_{X,U}(u)$ to 1 unless X is observed to be x_0, in which case $\lambda_{X,U}(u) = p(x_0|u)$. For each source node X, set $\pi_X(x)$ to a priori probability $p(x)$. A node can be activated only if all of its incoming messages exist. Usually, the source nodes and evidence nodes are the first to be activated. Once a node is activated, it updates its own belief $\text{Bel}(x)$, derives the messages $\lambda_{X,U}$ and $\pi_{X,Y}$, and passes them to its parents and children respectively. If the graph is a tree, the algorithm will finally converge to a unique and exact solution–the exact a posteriori probability (APP) of each variable.

2.3 Graphical Models for Error Correcting Codes

Under Pearl's belief propagation algorithm, the information originating from a certain node would gradually spread to other nodes. Eventually, each node in the network would have collected all the information available. At this point, if the graph is loopless, a steady state is reached. The belief would stay the same even if the procedure moves on. The final belief is nothing else but the exact APP. However, if the graph is loopy, information would circulate in the network. The algorithm is not guaranteed to converge. Of course, the results are no longer a posteriori probability. Exact probability inference in a multiply-connected network is in general NP-hard [24].

As for Pearl's BP algorithm with application to error correcting codes, the bad news is that most long compound codes, such as turbo code and LDPC code, are loopy. So far, these long codes have no other good choices but to be decoded by probability decoders. The good news is, exact APP is not required for decoding applications. The objective is to infer whether the transmitted symbol is zero or one.

Recently, it is shown that belief propagation for decoding a wide variety of long random error correcting codes is likely to yield very good performance, even though the corresponding Bayesian networks are loopy [7, 8]. Turbo decoding is an instance of belief propagation in a loopy Bayesian network. It seems that the effect of cycles tends to diminish if the cycles are long enough and random enough.

McEliece et.al. are the first to connect the iterative decoding algorithm with Pearl's belief propagation [7]. However, the first to describe error correcting codes by graphical models might be Tanner [25] in 1981. Roughly speaking, graphical models are basically identical to the underlying graphs in Bayesian network.

So far, the graphical symbol set for compound code has not yet been unified. Tanner was the first to introduce bipartite graphs to describe families of codes which

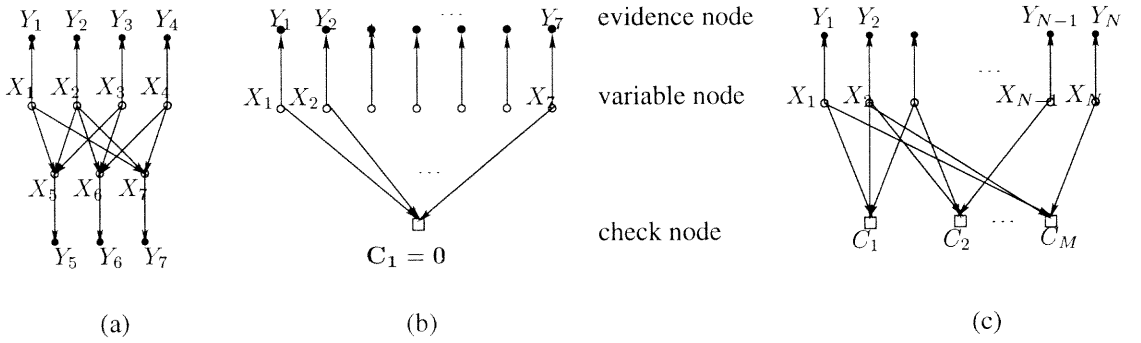


Figure 2.5 Graphical models for: (a) and (b) (7,4) Hamming code (c) LDPC code

are generalizations of the LDPC codes [25]. In Tanner's original formulation, there are two types of nodes/variables: nodes representing code digits and nodes for subcodes (code constraint). Wiberg introduced "hidden" (latent) state variables, which easily incorporate convolutional code [26]. Recently, the concept of factor graph [27] takes these theoretical graph models one step further, by adding a function node and applying it to various functions. A wide variety of algorithms, including turbo decoding, Kalman filter, and certain FFT are specified as an instance of sum-product algorithm.

Basically, a simple graphical model just enough to describe the iterative decoding algorithm is employed. The graphical model in this research is a combination of Tanner's bipartition graph and McEliece's Bayesian network [7]. Three types of nodes are defined: variable nodes, check nodes and evidence nodes. Variable node X_i represents a coded digit. Evidence node Y_j represents a received noise-corrupted symbol. Check node C_k corresponds to code constraint.

There could be more than one graphical model for the same code. Figure 2.5 (a) shows a Bayesian network for (7,4) Hamming code. Probability propagation derived in this context would yield a suboptimal iterative decoder, since it is loopy. Figure 2.5 (b) describes (7,4) Hamming code in a different form of graph. Here the check node $\mathbf{C_1}$ is equivalent to syndrome vector and it is viewed as an all-zero evidence node in the context of belief propagation. Given the codeword set, $\mathbf{C_1}$ "knows"

the conditional probability $p(\mathbf{c_1}|\mathbf{x})$. Based on this model, the belief propagation algorithm is nothing else but the MAP decoder. For detailed performance comparison of the two decoders, refer to [24]. The loopless graphical model is used in product code as subgraph for component code.

Graphical model for LDPC code is given in Figure 2.5 (c). If the evidence nodes and edges connected to them are removed, a bipartition graph is obtained. It is almost identical to Tanner's graph, except that the check node was called subcode and was associated with a single parity check in the latter. In LDPC code, code constraint equals parity check. But code constraint is preferred because this make it easy to incorporate generalized LDPC code [28] into this model.

Almost all iterative decoding schemes, including the sum-product algorithm for LDPC code, BCJR for turbo code and MAP decoding for block code, were developed well before BP was connected to probability decoding. Actually, BP has not brought any significant BER performance improvement so far. Instead, it inspires some alternative efficient implementation schemes for turbo decoding [29, 30, 31]. It is the conclusion "belief propagation would converge to exact APP under loopless condition" that results in a big step on iterative decoding theory. Setting out from "loopless" assumption, several groups of researchers were able to derive capacity for iterative decoding. Details are given in the next Chapter.

2.4 Summary

This chapter introduces some theoretical foundations for this work. First the noisy channel coding theorem is presented. The error performance bounds for binary input AWGN and unconstrained AWGN channels are given. In general, these bounds are termed "Shannon limit" in research papers. Then the theory of belief propagation on Bayesian network is introduced, which is well known in the artificial intelligence community but is relatively new to the coding theory community. A simple graphical

model just enough to describe the iterative decoding process is adopted throughout this dissertation. All of the work in the following chapters is based on belief propagation on graphical model, the newly established theoretical framework for iterative decoding.

CHAPTER 3

LOW-DENSITY PARITY-CHECK CODE: DECODING CAPACITY

A brief introduction on LDPC code is given in Section 3.1, about its definition, graphical model and Log-MAP decoding algorithm. Then the concept of iterative decoding capacity is defined in the context of density evolution and parameter evolution. Capacity of Log-MAP decoding is obtained using parameter-evolution, namely SNR-evolution and mutual information evolution. Finally, Max-Log-MAP decoding, a suboptimal algorithm is discussed. The most attractive advantage is its universal most powerfulness (UMP), that is, no SNR estimation is required. Density evolution procedure for Max-Log-MAP decoding is developed.

3.1 Low-Density Parity-Check Code

Low-density parity-check code was invented by Gallager in his thesis in 1963, along with several iterative decoding algorithms including the famous "sum-product" algorithm. However, LDPC code had been largely forgotten since the computing power at that time was too limited to demonstrate its potential under large code length. About five years ago, several groups of researchers rediscovered the power of LDPC code and "sum-product" decoding algorithm. D. MacKay et. al showed empirically that long LDPC code also offers near optimum performance [6], just as turbo code does.

3.1.1 Code Definition

Low-density parity-check code is defined by a sparse $M \times N$ parity check matrix $\mathbf{H}$ with elements from $\{0, 1\}$. For a regular LDPC code, $\mathbf{H}$ has uniform column weight j and uniform row weight k. It is denoted as regular (N,j,k) or (j,k) LDPC code. Generally, it follows that the coding rate r satisfies $r = (N - M)/N = 1 - j/k$.

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Table 3.1 Parity check matrix of a (3,4) LDPC code with $N = 20$

To derive the generator matrix, $\mathbf{H}$ is split into two sparse sub-matrices $\mathbf{H} = [\mathbf{C_1} \mid \mathbf{C_2}]$, where $\mathbf{C_2}$ is a square $M \times M$ matrix and $\mathbf{C_1}$, $M \times K$ matrix, $K = N - M$ being the size of source block length. The generator matrix is,

$$\mathbf{G^T} = \begin{bmatrix} \mathbf{I_K} \\ \mathbf{C_2^{-1}C_1} \end{bmatrix} \tag{3.1}$$

where $\mathbf{I_K}$ is a $K \times K$ identity matrix. Source binary vector $\mathbf{s}$ of length K is encoded into a vector $\mathbf{x}$ of length N by: $\mathbf{x} = \mathbf{G^T s}$. $\mathbf{x}$ satisfies $\mathbf{Hx} = \mathbf{0}$.

MacKay gives several construction schemes to produce the parity check matrix $\mathbf{H}$, and they yield similar BER performance. Basically Gallager's simple construction is adopted throughout this research.

The matrix is divided into j $M_1 \times N$ sub-matrices with $M_1 = M/j$, each containing a single 1 in each column. The first of these matrices contains all its 1's in descending order; that is, the i^{th} row contains 1's in columns $(i-1)k+1$ to ik. The other sub-matrices are merely random column permutations of the first sub-matrix. A 15×20 $\mathbf{H}$ matrix with $j = 3$ and $k = 4$ is shown in Table 3.1 as an example.

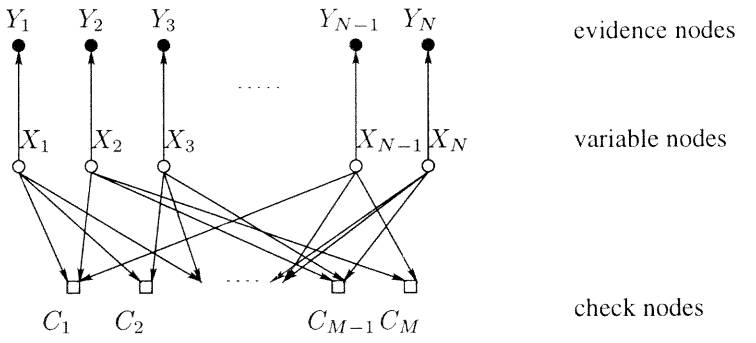


Figure 3.1 Graphical model for LDPC code

Following MacKay's scheme, a constraint is imposed on the permutation operation: no two columns should have overlap greater than 1. It is addressed that Gallager's code construction has an embedded random interleaver, very similar to turbo code's interleaver.

Figure 3.1 displays the graphical model for LDPC code. It's almost a direct mapping from the **H** matrix of LDPC code. The N coded digits are represented by N variable nodes. M parity checks are represented by M check nodes. Each non-zero element H_{mn} in the **H** matrix corresponds to an edge that connects variable node n and check node m. Using the graphical model, it is quite easy to describe the decoding algorithms.

3.1.2 Iterative Decoding Algorithms

3.1.2.1 Sum-product algorithm. The earliest iterative decoding scheme for LDPC code was proposed by Gallager in his Ph.D dissertation [4]. Later it was generalized by Wiberg [26] and elaborated by MacKay under the name of sum-product algorithm [6]. It is exactly Pearl's belief propagation (BP) except that different sets of symbols are used.

The decoding process comprises three steps: initialization, horizontal pass and vertical pass. The objective is to update the "pseudo-posterior probabilities" q_n^0 and q_n^1 through iterations, where $q_n^0 + q_n^1 = 1$. The quantity q_n^a with $a \in \{0, 1\}$ is

meant to be the probability that $x_n = a$, given the information from those check nodes connected to variable node n. The initial a priori probability of x_n is as, $p_n^a = P(x_n = a) = \alpha p(y_n|x_n = a)$ so that $p_n^0 + p_n^1 = 1$. This pair of values is equivalent to the concept of channel value or intrinsic information in Hagenauer's paper, where log-likelihood ratio (LLR) symbol set is employed.

The set of bits n that participates in check m is denoted by $N(m) = \{n : H_{mn} = 1\}$. Similarly, the set of checks in which bit n participates is denoted as $M(n) = \{m : H_{mn} = 1\}$. A set $N(m)$ with bit n excluded is denoted by $N(m)\backslash n$, and a set $M(n)$ with parity check m excluded, by $M(n)\backslash m$. The iterative decoding algorithm has two alternating parts, vertical pass and horizontal pass, in which certain quantities q_{mn}^a and r_{mn}^a, associated with each non-zero element in the matrix $\mathbf{H}$, are iteratively updated. The quantity q_{mn}^a is meant to be the probability that x_n is a, given the information obtained via checks other than check m. The quantity r_{mn}^a is meant to be the probability that check m is satisfied if bit n is considered fixed at a while the other bits have a separable distribution given by the probabilities $\{q_{mn'} : n' \in N(m)\backslash n\}$. The algorithm would produce the exact a posteriori probabilities (APP) for all the coded bits if the bipartition graph contains no cycles. This conclusion is derived from the fact that Pearl's belief propagation algorithm converges to exact APP if the Bayesian network is loopless.

The sum-product algorithm for LDPC code consists of the following steps:

- *Initialization:* The variables q_{mn}^0 and q_{mn}^1 are initialized to the values of p_n^0 and p_n^1 respectively.

- *Horizontal pass:* Run through all the checks m and compute for each $n \in N(m)$, $a = 0, 1$:

$$r_{mn}^a = (1/2)(1 + (-1)^a \delta r_{mn}) \tag{3.2}$$

where

$$\delta r_{mn} = \prod_{n' \in N(m) \backslash n} (q^0_{mn'} - q^1_{mn'}) \tag{3.3}$$

- *Vertical pass:* For each n and m, and for $a = 0, 1$, update:

$$q^a_{mn} = \alpha_{mn} p^a_n \prod_{m' \in M(n) \backslash m} r^a_{m'n} \tag{3.4}$$

where α_{mn} is a normalizing parameter so that $q^0_{mn} + q^1_{mn} = 1$.

- *Decisions:* For each bit n and $a = 0, 1$, update the "pseudo-posterior probabilities":

$$q^a_n = \alpha_n p^a_n \prod_{m \in M(n)} r^a_{mn} \tag{3.5}$$

where α_n is chosen such that $q^0_n + q^1_n = 1$. The decision so far is given by $\hat{\mathbf{x}} = [\hat{x}_n]$ with $\hat{x}_n = 1$ if $q^1_n > 0.5$; otherwise $\hat{x}_n = 0$. If $\hat{\mathbf{x}}$ is a valid codeword such that $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$, then the algorithm halts; otherwise, repeat the horizontal and vertical pass until some maximal number of iteration is reached without a valid decoding.

The sum-product algorithm is exactly the BP algorithm. The horizontal pass is actually the message generation process in check nodes. The vertical pass is the message generation process in variable nodes. The M check nodes are actually M parallel component decoders activated simultaneously.

3.1.2.2 Log-MAP algorithm. Representing the messages in LLR symbol set, sum-product algorithm could be proved to be equivalent to Log-MAP algorithm. Detailed proof is given in Appendix A.

Define the log-likelihood ratio symbol set as follows: intrinsic value $L_n \triangleq \log \frac{p^0_n}{p^1_n}$, a posteriori log-likelihood ratio for making final decisions $L(\hat{x}_n) \triangleq \log \frac{q^0_n}{q^1_n}$, extrinsic information in log measure $\xi_{mn} \triangleq \log \frac{r^0_{mn}}{r^1_{mn}}$ and the input to check node $\zeta_{mn} \triangleq \log \frac{q^0_{mn}}{q^1_{mn}}$.

- *Initialization:* Set the input to check node m from variable node n as channel value $\zeta_{mn}^{(0)} = L_n$.

- *Horizontal pass:*

$$\xi_{mn}^{(i)} = 2\tanh^{-1}\Big(\prod_{n' \in N(m)\backslash n} \tanh(\zeta_{mn'}^{(i-1)}/2)\Big) \tag{3.6}$$

- *Vertical pass:*

$$\zeta_{mn}^{(i)} = \log\frac{q_{mn}^{0}}{q_{mn}^{1}} = L_n + \sum_{m' \in M(n)\backslash m} \xi_{m'n}^{(i)} \tag{3.7}$$

- *Making decisions:* For each bit n "pseudo-posterior LLR":

$$L^{(i)}(\hat{x}_n) = L_n + \sum_{m \in M(n)} \xi_{mn}^{(i)} \tag{3.8}$$

  The decision so far is given by $\hat{\mathbf{x}} = [\hat{x}_n]$ such that $\hat{x}_n = 0$ if $L^{(i)}(\hat{x}_n) > 0$; otherwise $\hat{x}_n = 1$. If $\hat{\mathbf{x}}$ is a valid codeword such that $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$, then the algorithm halt; otherwise, repeat the horizontal and vertical pass until some maximal number of iteration is reached without a valid decoding.

In Hagenauer's papers [32], MAP decoding refers to decoding using probability symbol set. It is identical to sum-product algorithm. Log-MAP decoding is the same as MAP decoding except that LLRs replace probabilities.

3.1.3 Upper Bounds on ML Decoding

LDPC code is a linear block code. Through studying the codeword distance spectrum, one can easily upper bound the block error rate of LDPC code under ML decoding. Gallager gave an upper bound on the ensemble distance spectrum of (N,j,k) binary LDPC code [4]. Sason visualizes the codeword spectrum of (10000,j,2j) LDPC for $j = 3, 4, 5, 6$ in his paper [33]. The basic conclusion is, for relatively small values of j and k, increasing the value of j and k with the ratio of j/k fixed (coding

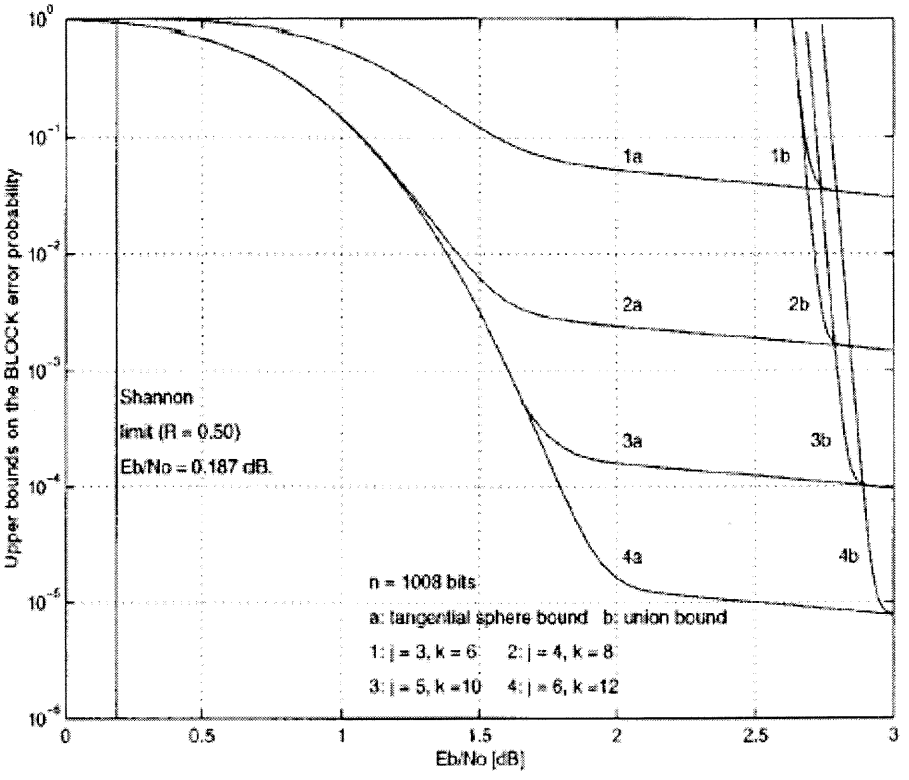
rate kept constant) reduces the number of codewords of low Hamming weight, thus lowers the error floor.

Using Gallager's distance spectrum, Sason was able to derive two upper bounds: one is union bound in Q-form; the other is tangential sphere bound which is much tighter than the union bound. For convenience of illustration, the results are shown in Figure 3.2. Comparing the two error bounds for block length $N = 1008$ and $N = 10000$ under identical j and k, it is found that increasing N would bring about dramatic improvement. Under the same N and coding rate j/k, larger j pulls the error bound towards the Shannon limit. This coincides with the intuitive understanding: denser parity check should improve the code performance.

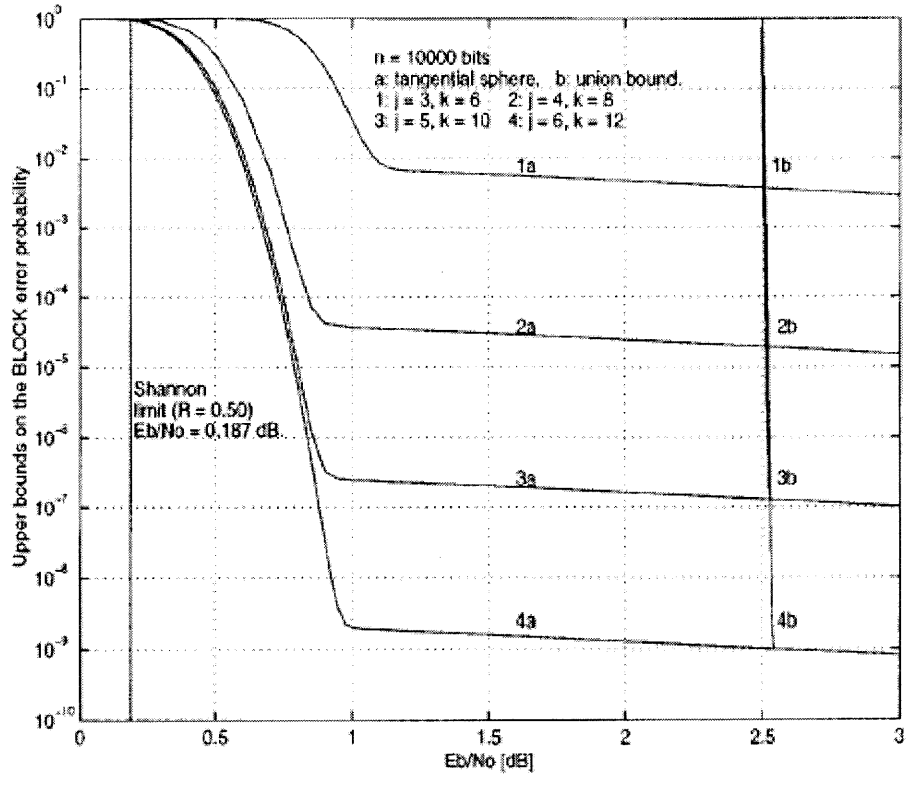
However, upper bounds derived from code weight spectrum represent the performance achievable by ML decoding, which is impractical for long code such as turbo code and LDPC code. Actually, simulations yield a totally different result. For $j \geq 3$, denser parity check decreases BER performance. A concatenated code with good distance spectrum does not guarantee a good BER performance under iterative decoding. There exists a convergence threshold γ_b^*. Only for $E_b/N_0 > \gamma_b^*$ will the decoding algorithm render error-free performance. This threshold has *no* direct connection with the code distance spectrum. Rather, it depends on the iterative decoding algorithm itself. It represents iterative decoding capacity.

3.2 Iterative Decoding Capacity

Iterative decoding is actually belief propagation (BP) on graphical model. If the graph is loopless, then BP algorithm would converge to the exact APP results for each bit. Then an interesting question is: what would happen if the underlying graph for turbo code or LDPC code is cycle-free? Several groups of researchers [34, 35, 36] set out from this question. They take different approaches and finally arrive at a common conclusion: there exists an iterative decoding capacity. This capacity has



(a)



(b)

Figure 3.2 Upper bounds on block error rates of ML decoding [33]

nothing to do with the code distance spectrum. Instead, it depends on the iterative decoding algorithm itself.

3.2.1 Density Evolution

The idea behind density evolution is very simple. The decoder is viewed as a signal processing system. Its input, $\mathbf{L} = [L_n]$, are independent random variables with Gaussian distributions. Its output, extrinsic information $\xi_{mn}^{(i)}$ is also a random variable with certain pdf. Through iterations, the pdf of $\xi^{(i)}$ evolves. This pdf is predictable since the signal processing system and the input pdf are known. Under the cycle-free assumption, it is feasible to calculate the pdf of $\xi^{(i)}$, for the left sides of both equation(3.6) and equation (3.7) are independent random variables. This simple and clear method for determining the capacity of iterative decoder is called density evolution. Exact threshold could be derived through this approach.

The threshold value, which represents a threshold channel parameter, defines the boundary of error-free region. For an AWGN channel, this parameter could be SNR per information bit denoted by $\gamma_b = E_b/N_0$. By the general concentration theorem [34], for almost all codes in a code ensemble and for almost all inputs, the error probability approaches zero through iterations if $E_b/N_0 > \gamma_b^*$, where the threshold value γ_b^* is derived by the density evolution procedures.

3.2.2 Parameter Evolution

The density evolution algorithm could be simplified by approximating the pdf of extrinsic information by the normal distribution. The iteration of the decoding algorithm is modeled as a simple one-parameter dynamical system. Quite accurate performance predictions can be made with this technique. Currently, two approaches are employed. One uses the mutual information as an evolving parameter while the other uses SNR of extrinsic information. These two methods have been successfully

applied to turbo code. In the following section, their effectiveness with respect to LDPC code is evaluated.

3.3 Capacity of Log-MAP Decoding

There exist two approximate approaches to estimate the capacity of turbo code. They are developed independently by two groups of researchers [35, 37]. They are based on the same idea: convert the infinite-dimensional problem of iteratively calculating the pdf of extrinsic information, which is needed to find out the capacity, to a one-dimensional problem of updating a parameter of the distribution. For the convenience of description, they are termed SNR evolution and mutual information evolution respectively.

For LDPC code, exact capacity could be achieved via density evolution. However, for most other turbo-like codes, exact density (pdf) of the extrinsic information is hard to be derived. Therefore, SNR evolution and mutual information evolution, which are based on Monte Carlo simulations, might be good choices. In this section, SNR evolution and mutual information evolution are used to estimate the capacity of LDPC code and the results are compared with the exact thresholds derived via density evolution. The basic conclusion is, although they are based on some rough assumptions, the results are in good agreement with the exact threshold. Mutual information evolution produces slightly better result.

3.3.1 Assumptions

- *Cycle-free Assumption:* So far, all methods for estimating the iterative decoder capacity are based on cycle-free assumption, which means that the graphical model is loopless. In density evolution, the cycle-free assumption makes it feasible to calculate the exact pdf evolution. In Monte-Carlo-simulation-based

methods, it is equivalent to the assumption that the intrinsic information and extrinsic values are pair-wise independent.

- *Gaussian Distribution Assumption:* In simulation-based methods, the extrinsic information ξ_{mn} is assumed to be Gaussian distributed. Then, the estimation of the exact pdf in density evolution is simplified to estimating the Gaussian parameters: σ and μ.

- $\mu_\xi = \sigma_\xi^2/2$*:* It is observed that in Log-MAP decoding of turbo code and LDPC code, the above equation stands. Later, it is revealed that in LDPC code, symmetry condition is preserved under density evolution through iterations for all extrinsic information, which is expressed as $f(x) = e^x f(-x)$, where $f(x)$ is the pdf. By enforcing this condition to Gaussian distribution, the equation $\mu_\xi = \sigma_\xi^2/2$ is obtained.

- *All-zero Codeword:* Without losing any generality, it is assumed that all-zero codeword is transmitted.

3.3.2 SNR Evolution

SNR evolution was first proposed by Gamal [35]. The main idea is to view the essential action of the constituent decoders as enhancing the SNR of the extrinsic information.

The input to a constituent decoder is written as,

$$\zeta_{mn}^{(i)} = L_n + \sum_{m' \in M(n)\ m} \xi_{m'n}^{(i)} \tag{3.9}$$

where $\mu_{L_n} = \sigma_{L_n}^2/2$ and $\mu_{\xi_{mn}^{(i-1)}} = \sigma_{\xi_{mn}^{(i-1)}}^2/2$. Therefore, it is easy to show that

$$SNR(\zeta_{mn}) = SNR(L_n) + \sum_{m' \in M(n)\backslash m} SNR(\xi_{m'n}) \tag{3.10}$$

Actually, SNR of extrinsic information fully represents its statistical character since $SNR = \mu^2/\sigma^2 = \mu/2$. By symmetry , $SNR(\xi_{mn}^{(i)})$ is the same for any m and n. Let's

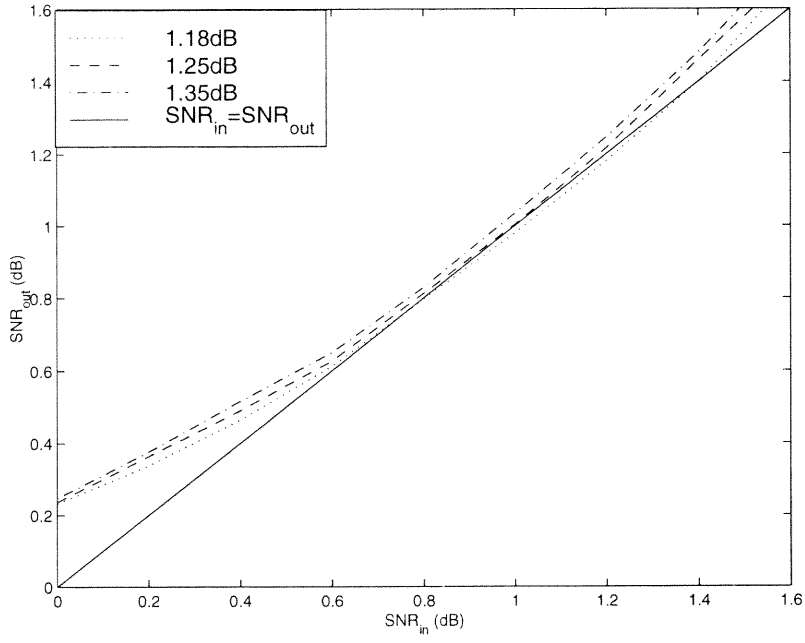


Figure 3.3 SNR evolution

consider the sum of extrinsic information $\sum_{m' \in M(n) \backslash m} \xi_{m'n}^{(i)}$ and denote its SNR at the input of iteration i as $S_n^{(i)}(v)$, where $v = SNR(L_n)$ denotes the decoder initial condition. Then the sequence $\{S_n^{(i)}(v)\}$ evolves recursively

$$S_n^{(0)} = 0$$
$$S_n^{(i)} = f_n(S_n^{(i-1)}(v), v) \tag{3.11}$$

f_n is identical for any n. The f_n depends on constituent decoder (in regular LDPC code, the parameters j and k). It was proved in [35] that the sequence $\{S_n^{(i)}(v)\}_{i=0}^{\infty}$ either has an accumulation point $\tau(v) < \infty$ or is unbounded. $\tau(v)$ is a non-decreasing function of the initial SNR v. There exists a threshold ν so that if $v < \nu$, $S_n^{(i)}(v)$ would converge to some $\tau(v)$; otherwise, $S_n^{(i)}(v)$ approaches infinity. At this point, it is possible to characterize the extrinsic information SNR input/output relation of the basic decoder and to determine if the iterative decoding process will converge or not at any E_b/N_0. Actually, it is possible to rely on simple Monte Carlo simulations to estimate the threshold ν.

In Figure 3.3, the relations between the extrinsic information SNR_{in} and SNR_{out} under different bit-energy-to-noise ratio E_b/N_0 for regular (3,6) LDPC code are demonstrated. Obviously,the SNR of decision variable $L_n^{(i)}$ would go to infinity if and only if $S_n^{(i)}$ goes to infinity. The bit error rate $P_e^{(i)}$ is simply the Q-function of SNR. To guarantee that $P_e^{(i)} \to 0$, the SNR_{in}/SNR_{out} characteristic should not intersect with $SNR_{in} = SNR_{out}$. In this way, the capacity of regular (j,k) LDPC code under AWGN channel and BP-based MAP decoding is obtained, as shown in Table 3.2.

3.3.3 Mutual Information Evolution

In this approach, mutual information is used as a parameter to characterize the density distribution. Stephan Brink is the first to use the concept of mutual information to describe convergence behavior of parallel concatenated convolutional codes (PCCC) [37]. Here this method is used to evaluate the the capacity of LDPC code.

The a priori information to a check node m is defined as $A_n^{(i)}$,

$$A_n^{(i)} = \sum_{m' \in M(n) \backslash m} \xi_{m'n}^{(i-1)} \tag{3.12}$$

To simplify notations, the symbol A is used for $A_n^{(i)}$, and x for x_n. The information contents of a priori knowledge could be measured by the average mutual information $I_A = I(X; A)$ between transmitted systematic bit X and the L-value A.

$$I_A = \frac{1}{2} \sum_{x=0,1} \int_{-\infty}^{+\infty} p_A(a|X = x) \log_2 \frac{2p_A(a|X = x)}{p_A(a|X = 0) + p_A(a|X = 1)} da \tag{3.13}$$

With the assumption that A is Gaussian distributed and $\mu_A = \sigma_A^2/2$, I_A only depends on σ_A,

$$I_A(\sigma_A) = 1 - \int_{-\infty}^{+\infty} \frac{e^{-\frac{(a-\sigma_A^2/2)^2}{2\sigma_A^2}}}{\sqrt{2\pi}\sigma_A} \log_2(1 + e^{-a}) da \tag{3.14}$$

The output extrinsic information for stage i is actually the input to a check node m for stage $i+1$,

$$E_n^{(i)} = \sum_{m' \in M(n) \backslash m} \xi_{m'n}^{(i)} \tag{3.15}$$

Similarly, the average mutual information $I_E = I(X;E)$ is written as,

$$I_E = \frac{1}{2} \sum_{x=0,1} \int_{-\infty}^{+\infty} p_E(\varepsilon|X=x) \log_2 \frac{2p_E(\varepsilon|X=x)}{p_E(\varepsilon|X=0) + p_E(\varepsilon|X=1)} d\varepsilon \tag{3.16}$$

Remember that in SNR evolution, both the a priori information and output extrinsic information are assumed Gaussian so that it is quite easy to plot the SNR transfer chart. Here, if the output extrinsic information is assumed to be Gaussian, the results should be the same as that of SNR evolution. Actually, it is no longer assumed to be Gaussian. p_E is estimated by Monte Carlo simulations. To measure I_E through the actual histogram of p_E is a more accurate approach.

Brink's original work deals with the iterative decoding of parallel concatenated convolutional code, where two constituent decoders work in turn to update the extrinsic information [37]. Accordingly, two I_A/I_E pairs, each corresponding to one constituent decoder, are plotted in the same figure to demonstrate the decoding trajectory. It is named Extrinsic Information Transfer Chart (EXIT chart). In Log-MAP algorithm for LDPC code, each check node is associated with a decoder and they are activated simultaneously. Therefore, the definitions of A and E are different from Brink's concept. Here A is defined as a priori information and E, output extrinsic information. The I_A/I_E relation is the same for every iteration. To find out the decoding trajectory, I_A/I_E and I_E/I_A are plotted in the same figure and a zigzag-path is drawn into the chart. As depicted in Figure 3.4, when the E_b/N_0 is not high enough, the tunnel is closed at some point so that the zigzag-path ended there. For the iterative decoder to converge, the E_b/N_0 must be high enough so that the trajectory survives the bottleneck region. This threshold represents the capacity of iterative decoding for regular LDPC code.

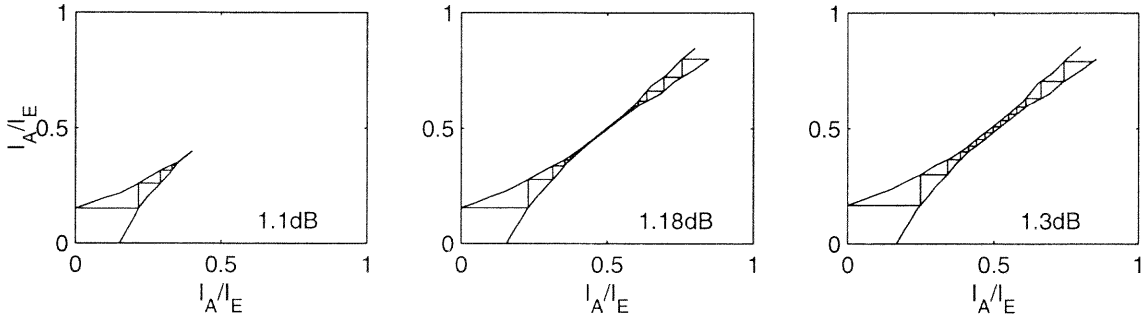


Figure 3.4 EXIT chart for (3,6) LDPC code

Table 3.2 Threshold values of Log-MAP decoding for AWGN channel

j	k	rate	threshold in dB		
			mutual inf. evolution	SNR evolution	density evolution
3	6	0.5	1.18	1.25	1.11
4	8	0.5	1.63	1.69	1.62
5	10	0.5	2.10	2.17	2.04

Table 3.2 compares the threshold derived from SNR evolution and mutual information evolution. The threshold obtained through density evolution is used as a benchmark to evaluate the two schemes.

3.3.4 Discussion

Considering the relatively rough assumptions for parameter-based evolution, the results obtained are surprisingly close to the exact thresholds. Figure 3.5 shows a real pdf of extrinsic information obtained from histogram and a Gaussian approximation. The difference is obvious.

Mutual information evolution is more accurate than SNR evolution because it is more robust against the changes in the shape of the a priori input distribution p_A. The following simulation demonstrates the robustness of average mutual information. Let a priori information A be a Gaussian r.v. with parameters $\mu_A = \sigma_A^2/2$. E is the output extrinsic information from regular (3,6) LDPC Log-MAP decoder. Let a priori information A' be the exact output of decoding iteration 1. E' is the corresponding

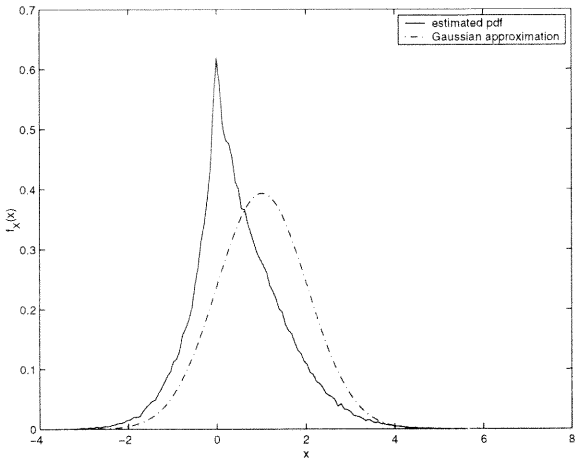


Figure 3.5 The real pdf of extrinsic information in comparison with Gaussian approximation

output of decoding iteration 2. Assume that $J(\sigma_A) = I_{A'}(A'; X)$ stands. That is, they have the same amount of information content on X.

$I_E(E; X)$ and $I_{E'}(E'; X)$ are compared in Figure 3.6. Although the distributions of A and A' are different, as shown in Figure 3.5, $I_E(E; X) \simeq I_{E'}(E'; X)$ given that $J(\sigma_A) = I_{A'}(A'; X)$. That's why mutual information evolution yields a quite accurate result in evaluating the capacity. Actually, A and E represent the same expression, while A is assumed to be Gaussian and E is not. It seems like a paradox, but this makes it possible to conveniently derive the EXIT chart with little degradation in accuracy.

In Brink's work, EXIT chart is not limited to turbo decoding analysis. The concept of mutual information is also used in iterative de-mapping and decoding, iterative channel estimation and decoding. Given the effectiveness of EXIT chart for LDPC code, it is expected to be an appropriate tool for "turbo system" with applications on LDPC code.

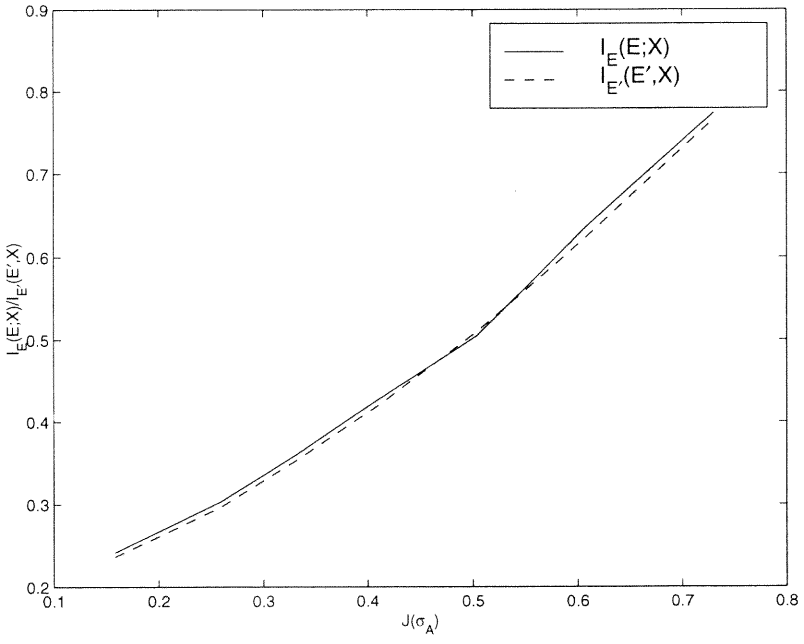


Figure 3.6 Average mutual information derived from real a priori information and Gaussian approximation

3.4 Capacity of Max-Log-MAP Decoding

3.4.1 Max-Log-MAP Decoding

As a simplified version of Log-MAP algorithm, the Max-Log-MAP algorithm is based on the following approximation,

$$\ln(e^x + e^y) = \max(x, y) + \ln(1 + e^{-|x-y|}) \simeq \max(x, y). \tag{3.17}$$

Applying this to horizontal pass,

$$\xi_{mn}^{(i)} = 2\tanh^{-1}\Big(\prod_{n' \in N(m)\ n} \tanh(\zeta_{mn'}^{(i-1)}/2)\Big) \tag{3.18}$$

the approximated extrinsic information is derived as follows,

$$\xi_{mn}^{(i)} = \min_{n' \in N(m)\backslash n} |\zeta_{mn'}^{(i-1)}| \prod_{n' \in N(m)\backslash n} \mathrm{sgn}(\zeta_{mn'}^{(i-1)}) \tag{3.19}$$

This equation was suggested in Hagenauer's milestone paper [32]. Ping Li derived it for single parity product code using a different approach [38]. An obvious advantage

of the Max-Log-MAP algorithm is that it reduces computation burden. Simple operations such as "$|\cdot|$" and "min" are used to replace "tanh" and "$\tanh^{-1}$". Another important feature lies in the fact that SNR estimation is not required. In the Log-MAP algorithm, the initial input value $L_{ch}y_n$ relies on both the channel value y_n and the channel estimation L_{ch}. A poorly estimated L_{ch} may result in some performance degradation.

For the convenience of description, the horizontal pass is re-written as,

$$\xi^{(i)} = \min_{K=1}^{k-1} |\zeta_K^{(i-1)}| \prod_{K=1}^{k-1} \mathrm{sgn}(\zeta_K^{(i-1)}) = g(\zeta_1^{(i-1)}, \zeta_2^{(i-1)}, \cdots, \zeta_{k-1}^{(i-1)}) \tag{3.20}$$

and the vertical pass as,

$$\zeta^{(i)} = \xi_0 + \sum_{J=1}^{j-1} \xi_J^{(i)} = f(\xi_0, \xi_1^{(i)}, \xi_2^{(i)}, \cdots, \xi_{j-1}^{(i)}) \tag{3.21}$$

It is easy to show that function $g(\cdot)$ and $f(\cdot)$ satisfy,

$$\begin{aligned} g(\alpha x_1, \alpha x_2, \cdots, \alpha x_n) &= \alpha g(x_1, x_2, \cdots, x_n) \\ f(\alpha x_1, \alpha x_2, \cdots, \alpha x_n) &= \alpha f(x_1, x_2, \cdots, x_n) \end{aligned} \tag{3.22}$$

provided that $\alpha \geq 0$. Therefore, if ξ_0 is initialized as $L_{ch}y$, $\zeta^{(i)}$ and $\xi^{(i)}$ would be proportional to L_{ch}. The value of L_{ch} does not affect the decoding results. Therefore, in the Max-Log-MAP algorithm, SNR estimation is not required. The initial input to decoder is simply set to the received channel values y.

The Max-Log-MAP algorithm is a typical suboptimal decoding algorithm. It has been heavily analyzed and simulated in turbo decoding. However, for LDPC code, most attention is focused on sum-product algorithm, which is equivalent to Log-MAP algorithm. To the author's knowledge, there have been no publications mentioning the Max-Log-MAP algorithm, though the derivation is quite straightforward. Fossorier proposed a reduced complexity decoding scheme for LDPC code [39]. It does not require SNR estimation and is named universal most powerful

(UMP) BP-based algorithm. It is almost the same as the Max-Log-MAP, except that ξ_0 is set to $|y|$ instead of y.

Currently, all the capacity estimations, no matter for LDPC code or turbo code, using Log-MAP decoder. There has not been any work on evaluating the capacity of Max-Log-MAP algorithm.

It has been hoped that parameter-evolution would work for Max-Log-MAP algorithm. In fact, the pdf of extrinsic information under Max-Log-MAP looks even more like Gaussian. However, notice that symmetry condition is no longer satisfied [34], μ_ξ could no longer be approximated by $\sigma_\xi^2/2$. To avoid working on the actual pdf, some efforts were made to parameterize the pdf so that parameter evolution might be employed to demonstrate the convergence behavior, just as in Log-MAP decoding. Unfortunately, these efforts failed to yield any useful results. There is no other way but to rely on the last resort–density evolution.

3.4.2 Density Evolution Procedure for Max-Log-MAP Decoding

Given the pdf of initial values $\zeta^{(0)}$ as $P^{(0)}$, and the decoding algorithm, theoretically, it is possible to compute the pdf of $\xi^{(i)}$ and $\zeta^{(i)}$ for any $i \geq 1$, denoted as $Q^{(i)}$ and $P^{(i)}$ respectively. Through observing the evolution of $P^{(i)}$ and $Q^{(i)}$, it is easy to find out if the algorithm would converge to the correct coded bit or not. The problem focuses on developing a numerical procedure to compute $Q^{(i)}$ and $P^{(i)}$.

For vertical pass, it is quite easy to evaluate the pdf of $\zeta^{(i)}$ given the pdf of ξ_0 and $\xi_J^{(i)}$,

$$\zeta^{(i)} = \xi_0 + \sum_{J=1}^{j-1} \xi_J^{(i)} \tag{3.23}$$

Since ξ_0 and $\{\xi_J^{(i)}\}_1^{j-1}$ are pair-wise independent, which is guaranteed by cycle-free assumption, the pdf of $\zeta^{(i)}$ is nothing else but the convolution of the pdf of individual random variables on the right side,

$$P^{(i)}(x) = f_{\xi_0}(x) \otimes Q^{(i)}_{\xi_1^{(i)}}(x) \otimes Q^{(i)}_{\xi_2^{(i)}}(x) \otimes \cdots Q^{(i)}_{\xi_{j-1}^{(i)}}(x) \tag{3.24}$$

Note that as a function, $Q^{(i)}_{\xi^{(i)}_J}(x)$ can be denoted as $Q^{(i)}(x)$, since $\{\xi^{(i)}_J\}_1^{j-1}$ are identical independently distributed (i.i.d.) random variables.

Now the problem left to be solved is stated as: how to compute the pdf of ξ, given that $\{\zeta_K\}_1^{k-1}$ are i.i.d. random variables with known pdf,

$$\xi = \min_{K=1}^{k-1} |\zeta_K| \prod_{K=1}^{k-1} \mathrm{sgn}(\zeta_K) \tag{3.25}$$

If a random variable Z is defined as the function of several random variables $Z = g(X_1, X_2, \cdots, X_n)$ and the joint pdf of $\mathbf{X}$ is known, the cumulative distribution function (cdf) of Z is found by,

$$\begin{aligned} F_Z(z) &= \Pr[g(X_1, X_2, \cdots, X_n) \leq z] \\ &= \int\int \cdots \int_{g(X_1, X_2, \cdots, X_n) \leq z} f_{X_1, X_2, \cdots, X_n}(x_1, x_2, \cdots, x_n) dx_1 dx_2 \cdots dx_n \end{aligned} \tag{3.26}$$

The pdf of Z is found by taking derivative of $F_Z(z)$ [40].

Here the actual joint pdf $f_{\zeta_1, \zeta_2, \cdots \zeta_{k-1}}(x_1, x_2, \cdots, x_{k-1})$ is simply the product of individual pdf's $P_{\zeta_1}(x_1) P_{\zeta_2}(x_2) \cdots P_{\zeta_{k-1}}(x_{k-1})$. Therefore, the cdf of ξ, denoted as $F_\xi(z)$, is written as,

$$F_\xi(z) = \int\int \cdots \int_{\mathrm{sign}(x_1, x_2, \cdots, x_{k-1}) \min_{K=1}^{k-1} |x_K| \leq z} P_{\zeta_1}(x_1) \cdots P_{\zeta_{k-1}}(x_{k-1}) dx_1 \cdots dx_{k-1} \tag{3.27}$$

where function $\mathrm{sign}(x_1, x_2, \cdots, x_{k-1})$ is define as $\prod_{K=1}^{k-1} \mathrm{sgn}(x_K)$.

To make things less complicated, define

$$\begin{aligned} \hat{F}_\xi(z) &= \Pr[g(Z_1, Z_2, \cdots, Z_{k-1}) \leq z] \quad \text{for } z \leq 0 \\ \tilde{F}_\xi(z) &= \Pr[g(Z_1, Z_2, \cdots, Z_{k-1}) > z] \quad \text{for } z > 0 \end{aligned} \tag{3.28}$$

Obviously,

$$F_\xi(z) = \begin{cases} \hat{F}_\xi(z) & z \leq 0 \\ 1 - \tilde{F}_\xi(z) & z > 0 \end{cases} \tag{3.29}$$

$P_{\zeta_1}(x)$,$P_{\zeta_2}(x)$,$\cdots$,$P_{\zeta_{k-1}}(x)$ are identical functions and could be denoted as $P(x)$. In equation 3.27, variables x_1,x_2,$\cdots$,x_{k-1} are equally positioned. Without losing any

generality, x_1 is assumed to be the outermost integral variable,

$$\begin{aligned}\hat{F}_\xi(z) &= \binom{k-1}{1}\Big[\int_{-\infty}^{z} P(x_1)\int\int\cdots\int_{R_1} P(x_2)P(x_3)\cdots P(x_{k-1})dx_2dx_3\cdots dx_{k-1}dx_1 \\ &\quad + \int_{-z}^{+\infty} P(x_1)\int\int\cdots\int_{R_2} P(x_2)P(x_3)\cdots P(x_{k-1})dx_2dx_3\cdots dx_{k-1}dx_1\Big] \quad (3.30)\end{aligned}$$

and

$$\begin{aligned}\tilde{F}_\xi(z) &= \binom{k-1}{1}\Big[\int_{z}^{+\infty} P(x_1)\int\int\cdots\int_{R_3} P(x_2)P(x_3)\cdots P(x_{k-1})dx_2dx_3\cdots dx_{k-1}dx_1 \\ &\quad + \int_{-\infty}^{-z} P(x_1)\int\int\cdots\int_{R_4} P(x_2)P(x_3)\cdots P(x_{k-1})dx_2dx_3\cdots dx_{k-1}dx_1\Big] \quad (3.31)\end{aligned}$$

where the integral regions are:

$$\begin{aligned}R_1: &\quad |x_2| > -x_1, |x_3| > -x_1, \cdots, |x_{k-1}| > -x_1, \text{sign}(x_2, x_3, \cdots, x_{k-1}) = 1 \\ R_2: &\quad |x_2| > x_1, |x_3| > x_1, \cdots, |x_{k-1}| > x_1, \text{sign}(x_2, x_3, \cdots, x_{k-1}) = -1 \\ R_3: &\quad |x_2| > x_1, |x_3| > x_1, \cdots, |x_{k-1}| > x_1, \text{sign}(x_2, x_3, \cdots, x_{k-1}) = 1 \\ R_4: &\quad |x_2| > -x_1, |x_3| > -x_1, \cdots, |x_{k-1}| > -x_1, \text{sign}(x_2, x_3, \cdots, x_{k-1}) = -1 \quad (3.32)\end{aligned}$$

Define

$$\begin{aligned}\psi_{-1}(x_1) &\triangleq \int\int\cdots\int_{R_2} P(x_2)P(x_3)\cdots P(x_{k-1})dx_2dx_3\cdots dx_{k-1} \\ \psi_{+1}(x_1) &\triangleq \int\int\cdots\int_{R_3} P(x_2)P(x_3)\cdots P(x_{k-1})dx_2dx_3\cdots dx_{k-1} \quad (3.33)\end{aligned}$$

so that for $z \leq 0$,

$$\hat{F}_\xi(z) = \binom{k-1}{1}\Big[\int_{-\infty}^{z} P(x_1)\psi_{+1}(-x_1)dx_1 + \int_{-z}^{+\infty} P(x_1)\psi_{-1}(x_1)dx_1\Big] \quad (3.34)$$

and for $z > 0$,

$$\tilde{F}_\xi(z) = \binom{k-1}{1}\Big[\int_{z}^{+\infty} P(x_1)\psi_{+1}(x_1)dx_1 + \int_{-\infty}^{-z} P(x_1)\psi_{-1}(-x_1)dx_1\Big] \quad (3.35)$$

Take derivative of $F_\xi(z)$ with respect to z, the pdf Q is obtained as,

$$Q_\xi(z) = \begin{cases} (k-1)[P(z)\psi_{+1}(-z) + P(-z)\psi_{-1}(-z)] & z \leq 0 \\ (k-1)[P(-z)\psi_{-1}(z) + P(z)\psi_{+1}(z)] & z > 0 \end{cases} \quad (3.36)$$

To evaluate ψ_{-1} and ψ_{+1}, it is convenient to define,

$$\begin{aligned}\phi_+(z) &\triangleq \int_{|z|}^{+\infty} P(x)dx \\ \phi_-(z) &\triangleq \int_{-\infty}^{-|z|} P(x)dx\end{aligned} \tag{3.37}$$

ψ^{+1} and ψ^{-1} is written as,

$$\begin{aligned}\psi^{+1}(z) &= \frac{[\phi_+(z)+\phi_-(z)]^{k-2}+[\phi_+(z)-\phi_-(z)]^{k-2}}{2} \\ \psi^{-1}(x_1) &= \frac{[\phi_+(z)+\phi_-(z)]^{k-2}-[\phi_+(z)-\phi_-(z)]^{k-2}}{2}\end{aligned} \tag{3.38}$$

From equations (3.39) and (3.36), the pdf of ξ is found as,

$$\begin{aligned}& Q^{(i+1)}(\xi) \\ = \;& \frac{k-1}{2}[(P^{(i)}(\xi)+P^{(i)}(-\xi))(\phi_+^{(i)}(\xi)+\phi_-^{(i)}(\xi))^{k-2} \\ & +(P^{(i)}(\xi)-P^{(i)}(-\xi))(\phi_+^{(i)}(\xi)-\phi_-^{(i)}(\xi))^{k-2}]\end{aligned} \tag{3.39}$$

In Figure 3.7, the pdf of extrinsic information $\xi^{(1)}$ is demonstrated under input E_b/N_0 of 1.2dB and regular (3,6) LDPC code. Solid line represents pdf derived from the density evolution algorithm. Dashed line represents pdf estimated from histogram of Monte Carlo simulation. The two lines agree with each other. This guarantees that the numerical procedure developed is correct.

Given the decision variable $L^{(i)}$,

$$L^{(i)} = \xi_0 + \sum_{J=1}^{j} \xi_J^{(i)} \tag{3.40}$$

it is easy to find out its pdf $D_L^{(i)}(x)$,

$$D_L^{(i)}(x) = f_{\xi_0}(x) \otimes Q_{\xi^{(i)}}^{(i)}(x)^{\otimes j} \tag{3.41}$$

.

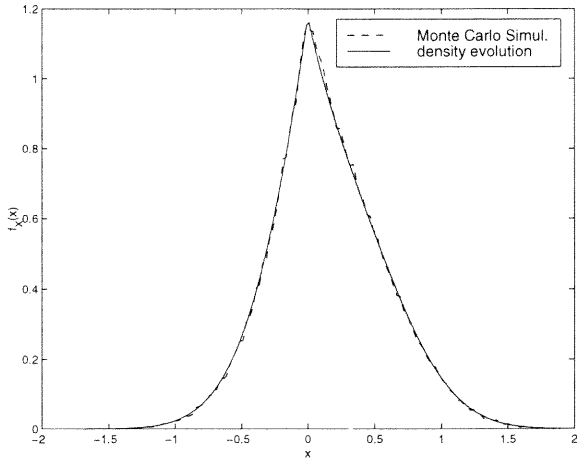


Figure 3.7 pdf of extrinsic information $\xi^{(1)}$

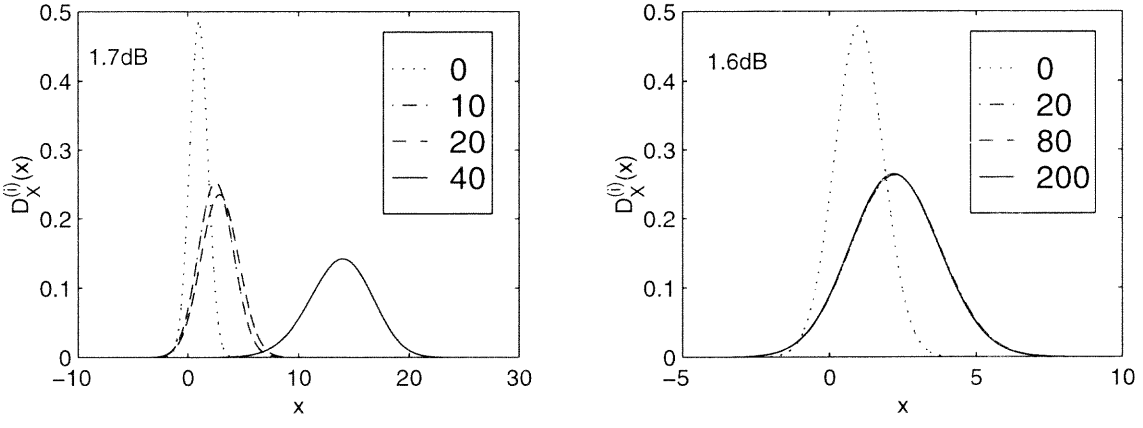


Figure 3.8 Visualized pdf evolution for Max-Log-MAP decoding.
left: E_b/N_0 above the threshold; right: E_b/N_0 below the threshold

3.4.3 Numerical Results

By convention, it is assumed that all-zero codeword is transmitted. Therefore, positive $L^{(i)}$ signifies correct decoding while negative $L^{(i)}$ means errors. Define $p_e^{(i)}$ as bit error probability at iteration i. Using density evolution, $p_e^{(i)}$ is estimated as,

$$p_e^{(i)} = \int_{-\infty}^{0} D_L^{(i)}(x)dx \tag{3.42}$$

If E_b/N_0 is higher than a threshold γ_b^*, then the iterative decoder would converge to the correct codeword such that $\lim_{i\to\infty} p_e^{(i)} = 0$;otherwise, $\lim_{i\to\infty} p_e^{(i)} = \delta > 0$. Figure 3.8 demonstrates a visualized evolution of pdf $D_L^{(i)}(x)$. It is a regular

Table 3.3 Threshold values of Max-Log-MAP and Log-MAP decoding for AWGN channel

j	k	rate	thresholds in dB	
			Max-Log-MAP	Log-MAP (sum-product)
3	6	0.5	1.7	1.11
4	8	0.5	2.5	1.62
5	10	0.5	3.1	2.04

(3,6) LDPC code, under Max-Log-MAP decoding and AWGN channel. When the input E_b/N_0 is higher than the threshold, which is 1.7 dB, the pdf would evolve until the part below zero diminishes. If the input E_b/N_0 is less than the threshold, no matter how many iterations, the pdf $D_L^{(i)}(x)$ would never evolve to above zero.

The thresholds for regular (3,6), (4,8) and (5,10) LDPC codes under Max-Log-MAP decoding are given in Table 3.3, in comparison with thresholds for Log-MAP algorithm. Previous simulations on turbo code suggest a gap of 0.5 dB between Max-Log-MAP and Log-MAP algorithm. Interesting enough, here for regular (3,6) LDPC, the gap is approximately 0.5 dB!

In Figure 3.9, the threshold is compared with simulated BER performance assuming information block length of 10080. Theoretically, if the information block length approaches infinity, the threshold should be exactly the "waterfall" point. The gap between theoretical capacity and simulation results is explained as: 1. the block length is not large enough; 2. the code construction method could be further improved.

3.5 Summary

So far, the approaches to estimate decoder capacity fall into two classes: parameter-evolution which is simulation-based and yields approximate result, and density-evolution which yields exact threshold but only applies to LDPC code. Using density evolution as the benchmark, two approximate methods, SNR-evolution and

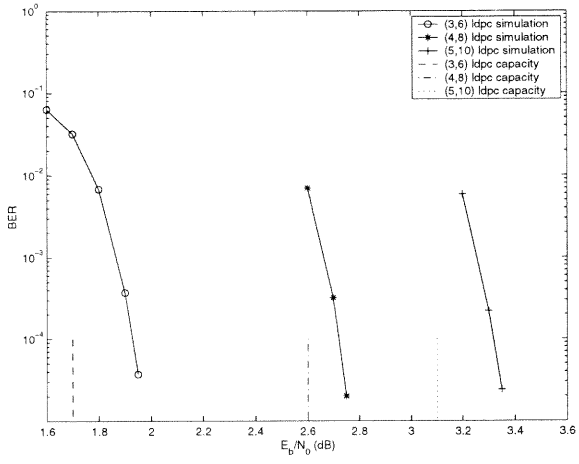


Figure 3.9 Theoretical capacity in comparison with simulated BER performance of Max-Log-MAP decoding

mutual information evolution, are evaluated and found to be quite accurate. To the author's knowledge, current researches on decoding capacity always assume Log-MAP decoder. The Max-Log-MAP decoder, a suboptimal one with reduced complexity, has never been considered. Density evolution procedure for Max-Log-MAP decoding is derived. The following conclusions are established, based on this work and the most recent research results [9, 34, 35, 36, 37],

1. For message-passing iterative decoding and memoriless channel, there exists a threshold γ_b^* determined by the decoder itself other than the code weight spectrum. The threshold could be illustrated as the boundary channel parameter for iterative decoding to yield error-free performance. Therefore, it represents the capacity of iterative decoding. If the actual channel parameter is "better" than the threshold (in AWGN channel, this means E_b/N_0 high enough), the probability of error would converge to zero as the number of iterations tends to infinity. This explains why upper bounds derived from code weight spectrum favor (5,10) regular LDPC code while simulation results favor (3,6) regular LDPC code.

2. For Log-MAP decoding, parameter evolution offers quite an accurate capacity estimation for LDPC code, despite the rough assumptions. Mutual information evolution scheme is more accurate than SNR evolution.

3. Compared with Log-MAP algorithm, Max-Log-MAP decoding not only reduces complexity, but also eliminates the requirement for channel parameter estimation.

4. A numerical procedure for density evolution under the Max-Log-MAP decoding is developed. Using this tool, the capacity of LDPC code could be easily computed.

CHAPTER 4

QUANTIZED DECODING FOR LOW-DENSITY PARITY-CHECK CODE

LDPC codes have certain advantages for implementations, such as fully parallelizable decoder structures. When one deals with implementation problems on a general purpose DSP or dedicated hardware, the performance loss due to finite precision must be taken into consideration. Previous research work revealed that uniform quantization should be imposed on messages represented by log-likelihood ratio symbol set rather than probability symbol set [41]. To the author's knowledge, no further progress has been made on quantized implementation of LDPC decoder.

This chapter is organized as follows. Section 4.1 introduces the quantized iterative decoding models, for both Log-MAP and Max-Log-MAP algorithms. Section 4.2 describes the discretized density evolution procedure in detail. Note that it is different from Richardson's original procedure [34]. The derived threshold value represents the capacity of quantized iterative decoding under certain idealized conditions. The capacity depends on quantization resolution as well as dynamic range. In Section 4.3, general LDPC decoder with infinite precision is discussed. The influence of clipping limit on convergence speed and BER performance is investigated. In Section 4.4, the issue of choosing a dynamic range for quantized iterative decoder is discussed. Quantization loss under a fixed dynamic range is evaluated. Some conclusions are summarized in Section 4.5.

4.1 Quantized Decoder Models

So far, the messages in decoding LDPC code could be in the form of probability, probability ratio or log-likelihood ratio. In sum-product (SP) algorithm, the message on a bit is a pair of probabilities p_{-1},p_1 satisfying $p_{-1} + p_1 = 1$. Such a pair can also be represented by the corresponding likelihood ratio (probability ratio) $\frac{p_1}{p_{-1}}$ or

log-likelihood ratio (LLR) $\ln \frac{p_1}{p_{-1}}$. Probability ratio symbol set was used by Ping [41] and the equivalent decoding procedure was named parity likelihood ratio (PLR) algorithm. The LLR symbol set was adopted by the density evolution procedure [34]. The equivalent sum-product algorithm in LLR symbol set is termed as Log-MAP algorithm.

SP, PLR and Log-MAP algorithms render the same BER performance under infinite precision. However, concerning sensitivity to quantization effects, Log-MAP algorithm is preferred. Quantized SP algorithm suffers severe BER performance degradation. PLR algorithm was originally proposed to overcome this problem [41], where the quantization operation is imposed in the exponential domain and thus results in non-uniform quantization levels s^i, $i = 0, \pm 1, \pm 2, \cdots$. If LLR symbol set is employed, the messages would be uniformly quantized as $i \ln s$, $i = 0, \pm 1, \pm 2, \cdots$. In this sense, Log-MAP algorithm is a more natural choice for quantized implementations. In addition, in the logarithmic domain it is easy to apply the density evolution technique to derive theoretical capacities, as detailed in the next section.

Log-MAP algorithm for LDPC code is conceptually identical to the famous Log-MAP decoder for turbo codes. Each parity check node functions as a component decoder extracting extrinsic information. Therefore, just as in turbo code, a suboptimal algorithm named Max-Log-MAP decoding is derived by substituting each "log-exponential" operation with "max" operation [14]. Max-Log-MAP decoding not only reduces the computational burden, but it also eliminates the requirement of channel parameter estimation. Both of these decoding procedures are considered in this study.

In designing a quantized iterative decoder, one faces the problem of quantization scope. That is, to quantize only the received channel values and allow higher precision for intermediate results, as Wu and Woerner did [42], or to use the same number of bits to represent all quantities involved in the decoding process. Surely,

the latter models the fixed-point implementations more accurately. In addition, the performance degradation due to limited internal precision is non-negligible. Therefore, in this study, quantization is imposed on both external and internal values.

4.1.1 Log-MAP Decoder

4.1.1.1 Basic notations. The notations are identical to that in the last chapter. For clarity, they are repeated as follows. Let $\mathbf{x} = [x_n]$ and $\mathbf{H} = [H_{mn}]$ be the codeword and the parity check matrix, respectively, of an LDPC code, such that $\mathbf{Hx} = \mathbf{0}$, where $x_n \in \{0, 1\}$. The received normalized symbol $y_n = (1 - 2 * x_n) + z_n$ is corrupted by an additive white Gaussian noise (AWGN) z_n, with zero mean and noise power σ^2. Then, the LLR of channel transition probability associated with coordinate n is given as,

$$L_n = \ln \frac{p(y_n|x_n = 0)}{p(y_n|x_n = 1)} = \frac{2}{\sigma^2} y_n. \tag{4.1}$$

Denote the set of bits that participate in check m by $N(m) = \{n : H_{mn} = 1\}$. Similarly, the set of checks in which bit n participates is denoted as $M(n) = \{m : H_{mn} = 1\}$. A set $N(m)$ with bit n excluded is denoted by $N(m)\backslash n$, and a set $M(n)$ with parity check m excluded by $M(n)\backslash m$. The decoding process proceeds by activating the check nodes and variable nodes in turns and updating messages ξ_{mn} and ζ_{mn}. ξ_{mn} is defined as the extrinsic information extracted from the check node m and to be passed to the variable node n. ζ_{mn} is the message from the variable node m to the check node n.

4.1.1.2 Decoding procedures under infinite precision. The Log-MAP decoding algorithm involves the following steps.

1. *Initialization:* Initially, set the input to the check nodes as LLR of channel transition probability $\zeta_{mn}^{(0)} = L_n$.

2. *Horizontal pass:* It is carried out in the check nodes to extract extrinsic information ξ_{mn}.

$$\xi_{mn}^{(i)} = 2\tanh^{-1}\Big(\prod_{n'\in N(m)\backslash n} \tanh(\zeta_{mn'}^{(i-1)}/2)\Big) \tag{4.2}$$

3. *Vertical pass:* It is carried out in the variable nodes to prepare for the next decoding cycle.

$$\zeta_{mn}^{(i)} = L_n + \sum_{m'\in M(n)\backslash m} \xi_{m'n}^{(i)} \tag{4.3}$$

4. *Decision stage:* For each coordinate n, combine all extrinsic information from neighboring check nodes $M(n)$ and channel value L_n to get decision variable $D^{(i)}(\hat{x}_n)$,

$$D^{(i)}(\hat{x}_n) = L_n + \sum_{m\in M(n)} \xi_{mn}^{(i)}. \tag{4.4}$$

The decision is given by $\hat{\mathbf{x}} = [\hat{x}_n]$ such that $\hat{x}_n = 0$ if $D^{(i)}(\hat{x}_n) > 0$; otherwise $\hat{x}_n = 1$. If $\hat{\mathbf{x}}$ is a valid codeword satisfying $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$, then the algorithm halts; otherwise, the horizontal and vertical passes are repeated until some maximal number of iterations is reached without a valid decoding.

4.1.1.3 Quantization. Consider applying the same quantization scheme to input values L_n and internal variables ξ_{mn} and ζ_{mn}. Set the quantization levels to $i\Delta$, $i = 0, \pm1, \pm2, \cdots, \pm(2^{q-1} - 1)$, where q denotes the number of bits to represent a value. Step width Δ depends on the dynamic range V_{lim} and q,

$$\Delta = V_{\text{lim}}/(2^{q-1} - 1). \tag{4.5}$$

Denote the quantization operation upon x as $Q(x)$,

$$x' = Q(x) = \begin{cases} V_{\text{lim}} & x > V_{\text{lim}} \\ -V_{\text{lim}} & x < -V_{\text{lim}} \\ \lfloor \frac{x}{\Delta} + \frac{1}{2} \rfloor \Delta & -V_{\text{lim}} \le x \le V_{\text{lim}} \end{cases} \tag{4.6}$$

where $\lfloor x \rfloor$ is the largest integer no greater than x. The operation $Q(\cdot)$ actually consists of two operations, clipping and discretizing.

In real world fixed-point implementation, x' is further converted to integer representation, which essentially yields the same performance as the above model. This research emphasizes the influence of quantization so that no further effort was made toward integer representation.

Now one faces the problem of discretizing the four types of decoding operations. In the initialization step, simply use quantized LLR $L'_n = Q(L_n)$. Discretized horizontal pass is described as follows. Define a two-input operator $\mathcal{H}_2^{\text{map}}$ as,

$$\mathcal{H}_2^{\text{map}}(a,b) \triangleq 2 * \tanh^{-1}(\tanh(a/2) * \tanh(b/2)). \tag{4.7}$$

It is easy to show that the horizontal pass, which involves $d_c - 1$ quantized inputs ζ'_1,ζ'_2, $\cdots$,ζ'_{d_c-1}, could be represented by identical nesting pair-wise operations,

$$\xi = \mathcal{H}_2^{\text{map}}(\zeta'_1, \mathcal{H}_2^{\text{map}}(\zeta'_2, \cdots, \mathcal{H}_2^{\text{map}}(\zeta'_{d_c-2}, \zeta'_{d_c-1})\cdots)). \tag{4.8}$$

In general, if a and b are quantized to l levels, $\mathcal{H}_2^{\text{map}}$ could fall on any of $l(l+1)/2$ non-uniform levels. Hence $Q(\cdot)$ operation is indispensable,

$$\mathcal{H}'^{\text{map}}_2(a,b) \triangleq Q(\mathcal{H}_2^{\text{map}}(a,b)) \tag{4.9}$$

Replacing $\mathcal{H}_2^{\text{map}}$ by $\mathcal{H}'^{\text{map}}_2$ in equation (4.8), the discretized horizontal pass is derived as,

$$\xi' = \mathcal{H}'^{\text{map}}_2(\zeta'_1, \mathcal{H}'^{\text{map}}_2(\zeta'_2, \cdots, \mathcal{H}'^{\text{map}}_2(\zeta'_{d_c-2}, \zeta'_{d_c-1})\cdots)). \tag{4.10}$$

Note that generally $\xi' \neq \xi$ even though equations (4.10) and (4.8) take identical inputs. Moreover, different nesting pattern might produce slightly different ξ'. However, equation (4.10) still corresponds to some valid discretized decoding scheme and ξ' would approach ξ as $\Delta \to 0$. More significantly, all intermediate results in equation (4.10) are quantized according to the rule $Q(\cdot)$ such that the implementation is simplified to recursive table-lookup. In contrast, intermediate results in equation (4.8) fall on large number of levels. It makes table-lookup impractical.

In the vertical pass and decision stage, ζ and D are the sum of several discretized values with the identical quantization rule $Q(\cdot)$ so that they themselves are uniformly discretized. Still, a clipping operation, denoted as $\zeta' = Q(\zeta)$ and $D' = Q(D)$, is required to prevent an overflow. If the summation is implemented by a recursive pair-wise summation, say $\mathcal{S}_2(a, b) = a+b$, more bits have to be allowed for representing intermediate variables. How much extra bit width is required depends on the maximum variable node degree d_v.

4.1.2 Max-Log-MAP Decoder

Max-Log-MAP decoding approximates the optimal Log-MAP algorithm by substituting each "log-exponential" operation with a "max" operation,

$$\ln(e^x + e^y) = \max(x, y) + \ln(1 + e^{-|x-y|}) \simeq \max(x, y) \tag{4.11}$$

Max-Log-MAP algorithm for LDPC codes is summarized in four steps very similar to Log-MAP algorithm, as described in the author's paper [14]. There are two differences. One is, Max-Log-MAP algorithm directly takes channel value y_n, instead of L_n, as its input. Therefore, channel parameter estimation, such as noise level σ, is not required. The other is, horizontal pass is simplified as,

$$\xi_{mn}^{(i)} = \min_{n' \in N(m) \backslash n} |\zeta_{mn'}^{(i-1)}| \prod_{n' \in N(m) \backslash n} \operatorname{sgn}(\zeta_{mn'}^{(i-1)}). \tag{4.12}$$

The remaining two steps are completely identical with that of Log-MAP decoding.

Similar to the Log-MAP decoding, the horizontal pass of Max-Log-MAP decoding could be implemented by a nesting pair-wise computation,

$$\xi' = \mathcal{H}'^{\max}_2(\zeta'_1, \mathcal{H}'^{\max}_2(\zeta'_2, \cdots, \mathcal{H}'^{\max}_2(\zeta'_{d_c-2}, \zeta'_{d_c-1}) \cdots)) \tag{4.13}$$

where the operation $\mathcal{H}'^{\max}_2$ is defined as,

$$\mathcal{H}'^{\max}_2(a, b) \triangleq \min(|a|, |b|) * \operatorname{sgn}(a) * \operatorname{sgn}(b) \tag{4.14}$$

Note that here the quantization operation $Q(\cdot)$ is unnecessary. Similarly, discretized horizontal pass could be implemented through a recursive table-lookup. The other three types of operations are discretized in the same way as in the Log-MAP decoding.

4.2 Theoretical Capacity

The average behavior of a message-passing decoder for an LDPC code ensemble is numerically computable using the newly developed technique called density evolution [34]. Assuming the code length to be infinite and underlying graphical model to be cycle-free, one could calculate a threshold value, that is, a threshold channel parameter that defines the boundary of error-free region. For an AWGN channel, this parameter could be SNR per information bit denoted by $\gamma_b = E_b/N_0$. By the general concentration theorem [34], for almost all codes in a code ensemble and for almost all inputs, the error probability approaches zero through iterations if $E_b/N_0 > \gamma_b^*$, where the threshold value γ_b^* is derived by the density evolution procedure. γ_b^* represents the capacity of iterative decoding.

In Richardson's original paper [34], the density evolution procedure for LDPC code under Log-MAP decoding (belief-propagation based decoding) is developed in the context of continuous message alphabets. As indicated in the paper, quantization performance could be evaluated using discrete message alphabets. Therefore, the first part of this section descibes the procedures for discretized density evolution.

In message-passing decoding with quantized messages, the threshold γ_b^* depends on two parameters, namely dynamic range V_{lim} and quantizing resolution q, as denoted by $\gamma_b^*(V_{\text{lim}}, q)$. In other words, the capacity of quantized iterative decoder of resolution q relies on the dynamic range V_{lim}. In the remaining part of this section, numerical results about capacity γ_b^* are reported.

4.2.1 Discretized Density Evolution Procedures

Discretized density evolution procedure was first proposed by Chung in his Ph.D dissertation [43]. It models the exact behavior of quantized iterative decoder for LDPC code. The general procedure described below applies to both Log-MAP and Max-Log-MAP decoding. For the latter, a more efficient procedure is available.

4.2.1.1 General procedure. Denote the probability mass function (pmf) of a quantized message m' by $p_{m'}(k) = \Pr[m' = k\Delta]$, where $k = 0, \pm 1, \pm 2, \cdots, \pm(2^{q-1} - 1)$. If the pmf of the initial message $\zeta^{(0)}$ is known, the problem is reduced to how to calculate the pmf of the output message, say $p_{\xi'}$ under horizontal pass and $p_{\zeta'}$ under vertical pass. Let $c' = \mathcal{R}_2(a', b')$ denote the quantized message with the same alphabet as a' and b', where $\mathcal{R}_2(a', b')$ could be any of the pair-wise operations involved, $\mathcal{H}'^{\text{map}}_2(a', b')$,$\mathcal{H}'^{\text{max}}_2(a', b')$ or $\mathcal{S}_2(a', b')$. Given the pmf of a' and b', the pmf of c' is derived as,

$$p_{c'}(k) = \sum_{(i,j):k\Delta=\mathcal{R}_2(i\Delta,j\Delta)} p_{a'}(i)p_{b'}(j). \tag{4.15}$$

Under the special case of $c' = a' + b'$, the pmf $p_{c'}$ is exactly the convolution summation of $p_{a'}$ and $p_{b'}$.

For the horizontal pass, apply equation (4.15) recursively to get $p_{\xi'}$. For the vertical pass, first find the pmf p_ζ via multiple convolutions, then merge the bins exceeding the dynamic range into the boundary bin as follows,

$$p_{\zeta'}(k) = \begin{cases} \sum_{j=2^{q-1}-1}^{+\infty} p_\zeta(j) & k = 2^{q-1} - 1 \\ \sum_{-\infty}^{j=-2^{q-1}+1} p_\zeta(j) & k = -2^{q-1} + 1 \\ p_\zeta(k) & -2^{q-1} + 1 < k < 2^{q-1} - 1 \end{cases} \tag{4.16}$$

4.2.1.2 Procedure for Max-Log-MAP decoding. The general procedure described above applies to Log-MAP and Max-Log-MAP decoding. To implement it, the horizontal pass must be treated as nested pair-wise operations. For Max-Log-MAP decoding, the pmf of ξ' in the horizontal pass could be derived in a more straight

forward way. Similarly to the procedure for continuous alphabet in Section 3.4.2, the pmf of ξ' is calculated by differentiation.

Define the cumulative mass function (cmf) as,

$$F_{\xi'}(k) = \Pr[\min(|\zeta_1'|, |\zeta_2'|, \cdots, |\zeta_{d_c-1}'|) \prod_{i=1}^{d_c-1} \mathrm{sgn}(\zeta_i') \leq k\Delta] \tag{4.17}$$

where k corresponds to quantization levels $k = 0, \pm 1, \pm 2, \cdots, \pm(2^{q-1}-1)$. The pmf is found by taking differentiation,

$$p_{\xi'}(k) = \begin{cases} F_{\xi'}(k) - F_{\xi'}(k-1) & k = 0, \pm 1, \pm 2, \cdots, \pm(2^{q-1}-2), 2^{q-1}-1 \\ F_{\xi'}(k) & k = -(2^{q-1}-1) \end{cases} \tag{4.18}$$

Define two cumulative functions for $0 < k \leq 2^{q-1} - 1$,

$$\begin{aligned} \phi_+(k) &\triangleq \Pr[\zeta' \geq k\Delta] = \sum_{i=k}^{+\infty} p_{\zeta'}(i) \\ \phi_-(k) &\triangleq \Pr[\zeta' \leq -k\Delta] = \sum_{i=-\infty}^{-k} p_{\zeta'}(i). \end{aligned} \tag{4.19}$$

They are set to 0 for any k out of the region. Then the cmf $F_{\xi'}$ is numerically calculated as functions of ϕ_+ and ϕ_-. For $k < 0$,

$$\begin{aligned} F_{\xi'}(k) &= \Pr[\min(|\zeta_1'|, |\zeta_2'|, \cdots, |\zeta_{d_c-1}'|) \prod_{i=1}^{d_c-1} \mathrm{sgn}(\zeta_i') \leq k\Delta] \\ &= \sum_{\substack{i=0 \\ \text{odd}}}^{d_c-1} (\phi_-(-k))^i (\phi_+(-k))^{d_c-1-i} \\ &= \frac{1}{2}(\phi_+(-k) + \phi_-(-k))^{d_c-1} - \frac{1}{2}(\phi_+(-k) - \phi_-(-k))^{d_c-1}. \end{aligned} \tag{4.20}$$

For $k > 0$,

$$\begin{aligned} F_{\xi'}(k) &= 1 - \Pr[\min(|\zeta_1'|, |\zeta_2'|, \cdots, |\zeta_{d_c-1}'|) \prod_{i=1}^{d_c-1} \mathrm{sgn}(\zeta_i') > k\Delta] \\ &= 1 - \sum_{\substack{i=0 \\ \text{even}}}^{d_c-1} (\phi_-(k+1))^i (\phi_+(k+1))^{d_c-1-i} \\ &= 1 - \frac{1}{2}(\phi_+(k+1) + \phi_-(k+1))^{d_c-1} - \frac{1}{2}(\phi_+(k+1) - \phi_-(k+1))^{d_c-1} \end{aligned} \tag{4.21}$$

Combining the above two equations, together with the case for $k = 0$, it is clear that,

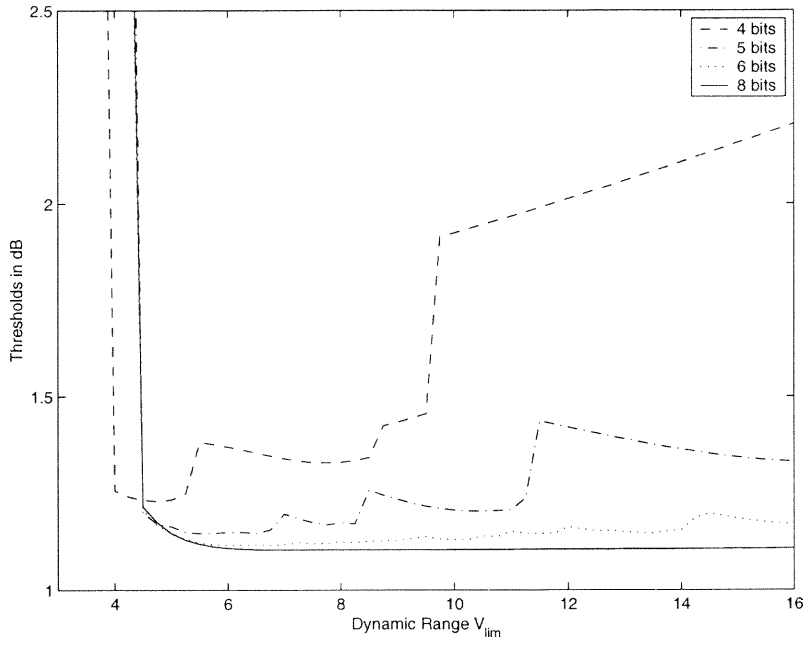
$$F_{\xi'}(k) = \begin{cases} \frac{1}{2}(\phi_+(-k) + \phi_-(-k))^{d_c-1} - \frac{1}{2}(\phi_+(-k) - \phi_-(-k))^{d_c-1} & k < 0 \\ 1 - (1 - p_{\zeta'}(0))^{d_c-1} + F_{\xi'}(-1) & k = 0 \\ 1 - \frac{1}{2}(\phi_+(k+1) + \phi_-(k+1))^{d_c-1} - \frac{1}{2}(\phi_+(k+1) - \phi_-(k+1))^{d_c-1} & k > 0 \end{cases} \tag{4.22}$$

Following equations (4.19),(4.22) and (4.18), the pmf of ξ', which is the output of the horizontal pass, could be computed. The procedure for the vertical pass is exactly the same as the one in Section 4.2.1.1.

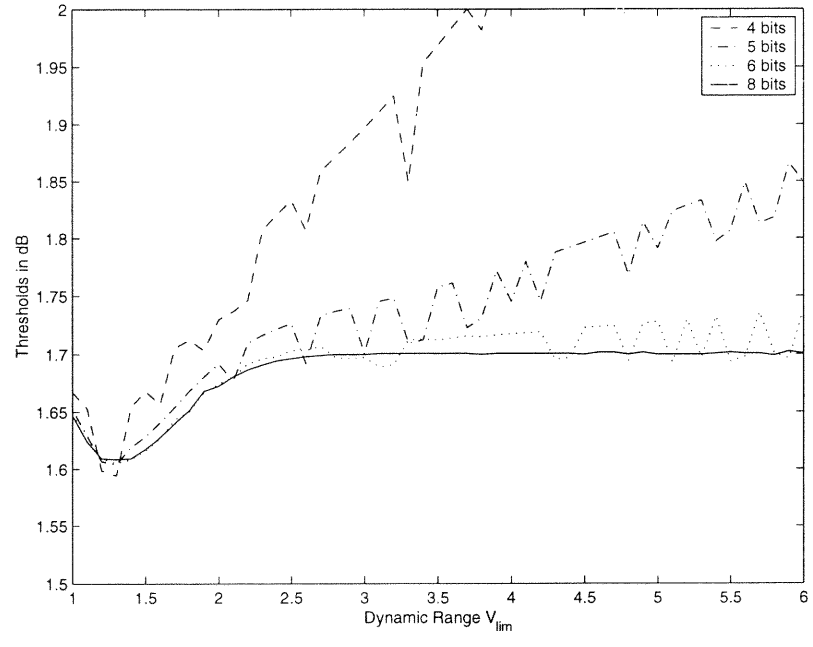
4.2.2 Numerical Results

Figure 4.1 (a) displays the capacities $\gamma_b^*(q)$ versus dynamic range V_{lim} under Log-MAP decoding, where $q = 4, 5, 6, 8$. Generally, a lower quantization resolution q renders higher threshold γ_b^*, which means decreased decoding capacity. The higher the quantizer resolution q, the less sensitive γ_b^* is to the dynamic range V_{lim} . That is, dynamic range tends to make little impact on γ_b^* under higher q, say $q = 8$. In contrast, under $q = 4$, dynamic range V_{lim} must be chosen carefully in order to maximize the capacity. The minimum threshold value under the 8-bit quantization scheme is about 1.11 dB. It is virtually the same as what can be achieved under infinite precision [34, 43]. When the quantization resolution is reduced to 4 bits, it is still possible to achieve a minimum threshold value as low as 1.24 dB. This implies that under infinite code length, quantized implementation with only 4 bits might perform very well, given that the dynamic range is properly chosen.

The threshold γ_b^* under Max-Log-MAP decoding is shown in Figure 4.1 (b). Note that the decoding input is y_n instead of $\frac{2}{\sigma_n^2} y_n$. It is different from Log-MAP decoding in that the optimum dynamic range is around 1.3 for any q considered and the minimum threshold derived is about 1.6 dB. This means that the capacity is maximized if dynamic range V_{lim} is set to 1.3. Like the Log-MAP decoding, the performance threshold γ_b^* depends more on dynamic range V_{lim} as the quantization resolution q decreases. For $q = 8$, the line is rather smooth in the range $V_{\text{lim}} > 1.5$



(a) Log-MAP



(b) Max-Log-MAP

Figure 4.1 Threshold $\gamma_b^*(q)$ for (3,6) LDPC code under AWGN channel

and shows a threshold of about 1.7 dB, a penalty of merely 0.1 dB over the lowest threshold. For the case of $q = 4$, γ_b^* goes up steeply as V_{lim} increases from 1.3. In summary, in order to maximize the decoding capacity in low resolution quantized decoders, dynamic range V_{lim} must be chosen carefully.

4.3 Clipping Effects

It is shown, in the last section, that dynamic range plays an important role in maximizing the decoding capacity when $q = 4$ or 5. However, it is not clear how dynamic range affects BER performance for LDPC codes with practical length. In this section, general unquantized Log-MAP decoder is investigated. To emphasize the fact that V_{lim} is an important design parameter for general iterative decoder, the term clipping limit is used instead of the term dynamic range.

As a matter of fact, the selection of clipping limit V_{lim} greatly influences the decoding performance of LDPC codes. Unlike turbo code, where generally no clipping operation is imposed on extrinsic information, LDPC decoder must be given a proper saturation point in order to yield good performance. Previous investigations on LDPC codes always used a good clipping limit for the decoder. However, this point has never been addressed in research papers.

The motivations for this study are summarized as follows. First, the different convergence behaviors of the density evolution procedure under different clipping limits give a better explanation of the role of clipping limit in a practical LDPC decoder. Second, quantization operation $Q(\cdot)$ could be viewed as a combination of two operations, clipping and discretization. It is natural to first investigate clipping effects. Third, the study of clipping effects would provide theoretical explanations on dynamic range selection.

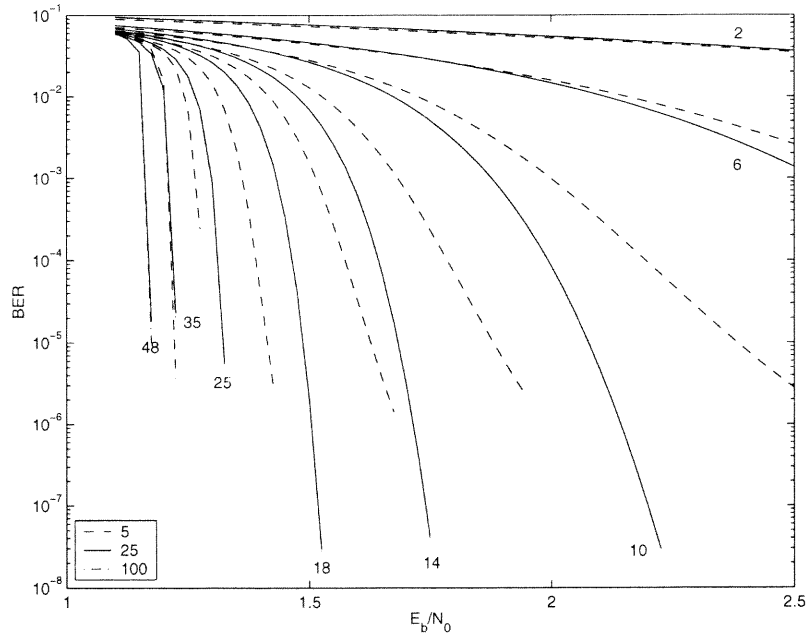


Figure 4.2 Convergence behavior predicted by density evolution

4.3.1 Convergence Speed

For infinite precision iterative LDPC decoder, increasing the clipping limit yields a constant decoding capacity. However, the convergence speed depends on the clipping limit. In fact, density evolution is also a power tool in describing the expected BER through decoding iterations. Set the E_b/N_0 to a certain value, bit error rate at decoding iteration i, denoted as $p_e^{(i)}$, is found out by equation (3.42). In an infinite precision decoder, $p_e^{(i)}$ depends on V_{lim} and E_b/N_0. The basic conclusion from density evolution is,

$$\lim_{i\to+\infty} p_e^{(i)} = \begin{cases} 0 & E_b/N_0 > \gamma_b^*(V_{\text{lim}}) \\ \delta > 0 & E_b/N_0 < \gamma_b^*(V_{\text{lim}}) \end{cases} \tag{4.23}$$

Specifically, numerical results show that $\gamma_b^*(5) = \gamma_b^*(25) = \gamma_b^*(100) = 1.1$ dB. This means that no matter what the V_{lim} is set to, say 5, 25 or 100, the BER $p_e^{(i)}$ would eventually converge to 0, given that $E_b/N_0 > 1.1$ dB.

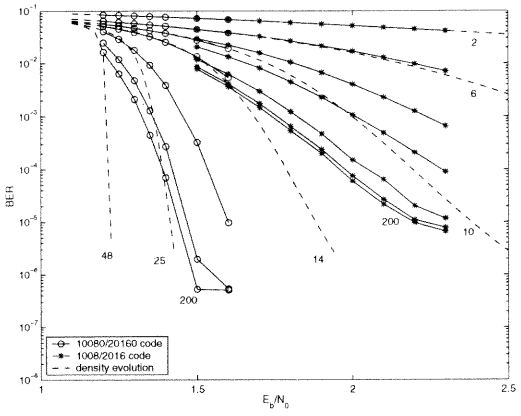
However, V_{lim} does affect the convergence speed of p_e, as shown in Figure 4.2. The dashed lines represent $p_e^{(i)}$ with V_{lim} set to 5. The solid lines correspond to $V_{\text{lim}} = 25$. The dash-dot lines correspond to $V_{\text{lim}} = 100$. The $p_e^{(i)}$ associated with

iterations $i = 2, 6, 10, 14, 18, 25, 35, 48$ are displayed in the Figure 4.2. It is quite clear that a larger number of iterations is required for p_e to converge below a certain value under lower dynamic range. That is, the lower the clipping limit V_{lim}, the slower the the convergence of the decoder. For $V_{\text{lim}} = 25$ and $V_{\text{lim}} = 100$, the convergence speeds are almost the same, such that the dash-dot lines overlap with the dashed lines.

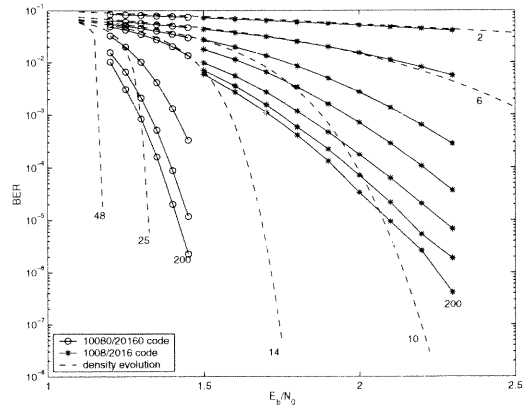
4.3.2 Clipping Effects in Practical Decoders

Suppose that an LDPC code is of infinite length such that the cycle length is large enough. In this case, $p_e^{(i)}$ computed by the density evolution procedure represents the exact BER convergence process. Clipping only affects the convergence speed. Eventually, $p_e^{(i)}$ would approach zero provided that $E_b/N_0 > 1.1$ dB. However, practical LDPC codes are of limited code length. The convergence behavior is expected to be the same as that predicted by density evolution only in the initial l iterations, if the graphical model defined by the code contains no loops of length up to $2l$. In order to find out the impact of cycles under different dynamic range V_{lim}, simulations are carried out on two (3,6) LDPC codes with the size of 1008/2016 and 10080/20160 respectively.

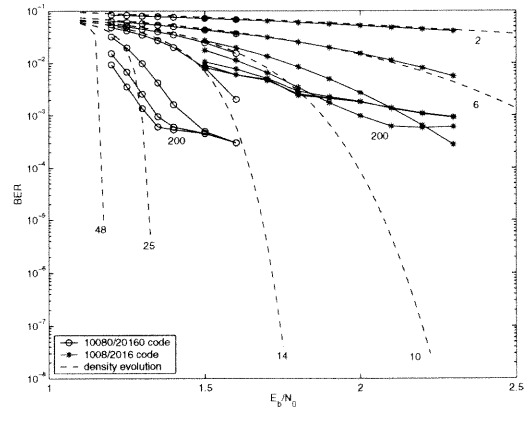
Figure 4.3 compares the convergence processes of actual LDPC decoding with that of density evolution. The BER curves of iteration 2, 6, 10, 14, 25, 48 and 200 are plotted in the figure. The bit error rates in the first several iterations are well predicted by density evolution, as expected. An apparent difference between 1008/2016 code and 10080/20160 code is, in the former the BER curve diverges from that of density evolution at iteration 10, while in the latter, divergence occurs at as late as iteration 25. In longer codes, the average loop length is longer such that cycle effects occur in later iterations. The longer the code length, the steeper the final BER curve is.



(a) $V_{\text{lim}} = 5$

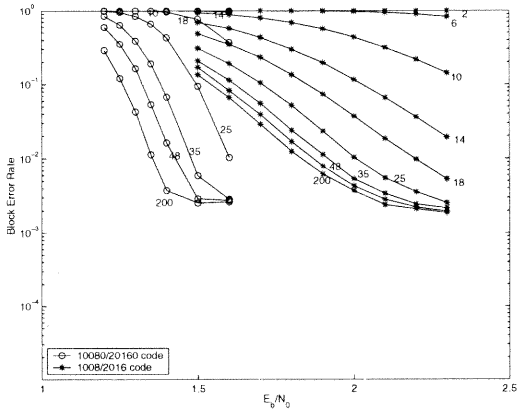


(b) $V_{\text{lim}} = 25$

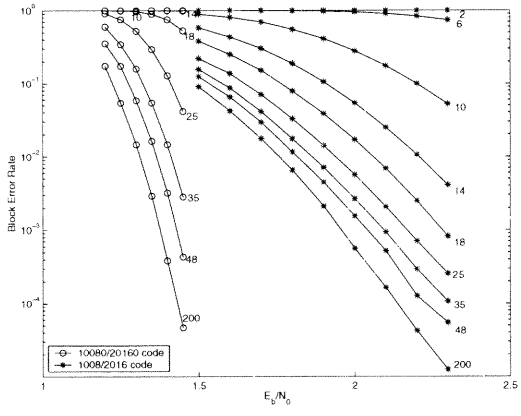


(c) $V_{\text{lim}} = 100$

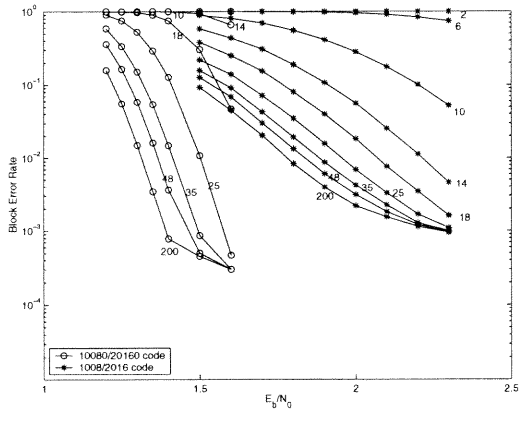
Figure 4.3 Convergence process of practical LDPC decoder:bit error rates



(a) $V_{\text{lim}} = 5$



(b) $V_{\text{lim}} = 25$



(c) $V_{\text{lim}} = 100$

Figure 4.4 Convergence process of practical LDPC decoder: block error rates

A proper clipping limit is a must in designing the LDPC decoder. Clipping limits which are too high or too low would result in severe BER performance degradation. Since the decoding proceeds block by block, the block error rates with respect to decoding iterations 2, 6, 10, 14, 18, 25, 35, 48 and 200 are compared in Figure 4.4. Clearly, the decoder with clipping limit $V_{\text{lim}} = 25$ outperforms the other two decoders.

If the clipping limit is too low, say $V_{\text{lim}} = 5$, the bit error rate decreases slowly through iterations, as shown in Figure 4.3. Cycle effects occur when bit error rate is still relatively high. In addition, the low clipping limit makes it difficult to correct some erroneous bits in following iterations. Therefore, the block error rate encounters an error floor at about 10^{-3}. On average, each erroneous block only contains 2-4 flipped bits at the error floor, but these errors could no longer be corrected due to the low clipping limit.

If the clipping limit is too high, say $V_{\text{lim}} = 100$, the block error rate converges in the same way as that of decoder with $V_{\text{lim}} = 25$ up to iteration 14. After that, some erroneous blocks converge in the wrong direction such that the bit error rate could be hardly improved or could even rises up through iterations. High clipping limit is even more detrimental at higher E_b/N_0, due to the severe dependency among extrinsic information associated with different bits, which is revealed in Section 6.2.1. Quite different from the case with $V_{\text{lim}} = 5$, almost all bits are flipped in an erroneous block when error floor is reached.

If the clipping limit V_{lim} is set to $+\infty$, that is, no clipping operation is imposed, the BER performance would be even worse than that of $V_{\text{lim}} = 100$. Therefore, it is reasonable to draw the following conclusion, that clipping is indispensable in LDPC decoder. Moreover, the clipping limit must be chosen carefully.

To the author's knowledge, iterative decoding for turbo codes generally does not require any clipping operation, though range-limiting does yield a slight improvement

on BER performance [44]. Note that LDPC decoder and turbo decoder are all based on belief propagation. However, the value of extrinsic information rises up faster in LDPC code. Theoretically speaking, in turbo code, soft input associated with every coded bit is involved in generating extrinsic information (that is the forward-backward algorithm). In contrast, for LDPC codes, extrinsic information is extracted only from those soft inputs associated with the same parity check node. Intuitively, turbo decoding is more stable.

4.4 Quantization Effects

The threshold derived through density evolution represents the pinch-off channel parameter for the infinite code length. It is shown in Section 4.2, that the capacity of quantized LDPC decoder depends on the dynamic range V_{lim}. Certain low dynamic ranges offer high capacity under low quantization resolution. However, as indicated in Section 4.3, for LDPC codes of practical length, the dynamic range V_{lim} offering maximal decoding capacity is not preferable because the corresponding decoder would suffer severe error floor. Therefore, choosing the dynamic range is an important issue in designing a quantized decoder.

In this section, first, the issue of choosing the dynamic range is addressed. Then, quantization loss is evaluated. In all simulations, a 1008/2016 regular (3,6) LDPC code is assumed.

4.4.1 Dynamic Range Selection

4.4.1.1 Infinite precision. Dynamic range V_{lim} needs to be chosen carefully, even in the case of no quantization. Most of the previous work on LDPC codes assumed V_{lim} to be around 25. This value is used as the high-end dynamic range to compare with other possible choices suggested by the density evolution.

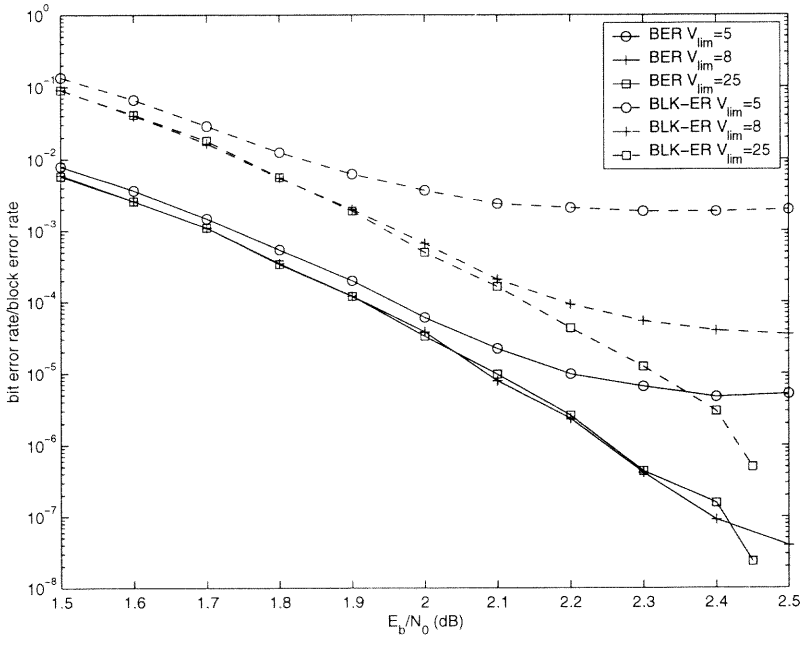
Figure 4.5 shows the influence of V_{lim} on block and bit error rates. Dashed lines represent block error rates, while solid lines represent BER. Under the Log-MAP decoding, a low dynamic range tends to yield a higher error floor, especially on the block error rate. However, concerning the BER above 10^{-7}, $V_{\text{lim}} = 8$ yields a performance almost as good as $V_{\text{lim}} = 25$.

The Max-Log-MAP decoder suffers similar error floor on block error rates, if the dynamic range is low. However, for the region with BER above 10^{-6}, the scheme with $V_{\text{lim}} = 1.3$ outperforms the one with $V_{\text{lim}} = 25$ and yields a gain of 0.1 dB on E_b/N_0.

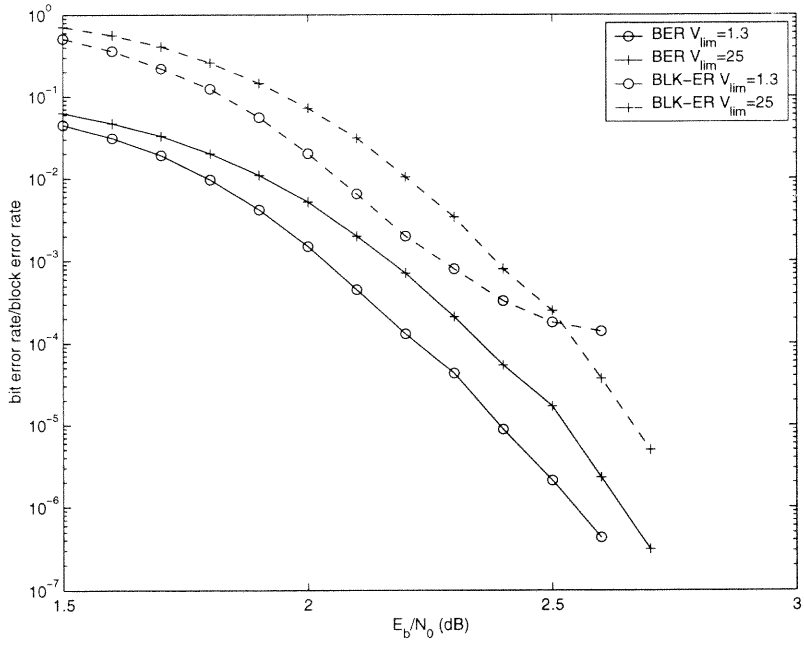
4.4.1.2 Quantized implementation. Figure 4.6 displays the influence of dynamic range in quantized schemes with $q = 4$. Similarly, dashed lines represent block error rates, while solid lines represent BER. The error floor is very similar to the that of unquantized schemes. This implies that error floor itself has nothing to do with the quantization operation, rather it should be connected with the low clipping limit.

For the Log-MAP algorithm, the capacity obtained through the density evolution suggests that for $q = 4$ schemes, $V_{\text{lim}} = 5$ is the optimum dynamic range and $V_{\text{lim}} = 8$ would suffer a penalty of 0.1 dB, while $V_{\text{lim}} = 10$ would suffer a further penalty of 0.6 dB. The simulation results reflect this point for BER less than 10^{-4}. However, the bit error rate curves of $V_{\text{lim}} = 5$ and $V_{\text{lim}} = 8$ intersect with each other at $E_b/N_0 = 2.3$. In order to achieve BER below 10^{-6}, $V_{\text{lim}} = 8$ is preferable.

Similarly, in the Max-Log-MAP decoding, lower dynamic range offers better performance under relatively low E_b/N_0. $V_{\text{lim}} = 1.3$, the optimum dynamic range suggested by density evolution, outperforms $V_{\text{lim}} = 3$ by about 0.2 dB at BER of 10^{-5}.

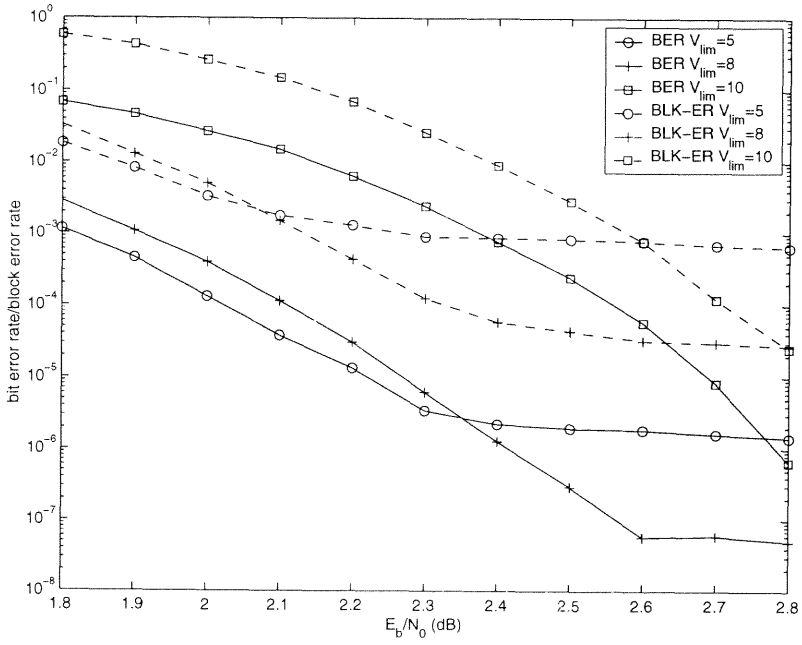


(a) Log-MAP

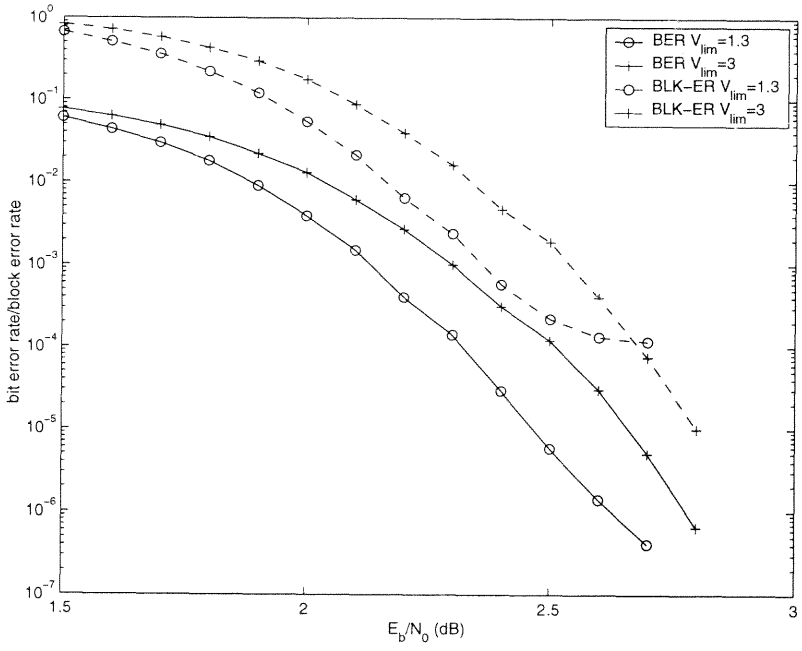


(b) Max-Log-MAP

Figure 4.5 Effects of dynamic range V_{lim} on decoding errors (infinite precision)



(a) Log-MAP



(b) Max-Log-MAP

Figure 4.6 Effects of dynamic range V_{lim} on decoding errors (4-bit quantization)

It is concluded that the threshold derived through the density evolution technique could be used as a reference in selecting the dynamic range for short LDPC code. However, the error floor effects due to low dynamic range must be taken into consideration, especially in short LDPC codes. The selection of dynamic range depends on the specific requirements of transmission quality, as well as the code length.

4.4.2 Quantization Loss

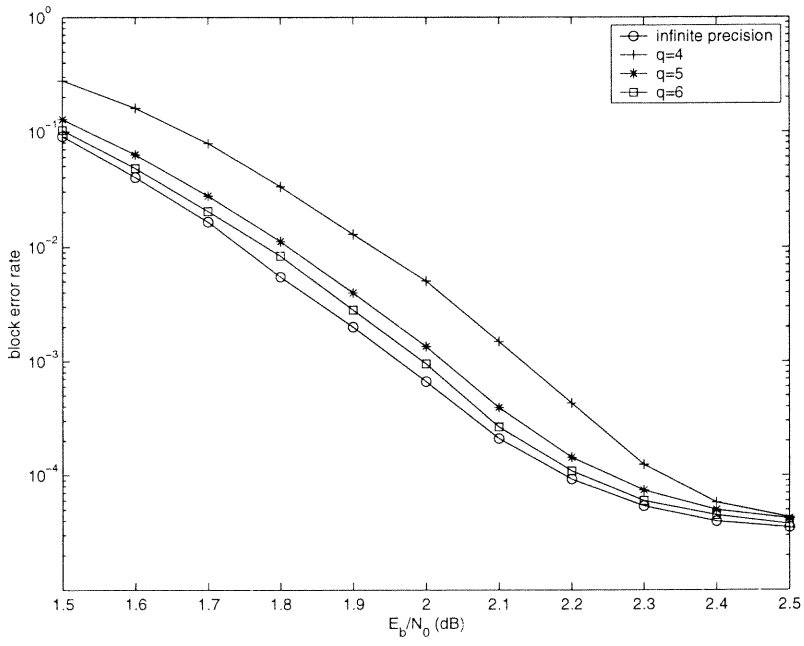
The E_b/N_0 penalty due to the quantization is evaluated under a fixed dynamic range. It is clear that the dynamic range V_{lim} plays an important role in the quantized schemes. Some of the previous work [41, 44] assumed different V_{lim} for the quantized and unquantized schemes. Such comparisons are somehow unfair.

Figure 4.7 highlights the quantization effects in the Log-MAP decoding with $V_{\text{lim}} = 8$. In order to achieve a BER of 10^{-5}, $q = 4$ quantization only requires about 0.2 dB higher E_b/N_0 than the infinite precision scheme. For $q = 5$, the gap is reduced to less than 0.1 dB.

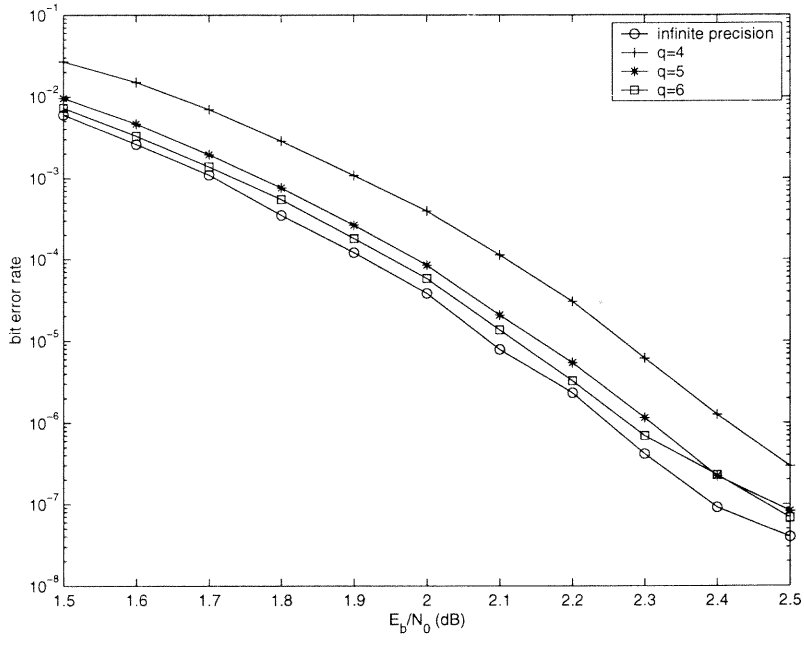
An earlier research assumed V_{lim} to be about 14 for $q = 4$ and found a penalty of up to 0.8 dB as well as a high error floor [41]. In contrast, this study reveals that the penalty could be significantly reduced to merely 0.2 dB by adopting a lower dynamic range.

Figure 4.8 displays the quantization effects under Max-Log-MAP decoding for $V_{\text{lim}} = 1.3$. The gaps between quantized and infinite precision schemes are even smaller than that of Log-MAP decoding. The penalty of $q = 4$ quantization over infinite precision is merely 0.1 dB.

Throughout the simulations, all errors made are detectable. That is, the decoder never converges to an incorrect but valid codeword. Generally, iterative

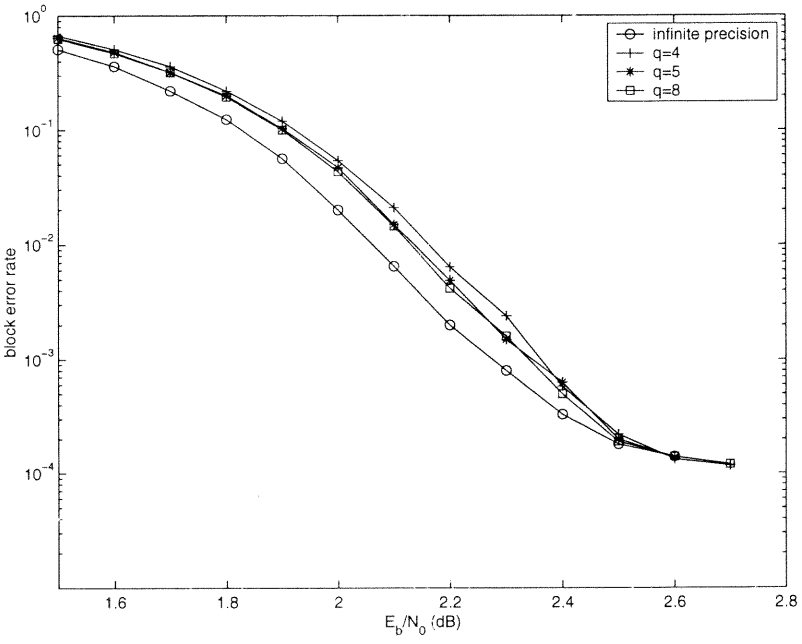


(a) Block error rate

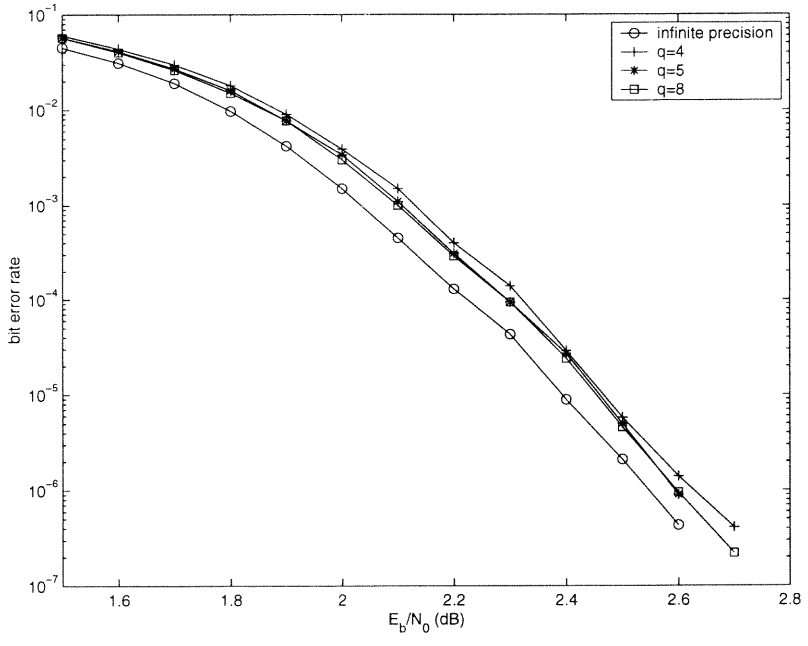


(b) Bit error rate

Figure 4.7 Quantization loss in Log-MAP decoding



(a) Block error rate



(b) Bit error rate

Figure 4.8 Quantization loss in Max-Log-MAP decoding

decoder works to correct errors, such as turbo decoder. For LDPC codes, those erroneous blocks beyond correction could be detected.

Discretized decoding avoids complicated computations by using a lookup table. For a quantization scheme of $q = 4$, the size of the table is only 15×15. This study shows that the price is remarkably small, only 0.1-0.2 dB higher E_b/N_0. It is found that the decoding model under consideration is very simple, with identical internal and external precision. Further optimizations are possible.

4.5 Summary

In this chapter, a systematic investigation on the design and analysis of the quantized decoder for LDPC codes is presented. The capacity of quantized LDPC decoder is numerically computed using density evolution. In addition, clipping effects and quantization effects are discussed. In brief, the following conclusions are drawn:

1. The key point in designing a quantized LDPC decoder is to pick a proper dynamic range V_{lim}. In choosing the dynamic range, two issues must be taken into consideration, theoretical capacity and clipping effects.

2. Unlike turbo code, where clipping operation is not required in the iterative decoding algorithm, LDPC code requires a carefully-chosen clipping limit in order to get good BER performance.

3. With an appropriate dynamic range, a 4-bit quantized scheme needs only 0.1-0.2 dB higher E_b/N_0 in order to achieve the same level of BER as the infinite precision implementation. A 4-bit quantized implementation implies that every quantity involved in the decoding process is represented by only 15 levels. The computations could be carried out by looking up a 15×15 table. The price is merely 0.2 dB extra E_b/N_0. This is quite an encouraging result for the practical fixed-point implementations.

4. Message-passing decoding for LDPC codes is verifiable in the sense that it has never been found to converge to an incorrect but valid codeword [6]. It only makes detectable errors. This wonderful property is preserved in the quantized decoding.

CHAPTER 5

LOW-DENSITY PARITY-CHECK CODE: SENSITIVITY TO SNR MISMATCH

Iterative decoding of low-density parity-check (LDPC) codes, which normally uses the maximum a posteriori (MAP) algorithm to estimate a posteriori probabilities, requires knowledge of signal-to-noise ratio (SNR) of the channel. In practical implementations, these statistics are usually estimated from raw channel measurements. It is not known how sensitive decoding error rate is to mismatch of channel SNR. In this chapter, the performance of iterative LDPC decoder in presence of SNR mismatch is investigated.

Theoretically speaking, it is necessary to estimate the SNR when using a MAP constituent decoder in any concatenated coding schemes including the well known turbo code. For turbo codes, the effects of an SNR mismatch on the bit error rate have been intensively investigated for additive white Gaussian noise (AWGN) channel and Rayleigh fading channels [45, 46, 47]. Some channel estimators for turbo codes are addressed by Summers and Valenti [45, 48] and they can be directly used in the context of LDPC codes. The question is, for LDPC codes, how much SNR mismatch can be tolerated? To the author's knowledge, there have been no papers published on this.

Conventional studies on SNR mismatch are mainly based on simulation results. In this research, the theoretical decoding capacity under a fixed level of channel SNR offset is computed using density evolution, as discussed in Section 5.1. Section 5.2 focuses on the effects of SNR mismatch on bit error rate. Some conclusions are given in Section 5.3.

5.1 Theoretical Capacity under SNR Offset

In this section, the capacity of Log-MAP decoding under a fixed SNR offset is investigated. This represents the case where the estimated channel SNR contains a constant bias.

5.1.1 System Model

Assume that binary phase-shift-keying (BPSK) transmission of coded bits is performed over AWGN channel. Under coherent demodulation along with perfect synchronization, the received data can be represented by

$$r_n = \sqrt{E_s}(1 - 2 * x_n) + n_n \tag{5.1}$$

where $x_n \in \{0, 1\}$ is the coded bit, n_n is a Gaussian random variable having zero mean and variance $\sigma^2 = N_0/2$, and the two-sided noise spectral density of the channel noise process is $N_0/2$ W/Hz. The symbol energy E_s is related to the energy per information bit by $E_s = E_b R$, where R is the code rate. Following the notation used in the previous chapters, the normalized quantity $y_n = r_n/\sqrt{E_s}$ is used as the channel value.

As indicated in Section 3.1.2, Log-MAP algorithm requires a channel estimation $L_{ch} = 2\mu^2/\sigma^2 = 4E_s/N_0$ to supply the proper combination of prior bit statistic in the initialization and vertical pass. Overstating L_{ch} has the qualitative effect of imbuing the channel measurements with more value than they deserve, while understating L_{ch} uses the a priori information about bits in too strong a manner.

Suppose that E_s is known and the noise level N_0 is to be estimated as a channel parameter. Denote the estimated channel SNR in dB as,

$$(E_s/N_0)^{\text{est}} = (E_s/N_0) + (E_s/N_0)^{\text{off}}. \tag{5.2}$$

where E_s/N_0 is the actual channel SNR and $(E_s/N_0)^{\text{off}}$, the channel SNR offset.

As illustrated in Section 3.4.1, Max-Log-MAP decoding does not require channel estimation. The performance of this suboptimal algorithm serves as a

bench mark to demonstrate to what extent the SNR mismatch affects the Log-MAP decoding.

5.1.2 Theoretical Capacity

Throughout this study, a rate 1/2 regular (3,6) LDPC code is assumed. Set the decoder input as $4((E_s/N_0) + (E_s/N_0)^{\text{off}})y_n$. Follow the density evolution procedure and the minimum E_b/N_0 to achieve error free decoding is obtained as a threshold $\gamma_b^*((E_s/N_0)^{\text{off}})$. In other words, the threshold γ_b^* represents the capacity of the iterative decoder under infinite code length.

5.1.2.1 Infinite Precision. The capacity γ_b^* under $(E_s/N_0)^{\text{off}}$ ranging from -3 dB to 3 dB is reported in Figure 5.1. The threshold γ_b^* reaches the lowest point when there is no SNR offset, as expected. A mismatch of -1 to 1 dB renders about 0.1 dB E_b/N_0 penalty. That is, under infinite code length, only 0.1 dB extra E_b/N_0 is required to achieve error free transmission due to the ± 1 dB gap between estimated and actual channel SNR. Notice that overestimation of SNR is less detrimental than underestimation. Log-MAP decoding outperforms Max-Log-MAP decoding unless the channel SNR is underestimated by more than 2.25 dB.

5.1.2.2 Finite Precision Decoding. In finite precision implementations, especially with low quantization resolutions, the threshold γ_b^* also depends on the dynamic range V_{lim}, as detailed in Chapter 4. Therefore, the robustness of quantized Log-MAP decoder to SNR mismatch must be investigated in the context of design parameters q and V_{lim}.

4-bit and 5-bit quantized decoders are used as examples. Two typical dynamic ranges $V_{\text{lim}} = 5$ and $V_{\text{lim}} = 8$, which were investigated in Chapter 4, are taken into consideration. As for the Max-Log-MAP decoding scheme, $V_{\text{lim}} = 1.3$ is adopted to

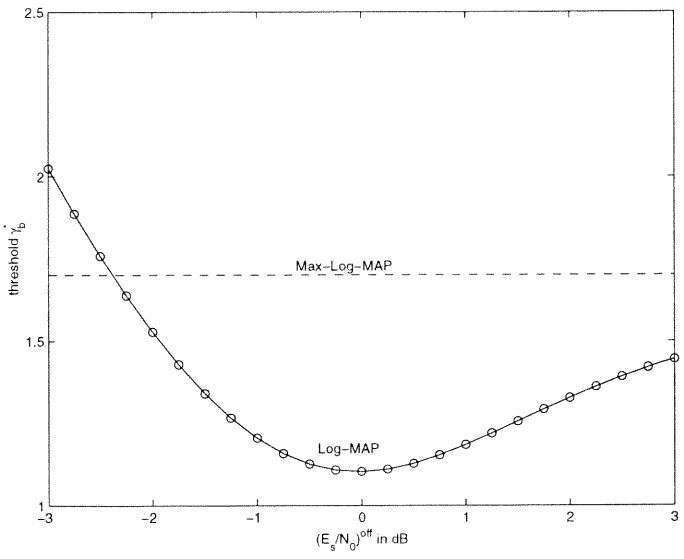


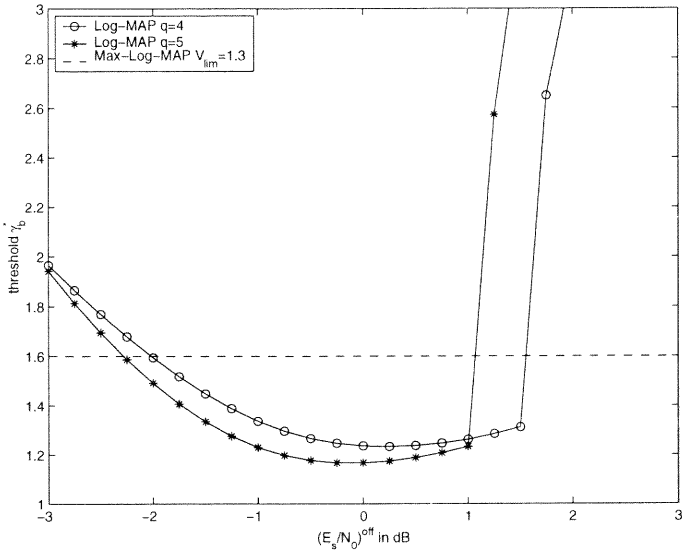
Figure 5.1 Threshold values versus $(E_s/N_0)^{\text{off}}$ (infinite precision)

provide highest capacity. Note that the capacity of Max-Log-MAP is the same for 4-bit and 5-bit schemes, if V_{lim} is set to 1.3.

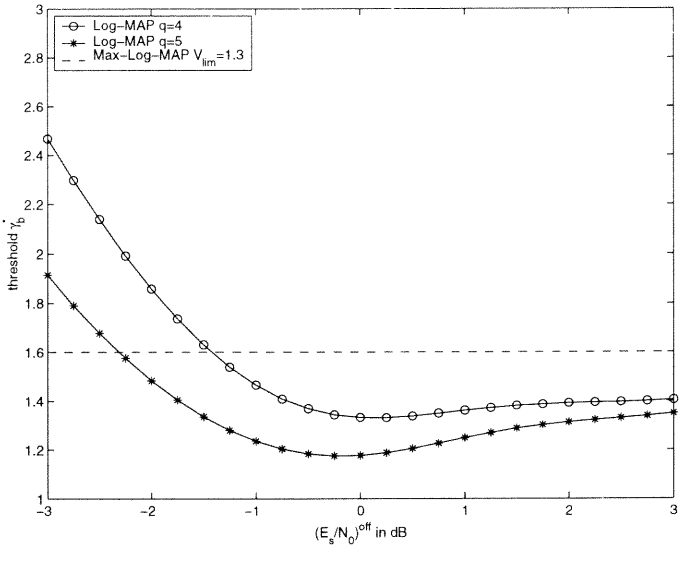
Given that $V_{\text{lim}} = 5$, an SNR mismatch of ± 1 dB renders little loss in capacity, as shown in Figure 5.2 (a). However, an overestimation exceeding 1.0-1.5 dB is disastrous. Underestimation of 2.0 dB or more would yield a capacity lower than the suboptimal Max-Log-MAP algorithm.

In contrast, under higher dynamic range $V_{\text{lim}} = 8$, overestimation brings about little loss in capacity, while underestimation exceeding 1.5 dB (for 4-bit scheme) or 2.25 dB (for 5-bit scheme) would yield a capacity below that of Max-Log-MAP algorithm, as shown in Figure 5.2 (b). The 5-bit quantized decoder is less sensitive to SNR mismatch than the 4-bit scheme. In fact, the 5-bit quantized decoder is as robust as the infinite precision decoder.

In conclusion, the sensitivity of the quantized decoder to SNR mismatch depends on resolution q and dynamic range V_{lim}. Under very low dynamic ranges, say $V_{\text{lim}} = 5$, overestimation of SNR might be disastrous. As demonstrated in Figure 5.2 (b), $V_{\text{lim}} = 8$ is a good dynamic range in general, for 4-bit and 5-bit



(a) dynamic range $V_{\text{lim}} = 5$



(b) dynamic range $V_{\text{lim}} = 8$

Figure 5.2 Threshold values versus $(E_s/N_0)^{\text{off}}$(4-bit and 5-bit quantized implementation)

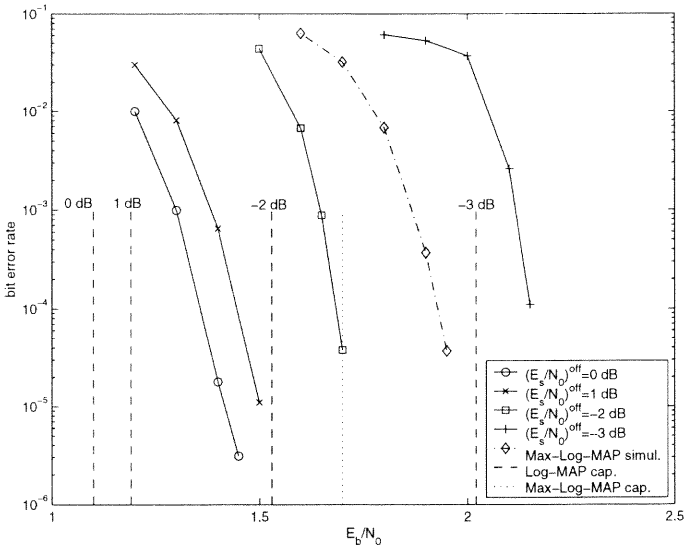


Figure 5.3 Simulation results in comparison with theoretical capacity

quantized Log-MAP decoders. Actually, there is no obvious difference in sensitivity to SNR mismatch between $(q, V_{\text{lim}}) = (5, 8)$ quantized decoder and infinite precision decoder.

5.2 Simulation Studies

The theoretical thresholds in Section 5.1.2 are obtained assuming infinite code length and cycle-free underlying graphical model. However, practical LDPC codes are of limited code length and contain cycles in the graphical model. It is necessary to carry out simulation studies to find out the effects of SNR mismatch in the context of relatively short code length. Actually, this is the first time that density evolution is used in the investigation of SNR mismatch. Therefore, another purpose for simulation studies is to confirm the effectiveness of the density evolution procedure. BER performances under fixed SNR offset and fixed E_b/N_0 are discussed below.

5.2.1 Fixed Channel SNR Offset

Figure 5.3 shows the simulation results of 10080/20160 regular (3,6) LDPC code, a relatively long code, with fixed SNR offset of 0 dB, 1 dB, -2 dB and -3 dB. The

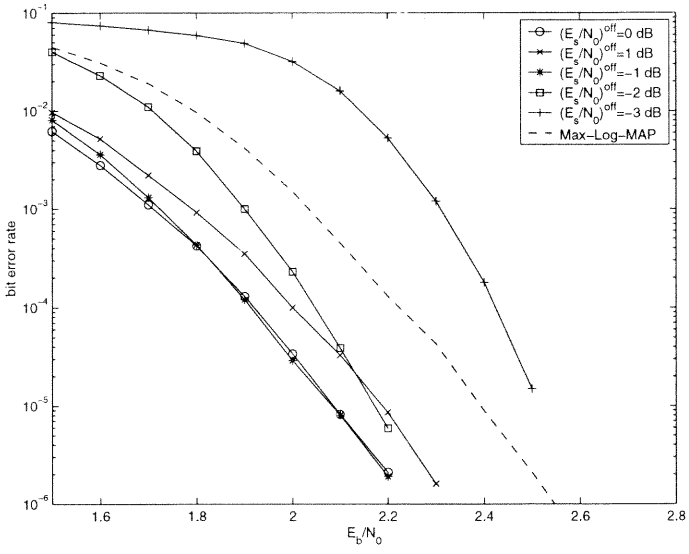


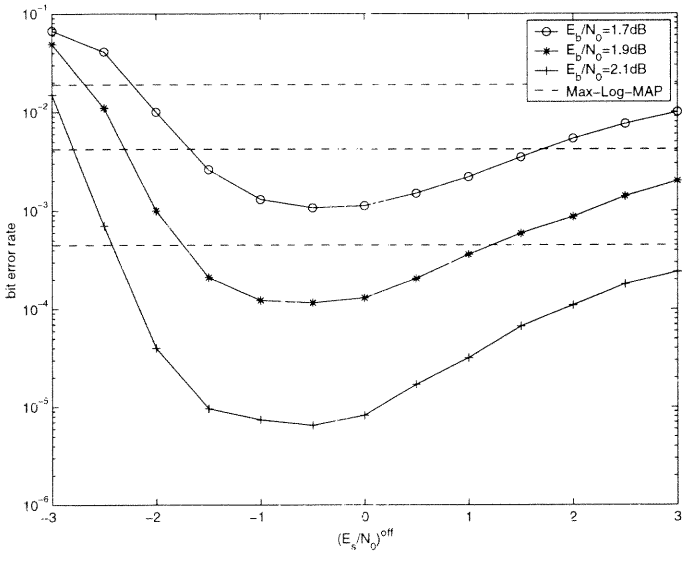
Figure 5.4 BER performance under fixed $(E_s/N_0)^{\text{off}}$

vertical dashed lines represent theoretical capacities obtained using density evolution. The thresholds $\gamma_b^*((E_s/N_0)^{\text{off}})$ well predict the E_b/N_0 penalty due to channel SNR offset.

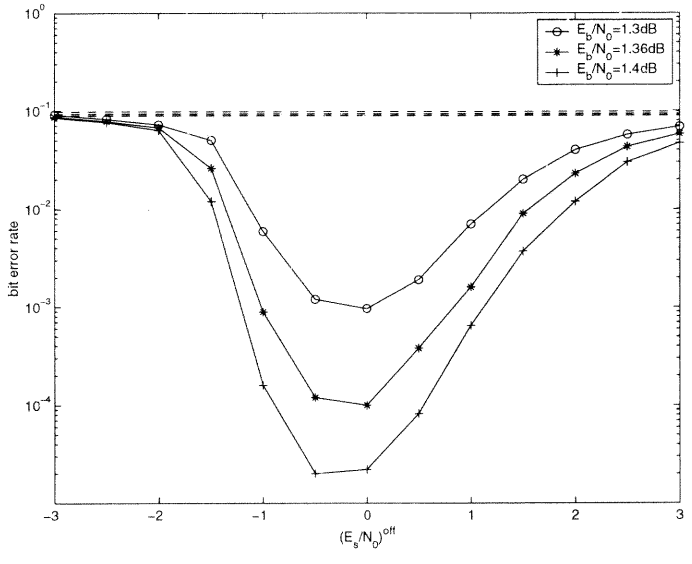
Figure 5.4 shows the simulation results of 1008/2016 regular (3,6) LDPC code, a shorter code, with fixed SNR offset of 0 dB, 1 dB, -1 dB, -2 dB and -3 dB. The performance of Max-Log-MAP decoding is drawn in dashed line for comparison. First, it is clear that E_b/N_0 penalty due to SNR offset is not as big as that of longer code. This implies that shorter LDPC code is less sensitive to SNR mismatch. This point is also supported by simulations on several fixed E_b/N_0, as shown below. Second, underestimation within 1dB might even improve BER performance in the region of higher E_b/N_0.

5.2.2 Fixed E_b/N_0

Figure 5.5 shows how mismatched SNR level affects BER performance when E_b/N_0 is fixed. The E_b/N_0 under consideration correspond approximately to BER of 10^{-3}, 10^{-4} and 10^{-5}. For 1008/2016 LDPC code, E_b/N_0 of 1.7 dB, 1.9 dB and 2.1 dB are considered. The performance of Max-Log-MAP decoding is shown in dashed lines



(a) 1008/2016 regular (3,6) LDPC code



(b) 10080/20160 regular (3,6) LDPC code

Figure 5.5 BER versus channel SNR offset for some fixed actual E_b/N_0

for comparison. For 10080/20160 LDPC code, E_b/N_0 of 1.3 dB, 1.36 dB and 1.4 dB are considered.

Note that in short LDPC code, minimum BER is not achieved under exact channel SNR estimation. This effect, however, occurs only for the short code length under consideration. For long codes, it is possible to prove, using density evolution, that the minimum will indeed be obtained at 0 dB offset. The simulation results confirm that longer LDPC code is more sensitive to SNR mismatch. The performance of Max-Log-MAP decoding is comparable to that of Log-MAP only in short LDPC codes. Generally, it is not difficult to design a channel estimator offering an SNR offset with $-3 \leq (E_b/N_0)^{\text{off}} \leq 3$ in dB [45]. Therefore, when considering the BER performance, Log-MAP decoding is a better choice.

5.3 Summary

In this chapter, the sensitivity of Log-MAP LDPC decoder to channel SNR mismatch is investigated, on AWGN channel. It is found that:

1. As a tool to compute the decoding capacity of LDPC codes under fixed SNR offset, the density evolution procedure works very well. The theoretical thresholds obtained well predict the pinch-off E_b/N_0 in presence of channel SNR offset.

2. Log-MAP LDPC decoder would be extremely sensitive to SNR mismatch, if the dynamic range is too low, say $V_{\text{lim}} = 5$. Considering the quantization effects and robustness to SNR offset, the dynamic range of $V_{\text{lim}} = 8$ is a good choice for 4-bit or 5-bit quantized decoders.

3. The amount of SNR offset that can be tolerated depends on the code length. Generally, shorter LDPC code is less sensitive to SNR mismatch. Longer LDPC

code requires more accurate channel estimation. This conclusion is the same as that for turbo code.

4. Although turbo codes and LDPC codes share similar elegant Log-MAP decoding algorithms, some difference is observed concerning the robustness to SNR mismatch. For turbo code of block length 600, the gap between Log-MAP and Max-Log-MAP decoding eventually disappears when BER drops below 10^{-5} [46]. Therefore, from a practical point of view, Max-Log-MAP decoding is strongly recommended to eliminate the requirement of SNR estimation. For LDPC codes, simulations were carried out on even shorter code such as 252/504. The gap between Log-MAP and Max-Log-MAP remains significant. Therefore, to achieve good BER performance, using Log-MAP decoder with a good SNR estimator is recommended.

CHAPTER 6

ITERATIVE DECODING OF PRODUCT CODE

In this chapter, two issues on iterative decoding of product code are investigated. One is, improving BER performance by mitigating cycle effects. The other is, parallel decoding structure. Traditional iterative decoding algorithm strictly follows Pearl's belief propagation procedure. Satisfying BER performance has been achieved. However, the performance could be further improved by scaling the extrinsic information, as shown in Section 6.2. In the original SISO iterative decoder for turbo code and product code, constituent decoders are activated in serial mode. Considering the fact that in product code the component codes in each dimension are equally positioned, parallel decoding should be a conceptually better choice. The decoder structure, BER performance and convergence speed of parallel and serial decoding are compared in Section 6.4.

6.1 Product Code

Product code was introduced in as early as 1954 by Elias [12]. It is based on simple and short linear block code. Its large minimum Hamming distance promises an excellent BER performance under maximum likelihood (ML) decoding. However, the huge codeword set makes ML decoding impractical. Elias suggested to decode product code by sequentially decoding the rows and columns. Unfortunately, the first iterative decoding algorithms gave rather poor results because they relied on hard-input/hard-output component decoders [49, 50].

Since then, product code had been largely forgotten until 1993 when a break through was made by applying soft-input/soft-output iterative decoding algorithm to it [13]. Later Hagenauer unified all existing iterative decoding algorithms for product code and turbo code into a well-established, harmonized theoretical framework [32].

Pyndiah proposed a suboptimal low-complexity soft-input/soft-output algorithm for the component decoder to push product code a step further to practical implementation [51].

To compare product codes to turbo codes would be as hard as to compare convolutional code to block code. They have their respective advantages, depending on applications. Generally, product code has a much lower error floor due to the large minimum Hamming distance. For higher rate applications, it outperforms turbo code. It allows for shorter code length and lower complexity decoder. In fact, commercialized products on product code have been put to market by AHA and Efficient Channel Coding Inc..

6.1.1 Code Structure

Product code, based on simple and short linear block codes, allows the construction of fairly long codes with low complexity as well as a simple iterative decoding algorithm. Here the concept of D-dimensional product code is depicted.

Consider D systematic linear block codes $\mathcal{C}^i$ with parameters (n_i, k_i, d_i) where n_i, k_i and d_i $(i = 1, 2, \cdots, D)$ stand for the codeword length, number of information bits and minimum Hamming distance respectively. The D-dimensional product code $\mathcal{P} = \mathcal{C}^1 \bigotimes \mathcal{C}^2 \bigotimes \cdots \mathcal{C}^D$ is constructed in the following steps: 1) place information bits in a hypercube of dimension D with the length in each dimension defined by $k_1, k_2, \cdots, k_D$; 2) encode the i^{th} dimension with the linear block code $\mathcal{C}^i$. For $i = 1$, all the information bits are encoded. For $i \geq 2$, all the information bits together with previously-obtained parity bits are encoded; 3) Repeat step 2 for $i = 1, 2, \cdots, D$ until the $n_1 \times n_2 \times \cdots n_D$ hypercube, which is the array of coded bits, is filled up. Viewed along the direction associated with any dimension i, the coded bits form a codeword of $\mathcal{C}^i$. Figure 6.1 discribes the code construction process for a two-dimensional product code.

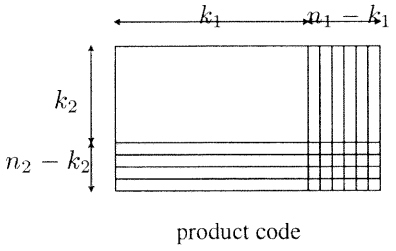


Figure 6.1 Encoding process

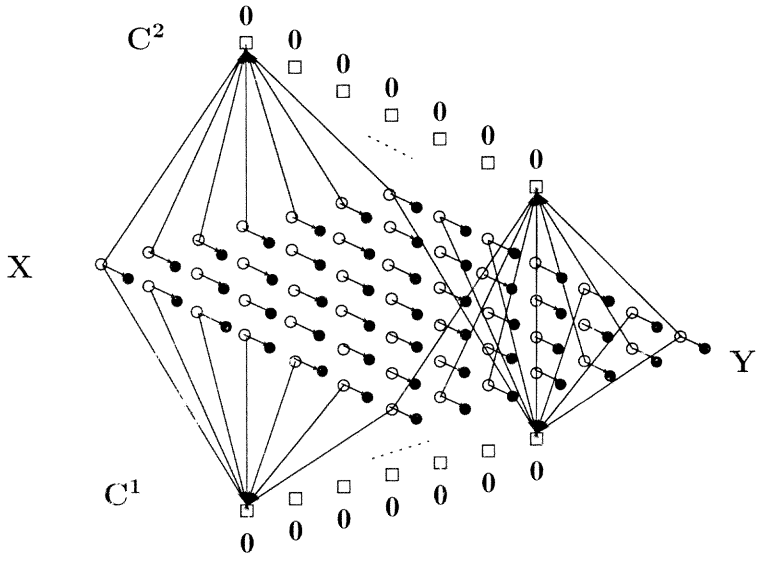


Figure 6.2 Graphical model for two-dimensional product code

The product code $\mathcal{P}$ obtained is a new linear block code with parameters (n, k, d) where $n = \prod_{i=1}^{D} n_i$, $k = \prod_{i=1}^{D} k_i$ and $d = \prod_{i=1}^{D} d_i$. The new minimal Hamming distance d increases rapidly as the number of dimension increases.

Product code with check-on-check is sometimes named serially concatenated code, for a component encoder takes in both parity bits and systematic bits from the previous encoding stage. However, the order of component encoders does not affect the final coding results. The D constituent encoders are equally positioned. From this point of view, product code resembles PCCC except that the component codes are linear block code and the interleaver is non-random.

In product codes, each row/column of coded bits is constrained by a short linear block code. The graphical model for Hamming (7,4)×(7,4) two dimensional product

code is obtained by applying the polytree model given in Figure 2.5 (b) to each row/column, as shown in Figure 6.2. Generally, for D-dimensional product code, the nodes associated with coded bits are denoted by a vector $\mathbf{X}$, and their corresponding evidence noisy vector as $\mathbf{Y}$. The check nodes for dimension i, where $1 \leq i \leq D$, are denoted by vector $\mathbf{C_i}$. Although each constitute subgraph is a polytree, the whole network is loopy. Since a uniform block interleaver is embedded, the graphical model is highly uniform.

6.1.2 Iterative Decoding Based on Belief Propagation

Iterative decoding proceeds by recursively updating the probability estimations for each coded bits. That's why it was easily connected to Pearl's belief propagation algorithm [7], a procedure to solve probability inference problem. Iterative decoder structure was depicted in the original turbo code paper [2] as serially concatenated (component) decoders. This decoder structure was later adopted in most turbo and turbo-like codes.

Hagenauer presented the method of iterative decoding in a unified framework [32]. He also tried to define what is meant by the various soft values. When being connected with the probability inference problem [7], the theoretical framework for iterative decoding is finally explicit and complete. In this thesis, the author basically follows Hagenauer's symbol set. However, with the new concepts clarified in the milestone paper [7], the author is able to re-explain the meaning of soft values.

The objective of iterative decoding is to estimate the probability distribution of each coded bits, conditioned on the received channel values $\mathbf{y}$ and code constrains $\mathbf{C^1}$ and $\mathbf{C^2}$, which is similar to the concept of belief $\mathrm{Bel}(x)$, as discussed in chapter 2. According to belief propagation, the final $\mathrm{Bel}(x)$ is computed by equation (2.23), as follows,

$$\mathrm{Bel}(x) = \alpha \pi_X(x) \lambda_X(x). \tag{6.1}$$

Expressed in log likelihood ratio symbol set, Bel(x) is associated with the final decision variable $L(\hat{x})$,

$$L(\hat{x}_n) = L_{ap}(x_n) + L_y(x_n) + L_e^{\mathbf{C^1}}(x_n) + L_e^{\mathbf{C^2}}(x_n).. \tag{6.2}$$

L_{ap} corresponds to π_X, the a priori probability. Since x_n is equally likely to be 0 or 1, the term L_{ap} is generally omitted. As depicted in equation (2.22), $\lambda_X(x)$ represents information from X's child. It's the probability of the evidence that the children nodes "know", conditioned on x. In the graphical model of two-dimensional product code, any coded bit x_n has 3 children, received value y_n, code constraint $\mathbf{C^1}$ and $\mathbf{C^2}$. Correspondingly, the messages they submit to x_n are denoted as $L_y(x_n)$, $L_e^{\mathbf{C^1}}(x_n)$, and $L_e^{\mathbf{C^2}}(x_n)$. Assuming an AWGN channel with noise power $N_0/2 = \sigma^2$, $L_y(x_n)$ is written as,

$$L_y(x_n) = L(y_n|x_n) = \ln \frac{p(y_n|x_n = +1)}{p(y_n|x_n = -1)} = \frac{2}{\sigma^2} y_n = L_{ch} y_n. \tag{6.3}$$

Like $L_y(x_n)$, extrinsic information $L_e^{\mathbf{C^1}}$ and $L_e^{\mathbf{C^2}}$ are also associated with probabilities conditioned on x, except that they are extracted with respect to code constraint $\mathbf{C^1}$ and $\mathbf{C^2}$. There have been numerous algorithms for calculating extrinsic information. They will be detailed in Section 6.1.3.

Node X compute its belief by combining the messages collected from its children nodes. To proceed with the belief propagation, node X would pass some messages back to its children. As indicated in equation (2.24), X would pass the new belief to its child $\mathbf{C^i}$ but excluding the information from $\mathbf{C^i}$. (Since Y is a leaf node with a single parent, $L_y(x)$,the message to X, is unchanged throughout the decoding process. Therefore the message from X to Y is omitted). Specifically, the information to $\mathbf{C^1}$ is,

$$L^{\mathbf{C^1}}(x_n) = L_y(x_n) + L_e^{\mathbf{C^2}}(x_n) \tag{6.4}$$

and the information to $\mathbf{C^2}$ is,

$$L^{\mathbf{C^2}}(x_n) = L_y(x_n) + L_e^{\mathbf{C^1}}(x_n). \tag{6.5}$$

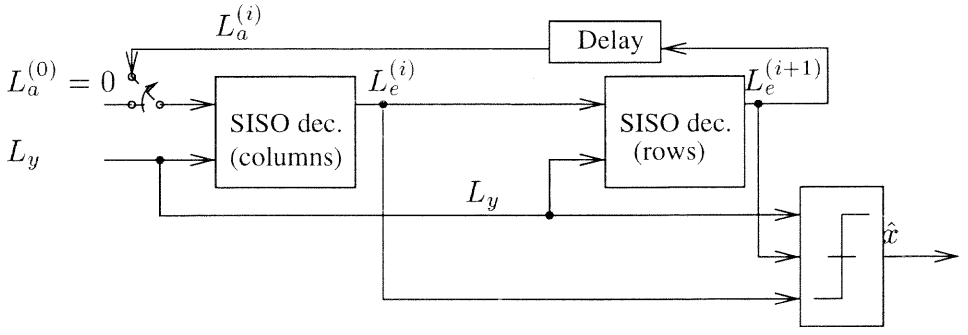


Figure 6.3 BP-based iterative decoder structure

This rule prevents the information originating from a node to flow back to it. Repeat the message passing process. Finally the information from each node would spread to the whole network. If there is no loop in the graph, exact a posteriori probability (APP) of X on the evidence could be obtained. For loopy network, the information would circulate such that the algorithm would not render an exact APP. However, if the loop girth is long enough and the loop pattern random enough, the final results are still good enough for making decisions. That is the reason why long random codes such as LDPC codes and turbo codes yield surprisingly good performance under BP-based iterative decoding. In product codes, the loops are short and highly uniform. Some modifications on decoding structure would lead to better performance. This will be discussed in the next section.

Figure 6.3 shows the decoder structure for two dimensional product code. The same scheme is also adopted by the original turbo code (parallel concatenated convolutional code) with two constituent encoders, except that here in product code, the block interleaver/deinterleaver is embedded. The operations in component decoder exactly correspond to the calculations in code constraint nodes $\mathbf{C^i}$. The decision unit generates the belief $L(\hat{x}_n)$ to make hard decisions. This serially concatenated decoding structure represents one of many possible node activation schedules, $\mathbf{X}, \mathbf{C^1}, \mathbf{X}, \mathbf{C^2}, \mathbf{X}, \mathbf{C^1}, \cdots$. An alternative node activation schedule leads to parallel iterative decoding, which would be detailed in Section 6.4.

To highlight results from different decoding stages, the extrinsic information from decoding stage i is denoted as $L_e^{(i)}, i = 0, 1, 2, \cdots$, omitting the specific column or row code constraint $\mathbf{C}$, as shown in the figure. The decoder structure fits perfectly into the belief propagation framework, though the former had been developed without the knowledge of probability inference. The term BP-based iterative decoding is used.

6.1.3 Log-MAP Extrinsic Information

Within the decoder structure specified above, there have been numerous algorithms to compute the extrinsic information. They could be classified into two types, optimal and suboptimal. The former generates soft output resulting in minimized bit error rate. The latter usually trades BER performance off for lower complexity. Suboptimal extrinsic information extraction for block codes is discussed in several papers[52, 51, 53]. This research focuses on the optimum performance that could be achieved. Therefore, optimum decoding models are preferable. They are associated with MAP decoding, as detailed below.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, $x_n \in GF(2) = \{+1, -1\}$, denotes a codeword of a linear block code $\mathcal{C}$ and $\mathbf{x}' \in GF(2)^N$ a codeword of the corresponding dual code $\mathcal{C}^{\perp}$. A symbol-by-symbol maximum a posteriori (MAP) decoder is written as,

$$\hat{x}_n^{\text{MAP}} = \arg\max_{x_n} p(x_n|\mathbf{y}) \tag{6.6}$$

which minimizes the bit error probability of sequence $\mathbf{x}$.

Therefore, the key point in MAP decoding is to evaluate the a posteriori probability $p(x_n|\mathbf{y})$. For binary code, the soft output of MAP decoder is usually in the form of log a posteriori probability ratio,

$$L(\hat{x}_n) \triangleq L(x_n|\mathbf{y}) = \ln \frac{p(x_n = +1|\mathbf{y})}{p(x_n = -1|\mathbf{y})} \tag{6.7}$$

The MAP decision $\hat{x}_n$ is obtained by $\hat{x}_n = \text{sign}[L(\hat{x}_n)]$.

There exists several algorithms to calculate $L(\hat{x}_n)$. The well known BCJR algorithm, originally proposed for convolutional codes, can be applied to any code where a trellis can be drawn. Linear block code has an irregular trellis as opposed to the regular trellis. It is possible to compute the extrinsic information using BCJR algorithm. However, straightforward implementations using the whole codeword set $\mathcal{C}$ or dual codeword set $\mathcal{C}^{\perp}$ are often more preferable, for the codeword set of short block code is not that big.

Let's state the problem of MAP decoding as, finding out the exact a posteriori probability distribution $p(x_n|\mathbf{y})$ for each bit of a codeword, given the probability $p(x_n; y_n) \stackrel{\triangle}{=} p(y_n|x_n)P_a(x_n)$, $1 \leq n \leq N$. Roughly speaking, $P_a(x_n)$ represents the current probability information on x_n extracted from sources other than the channel value y_n and code constraint $\mathcal{C}$. Following Hagenauer's notations, the corresponding log-likelihood ratio is denoted as,

$$\begin{aligned} L(x_n; y_n) &= \ln \frac{p(x_n = +1; y_n)}{p(x_n = -1; y_n)} \\ &= \ln \frac{p(y_n|x_n = +1)}{p(y_n|x_n = -1)} + \ln \frac{P_a(x_n = +1)}{P_a(x_n = -1)} \\ &= L_y(x_n) + L_a(x_n). \end{aligned} \tag{6.8}$$

Using the definition of equation (6.7) and the codeword set $\mathcal{C}$, the soft output is written as,

$$\begin{aligned} L(\hat{x}_n) &= \ln \frac{P(x_n = +1|\mathbf{y})}{P(x_n = -1|\mathbf{y})} \\ &= \ln \frac{\sum_{\mathbf{x}\in\mathcal{C}, x_n=+1} P(\mathbf{x}|\mathbf{y})}{\sum_{\mathbf{x}\in\mathcal{C}, x_n=-1} P(\mathbf{x}|\mathbf{y})} \\ &= \ln \frac{\sum_{\mathbf{x}\in\mathcal{C}, x_n=+1} (\prod_{j=1}^{N} p(y_j|x_j) P_a(\mathbf{x}))}{\sum_{\mathbf{x}\in\mathcal{C}, x_n=-1} (\prod_{j=1}^{N} p(y_j|x_j) P_a(\mathbf{x}))} \\ &= \ln \frac{p(x_n = +1; y_n) \sum_{\mathbf{x}\in\mathcal{C}, x_n=+1} \prod_{j=1, j\neq n}^{N} p(x_j; y_j)}{p(x_n = -1; y_n) \sum_{\mathbf{x}\in\mathcal{C}, x_n=-1} \prod_{j=1, j\neq n}^{N} p(x_j; y_j)} \\ &= L(x_n; y_n) + \ln \frac{\sum_{\mathbf{x}\in\mathcal{C}, x_n=+1} \prod_{j=1, j\neq n}^{N} \exp(L(x_j; y_j) x_j/2)}{\sum_{\mathbf{x}\in\mathcal{C}, x_n=-1} \prod_{j=1, j\neq n}^{N} \exp(L(x_j; y_j) x_j/2)} \end{aligned}$$

$$= L_y(x_n) + L_a(x_n) + L_e(\hat{x}_n) \tag{6.9}$$

Note that before the decoding operation, all the available information about x_n is $L(x_n; y_n)$. With the knowledge of code constraint $\mathcal{C}$, a new term $L_e(\hat{x}_n)$ is generated. It is an estimated reliability value independent of $L(x_n; y_n)$. Rather, it utilizes the code constraint and input values other than $L(x_n; y_n)$. In the case of multiple constituent decoders, extrinsic information L_e will be used as a priori information for other decoders. Further examination of the term L_e would reveal that it is exactly equivalent to message passed to parent, that is, λ_{X,U_k} of equation (2.25).

Hartmann and Rudolph [54] found another way to calculate the probability $P(x_n = \pm 1|\mathbf{y})$ using the codewords of the dual code $\mathcal{C}^{\perp}$. In coding systems where the dual code has fewer codewords than the original code $\mathcal{C}$, this implementation is preferable. Let $\mathbf{x}'$ denotes the codeword of dual code $\mathcal{C}^{\perp}$, the soft output is,

$$\begin{aligned} L(\hat{x}_n) &= L_y(x_n) + L_a(x_n) \\ &\quad + \ln \frac{\sum_{\mathbf{x}' \in \mathcal{C}^{\perp}} \prod_{j=1, j\neq n}^{N} (\tanh(L(x_j; y_j)/2))^{(1-x'_j)/2}}{\sum_{\mathbf{x}' \in \mathcal{C}^{\perp}} (-x'_j) \prod_{j=1, j\neq n}^{N} (\tanh(L(x_j; y_j)/2)^{(1-x'_j)/2}} \\ &= L_y(x_n) + L_a(x_n) + L_e(\hat{x}_n) \end{aligned} \tag{6.10}$$

Equation (6.10) is adopted in the simulations. In either of the two algorithm, $L_e(\hat{x}_n)$ depends on the input vector $\{L(x_j; y_j)\}_{j=1, j\neq n}^{N}$, denoted as,

$$L_e(\hat{x}_n) = f_e(L(x_1; y_1), \cdots, L(x_{n-1}; y_{n-1}), L(x_{n+1}; y_{n+1}), \cdots, L(x_N; y_N)). \tag{6.11}$$

$L_e(\hat{x}_n)$ does not depend on input $L(x_n; y_n)$.

6.2 Scaled Factor Decoding

When extracting the extrinsic information, as depicted in equation (6.9), $P_a(\mathbf{x})$ is assumed to be separable,

$$P_a(\mathbf{x}) = \prod_{j=1}^{N} P_a(x_j). \tag{6.12}$$

That is, previous probability estimations on x_j and x_k are independent for any $j \neq k$. This condition equals the cycle-free assumption in Pearl's belief propagation. For two dimensional product code, the a priori probability would be independent to each other only at decoding stage 1 and 2.

Ever since turbo code was proposed, it has been realized that the dependency of extrinsic information decreases the BER improvement through iterations [32]. However, to the author's knowledge, little has been done on its impact on decoding results. So far, there has been no measure on the degree of dependency among extrinsic information. In this study, linear correlation coefficients are used to reflect the dependency.

Recall that in previous chapters, the decoder for LDPC code was viewed as a signal processing system with Gaussian distributed inputs. Statistical parameters of extrinsic information, such as SNR, mutual information, were used to describe the convergence behavior of iterative decoder. The graphical model was assumed to be cycle-free such that the extrinsic information for each bit were independent. The results of parameter evolution or density evolution well predict the performance of long random codes. In contrast, the graphical model of product code contains many uniform short cycles. The mutual dependency between a priori information $L_a^{(i)}(x_l)$ and $L_a^{(i)}(x_n)$ must be taken into considerations for any $n \neq l$. Linear correlation coefficient is used to measure dependency.

Studies show that a priori information would be heavily correlated merely after decoding stage 3. Adding a scaling factor at the initial several stages would mitigate cycles effects and improve the performance. This modified BP-algorithm is termed scaled factor decoding(SFD).

6.2.1 Statistical Behavior of a Prior Information

6.2.1.1 Dependency Analysis. For the convenience of description, the notations are changed slightly in this section. Two dimensional coordinate is adopted to highlight the decoding operations on rows and columns. At decoding stage i, the input associated with coordinate (l, m) is written as,

$$L^{(i)}(x_{l,m}; y_{l,m}) = L_{ch} y_{l,m} + L_a^{(i)}(x_{l,m}). \tag{6.13}$$

The intrinsic information $y'_{l,m} = L_{ch} y_{l,m}$ is a Gaussian random variable unchanged throughout decoding stages. Generally, $L_a^{(i)}(x_{l,m})$, is set to the extrinsic value extracted from the previous decoder,

$$L_a^{(i)}(x_{l,m}) = L_e^{(i-1)}(x_{l,m}). \tag{6.14}$$

Denote the whole set of input channel values as $\mathbf{Y} = \{y_{l,m}\}$, where $1 \leq l, m \leq N$. Divide $\mathbf{Y}$ into row vectors, $\mathbf{Y}_l^{-} = \{y_{l,m}\}_{m=1}^{N}$ with $l = 1, 2, \cdots, N$, and column vectors $\mathbf{Y}_m^{|} = \{y_{l,m}\}_{l=1}^{N}$ with $m = 1, 2, \cdots, N$. View the decoder as a non-linear signal processing system. The a priori information $L_a^{(i)}(x_{l,m})$ relies on the input $\mathbf{Y}$.

At stage 1, $L_a^{(1)}(x_{l,m}) = 0$. Without losing any generality, suppose that the first decoding stage deals with columns. Then at stage 2, the a priori information would depend on column vector,

$$L_a^{(2)}(x_{l,m}) = f_l^{(2)}(\mathbf{Y}_m^{|} \backslash y_{l,m}). \tag{6.15}$$

The a priori information $L_a^{(2)}$ associated with coded bits of the same column, let's say $L_a(x_{l,m})$ and $L_a(x_{k,m})$, would be dependent to each other, for they rely on overlapped information source, $\{\mathbf{Y}_m^{|} \backslash y_{l,m}\}$ and $\{\mathbf{Y}_m^{|} \backslash y_{k,m}\}$. However, the next decoding stage processes information row by row such that only the dependency within the same row is concerned. $L_a^{(2)}(x_{l,m})$ and $L_a^{(2)}(x_{l,n})$ depend on non-overlapping set of channel inputs, say $\{\mathbf{Y}_m^{|} \backslash y_{l,m}\}$ and $\{\mathbf{Y}_n^{|} \backslash y_{l,n}\}$ respectively. Therefore, at decoding stage 2, the independency requirement is still fulfilled.

At decoding stage 3, a priori information $L_a^{(3)}(x_{l,m})$ is connected with channel value set $\mathbf{Y}$ excluding the row vector $\mathbf{Y}_l^-$,

$$L_a^{(3)}(x_{l,m}) = f_m^{(3)}(\mathbf{Y} \backslash \mathbf{Y}_l^-). \tag{6.16}$$

Now the a priori information in the same column, $L_a^{(3)}(x_{l,m})$ and $L_a^{(3)}(x_{k,m})$ for any $l \neq k$, are mutually dependent, since they share some common information source.

$L_a^{(i)}(x_{l,m})$ for any $i \geq 4$ would rely on the whole input set $\mathbf{Y}$, showing that information circulate within the loopy graphical model. The dependency among a priori information becomes more and more severe through decoding stages.

6.2.1.2 Statistical parameters. It's hard to measure the dependency among a priori information, for the decoder is a non-linear signal processing system. One simple approach is to view $L_a^{(i)}(x_{l,m})$ as a random variable and observe the evolution of its statistical parameters through decoding stages.

Assuming all-zero codeword, $L_a^{(i)}(x_{l,m})$ could be denoted as $L_a^{(i)}$, for the a priori information for any coordinate (l, m) has very similar statistical parameters. The statistical parameters are estimated by Monte Carlo simulations, where $(16, 11)^2$ extended Hamming code on AWGN channel is assumed.

The decoder of stage i takes in two random signal for each coordinate, $L_{ch}y$ and $L_a^{(i)}$. The former is a random variable unchanged throughout the decoding process. $L_a^{(i)}$ is Gaussian-like whose mean and variance depend on i, as illustrated in Figure 6.4. At stage 2, the extrinsic information $L_a^{(2)}$ is almost as "powerful" as the channel value.

To measure the dependency among the a priori information, the linear correlation coefficient between a certain coordinate and all other coordinates are estimated. Figure 6.5 shows the correlation coefficients of a priori information from stage 2 to stage 4 at E_b/N_0 of 2 dB. At decoding stage 2, a priori information in the same column are strongly correlated, since they are derived from the same column

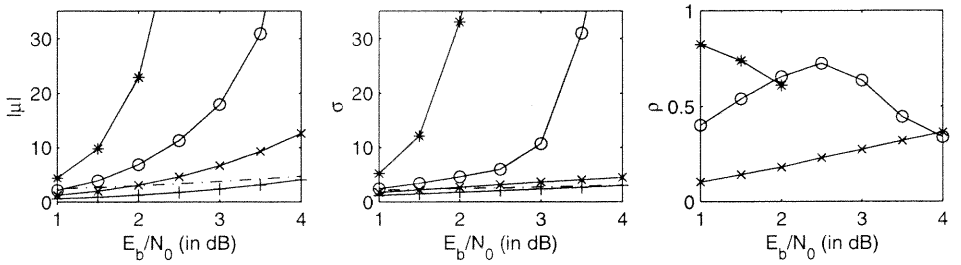


Figure 6.4 Statistical parameters for a priori information in BP-based algorithm. From left to right: mean $|\mu|$, standard deviation σ and correlation coefficients. Dashdot line: random variable $L_{ch}y$; plus, x-mark, circle and star: decoding stage 2 to 5

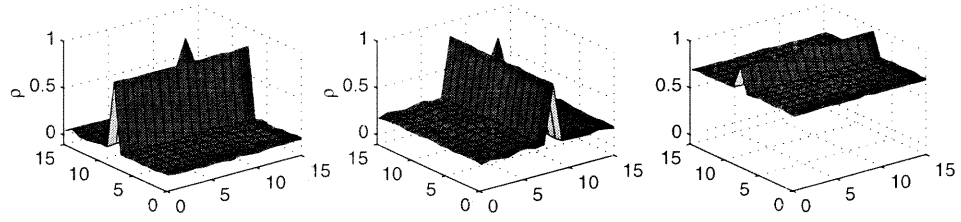


Figure 6.5 Correlation plane for a priori information at E_b/N_0 of 2 dB. From left to right: decoding stage 2 to 4

decoder. As predicted in dependency analysis, the information in the same row are uncorrelated. At decoding stage 3, the correlation surface is lifted to about 0.2. At decoding stage 4, it is up to 0.7. The surface is almost smooth, showing that they are equally correlated. The ridge is out of concern since the next decoding step always proceeds along the other direction.

The concerned correlation coefficient for decoding stage 3, 4 and 5 is given in Figure 6.4. It varies on different E_b/N_0 and decoding stages. At high E_b/N_0 it is up to 0.36 merely at stage 3, which means dependency is not negligible. Through iterations, the linear correlation coefficient goes up first and then drops. Definitely, the a priori values become more and more dependent. However, after saturation, some a priori value reaches infinity, the dependency tends to be nonlinear so that the correlation coefficient decreases.

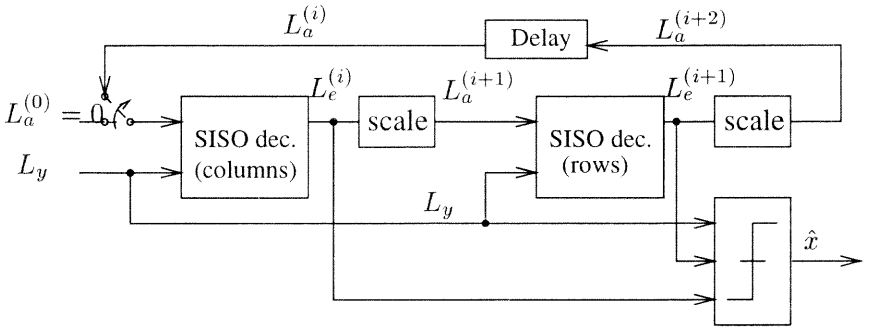


Figure 6.6 Scaled factor decoder structure

6.2.2 Scaled Factor Decoding

Following Pearl's belief propagation procedure, the extrinsic information extracted from the previous decoder is directly used as a priori probability for the next decoder, as shown in equation (6.14). Such algorithms are adopted by most iterative decoders [32].

Definitely, ignoring the dependency or equivalently the loops would result in a degraded BER performance. For product code, the dependency among a priori values should not be neglected because it is rather severe at as early as decoding stage 3. The initial several decoding stages are crucial since they determine the approximate convergence direction. It is proposed to pass scaled extrinsic information to the next decoding stage,

$$L_a^{(i)}(x_{l,m}) = \beta^{(i)} L_e^{(i-1)}(x_{l,m}) \tag{6.17}$$

where $\beta^{(i)}$ is the scaling coefficient for decoding stage i and $0 \leq \beta^{(i)} \leq 1$. It is termed scaled factor decoding (SFD). The decoder structure is displayed in Figure 6.6.

6.2.3 Simulation Results

The evolution of scaling coefficient is set as follows, unless otherwise specified,

$$\beta^{(i)} = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 1, 1, \cdots] \tag{6.18}$$

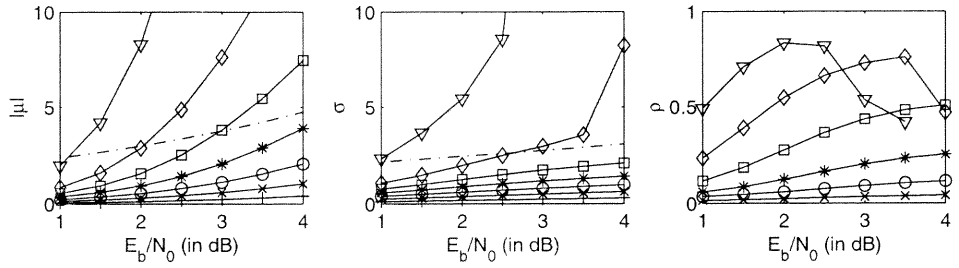


Figure 6.7 Statistical parameters of a priori information in SFD. From left to right: mean μ, standard deviation σ and correlation coefficients. Dashdot line: random variable y'; plus, x-mark, circle, star,square, diamond and triangle: decoding stage 2 to 8

6.2.3.1 Statistical parameters. Comparing with that of BP-based algorithm,the mean value and standard deviation increases much more slowly, as shown in Figure 6.7. The correlation coefficient goes up gradually and reach saturation at decoding stage 7. In contrast, for BP-based algorithm, it reaches saturation at stage 3. The dependency among input L-values at the first several decoding stages is very low.

6.2.3.2 BER performance. The simulated BER performance is given in Figure 6.8. The solid line plus diamonds represents scaled factor decoding. The solid line with x-marks, BP-based algorithm. The BER improvement increases as E_b/N_0 goes up. To achieve a BER of 10^{-5}, scaled factor decoding requires E_b/N_0 of 0.35 dB less.

It is found that as long as scaling is introduced before saturation, improvement is always guaranteed. However, a set of "good" coefficients for a certain range of E_b/N_0 might be "not-so-good" for other E_b/N_0. It is highly probable that a "good-for-all-SNR" scaling factor sequence does not exist.

Basically, the conclusion is two-folded: uniform short cycles degrade the BER performance of BP-based algorithm. SFD is quite effective in mitigating the impact of short cycles. Although the simulations are based on two dimensional Hamming

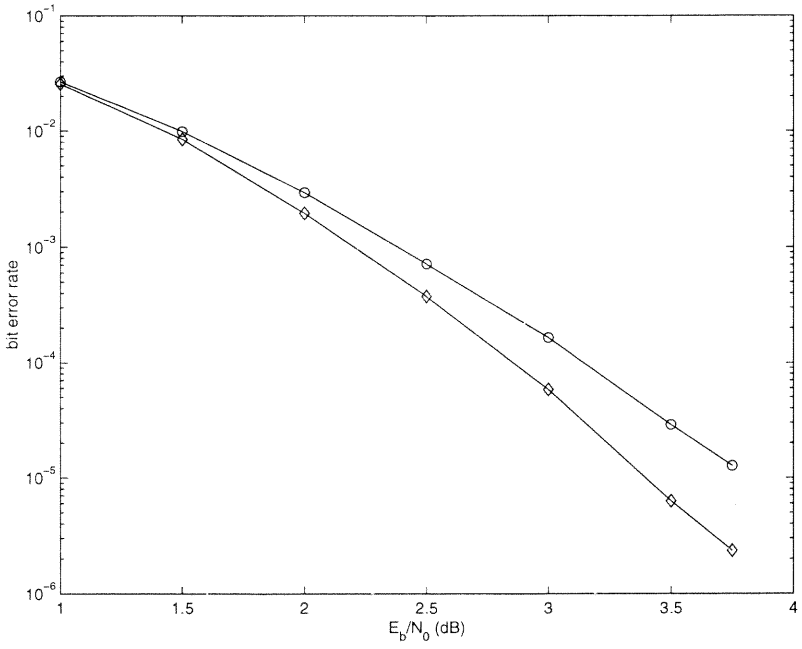


Figure 6.8 BER performance. Circle: BP-based algorithm; diamond: scaled factor decoding;

product code, SFD could be extended to product codes with higher dimension and any other component code. Section 6.4 gives results on three dimensional product code and again BER performance improvement is observed.

Improving performance by scaling the extrinsic information is not unique to Log-MAP decoding or any specific component code. Other researchers have applied it to product codes with BCH code [51] or single parity code [55] as component code. The decoding algorithm for component decoders could be suboptimal. This work is unique in that it is carried out in the theoretical framework of belief propagation. It is revealed that the improvement is due to the uniform short cycles in the underlying graphical model. This explains why scaling brings about no improvement in LDPC code and turbo code.

6.3 Discussion on SFD: How Good It Is

At this point, one interesting question is, how far away it is from the optimum results, for example, in maximum likelihood (ML) sense?

6.3.1 Simulated Lower Bound

One easy way to answer this question is to compare the decoding results to some upper bounds. However, current upper bounds are so loose that they give block error rate higher than the BP-based iterative decoding results. This means that the existing upper bound could not be used to demonstrate how close the iterative decoding algorithms are to optimum results.

Lucas proposed a simulated soft decision maximum likelihood (SDML) [52]. The main idea is to estimate the SDML performance via Monte Carlo simulations. Suppose all-zero codeword is transmitted. If the received sequence $\mathbf{y}$ is more close to a non-zero codeword in Euclidean distance, an error occurs. Repeat the experiment for N times and count the number of errors and denote is as n_e. If N is big enough, n_e/N approaches the block error rate under ML decoding. However, the problem is, the codeword set is so big that exhaustive search for closest codeword is impractical. Lucas only considers the received sequence that is *not* correctly decoded by the decoderIn [52]. The search scope only includes two codewords, all-zero and the estimated codeword. The error count derived, denoted as n'_e, is less than n_e, and a simulated SDML lower bound is obtained as n'_e/N.

The simulated SDML lower bound relies on the specific decoder. Generally, it is still not tight enough to evaluate decoding algorithm. A tighter simulated lower bound is proposed. It is based on exhaustive search for ML solutions. The searching scope is restricted to low-weight codewords. This simple algorithm renders a rather tight bound at high E_b/N_0, since a real ML decoder's output falls within low-weight codewords scope with very high probability. The procedure is as follows: 1) search

the code set $\mathcal{C}$ for low weight codeword. If the Hamming weight is less than a certain threshold, include this codeword in the subset $\mathcal{C}_{\mathcal{L}}$, $\mathcal{C}_{\mathcal{L}} \subset \mathcal{C}$. 2) Assuming all-zero codeword transmitted and sequence $\mathbf{y}$ received, search in $\mathcal{C}_{\mathcal{L}}$ for the codeword closest to $\mathbf{y}$,

$$\hat{\mathbf{x}}^{\mathrm{ML}} = \arg \max_{\mathbf{x} \in \mathcal{C}_{\mathcal{L}}} p(\mathbf{y}|\mathbf{x}) \tag{6.19}$$

If $\hat{\mathbf{x}} \neq \mathbf{0}$, count it as an error event. 3) Repeat step 2 for N times and get the number of error event n'_e. Use n'_e/N as a simulated lower bound for block error probability.

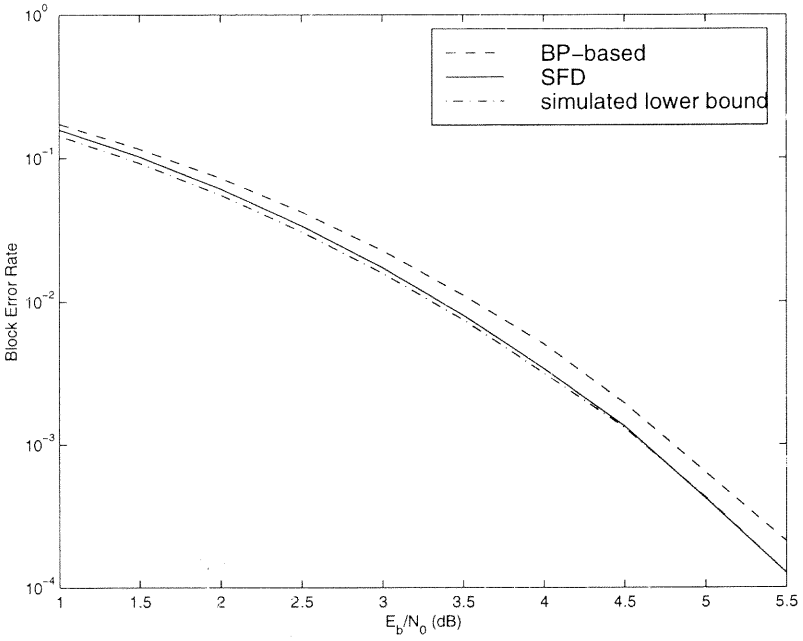
6.3.2 Performance Comparisons

The decoding of two-dimensional Hamming (7,4) check-on-check product code is simulated. The non-zero codeword weight ranges from 9 to 49. The decoding performance of SFD and BP-based algorithm are compared in Figure 6.9 (a). The lower bound is derived through searching the low weight code set $\mathcal{C}_{\mathcal{L}}$, which includes all the codewords whose Hamming weight is below 17.

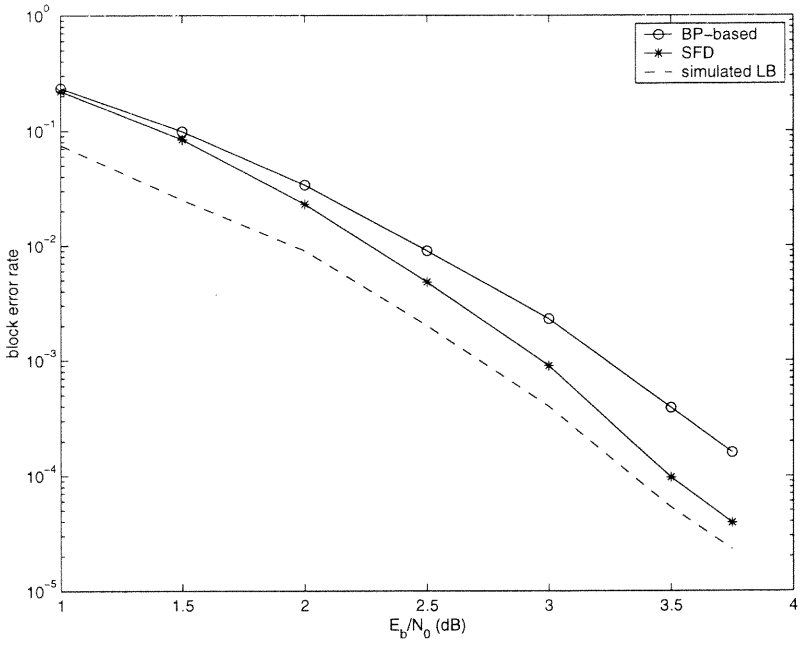
At higher E_b/N_0, the gap between SFD and simulated lower bound is extremely small. This again demonstrates the powerfulness of SFD. It is addressed that the simulated lower bound is tighter at high E_b/N_0 because the received sequence is less likely to fall near a high-weight codeword.

For two dimensional (16,11) extended Hamming code, which was used for simulations in the previous section, the codeword set is much bigger. The simulated ML decoding could only use the subset with minimum Hamming weight, which is 16. This lower bound derived is much looser. However, the results in Figure 6.9 (b) still confirms that SFD offers near-optimum performance.

In iterative decoding process, the MAP component decoders work in turns to minimize the bit error rate. If the underlying graph is cycle-free, MAP results could be achieved. SFD was proposed to mitigate the impact of short cycles. It is quite impressing that a result approaching the ML performance is obtained, which



(a) (7,4) Hamming code



(b) (16,11) extended Hamming code

Figure 6.9 Block error rates comparison

minimizes the block error rate. The result strongly implies that as a modified version of BP-based algorithm, SFD offers near optimum performance, especially under high E_b/N_0.

6.4 Parallel Iterative Decoding

Like turbo code, the first SISO iterative decoder for product code activates the constituent decoders in serial mode. The industry implementations on software or hardware also follow this structure. Later it is found that traditional iterative decoding is an instance of Pearl's belief propagation on graphical model. There exists more than one node activation mode in belief propagation. Correspondingly, alternative decoder structure should also work well. So far, studies on generalized turbo decoding have been focused on parallel concatenated convolutional code (PCCC). In [30], three alternative node activation schedules other than traditional turbo decoding were proposed. Simulations based on PCCC with two constituent encoders show that they lead to approximately the same residual BER as traditional turbo decoding algorithm. Parallel iterative decoding was proposed in [29, 56], where constituent decoders are activated simultaneously. Some BER performance improvement was observed on PCCC with three constituent encoders.

In this section, parallel iterative decoding is extended to product code. Given the parallel encoding structure in PCCC and product code, parallel decoding is conceptually better in that the constituent decoders for each dimension make equal contributions to the final decoding results. In addition, parallel processing generally renders lower latency, at the price of more processors or hardware. The bit error rate performance of serial and parallel decoding on three dimensional product code with (16,11) extended Hamming code is compared. Decoding delays are compared in terms of average decoding stages.

6.4.1 Decoder Structure

It is interesting to notice an important principle in Pearl's belief propagation. The information flowing out of a node will not be passed back to it by its direct parents or children, as shown in equations (2.25) and (2.24). This principle is well reflected in traditional iterative decoding. The extrinsic information from a component decoder always exclude the input portion obtained from the parent, as depicted in equation (6.9). The a priori information to a component decoder always excludes the information extracted from this decoder in previous iteration. If this principle is strictly followed, Pearl's belief propagation on cycle-free graphical model renders exact a posteriori probability for each coded bit. In iterative decoding, the graphical model usually contains cycles. However, satisfying BER performance would be obtained, as long as the principle is followed.

Take D-dimensional product code as an example. Traditional iterative decoding represents the node activation schedule of,

$$\mathbf{X} \rightarrow \mathbf{C^1} \rightarrow \mathbf{X} \rightarrow \mathbf{C^2} \cdots \rightarrow \mathbf{X} \rightarrow \mathbf{C^D} \rightarrow \mathbf{X} \rightarrow \mathbf{C^1} \rightarrow \mathbf{X} \rightarrow \mathbf{C^2} \cdots \tag{6.20}$$

Activate all the code constraint nodes simultaneously as,

$$\mathbf{X} \rightarrow \{\mathbf{C^1}, \mathbf{C^2} \cdots \mathbf{C^D}\} \rightarrow \mathbf{X} \rightarrow \{\mathbf{C^1}, \mathbf{C^2} \cdots \mathbf{C^D}\} \rightarrow \cdots \tag{6.21}$$

a fully parallel decoder is obtained.

Let's describe serial and parallel decoding by formulas. In serial decoding, each decoding stage contains only one component decoder corresponding to certain component encoder. At decoding stage i, the decoder takes in a channel value $L_y(x_n)$ and a priori log probability ratio of x_n, denoted as $L_a^{(i)}(x_n)$,

$$L(x_n; y_n) = L_y(x_n) + L_a(x_n). \tag{6.22}$$

Generally, $L_a^{(i)}(x_n)$ is associated with extrinsic information derived in the previous $D-1$ stages:

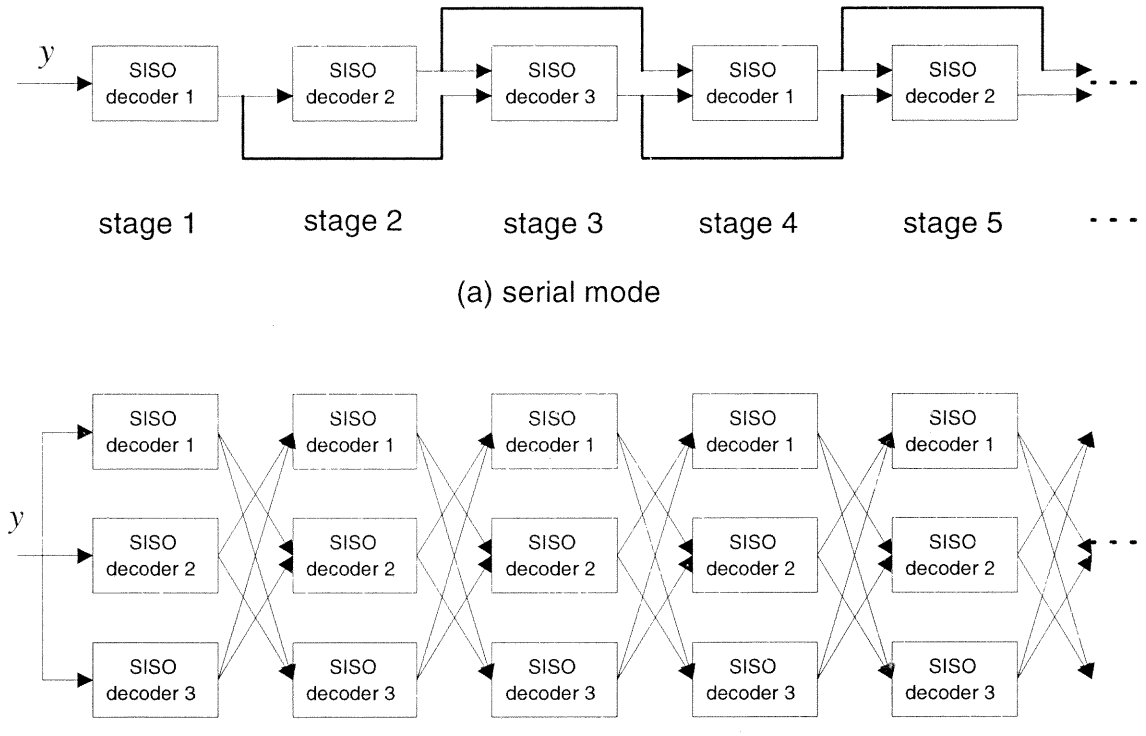


Figure 6.10 Parallel and serial decoder structure

$$L_a^{(i)}(x_n) = \sum_{k=1}^{D-1} L_e^{(i-k)}(x_n) \tag{6.23}$$

where $L_e^{(i)}$ is initialized to 0 for $i \leq 0$. An example of three dimensional product code decoder in serial mode is shown in Figure 6.10 (a).

In a fully parallel decoding structure, D component decoders work simultaneously at every decoding stage. At decoding stage i, decoder j $(1 \leq j \leq D)$ takes in the a priori log probability ratio $L_{a_j}^{(i)}$ and outputs extrinsic information $L_{e_j}^{(i)}$. $L_{a_j}^{(i)}$ is associated with previous extrinsic information by,

$$L_{a_j}^{(i)}(x_n) = \sum_{k=1, k\neq j}^{D} L_{e_k}^{(i-1)}(x_n) \tag{6.24}$$

Figure 6.10 (b) shows the parallel decoder structure.

Scaled factor decoding could also be extended to multi-dimension product code. The a priori information is set to

$$L_a^{(i)}(x_n) = \beta^{(i)} \sum_{k=1}^{D-1} L_e^{(i-k)}(x_n) \tag{6.25}$$

for serial decoding, or

$$L_{a_j}^{(i)}(x_n) = \beta^{(i)} \sum_{k=1, k\neq j}^{D} L_{e_k}^{(i-1)}(x_n) \tag{6.26}$$

for parallel decoding, where $\beta^{(i)}$ is the scaling coefficient for decoding stage i and $0 \leq \beta^{(i)} \leq 1$.

6.4.2 Simulation Results

Parallel and serial decoding of three dimensional product code, with (16,11) extended Hamming code for each dimension, are simulated. Both SFD and belief propagation based algorithm are taken into consideration. Scaling coefficient $\beta^{(i)}$ is set to 0.5 for $1 \leq i \leq 20$; otherwise, $\beta^{(i)} = 1$.

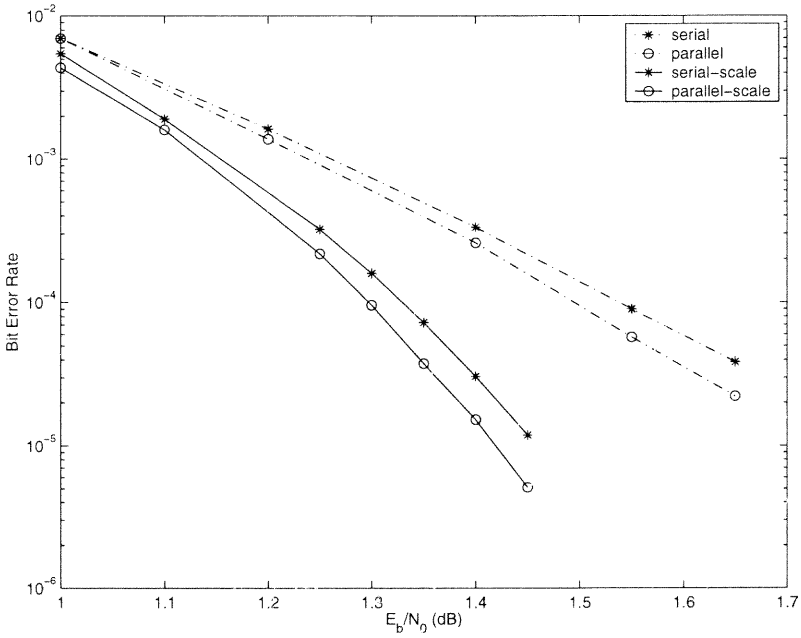


Figure 6.11 BER performance

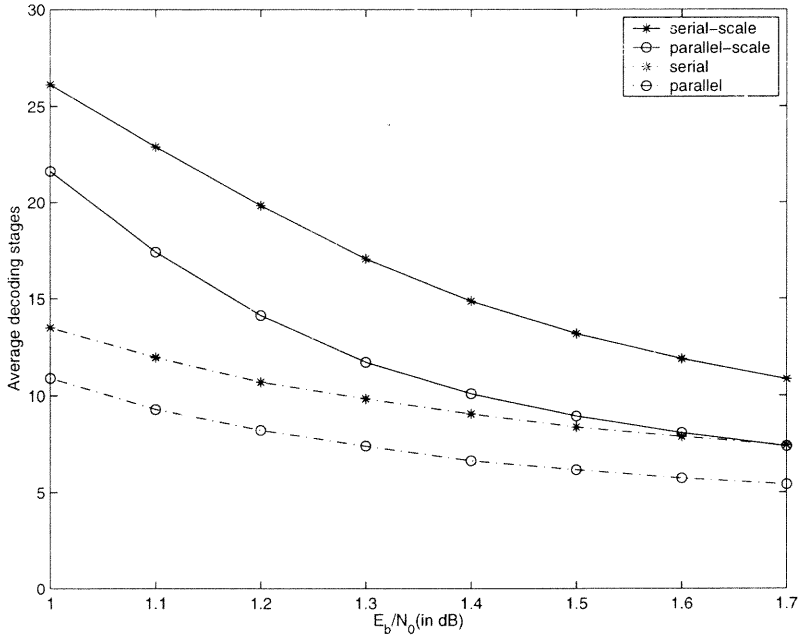


Figure 6.12 Average number of decoding stages

BER performance on AWGN channel is given in Figure 6.11. For both decoding procedures, with and without scaling, parallel decoding offers a slight BER improvement over traditional serial decoding. Set the stopping criteria as: converge to a valid codeword or maximum number of decoding cycles is reached. In this way, the average decoding stages are estimated and shown in Figure 6.12. Parallel decoder converges faster than serial decoder in terms of decoding stages.

It is noticed that in two dimensional codes, parallel decoding yields no BER performance improvement for either PCCC or product code. While in three dimensional codes, some improvement is achieved for both PCCC [29, 56] and product code. It seems that only in three or higher dimensional compound codes would the inherent bias effect in serial decoding significantly degrades BER performance. Although parallel decoding can hardly offer any improvement in terms of BER performance, parallelism does reduce decoding delays by 20-30%.

6.5 Summary

In this chapter, the author makes the efforts to depict the iterative decoding of product code in the framework of belief propagation. The elegant soft-in soft-out iterative decoding algorithm yields excellent BER performance. However, it is found that further BER performance improvement is achievable by scaling the extrinsic information. Parallel decoding, an alternative decoder structure within the framework of belief propagation, is investigated. The following conclusions are drawn,

1. The graphical model of product codes contains short uniform cycles. This makes it somewhat different from LDPC codes and turbo codes, which have long random cycles. Although BP-based algorithm achieves great success in decoding product codes, the short uniform cycles have detrimental effects

on BER performance because they make the a priori information severely dependent on each other at very early decoding stages.

2. Scaled factor decoding (SFD) set the a priori information to scaled extrinsic information, instead of the exact extrinsic information. Simulation results prove that this simple modification make the a priori information less dependent on each other. Some improvement on BER performance over the BP-based algorithm is observed. This means the scaling operation mitigates the negative impact of uniform short cycles in BP-based decoding.

3. A comparison between simulated lower bound, BP-based decoding and SFD shows that SFD offers near-optimum performance.

4. Parallel iterative decoding is conceptually better in that the constituent decoders for each dimension make equal contributions to the final decoding results. However, the improvement over traditional serial decoding is merely 0.05 dB on E_b/N_0. Indeed the average number of decoding stages is reduced by 20-30%.

CHAPTER 7

CONCLUSIONS

7.1 Conclusions

Through the studies in Chapters 3, 4 and 5, it is found that the density evolution procedure is a powerful tool in estimating the capacity of iterative decoding for LDPC codes. In Chapter 3, the capacity of Max-Log-MAP decoding is derived using density evolution. Interestingly enough, it is about 0.5-0.6 dB away from that of Log-MAP decoding, almost the same as the gap in turbo code. In Chapter 4, discretized density evolution is developed for quantized Log-MAP and Max-Log-MAP decoding. The theoretical capacities obtained reflect the quantization loss very well. In chapter 5, density evolution is used in the presence of mismatched channel SNR. Again it works very well. The theoretical thresholds obtained predict well the E_b/N_0 penalty due to mismatched channel SNR.

LDPC codes have certain advantages for implementation, such as fully parallelizable decoder structures. Chapters 4 and 5 deal with two implementation issues, the influences of quantization and SNR mismatch. It is indicated that the key point in designing a decoder under low quantization resolution is to pick a proper dynamic range. The quantization loss of a 4-bit decoder over infinite precision scheme could be kept remarkably low, if the dynamic range is chosen wisely. This is quite an encouraging result for practical fixed-point implementations. In general, the amount of SNR offset that can be tolerated depends on the code length. Longer LDPC codes require more accurate channel estimations. In turbo code, Max-Log-MAP decoding is recommended to replace Log-MAP decoding if accurate SNR estimation is not available. In contrast, for LDPC codes, Log-MAP decoding still outperforms Max-Log-MAP decoding even if an SNR offset of -2 to +3 dB is involved.

Chapter 6 discusses iterative decoding for product codes. Linear correlation coefficients are used to measure the dependency among extrinsic information. It is found that for product code, which contains many uniform short cycles in its graphical model, the extrinsic information (or a priori information, for the next decoding stage) becomes heavily dependent on each other only after two or three decoding stages. Scaling the extrinsic information would mitigate the dependency at early decoding stages and eventually improve BER performance. The modified algorithm is termed scaled factor decoding (SFD). Compared with BP-based decoding, SFD requires about 0.3 dB less E_b/N_0 to achieve a BER of 10^{-5}, in two and three dimensional product code with (16,11) extended Hamming code as its component code.

Parallel iterative decoding for product code is also presented in Chapter 6. It is observed that in three dimensional product code, parallel decoding only brings about a very slight improvement, say 0.05 dB, over serial decoding, though parallel decoding is conceptually better than serial decoding. However, parallel decoding reduces the average number of decoding stages by about 20-30%.

It is known that belief propagation renders optimum decoding results only when the graphical model is loop-free. In the context of compound codes, such as turbo code, LDPC code and product code, the graphical models contain many cycles. A direct application of Pearl's algorithm may work well for some of these codes, such as turbo code and product code. For LDPC code, a range-limiting operation must be included in the decoding algorithm. For product code, a scaling operation on extrinsic information further improves the BER performance. Therefore, scaling and clipping could be viewed as operations mitigating the negative cycle effects in belief propagation.

7.2 Contributions and Future Work

This dissertation documents the research that has been done to date. In summary, the following contributions have been made:

1. Performed a thorough survey on the most recent research on capacity of the iterative decoding algorithm. Two approximate schemes were evaluated, namely SNR-evolution and mutual information evolution on LDPC code. Despite the rough assumptions, they offer quite accurate capacity estimation.

2. Derived the procedures for Max-Log-MAP decoding on LDPC code. This suboptimal algorithm not only reduces the computation burden, but also eliminates the requirement for SNR estimation.

3. Developed a numerical procedure for density evolution under Max-Log-MAP decoding. Using this tool, the capacity of LDPC codes on any memoriless channel could be easily computed.

4. Derived the decoding capacity of LDPC codes under quantized implementations. The most important design parameter in quantized decoder is the dynamic range. Simulation results indicate that a 4-bit quantized implementation only requires 0.1-0.2 dB higher E_b/N_0 in order to achieve the same level of BER as the infinite precision implementation.

5. Investigated the influence of clipping limit on general Log-MAP decoder for LDPC codes.

6. Investigated the sensitivity of Log-MAP LDPC decoder to channel SNR mismatch. The theoretical decoding capacity in presence of fixed channel SNR offset was derived.

7. Studied product code as an extreme case of graphical code, which contains uniform short cycles. Proposed to use linear correlation coefficient to measure the dependency among extrinsic information.

8. Improved the BP-based decoding algorithm by scaling the extrinsic information (scaled factor decoding).

9. Studied parallel iterative decoding for product code.

As a new and active research area, iterative decoding deserves much more research efforts. Further research is suggested as.

1. Quantization issues on turbo codes. Current research is based on simulation results [44, 57, 58]. Given that density evolution has been extended to turbo code [59], it would be quite interesting to find out the capacity of quantized turbo decoders. Note that currently the quantization schemes for turbo code employ higher dynamic ranges than those that have been suggested for LDPC code. It would be interesting to investigate if lowering the dynamic range would bring about any improvements for turbo code.

2. Clipping and scaling operations mitigate the negative cycle effects in LDPC code and product code, respectively. However, it is not clear if they work for general graph based codes. Further survey and investigations are suggested.

APPENDIX A

EQUIVALENCE OF SUM-PRODUCT ALGORITHM AND LOG-MAP ALGORITHM

In this appendix, it will be shown that sum-product algorithm for LDPC code is identical to Hagenauer's iterative decoding schedule executed in parallel mode, where the component decoder is nothing else but the optimum "symbol-by-symbol" LOG-MAP decoder using dual code.

For the convenience of proving, the sum-product algorithm for decoding LDPC code with parity check matrix $\mathbf{H} = [H_{mn}]$ is repeated as below. Denote the set of bits n that participate in check m by $N(m) = \{n : H_{mn} = 1\}$. Similarly, the set of checks in which bit n participates is denoted as $M(n) = \{m : H_{mn} = 1\}$. Denote a set $N(m)$ with bit n excluded by $N(m)\backslash n$, and a set $M(n)$ with parity check m excluded by $M(n)\backslash m$. The sum-product algorithm for LDPC code consists of the following steps:

- *Initialization:* The variables q^0_{mn} and q^1_{mn} are initialized to the values of p^0_n and p^1_n respectively.

- *Horizontal pass:* Run through all the checks m and compute for each $n \in N(m)$, $a = 0, 1$:

$$r^a_{mn} = (1/2)(1 + (-1)^a \delta r_{mn}) \tag{A.1}$$

  where

$$\delta r_{mn} = \prod_{n' \in N(m)\backslash n} (q^0_{mn'} - q^1_{mn'})$$

- *Vertical Pass:* For each n and m, and for $a = 0, 1$, update:

$$q^a_{mn} = \alpha_{mn} p^a_n \prod_{m' \in M(n)\backslash m} r^a_{m'n} \tag{A.2}$$

  where α_{mn} is a normalizing parameter such that $q^0_{mn} + q^1_{mn} = 1$.

- *Decisions:* For each bit n and $a = 0, 1$, update the "pseudo-posterior probabilities":

$$q_n^a = \alpha_n p_n^a \prod_{m \in M(n)} r_{mn}^a \tag{A.3}$$

where α_n is chosen such that $q_n^0 + q_n^1 = 1$. The decision so far is given by $\hat{\mathbf{x}} = [\hat{x}_n]$ such that $\hat{x}_n = 1$ if $q_n^1 > 0.5$; otherwise $\hat{x}_n = 0$. If $\hat{\mathbf{x}}$ is a valid codeword such that $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$, then the algorithm halts; otherwise, the horizontal and vertical pass are repeated until some maximal number of iteration is reached without a valid decoding.

Define the log-likelihood ratio symbol set as follows: intrinsic value $L_n \triangleq \log \frac{p_n^0}{p_n^1}$, a posteriori log-likelihood ratio for making final decisions $L(\hat{x}_n) \triangleq \log \frac{q_n^0}{q_n^1}$, extrinsic information in log measure $\xi_{mn} \triangleq \log \frac{r_{mn}^0}{r_{mn}^1}$ and the input to check node $\zeta_{mn} \triangleq \log \frac{q_{mn}^0}{q_{mn}^1}$. Then the horizontal pass of iteration i in log-likelihood domain could be represented by,

$$\begin{aligned} \xi_{mn}^{(i)} &= \log \frac{1 + \delta r_{mn}}{1 - \delta r_{mn}} \\ &= \log \frac{1 + \prod_{n' \in N(m) \backslash n} (e^{\zeta_{mn'}^{(i-1)}} - 1)}{1 - \prod_{n' \in N(m) \backslash n} (e^{\zeta_{mn'}^{(i-1)}} - 1)} \\ &= 2 \tanh^{-1} \Big(\prod_{n' \in N(m) \backslash n} \tanh(\zeta_{mn'}^{(i-1)}/2) \Big) \end{aligned} \tag{A.4}$$

Similarly, the vertical pass is simply,

$$\zeta_{mn}^{(i)} = \log \frac{q_{mn}^0}{q_{mn}^1} = L_n + \sum_{m' \in M(n) \backslash m} \xi_{m'n}^{(i)} \tag{A.5}$$

and the decision variable,

$$L^{(i)}(\hat{x}_n) = L_n + \sum_{m \in M(n)} \xi_{mn}^{(i)}. \tag{A.6}$$

Each check node could be viewed as a MAP decoder and they are activated simultaneously to extract extrinsic information ξ_{mn} and pass it to the variable nodes.

The variable nodes collect extrinsic information from the check nodes and combine them to prepare for the next decoding iteration. Comparing the horizontal pass with Hagenauer's equation (13) in his paper [32], it is quite clear that they are exactly identical. The only difference is that here parallel/distributive decoding procedure is employed while Hagenauer uses serial decoding fashion.

REFERENCES

[1] Shannon, C.E., "A mathematical theory of communication," *Bell Sys. Tech. J.*, 1948.

[2] Berrou, C. and Glavieux, A., "Near-optimum error correcting coding and decoding: Turbo-codes," *IEEE Transaction on Communications*, vol. 44, pp. 1261–1271, Dec. 1996.

[3] Proakis, J.G., *Digital Communications*. McGraw-Hill, 1995.

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

[5] MacKay, D.J.C. and Neal, R.M., "Near shannon limit performance of low density parity check codes," *Electronics Letters*, pp. 457–458, Mar. 1997.

[6] MacKay, D.J.C., "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, pp. 399–431, Mar. 1999.

[7] McEliece, R.J., MacKay, D.J.C., and Cheng, J., "Turbo decoding as an instance of pearl's 'belief propagation' algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 140–152, Feb. 1998.

[8] Kschischang, F.R. and Frey, B.J., "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 219–230, Feb. 1998.

[9] Richardson, T.J., Shokrollahi, M.A., and Urbanke, R.L., "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, pp. 619–637, Feb. 2001.

[10] Luby, M.G., Mitzenmacher, M., Shokrollahi, M.A., and Spielman, D.A., "Improved low-density parity-check codes using irregular graphs," *IEEE Transactions on Information Theory*, pp. 585–598, Feb. 2001.

[11] Chung, S., "On the design of low-density parity-check codes within 0.0045 db of the shannon limit," *IEEE Communications Letters*, Feb. 2001.

[12] Elias, P., "Error-free coding," *IRE Trans. Information Theory*, vol. 4, pp. 29–37, Sept. 1954.

[13] Lodge, J., Young, R.and Hoeher, P., and Hagenauer, J., "Separable map 'filters' for the decoding of product and concatenated codes," in *International Communication Conference*, 1993.

[14] Wei, X. and Akansu, A.N., "Density evolution for low-density parity-check codes under max-log-map decoding," *IEE Electronics Letters*, pp. 1225–1226, Aug. 2001.

[15] Wei, X. and Akansu, A.N., "Quantized decoding for low-density parity-check codes," *submitted to IEEE Communications Letters.*

[16] Wei, X. and Akansu, A.N., "Decoding capacity of low-density parity-check codes under snr mismatch," *submitted to IEE Electronics Letters.*

[17] Wei, X. and Akansu, A.N., "An improved iterative decoding algorithm for turbo product code," in *Proceedings of 38th Allerton Conference on Communications,Control,and Computing*, 2000.

[18] Wei, X. and Akansu, A.N., "Iterative decoding of product code: Correlated extrinsic information and its impact," in *The 35th Annual Conference on Information Sciences and Systems*, 2001.

[19] Wei, X. and Akansu, A.N., "Iterative decoding of product code: approaching ml performance," in *Proceedings of 5th World Multiconference on Systemics, Cybernetics and Informatics*, 2001.

[20] Wei, X. and Akansu, A.N., "On parallel iterative decoding of product code," in *Proceedings of IEEE Vehicular Technology Conference*, 2001.

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

[22] Wu, Y., *Design and Implementation of Parallel and Serial Concatenated Convolutional Codes.* Ph.D Proposal Report, 1999.

[23] Cover, T. and Thomas, J., *Elements of Information Theory.* New York: Wiley Interscience, 1991.

[24] Frey, B.J., *Graphical Models for Machine Learning and Digital Communication.* MIT Press, 1998.

[25] Tanner,R.M., "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, pp. 533–547, Sept. 1981.

[26] Wiberg, N., *Codes and Decoding on General Graphs.* Linkoping Studies in Science and Technology Dissertation No. 440, 1996.

[27] Kschischang, F.R., Frey, B.J., and Loeliger, H., "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, pp. 498–519, Feb. 2001.

[28] Lentmaier, M. and Zigangirov, K.S., "On generalized low-density parity-check codes based on hamming component codes," *IEEE Communications Letters*, Aug. 1999.

[29] Kim, S. and Wicker, S.B., "Improved turbo decoding through belief propagation," in *Proceeding of GLOBECOM*, 1999.

[30] Meshkat, P. and Villasenor, J.D., "Generalized version of turbo decoding in the framework of bayesian networks and pearl's belief propagation algorithm," in *International Communication Conference*, 1998.

[31] Yoon, S. and Bar-Ness, Y., "Parallel decoding of turbo codes using blocked belief propagation algorithm," in *Proceedings of 39th Allerton Conference on Communications, Control, and Computing*, 2001.

[32] Hagenauer, J., Offer, E., and Papke, L., "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, pp. 429–445, Mar. 1996.

[33] Sason, I. and Shamai, S., "Improved upper bounds on the ensemble performance of ml decoded low density parity check codes," *IEEE Communications Letters*, Mar. 2000.

[34] Richardson, T.J. and Urbanke, R.A., "The capacity of low-density parity check codes under message-passing decoding," *IEEE Transactions on Information Theory*, pp. 599–618, Feb. 2001.

[35] Gamal, H.E. and Hammons, A.R.Jr., "Analyzing the turbo decoder using the gaussian approximation," *IEEE Transactions on Information Theory*, pp. 671–686, Feb. 2001.

[36] Chung, S., Richardson, T.J., and Urbanke, R.L., "Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation," *IEEE Transactions on Information Theory*, pp. 657–670, Feb. 2001.

[37] Brink, S.T., "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transaction on Communications*, pp. 1727–1737, Oct. 2001.

[38] Ping, L., Chan, S., and Yeung, K.L., "Iterative decoding of multi-dimensional concatenated single parity check codes," in *International Communication Conference*, 1998.

[39] Fossorier, M.P.C., Mihaljevic, M., and Imai, H., "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Transaction on Communications*, May 1999.

[40] Leon-Garcia, A., *Probability and Random Processes for Electrical Engineering.* Addison Wesley, 1994.

[41] Ping, L. and Leung, W.K., "Decoding low density parity check codes with finite quantization bits," *IEEE Communications Letters*, pp. 62–64, Feb. 2000.

[42] Wu, Y. and Woerner, B.D., "The influence of quantization and fixed point arithmetic upon the ber performance of turbo codes," in *Proceedings of IEEE Vehicular Technology Conference*, 1999.

[43] Chung, S., *On the Construction of Some Capacity-Approaching Coding Schemes.* Ph.D Dissertation, 2000.

[44] Michel, H., Worm, A., and Wehn, N., "Influence of quantization on the bit-error performance of turbo-decoders," in *Proceedings of IEEE Vehicular Technology Conference*, pp. 581–585, 2000.

[45] Summers, T.A. and Wilson, S.G., "Snr mismatch and online estimation in turbo decoding," *IEEE Transaction on Communications*, pp. 421–423, Apr. 1998.

[46] Worm, A., Hoeher, P., and Wehn, N., "Turbo-decoding without snr estimation," *IEEE Communications Letters*, pp. 193–195, June 2000.

[47] Jordan, M. and Nichols, R., "The effects of channel characteristics on turbo code performance," in *Proc. Milcom'96*, pp. 17–21, Oct. 1996.

[48] Valenti, M.C. and Woerner, B.D., "Performance of turbo-codes in interleaved flat fading channels with estimated channel state information," in *Proceedings of IEEE Vehicular Technology Conference*, pp. 66–70, May 1998.

[49] Reddy, S.M., "On decoding iterated codes," *IEEE Transactions on Information Theory*, pp. 624–627, Sept. 1970.

[50] Reddy, S.M. and Robinson, J.P., "Random error and burst correction by iterated codes," *IEEE Transactions on Information Theory*, pp. 182–185, Jan. 1972.

[51] Pyndiah, R.M., "Near-optimum decoding of product codes: Block turbo codes," *IEEE Transaction on Communications*, vol. 46, pp. 1003–1010, Aug. 1998.

[52] Lucas, R., Bossert, M., and Breitbach, M., "On iterative soft-decision decoding of linear binary block codes and product codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 276–296, Feb. 1998.

[53] Fang, J., Buda, F., and Lemois, E., "Turbo product code: A well suitable solution to wireless packet transmission for very low error rates," in *2nd International Symposium on Turbo Codes & Related Topics*, 2001.

[54] Hartmann, C.R.P. and Rudolph, L.D., "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Transactions on Information Theory*, vol. 22, pp. 514–517, Sept. 1976.

[55] Hunt, A., *Hyper-codes: high-performance low-complexity error-correcting codes.* Master's thesis, 1998.

[56] Heegard, C. and Wicker, S.B., *Turbo Coding.* Kluwer Academic, 1998.

[57] Michel, H. and Wehn, N., "Turbo-decoder quantization for umts," *IEEE Communications Letters*, pp. 55–57, Feb. 2001.

[58] Montorsi, G. and Benedetto, S., "Design of fixed-point iterative decoders for concatenated codes with interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 871–882, May 2001.

[59] Divsalar, D., Dolinar, S., and Pollara, F., "Iterative turbo decoder analysis based on density evolution," *IEEE Transactions on Information Theory*, pp. 891–907, May 2001.