u Ottawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Zhong Cheng

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

On the Design of Raptor Codes Over Gaussian Channels

TITRE DE LA THÈSE / TITLE OF THESIS

Yongyi Mao

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

F. Danilo-Lemoine

A. Yongacoglu

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

ON THE DESIGN OF RAPTOR CODES OVER GAUSSIAN CHANNELS

ZHONG CHENG

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of
Master of Applied Science, Electrical Engineering

September 2006

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 978-0-494-25753-1
Our file Notre référence
ISBN: 978-0-494-25753-1

To my parents

On The Design Of Raptor Codes Over Gaussian Channels

Master of Applied Science, Electrical Engineering Thesis
School of Information Technology and Engineering
University of Ottawa

by Zhong Cheng
September 2006

Abstract

We investigate the design of Raptor codes — a class of fountain codes — over Gaussian channels. In this thesis, we prove that there exists no universal choice of input degree distribution for constructing capacity-achieving Raptor codes using mean-LLR EXIT chart approach. An approach of finding input and output degree distribution jointly is proposed. Through the codes constructed by this approach, we show our approach uniformly outperforms an existing heuristic approach over all Gaussian channels. However, there still exists a gap to channel capacity for Raptor codes constructed by mean-LLR EXIT chart based approach.

Acknowledgments

First and foremost I would like to express my deep and sincere appreciation to my supervisor, Prof. Yongyi Mao, for his open-minded guidance, enthusiastic encouragement and deep understanding on my thesis work. His wide knowledge, profound insight, and logical thinking, especially the excellent mathematical talent have been of precious value for me. His supervising style, full of personality, makes me enjoy my life of study very much.

Next, I am deeply grateful to Jeff Castura, for his generous help to share the source code for simulations, helpful tutorials and suggestions. I would not be able to test the performance of the constructed codes easily without his help.

Many thanks go to Xueying Xie for discussions of mathematical proofs and for other friendly suggestions on programming. Ketai Hu, my study partner during the two years of graduate study, offered me advice on simulations based on his newly designed decoding algorithm, to whom I owe my warm thanks. I also wish to thank Lei Jin, my office partner, for her kind sharing of useful resources.

Additionally, the rest of our group members are all appreciated, from whom I benefit much through occasional discussions.

Finally, I have my great regard to my family. My loving gratitude goes to my wife Jie Tang, for her loving support both materially and spiritually, and my special gratitude goes to my brother Shi Cheng, for his generous sharing of ideas on research. I also owe my warm thanks to my parents for their endless encouragement and stimulation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Fountain codes

Fountain codes are the first realization of rateless codes. Including both LT codes and Raptor codes, they are originally designed for erasure channels (such as that of Internet) with reliability and efficiency achieved simultaneously [1, 2]. Without any channel knowledge known at transmitter or receiver, fountain codes are capable of achieving the capacity over erasure channels universally. Such a success has indicated a new paradigm for communications under channel uncertainty [3, 4].

Research on communications under other settings of channel uncertainty has then been stimulated by the invention of fountain codes. In recent years, fountain codes were tested over binary symmetric channels (BSC), binary input additive white Gaussian noise (BIAWGN) channels and fading channels [5–8]. With the rapid development of wireless communication technology, channel uncertainty remains as a bottle neck, where typically the channel uncertainty introduces random fading to the strength of the received signal. As an application, fountain codes have been used for communications over wireless channels, and capacity-approaching performance is shown in [7].

1.2 Motivation and thesis contribution

In addition to the applications over noisy channels accompanied by the performance tests, much attention has been received in the area of fountain code construction with reliability and efficiency achieved simultaneously. Specifically, this thesis investigates the design of Raptor codes — a class of fountain codes — over BIAWGN channels. We are primarily motivated by the theoretical results and the design approach based on extrinsic information transfer (EXIT) chart presented in [8].

Fountain codes can be represented by factor graphs and thus the iterative sum-product decoding algorithm is applied to its decoding over noisy channels [6, 8]. Over BIAWGN channels, LT codes exhibit performance with an error floor, while as an extension of LT codes, Raptor codes are possible to achieve capacity with the remaining errors of LT codes being corrected by the pre-code. Via information-theoretical analysis, the capacity-achieving Raptor codes must have the properties that the fraction of degree 1 output symbols converges to zero and the fraction of degree 2 output symbols converges to a lower bound determined by the underlying channel, as the length of codes approaches infinity. These results essentially imply the non-existence of universal Raptor codes over BIAWGN channels. Hence, the Raptor code construction for a given BIAWGN channel is imperative.

As a variant of density evolution (DE) [9], the semi-Gaussian approximation based on EXIT chart, originally proposed in [10] for low-density parity-check (LDPC) code construction, has been modified slightly in [8]. The mean of log-likelihood ratio (LLR) messages from input symbols to output symbols is used as the evolving parameter, and hence the approach is called the mean-LLR EXIT chart approach in this thesis. Raptor codes can be designed by linear programming based on mean-LLR EXIT chart.

In this thesis, we investigate the effectiveness of the mean-LLR EXIT chart approach theoretically. The decoding error probability converges to a certain low level during iterations as the mean of LLR messages passed from input symbols to output symbols increases. Nevertheless, due to the difference between LDPC codes and Raptor codes, whether mean-LLR EXIT chart approaches necessarily result in capacity-achieving Raptor codes is of research interest.

The contribution of this thesis is summarized as follows.

1) We prove that there exists no universal choice of input degree distribution for designing capacity-achieving Raptor codes using mean-LLR EXIT chart approach over BIAWGN channels.

2) A method of jointly finding input degree distribution and output degree distribution is proposed.

3) By simulation of codes constructed for various Gaussian channels, the new method is shown to uniformly perform better than an existing heuristic-α method [11].

4) The simulation results also show that in high signal-to-noise ratio (SNR) region, a performance gap to channel capacity exists for Raptor codes designed by mean-LLR EXIT chart based approach.

Based on the linear programming setting, a theoretical treatment is provided by a precise mathematical proof in the thesis, which argues that if the average degree of input symbols α is set *a priori*, then there always exists a typical channel, and other channels with higher SNR, for which the constructed Raptor codes will have output symbol degrees concentrated at the maximally allowed value in the linear program.

However, the existing theoretical results show that the fraction of degree 2 output symbols has to converge to a lower bound determined by channel, which essentially forms a necessary condition for Raptor codes to achieve capacity. If we fix α *a priori*, the approach may not be able to construct capacity-achieving Raptor codes for a certain range of Gaussian channels, since the necessary condition is violated. It means no universal choice of input degree distribution, or equivalently, α, can be used to construct capacity-achieving Raptor codes.

Then arises the question how we can find an optimal α for the design of Raptor codes. We explain an existing heuristic-α method, suggested in [11], where α is chosen slightly above a lower bound. The lower bound is derived from the maximal LLR mean and the mean of channel LLR. However, this method provides only an empirical way of choosing α, and although the lower bound can be obtained clearly, the choice of α is not optimal.

We propose a new approach for the construction of Raptor codes, namely, the optimizing-α method, where instead of choosing α *a priori*, we search for the suitable α over a fairly large range and simultaneously construct the corresponding degree distribution. For a given proper range of α, such an approach generates the optimized codes in terms of the code rate, and can be generally used without knowing the changing behavior of code rate with respect to α.

From the degree distributions constructed by both the heuristic-α approach and the optimizing-α approach for various Gaussian channels, it can be seen that the codes meet the necessary condition for achieving good performance in low SNR channel region, where the fraction of degree 2 output symbols is lower bounded according to the specific channel parameter, while in high SNR channel region, the necessary constraint is still violated. Furthermore, we resort to Monte Carlo simulations to test the performance of the codes pairwise designed by both approaches.

In the thesis, for the purpose of comparison, simulations are performed not only for Raptor codes, but also for LT codes alone. In terms of the different criterion applied to evaluate different codes, two types of simulations are carried out for each constructed degree distribution. Results are rather straightforward, which show that the new optimizing-α approach outperforms the heuristic-α approach uniformly.

In the context of Raptor codes, realized rate is commonly used to judge the performance of codes. In our simulations, it is shown that the Raptor codes constructed by both mean-LLR EXIT chart-based approaches exhibit a performance gap to capacity over high SNR channels. This result can be interpreted as being attributed to the violation of the necessary condition for capacity-achieving performance in high SNR channel.

1.3 Thesis organization

The remaining parts of thesis are organized as follows.

In Chapter 2, after briefly introducing the basis of rateless codes, we orderly introduce the necessary background around the concepts of the sum-product algorithm associated with factor graph, the paradigm of codes on graph, LDPC codes, and properties of fountain codes over erasure channels.

We investigate in Chapter 3 the existing theoretical results of Raptor codes over binary input memoryless symmetric channels (BIMSCs), and an existing approach for Raptor code construction for BIAWGN channels. The discussion around the existing approach is presented, followed by a heuristic-α method [11] and a new modified optimizing-α method. Simulation results are also studied.

Chapter 4 briefly concludes the thesis and the future work is to be expected.

Chapter 2

Background

2.1 Rateless coding paradigm

The performance of a communication system is generally affected by the channel statistics, and hence, channel knowledge is of particular significance in the design of the communication system. However, in many communication situations, especially over wireless channels, channel knowledge is difficult to obtain, which demands the design of the communication system not to rely on complete channel knowledge. For example, in the design of a wireless communication system, one may need to take into consideration that channel state information (CSI) is missing, either at the transmitter, or at the receiver, or at both.

Consider wireless fading channel as an example, where the channel state information — the fading coefficient is not available at the transmitter. Using a conventional fixed-rate coding scheme, a k-bit information sequence is encoded into an n-symbol codeword, giving rise to *code rate* k/n. For each channel state sequence of length n, there is an upper limit of communication rate — the capacity [12], when the transmission power and noise variance are fixed. According to the channel coding theorem in [12], the code rate below the capacity of a channel permits reliable communication, while the code rate above the capacity results in unreliable communication, or outage events. Since the channel capacity is a random variable in fading channel, fixing communication rate *a priori* will inevitably result in outage events, and the probability of such events increases

with the choice of code rate. Thus there exists a clear trade-off between *efficiency* and *reliability*.

To solve this problem, some other coding schemes have been proposed, such as adaptive coding and modulation, which typically involves feeding back the channel state information to the transmitter. This may allow the transmitter to choose a code with rate better matching channel capacity, but when the channel varies rather frequently, such an approach fails to be efficient.

Fountain codes [1, 2] are the first realization of rateless codes. Originally designed for data transmission over the Internet (modeled as an erasure channel), they have been demonstrated to achieve efficiency and reliability simultaneously without channel knowledge [7, 13]. Different from the fixed-rate codes, in the rateless coding paradigm, the transmitter encodes a k-bit information sequence into an infinite sequence of codeword symbols, and transmits the encoded symbols sequentially through the channel. The receiver keeps trying to decode the information sequence at a set of consecutive time instants as the noisy symbols keep arriving. When the decoder successfully decodes, it sends an `ACK` through a dedicated feedback channel to the transmitter to terminate the transmission of the current codeword. The transmitter then stops the transmission and starts to transmit the stream of next codeword symbols. We note that such a feedback channel with only one bit information transmission can be easily implemented.

LT codes and Raptor codes are two classes of fountain codes, with difference being that Raptor codes are simply the extension of LT codes via a pre-code. Over erasure channels, both LT codes and Raptor codes have been demonstrated to be capacity-achieving universally, namely, that the codes can be efficient and reliable for all erasure channels simultaneously, without the channel knowledge known at the transmitter or the receiver [1, 2].

Since being invented for erasure channels, fountain codes have also been tested over a large range of channels, such as BSC channels, Gaussian channels, and fading channels [5–8]. Specifically, LT codes have been shown to exhibit an error floor behavior over BSC and Gaussian channels, while Raptor codes can achieve the capacity of these two classes of channels [5, 6, 8]. Over fading channels, fountain codes can achieve an averaged realized rate quite close to the averaged realized capacity [7]. We note however that

Raptor codes are no longer universally capacity-achieving over Gaussian channels, and designing Raptor codes for Gaussian channels involves additional complexity. Some of these aspects will be reviewed in Section 3.1.1, and further investigation of these aspects forms the central topic of this thesis.

2.2 Factor graphs and codes on graph

2.2.1 Factor graphs and the sum-product algorithm

Derived from Tanner graph [14] and Tanner-Wiberg-Loeliger graph [15], a factor graph is a graphical representation of multivariate functions. A multiplicative factor graph, or simply *factor graph*, is a bipartite graph consisting of two types of nodes, *variable nodes* and *function (factor) nodes*, where a variable node is connected to a function node by an *edge* if the variable is an argument of the function, and we term the number of the edges connected to a node the *degree* of the node. For example, Figure 2.1 is the factor graph representation of the function

$$F(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_2) f_2(x_2, x_3, x_4) f_3(x_2, x_4) f_4(x_3, x_5),$$

where the function F is termed *global function*, and the functions f_1, f_2, f_3, and f_4 are all termed *local functions*. Thus, the factor graph represents the product of all the functions represented by the function nodes.

In many situations, upon the factor graph representation of a global function as the product of local functions, one might be interested in computing the marginal function involving only one variable. For example, the global function is a joint probability density function (pdf), and we need to marginalize it to obtain the pdf of a single random variable. The sum-product algorithm provides an efficient way to compute all the marginal functions simultaneously by the distributive law and reuse of the intermediate results.

Suppose the global function $F(x_1, \ldots, x_n)$ can be factorized into the product of local functions as

$$F(x_1, x_2, \ldots, x_n) := \prod_{i=1}^{m} f_i(X_i), \tag{2.1}$$

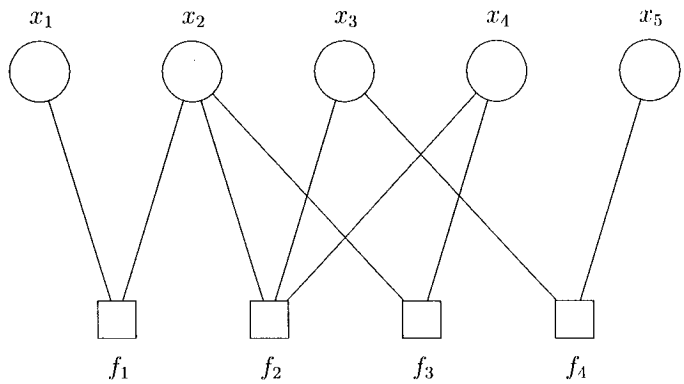

Figure 2.1: A factor graph representation of $f_1(x_1, x_2)f_2(x_2, x_3, x_4)f_3(x_2, x_4)f_4(x_3, x_5)$ with square representing function node, and circle representing variable node.

where X_i is the subset of $\{x_1, x_2, \ldots, x_n\}$ which contains the variables involved in the local function f_i. Clearly, the structure of factorizing F can be represented by a factor graph. Next, we term the function

$$F_i(x_i) := \sum_{\{x_1, x_2, \ldots, x_n\} \setminus \{x_i\}} F(x_1, x_2, \ldots, x_n) \tag{2.2}$$

as *marginal function*, where $i \in \{1, 2, \ldots, n\}$. Based on the full specifications on all the local functions, the sum-product algorithm is then exploited to compute all the marginal functions.

Example 1 *Suppose the factor graph in Figure 2.1 represent the global function F, the*

sum-product algorithm is to compute the marginal function

$$
\begin{aligned}
F_1(x_1) &= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_1(x_1,x_2)f_2(x_2,x_3,x_4)f_3(x_2,x_4)f_4(x_3,x_5),\\
F_2(x_2) &= \sum_{x_1}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_1(x_1,x_2)f_2(x_2,x_3,x_4)f_3(x_2,x_4)f_4(x_3,x_5),\\
F_3(x_3) &= \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} f_1(x_1,x_2)f_2(x_2,x_3,x_4)f_3(x_2,x_4)f_4(x_3,x_5),\\
F_4(x_4) &= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_5} f_1(x_1,x_2)f_2(x_2,x_3,x_4)f_3(x_2,x_4)f_4(x_3,x_5),\\
F_5(x_5) &= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_4} f_1(x_1,x_2)f_2(x_2,x_3,x_4)f_3(x_2,x_4)f_4(x_3,x_5).
\end{aligned}
$$

As seen from Example 1, the sum-product algorithm is so named due to the fact that only summation and multiplication are involved in computing marginal functions. On the factor graph representation of a global function, the sum-product algorithm operates in an iterative way with the messages passed bidirectionally along each edge between the neighboring variable node and function node. In terms of the direction of messages, two different types of messages are considered. We then denote the message passed from a function node f to a variable node u by $\mu_{f\to u}$, and the message passed from a variable node u to a function node f by $\mu_{u\to f}$, where both $\mu_{f\to u}$ and $\mu_{u\to f}$ are functions of variable u.

Using the two types of messages, we will then pass messages in the graph following the certain message updating rules. Clearly, two message passing rules are needed to make the algorithm iterative, which are function-node update rule and variable-node update rule, where a downstream message is always computed by using the upstream messages. We now give the precise description of the update rules.

For convenience, in the factor graph representation, we denote the set of neighbors of a particular node v by $\mathcal{N}(v)$. Then, at function node, the message passed from function node f to variable node u can be computed using the incoming messages from variable nodes $\mathcal{N}(f)\setminus\{u\}$ as follows,

Function node update

$$\mu_{f\to u}(u) := \sum_{\mathcal{N}(f)\setminus\{u\}} f(\mathcal{N}(f)) \prod_{u_i\in\mathcal{N}(f)\setminus\{u\}} \mu_{u_i\to f}(u_i) \tag{2.3}$$

Example 2 *Figure 2.2 shows a function node connected with several variable nodes, so the message passed from function node f to variable node u is computed as*

$$\mu_{f\to u}(u) = \sum_{u_1,u_2,u_3} f(u,u_1,u_2,u_3)\mu_{u_1\to f}(u_1)\mu_{u_2\to f}(u_2)\mu_{u_3\to f}(u_3).$$

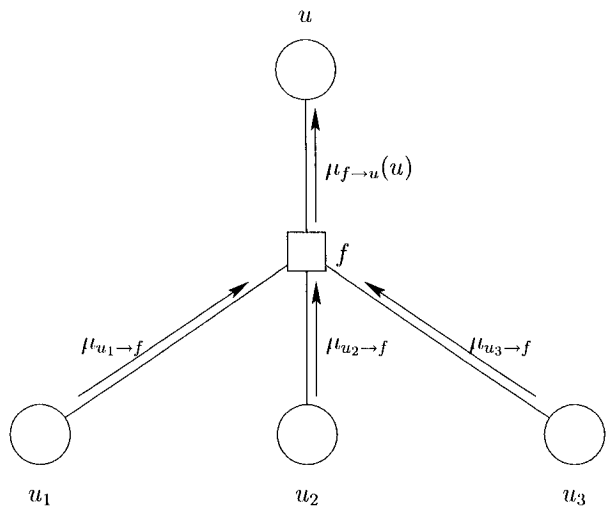

Figure 2.2: Message passing at function node f according to (2.3).

At variable node, the message passed from variable node u to function node f can be computed using the incoming messages from function nodes $\mathcal{N}(u)\setminus\{f\}$ as follows,

Variable node update

$$\mu_{u\to f}(u) := \prod_{f_i\in\mathcal{N}(u)\setminus\{f\}} \mu_{f_i\to u}(u) \tag{2.4}$$

Example 3 *Figure 2.3 shows a variable node with several neighboring function nodes, so the message passed from variable node u to function node f is computed as*

$$\mu_{u\to f}(u) = \mu_{f_1\to u}(u)\mu_{f_2\to u}(u)\mu_{f_3\to u}(u)$$

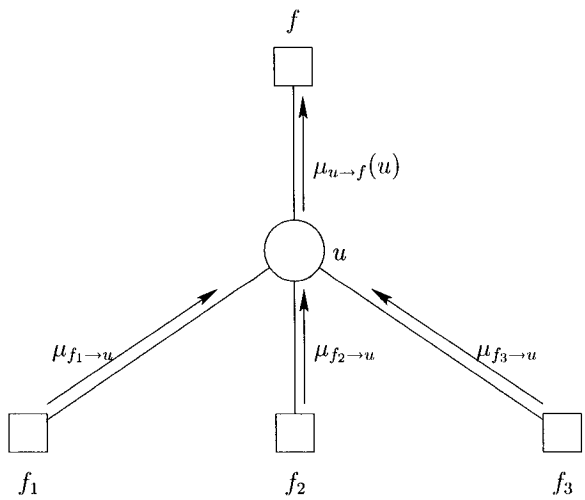

Figure 2.3: Message passing at variable node u according to (2.4).

Besides the iterative updating rules at function node and variable node, another type of messages termed *summary message* needs to be computed, especially when decision is to be made. We denote the summary message of a variable node u by μ_u, which is again a function of variable u. Specifically, the summary message μ_u is computed by using all the incoming messages passed from neighboring function nodes as follows,

Summary message

$$\mu_u(u) := \prod_{f_i\in\mathcal{N}(u)} \mu_{f_i\to u}(u) \tag{2.5}$$

Example 4 *Figure 2.4 shows a variable node connected to several function nodes, so the summary message of variable node u is computed as*

$$\mu_u(u) = \mu_{f_1\to u}(u)\mu_{f_2\to u}(u)\mu_{f_3\to u}(u)\mu_{f_4\to u}(u).$$

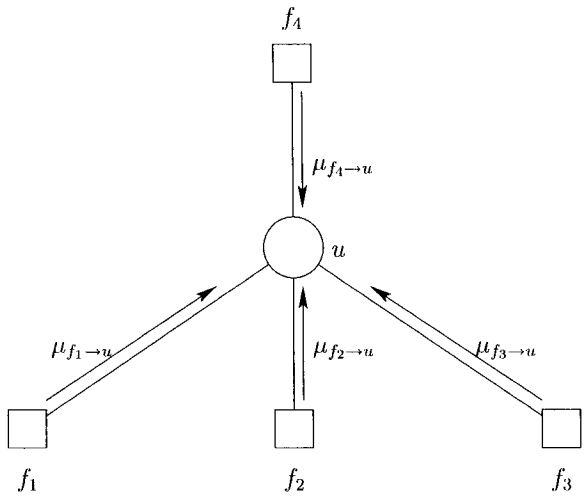

Figure 2.4: Summary message for variable node u according to (2.5).

Based on these well defined local message updating rules in (2.3), (2.4) and (2.5), the sum-product algorithm can be built freely and efficiently in terms of the structure of graph. However, the sum-product algorithm essentially involves three stages as follows,

1) initialization stage, where in terms of the problem setting, a set of messages are initialized on the graph;

2) propagation stage, where messages are passed on the graph according to message update rules orderly;

3) termination stage, where by some certain criterion, the message passing is terminated, and the summary messages are computed if needed.

As an illustration, ideally, we will deal with a cycle-free, or rooted tree graph, where the messages passed to a particular node from all its neighboring nodes are independent of each other. The three stages of the algorithm are then described as

Sum-product algorithm on tree graphs

Initialization: Leaf nodes pass messages first. If a leaf node is variable node u, then $\mu_{u\to f}(u) := 1$, where f is the only neighbor of u; if a leaf node is function node f, then $\mu_{f\to u}(u) := f(u)$, where u is the only neighbor of f.

Propagation: A node only passes messages to a neighbor if the messages from all other neighbors have arrived.

Termination: When each variable node receives messages from all neighbors, it computes its summary message. When every variable node has computed its summary message, the algorithm terminates.

It has been shown in [16, 17] that if factor graph $\mathcal{G}$ representing the factorization of $f(x_1, x_2, \ldots, x_n)$ in (2.1) is cycle-free, then all the marginal functions can be expressed arithmetically, which means, applying the sum-product algorithm, we obtain the resulting summary message $\mu_{x_i}(x_i)$ for each x_i, $i \in \{1, 2, \ldots, n\}$ exactly same as $F_i(x_i)$ in (2.2).

The sum-product algorithm on tree graphs not only always produces the exact results of marginal functions, but also maintains the efficiency of computation. Essentially, in the sum-product algorithm, the fundamental law that allows the saving of computation is the distributive law between multiplication and addition [18], where the same factor of additions can be extracted out and multiplied later. Therefore, the algorithm requires the intermediate results computed for calculating one marginal function be saved and reused for calculating other marginal functions if needed.

Example 5 *Suppose we apply the sum-product algorithm on the factor graph in Figure 2.1, typically for calculating F_1 and F_2. We will obtain the following computations.*

$$F_1(x_1) = \sum_{x_2} f_1(x_1, x_2) \sum_{x_3, x_4} f_2(x_2, x_3, x_4) f_3(x_2, x_4) \sum_{x_5} f_4(x_3, x_5)$$

$$F_2(x_2) = \sum_{x_1} f_1(x_1, x_2) \sum_{x_3, x_4} f_2(x_2, x_3, x_4) f_3(x_2, x_4) \sum_{x_5} f_4(x_3, x_5)$$

Note that during the execution of the sum-product algorithm on the graph, both of the terms $\sum_{x_3,x_4} f_2(x_2,x_3,x_4)f_3(x_2,x_4)$ *and* $\sum_{x_5} f_4(x_3,x_5)$ *need to be only computed once and reused for computing* F_1 *and* F_2.

For factor graphs with cycles, depending on the structure of graph, a large range family of variants dealing with the executions of initialization, propagation, and termination stages has been exploited [19–22]. However, there is no existing implementation of the sum-product algorithm which can induce an exact solution for computing marginal functions. In recent years, accompanied by the notion of codes on graph, which we will postpone to the discussion later in this chapter, the sum-product algorithm became more and more widely used as decoding algorithm. Although cycles may exist in the graph, experimental results have suggested that when the graph is large and sparse (in the sense of having small quantities of edges), the computation of all marginal functions can be well approximated [17, 23]. In such case, demonstrated by the decoding of codes on graphs [19], if the objective is to find the value of x_i that maximizes the marginal function $F_i(x_i)$ for each i, the algorithm performs extremely well.

In the context of decoding for codes on graph, based on the well-established *maximum a posteriori probability* (MAP) criterion [24] and factor graph representation of code, if we let the global function $F(x_1,x_2,\ldots,x_n)$ in (2.1) be the joint pdf of all variable nodes, and the marginal function $F_i(x_i)$ be the marginal pdf for each x_i, the decoding problem can be formulated as the problem of simultaneously finding many marginal functions (scaled by a factor) of a global function, which can be essentially solved by the sum-product algorithm. To make a decision, the decoder only needs to find the value of x_i maximizing the marginal pdf $F_i(x_i)$. This procedure is terminologically known as *belief-propagation* (BP) algorithm, in which the information from received noisy codeword symbols is passed in the graph, and the belief of a transmitted codeword symbol becomes more and more precise during iterations.

To illustrate the three stages of decoding procedure by the sum-product algorithm, we typically focus on the case that the factor graph is constructed from the parity-check matrix of a code, which is shown as

Sum-product decoding algorithm (applied to a parity-check matrix)

Initialization: Set the messages from channel according to the specific graph structure, with all the other messages set to a constant;

Propagation: Messages iterate in the way that all variable nodes representing codeword symbols pass messages, then all function nodes representing parity-check constraints pass messages;

Termination: After a pre-determined criterion is reached, compute the summary messages for all variable nodes and terminate the algorithm.

Clearly, with cycles in the graph, the sum-product algorithm does not guarantee to result in the exact marginal functions desired for decoding. Nevertheless, under the MAP detection rule, the performance of the algorithm is close to optimum, which has been acted as the standard approach for decoding a large family of error correcting codes with the performance extremely close to Shannon's theoretical limit [19, 25–27].

2.2.2 Codes on graphs

With the invention of factor graph, the traditional linear block code representation has shifted to the paradigm of codes on graph, typically accompanied by the rediscovery of LDPC codes [19, 26, 28]. The state-of-the-art approach of coding and decoding requires both the graph representation of codes and the sum-product decoding algorithm. We now use the factor-graph language to illustrate this subject.

For simplicity, we introduce a remarkable boolean function $\delta(x)$, namely the *indicator function*, which specifies parity-check constraint of a code, and can be used as the factor of a global function. For x defined as a boolean variable, $\delta(x)$ is defined as

$$\delta(x) := \begin{cases} 1, & \text{if } x \text{ is } \texttt{true}; \\ 0, & \text{if } x \text{ is } \texttt{false}. \end{cases} \tag{2.6}$$

Consider a linear block code C of dimension k and codeword length n, completely

specified by its generator matrix G. We explain the graph representation of generator matrix. For convenience, we illustrate by using an example.

Let the following matrix G be the generator matrix of a Hamming code C.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{2.7}$$

If we denote the set of information bits by vector (m_1, m_2, m_3, m_4) and the codeword by vector $(c_1, c_2, \ldots, c_7)$, then in terms of G, the following constraints are satisfied simultaneously.

$$\begin{aligned} c_1 &= m_1, \\ c_2 &= m_2, \\ c_3 &= m_3, \\ c_4 &= m_4, \\ c_5 &= m_1 \oplus m_2 \oplus m_3, \\ c_6 &= m_2 \oplus m_3 \oplus m_4, \\ c_7 &= m_1 \oplus m_2 \oplus m_4, \end{aligned}$$

where $\oplus$ denotes the operation of modulo 2 addition.

By using the indicator function, these constraints are well defined as follows.

$$\begin{aligned}
f_1(m_1, c_1) &:= \delta(m_1 \oplus c_1 = 0), \\
f_2(m_2, c_2) &:= \delta(m_2 \oplus c_2 = 0), \\
f_3(m_3, c_3) &:= \delta(m_3 \oplus c_3 = 0), \\
f_4(m_4, c_4) &:= \delta(m_4 \oplus c_4 = 0), \\
f_5(m_1, m_2, m_3, c_5) &:= \delta(m_1 \oplus m_2 \oplus m_3 \oplus c_5 = 0), \\
f_6(m_2, m_3, m_4, c_6) &:= \delta(m_2 \oplus m_3 \oplus m_4 \oplus c_6 = 0), \\
f_7(m_1, m_2, m_4, c_7) &:= \delta(m_1 \oplus m_2 \oplus m_4 \oplus c_7 = 0).
\end{aligned}$$

Thus, the global function is expressed as the product of all these indicator functions

$$f_1(m_1, c_1) f_2(m_2, c_2) f_3(m_3, c_3) f_4(m_4, c_4) f_5(m_1, m_2, m_3, c_5) f_6(m_2, m_3, m_4, c_6) f_7(m_1, m_2, m_4, c_7),$$

which specifies the code C completely. Thereby the factorization structure of the global function of C can easily be represented by factor graph, shown in Figure 2.5.

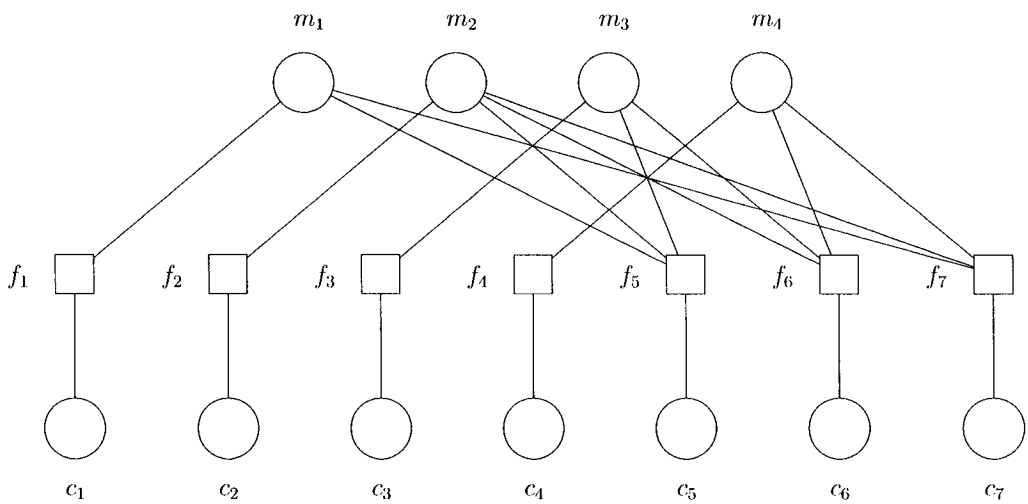

Figure 2.5: Factor graph corresponding to generator matrix G in (2.7).

Alternatively, the linear block code C of dimension k and codeword length n can be determined by its parity-check matrix H. We will use the above example to explain the

graph representation of parity-check matrix.

Let the following matrix H be the parity-check matrix transformed from (2.7).

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

In terms of H, each codeword $(c_1, c_2, \ldots, c_7)$ satisfies the following constraints.

$$\begin{aligned} c_1 \oplus c_2 \oplus c_3 \oplus c_5 &= 0, \\ c_2 \oplus c_3 \oplus c_4 \oplus c_6 &= 0, \\ c_1 \oplus c_2 \oplus c_4 \oplus c_7 &= 0. \end{aligned}$$

By using the indicator function, these constraints are well defined as follows.

$$\begin{aligned} f_1(c_1, c_2, c_3, c_5) &:= \delta(c_1 \oplus c_2 \oplus c_3 \oplus c_5 = 0), \\ f_2(c_2, c_3, c_4, c_6) &:= \delta(c_2 \oplus c_3 \oplus c_4 \oplus c_6 = 0), \\ f_3(c_1, c_2, c_4, c_7) &:= \delta(c_1 \oplus c_2 \oplus c_4 \oplus c_7 = 0). \end{aligned}$$

Thus, the global function is expressed as the product of all these indicator functions

$$f_1(c_1, c_2, c_3, c_5) f_2(c_2, c_3 c_4, c_6) f_3(c_1, c_2, c_4, c_7)$$

which specifies all constraints of code C. Thus the factorization structure of the global function of C can easily be represented by factor graph, shown in Figure 2.6.

Beyond this special example, there exists a large family of randomly constructed codes such as LDPC codes and fountain codes, where a particular code is generated in the way that an edge attached to a node of degree i follows a probability distribution under the graph representation. In the context of codes on graph, degree distribution is an important parameter of codes. Obviously, two types of degree distributions corresponding to two types of nodes in factor graph are involved, namely, *variable degree distribution* and *function (check) degree distribution.* With such notion, the modern codes can be

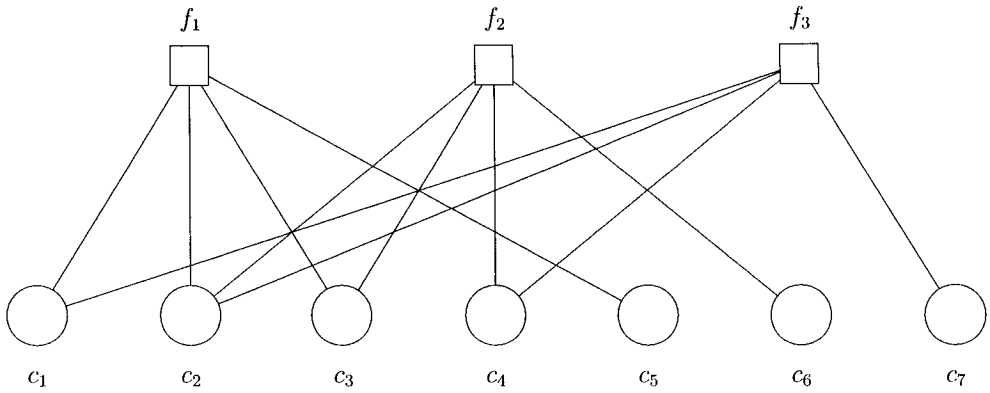

Figure 2.6: Factor graph corresponding to parity-check matrix H in (2.8).

defined. Degree distribution plays an important role on the performance of codes, and we will investigate it more precisely later in both LDPC codes scenario and fountain codes scenario.

2.3 Low-density parity-check (LDPC) codes

Experiencing invention in 1960s by Gallager [28] and rediscovery during the recent years [26], LDPC codes become an important class of linear block codes, which are so called due to the fact that in terms of the parity-check matrix, there is only a small fraction of "1"'s in it.

Based on the factor graph representation of LDPC codes, each of the codeword bits represents a *variable node*, while each parity-check constraint represents a *check node*, equivalent to function node in general factor graph. The factor graph of an example of LDPC codes is shown in Figure 2.7. The ensemble of LDPC codes with same length is identified by the degree distributions $\{\lambda_i\}$ and $\{\rho_j\}$, and each of codes in the ensemble has nearly the same performance. In LDPC codes literature, $\{\lambda_i\}$ is termed *variable (left) degree distribution*, and $\{\rho_j\}$ is termed *check (right) degree distribution*, where λ_i and ρ_j are the probability of an edge being attached to a degree i variable node and a degree j check node respectively. If there is only one element involved in $\{\lambda_i\}$ or $\{\rho_j\}$, such a degree distribution is a *regular* distribution, and an *irregular* distribution otherwise. Given those two degree distributions $\{\lambda_i\}$ and $\{\rho_j\}$, the rate of the ensemble can be

obtained as

$$R = 1 - \frac{\sum \rho_j / j}{\sum \lambda_i / i}. \tag{2.9}$$

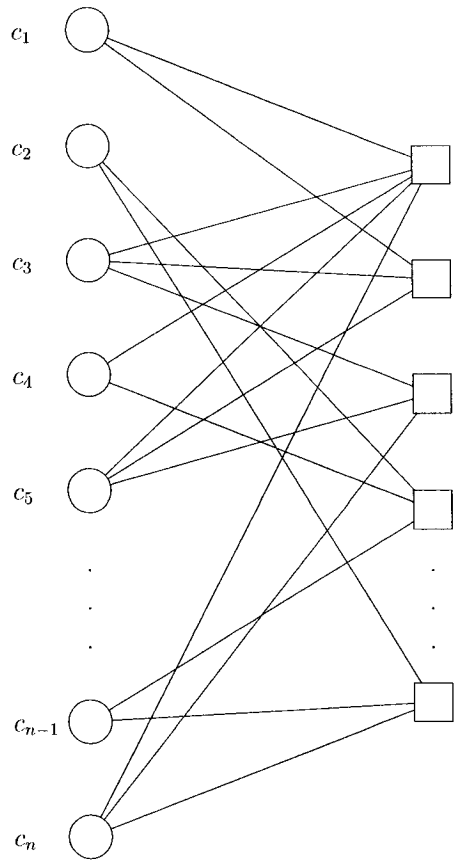

Figure 2.7: The factor graph of a LDPC code.

The suboptimal decoding algorithm, the sum-product algorithm, is widely applied to decode LDPC codes with low complexity. With respect to such message-passing decoding algorithm, the asymptotic performance of the ensemble of LDPC codes can be analyzed via a powerful tool, termed *density evolution* (DE) [9]. To validate the density evolution, two assumptions need to be imposed essentially. The following definition is then useful.

Definition 1 [Symmetric Distribution] *Symmetric distribution is the distribution of random variable X with pdf $f(x)$ if and only if*

$$f(x) = e^x f(-x).$$

As an example, a Gaussian random variable follows the symmetric distribution if and only if its variance is twice the mean.

On one hand, the pdf $f(x)$ of LLR messages during the iterative decoding keeps satisfying the condition of symmetric distribution [9], and hence, the all-zero codeword, or equivalently, the all-one codeword after the binary phase shift keying (BPSK) modulation is considered to be transmitted. On the other hand, after round l of the iterative decoding process, the paths of the messages passed along are assumed to be strict trees under the factor graph representation. Thus, when l becomes larger, the factor graph will be assumed to be a group of trees, without any cycles in it, which guarantees the independence among the messages. This fact is impossible for a finite length code, but only valid for an infinite length code.

As an analytical tool for LDPC codes, DE determines the threshold for the given ensemble parameters $\{\lambda_i\}$ and $\{\rho_i\}$, which equivalently gives the average asymptotic performance of the family of codes over a range of channels. *Threshold*, which is a channel parameter, divides the range of channels into two parts of different code performance. We refer to Gaussian channels as an illustration. If the threshold is the channel variance σ^{*2}, then the error probability after l iterations of decoding algorithm tends to zero with l approaching infinity for all the channels with variance smaller than σ^{*2}, whereas for all the channels with variance greater than σ^{*2}, the decoding error probability is bounded away from zero.

DE is not only good at analysis, but acts as a designing tool of LDPC codes as well. Nevertheless, it is such a computation-intensive method that we are interested in other approximation schemes as the replacements. As a result, semi-Gaussian approximation approach based on EXIT chart, known as a one-dimensional (1-D) method typical for Gaussian channels, showed its good performance by the gap between the Shannon limit and the design code rate, and its less computational cost with only one parameter evolving [10].

The work of [10] suggested that due to the central limit theorem, the pdf of LLR messages at the output of variable nodes can be well approximated by symmetric Gaussian or a mixture of symmetric Gaussian pdf's, while the density at the output of check nodes keeps its true pdf. EXIT chart provides a way to describe the transition of information

on transmitted codeword bits during the decoding iterations, which is always expressed as a function $I_{out} = f(I_{in}, I_0)$, where I_0, I_{in}, and I_{out} are channel information, information from previous iteration, and output information of current iteration, respectively. Based on such semi-Gaussian assumption, the error probability in iterative decoding, which is well approximated by the tail probability of symmetric Gaussian distribution for LLR, can be chosen as the measure of the information. Therefore, the EXIT chart approach, combined with the semi-Gaussian assumption, sufficiently provides a good approximation of DE. On the other hand, for a given Gaussian channel, a good ensemble of LDPC codes can be found by this method together with linear programming, with little computational cost.

2.4 Fountain codes

Fountain codes are a class of rateless codes, originally designed for erasure channels. From a given set of *input symbols*, the potentially unlimited number of encoding symbols, which we term as *output symbols*, can be generated. The original input symbols can be recovered from any set of the output symbols that in aggregate are slightly longer than the input symbols in length. LT codes and Raptor codes are two categories of fountain codes.

2.4.1 LT codes

The first realization of a class of rateless codes for erasure channels is furnished by LT codes, which was introduced by Luby in his recent landmark paper [1]. In this class, let k be a positive integer, and let $\mathcal{D}$ be a *degree distribution* on F_2^k, the LT codes are typically specified by the parameters k and $\mathcal{D}$ jointly. The encoding process is rather straightforward: we denote the binary input symbol vector by $(x_1, x_2, \ldots, x_k)$, and generate each output symbol independently and randomly by first sampling from the distribution $\mathcal{D}$ to obtain a positive number d between 1 and k. Next, d distinct input symbols are chosen uniformly at random, and the value of the output symbol is calculated as the exclusive-or of the d input symbols. This encoding process generates a limitless stream of output symbols.

This process can be illustrated by a toy example of LT codes.

Example 6 *Let* $k = 10$ *and* $\mathcal{D}$ *be* $\{0.1, 0.4, 0.2, 0, 0, 0.2, 0, 0.1, 0, 0\}$ *on* $\{1, \ldots, k\}$*, then a possible realization of the generator matrix is shown as follows.*

$$\begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & \cdots \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \cdots \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \cdots \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & \cdots \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & \cdots \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & \cdots \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots
\end{bmatrix}$$

Clearly the generator matrix has infinite number of columns.

The LT process, referred by Luby in [1], essentially plays an important role in the design and analysis of a good degree distribution $\mathcal{D}$ for LT codes. It has been verified that, the LT process ends successfully if and only if the decoder succeeds to recover all the input symbols. Correspondingly, the sum of degrees of these output symbols equals to the total number of operations in decoding. Thus, obviously, two aspects of design goals are to be met for designing degree distributions, which are exactly the requirements on optimality and efficiency.

- Upon the success of LT process, as few output symbols as possible are needed on average.
- To keep the number of symbol operations low on average in encoding and decoding, as low average degree of output symbols as possible is required.

An example of the good degree distribution is Robust Soliton distribution, invented by Luby in [1].

It has been shown that LT codes can achieve the capacity of any erasure channels without channel statistics known at the transmitter or the receiver. However, if the number of collected output symbols n is close to the number of input symbols k, the encoding complexity (measured in terms of the number of XOR operations) per input symbol grows as the order of $\log(k)$, which prevents the LT codes being transmitted with large k. We shall resort to a new class of fountain codes, Raptor codes, to achieve a constant encoding and decoding cost per input symbol.

2.4.2 Raptor codes

In Raptor codes [2], a linear block *pre-code* $\mathtt{C}$ of length k' and dimension k, usually a high-rate LDPC code is used to encode a k-symbol input vector first, and from the resulted k'-symbol vector, the output symbols are generated by LT codes $(k', \mathcal{D})$. Figure 2.8 shows an example of Raptor codes on factor graph representation. In fountain codes literature, each of k' input symbols of LT codes represents an *input node*, and each parity-check constraint of LT codes represents an *output node*. Often, the degree distribution $\mathcal{D}$ is referred to as *output node degree distribution* with the formal definition as follows.

Definition 2 [Output Node Degree Distribution] *The output node degree distribution of Raptor codes is defined as*

$$\Omega(x) := \sum_{d=1}^{k} \Omega_d x^d,$$

where Ω_d is the probability of a degree d output node being chosen.

Thereby Raptor codes are defined by the parameters $(k, \mathtt{C}, \Omega(x))$.

By choosing a good degree distribution $\Omega(x)$, Raptor codes can achieve the capacity of erasure channels universally with a constant encoding and decoding cost per input symbol, which was described in [2] thoroughly. According to the graph representation, decoding process can be separated into two stages, where in the first stage the LT codes decoding recovers the k'-symbol vector with a certain amount of errors, and in the second stage the k-symbol original input vector is recovered with those left errors being corrected by the LDPC pre-code.

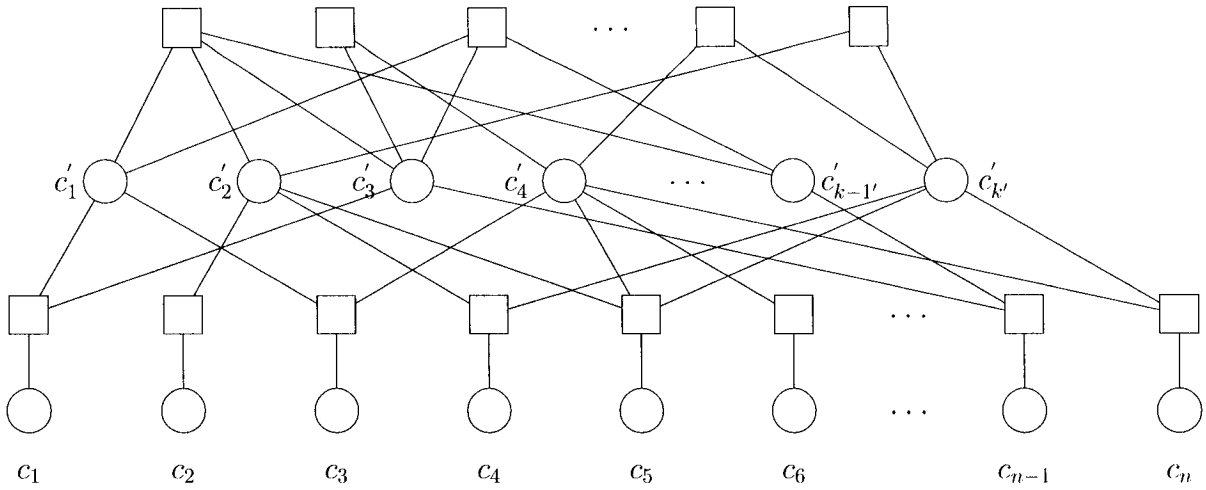

Figure 2.8: The factor graph of a Raptor code

Different from LDPC codes in Section 2.3, Raptor codes have only one type of degree distribution on output symbols specified as the parameter of codes. However, for the purpose of analysis, the degree distribution on input symbol side needs to be considered, with the following definition.

Definition 3 [Input Node Degree Distribution] *The input node degree distribution of Raptor codes is defined as*

$$I(x) := \sum_d I_d x^d,$$

where I_d is the probability of a degree d input node being chosen.

We notice that, for the purpose of convenience, the edge degree distributions need to be used instead of the node degree distributions. The following definitions are then needed.

Definition 4 [Input Edge Degree Distribution] *The input edge degree distribution is defined as*

$$\iota(x) := \sum_d \iota_d x^{d-1},$$

where ι_d is the probability of an edge being connected to a degree d input node.

Definition 5 [Output Edge Degree Distribution] *The output edge degree distribution is defined as*

$$\omega(x) := \sum_{d} \omega_d x^{d-1},$$

where ω_d is the probability of an edge being connected to a degree d output node.

It can be easily verified that

$$\iota(x) = \frac{I'(x)}{I'(1)} \tag{2.10}$$

and

$$\omega(x) = \frac{\Omega'(x)}{\Omega'(1)}, \tag{2.11}$$

where $I'(x)$ and $\Omega'(x)$ are the derivatives of $I(x)$ and $\Omega(x)$ with respect to x respectively.

It has been shown in [2], that both the degree distributions $I(x)$ and $\iota(x)$ can be well approximated by $e^{\alpha(x-1)}$, which follows the form of Poisson distribution, and determined by the only parameter α, termed the *average degree of input symbols*. With the optimizing method, [2] presented several optimized output node degree distributions which are close to the Soliton distribution.

At this end, we are ready to turn to the discussion of the designing approach for Raptor codes over Gaussian channels.

Chapter 3

Raptor Codes over Gaussian Channels

3.1 Existing results and open questions

As the development of this thesis is primarily built upon the work of [8], we here briefly outline the results thereof.

3.1.1 Theoretical results

The work of [8] presents several important results of Raptor codes over binary input memoryless symmetric channels (BIMSCs).

BIMSCs include three well-known examples, which are the binary erasure channel (BEC) with erasure probability ϵ, denoted by BEC(ϵ), the binary symmetric channel (BSC) with error probability ϵ, denoted by BSC(ϵ), and the BIAWGN with noise variance σ^2, denoted by BIAWGN(σ), where the decoder's LLR messages follow the pdf of symmetric distribution defined in Section 2.3.

Over such BIMSCs, we consider transmission with BPSK modulation, where the input symbol is a variable from $\{-1, +1\}$. We assume that the transmitter uses a Raptor code and the receiver uses the sum-product algorithm to decode. Under the one-to-one correspondence between $\{-1, +1\}$ and $\{0, 1\}$ defined by $-1 \leftrightarrow 1$ and $+1 \leftrightarrow 0$, we may

treat the output symbols of Raptor codes as being obtained by taking the real product of some subset of input symbols. Nevertheless we will abuse our terminology and call multiplication operation on $\{-1,+1\}$ "XOR".

Using distribution of LLR messages (see in Section 2.3), the capacity of a BIMSC can be characterized as follows. Assuming that the all-one codeword after modulation is sent through channel $\mathcal{C}$ and $f(x)$ is the pdf of channel LLR, we give the channel capacity as

$$\mathrm{Cap}(\mathcal{C}) := 1 - \int_{-\infty}^{+\infty} \log_2(1+e^{-x}) f(x) dx. \tag{3.1}$$

So, for the three examples of BIMSCs, we have

$$\begin{aligned} \mathrm{Cap}(\mathrm{BEC}(\epsilon)) &= 1-\epsilon, \\ \mathrm{Cap}(\mathrm{BSC}(\epsilon)) &= 1-h(\epsilon), \\ \mathrm{Cap}(\mathrm{BIAWGN}(\sigma)) &= 1 - \frac{1}{2\sqrt{\pi\mu}} \int_{-\infty}^{+\infty} \log_2(1+e^{-x}) \\ &\quad \cdot e^{-\frac{(x-\mu)^2}{4\mu}} dx, \end{aligned}$$

where $\mu = 2/\sigma^2$.

Furthermore, letting Z denote the random variable describing the channel LLR, we denote by $\mathrm{B}(\mathcal{C})$ the expectation $\mathrm{E}(\tanh(Z/2))$, another parameter derived from the channel. So we have

$$\mathrm{B}(\mathcal{C}) := \int_{-\infty}^{+\infty} \tanh\left(\frac{x}{2}\right) f(x) dx. \tag{3.2}$$

Hence, for example,

$$\begin{aligned} \mathrm{B}(\mathrm{BEC}(\epsilon)) &= 1-\epsilon, \\ \mathrm{B}(\mathrm{BSC}(\epsilon)) &= (1-2\epsilon)^2, \end{aligned}$$

and

$$\mathrm{B}(\mathrm{BIAWGN}(\sigma)) = \frac{1}{2\sqrt{\pi\mu}} \int_{-\infty}^{+\infty} \tanh\left(\frac{x}{2}\right) e^{-\frac{(x-\mu)^2}{4\mu}} dx.$$

Based on (3.1) and (3.2), we will introduce one of the main parameters of the channel, denoted by $\Pi(\mathcal{C})$ and defined as

$$\Pi(\mathcal{C}) := \frac{\mathrm{Cap}(\mathcal{C})}{\mathrm{B}(\mathcal{C})}. \tag{3.3}$$

The following examples can be easily verified,

$$\begin{aligned} \Pi(\mathrm{BEC}(\epsilon)) &= 1, \\ \Pi(\mathrm{BSC}(\epsilon)) &= \frac{1-h(\epsilon)}{(1-2\epsilon)^2}, \end{aligned}$$

and

$$\Pi(\mathrm{BIAWGN}(\sigma)) = \frac{1-\frac{1}{2\sqrt{\pi\mu}}\int_{-\infty}^{+\infty}\log_2(1+e^{-x})e^{-\frac{(x-\mu)^2}{4\mu}}dx}{\frac{1}{2\sqrt{\pi\mu}}\int_{-\infty}^{+\infty}\tanh(\frac{x}{2})e^{-\frac{(x-\mu)^2}{4\mu}}dx}.$$

In the context of Raptor codes, for Raptor codes $(k, \mathtt{C}, \Omega(x))$ over a specific BIMSC $\mathcal{C}$, if the error probability of the sum-product decoding algorithm applied to $k/Cap(\mathcal{C}) + o(k)$ output symbols approaches zero as k goes to infinity, the Raptor codes are termed *capacity-achieving* for channel $\mathcal{C}$ [8].

Using an information-theoretic approach, [8] shows that the fractions of output symbols of degree 1 and 2 in the degree distributions have to converge to certain lower bound depending on the underlying channel, if the Raptor codes constructed is capacity-achieving for the channel. The lower bounds of the fractions of degree 1 and 2 output symbols are given respectively as

$$\Omega_1^{(k)} > 0 \text{ and } \lim_{k\to\infty}\Omega_1^{(k)} = 0 \tag{3.4}$$

and

$$\lim_{k\to\infty}\Omega_2^{(k)} = \frac{\Pi(\mathcal{C})}{2}, \tag{3.5}$$

provided that $(k, \mathtt{C}, \Omega^{(k)}(x))$ is capacity-achieving for channel $\mathcal{C}$ with $\mathrm{B}(\mathcal{C}) \neq 0$ and $\mathrm{Cap}(\mathcal{C}) \neq 0$.

For Raptor codes achieving the capacity of BIMSC $\mathcal{C}$, $\Pi(\mathcal{C})/2$ is asymptotically the fraction of output symbols of degree 2, and we denote it by $\Omega_2(\mathcal{C})$ to indicate that it is a parameter of the channel itself, and only determined by channel.

It can be easily seen that, if the BIMSC $\mathcal{C}$ is BEC(ϵ), $\Omega_2(\mathcal{C})$ is independent of the erasure probability of the channel, which exactly takes the value of $1/2$. If the channel is either BSC(ϵ), or BIAWGN(σ), this fraction varies between $1/\ln(16)$ and $1/2$, depending on the particular channel chosen. A plot of $\Omega_2(\mathcal{C})$ over BIAWGN channels is shown in Figure 3.1.

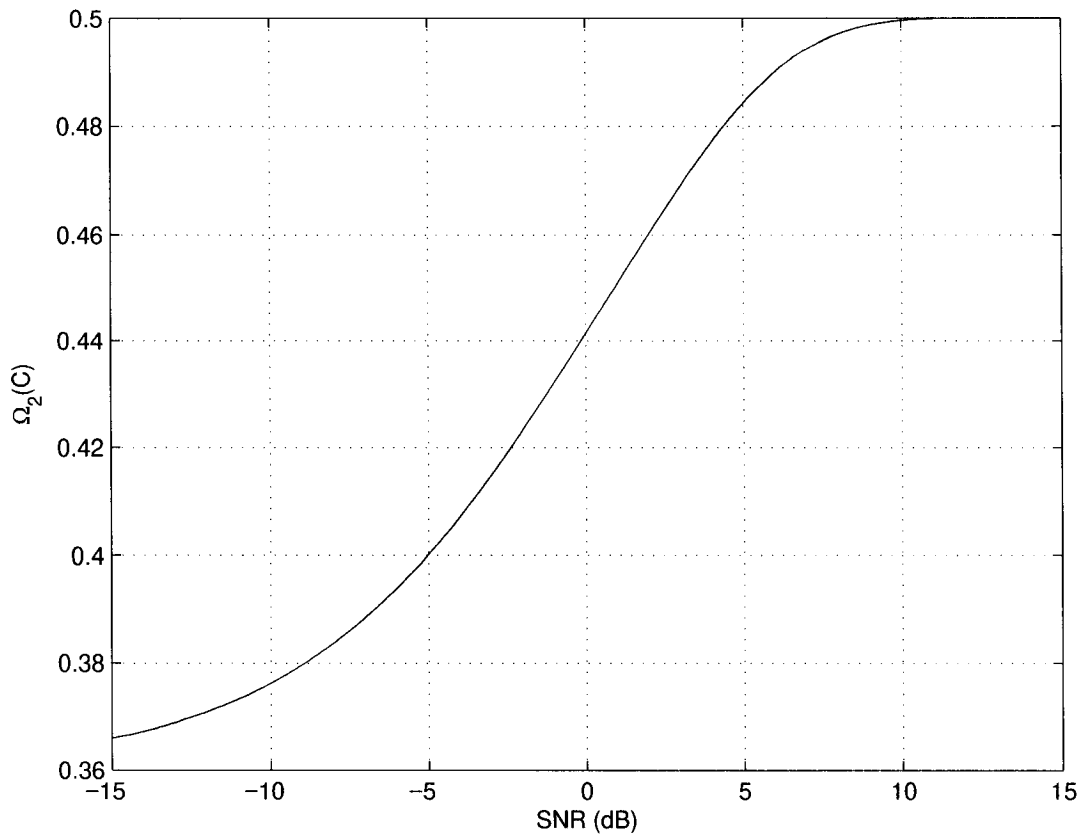

Figure 3.1: $\Omega_2(\mathcal{C})$ as a function of SNR over BIAWGN channels.

Thus, for BIAWGN channels, Ω_2, and hence the degree distribution must adapt to the channel, which results in non-existence of universal Raptor codes. Here universality refers

to that a single code can achieve the capacities of all channels in a family simultaneously.

3.1.2 Mean-LLR EXIT chart approach for code construction

In Section 2.3, we introduced LDPC codes designed by semi-Gaussian approximation method based on EXIT chart. The work of [8] suggests that a variation of EXIT chart based approach, which we refer to as *mean-LLR EXIT chart* approach, can be applied to the construction of Raptor codes.

We assume that the communication channel is a BIAWGN which has variance σ^2. Under the tree assumption and the semi-Gaussian assumption, at any given round, the messages passed from input bits to output bits follow the same symmetric Gaussian distribution, while the messages passed from output bits to input bits follow the same non-Gaussian distribution. Instead of using decoding error probability as the parameter of recursion, we use the mean of messages from input bits to output bits. By the assumption that the codeword transmitted over the channel is the all-one codeword after modulation, which is valid because of the symmetry of the channel, the mean is expected to increase during iterations. The increasing of mean from iteration to iteration guarantees the decreasing of probability of error in decoding.

It is then possible to show that, this EXIT chart based approach leads to a linear programming problem, which can be solved by the existing algorithms such as simplex method.

But, the difference between LDPC codes and Raptor codes requires different problem settings. In our 1-D framework, we refer to as $f_d(\mu)$ the EXIT chart of codes whose output bits are all of degree d. That is, $f_d(\mu)$ is a function of the mean of messages passed from input bits to output bits at some round, which is assumed to follow symmetric Gaussian distribution, and we call $f_d(\mu)$ an "elementary" EXIT chart. Then $f_d(\mu)$ can be interpreted as the expected value of the messages passed from an output bit of degree d to an input bit at the same round, which is expressed as

$$f_d(\mu) := 2\mathrm{E}\left(\operatorname{atanh}\left(\tanh\left(\frac{Z}{2}\right)\prod_{i=1}^{d-1}\tanh\left(\frac{X_i}{2}\right)\right)\right), \tag{3.6}$$

with following specifications,

1) X_i $(i = 1, \ldots, d-1)$ is the symmetric Gaussian random variable with mean μ;

2) Z is the LLR of the channel message, also having symmetric Gaussian pdf with mean $2/\sigma^2$, and σ^2 is the variance of the channel;

3) $X_1, \cdots, X_{d-1}, Z$ are independent of each other.

Figure 3.2 shows an example of these elementary EXIT chart curves.

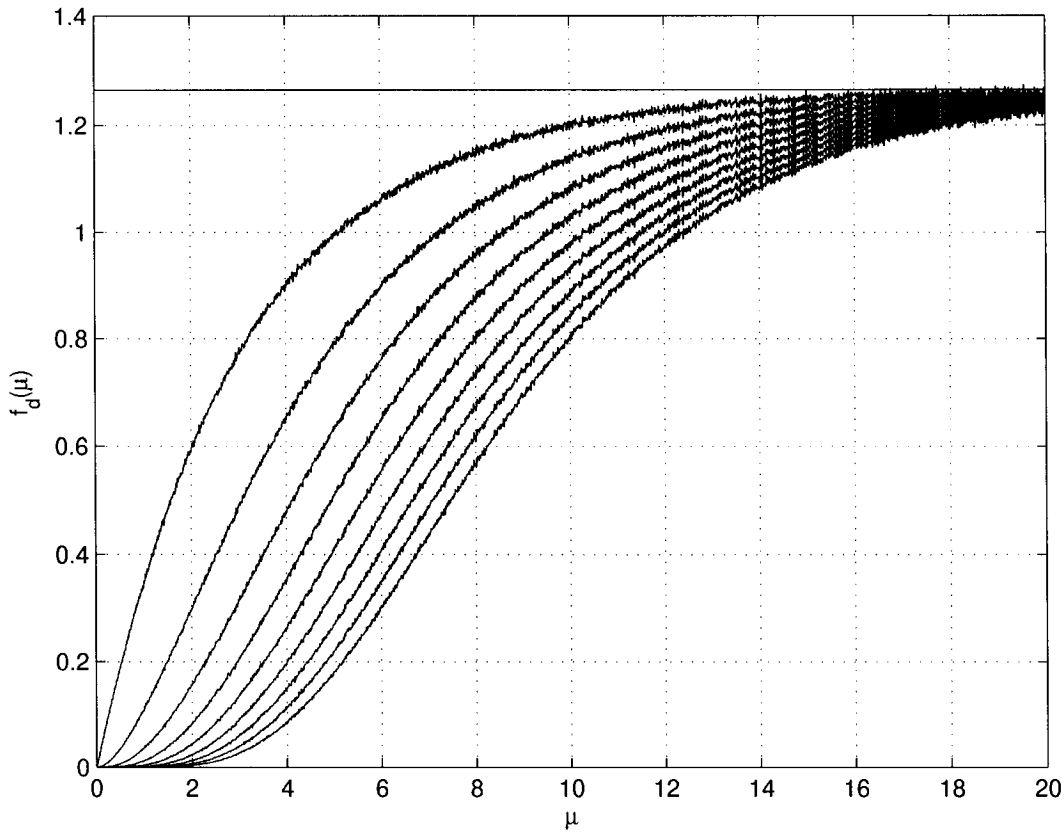

Figure 3.2: Elementary EXIT chart using mean of symmetric Gaussian as parameter. Under the -1.9895 dB channel, the curves correspond to the output bits from degree 1 to degree 10.

Let α denote the average degree of input bits, and $\omega(x) = \sum_d \omega_d x^{d-1}$ is the output edge degree distribution of the Raptor codes, the mean of messages passed from input bits to output bits at the next round equals to

$$\alpha \sum_d \omega_d f_d(\mu).$$

The design code rate is $R = 1/(\alpha \sum \omega_d/d)$, and hence, for a fixed average degree of input bits α, the objective of design is to minimize $\sum \omega_d/d$, while the condition $\alpha \sum_d \omega_d f_d(\mu) > \mu$ holds for all $\mu \in (0, \mu_0]$, where μ_0 corresponds to the fact that when the decoder stops decoding on the LT-code part, the remaining error of the decoded bits would be erased sufficiently by the high rate LDPC pre-code part. Thus, the linear program can be formulated as

Find $\{\omega_d\}$ that

$$\begin{aligned}
&\text{minimizes} && \sum_{d=1}^{D} \omega_d/d \\
&\text{subject to} && \forall\ \mu_i \in (0, \mu_0],\ i = 1, \ldots, N: \ \alpha \sum_{d=1}^{D} \omega_d f_d(\mu_i) > \mu_i \\
& && \sum_{d=1}^{D} \omega_d = 1 \\
& && \forall\ d = 1, \cdots, D: \ \omega_d \geq 0 \\
& && \text{where } D \text{ is the given maximum degree of output bits.}
\end{aligned} \tag{3.7}$$

It is worth noting that for a given channel parameter, the average degree of input bits α has to be set *a priori*, which we will show later in this chapter may not generate capacity-achieving Raptor codes for some certain range of channels.

3.1.3 Open questions

The key results of [8] may be summarized as follows.

1) There exists no universal *output* degree distribution of Raptor codes that achieve the capacity of *all* BIAWGN channels.

2) Given a BIAWGN channel, mean-LLR EXIT chart based linear programming as presented in Section 3.1.2 may be used for the construction of Raptor codes.

Although the authors of [8] presented one example construction of Raptor codes based on this linear programming approach, it remains open how the linear program should be set up, namely, how one should set up parameter α for a given channel. We note that a private communication [11] with the authors of [8], suggests that the choices of α are made heuristically in their linear program, the details of which is postponed to a later section.

Motivated by this fact, this thesis makes an effort to address the following specific questions.

1) Does there exist a universal *input* degree distribution of Raptor codes, or equivalently, α, that can be used to construct capacity-achieving Raptor codes for *all* BIAWGN channels based on the linear programming approach?

   One may intuitively reason that the answer to this question is "No", since a low-capacity channel requires higher n and higher n leads to higher α. This argument is however flawed for the following reason.

   With simple counting "1"'s in the generator matrix row-wise and column-wise, it is easy to see

   $$k\alpha = n\beta$$

   where β is the average degree of output bits. Fixing k, α only grows with n if β is also fixed. But using the mean-LLR EXIT chart based linear programming, β in fact depends on the obtained $\{\omega_d\}$ from the linear program, which depends on α itself. Aware of this subtlety, this question is indeed open.

2) How can we find the universal α if it exists, or an optimal α otherwise for the design of Raptor codes?

The remainder of this chapter answers these two questions. Specifically in Section 3.2, we show that there exists no universal choice of α for constructing Raptor codes over all BIAWGN channels, where we postpone the proof to Section 3.4. In Section 3.3, we present a method in which α and $\{\omega_d\}$ are determined jointly. Section 3.5 presents simulation results of the presented method and makes comparison with a heuristic approach of finding α given by [11].

3.2 There exists no universal α

We first validate the approach of mean-LLR EXIT chart that updates mean of LLR instead of probability of error.

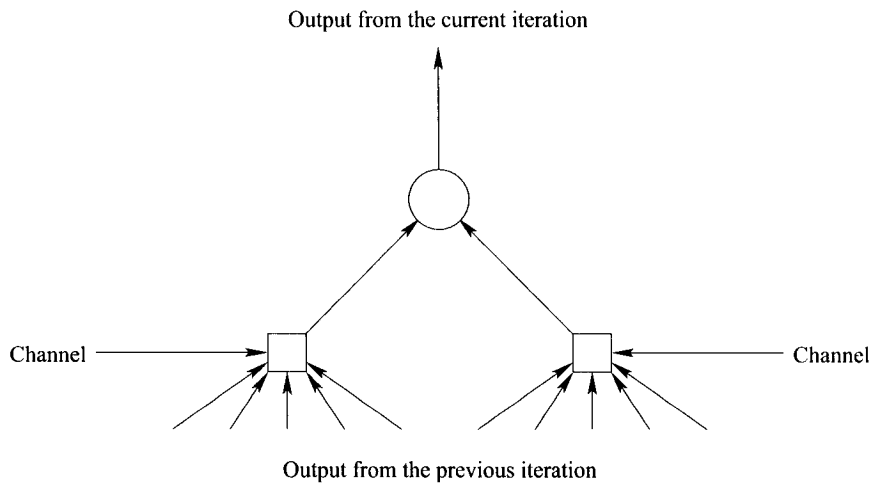

Figure 3.3: .Depth-one tree for a LT code.

Figure 3.3 shows a factor graph for one iteration of decoding LT codes. Such a graph is referred to as a depth-one tree [10]. We denote the mean of symmetric Gaussian density at the output of input bits from the previous iteration by μ_{in} and that from the current iteration by μ_{out}. Accordingly, let $P_{e,in}$ denote the decoding error probability of the previous iteration, and $P_{e,out}$ denote that from the current iteration. Then we have the following simple result.

Proposition 1 *Let μ_{in}, μ_{out}, $P_{e,in}$, and $P_{e,out}$ be defined as above. Then, under the assumption of symmetric Gaussian density at the output of input bits, if $\mu_{in} < \mu_{out}$, we have*

$$P_{e,in} > P_{e,out}$$

Proof: For convenience, we use $\mu_{chk,in}$ to denote the expectation of the messages at the output of output bits from the previous iteration, and $\mu_{chk,out}$ to denote that from the current iteration. Abusing the notation of the input edge degree distribution ι_d with the expectation α, we express the pdf at the output of input bits for the previous iteration as

$$\sum_d \iota_d \mathcal{G}(\mu_{d,in}), \tag{3.8}$$

and the pdf at the output of input bits for the current iteration as

$$\sum_d \iota_d \mathcal{G}(\mu_{d,out}), \tag{3.9}$$

where $\mathcal{G}(\mu)$ is a symmetric Gaussian density with mean μ, $\mu_{d,in}$ and $\mu_{d,out}$ are the mean of density at the output of a degree d input bits for the previous iteration and the current iteration respectively. Obviously,

$$\mu_{d,in} = (d-1)\mu_{chk,in}$$

and

$$\mu_{d,out} = (d-1)\mu_{chk,out}.$$

As a consequence, (3.8) and (3.9) are functions of $\mu_{chk,in}$ and $\mu_{chk,out}$ respectively.

At the same time, it has been shown that

$$\mu_{in} \approx \alpha\mu_{chk,in}$$

and

$$\mu_{out} \approx \alpha\mu_{chk,out},$$

when large block codeword length is applied [8].

Therefore, if $\mu_{in} < \mu_{out}$, then $\mu_{chk,in} < \mu_{chk,out}$. Since the decoding error probability is the tail probability of the mixture of Gaussian distribution, we have $P_{e,in} > P_{e,out}$. Therefore, after a certain rounds of iterations, the decoding error probability achieves below a specific low level. □

Similar to the case of LDPC codes, the mean-LLR EXIT chart approach for Raptor codes guarantees a good convergence behavior of the error probability in decoding. However, due to the difference in the settings between approach for LDPC codes and that for Raptor codes, we will theoretically investigate the effect of the settings on codes constructed. Then the following result applies to that, the proof of which is presented in Section 3.4.

Theorem 1 *For any given α, μ_0 in the linear program (3.7), there exists SNR^*, such that, as $SNR > SNR^*$, the solution is*

$$\Omega(x) = x^D.$$

We now provide some implications of the theorem.

In Section 3.1.1, we investigated the properties of capacity-achieving Raptor codes over BIMSCs. Typically, for BIAWGN channels, the constructed degree distribution is necessary to have its element Ω_2 within a certain range determined by the underlying channel parameter. Nevertheless, the theorem essentially suggests that using the mean-LLR EXIT chart approach, we have to choose α carefully. To be specific, abusing the notations in the approach settings, for parameters σ^2, μ_0, N and D fixed, we have no knowledge on how to choose α *a priori*. It can be easily checked that, if a small α is chosen, we may not obtain a suitable solution since the linear programming itself fails; if a typical large α is chosen, we have the opportunity to obtain the codes with the regular degree distribution $\Omega_D = 1$ holding, and thus $\Omega_2 = 0$ which contradicts with the theoretical result that Ω_2 has to be lower bounded away from zero.

Formally, this leads to the following result.

Corollary 1 *There exists no universal choice of α for the linear program (3.7) to find capacity-achieving Raptor codes for all BIAWGNs.*

That is, if choosing the parameter α *a priori*, in spite of the convergence behavior of the error probability in decoding guaranteed, we are not able to find the capacity-achieving Raptor codes for a range of channels, since the necessary condition on Ω_2 no longer holds for codes constructed for those set of channels.

3.3 Improved EXIT chart approach for code construction

With the main structure of the mean-LLR EXIT chart approach unchanged, we search for an efficient way of choosing a suited α. A practical method on α choosing was suggested by [11], which essentially chooses an α by a lower bound, and we term it the *heuristic-α* approach, which is explained next.

Let σ^2 be the noise variance of BIAWGN and μ_{chk} denote the expected value of messages passed from output bits to input bits. Based on the factor graph representation of Raptor codes, we argue that on a function node, since the messages passed from input bits to output bits are not clean enough, then μ_{chk} is at most the value of the mean of the LLR of the channel, i.e.,

$$\mu_{chk} \leq \mu_{LLR},$$

where μ_{LLR} denotes the mean of the LLR of the channel and equals $2/\sigma^2$. On a variable node, abusing the notation of the mean of messages from input bits to output bits μ, we have

$$\mu = \alpha\mu_{chk}.$$

Thus, we obtain

$$\mu \leq \alpha\mu_{LLR}.$$

Setting the upper limit of μ to be μ_0, we have

$$\alpha \geq \frac{\mu_0}{\mu_{LLR}}. \tag{3.10}$$

Practically, we pick an α slightly greater than the lower bound *a priori*.

Obviously, the meaning of "slightly greater" is not defined precisely. As a consequence, although the lower bound in (3.10) can be determined explicitly, the problem exists in how far α should be chosen away from the lower bound.

Instead of setting α *a priori*, we propose an alternative approach which maximize $R = 1/(\alpha \sum_d \omega_d/d)$ over α and $\{\omega_d\}$ jointly. That is, we propose to solve the following optimization problem.

Find $(\alpha, \{\omega_d\})$ **that**

$$\begin{aligned}
\text{minimizes} \quad & \alpha \sum_{d=1}^{D} \omega_d/d \\
\text{subject to} \quad & \forall\ \mu_i \in (0, \mu_0],\ i = 1, \ldots, N: \ \alpha \sum_{d=1}^{D} \omega_d f_d(\mu_i) > \mu_i \\
& \sum_{d=1}^{D} \omega_d = 1 \\
& \forall\ d = 1, \cdots, D: \ \omega_d \geq 0 \\
& \text{where } D \text{ is the given maximum degree of output bits.}
\end{aligned} \tag{3.11}$$

We note that this optimization problem is not convex since the cost function is not convex. The proof is provided in Appendix I. For the above reason, we choose to sample α within a certain range and for each α sample, solve the linear program in (3.7), and determine optimal α by comparing the solutions of these linear programs. This approach, termed *optimizing-α* approach, is described as follows.

Optimizing-α approach

Initialization:

Set a range for α with lower limit α_L and upper limit α_H. Normally $\alpha_L = 0$ and pick a typical large value as α_H. Sample interval $[\alpha_L, \alpha_H]$ with small Δ and obtain a set $\{\alpha_i\}$, $i = 1, \ldots, M$. Further, choose a small enough δ as precision control.

while $\Delta > \delta$ **do**

  for $i = 1$ to M **do**

    For each α_i, find $\{\omega_d^i\}$ together with the designed code rate R_i according to (3.7) (Note if linear program fails, set $R_i = 0$).

  end for

  Sort R_i $(i = 1, \ldots, M)$ in descending order and choose the first N corresponding α_is, denoted by $\{\alpha_i'\}$, $i = 1, \ldots, N$.

  for $i = 1$ to N **do**

    For each α_i', obtain $\alpha_i = \alpha_i' - \Delta/2$ and $\alpha_{i+N} = \alpha_i' + \Delta/2$.

  end for

  Set $M = 2N$ and $\Delta = \Delta/2$.

end while

Choose α_1' to be the optimized α and corresponding $\{\omega_d'^1\}$ to be constructed degree distribution.

The solution of this problem depends on the chosen range of α. Compared with the heuristic-α approach, this optimizing-α approach can be widely applied without knowing the changing behavior of the code rate with respect to α. An example of the optimized overall EXIT chart (scaled by α) derived by the optimizing-α approach is shown in Figure 3.4, which is above the 45 degree line to guarantee the convergence of the error probability in decoding.

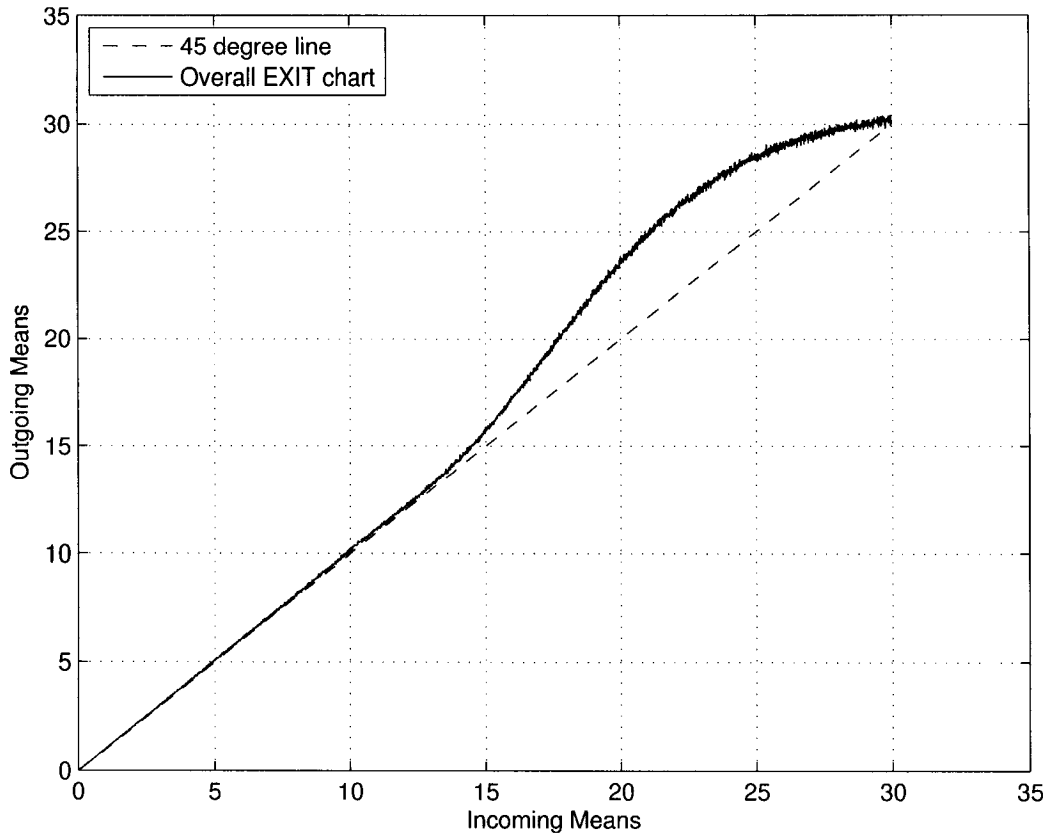

Figure 3.4: Overall EXIT chart (scaled by α) by optimizing-α approach under -1.9895 dB channel.

3.4 Proof of theorem 1

To prove the theorem, we first establish some elementary results.

Lemma 1 *Let X be a symmetric Gaussian random variable with mean μ, and $h(x)$ be an increasing function of x with the property that $h(-x) = -h(x)$, then the expectation of the function $E\left(h(X)\right)$ increases with μ.*

Proof: We express the expectation of the function $h(X)$ as

$$\mathrm{E}\left(h(X)\right) = \int_{-\infty}^{+\infty} h(x) f_X(x) dx,$$

where $f_X(x)$ is the pdf of the random variable X.

By the symmetric condition of X, $f_X(-x) = e^{-x} f_X(x)$ holds. Thus,

$$\begin{aligned}
\mathrm{E}(h(X)) &= \int_{-\infty}^{0} h(x) f_X(x) dx + \int_{0}^{+\infty} h(x) f_X(x) dx \\
&= -\int_{0}^{+\infty} h(x) f_X(-x) dx + \int_{0}^{+\infty} h(x) f_X(x) dx \\
&= \int_{0}^{+\infty} h(x)(f_X(x) - e^{-x} f_X(x)) dx \\
&= \int_{0}^{+\infty} h(x)(1 - e^{-x}) f_X(x) dx.
\end{aligned}$$

Since $h(x)$ is an increasing function of x, and the term $1 - e^{-x}$ also increases with x, we construct a new function for convenience

$$H(x) = \begin{cases} h(x)(1 - e^{-x}) & \text{if } x > 0; \\ 0 & \text{if } x \leq 0, \end{cases}$$

where obviously $H(x)$ strictly increases when $x > 0$.

By such definition of $H(x)$, $\mathrm{E}(h(X))$ can be easily rewritten as

$$\mathrm{E}(h(X)) = \mathrm{E}(H(X)).$$

Now we consider the parameter change of the random variable X, during which the mean μ becomes $\mu + \delta$ with a small positive change of δ, as a consequence, we generate another random variable X' which follows the distribution specified by its mean $\mu + \delta$.

X and X' follow the joint symmetric Gaussian distributions, and either one follows its marginal distribution. On the other hand, since the variance of a symmetric Gaussian random variable is twice the mean, the relation between X and X' is

$$X' = (X - \mu)\sqrt{(\mu + \delta)/\mu} + \mu + \delta.$$

This is a linear transformation from X to X' with the fact that $X' > X$ when $X > 0$, which typically means that if we draw the random variable X and then transform it to X', the sample value of X' is always greater than that of X. Since $H(x)$ is also increasing with x, intuitively, we see that $\mathrm{E}(H(X')) > \mathrm{E}(H(X))$. □

Lemma 2 *Suppose the multivariate function* $y = g(x_1, x_2, \ldots, x_n)$ *has the following properties,*

1) $\forall\ i \in (1, \ldots, n)$, *when* $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ *are fixed,*

$$g(x_1, \ldots, -x_i, \ldots, x_n) = -g(x_1, \ldots, x_i, \ldots, x_n);$$

2) $\forall\ i \in (1, \ldots, n)$, *if* $x_i = 0$, *then* $y = 0$;

3) If $x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0$, *then* $y \geq 0$;

4) If $x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0$, $\forall\ i \in (1, \ldots, n)$, *when* $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ *are fixed, the function* y *strictly increases with* x_i.

Let Y *be a random variable defined as* $Y = g(X_1, X_2, \ldots, X_n)$, *where* $X_1, X_2, \ldots, X_n$ *are all independent symmetric Gaussian random variables with mean* $\mu_1, \mu_2, \ldots, \mu_n$ *respectively, then the expectation* $E(Y)$ *increases with* μ_1 *provided that* $\mu_2, \mu_3, \ldots, \mu_n$ *are all fixed.*

Proof: First we consider the conditional expectation $\mathrm{E}(Y|X_2 = x_2, X_3 = x_3, \ldots, X_n = x_n)$ when $x_2, x_3, \ldots, x_n$ are all particular positive values, which can be expanded as

$$\mathrm{E}(Y|X_2 = x_2, X_3 = x_3, \ldots, X_n = x_n) = \mathrm{E}(g(X_1, x_2, x_3, \ldots, x_n)).$$

where the function g becomes involving only the variable x_1 which can be denoted by $g^*(x_1)$ with the property that g^* increases with x_1, and $g^*(-x_1) = -g^*(x_1)$. By Lemma 1, $\mathrm{E}(g^*(X_1))$ increases with μ_1, which equivalently means $\mathrm{E}(Y|X_2 = x_2, X_3 = x_3, \ldots, X_n = x_n)$ increases with μ_1, if $x_2, x_3, \ldots, x_n$ are all positive.

Now we turn to the expression of $\mathrm{E}(Y)$ as

$$\begin{aligned}\mathrm{E}(Y) &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty}\mathrm{E}(Y|X_2 = x_2, X_3 = x_3, \ldots, X_n = x_n) \\ &\quad \times f_{X_2,X_3,\ldots,X_n}(x_2, x_3, \ldots, x_n)dx_2dx_3\ldots dx_n.\end{aligned}$$

Due to the assumed properties of the function g and the independence among all X_is, by splitting the integration interval for each variable, it can be obtained that

$$\begin{aligned}\mathrm{E}(Y) &= \int_{0}^{+\infty}\int_{0}^{+\infty}\cdots\int_{0}^{+\infty}\mathrm{E}(Y|X_2 = x_2, X_3 = x_3, \ldots, X_n = x_n)(f_{X_2}(x_2) - f_{X_2}(-x_2)) \\ &\quad \times (f_{X_3}(x_3) - f_{X_3}(-x_3))\cdots(f_{X_n}(x_n) - f_{X_n}(-x_n))dx_2dx_3\ldots dx_n,\end{aligned}$$

where $f_{X_i}(-x_i)$ is the mirror pdf of $f_{X_i}(x_i)$ around 0. Note that for $x_i > 0$, the term $f_{X_i}(x_i) - f_{X_i}(-x_i)$ is always greater than 0.

The only term that involves μ_1 is the conditional expectation

$$\mathrm{E}(Y|X_2 = x_2, X_3 = x_3, \ldots, X_n = x_n).$$

However, for $x_2, x_3, \ldots, x_n$ all greater than 0, it increases with μ_1. As the result, with all the other parameters $\mu_2, \mu_3, \ldots, \mu_n$ fixed, the integration for $\mathrm{E}(Y)$ also increases with μ_1. □

Lemma 3 *The elementary EXIT chart is defined as in (3.6), following the same notations, then $f_d(\mu)$ increases with the mean μ and decreases with the channel variance σ^2.*

Proof: Let the function $y = g(z, x_1, \ldots, x_{d-1})$ be defined as

$$y := 2\text{atanh}\left(\tanh\left(\frac{z}{2}\right)\prod_{i=1}^{d-1}\tanh\left(\frac{x_i}{2}\right)\right).$$

By checking this function g, we see all the properties assumed for g in Lemma 2 hold. Therefore, by Lemma 2, we intuitively see the behavior of $f_d(\mu)$ which is $\mathrm{E}(Y)$ with respect to μ. Based on the fact that all the X_i's have the same mean μ, instead of increasing the mean of one random variable, we increase the mean for each X_i step by step, thus, equivalently, μ is increased finally. Then we argue that in each step $f_d(\mu)$ increases, which thereby allows the elementary EXIT chart $f_d(\mu)$ to increase with respect to the parameter μ.

Similarly, since the mean of channel LLR is defined as $2/\sigma^2$, with the decreasing of σ^2, the mean of channel LLR increases, and hence $f_d(\mu)$ increases. □

We now turn to the proof of Theorem 1 using the above results.

Proof: The elementary EXIT chart is well defined in (3.6) with the corresponding parameter settings. Also, we will use the notations in the mean-LLR EXIT chart approach described in (3.7).

By exploiting the linear programming for code construction, our objective is to minimize $\sum_{d=1}^{D}\omega_d/d$ for any given α, while ensuring the condition $\alpha\sum_{d=1}^{D}\omega_d f_d(\mu_i) > \mu_i$ for all $\mu_i \in (0, \mu_0]$ $(i = 1, 2, \ldots, N)$. It is worth noting that the approach will put more point mass to high degree ω_d, with a limit at ω_D, if possible.

For a finite N and given α, we can certainly argue that we are able to find a channel with SNR^*, where the condition $f_D^*(\mu_i) \geq \mu_i/\alpha$ holds for all $\mu_i \in (0, \mu_0]$, if we denote the elementary EXIT chart for such channel by $f_d^*(\mu)$.

Then, for any channels with $\mathrm{SNR} > \mathrm{SNR}^*$, by Lemma 3, the EXIT chart $f_d(\mu_i) > f_d^*(\mu_i)$ for all $\mu_i \in (0, \mu_0]$ and $d = 1, \ldots, D$.

Hence, the inequality $\alpha f_D(\mu_i) > \mu_i$ also holds for all $\mu_i \in (0, \mu_0]$. The solution of the linear programming is $\omega_D = 1$, which is equivalent to $\Omega_D = 1$. □

3.5 Simulation study

For Raptor codes, word error rate as a function of rate R is sufficient to judge the performance. Provided that the transmission terminates at rate R, the probability that the transmitted codeword can not be decoded successfully is termed the *word error rate* (WER) at rate R. This criterion is commonly used to evaluate the code performance in communication scenario. However, the mean-LLR EXIT chart based approach, as an approximation of density evolution, guarantees the convergence of bit error probability, but does not take into account the word error probability. On the other hand, in Raptor coding scheme, the inner code, LT codes, can be treated to provide a much better channel for the outer LDPC code. Thereby, LT codes should be evaluated in terms of the notion of bits per channel use, equivalently, *bit error rate* (BER). For LT codes alone, the code is truncated as a fixed-rate code with respect to the reception overhead, where *reception overhead* is a metric of the excessive reception beyond the capacity of channel, or the normalized excessively received codeword bits. BER is then examined at a fixed-rate, or equivalently, a fixed overhead.

Based on the obtained degree distributions by the heuristic-α approach and the optimizing-α approach for various channels, which are shown in Table 3.1 and 3.2, we performed Monte Carlo simulations to test the performance. Note that we have chosen the α to be 1.33% higher than the lower bound specified in (3.10) for various channels respectively in the heuristic-α approach. For each generated degree distribution, two different tests are involved. On one hand, we perform the fixed-rate simulations for the truncated LT codes picked arbitrarily from the ensemble. On the other hand, another test typically for the Raptor codes is performed, where the code is randomly generated, and the decoder attempts to decode the codeword at different time instants until the codeword is fully decoded. A new reduced-complexity decoding algorithm is applied [29]. Clearly, the time instant, or equivalently, the number of bits used when the decoder decodes successfully is a random variable, denoted by N, and we term the information bit number k divided by the expected value of N as *realized rate*.

In LT codes simulations, we choose the length of the information bits $k = 10000$ with the performance evaluated at reception overheads of 0.05, 0.075, 0.1, and 0.125

SNR(dB)	-6.9897	-3.9896	-1.9895	0.2022	5.0000
Ω_1	0.003393	0.003545	0.003238	0.023964	0.179434
Ω_2	0.400887	0.421847	0.457324	0.404842	
Ω_3	0.158528	0.185615	0.168358	0.277256	0.553762
Ω_4					0.177369
Ω_5	0.233464	0.192157		0.071497	
Ω_6			0.068681	0.030744	
Ω_7			0.159760		
Ω_{10}		0.044431			
Ω_{11}				0.090701	
Ω_{12}	0.069386	0.013123			
Ω_{15}	0.028472	0.051625			
Ω_{18}				0.027676	
Ω_{23}		0.009442			
Ω_{39}			0.081192		
Ω_{43}			0.061446		
Ω_{51}	0.038970				0.022724
Ω_{52}					0.066711
Ω_{58}				0.002484	
Ω_{59}	0.066899				
Ω_{63}		0.038225			
Ω_{65}				0.070837	
Ω_{78}		0.039992			

Table 3.1: Degree distributions of Raptor codes optimized by heuristic-α approach.

SNR(dB)	-6.9897	-3.9896	-1.9895	0.2022	5.0000
Ω_1	0.003623	0.004772	0.005350	0.028586	0.161179
Ω_2	0.419636	0.425872	0.444633	0.387536	
Ω_3	0.170084	0.213558	0.257273	0.343613	0.756820
Ω_5	0.231395	0.128390			
Ω_6		0.070213	0.028870	0.048782	
Ω_7			0.118087		
Ω_8	0.025704	0.004210			
Ω_9			0.061844	0.098832	
Ω_{10}				0.003167	
Ω_{12}	0.067674				
Ω_{13}		0.088615			
Ω_{18}				0.040134	
Ω_{24}					0.022446
Ω_{25}	0.002972				
Ω_{26}			0.017757		0.042065
Ω_{30}	0.053731				
Ω_{31}			0.043975		
Ω_{34}		0.000464		0.005308	
Ω_{36}		0.038003			
Ω_{37}		0.002567			
Ω_{39}				0.024227	
Ω_{200}	0.025180	0.023338	0.022312	0.019814	0.017484

Table 3.2: Degree distributions of Raptor codes optimized by optimizing-α approach.

respectively. In Raptor codes simulations, since the LDPC pre-code is less critical on the performance compared with the LT component, a rate 0.95 left-regular 4 right-Poisson LDPC pre-code is used as the setting in [6]. We test over 200 codewords and k is set to be 9500.

Simulation results of the LT codes for channels with the increasing SNRs are presented in Figures (3.5) to (3.9) respectively, where instead of reception overhead, we equivalently generate the plots according to inverse rate $1/R$. Next, Figures (3.10) to (3.14) present the simulation results of the Raptor codes for channels with the order of the increasing SNR. Then, we compare both the realized rates of codes constructed by the optimizing-α and the heuristic-α approach with the capacity of BIAWGN channels in Figure 3.15. Finally, Figure 3.16 shows the Ω_2's generated by both approaches over various channels, compared with the theoretical lower bound of Ω_2 defined in (3.5).

From Figures 3.5 to 3.15, it can be shown that the optimizing-α approach uniformly outperforms the heuristic-α approach across all channel SNRs. This is observed both for Raptor codes and for LT codes.

For purpose of comparison, we follow the same criterion to choose α in the heuristic-α approach. The performance advantage of the optimizing-α approach over the heuristic-α approach normalized by channel maintains the consistence from low SNR channels to high SNR channels.

The performance gap between the optimizing-α approach and the heuristic-α approach for LT codes exhibits a similar trend to that for Raptor codes, suggesting the effect of LT inner code and LDPC outer code may be roughly decoupled, namely, LT codes serve to provide a better channel.

Compared with capacity, Figure 3.15 shows that at high SNR region there is still a quite visible gap for Raptor codes constructed by the optimizing-α approach. Furthermore, the gap becomes larger at higher SNR.

One explanation of this outcome is that at high SNR region, even the optimizing-α could not produce codes with Ω_2 above the necessary bound for achieving good performance, which we refer to Figure 3.16.

This fact may be intuitively interpreted as follows. High SNR induces low value of α either by the heuristic-α approach or by the optimizing-α approach. Due to the central

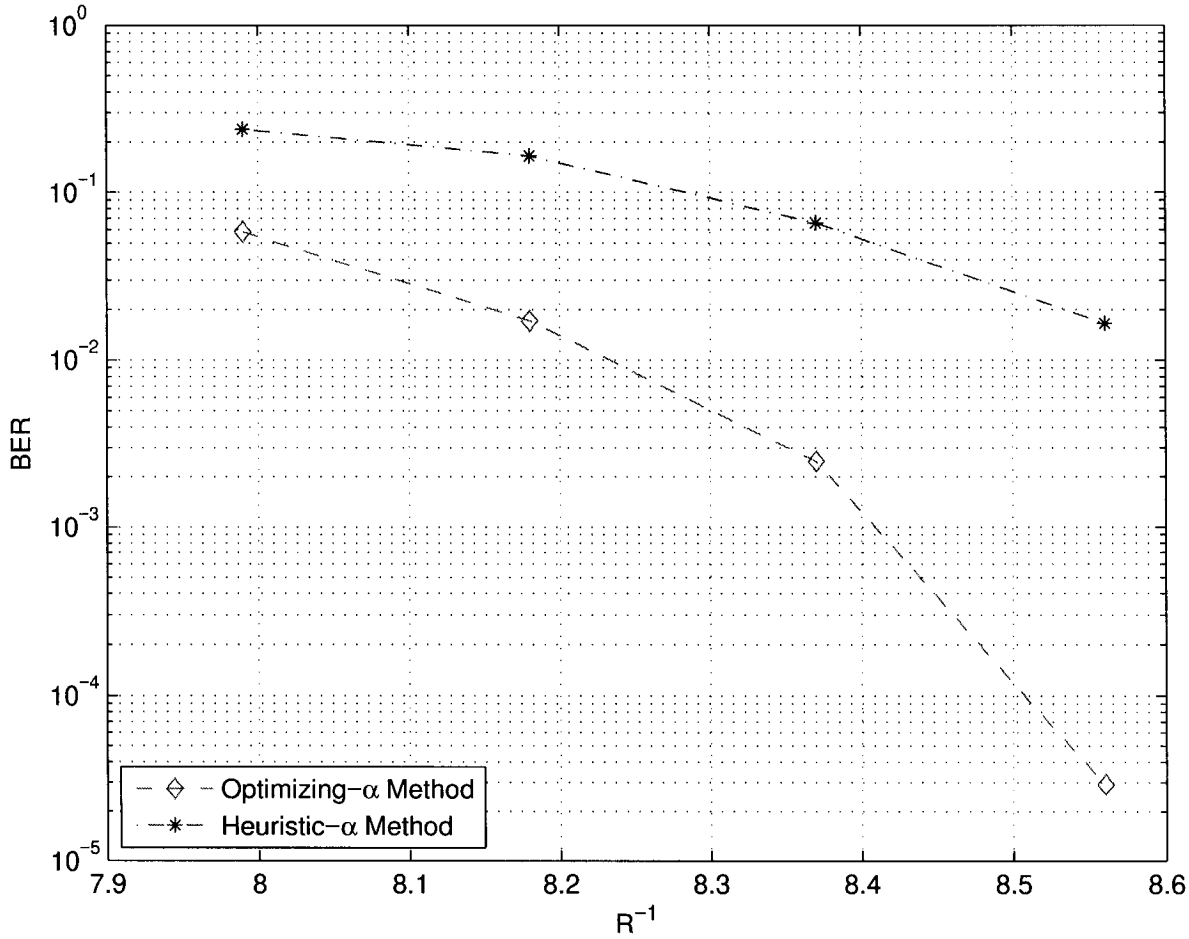

Figure 3.5: Bit error rate of the LT code as a function of $1/R$ for the two approaches over -6.9897 dB channel.

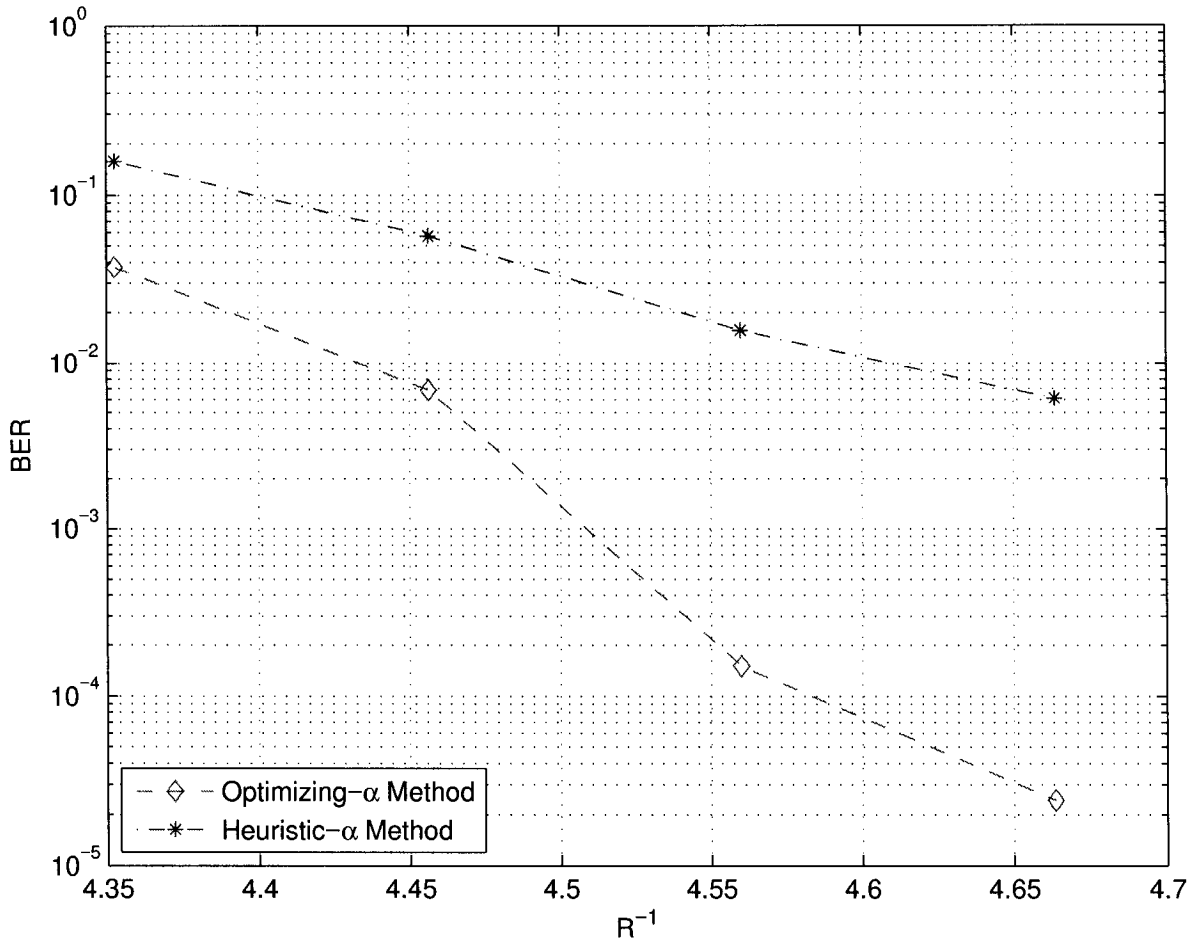

Figure 3.6: Bit error rate of the LT code as a function of $1/R$ for the two approaches over -3.9896 dB channel.

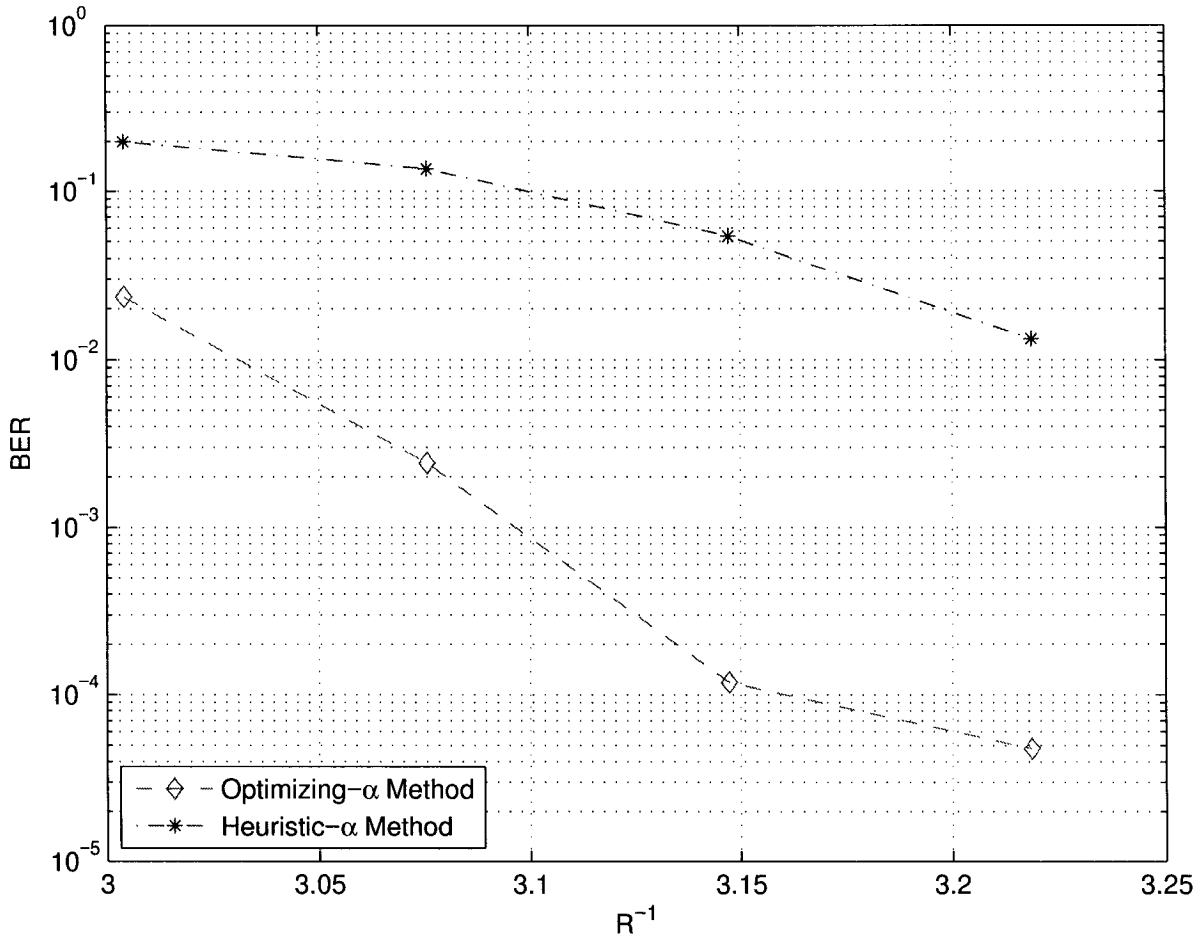

Figure 3.7: Bit error rate of the LT code as a function of $1/R$ for the two approaches over -1.9895 dB channel.

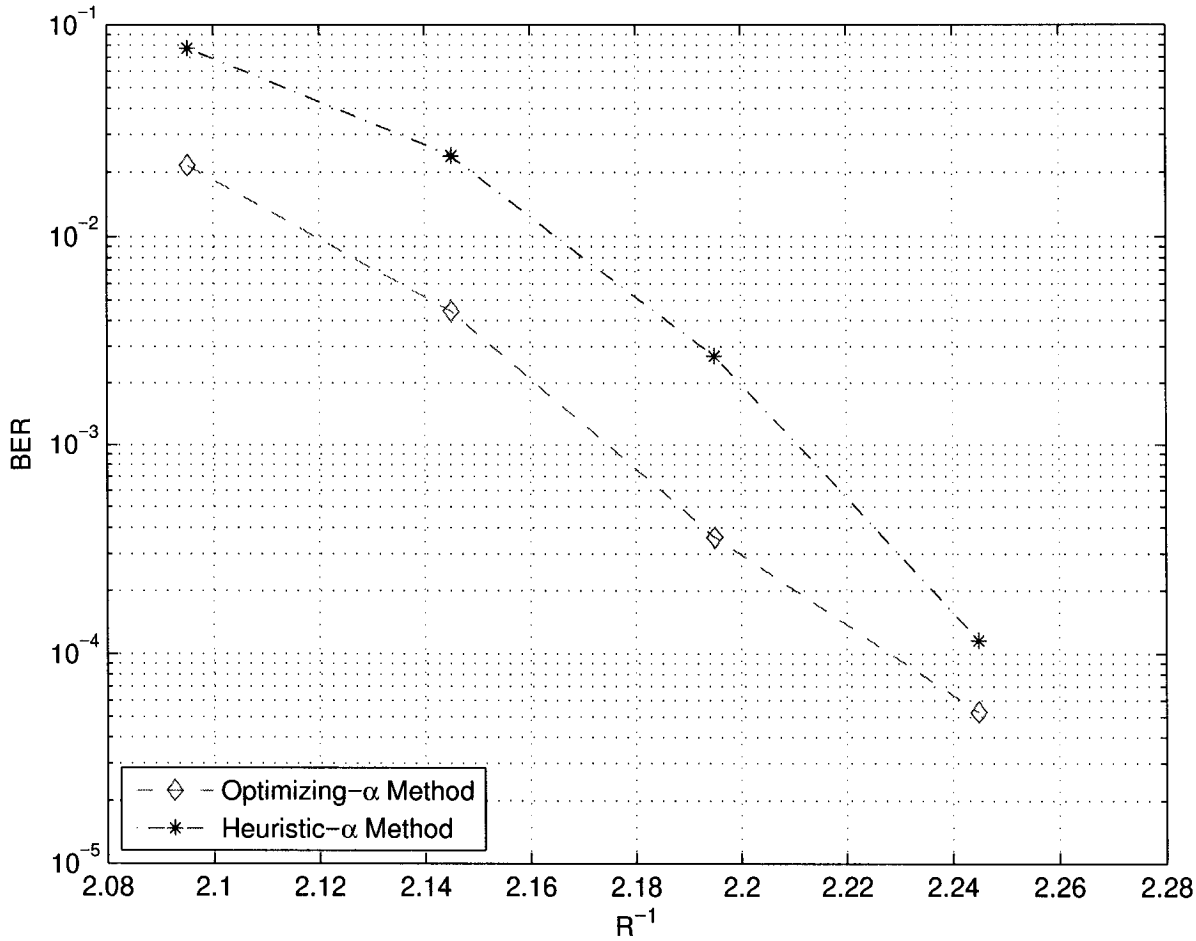

Figure 3.8: Bit error rate of the LT code as a function of $1/R$ for the two approaches over 0.2022 dB channel.

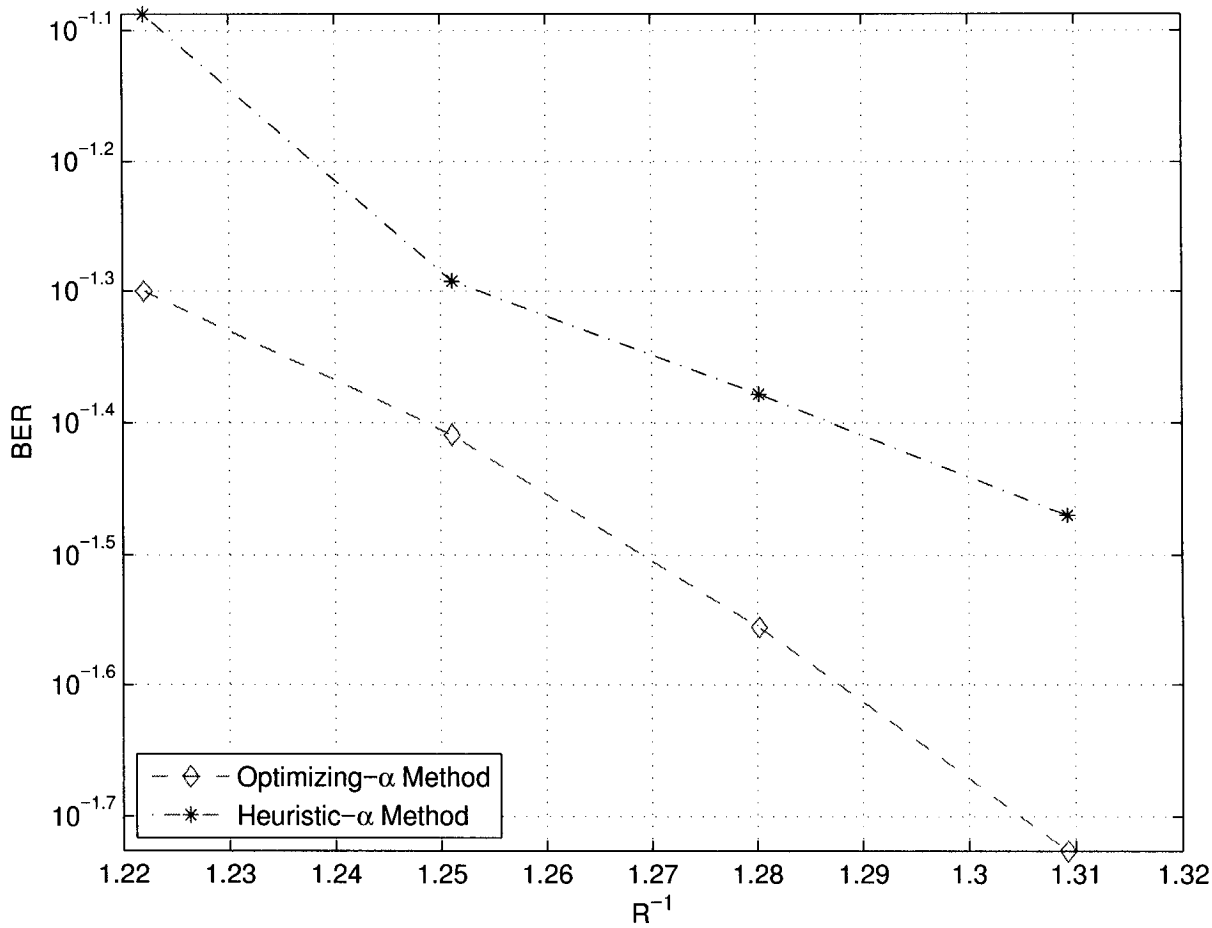

Figure 3.9: Bit error rate of the LT code as a function of $1/R$ for the two approaches over 5 dB channel.

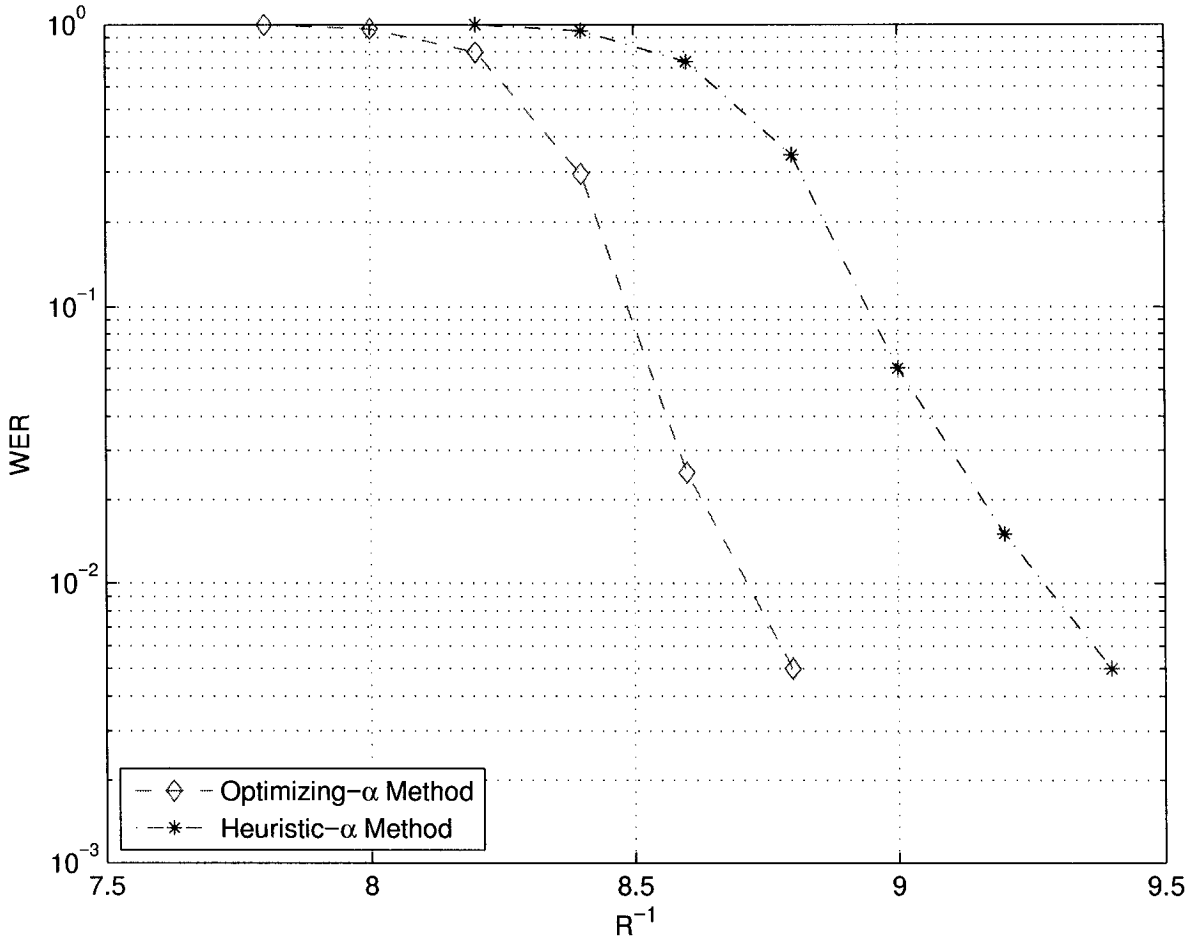

Figure 3.10: Word error rate of the Raptor code as a function of $1/R$ for the two approaches over -6.9897 dB channel.

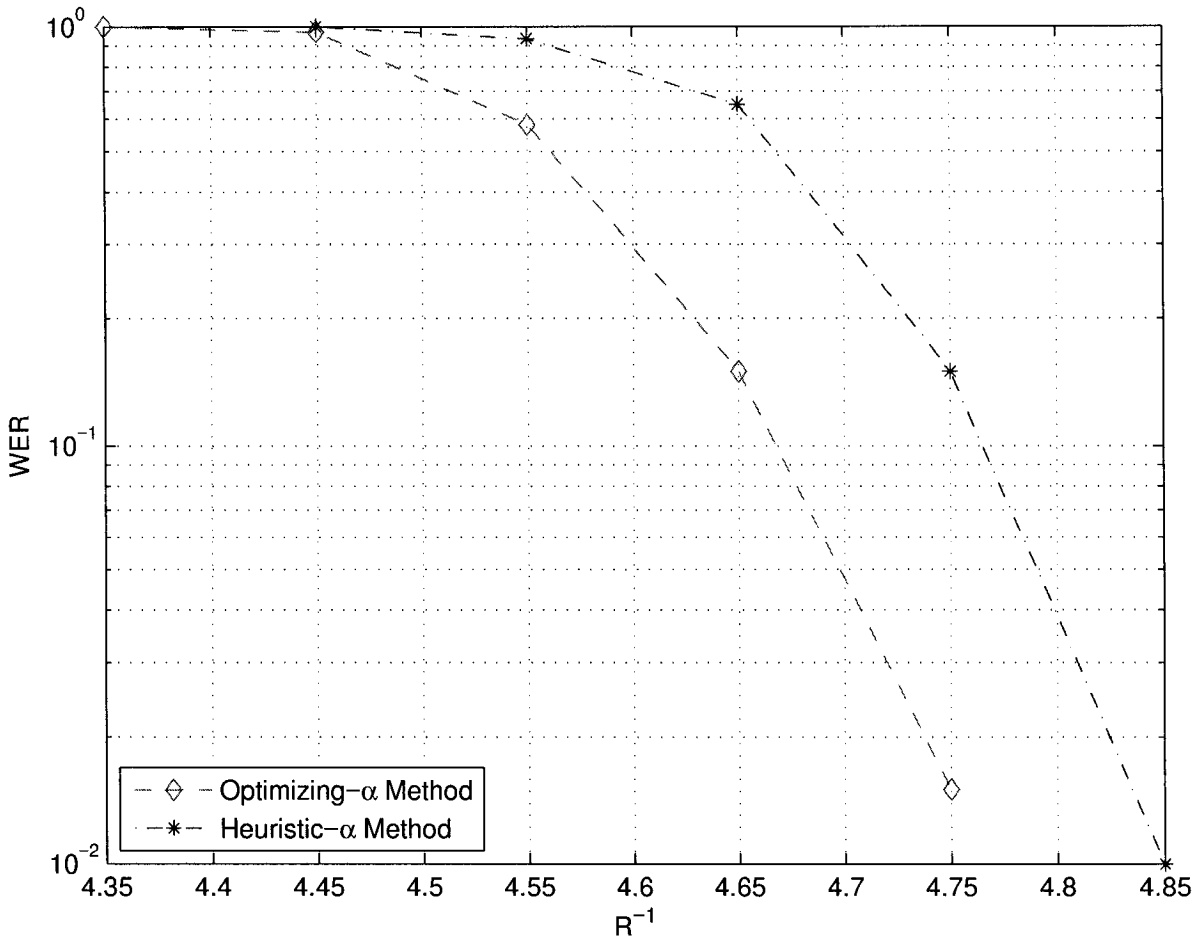

Figure 3.11: Word error rate of the Raptor code as a function of $1/R$ for the two approaches over -3.9896 dB channel.

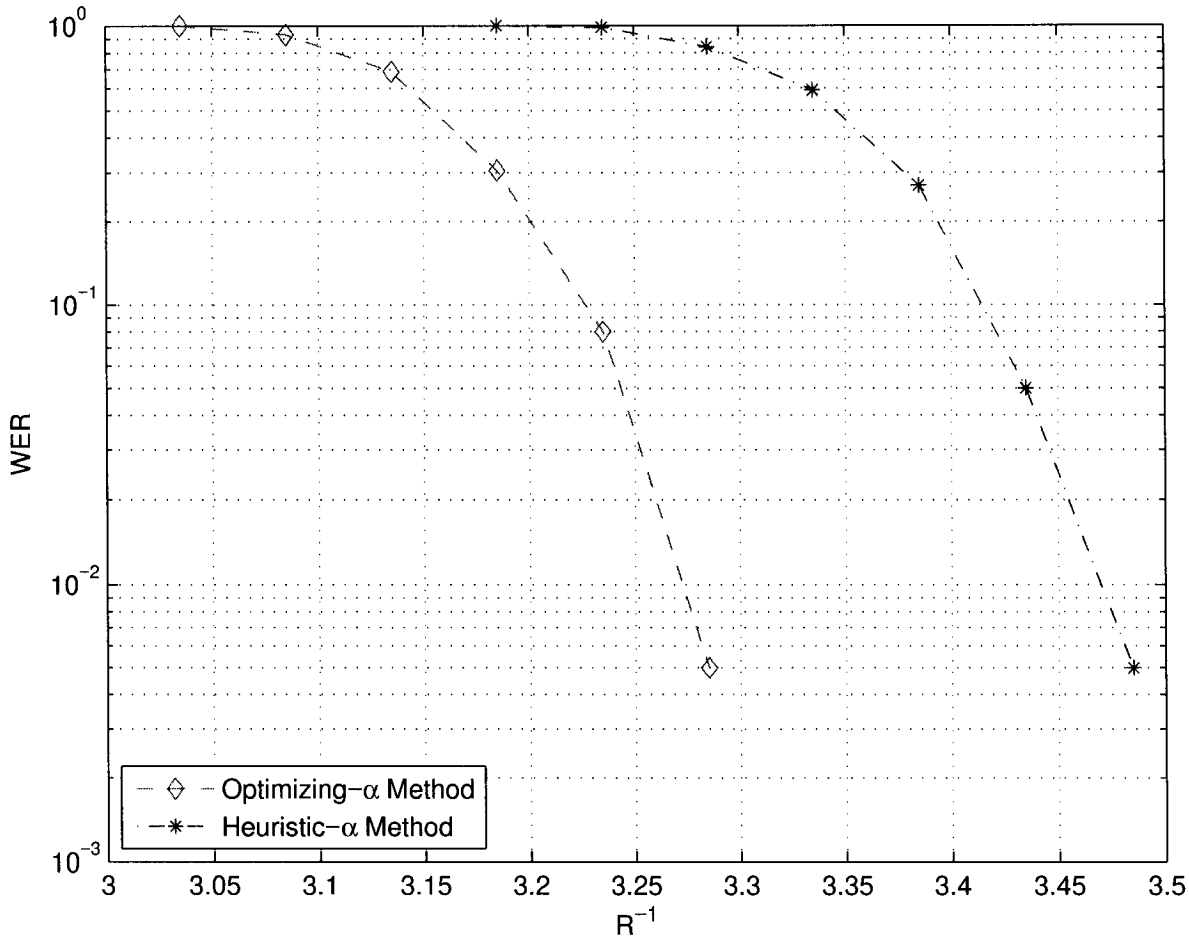

Figure 3.12: Word error rate of the Raptor code as a function of $1/R$ for the two approaches over -1.9895 dB channel.

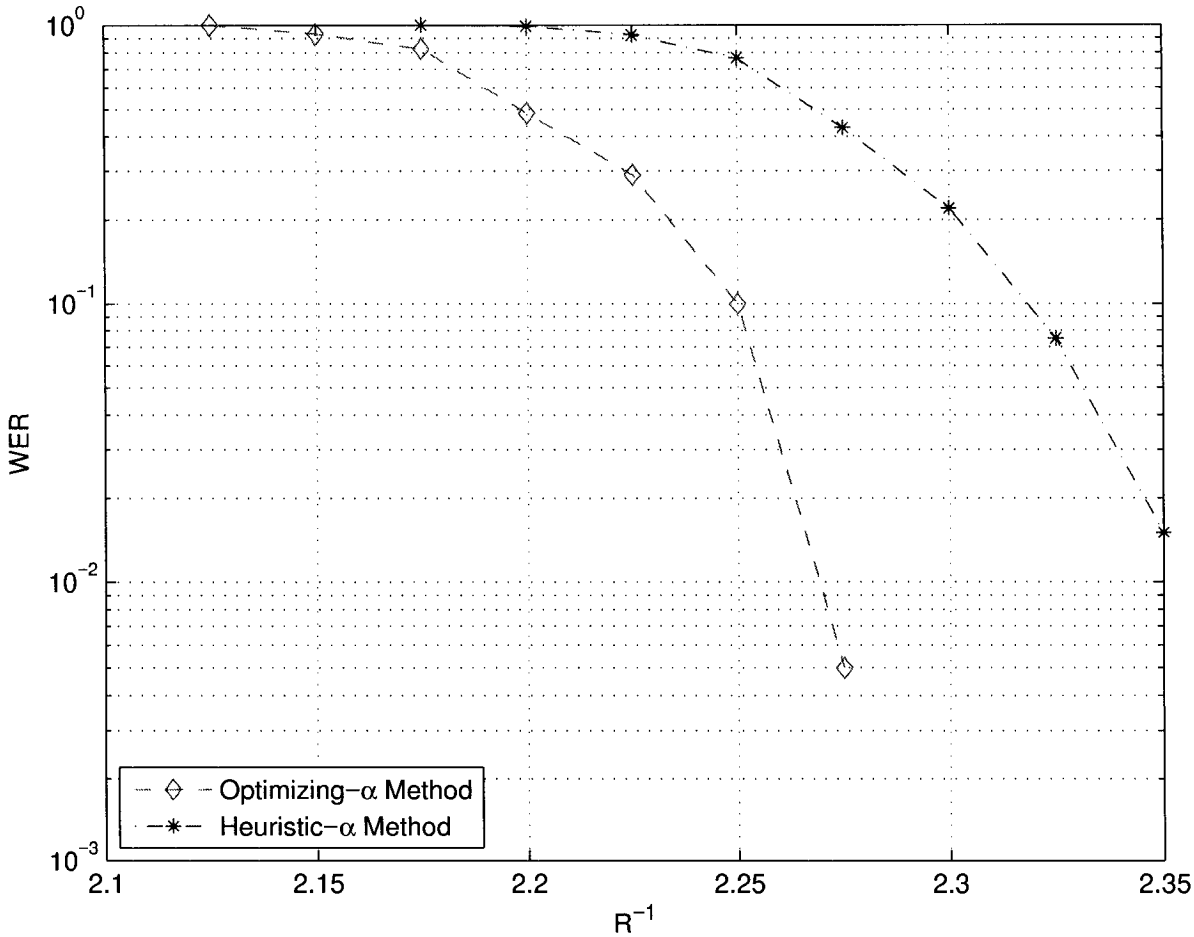

Figure 3.13: Word error rate of the Raptor code as a function of $1/R$ for the two approaches over 0.2022 dB channel.

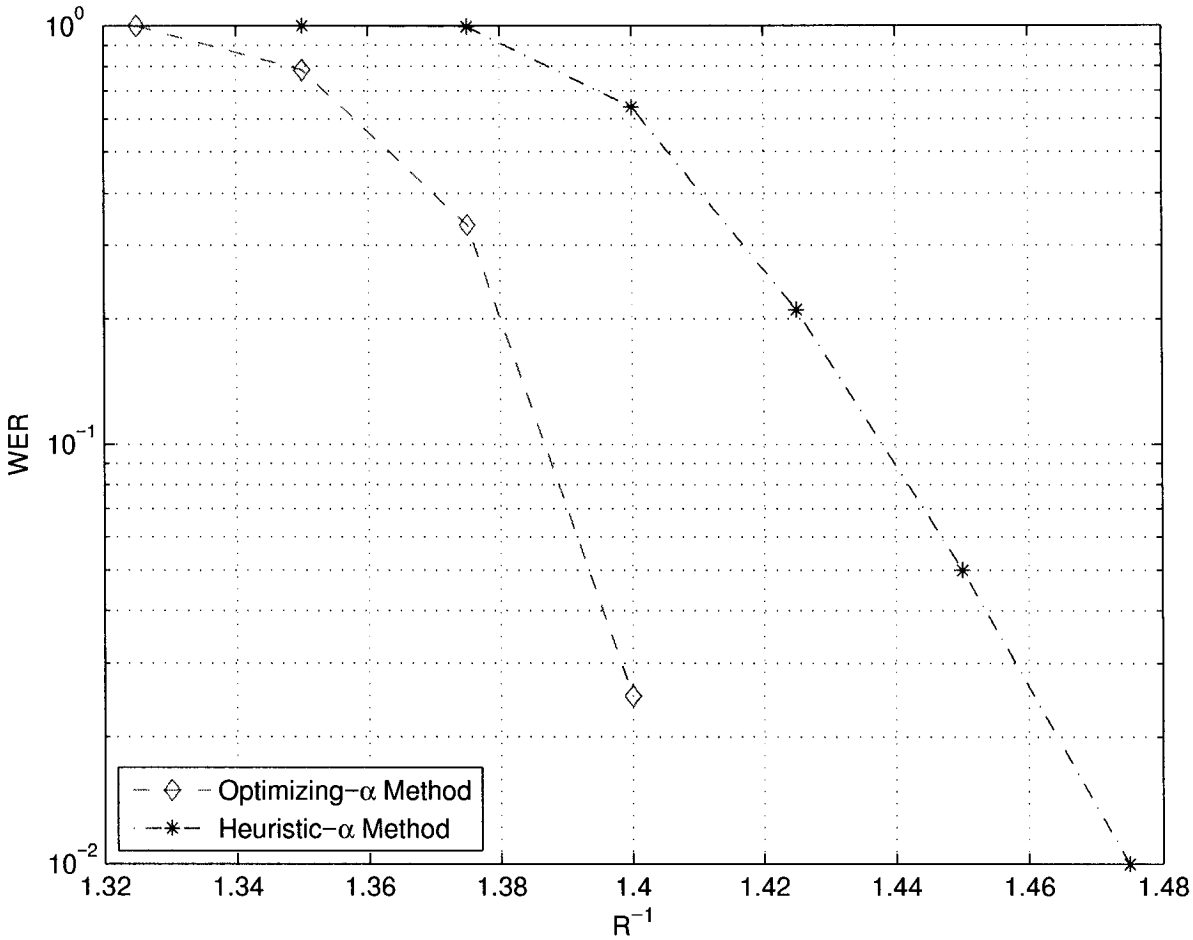

Figure 3.14: Word error rate of the Raptor code as a function of $1/R$ for the two approaches over 5 dB channel.

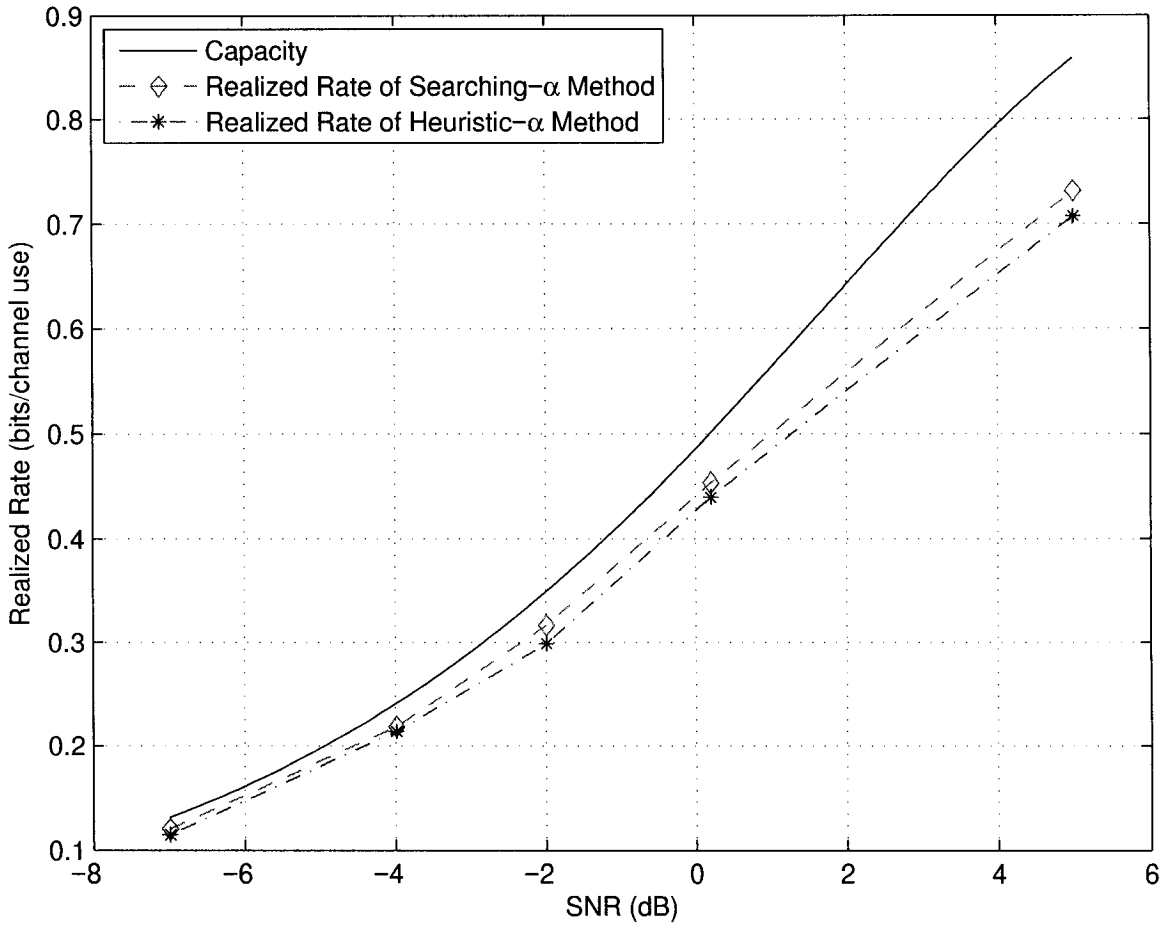

Figure 3.15: Realized rate vs. channel SNR.

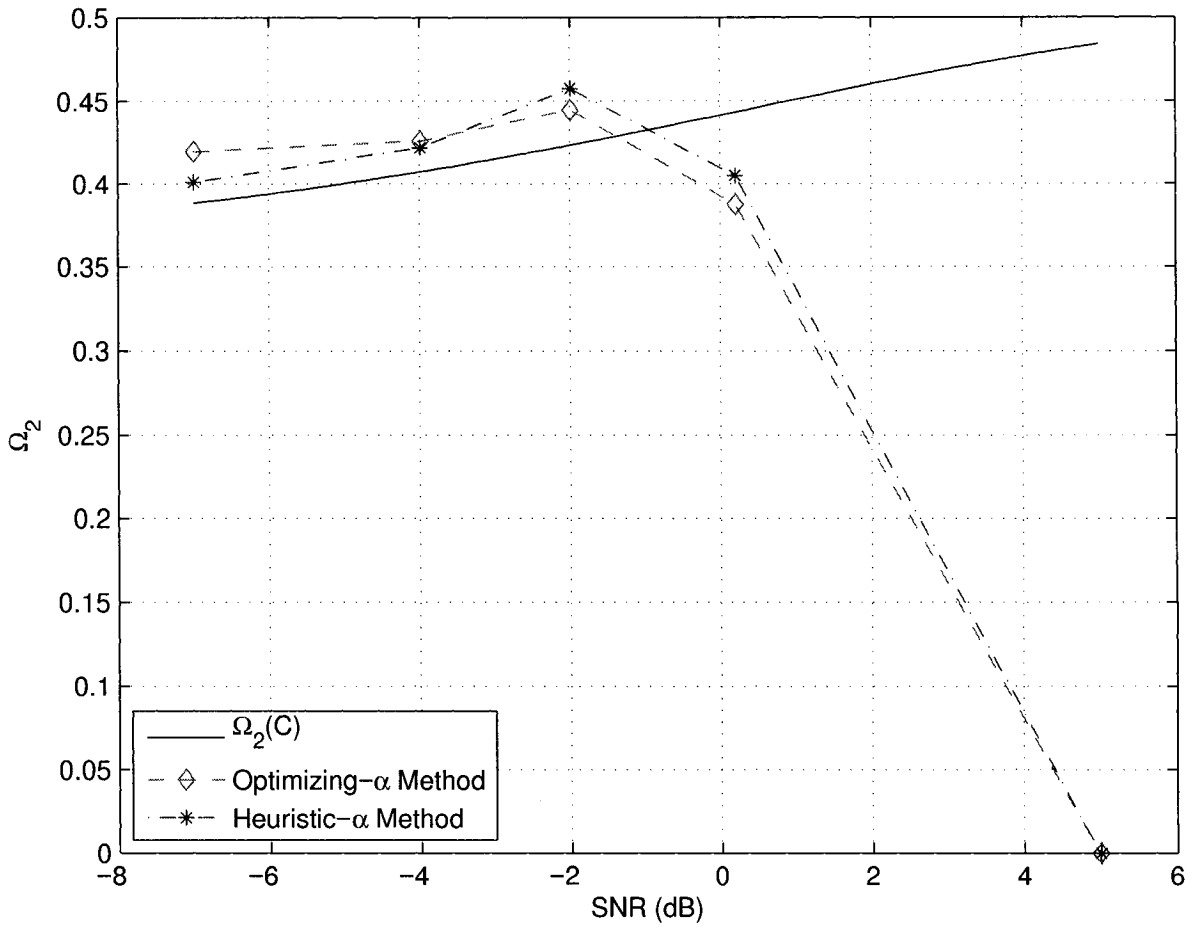

Figure 3.16: Ω_2 vs. channel SNR.

limit theorem, this effectively makes the assumption of Gaussian density at the output of input bits inaccurate. As a consequence, the linear program in (3.7) based on mean-LLR EXIT chart no longer applies to the case.

Chapter 4

Conclusion and Suggestions

Without channel knowledge, fountain codes universally achieve the capacity of erasure channels, and have been tested over a large range of noisy channels. The modern rateless coding paradigm motivates us to study the design of Raptor codes over Gaussian channels.

Previous results [8] on this subject have shown

- that there exists no universal output degree distribution of Raptor codes that achieve the capacities of all BIAWGNs, where the fraction of degree 2 output symbols is required to converge to a lower bound determined by the channel for capacity-achieving codes; and
- linear programming based on mean-LLR EXIT chart can be applied to construct Raptor codes for a given Gaussian channel, and the input degree distribution is set *a priori*.

Following up on these results, this thesis contributes to the subject matter in the following aspects.

- We prove that there exists no universal choice of input degree distribution for constructing capacity-achieving Raptor codes using mean-LLR EXIT chart approach, since the necessary condition may be violated over a range of channels for a fixed input degree distribution.

- We propose a method of jointly finding input degree distribution and output degree distribution, extending the linear programming approach in [8]. This method provides a more precise way of choosing the suitable input degree distribution.
- We show by simulations that our method uniformly outperforms an existing heuristic-α approach [11] over all simulated Gaussian channels.
- Our simulation results also suggest that at high SNR, there still exists a gap to channel capacity for Raptor codes constructed using mean-LLR EXIT chart based approaches. This is due to the fact that the codes constructed by such approaches do not meet the necessary condition of achieving good performance.

At the point, we shall turn to consider far beyond this. Some future work exists in the following areas.

- On fountain code construction, problems exist in how to design good Raptor codes over high SNR Gaussian channels.
- In addition to the code design for a certain channel, the construction of fountain codes with the evolving degree distribution suitable for an ordered set of Gaussian channels needs to be considered.

  By doing so, the evolving fountain codes are capable of achieving the capacities for those set of channels respectively. More generally, any truncated high rate fountain codes are capacity-achieving. Terminologically, in rateless coding context, such codes are named Perfect Rate-Compatible codes and have been demonstrated to be existing over Gaussian channels theoretically [30].
- Attention has been given to the much wider rateless code construction scenario for noisy channels. Based on the existing Irregular Repeat Accumulate (IRA) codes, a new approach of constructing rateless codes, namely the check splitting method, was proposed in [31]. This approach focuses on preventing the cycles while keeping the parity-check valid in the codes on graph, and also provides some implications on the properties of graph structure for capacity-achieving rateless codes.

- The work of [32–34] suggests that LDPC codes can be adapted to approach the Slepian-Wolf bound in source coding as they approach the Shannon limit in channel coding. Based on the joint source coding scheme, the evolving rateless coding scheme is considered as a joint-source channel coding problem. Theoretical analysis is needed for this approach.

- Other work such as [35] also gives the suggestion on generating the infinite codeword stream through a way of parallel concatenation of Low Density Generator Matrix (LDGM) codes, and the realization is to be expected.

Appendix

I Proof of non-convexity

In this section, we show $f(\alpha, \{\omega_d\}) = \alpha \sum_d \omega_d/d$ is not a convex function by a counterexample.

We choose two points $(\alpha_1, \{\omega_d^1\})$ and $(\alpha_2, \{\omega_d^2\})$ subject to the constraints defined in (3.11). By Corollary 1, we can certainly assume that with α_1 larger than some bound, $\sum_{d=1}^{D} \omega_d^1/d = 1/D$ holds, where D is the given maximum degree of output bits. With α_2 smaller than that bound, it turns out that $\sum_d \omega_d^1/d < \sum_d \omega_d^2/d$. Clearly, $\alpha_1 > \alpha_2$.

We pick $\theta = 0.5$. Let α' denote $\theta\alpha_1 + (1-\theta)\alpha_2$, and let ω_d' denote $\theta\omega_d^1 + (1-\theta)\omega_d^2$. Therefore,

$$f(\alpha', \{\omega_d'\}) = \frac{1}{4}(\alpha_1 + \alpha_2) \sum_{d=1}^{D} \frac{\omega_d^1 + \omega_d^2}{d}$$

and

$$\theta f(\alpha_1, \{\omega_d^1\}) + (1-\theta) f(\alpha_2, \{\omega_d^2\}) = \frac{1}{2}\left(\alpha_1 \sum_{d=1}^{D} \frac{\omega_d^1}{d} + \alpha_2 \sum_{d=1}^{D} \frac{\omega_d^2}{d}\right).$$

Let $\tau = f(\alpha', \{\omega'_d\}) - (\theta f(\alpha_1, \{\omega^1_d\}) + (1-\theta) f(\alpha_2, \{\omega^2_d\}))$, thus we have

$$\begin{aligned}
\tau &= \frac{1}{4}(\alpha_1 + \alpha_2) \sum_{d=1}^{D} \frac{\omega^1_d + \omega^2_d}{d} - \frac{1}{2}\left(\alpha_1 \sum_{d=1}^{D} \frac{\omega^1_d}{d} + \alpha_2 \sum_{d=1}^{D} \frac{\omega^2_d}{d}\right) \\
&= \frac{\alpha_2 - \alpha_1}{4} \sum_{d=1}^{D} \frac{\omega^1_d}{d} + \frac{\alpha_1 - \alpha_2}{4} \sum_{d=1}^{D} \frac{\omega^2_d}{d} \\
&= \frac{\alpha_1 - \alpha_2}{4}\left(\sum_{d=1}^{D} \frac{\omega^2_d}{d} - \sum_{d=1}^{D} \frac{\omega^1_d}{d}\right).
\end{aligned}$$

Obviously, we obtain $\tau > 0$, which disproves the convexity of function f.

II Notation

$\mathrm{B}(\mathcal{C})$	Expectation of $\tanh(Z/2)$ as a parameter of $\mathcal{C}$
$\mathtt{C}$	Pre-code of Raptor codes
$\mathrm{Cap}(\mathcal{C})$	Capacity of channel $\mathcal{C}$
D	Maximum output bit degree of Raptor codes
$\mathcal{D}$	Degree distribution of LT codes
$F(x_1, \ldots, x_n)$	Global function
$F_i(x_i)$	Marginal function
$f_d(\mu)$	Elementary EXIT chart
$f_i(X_i)$	Local function
$\mathcal{G}(\mu)$	Symmetric Gaussian density with mean μ
$I(x)$	Input node degree distribution
I_d	Probability of a degree d input node being chosen
$\mathcal{N}(v)$	Set of neighbors of node v
$P_{e,in}$	Decoding error probability of previous iteration
$P_{e,out}$	Decoding error probability of current iteration
Z	Channel LLR
α	Average degree of input symbols
β	Average degree of output symbols
$\delta(x)$	Indicator function
$\iota(x)$	Input edge degree distribution
ι_d	Probability of an edge connected to a degree d input node
μ_0	Maximum mean of symmetric Gaussian from input to output bits
μ_{chk}	Expectation of LLR from output to input bits
$\mu_{f\to u}$	Message passed from function node f to variable node u
μ_{in}	Mean of symmetric Gaussian from previous iteration
μ_{LLR}	Mean of channel LLR
μ_{out}	Mean of symmetric Gaussian from current iteration
$\mu_{u\to f}$	Message passed from variable node u to function node f
μ_u	Summary message of variable node u

$\{\lambda_i\}$	LDPC variable degree distribution
$\{\rho_j\}$	LDPC check degree distribution
$\Pi(\mathcal{C})$	$\mathrm{Cap}(\mathcal{C})/\mathrm{B}(\mathcal{C})$ as a parameter of $\mathcal{C}$
$\Omega(x)$	Output node degree distribution
$\Omega_2(\mathcal{C})$	Lower bound of Ω_2 for capacity-achieving Raptor codes
Ω_d	Probability of a degree d output node being chosen
$\omega(x)$	Output edge degree distribution
ω_d	Probability of an edge connected to a degree d output node

References

[1] M. Luby, "LT codes," in *43rd Annual IEEE Symp. on the Found. of Comp. Sci.*, 2002, pp. 271–280.

[2] A. Shokrollahi, "Raptor codes," in *Proc. IEEE Int. Symp. on Inform. Theory*, 2004, p. 36.

[3] A. Lapidoth and P. Narayan, "Reliable communication under channel uncertainty," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2148–2175, Oct. 1998.

[4] M. Medard, "The effect upon channel capacity in wireless communications of perfect and imperfect knowledge of the channel," *IEEE Trans. Inform. Theory*, vol. 46, no. 3, pp. 933–946, May 2000.

[5] O. Etesami, M. Molkaraie, and A. Shokrollahi, "Raptor codes on symmetric channels," in *Proc. IEEE Int. Symp. on Inform. Theory*, 2004, p. 38.

[6] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in *Proc. IEEE Int. Symp. on Inform. Theory*, 2004, p. 37.

[7] J. Castura and Y. Mao, "Rateless coding over fading channels," *IEEE Comm. Lett.*, vol. 10, no. 1, pp. 46–48, Jan. 2006.

[8] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Trans. Inform. Theory*, vol. 52, no. 5, pp. 2033–2051, May 2006.

[9] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under messages-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[10] M. Ardakani and F. Kschischang, "A more accurate one-dimensional analysis and design of irregular ldpc codes," *IEEE Trans. Commun.*, vol. 52, no. 12, pp. 2106–2114, Dec. 2004.

[11] A. Shokrollahi, 2006, private communication.

[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* John Wiley and Sons, 1991, ch. 8, pp. 184, 198.

[13] S. C. Draper, B. J. Frey, and F. R. Kschischang, "Efficient variable length channel coding for unknown DMCs," in *Proc. IEEE Int. Symp. on Inform. Theory*, 2004, p. 379.

[14] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[15] N. Wiberg, H.-A. Loeliger, and R. Kotter, "Codes and iterative decoding on general graphs," *Eur. Trans. Telecomm.*, vol. 6, pp. 513–525, Sep./Oct. 1995.

[16] G. D. Forney, "On iterative decoding and the two-way algorithm," *Int. Symp. on Turbo Codes and Related Topics, Brest, France*, pp. 12–25, Sep. 1997.

[17] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[18] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inform. Theory*, vol. 46, pp. 325–343, Mar. 2000.

[19] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[20] Y. Mao and A. H. Banihashemi, "Decoding low-density parity-check codes with probabilistic scheduling," *IEEE Comm. Lett.*, vol. 5, no. 10, pp. 414–416, Oct. 2001.

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

[22] H. Xiao and A. H. Banihashemi, "Graph-based message-passing schedules for decoding LDPC codes," *IEEE Trans. Comm.*, vol. 52, no. 12, pp. 2098–2105, Dec. 2004.

[23] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, pp. 2173–2200, 2001.

[24] J. G. Proakis, *Digital Communications, Fourth Edition.* MCGraw-Hill College, Aug. 2000, ch. 5, p. 242.

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

[26] D. J. C. Mackay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *IEE Electron. Lett.*, vol. 33, no. 6, pp. 457–458, Mar. 1997.

[27] S. Y. Chung, G. D. Forney, T. J. Richardson, and R. L. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Letter*, vol. 5, pp. 58–60, Feb. 2001.

[28] R. G. Gallager, *Low Density Parity Check Codes.* Cambridge, MA: MIT Press, 1963.

[29] K. Hu, J. Castura, and Y. Mao, "Reduced-complexity decoding of raptor codes over fading channels," in *IEEE GLOBECOM*, 2006.

[30] U. Erez, M. D. Trott, and G. W. Wornell, "Rateless coding and perfect rate-compatible codes for Gaussian channels," in *Proc. IEEE Int. Symp. on Inform. Theory*, 2006.

[31] M. Good and F. R. Kschischang, "Incremental redundancy via check splitting," in *23rd Biennial Symp. on Comm.*, 2006, pp. 55–58.

[32] J. Garcia-Frias and W. Zhong, "LDPC codes for compression of multi-terminal sources with hidden Markov correlation," *IEEE Comm. Lett.*, vol. 7, no. 3, pp. 115–117, March 2003.

[33] F. Cabarcas and J. Garcia-Frias, "Approaching the Slepian-Wolf boundary using practical channel codes," in *Proc. IEEE Int. Symp. on Inform. Theory*, 2004.

[34] A. W. Eckford and W. Yu, "Density evolution for the simultaneous decoding of LDPC-based Slepian-Wolf source codes," in *Proc. IEEE Int. Symp. on Inform. Theory*, 2005.

[35] W. Zhong, H. Chai, and J. Garcia-Frias, "Approaching the Shannon limit through parallel concatenation of regular LDGM codes," in *Proc. IEEE Int. Symp. on Inform. Theory*, 2005, p. 5.