A Comparison of Rateless Codes for Wireless Communication Systems

by

Haoming Li

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

Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems & Computer Engineering
Carleton University
Ottawa, Ontario, Canada

August 2007

Your file Votre référence
ISBN: 978-0-494-33656-4
Our file Notre référence
ISBN: 978-0-494-33656-4

Abstract

Rateless codes have drawn much attention in recent years as they can approach channel capacity without requiring channel information at the transmitter. In this thesis, we compared the performance of three rateless codes, namely, Raptor codes, rate-compatible irregular repeat-accumulate (RC-IRA) codes, and rate-compatible quasi-cyclic LDPC (RC/QC-LDPC) codes proposed in 3GPP2 and 802.20 standards. The comparison is focused on short and moderate message word lengths under BPSK and higher-order modulations. Both AWGN and frequency-nonselective Rayleigh fading channels are investigated. Our results suggest that RC-IRA codes and RC/QC-LDPC codes are better than Raptor codes in terms of performance. The advantage of RC-IRA codes over RC/QC-LDPC codes is the ability to achieve rate 1 at high SNR's. At low SNR's, however, RC/QC-LDPC codes provides slightly higher throughput than RC-IRA codes, either in AWGN or frequency-nonselective Rayleigh fading channels.

Acknowledgments

I am deeply grateful to my advisor, Professor Ian D. Marsland, for his guidance and insights on digital communication. As a mentor and a friend, he gave me invaluable advices not only in my study direction but also in many details. He kept encouraging me to go further in my study. Without all these helps, this work would never be completed. I would also like to thank all of my course instructors during my graduate study, for their knowledge, wisdoms and inspiration. I am indebted to Moshe Good of University of Toronto for providing a copy of his Master's thesis and clarifying many details of the check splitting method. I wish to thank the anonymous reviewers of Information Theory Workshop (ITW) 2007 for their suggestions, which greatly improved this work.

I would like to thank the Faculty of Graduate Studies and Research, and the Department of Systems and Computer Engineering, for the financial support during my graduate study.

My friends Xiaosong Lv and Jie Liu helped me in many ways during my first year study. In my thesis preparation, Yaobing Wen and Chaowen Gu provided many supports on the LaTeX script writing. Special thanks to Chenjuan Zhou for her helps on the simulations and constant supports and encouragements during my graduate study.

Finally, I would like to thank my parents for bringing me to this world, continually encouraging me to look for the truth, and supporting my choice to study in Carleton Univerty.

Contents

List of Tables

List of Figures

List of Abbreviations

3GPP2	3rd Generation Partnership Project 2	2
ACE	approximate cycle EMD	20
ACM	adaptive coding/modulation	2
ADS	adaptive decoding start	45
ARR	average realized rate	45
AWGN	additive Gaussian white noise	3
BER	bit error rate	7
BICM	bit-interleaved coded modulation	76
BICM-ID	iterative decoding in BICM	76
BP	belief propagation	4
BPCM-ID	belief propagation coded modulation iterative decoding	78
BPSK	binary phase-shift keying	3
BRGL	binary reflected Gray labeling	79
CARR	conditional ARR	104
CRC	cyclic redundancy check	16
CS	check splitting	31
CSI	channel state information	95
DFR	decoding failure rate	105
E^2RC	efficiently-encodable rate-compatible	31

EA	early-abort	101
eIRA	extended irregular repeat-accumulate	31
EMD	extrinsic message degree	20
FEC	forward error correction	36
HARQ	hybrid automatic repeat-request	2
IDM	iterative de-mapping	76
IR	incremental redundancy	
k-SR	k-step self-recoverable node	30
LDPC	low-density parity-check	1
LLR	log-likelihood-ratio	12
LT	Luby Transform	24
MAP	maximum *a posteriori* probability	10
MARR	modified ARR	105
MLC	multi-level codes	75
ML	maximum likelihood	10
MLSD	maximum-likelihood sequence decoder	10
MSD	multi-stage decoding	75
MSP	modified set-partitioning	81
NP	non-deterministic polynomial-time	10
NSRR	non-strictly-right-regular	43
PAM	pulse-amplitude modulation	64
PEG	progressive-edge-growth	19
QAM	quadrature amplitude modulation	3
QC	quasi-cyclic	21
RBCM	reliability-based coded modulation	85

List of Symbols

Symbol	Explanation	page where symbol is defined
α_n	the arrival angle of the n^{th} ray in Jakes method	100
$\Lambda(x)$	variable degree distribution from the node perspective	7
$\lambda(x)$	variable degree distribution from the edge perspective	7
$\mu_{f_j \to v_i}$	the message from the factor node f_j to the variable node v_i	14
$\mu_{v_i \to f_j}$	the message from the variable node v_i to the factor node f_j	14
$\Omega(x)$	check node degree distribution of LT codes from the node perspective	42
ρ	minimal rate	96
$\rho(x)$	check degree distribution from the edge perspective	7
$P(x)$	check degree distribution from the node perspective	7
$\bar{\gamma}$	average SNR	95
θ_i	the phase of the discrete-time channel gain h_i	98
θ_n	the initial phase of the n^{th} ray in Jakes method	100
$\mathrm{AS}_{\mathrm{SD}}^{(l)}$	after the i^{th} iteration, the sum of absolute values of soft decisions for all information bits	101
B_d	maximum Doppler shift	100

$\mathbf{c}$	codeword	8
$C(\beta)$	the capacity corresponding to the input cost β	1
C_{erg}	the ergodic capacity of a Rayleigh fading channel	95
$\hat{\mathbf{c}}$	estimated codeword	10
$\mathtt{CHK}(v_i, v_j)$	the factor node operation applied to messages from variable nodes v_i and v_j	14
$\hat{\mathrm{c}}_i$	the i^{th} estimated code bit	14
CM	correlation metric	10
$\binom{n}{k}$	number of ways of choosing k from n possibilities	21
$\mathbf{c}_p$	pre-coded word in a Raptor code	25
$C_{UC}(\gamma_i)$	the capacity of an AWGN channel with unconstrained input alphabet and the SNR γ_i	95
d_c	degree of check node	5
$d_c^{(avg)}$	average right degree	43
d_L	the left degree of the LDPC H-graph in a Raptor code	41
d_{min}	minimum Hamming distance	20
d_s	degree of variable node	5
$d_s^{(avg)}$	average left degree	43
E_{mb}/N_0	transmitted energy per message bit over N_0	67
$E_s^{(avg)}$	average transmitted energy per symbol	66
E_s/N_0	transmitted energy per symbol over N_0	47
$\mathbf{E}$	expectation operation	96
$E_1(x)$	exponential integral	96
f_j	the j^{th} factor node	14
$\mathbf{G}$	generator matrix	8
$\mathbf{G}_c$	the generator matrix of LT encoder in a Raptor code	26

Chapter 1

Introduction

The noisy-channel coding theorem due to Claude Shannon states that to be received with arbitrarily small frequency of errors, digital data has to be transmitted at a rate less than capacity [1, page 81, 324]. Supposing the information source is a stationary binary random process with equally probable 0 and 1, the theorem can be simplified as the following inequality [2, page 59],

$$\frac{N_m}{N_c} \leq \frac{C(\beta)}{1 - H_2(P_e)}, \tag{1.1}$$

where N_m/N_c is the code rate (N_m is the message word length and N_c is the codeword length), $C(\beta)$ indicates the capacity corresponding to the input cost β (i.e. transmission power), and $H_2(P_e)$ is the binary entropy of message bit error rate, P_e.

Using a very large N_c, low-density parity-check (LDPC) codes and Turbo codes can be designed at a rate very close to the capacity of a given channel, while offering a zero-approaching P_e. In wireless communication, however, the channel keeps changing and so does the capacity, $C(\beta)$. To ensure reliable communication, traditional approaches involve fixed rates, N_m/N_c, which are designed for worst-case $C(\beta)$, and

therefore are inefficient when the channel is good. Improved approaches are adaptive coding/modulation (ACM), using variable N_m/N_c to match the actual instantaneous $C(\beta)$. The receiver continually estimates $C(\beta)$ and sends measurements back to the transmitter, which changes N_m/N_c accordingly. However, if the channel condition is imprecisely known at the transmitter and a code rate higher than $C(\beta)$ happens to be chosen, P_e will increase such that the inequality in Eq. (1.1) can hold.

A better approach is to use rateless codes with fixed N_m and ever-increasing N_c to adapt to any channel from $C(\beta) = 1$ down to $C(\beta) \simeq 0$. Rateless codes, unlike traditional fixed-rate fixed-length block codes, are characterized by a continuous stream of code bits, all generated from a single fixed-length message word. The transmitter continuously produces and transmits these code bits until the receiver collects enough information to reliably recover the original message word. If the channel is in a good state, only a small number of code bits will need to be transmitted, whereas if the channel is poor, reliable communication requires transmission of a larger number of bits. As such, the code automatically adapts to the channel, so, unlike with traditional ACM schemes, knowledge of the channel state is not required at the transmitter. The idea behind rateless codes, termed incremental redundancy (IR) by Mandelbaum in 1974 [3], is to fully exploit every transmitted symbol at the receiver. IR is also referred to as type II HARQ (hybrid automatic repeat-request).

This thesis focuses on how to apply rateless codes for practical wireless communication systems. In particular, we consider three different rateless codes. One is Raptor codes [4] from Fountain codes family. The other two come from rate-compatible low-density parity-check (RC-LDPC) codes family: rate-compatible irregular repeat-accumulate (RC-IRA) codes [5], and rate-compatible quasi-cyclic LDPC (RC/QC-LDPC) codes proposed in 3GPP2 (3rd Generation Partnership Project 2) and 802.20

MBWA (Mobile Broadband Wireless Access) [6,7]. While previous literature on rateless codes mainly use binary phase-shift keying (BPSK), we show the performance of rateless codes also under 16-symbol quadrature amplitude modulation (16-QAM) and 64-QAM, with different message word lengths and under different channels, namely, discrete-time memoryless channels with additive Gaussian white noise (AWGN) and Rayleigh fading channels.

Background knowledge on LDPC codes and rateless codes are given in Chapter 2 and 3, respectively. The performance comparison of the three codes over the binary-input AWGN channel are then presented in Chapter 4, where we also studied possible directions to improve RC-IRA codes. Applying rateless codes in the AWGN channel with higher-order modulation schemes is discussed in Chapter 5. In these two chapters, we also investigate different decoding scheduling on RC-IRA codes, and discuss several directions to improve their throughput. To examine the performance of rateless LDPC coded modulation systems, in Chapter 5, we use short-block RC-IRA codes to show the effects of iterative de-mapping, labeling, and reliability-based coded modulation. In Chapter 6, we apply RC-IRA and RC/QC-LDPC codes over fast and slow Rayleigh fading channels, and present their throughput and outage probability performance, using different message word length and modulations. Chapter 7 serves as a summary of our work and also includes some suggestions on possible future work.

Chapter 2

Background on LDPC Codes

Attracted by the capacity-approaching property of LDPC codes, we mainly construct rateless codes based on LDPC codes. In this chapter, we present some basic knowledge on LDPC codes.

2.1 Regular LDPC Codes

LDPC codes, proposed by Gallager in 1962 [8], are capacity-approaching linear block codes, but due to insufficient processing power at that time, people were unable to decode LDPC codes with satisfying performance. However, since the 1990's, by combining *belief propagation* (BP) algorithm [9] and modern powerful processors, we are now able to decode LDPC codes. In 2001, LDPC codes were reported to approach the capacity of the binary-input AWGN channel within 0.0045 dB [10].

Like other linear block codes, an LDPC code with N_m message bits and N_c code bits can be represented by N_j independent linear equations over a Galois field with m elements (m is assumed as 2 throughout this thesis), where N_j, equal to $N_c - N_m$, is the number of parity bits. These equations are represented by a N_j-by-N_c parity-check matrix, $\mathbf{H}$. Each row of $\mathbf{H}$ represents a parity check equation and each column

corresponds to a code bit. There are much fewer 1's than 0's in **H**; hence the name, *low-density parity-check* codes. **H** can also be represented by a *bipartite* graph, $\mathcal{G}_H$, also known as a *Tanner graph* [11]. In $\mathcal{G}_H$, each code bit is represented by a *variable node*. Each linear equation, composed of d_c code bits, is represented by a *check node* with d_c edges connected to d_c variable nodes (code bits), where d_c is defined as the degree of the check node. Similarly, the degree of a variable node is defined as d_s, the number of edges incident to the variable node. Given a $\mathcal{G}_H$, if d_s is the same for all variable nodes and d_c is the same for all check nodes, then the LDPC code is called *regular* and notated as (d_s,d_c)-regular. Since $d_s N_c = d_c(N_c - N_m)$, the rate of a (d_s,d_c)-regular LDPC code is $(1 - d_s/d_c)$. A simple (3,4)-regular LDPC code with rate 1/4 and its corresponding Tanner graph are shown in Fig. 2.1.

If a set of distinct nodes and edges in $\mathcal{G}_H$ form a closed path, we call the path a *cycle*. The *length* of a cycle is the number of nodes (or edges) involved in the cycle and is always even for a bipartite graph. A (2,3)-regular LDPC code and its corresponding Tanner graph are shown in Fig. 2.2, where a cycle is highlighted with bold lines. The length of the cycle is 6 and we call it a length-6 cycle.

The 4-by-6 **H** actually gives us rate 1/2 instead of 1/3 because the rank of **H** is 3. This can be easily seen by adding all rows of **H** together. Since $d_s = 2$, we have two "1" added together for each column; the sum will be a all-zero vector. Therefore, the rank of **H** is at most $N_j - 1$. In fact, this holds for any regular LDPC codes with even d_s.

In the Tanner graphs of above examples, variable nodes are customarily put on the left and check nodes on the right. If the variable node degree is the same for all variable nodes, the code is said to be *left-regular*; if the check node degree is the same for all check nodes, the code is said to be *right-regular*.

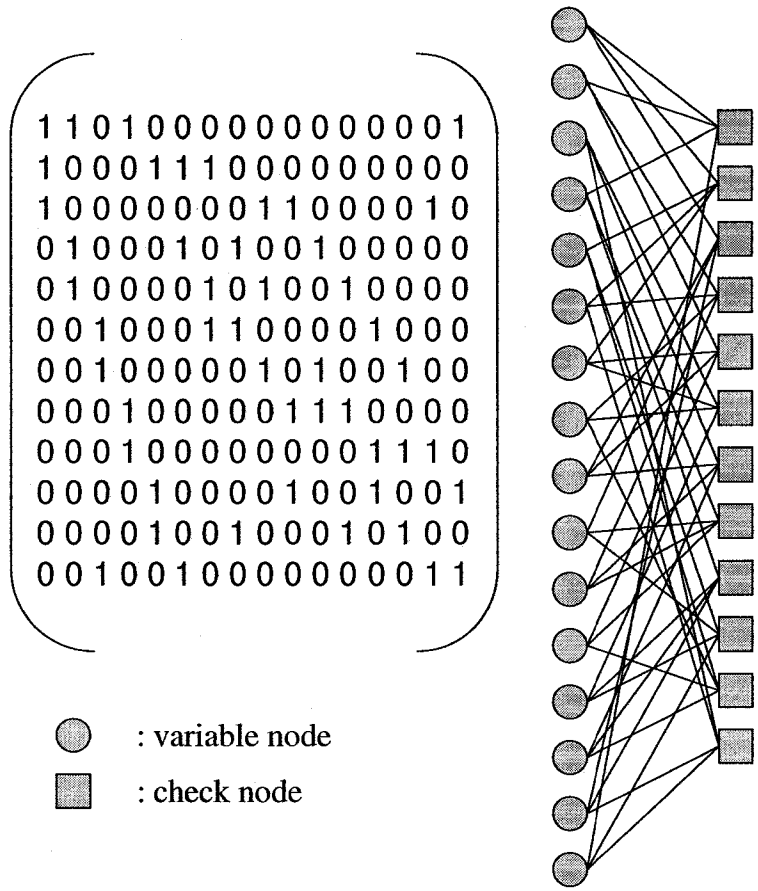

Fig. 2.1: A (3,4)-regular LDPC code: parity-check matrix and Tanner graph

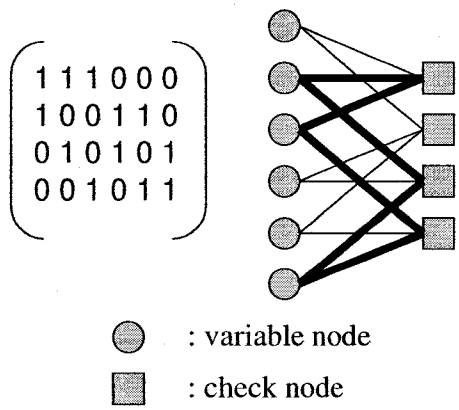

Fig. 2.2: A (2,3)-regular LDPC code: parity-check matrix and Tanner graph

2.2 Irregular LDPC Codes

Regular LDPC codes were extended to have unequal variable degree (d_s) and check degree (d_c) in [12], termed *irregular* LDPC codes. Irregular LDPC codes tend to have lower bit error rate (BER) than regular LDPC codes, provided the decoder uses the BP algorithm. To be specific, plotting BER against signal-to-noise ratio (SNR) curves for irregular and regular LDPC codes, we will observe the BER of irregular LDPC codes decreases slowly with increasing SNR, then drops sharply when the SNR keeps increasing, and finally reaches a floor even when the SNR becomes very high. The second region is vividly called the waterfall region and the third one is the error-floor region. We will also observe regular LDPC codes have a higher BER in the waterfall region and a lower BER in the error-floor region. Since most of practical applications involve SNR in the waterfall region, irregular LDPC codes are favored. The reasons of above phenomena are related to the BP algorithm; we will discuss it later in Section 2.4.

To specify the degree profile of an irregular LDPC code, we only need the *variable degree distribution*, $\lambda(x)$ and the *check degree distribution*, $\rho(x)$, since permutating rows or columns of $\mathbf{H}$ does not change the codebook. A polynomial is used to express $\lambda(x)$ as $\lambda(x) := \Sigma_i \lambda_i x^{i-1}$, where λ_i is the fraction of edges incident to variable nodes with degree i [12]. Similarly, $\rho(x)$ is defined as $\rho(x) := \Sigma_i \rho_i x^{i-1}$, where ρ_i is the fraction of edges emanating from check nodes with degree i. The above distributions are defined from the edge perspective. We can also define them from the node perspective [13, page 73]: $\Lambda(x) := \Sigma_i \Lambda_i x^i$ and $P(x) := \Sigma_i P_i x^i$, where Λ_i (P_i) is the fraction of variable (check) nodes with degree i. Since all edges emanating from check nodes are incident to variable nodes, we have the constraint $\Sigma_i \Lambda_i i = \Sigma_i P_i i$. Finally, given $\lambda(x)$, $\Lambda(i) = N_c \frac{\lambda(i)/i}{\Sigma_j \lambda(j)/j}$, and a similar equality applies for $P(i)$.

Density evolution is proposed as a useful tool to analyze the decoding performance of LDPC codes [14]. It can be used to find optimal $\lambda(x)$ and $\rho(x)$ that provide the best error-correcting capabilities in a BP decoder [15]. The numerical results in [15] show that in the binary-input AWGN channel, optimal LDPC codes tend to have consecutive check degrees (check-concentrated degrees) and a few separate variable degrees, usually including 2, 3, the maximal variable degree, and a few degrees in-between. This observation agrees with the statement given by Luby *et al* that right-regular LDPC codes are capacity-achieving if using a BP decoder in erasure channels [12, 16]. These experimental patterns on check and variable degrees can be used to speed up the searching algorithm of optimal degree distributions.

Once we have optimized degree distributions, it is necessary to connect edges between check and variable nodes. Before discussing this problem, we need know how to use $\mathbf{H}$ to pick a codeword for a given message word at the transmitter and to estimate a message word by received samples at the receiver, i.e. encoding and decoding. The construction of $\mathbf{H}$ is closely related to how we decode LDPC codes.

2.3 LDPC Encoder

To conveniently encode a given message word, $\mathbf{m}$, we need find the *generator matrix*, $\mathbf{G}$, such that the codeword, $\mathbf{c} = \mathbf{mG}$. Since a valid $\mathbf{c}$ has to satisfy $\mathbf{cH}^T = \mathbf{0}$, we have $\mathbf{mGH}^T = \mathbf{0}$, where $\mathbf{H}^T$ indicates the transpose of $\mathbf{H}$. To ensure that $\mathbf{mGH}^T = \mathbf{0}$ holds for any $\mathbf{m}$, we need $\mathbf{GH}^T = \mathbf{0}$, i.e. $\mathbf{G}$ is a null space of $\mathbf{H}$. Instead of listing all of 2^{N_m} codewords, this full-rank $\mathbf{G}$ compactly represents a subspace of a vector space of N_c-tuples, each row being a basis vector. Applying elementary row operations and column permutations on $\mathbf{G}$, we have multiple combinatorially equivalent generator matrices [17, page 45]. All of these $\mathbf{G}$'s represents the same subspace. To facilitate

encoding, we can pick one $\mathbf{G}$ with the form $[\mathbf{I}_{N_m}|\ \mathbf{P}]$, where $\mathbf{I}_{N_m}$ is an N_m-by-N_m identity matrix and $\mathbf{P}$ is an N_m-by-N_j matrix. Thus, first N_m bits of each codeword generated from this $\mathbf{G}$ are message bits and the other code bits are linear combination of message bits; the encoding complexity depends on the number of 1's in the N_m-by-N_j matrix. Codes generated from this form of $\mathbf{G}$ are called *systematic* codes. LDPC codes are designed starting from $\mathbf{H}$, therefore a systematic $\mathbf{G}$ can be obtained by reducing the $\mathbf{H}$ to the reduced echelon form $[\mathbf{P}^T|\ \mathbf{I}_{N_j}]$ through Gaussian elimination.

The above LDPC encoder has a practical problem: the encoding complexity is $o(N_m N_c)$ because Gaussian elimination applied to $\mathbf{H}$ makes the reduced echelon form of $\mathbf{H}$ not sparse anymore. Unless the rate is very high (i.e. $\mathbf{I}_{N_m}$ dominates $\mathbf{G}$), the encoding complexity is non-linear on N_m or N_c. To solve this problem, $\mathbf{H}$ can be designed with a lower-triangular form $[\mathbf{H}_1|\ \mathbf{H}_2]$, where $\mathbf{H}_1$ is a sparse N_j-by-N_m matrix and $\mathbf{H}_2$ is a low-triangular N_j-by-N_j square matrix. Assigning information bits as the first N_m code bits, the encoder can directly use $\mathbf{H}$ to encode $\mathbf{m}$ via back-substitution - there is no need for $\mathbf{G}$. However, low-triangularity poses a constraint on the design of $\mathbf{H}$, and therefore reduces the code performance. One can relax the requirement of low-triangularity and design a $\mathbf{H}$ with almost low-triangular form. Based on this quasi-low-triangular form, an efficient LDPC encoding method was proposed in [18] such that the encoding complexity is upper bounded by $N_c + g^2$, where g is a measurement on the distance between $\mathbf{H}$ and a lower-triangular matrix. An extreme case is $g = 0$, where a low-triangular $\mathbf{H}$ has linear encoding complexity. For LDPC codes with good performance, g can be adjusted in the order of $\sqrt{N_c}$ through row/column permutations on $\mathbf{H}$ [18]. Thus, we have a linear encoding complexity. This encoding method is based on block matrix calculation and can be found in [18].

2.4 LDPC Decoder

To minimize the *word error rate* (WER), the optimum decoder is a maximum *a posteriori* probability (MAP) decoder on a sequence-by-sequence basis, estimating the most likely codeword based on the N_c-dimensional *received sample vector*, $\mathbf{r}$. Provided 2^{N_m} message words are equally probable, finding the $\mathbf{c}$ that maximizes the probability $\mathrm{P}(\mathbf{c}|\mathbf{r})$, is equivalent to finding the $\mathbf{c}$ that maximizes $\mathrm{P}(\mathbf{r}|\mathbf{c})$. The MAP decoder then becomes the *maximum-likelihood sequence decoder* (MLSD) [19, page 243]. Knowing the channel can help us build a simpler MLSD. In a binary-input AWGN channel, $\mathbf{r} = \mathbf{s} + \mathbf{n}$, where $\mathbf{s}$ is the *transmitted symbol vector* mapped from $\mathbf{c}$ by $\mathbf{s} = 2\mathbf{c} - 1$, and $\mathbf{n}$ is discrete-time white Gaussian noise sequence. The MLSD can then be simplified to finding the $\mathbf{c}$ that maximizes the *correlation metric*, $CM = \mathbf{r} \cdot \mathbf{s} = \mathbf{r} \cdot (2\mathbf{c} - 1)$, where $\cdot$ operation is dot-product. For simplicity, CM's are calculated in the log domain, since we only care which codeword is most likely and not care the actual value of $\mathrm{P}(\mathbf{r}|\mathbf{c})$. A straightforward *maximum likelihood* (ML) decoder calculates 2^{N_m} CM's to find the estimated codeword, $\hat{\mathbf{c}}$. Its exponential computational complexity is prohibitive for practical applications. Can we solve ML sequence-by-sequence decoding problem in linear or polynomial time? The answer is very hard, if not impossible. In 1978, Berlekamp *et al* showed that ML sequence-by-sequence hard-decision decoding (minimum distance decoding) on binary linear codes is NP-complete [20]. Based on this result, Koetter and Vardy showed ML sequence-by-sequence soft-decision decoding on general linear codes is NP-hard [21]. NP-complete problems are the hardest non-deterministic polynomial-time (NP) problems and NP-hard problems are, informally speaking, at least as hard as any NP problems. No polynomial-time algorithm has been found so far for any NP-complete problems. For formal descriptions of these terms, please refer to books on computational complexity theory. A good reference list can be found in [22, page 1021].

Fortunately, cycle-free (tree-like) LDPC codes can have efficient iterative decoding algorithms that will converge to optimal MAP decisions, either on a sequence-by-sequence or a symbol-by-symbol basis. The decoding algorithm for MLSD is called the *max-sum* or *min-sum* algorithm [23] and the decoding algorithm for symbol-by-symbol MAP is called the *sum-product* algorithm, or the BP algorithm. They have the same computational complexity. The high efficiency of these algorithms is made possible by graphical representations of $\mathbf{H}$. One such representation is the Tanner graph we introduced before. It was generalized as *factor graphs* by Kschischang *et al* in 2001 [24], where *factor nodes* and variable nodes correspond to check nodes and variable nodes in Tanner graphs, respectively. BP decoding algorithm, along with other graph-based decoding algorithms such as the forward/backward algorithm (BCJR algorithm [25]) and Viterbi algorithm, are generalized altogether as the sum-product algorithm. In this thesis, the terminology in factor graph and Tanner graph is used interchangeably.

However, good linear block codes always have cycles. Cycle-free binary linear code of rate $r(> 0.5)$ contains at least $N_c(r - 0.5)$ codewords of Hamming weight 2 [13, page 59]. For $r < 0.5$, it was shown that the minimum Hamming distance is at most $2/r$ [26]. If we apply the min-sum or sum-product algorithm to factor graphs with cycles, they do not converge to optimal MAP decisions and become sub-optimal. Surprisingly, previous simulation results show that the sum-product algorithm still works well for graphs with cycles, while the min-sum performs slightly worse than the sum-product algorithm. Therefore, the sum-product algorithm is used throughout the following discussions. Reader can refer to [23, 24, 27] for a comprehensive treatment on the min-sum algorithm.

The sum-product algorithm is an iterative message-passing algorithm. It is eas-

ier to understand this algorithm from the perspective of the BP algorithm: belief propagation. In the beginning, based on the received values, each variable node has initial belief or probability on its own bit decision, e.g. a log-likelihood-ratio, and will broadcast this belief to its neighboring factor nodes. Now we have beliefs flowing from variable nodes to factor nodes. After receiving these beliefs, each factor node has certain belief on bit decisions of its neighboring variable nodes, and will carry out some factor node operation and send results back to its neighboring variable nodes. To avoid double-counting beliefs, each neighboring variable node only receives the belief solely based on other incident messages to the factor node. At this point, each variable node has incident messages from neighboring factor nodes and its initial belief from the received value; it will have an updated belief on its bit decision. Then each variable node sends this belief to neighboring factor nodes. Similarly, to avoid double-counting messages, each neighboring factor node will only receive the belief solely based on the sum of other incident messages to the variable node. The decoder terminates after repeating the above process for L runs. The bit decisions of the variable nodes provide $\hat{\mathbf{c}}$. To be concrete, let us look at a simple example of an LDPC code to clarify the details of the BP algorithm. BP has some variants. We will focus on the *log-likelihood-ratio* (LLR) BP algorithm due to its reduced computational complexity. Refer to [28] for details of other variants.

Given

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

we have corresponding factor graph shown in Fig. 2.3, where four sub-figures exhibit four different stages of the BP algorithm.

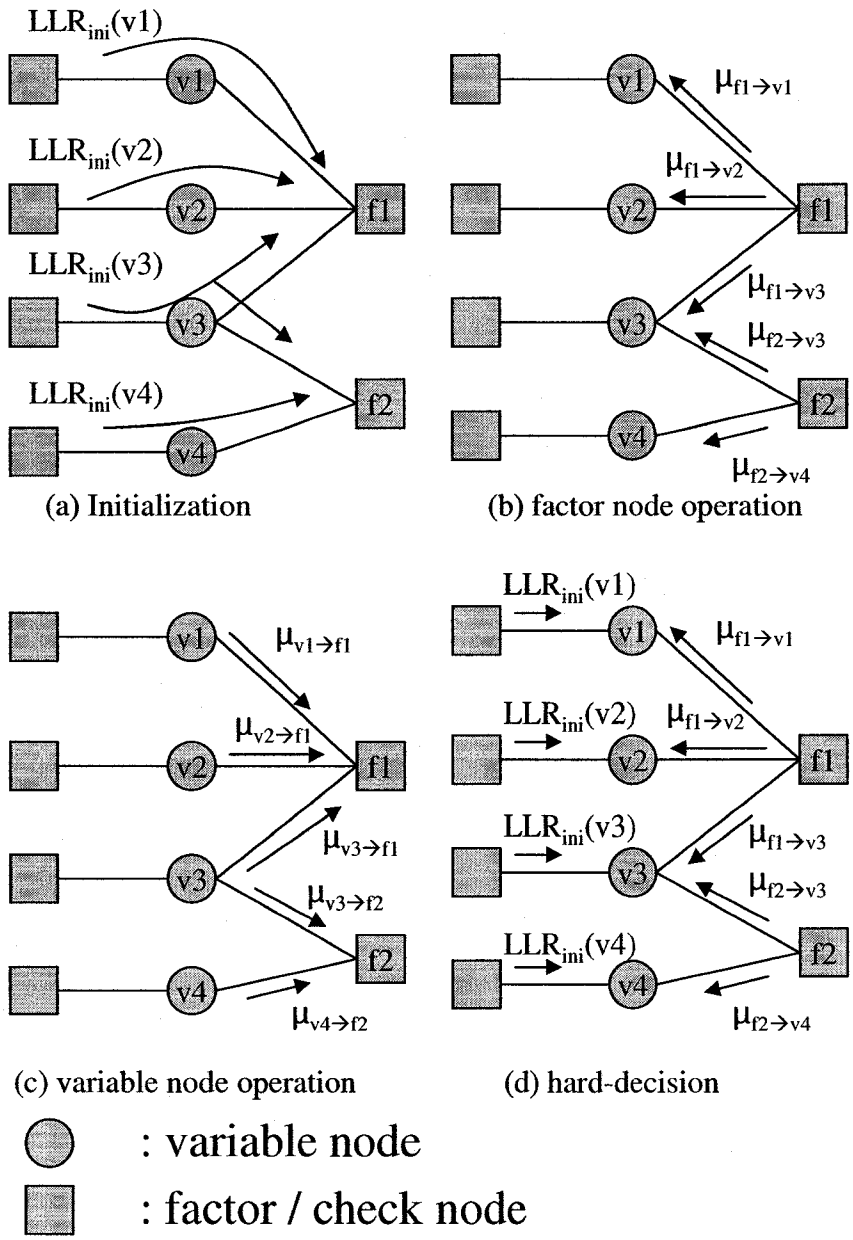

Fig. 2.3: BP decoding on factor graph

Step 1. Initialization

This stage is shown in Fig. 2.3(a). To estimate $\hat{\mathbf{c}} = [\hat{c}_1\ \hat{c}_2\ \hat{c}_3\ \hat{c}_4]$ from a given $\mathbf{r} = [r_1\ r_2\ r_3\ r_4]$, the initialization stage is only carried out once. Compared to a Tanner graph, N_c factor nodes are added to initialize the message flow (in our case N_c is 4). We are using the LLR-BP algorithm, so the initial belief of variable node v_i is given by $\mathrm{LLR}_{ini}(v_i) = \ln \frac{\mathrm{P}(r_i|c_i=0)}{\mathrm{P}(r_i|c_i=1)}$, where $i \in \{1, 2, 3, 4\}$. Each variable node broadcasts its LLR_{ini} to their neighboring factor nodes.

Step 2. Factor node operation

This stage is shown in Fig. 2.3(b). The message from the variable node v_i to the factor node f_j is denoted as $\mu_{v_i \to f_j}$, and the message from f_j to v_i is $\mu_{f_j \to v_i}$. The factor node f_1 in Fig. 2.3 performs the factor node operation,

$$\mathtt{CHK}(v_1, v_2) = \ln(\frac{1 + e^{m_1+m_2}}{e^{m_1} + e^{m_2}}), \tag{2.1}$$

where m_1 is a short-hand notation for $\mu_{v_1 \to f_1}$ and m_2 is $\mu_{v_2 \to f_1}$. The factor node f_1 then sends $\mathtt{CHK}(v_1, v_2)$ to v_3. Similarly, $\mathtt{CHK}(v_2, v_3)$ is sent to v_1 and $\mathtt{CHK}(v_1, v_3)$ is sent to v_2. The factor node operation defined in Eq. (2.1) can be transformed into two Jacobian logarithms and efficiently calculated using table-lookup techniques [29, 30].

If the degree of a factor node f_i is larger than 3, the factor node operation can be carried out in a recursive way. This is understandable since f_i is equivalent to multiple cascaded factor nodes, as shown in Fig. 2.4.

The factor node operation over multiple variable nodes is given by,

$$\begin{aligned} \mathtt{CHK}(v_1, v_2, ..., v_{d_c-1}) &= \mathtt{CHK}(\mathtt{CHK}(v_1, v_2), v_3, ..., v_{d_c-1}) \\ &= \mathtt{CHK}(\mathtt{CHK}(...(\mathtt{CHK}(v_1, v_2), v_3), ..., v_{d_c-2}), v_{d_c-1}). \end{aligned} \tag{2.2}$$

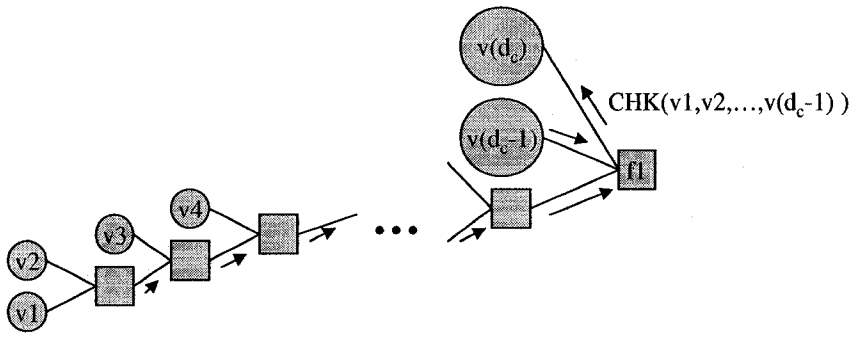

Fig. 2.4: Factor node operation over multiple variable nodes

This message is sent to v_{d_c}. Messages to other variable nodes can also be calculated by Eq. (2.2). In fact, by remembering the intermediate messages in Fig. 2.4, factor node operations can be carried out in an efficient way [30].

There is an important property of CHK operation: any zero incident message from a variable node v_i forces all outgoing beliefs from its neighboring factor nodes to be zero, except the beliefs sent back to v_i. Therefore, whenever we insert an *erasure* to a variable node v_i (i.e. $\text{LLR}_{ini}(v_i) = 0$), all variable nodes that are connected to neighboring factor nodes of v_i, will receive a zero belief. If two erasures are inserted for one factor node f_i, all outgoing messages from f_i become zero. This property introduces a concept called the *stopping set*. Denoting the set of variable nodes in a given factor graph as $\mathcal{V}$, a stopping set $\mathcal{S}$ is a subset of $\mathcal{V}$, such that all neighboring factor nodes of $\mathcal{S}$ are connected to $\mathcal{S}$ at least twice [31]. Since the union of two stoping sets $\mathcal{S}_1$ and $\mathcal{S}_2$ is still a stopping set, any subset of $\mathcal{V}$ contains one unique maximal stopping set. One can easily see that if the variable nodes in $\mathcal{S}$ do not have reliable LLR_{ini}'s, all neighboring factor nodes of $\mathcal{S}$ may be paralyzed. The concept of stopping set is important for designing a good LDPC code and will be elaborated in Section 2.5.

Step 3. Variable node operation

This stage is shown in Fig. 2.3(c). Variable nodes send beliefs on their bit decisions to neighboring factor nodes. The beliefs shown in Fig. 2.3(c) are given by,

$$\mu_{v_1 \to f_1} = \mathrm{LLR}_{ini}(v_1),$$
$$\mu_{v_2 \to f_1} = \mathrm{LLR}_{ini}(v_2),$$
$$\mu_{v_3 \to f_1} = \mathrm{LLR}_{ini}(v_3) + \mu_{f_2 \to v_3},$$
$$\mu_{v_3 \to f_2} = \mathrm{LLR}_{ini}(v_3) + \mu_{f_1 \to v_3},$$
$$\mu_{v_4 \to f_2} = \mathrm{LLR}_{ini}(v_4).$$

The rule is mentioned before: a factor node only receive the sum of beliefs from other neighboring factor nodes to the variable node. The node v_i first calculates the soft decision of v_i, i.e. the sum of all incident messages from neighboring factor nodes including $\mathrm{LLR}_{ini}(v_i)$. The belief from v_i to f_j is simply the soft decision of v_i minus the earlier belief from f_j to v_i.

Step 4. Hard decision on variable nodes

This stage is shown in Fig. 2.3(d). The hard decision of v_i is based on the soft decision of v_i. We then have the estimated codeword $\hat{\mathbf{c}}$, each component of which corresponds to the hard decision of a variable node. If $\hat{\mathbf{c}}\mathbf{H}^T$ is zero, the algorithm terminates. However, this way to check decoding success runs a risk of undetected word error (UWE) - the decoder might converge to another valid codeword at a certain point during iterations. To reduce the *undetected word error rate* (UWER), we package message words with a cyclic redundancy check (CRC) code. After the hard decision of each iteration, the decoder verifies the CRC of $\mathbf{m}$, denoted as CRC($\mathbf{m}$), to detect the

decoding success. By doing so, we assumed the probability that $\mathrm{CRC}(\hat{\mathbf{m}}) = \mathrm{CRC}(\mathbf{m})$ but $\hat{\mathbf{m}} \neq \mathbf{m}$ is smaller than the UWER.

Step 5. Iterations terminate

Step 2, 3 and 4 comprise one *iteration*, corresponding to Fig. 2.3(b)-(d). The decoder keeps iterating until $\mathrm{CRC}(\hat{\mathbf{m}})$ is correct or no messages change. The latter only happens for acyclic factor graphs, where the BP algorithm will finally converge to the optimal MAP decoding decision. For factor graphs with cycles, messages running in the graph are always changing. For a practical LDPC decoder, if the number of iterations is larger than a pre-set number L and $\mathrm{CRC}(\hat{\mathbf{m}})$ is still incorrect, a decoding failure is declared. Simulations on LDPC codes suggest that $L = 100$ is large enough to provide good decoding performance.

The above example is an acyclic factor graph and therefore, optimal decoding decision can be obtained by the BP algorithm. Take v_1 as an example, the log-ratio of MAP decoding on a symbol-by-symbol basis gives us,

$$\begin{aligned}
\ln \frac{\mathrm{P}(c_1 = 0|\mathbf{r})}{\mathrm{P}(c_1 = 1|\mathbf{r})} &= \ln \frac{\mathrm{P}(\mathbf{r}, c_1 = 0)}{\mathrm{P}(\mathbf{r}, c_1 = 1)} \\
&= \ln \frac{\sum_{\mathbf{c}:c_1=0} \mathrm{P}(\mathbf{r}, \mathbf{c})}{\sum_{\mathbf{c}:c_1=1} \mathrm{P}(\mathbf{r}, \mathbf{c})} \\
&= \ln \frac{\sum_{\mathbf{c}:c_1=0} \mathrm{P}(\mathbf{r}|\mathbf{c})}{\sum_{\mathbf{c}:c_1=1} \mathrm{P}(\mathbf{r}|\mathbf{c})} \\
&= \ln \frac{\sum_{\mathbf{c}:c_1=0} \{\prod_{i=1}^{4} \mathrm{P}(r_i|\mathbf{c})\}}{\sum_{\mathbf{c}:c_1=1} \{\prod_{i=1}^{4} \mathrm{P}(r_i|\mathbf{c})\}} \\
&= \ln \frac{\sum_{\mathbf{c}:c_1=0} \{\prod_{i=1}^{4} \mathrm{P}(r_i|c_i)\}}{\sum_{\mathbf{c}:c_1=1} \{\prod_{i=1}^{4} \mathrm{P}(r_i|c_i)\}}.
\end{aligned} \tag{2.3}$$

The second equality is due to $\mathrm{P}[(A \cup B)M] = \mathrm{P}(AM) + \mathrm{P}(BM)$, where the events A and B are mutually exclusive [32, page 29]. The third equality comes from the

assumption that $\mathbf{m}$ is equally likely and therefore $\mathbf{c}$ is equally likely. The fourth equality is due to two assumptions: one is a memoryless channel; another is a white noise sequence independent of transmitted symbols. The fifth equality is based on the conditional independence between r_1 and code bits other than c_1: $\mathrm{P}(r_1 c_2 c_3 c_4 | c_1) = \mathrm{P}(r_1|c_1)\mathrm{P}(c_2 c_3 c_4|c_1)$. Running the BP algorithm in Fig. 2.3, one can easily verify that the sum of incident beliefs to v_1 will converge to Eq. (2.3) after two iterations.

The order of passing messages in the BP algorithm is termed scheduling. In the above example, each variable node sends messages parallel and so does each factor node. This scheduling is called flooding and is convenient for hardware implementation. However, practical BP decoders still have implementation problems to be carefully addressed. One example is memory access [33]. There are also other kinds of scheduling. We are mainly interested in flooding and Turbo-like scheduling, which will be discussed in Section 3.3.3.

The iterative process of the BP algorithm suggests that variable nodes with higher d_s tend to have more beliefs and therefore have more reliable hard decisions, while factor nodes with higher d_c tend to send less reliable messages. Therefore, variable nodes with higher d_s in irregular LDPC codes will converge first then pass their reliable messages to neighboring factor nodes; other variable nodes with lower d_s can make use of these messages and converge later. This speeds up the convergence of irregular LDPC codes and gives its lower BER in the waterfall region [12]. However, the existence of degree-2 variable nodes increases the probability of having a small maximal stopping set in irregular LDPC codes. And a small maximal stopping set causes the error-floor. On the contrary, regular LDPC codes have larger minimum distance as long as $d_s \geq 3$. Gallager showed that regular LDPC codes with $d_s \geq 3$ has a minimum distance that increases linearly with N_c for fixed d_s and d_c, and

regular LDPC codes with $d_s = 2$ has a minimum distance that increases at most logarithmically with N_c [8, page 7].

2.5 Design of Irregular LDPC Codes

2.5.1 Random Construction

From now on we focus on irregular LDPC codes due to lower BER in the waterfall region than regular LDPC codes. The object of design is to find an $\mathbf{H}$ that provides low BER. Density evolution can help us find good $\lambda(x)$ and $\rho(x)$. The rest of the problem is to assign degrees to each variable node and factor node. Experiments suggest good irregular LDPC codes tend to be check-concentrated. If we concentrate on designing a good right-regular LDPC code, the problem is further simplified to placing 1's for each column in $\mathbf{H}$ according to given $\lambda(x)$, or equivalently, connecting edges between variable nodes and factor nodes under the constraint that d_c is as concentrated as possible. Since cycles in factor graph make the BP decoder sub-optimal, we shall construct a factor graph with a large *girth*, the minimum length of cycles in a graph [34, page 83]. When N_c is large, randomly placing 1's in $\mathbf{H}$ often suffices to obtain large girth and has been shown to approach the capacity of the binary-input AWGN channel within 0.0045 dB [10].

While N_c is small or medium (under a few thousands), X.-Y. Hu *et al* proposed the progressive-edge-growth (PEG) algorithm to construct a check-concentrated $\mathbf{H}$ with large girth [35]. The PEG algorithm starts from a factor graph without any edges. Given a variable node v_i, PEG picks one check node at a time until $d_s(v_i)$ edges are connected. The first priority is to pick the farthest check node. In the case of tie, the second priority is to pick the check node with minimal d_c. If there is still a tie, PEG randomly picks one check node to connect. PEG always starts from

variable nodes with the lowest d_s, thus short-length cycles solely composed of nodes with lower d_s are less likely than short-length cycles involving nodes with higher d_s. This is good for increasing the minimum distance [36]. To understand the reason, let us look at a graph $\mathcal{G}$ that has a cycle composed of only degree-2 variable nodes. If we flipped these degree-2 variable nodes, all constraints in $\mathcal{G}$ still hold. Thus, the minimum distance d_{min} is no larger than the number of degree-2 variable nodes in the cycle. This example also suggests that there are some relationships between girth and d_{min}.

In 2003, Tian *et al* proposed extrinsic message degree (EMD) as a metric to measure cycle connectivity in bipartite graphs, and to expose some relations between stopping set, cycles, and d_{min} [37]. They showed that for a code with d_{min} represented by $\mathcal{G}$, there exists at least one stopping set with d_{min} variable nodes, denoted as $\mathcal{S}_{d_{min}}$. This means small d_{min} implies small stopping sets. Thus, if the stopping set is not small, then d_{min} cannot be small. They also showed $\mathcal{S}_{d_{min}}$ must be comprised of multiple cycles, provided that $d_s \geq 2$ for all variable nodes and there is no cycle composed of only degree-2 variable nodes. Therefore, to construct a code with large d_{min}, we try to avoid short cycles composed of only low degree variable nodes and allow short cycles including high degree variable nodes. Since stopping sets are composed of multiple cycles, compared to a low degree variable node, a high degree variable node in a short cycle has more extrinsic connections, and therefore have more chances to transverse other cycles and increase the size of the stopping set. Based on the above results, the authors of [37] proposed the approximate cycle EMD (ACE) algorithm to construct LDPC codes with high d_{min}. Note that small stopping sets do not necessarily imply small d_{min}. As we discussed in Section 2.4, the problem with small stopping sets is related to the factor node operation in the BP algorithm.

2.5.2 Algebraic Construction

For short to moderate block lengths, directly designing codes with large d_{min} is very hard. In fact, finding the d_{min} of a given linear block code is already intractable. Since the sum of two codewords is also a codeword, d_{min} is the minimum Hamming weight of the code, i.e. the minimum number of dependent columns in $\mathbf{H}$. Thus, we need $\binom{N_c}{d}$ vector summations to verify whether d_{min} is equal to d. For a binary linear block code, finding d_{min} had been conjectured as NP-complete in [20] and proved by Vardy in [38]. To simplify this problem, in addition to random construction, we also rely on well-established algebra theory to construct good short block codes. These codes are constructed based on finite geometry [39] or circulant permutation matrices [40]. Many codes based on finite geometry have equivalent forms on circulant permutation matrices [41]. LDPC codes based on circulant permutation matrices are known as quasi-cyclic LDPC (QC-LDPC) codes. Shifting any codeword by a certain number of bits, we still have a valid codeword; hence, *quasi-cyclic* codes. QC-LDPC codes can be constructed with good girth and large d_{min}. Also, as quasi-cyclic (QC) codes, QC-LDPC codes can be systematically encoded with linear complexity using a shift register [17, page 256-258]. This is similar to how we encode cyclic codes [42, page 112-116]. QC-LDPC codes outperform randomly constructed LDPC codes for short to moderate block lengths [43]. For large block lengths, random LDPC codes have better distance property than QC-LDPC codes.

A binary (d_s, d_c)-regular QC-LDPC code $\mathbf{H}$ is composed of $n_j n_c$ circulant permutation matrices, given by

$$\mathbf{H} = \begin{bmatrix} I(m_{0,0}) & I(m_{0,1}) & I(m_{0,2}) & \cdots & I(m_{0,n_c-1}) \\ I(m_{1,0}) & I(m_{1,1}) & I(m_{1,2}) & \cdots & I(m_{1,n_c-1}) \\ \vdots & & & \ddots & \vdots \\ I(m_{n_j-1,0}) & I(m_{n_j-1,1}) & I(m_{n_j-1,2}) & \cdots & I(m_{n_j-1,n_c-1}) \end{bmatrix},$$

where $I(m_{i,j})$ is an m-by-m identity matrix shifted leftwards $m_{i,j}$ positions ($0 \leq m_{i,j} \leq m-1$). The N_j-by-N_c $\mathbf{H}$ is thus specified by a $n_j - by - n_c$ base matrix $\mathbf{H}_B$ over GF(m),

$$\mathbf{H}_B = \begin{bmatrix} m_{0,0} & m_{0,1} & m_{0,2} & \cdots & m_{0,n_c-1} \\ m_{1,0} & m_{1,1} & m_{1,2} & \cdots & m_{1,n_c-1} \\ \vdots & & & \ddots & \vdots \\ m_{n_j-1,0} & m_{n_j-1,1} & m_{n_j-1,2} & \cdots & m_{n_j-1,n_c-1} \end{bmatrix}.$$

For (d_s, d_c)-regular QC-LDPC codes, it can be easily seen that $d_s = n_j$ and $d_c = n_c$. Also, $\mathbf{H}$ is not full rank. In fact, any m rows in $\mathbf{H}$ that are corresponded to each row in $\mathbf{H}_B$, always add up to an all-one vector. Therefore, out of N_j rows of $\mathbf{H}$, there are n_j mutually exclusive sets of rows that we can form an all-one vector. This reduces the row space of $\mathbf{H}$ from N_j to at most $N_j - n_j + 1$. Regular QC-LDPC codes can be pruned as irregular QC-LDPC codes by replacing some circulant permutation matrices as all-zero matrices. The resultant codes remain QC. Simulation results in the binary-input AWGN channel showed that irregular QC-LDPC codes have lower BER in the waterfall region than regular QC-LDPC codes, while the error-floor problem still exists in irregular QC-LDPC codes [43].

QC-LDPC codes constructed for large block length usually perform worse than randomly constructed LDPC codes, although some work has been done to build better large-block QC-LDPC codes [44]. They are not of our primary interests in this thesis.

In the above discussions, we presented some methods to increase the girth and the minimum distance of LDPC codes. These methods will also be used to design rateless codes.

Chapter 3

Background on Rateless Codes

In this chapter, we introduce two rateless code families closely related to LDPC codes, namely, Fountain codes and RC-LDPC codes.

3.1 Fountain Codes: LT and Raptor Codes

In 1974, Mandelbaum proposed the concept of incremental redundancy (IR) [3]. Over the following years, people have been combining the IR concept with error control codes to fully exploit every transmitted symbol at the receiver. The codes incorporated into IR are called *rateless* codes because the number of code bits required by the receiver to successfully decode a codeword varies with the channel condition. We have a smaller realized rate for bad channel conditions and larger realized rate for good channel conditions.

Fountain codes, as a family of rateless codes, were originally invented by Luby in 1998 for Internet download applications. If the download application uses fixed-length binary packets (N_m bits), a transmission of one such packet is equivalent to a use of 2^{N_m}-ary erasure channel - CRC embedded in the packet can be used to detect

erasures [45, page 589]. Conventional retransmissions in the Internet are inefficient from the perspective of information theory. Fountain codes were proposed to increase transmission efficiency by using IR.

Luby Transform (LT) codes [46] is the first memeber in the Fountain codes family. Each code bit of a LT code is the sum of randomly picked message bits, where the distribution of the number of message bits contributing to one code bit is the robust soliton distribution [46]. Therefore, the design of LT codes starts from the generator matrix $\mathbf{G}$. Both encoding and decoding are based on $\mathbf{G}$.

Raptor codes [4] are the serial concatenation of a very high rate low-density parity-check (LDPC) pre-code with a LT code. Raptor codes have two advantages over LT codes: higher throughput and linear encoding complexity. By combining LDPC pre-coding with LT codes, Raptor codes have lower error floors in their WER performance. By optimizing the LT code degree distribution, the number of edges per code bit in the corresponding Tanner graph of Raptor codes does not grow with the codeword length; hence, linear encoding complexity.

3.1.1 System Description

The discrete-time system model employing Raptor codes is shown in Fig. 3.1. Each message word contains N_m message bits, in which CRC information is already embedded for error detection at the receiver. The message bits are first pre-coded into N_p intermediate bits via a high rate LDPC encoder; the intermediate bits are then encoded into N_c code bits via a LT encoder. Since Raptor codes are rateless codes, N_c is an unspecified number that keeps increasing until the receiver successfully decodes the codeword and sends back an acknowledgment. If the channel is really poor, N_c could grow to be a very large number. With BPSK, N_c code bits are mapped into

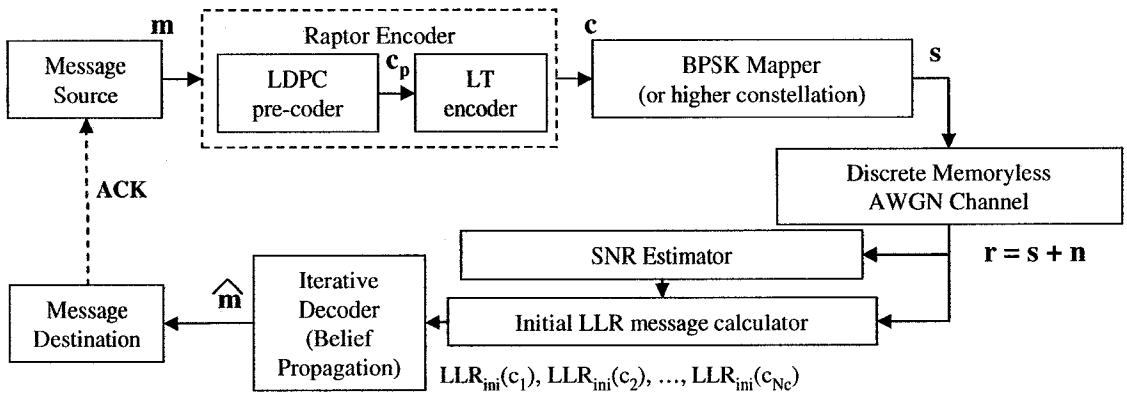

Fig. 3.1: System model using Raptor codes.

$N_s = N_c$ real-valued symbols for channel transmission. The channel model we are using is an AWGN channel with double-sided power spectral density $N_0/2$. The output of the channel would be N_s corrupted samples. The receiver first estimates the SNR of the received samples. Then using the estimated SNR and received samples, the receiver calculates the log-likelihood-ratio (LLR) for each code bit. The Raptor iterative decoder uses the LLR's as the initial messages of the BP algorithm. After T_s more symbols (T_c code bits) are received, the decoder attempts decoding and calculates the CRC. When a correct CRC is obtained, the receiver sends an acknowledgment to the transmitter and the transmitter can then start to transmit the next message word. The following paragraphs describe the details of each block in the system model.

3.1.2 Encoding

Each message word $\mathbf{m}$ containing N_m message bits, is first input into a high rate LDPC pre-encoder characterized by a sparse (N_p-N_m)-by-N_p *parity-check matrix* $\mathbf{H}_p$. Since $\mathbf{H}_p$ is usually not lower-triangular, we need a systematic generator matrix $\mathbf{G}_p$ to facilitate encoding such that the output of pre-encoder $\mathbf{c}_p$ is simply $\mathbf{m} \cdot \mathbf{G}_p$. By applying Gaussian elimination on $\mathbf{H}_p$, we could find $\mathbf{G}_p$, a null space of $\mathbf{H}_p$ with the

form $[\mathbf{I}_{N_m} | \mathbf{P}]$, where $\mathbf{I}_{N_m}$ is an N_m-by-N_m identity matrix and $\mathbf{P}$ is an N_m-by-(N_p-N_m) matrix. We know the sparseness of $\mathbf{H}_p$ does not guarantee the sparseness of $\mathbf{P}$; on the contrary, due to the process of Gaussian elimination, $\mathbf{P}$ is highly likely dense, which implies that average $N_p - N_m$ multiplications are involved in encoding one message bit. However, note we are using a high rate LDPC encoder. This implies N_p-N_m is only a small portion of N_m. Therefore, the encoding complexity could be acceptable even when N_m is large. To further reduce encoding complexity, one can also consider the efficient LDPC encoding method proposed by Richardson and Urbanke in [18].

The intermediate codeword $\mathbf{c}_p$ is then input into a LT encoder characterized by a generator matrix $\mathbf{G}_c$ with N_p rows and possibly infinite columns. With optimized degree distribution of check nodes, $\mathbf{G}_c$ will remain sparse while it is expanding. Hence, the sparseness of $\mathbf{G}_c$ and $\mathbf{H}_p$, combined with the high rate of LDPC pre-coding, ensures linear encoding and decoding complexity of Raptor codes.

In Fig. 3.2, "LDPC H-graph" represents $\mathbf{H}_p$ and "LT G-graph" represents a truncated $\mathbf{G}_c$ with N_c columns. The squares are check nodes and the circles are code bits. In factor graph theory, check nodes are generalized as factor nodes and code bits are generalized as variable nodes.

3.1.3 Mapping

The output of the Raptor encoder, codeword $\mathbf{c} = \mathbf{c}_p \cdot \mathbf{G}_c = \mathbf{m} \cdot \mathbf{G}_p \cdot \mathbf{G}_c$ is passed through a BPSK mapper to form the transmitted symbol vector $\mathbf{s}$ including N_s symbols.

3.1.4 AWGN Channels

The transmitted symbol vector $\mathbf{s}$ is corrupted by discrete-time white Gaussian noise sequence $\mathbf{n}$ to form the received sample vector $\mathbf{r} = \mathbf{s} + \mathbf{n}$.

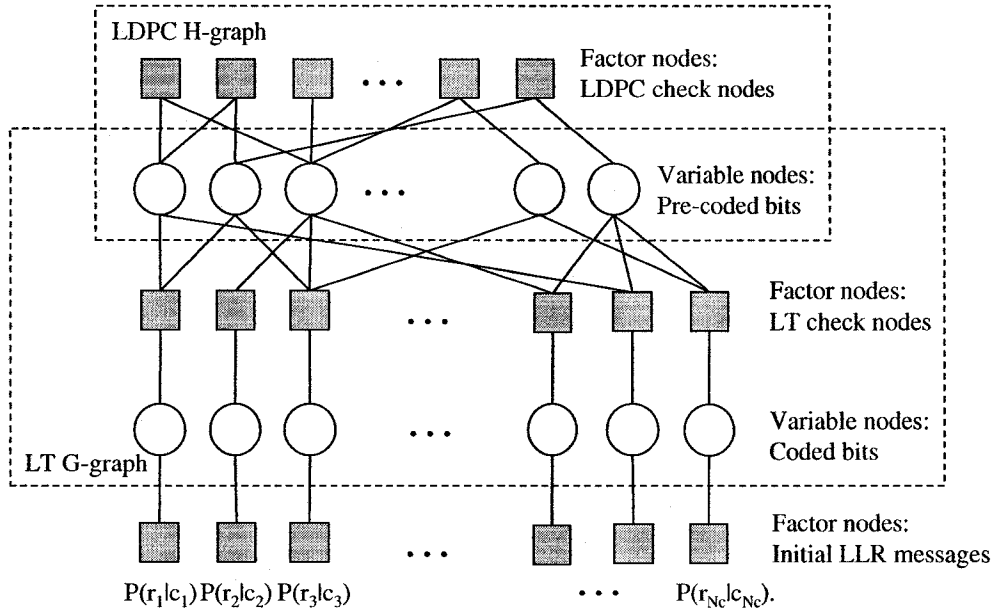

Fig. 3.2: Factor graph of Raptor codes truncated to N_c code bits.

3.1.5 Decoding

Iterative decoding of Raptor codes is also based on the BP algorithm working in the log domain, as we discussed in Section 2.4. Observing Fig. 3.2, we find two subgraphs in $\mathcal{G}$: one is LDPC H-graph and another is LT G-graph. They can be viewed as one combined factor graph where the BP decoder can be applied. For each decoding attempt, the outputs of LLR calculator are used to initialize N_c variable nodes, which represent N_c code bits. To ensure the decoding fairness between bit 0 and bit 1, zero messages are used to initialize the N_p variable nodes, which represent the N_p pre-code bits. In every iteration, each variable node exchanges messages with its neighboring factor nodes and vice versa, and then the decoder verifies the estimated message word $\hat{\mathbf{m}}$ by calculating the CRC. If the CRC is correct, the receiver informs the transmitter to transmit the next message word by sending back an acknowledgment. If the CRC is incorrect, iterative decoding continues until L iterations have elapsed. By then, the decoder will collect T_s more symbols for the next decoding attempt. In the beginning

of each decoding attempt, the decoder discards those values of variable nodes and factor nodes in the last decoding attempt and starts from the scratch to ensure fairness between received code bits. In such a way, the best decoding performance could be obtained by setting $T_s = 1$ and $L = +\infty$, but this is impractical due to high computational costs and the need for very fast processing speeds to avoid additional delay.

Raptor codes have been shown to uniformly approach the binary-input AWGN channel capacity, especially at low SNR's [47,48]. However, Raptor codes have fixed-rate pre-coding and are typically nonsystematic, so there is a throughput cap at high SNR's. This drawback motivates researches on another family of rateless codes: rate-compatible LDPC codes.

3.2 Rate-compatible LDPC Codes

Another promising rateless coding technique involves rate-compatible low-density parity-check (RC-LDPC) codes. Unlike LT and Raptor codes, which are defined in terms of their generator matrices, a RC-LDPC code is defined in terms of an ever-expanding parity check matrix. Applied with BP decoding on the obtained sparse parity check matrices, RC-LDPC codes could have linear decoding complexity.

To ensure linear encoding complexity, each $\mathbf{H}$ of a family of RC-LDPC $\mathbf{H}$ matrices is designed with lower-triangular form. To maintain the IR property while expanding, a new $\mathbf{H}$ should not change the local functions of previous factor nodes (i.e. check nodes). The simplest method to satisfy the two requirements is starting with a mother code with lower-triangular $\mathbf{H}$ matrix, $\mathbf{H}_{mth}$, and appending one check and one new code bit at a time, without changing the variable lists of previous check nodes. As pointed out in [49], this method has an inherent trade-off: the row weight of $\mathbf{H}_{mth}$ has

to be large to ensure good performance at high SNR's, whereas large row weight for new check nodes causes short cycles that will reduce the throughput at low SNR's. Therefore, the first RC-LDPC code was designed with puncturing and extending [50]. First, we choose a fixed-rate LDPC code as the mother code. Then we extend the parity check matrix of the mother code, $\mathbf{H}_{mth}$, to achieve lower rates and puncture $\mathbf{H}_{mth}$ to achieve higher rates. When the BP decoder attempts to decode the RC-LDPC code at rates higher than the mother code rate, all punctured code bits in $\mathbf{H}$ are treated as erasures and will be initialized as zero messages (for LLR-BP). The RC-LDPC code in [50] has $N_m = 1024$ and the rate ranges from 8/11 to 8/20. The average realized rate can approach within 1 dB of the binary-input AWGN capacity.

The following researches on RC-LDPC codes are then focused on how to judiciously puncture and extend the mother code for short block codes [49, 51–53]. When a BP decoder is used, we know that puncturing nodes with high d_s propagates more erasures in $\mathcal{G}$ and therefore paralyzes more check nodes. So a good choice for irregular RC-LDPC codes is to puncture degree-2 code bits [49]. We also know if two variable nodes of one check node are punctured, all outgoing messages from the check node are zero. Therefore, we have one more puncturing guide: at most puncturing one neighboring variable node for each check node. This guide was generalized as the concept of recoverable punctured nodes in [54]. (The idea first appeared in [55], but the discussion therein is rather abstract.) To make this thesis more self-contained, we repeat the definition on recoverable nodes in [54] as follows.

If a punctured variable node v_i has at least one neighboring check node whose neighboring variable nodes are all unpunctured except for v_i, v_i is called *1-step recoverable* (1-SR) node. The name comes from the fact that over an erasure channel, v_i will be recovered after the first iteration. A 2-SR node v_j has at least one neighboring

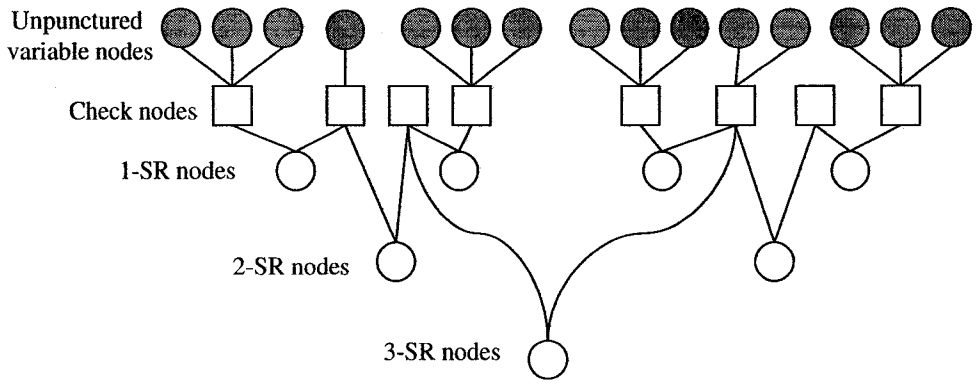

Fig. 3.3: Examples of k-SR nodes.

check node f_j whose neighboring variable nodes are all unpunctured or 1-SR nodes, except for v_j. Since the f_j has at least 1-SR node, the v_j can be recovered after exactly two iterations. If we call unpunctured variable nodes as 0-SR nodes, a k-SR node v_k can be defined as a punctured node that has at least one neighboring check node f_j connected to at least one $(k-1)$-SR node, v_k, and m-SR nodes where $0 \leq m \leq k-1$. Fig. 3.3 shows some examples of 1-SR, 2-SR and 3-SR nodes.

When we gradually puncture a mother code towards the lowest designed rate, lower-SR nodes are punctured first such that variable nodes can be recovered as early as possible during iterations. Therefore we hope the number of lower-SR nodes is as large as possible. When $\mathbf{H}$ has a lower-triangular form with dual-diagonal structure in the parity submatrix part,

$$\begin{bmatrix} 1 & & & & & & & \\ 1 & 1 & & & & & & \\ & 1 & 1 & & & & & \\ & & 1 & 1 & & & & \\ & & & \ddots & \ddots & & & \\ & & & & & 1 & 1 & \\ & & & & & & 1 & 1 \end{bmatrix},$$

we can have two nice properties: at least $\lceil N_j/2 \rceil$ 1-SR nodes, and no cycles composed of only degree-2 variable nodes. Irregular LDPC codes based on the above $\mathbf{H}$ is called

extended irregular repeat-accumulate (eIRA) codes [56]. Inspired by eIRA codes, Kim *et al* proposed a construction method of RC-LDPC codes to maximize the number of lower-SR nodes [57]. This code, termed efficiently-encodable rate-compatible (E^2RC) codes, have lower BER than eIRA codes from rate 0.5 to 0.9. The difference increases at higher puncturing rates: E^2RC codes have around 1 dB gain over eIRA codes when BER=10^{-5} and $r = 0.9$.

3.3 Rate-compatible IRA Codes

In 2006, Good and Kschischang proposed a rate-compatible irregular repeat-accumulate [58] (RC-IRA) code, with throughput close to capacity across a wide SNR range for large block size [5]. Check splitting (CS) is used to extend the parity check matrix of a RC-IRA code. To generate a new code bit, an old check node is split into two check nodes and the new code bit is used to join these two check nodes together. The information variable nodes originally joined to the old check node are randomly split into two equally-sized groups. Each group is allocated to one new check node. The major advantage of using CS for IRA codes is that the information variable degree distribution remains fixed while check splitting. As such, RC-IRA codes show very good throughput performance through a wide SNR range. Another advantage of IRA codes is its linear encoding complexity. Note that RC-IRA codes are systematic, since a nonsystematic IRA code is useless for a BP decoder [58].

3.3.1 Check Splitting

The CS method proposed in [5] can avoid the trade-off pointed out in [49]: the row weight of $\mathbf{H}_{mth}$ has to be large to ensure good performance at high SNR's, whereas large row weight for new check nodes causes short cycles that will reduce the throughput at low SNR's.

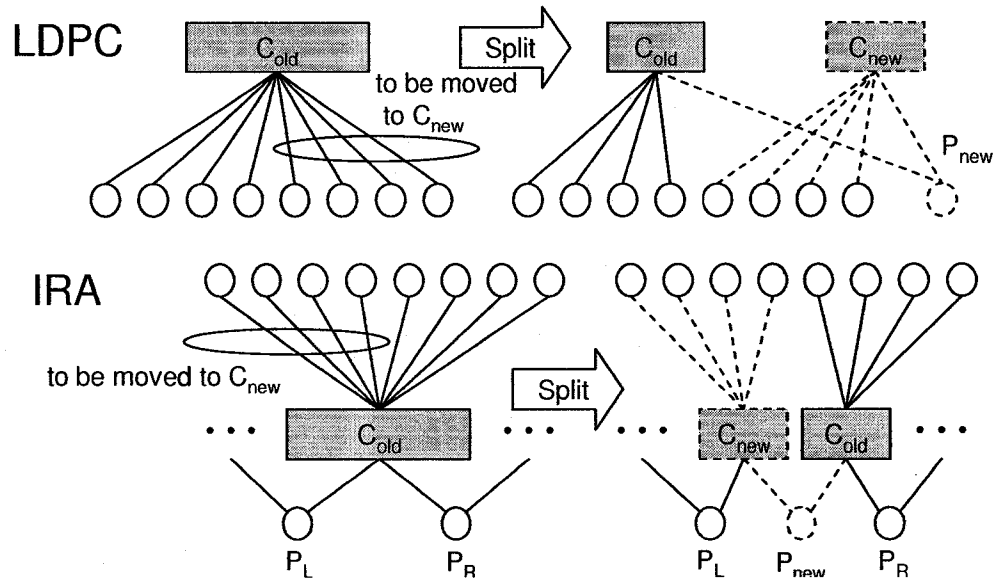

Fig. 3.4: Check-splitting for LDPC and IRA codes.

For a new check node, C_{new}, and a new parity bit, P_{new}, we split one previous check node, C_{old}, by moving half of the symbols that were connected with C_{old} to C_{new}. Then we connect both C_{old} and C_{new} with P_{new} such that the local function of C_{old} is not affected by the splitting. See Fig. 3.4 for an example of the above CS process. It can be observed that the CS for an IRA code is similar to the CS for an LDPC code.

We can also view the LDPC code example in terms of the $\mathbf{H}$ matrix. Given a lower-triangular $\mathbf{H}_{mth}$ with row weight 8,

$$\begin{bmatrix} . & . & . & . & . & . & . & . & . & . & . & 1 & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & \end{bmatrix},$$

we add one new check node with degree 5 and the $\mathbf{H}$ matrix becomes:

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

In CS, gradually reduced row weights of previous check nodes allow us to use lower row weights for new check nodes to avoid short-length cycles while maintaining near-uniform row weights for better throughput. Another advantage is that CS never decreases the lengths of existing cycles.

For rates higher than the mother code rate, we could combine check nodes to obtain the $\mathbf{H}$ matrices until the rate reaches almost 1, where there is only one huge check node summing up all copies of information symbols. Note this procedure is very similar to the row combining that was proposed for fixed block length to facilitate analog decoding [52, 53, 59].

For rates lower than the mother code rate, we keep splitting until the degree of each check node becomes 3, except the first check node in the accumulator. This $\mathbf{H}$ matrix, $\mathbf{H}_{deg3}$ usually corresponds to a small rate (around 0.15 if using the variable degree distribution in [5]). To have RC-IRA codes operate in a wider SNR range, we can continue to split $\mathbf{H}_{deg3}$. Starting from this point, CS is actually repeating parity symbols. In practice, it is convenient to set a *minimum rate*, r_{min}, for rateless codes. Let us denote as $\mathbf{H}_{dec}$ the parity-check matrix corresponding to r_{min}.

If PEG is used to generate $\mathbf{H}_{mth}$, it is important to randomize the list of neighboring variable nodes for each check node before proceeding to check splitting. Otherwise, simply cutting the list into half to perform CS introduces some deterministic structures due to PEG construction. Our simulations show that these structures greatly reduce the performance of RC-IRA codes.

3.3.2 Encoding

While splitting from the mother code, we add new rows but keep previous rows unchanged in the parity check matrix. This special matrix, $\mathbf{H}_{enc}$, could be used solely

for the encoding of RC-IRA codes. Since RC-IRA codes can be viewed as punctured codes from $\mathbf{H}_{dec}$, a simpler way to encode is to generate all code bits before hand based on $\mathbf{H}_{dec}$, which is already stored in the RC-IRA decoder for the convenience of hardware implementation.

3.3.3 Decoding

There is one challenge in the decoder design: we have to change the $\mathbf{H}$ matrix for each new code bit. Certainly, we could not afford to store all these matrices. But after we generated the whole $\mathbf{H}$ matrices family, we can store the $\mathbf{H}_{mth}$ and a split check node list (SC-list) including each split check node that corresponds to each newly added check node. Since for each newly added check node, we simply cut the split check node into half, therefore the decoder could use the $\mathbf{H}_{mth}$ and the SC-list to recover the whole $\mathbf{H}$ matrices family while decoding. The additional computations due to check splitting is much smaller than computations of one iteration for a BP decoder. However, this method is not convenient for hardware decoding because the receiver has to change the decoder for every T_s more symbols.

Another option is using a single decoder corresponding to $\mathbf{H}_{dec}$ [60], as in rate-compatible punctured codes (RCPC). In each decoding attempt, the decoder only activates the variable nodes in the $\mathbf{H}_{dec}$ graph when their corresponding code bits are transmitted.

The decoder can choose the BP algorithm with flooding (LDPC-like scheduling), or BP and BCJR algorithm with Turbo-like scheduling. The latter can increase the throughput of IRA codes at low SNR's [61,62]. We use a very simple systematic IRA code as an example to illustrate BP and BCJR algorithm with Turbo-like scheduling. The factor graph is shown in Fig. 3.5. The combined algorithm is described in five steps.

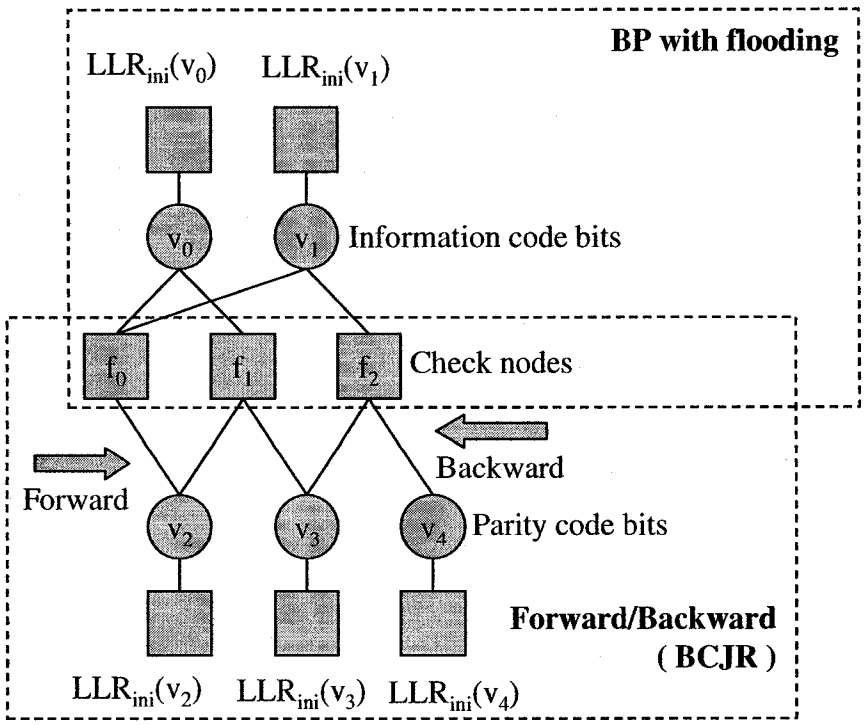

Fig. 3.5: BP and BCJR with Turbo-like scheduling.

Step 1. Initialization of BP

All information variable nodes $\{v_0, v_1\}$ pass LLR_{ini}'s to neighboring factor nodes.

Step 2. Forward algorithm

The leftmost factor node f_0 passes $\texttt{CHK}(v_0, v_1)$ to v_2; v_2 sums up $\text{LLR}_{ini}(v_2)$ and the incident message from f_0, and then sends the result to f_1; f_1 sends $\texttt{CHK}(v_0, v_2)$ to v_3; v_3 does the same thing as v_2; f_2 does the same thing as f_1. The forward algorithm ends at the rightmost factor node f_2.

Step 3. Backward algorithm

The rightmost parity variable node v_4 simply passes $\text{LLR}_{ini}(v_4)$ to f_2; f_2 passes $\texttt{CHK}(v_1, v_4)$ to v_3; v_3 sums up $\text{LLR}_{ini}(v_3)$ and the incident message from f_2, and then

sends the result to f_1; f_1 sends $\texttt{CHK}(v_0, v_3)$ to v_2; v_2 does the same thing as v_3. The backward algorithm ends at the leftmost parity variable node v_2.

Step 4. BP decoding

Step 4.1: All factor nodes $\{f_0, f_1, f_2\}$ generate outgoing messages to neighboring information variable nodes, using recently received messages during the iteration of foraward/backward algorithm.

Step 4.2: All information variable nodes $\{v_0, v_1\}$ generate outgoing messages to neighboring factor nodes for next iteration.

Step 5. Iterative processing

Repeat step 2,3,4 for each iteration until decoding success or $N_c^{(max)}$ is reached.

3.4 Rate-compatible QC-LDPC Codes

Punctured QC-LDPC codes are currently proposed as candidates of forward error correction (FEC) code in 3GPP2 (3rd Generation Partnership Project 2) [6] and 802.20 MBWA (Mobile Broadband Wireless Access) [7]. In 3GPP2, they are proposed as FEC code candidates in the forward data channel, forward superposed data channel, and reverse OFDMA data channel. These proposed codes are our main interest in this section. For simplicity, they will be referred to as rate-compatible QC-LDPC (RC/QC-LDPC) codes, unless otherwise indicated.

Degree profiles of RC/QC-LDPC codes proposed in [7], are obtained by using density evolution over multi-edge-type LDPC codes [13, chapter 7]. The mother code of RC/QC-LDPC, $\mathbf{H}_{mth}$ is a QC-LDPC code, which is constructed by replacing circulant permutation matrices into the m-ary base matrix $\mathbf{H}_B$. Since they are punctured

codes, $\mathbf{H}_{mth}$ and $\mathbf{H}_B$ correspond to the lowest designed rate $N_m/N_c^{(max)}$. The $(N_c^{(max)}$-$N_m)$-by-$N_c^{(max)}$ parity matrix $\mathbf{H}_{mth}$ is given by

$$\mathbf{H}_{mth} = \begin{bmatrix} \mathbf{H}'_{mth} & \mathbf{0} \\ \mathbf{H}'_{emth} & \mathbf{I} \end{bmatrix},$$

where $\mathbf{0}$ is an all-zero matrix and $\mathbf{I}$ is an identity matrix. The number of columns in $\mathbf{H}'_{mth}$ is larger than N_m and the exact value depends on the code design. Code bits corresponding to $\mathbf{I}$ have degree one and therefore are linearly encodable. Since they have smallest degree, they will be punctured first. So $\mathbf{H}_{mth}$ can be viewed as extended $\mathbf{H}'_{mth}$; $\mathbf{H}'_{emth}$ and $\mathbf{I}$ comprise the extended part. The puncturing rule of $\mathbf{H}'_{mth}$ follows from the k-SR node concept. The details can be found in [63].

Now let us examine the property of $\mathbf{H}'_{mth}$. Replacing $\mathbf{H}'_{mth}$ back to its base matrix form, $\mathbf{H}'_B$, and then replacing non-zeros in $\mathbf{H}'_B$ as 1 and zeros as 0, we have a binary base matrix $\mathbf{H}''_B$. Since the number of columns in $\mathbf{H}'_{mth}$ is larger than N_m, $\mathbf{H}''_B$ can be expressed as $[\mathbf{H}''_{BI}|\mathbf{H}''_{BP}]$, where $\mathbf{H}''_{BI}$ corresponds to the information part and $\mathbf{H}''_{BP}$ corresponds to the parity part. The required structure of $\mathbf{H}''_{BP}$ is given by

$$\begin{bmatrix} 1 & 1 & & & & & & \\ & 1 & 1 & & & & & \\ & & 1 & 1 & & & & \\ & & & 1 & 1 & & & \\ 1 & & & & \ddots & \ddots & & \\ & & & & & & 1 & 1 \\ 1 & & & & & & & 1 \end{bmatrix}.$$

It has been shown in [64] that when $\mathbf{H}''_B$ has the above dual-diagonal structure with one degree-3 parity bit, fast encoding based on [18] is possible. The factor graph of $\mathbf{H}''_B$ is shown in Fig. 3.6.

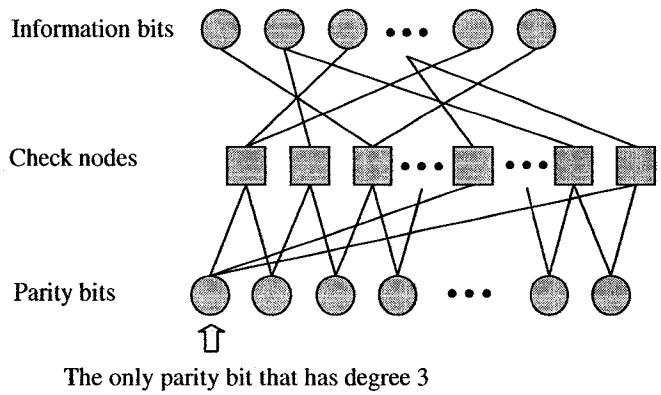

Fig. 3.6: The factor graph of the binary base matrix of QC-LDPC codes for fast encoding.

The m-ary counterpart of $\mathbf{H}''_{BP}$ is given by

$$\begin{bmatrix} m_{0,0} & 1 & & & & & & & \\ & m_{1,1} & 1 & & & & & & \\ & & m_{2,2} & 1 & & & & & \\ & & & m_{3,3} & 1 & & & & \\ y & & & & \ddots & \ddots & & & \\ & & & & & m_{n_j-2,n_j-2} & 1 & & \\ x & & & & & & & m_{n_j-1,n_j-1} & \end{bmatrix}.$$

If x,y,and $m_{i,i}$ $(0 \leq i \leq n_j - 1)$ satisfy certain relations, the encoding complexity becomes linear, regardless of the size of circulant permutation matrices. Based on this dual-diagonal structure, a procedure to construct good QC-LDPC codes was given in [64]. The main rules include using density evolution optimized degree profiles, minimizing the length of cycles in $\mathbf{H}''_B$ (called block-cycle), maximizing girth and minimizing the number of shortest cycles in $\mathbf{H}_{mth}$.

3.4.1 Encoding

The RC/QC-LDPC codes we are using are generated through the procedures in the 3GPP2 LDPC proposal [6]. In a RC/QC-LDPC encoder, $\mathbf{m}$ is first encoded into

$N_c^{(max)}$ code bits via the fully expanded $\mathbf{H}$, i.e. $\mathbf{H}_{mth}$, which is given by

$$\mathbf{H}_{mth} = \begin{bmatrix} \mathbf{H}'_{mth} & \mathbf{0} \\ \mathbf{H}'_{emth} & \mathbf{I} \end{bmatrix}.$$

The codeword $\mathbf{c}'$, orthogonal to the vector space spanned by $\mathbf{H}_{mth}$, can be split into two parts. The first part corresponds to columns in $\mathbf{H}'_{mth}$ and can be calculated via the method in [18]. The second part corresponds to the identity matrix $\mathbf{I}$ and can be easily calculated via back-substitution. After $\mathbf{c}'$ is obtained through $\mathbf{H}_{mth}$, the first few codes bits (termed state columns in [6]) are discarded and the remaining code bits are interleaved, resulting in the final codeword $\mathbf{c}$. Refer to [6, section 2.6.3.4] [7, Section 2.e] for details. Note that the "truncation" in [6, page 2-47] is related to the processing capability of mobile terminals, and has nothing to do with the code bit discarding we mentioned above.

3.4.2 Decoding

The erasure decoder used by RCPC can be used for RC/QC-LDPC codes. Zero-valued messages are used to initialize punctured code bits. Therefore, even when the $\mathbf{H}$ is smaller than $\mathbf{H}'_{mth}$, the decoder still uses $\mathbf{H}'_{mth}$, allowing lower-SR code bits to be recovered and to propagate messages to recover higher-SR code bits. Once $\mathbf{H}$ expands to be larger than $\mathbf{H}'_{mth}$, the decoder can only use $\mathbf{H}$ due to two facts: recovered degree-1 parity bits still pass zero-valued messages to check nodes; and we are not interested in hard decisions of parity bits, provided CRC's are embedded in message words.

Chapter 4

Performance Comparison over Binary-input AWGN Channel

The performance evaluation results presented in [5, 60] for RC-IRA codes are all for long message word (9500 bits), as are most of the published results for Raptor codes (eg. [4, 47, 48, 65]). In this chapter we explore and compare the performance of these codes with much shorter message word lengths. In the following sections, we first present the simulation model that will be used to compare the throughput of Raptor codes, RC-IRA codes, and RC/QC-LDPC codes at short block lengths. This is followed by simulation results and related discussions. In the last section, we discuss possible directions to improve the CS method and provide some preliminary results.

4.1 System and Simulation Model

The discrete-time system model, shown in Fig. 4.1, is similar to Fig. 3.1. Compared to the system model, our simulation model is slightly simplified. At the transmitter,

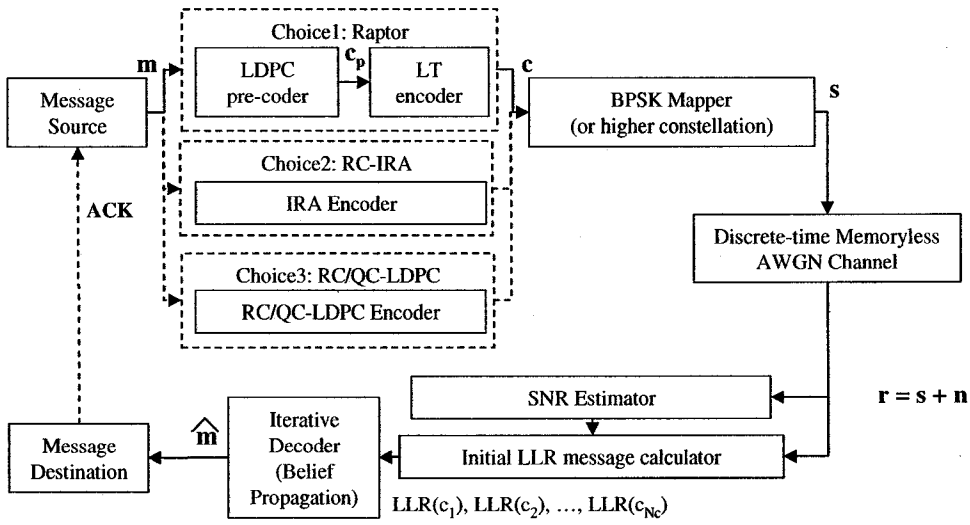

Fig. 4.1: System model.

the CRC is not embedded in the message word; at the receiver, the message word is taken as a known to simplify message verification, and effects of CRC errors are ignored, as is the corresponding reduction in capacity. Furthermore, it is assumed that the receiver has perfect knowledge of N_0.

Raptor codes

The method of generating $\mathbf{H}_p$ of the Raptor codes is the same as in [47,48]. The left degree of the LDPC H-graph, d_L, is fixed as 4, and the right degree is approximately a Poisson distribution as a result of uniformly choosing variable nodes (pre-code bits) for each check node. The pre-coding rate is 0.95.

For the LT encoder, we use the same check node degree distribution polynomial $\Omega(x)$ as in [47, 48]:

$$\begin{aligned}\Omega(x) = {} & 0.007969x + 0.493572x^2 + 0.166220x^3 \\ & + 0.072646x^4 + 0.082558x^5 + 0.056058x^8 \\ & + 0.037229x^9 + 0.055590x^{19} \\ & + 0.025023x^{65} + 0.003135x^{66}.\end{aligned} \tag{4.1}$$

The pre-code bits are uniformly chosen for each check node in the LT G-graph. Under short message word lengths, especially when $N_m = 188$, we need to pick Raptor code realizations such that the degree-1 check nodes in $\mathbf{G}_c$ are generated early to ensure good throughput of Raptor codes at high SNR's. When $N_m = 1528$ or $N_m = 9204$, the throughput is not very sensitive to the positions of degree-1 check nodes. In our simulations, $\mathbf{G}_c$ is not systematic; hence, we will still suffer some throughput loss at high SNR's, in exchange for better results at low-to-moderate SNR's.

We optimized $\mathbf{H}_p$ and $\mathbf{G}_c$ by trying to avoid length-4 cycles in the combined Raptor factor graph. However, under short message word length, i.e., $N_m = 188$ or $N_m = 1528$, there is not much room for optimization since the very high outer code rate of 0.95 will unavoidably incur many length-4 cycles in $\mathbf{H}_p$ alone.

RC-IRA codes

For the RC-IRA codes, the same variable degree distribution as in [5] is used to generate $\mathbf{H}_{mth}$:

$$\lambda(x) = \Sigma\lambda_i x^{i-1} = 0.0599x + 0.2411x^2 + 0.2653x^7 + 0.4337x^9, \tag{4.2}$$

where λ_i is the fraction of edges connected to variable nodes with degree i. We also choose the same mother code rate, 0.923. The reason of choosing this mother code rate is shown as follows. In the factor graph of a RC-IRA code excluding the accumulator part, the number of edges from factor nodes is equal to the number of edges incident to information bits. Thus, $d_s^{(avg)} N_m = d_c^{(avg)}(N_c - N_m)$ holds, where $d_s^{(avg)}$ is the average left degree and $d_c^{(avg)}$ is the average right degree. By $r = N_m/N_c$, we have $r = d_c^{(avg)}/(d_c^{(avg)} + d_s^{(avg)})$. Since check-concentrated RC-IRA codes are preferred, RC-IRA codes are designed as strictly-right-regular (SRR). And for the programming convenience of check splitting, $d_c^{(avg)}$ is preferred as 2^i, where i is an integer. Noticing that the above optimized $\lambda(x)$ gives us $d_s^{(avg)} = 5.33$, we have $r = 2^i/(2^i + 5.33)$. Plugging i=1, 2, 3, 4, 5, 6, we have $r = 0.27, 0.43, 0.60, 0.75, 0.857$ and 0.923, correspondingly. Note that choosing 0.923 as the mother code rate is only for programming convenience and the CS method also works for non-strictly-right-regular (NSRR) mother codes [60].

The PEG algorithm is used to generate $\mathbf{H}_{mth}$ of RC-IRA codes. We start from symbols with lower degree, allocating longer cycles to lower-degree symbols. Thus, a good EMD [37] distribution will naturally be obtained. Therefore, we did not incorporate the ACE algorithm into PEG to generate $\mathbf{H}_{mth}$.

To make a SRR $\mathbf{H}_{mth}$, in the near-end of PEG algorithm, we need switch the priority order in picking check nodes: check nodes with minimal d_c are picked first; farthest check nodes are then chosen. In the case of tie, one check node is randomly picked. Although SRR gives smaller girth than NSRR, as will be shown later, this does not make much difference in terms of throughput.

Table 4.1: Typical values of the stepsize T_s

T_s	$E_s/N_0(dB)$											
	-9	-8	-7	-6	-5	-4	-3	-2	-1	0-2	3	4-8
$N_m = 188$ or 192	24			12		4						2
$N_m = 1528$ or 1536	36			24		12						6
$N_m = 9204$ or 9500	276	228	180	156	120	108	84	48	36	24	12	6

RC/QC-LDPC codes

As mentioned before, a practical RC/QC-LDPC transmitter chops and interleaves the codeword before mapping. Thus, the decoder needs know which received sample corresponds to which code bit in **H**. For simplicity, we use all-zero message words (and therefore all-zero codewords) for simulations of RC/QC-LDPC codes, without chopping or interleaving. We also avoided the de-interleaving at the receiver, since the noise is white and the channel is memoryless. The validity of using all-zero codeword has been shown in [66]. Random message words are used for Raptor and RC-IRA codes.

Decoding

The BP algorithm with LDPC-like scheduling is used for the Raptor decoder and RC/QC-LDPC decoder. The BP and BCJR algorithms with Turbo-like scheduling is used for the RC-IRA decoder.

The maximum number of iterations per decoding attempt, L, is set to 80 for $N_m = 1528$ or 188, and 100 for $N_m = 9500$ or 9204. We let T_s, the number of additional received samples between consecutive decoding attempts, vary with SNR, with smaller T_s used for larger SNR's. Typical T_s values are shown in Table 4.1.

When $N_m = 9204$ or 9500, at least 1024 message words are simulated for each E_s/N_0. When $N_m = 1528$ at least 3000 words are simulated and at least 11000 words are simulated when $N_m = 188$. To ensure fair comparisons, the same message words

and noise sequences are used for all simulation scenarios. The average realized rate (ARR) is compared to the capacity of the binary-input AWGN channel to evaluate the throughput performance. Supposing N_w codewords are transmitted for one simulation scenario, the ARR is defined as

$$\text{ARR} = \frac{N_m}{\frac{1}{N_w}\sum_{i=1}^{N_w} N_s(i)}, \tag{4.3}$$

where $N_s(i)$ denotes the number of transmitted symbols before the successful decoding of the i^{th} codeword.

For simulation efficiency, the maximal number of code bits, $N_c^{(max)}$ is set. After receiving $N_c^{(max)}$ code bits, if $\hat{\mathbf{m}} \neq \mathbf{m}$, the decoder declares a decoding failure. A larger $N_c^{(max)}$ gives us a smaller lowest rate r_{min}, allowing a wider SNR range in simulations. For RC-IRA codes, we use $r_{min} = 0.022,\ 0.085,\ 0.15$ for short block length ($N_m \approx 188$), moderate block length ($N_m \approx 1528$), and large block length ($N_m \approx 9204$), respectively. For Raptor codes, we use lower rates than RC-IRA codes in corresponding block lengths. RC/QC-LDPC codes proposed in [6] only support $r_{min} = 0.2$ for any block length, limiting the lowest E_s/N_0 in its simulations to around -2 dB.

When $N_m = 9204$ or 9500, we use a technique called adaptive decoding start (ADS) to speed up simulations, especially at low SNR's where the gap between the throughput and the capacity is relatively large. First we attempt decoding at the Shannon capacity point. After N_{ADS} blocks are simulated, we calculate the average number of required symbols for successful decoding, denoted as $N_s^{(avg)}$. The next codeword then uses $N_s^{(avg)} - 2T_s$ or $N_s^{(avg)} - 3T_s$ as the new decoding start point, indicating the start of the ADS stage. If the first decoding attempt succeeds, we search

N_s backwardly until a decoding failure. The N_s corresponding to the last successful decoding attempt is recorded for calculating the realized rate of this codeword. If the first decoding attempt fails, the decoder searches N_s forwardly as the original decoding does. Once the ADS stage starts, we can update $N_s^{(avg)}$ every N_{ADS} codewords. Note that ADS is purely for simulation purpose and will provide a worse throughput than starting decoding at the Shannon capacity point. For example, decoding may succeed at 20000 code bits, but fail at 20100 code bits and succeed again at 20200 code bits. If the first decoding point happens to be 20100, then we will do forward search to end at 20200 code bits. If first decoding is 20200, then backward search will stop at 20100 and use 20200 to calculate average realized rate. Therefore, the simulation result using ADS is a lower bound of one without ADS. Our simulation results show that when $N_m = 9204$ or 9500, this bound is very tight - for a Raptor code over a rate 1/2 AWGN channel, ARR without ADS is 0.45958 and ARR with ADS is 0.45956. However, when $N_m = 1500$ or 188, the bound is loose and we choose not to use ADS.

4.2 Simulation Results

Using the Monte-Carlo method, we compared the throughput of Raptor codes, RC-IRA codes, and RC/QC-LDPC codes in the E_s/N_0 range from -9 dB to 8 dB. Three different message word size are simulated for Raptor codes: $N_m = 188, N_m = 1528, N_m = 9500$. For RC-IRA codes, we also used $N_m = 188$ and $N_m = 1528$, but used $N_m = 9204$ instead of $N_m = 9500$. For RC/QC-LDPC codes, we set N_m as proper values to avoid zero-padding: $N_m = 192$ and $N_m = 1536$. The mapping is BPSK.

4.2.1 Throughput

Fig. 4.2, 4.3 and 4.4 show the system throughput as a function of E_s/N_0, the transmitted energy per symbol (E_s) over N_0. When $N_m = 188$, the gap between the throughput of RC-IRA codes and the capacity is no more than 1 dB, if the SNR is higher than -2 dB. This is amazing for such a short message word length. When $N_m = 1528$, the gap reduces to around 0.5 dB for channels with a capacity higher than 0.5. For all three message word size, we observe that RC-IRA codes outperform Raptor codes when E_s/N_0 is larger than about -3 dB. This advantage of RC-IRA codes over Raptor codes becomes larger when N_m becomes smaller. To explain why Raptor codes perform worse in high capacity channels, we notice that the LT G-graph does not cover all pre-code bits in a timely manner. A deep analysis can be found in [67]. Using the systematic Raptor codes proposed in [65] might mitigate this capacity loss somewhat, but the throughput cap of the LDPC code used in the Raptor code is the primary factor limiting the performance. Using a higher-rate LDPC pre-code would help, but finding such a higher rate code is a challenge because of the large number of short cycles.

It can be observed that RC/QC-LDPC codes also suffer the throughput cap at higher SNR's because the large number of punctured code bits paralyzes the message passing. This is a common drawback for RCPC's with a low-rate mother code. In the context of 802.20 and 3GPP2, however, the throughput cap has little effects on the performance of RC/QC-LDPC codes. Fixed physical frames require that the codeword $\mathbf{c}$ encoded from $\mathbf{H}_{mth}$ should be split into equal portions for transmission. In [7], the first transmission of $\mathbf{c}$ is large enough to ensure punctured code bits can be recovered; the penalty is a lower maximum realized rate.

In very low capacity channels, RC-IRA codes are slightly worse than Raptor codes.

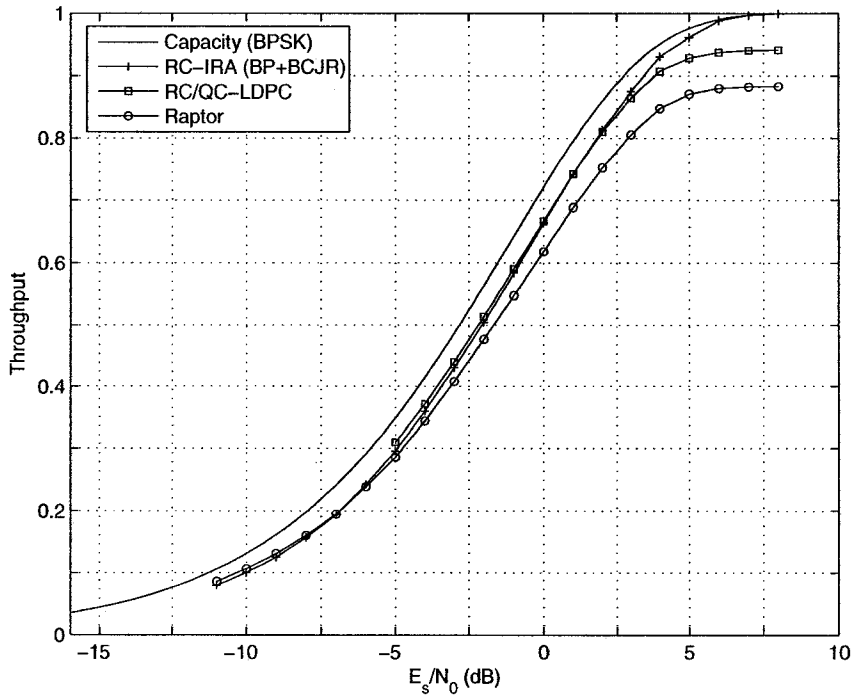

Fig. 4.2: Throughput vs. E_s/N_0 ($N_m = 188$).

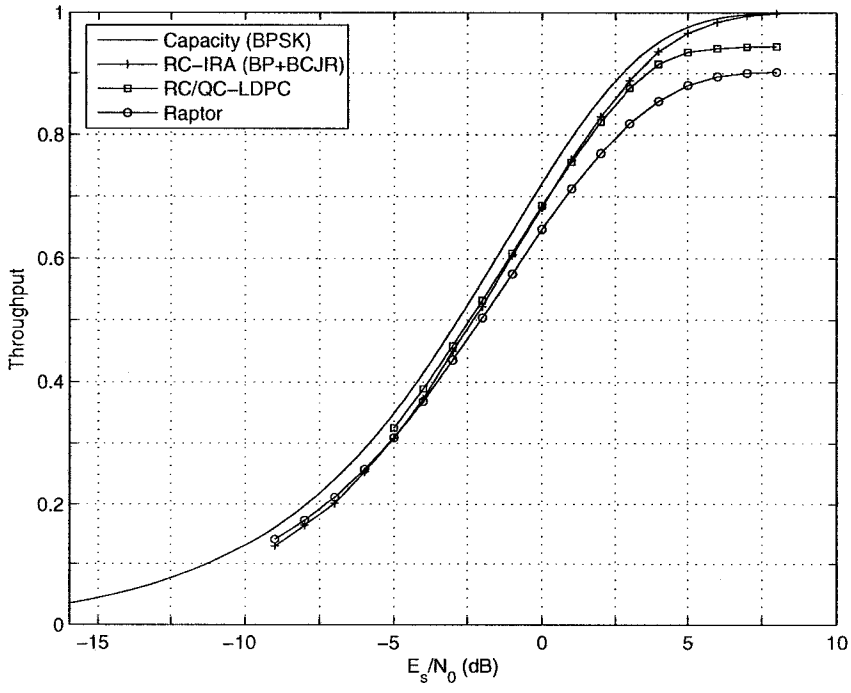

Fig. 4.3: Throughput vs. E_s/N_0 ($N_m = 1528$).

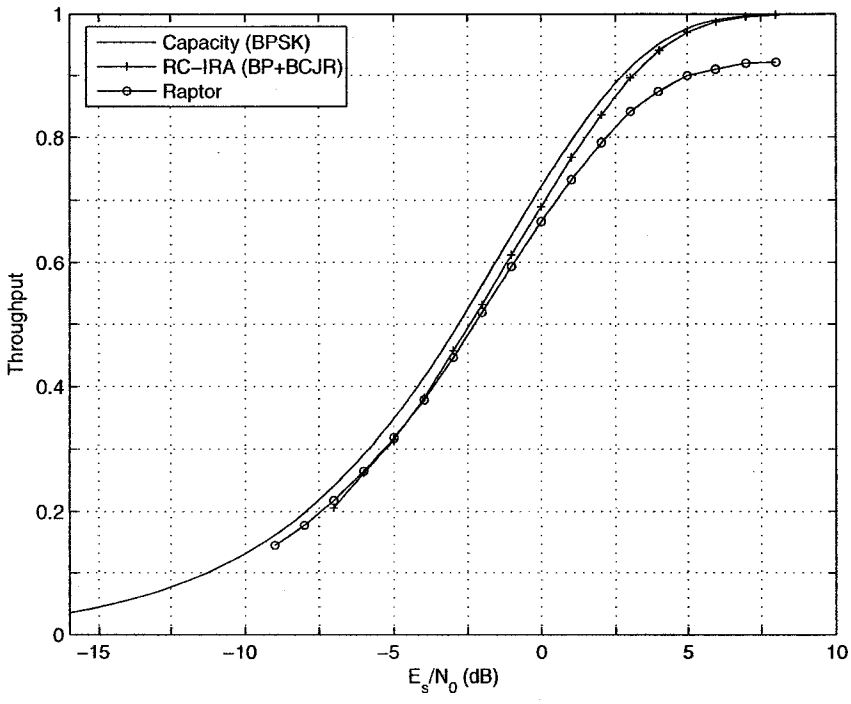

Fig. 4.4: Throughput vs. E_s/N_0 ($N_m \approx 9500$).

One reason is that $\mathbf{H}$ becomes too sparse when the rate is low [60]. One may notice an $\mathbf{H}$ matrix of a RC-IRA code has a sparser parity part than the information part during check splitting. When the rate becomes low, making use of the parity part should provide better throughput. In that case, it is not an IRA code any more, but an LDPC code. Possible solutions were discussed in [60].

In Fig. 4.2 and 4.3, the ARR of RC-IRA codes and RC/QC-LDPC codes cross at around 0.5 dB (ARR $\approx$ 0.75) for short block length and at around 0.2 dB (ARR $\approx$ 0.72) for moderate block length. RC/QC-LDPC codes perform better than RC-IRA codes at lower SNR's because the former has less constraints on designing $\mathbf{H}_{mth}$ and deterministic construction does a better job for LDPC codes with short to moderate block length than pseudo-random construction (PEG plus randomly picking check node candidates). RC-IRA codes need to adhere to the particular dual-diagonal structure in $\mathbf{H}_{mth}$, while RC/QC-LDPC codes have a raised dual-diagonal structure

in $\mathbf{H}'_{mth}$ and therefore have less constraints on placing 1's in $\mathbf{H}'_{mth}$. From the perspective of encoding complexity, the performance gain of RC/QC-LDPC codes at lower SNR's is also understandable since RC/QC-LDPC codes have slightly more complex encoder structure than RC-IRA codes.

From the perspective of $\mathbf{H}$ sparsity, RC/QC-LDPC codes are sparser than RC-IRA codes at higher SNR's and denser at lower SNR's. In well-designed LDPC codes, usually denser $\mathbf{H}$ leads to better error-correcting capabilities. After plotting the number of edges and average left degree $d_s^{(avg)}$ against the number of code bits N_c in Fig. 4.5 and 4.6, we can easily see that the number of edges in RC-IRA codes increases slower than RC/QC-LDPC codes. Compared to RC-IRA codes, the $d_s^{(avg)}$ of RC/QC-LDPC codes is higher at lower SNR's and lower at higher SNR's. This agrees with the throughput difference we observed.

4.3 Variants of RC-IRA Codes

We demonstrated that under different message word size, RC-IRA codes using the CS method outperform Raptor codes and RC/QC-LDPC codes in high capacity channels, but the performance is still worse for low-capacity channels. In this section, we try to improve the throughput of RC-IRA codes in low capacity channels. Several variants of RC-IRA codes are introduced and their throughput in the binary-input AWGN channel are presented.

4.3.1 Selected Check Splitting

In Section 3.3.1, we mentioned CS never decreases the lengths of existing cycles. However, we could selectively split symbols connected to previous check nodes to increase the length of short cycles, especially those short length cycles composed of

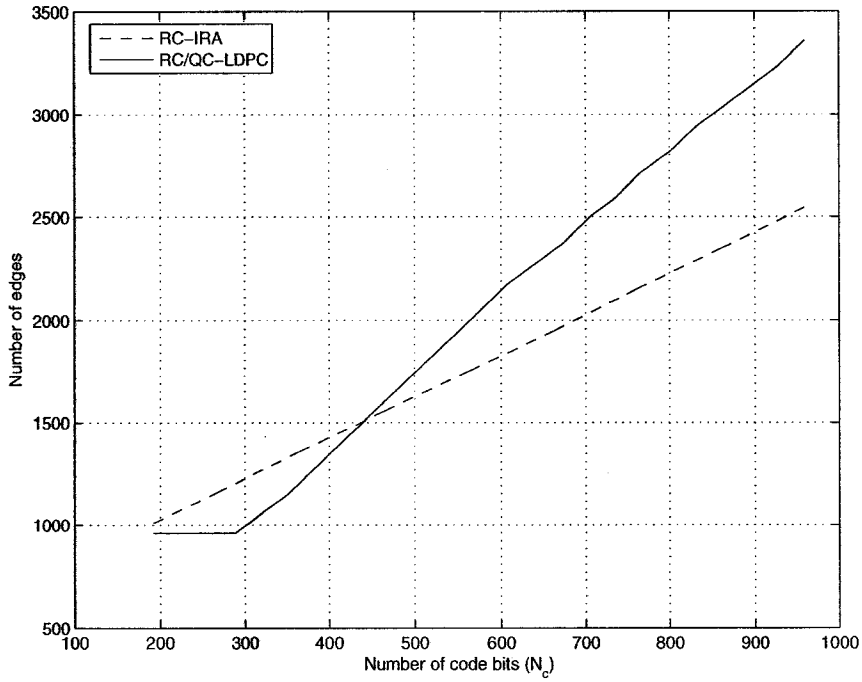

Fig. 4.5: The number of edges vs. N_c ($N_m = 188$ or 192).

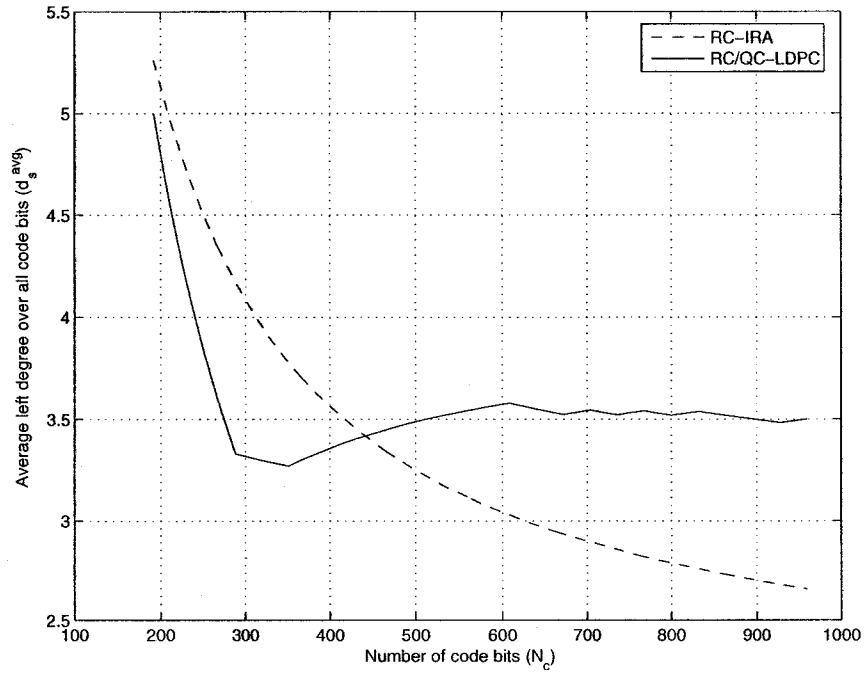

Fig. 4.6: Average left degree over all code bits ($d_s^{(avg)}$) vs. N_c ($N_m = 188$ or 192).

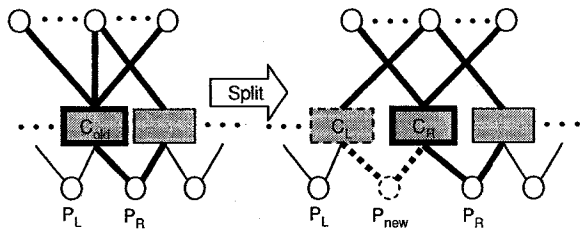

Fig. 4.7: SCS for IRA codes.

only low degree symbols such as degree-2 and degree-3 symbols (we call these cycles "bad cycles"). By extending "bad cycles", we could increase the minimum Hamming distance of the code. The proof can be found in [37].

We use a RC-IRA code as an example to show a heuristic way of selective check splitting (SCS). Our question, how to selectively split a check node, could be described as how to equally divide the information symbols connected with this check node into two groups. When splitting a given check node, C_{old}, into two check nodes, C_L and C_R, we refer to the two parity variable nodes, P_L and P_R, connected to C_{old} as the source nodes, and all degree-2 information variable nodes as the destination nodes. Using a breadth-first search algorithm, we find the distance between each source and destination node. To extend "bad cycles", by making use of the distance information, we try to connect those destination nodes closer to P_L with C_R and those destination nodes closer to P_R with C_L. As shown in Fig. 4.7, an information symbol is put far away from P_R so that we extend a length-4 cycle to a length-6 one. We then repeat this with all degree-3 variable nodes as the destination nodes. Note that since our interest is extending "bad cycles", the breadth-first search should be performed under the constraint that variable nodes on the path can only have degree 2 or degree 3 and cannot pass through C_{old}.

To illustrate the effects of SCS on the error floor of the code, we observed the word error rate (WER) as a function of the number of received code bits, as shown

in Fig. 4.8 and 4.9. We found that the CS method has high error floors while the SCS method does not have noticeable ones. The reason of the error floors in the CS method is the length-4 and length-6 "bad cycles", which we could observe from the denoted length distributions of "bad cycles" shown in Fig. 4.8 and 4.9. For example, after 600 code bits have been produced, the CS method gives two cycles at length 4 and eleven at length 6, whereas the SCS method gives a minimum cycle length of 10. This phenomenon agrees with the fact that the ACE algorithm could reduce the error floor [37]. Note that after receiving more code bits, the minimum Hamming distance of rateless codes will increase and the error floor will gradually drop.

Although SCS does lower the error-floor, under all three message word size, our simulation results on throughput do not show any noticeable advantage for the SCS method over the CS method. This phenomenon is likely due to the fact that reducing a low WER (such as 10^{-3}) to an even lower one (such as 10^{-5}) will have negligible impact on throughput. This suggests that methods to improve the throughput of RC-IRA codes should not focus on the error floor or even the cycle length distribution of the graph, but more on how to reduce the WER when only a small number of code bits have been transmitted.

4.3.2 Choices of Mother Code and Decoding Scheduling

To increase the length of "bad cycles", we can also design the mother code at a low rate with the help of the PEG algorithm. In fact, it has been shown in [5] that for $N_m = 9500$, there is no noticeable difference in the throughput between the mother code rate 0.75 and 0.923. To further investigate the effects of the mother code rate for small and moderate block length, we generated RC-IRA codes with different N_m (188 and 1528) and different mother code rates (0.15, 0.28, 0.75 and 0.923). In Fig. 4.10

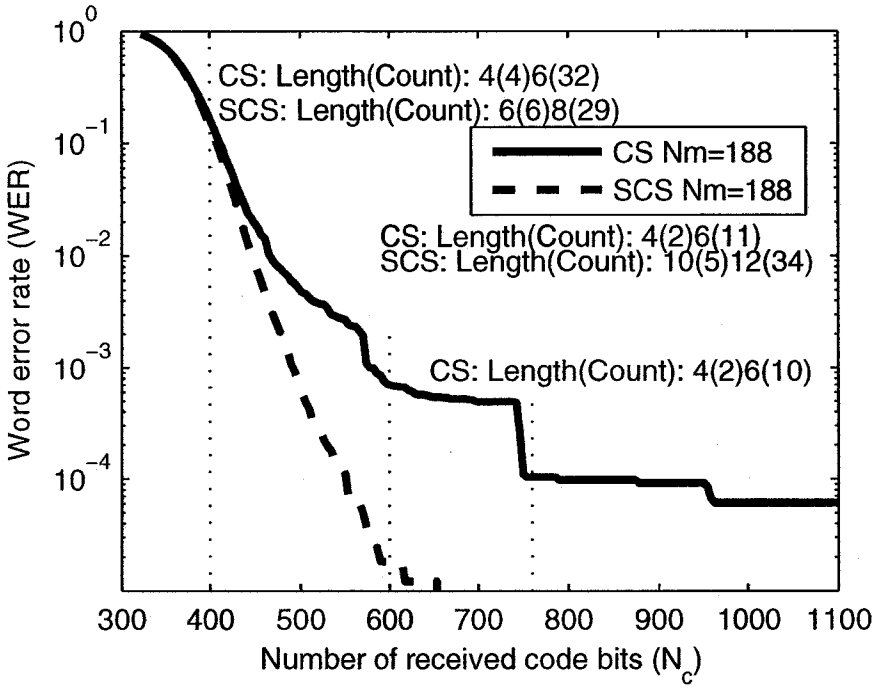

Fig. 4.8: WER vs. number of received code bits ($E_s/N_0 = -2$ dB).

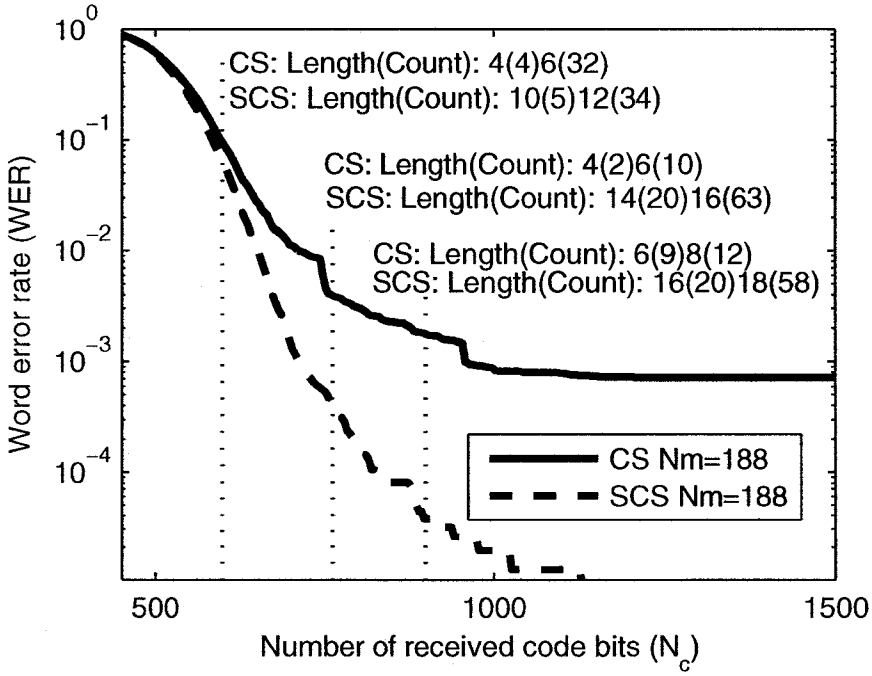

Fig. 4.9: WER vs. number of received code bits ($E_s/N_0 = -4$ dB).

and 4.11, no noticeable difference was found for these mother code rates.

To further investigate the difference of applying the BP and BCJR algorithms and applying only the BP algorithm, we compared the throughput between these two decoding options when $r_{mth} = 0.923$ and 0.28. As shown in Fig. 4.12 and 4.13, the results indicate that applying only the BP algorithm provides the similar throughput as using both the BP and BCJR algorithms, except a slightly worse throughput at low SNR's, which is due to the fact that the increased length of the accumulator slows down the message propagation between parity bits.

4.3.3 Parallel Edges

Note that combining check nodes will certainly generate double or even triple parallel edges, especially when the rate is close to 1. Better throughput at high SNR's should be expected by merging multiple edges coming from the same information symbol to the same check node into a single edge. Therefore, we also studied the effects of merging parallel edges. Fig. 4.14 and 4.15 show no difference in all scenarios, except a slight gain at high SNR's when $N_m = 188$. We can either use a high-rate mother code, or just keep those parallel edges since deleting parallel edges require changes on the decoding algorithm.

4.3.4 Right-regular Mother Codes

We also ran simulations to investigate the difference between using strictly right-regular mother codes and non-strictly right-regular ones. No discernable difference in performance was found, and so the results are not presented here.

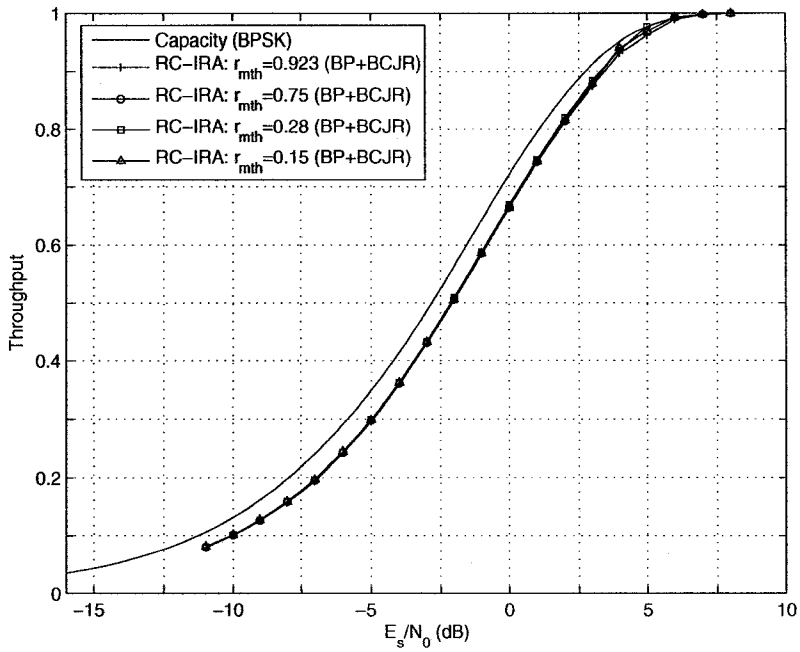

Fig. 4.10: Throughput vs. E_s/N_0 for RC-IRA codes ($N_m = 188$).

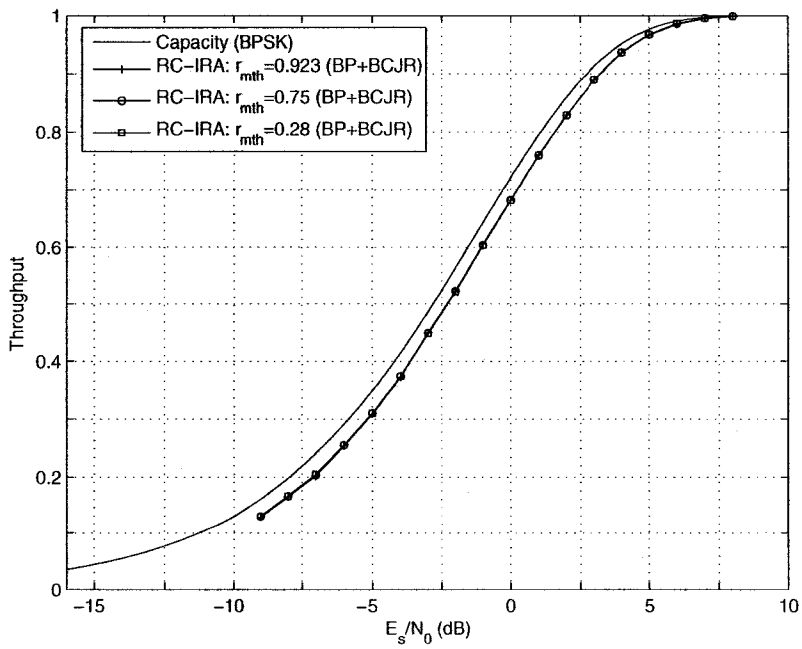

Fig. 4.11: Throughput vs. E_s/N_0 for RC-IRA codes ($N_m = 1528$).

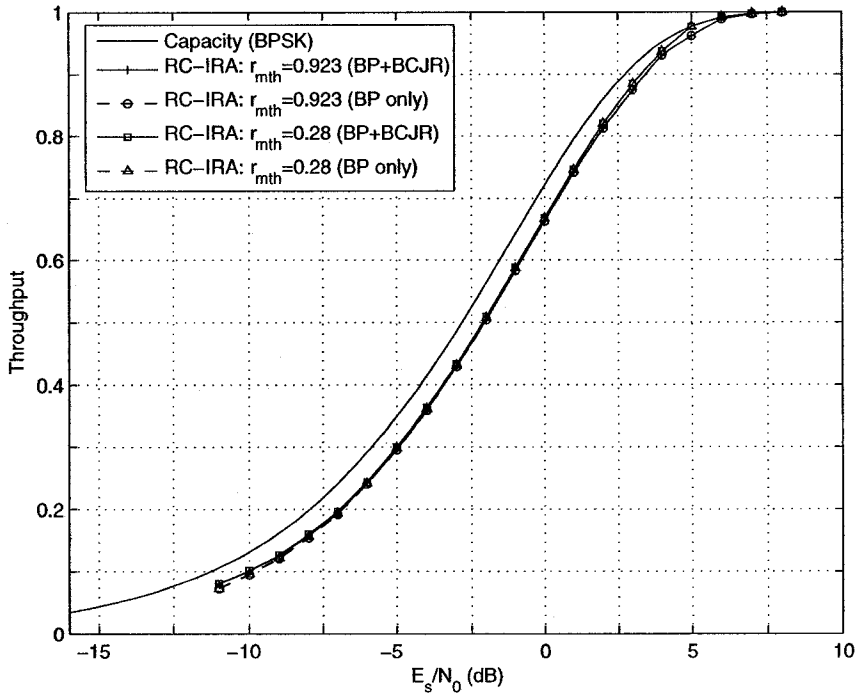

Fig. 4.12: Throughput vs. E_s/N_0 for RC-IRA codes ($N_m = 188$).

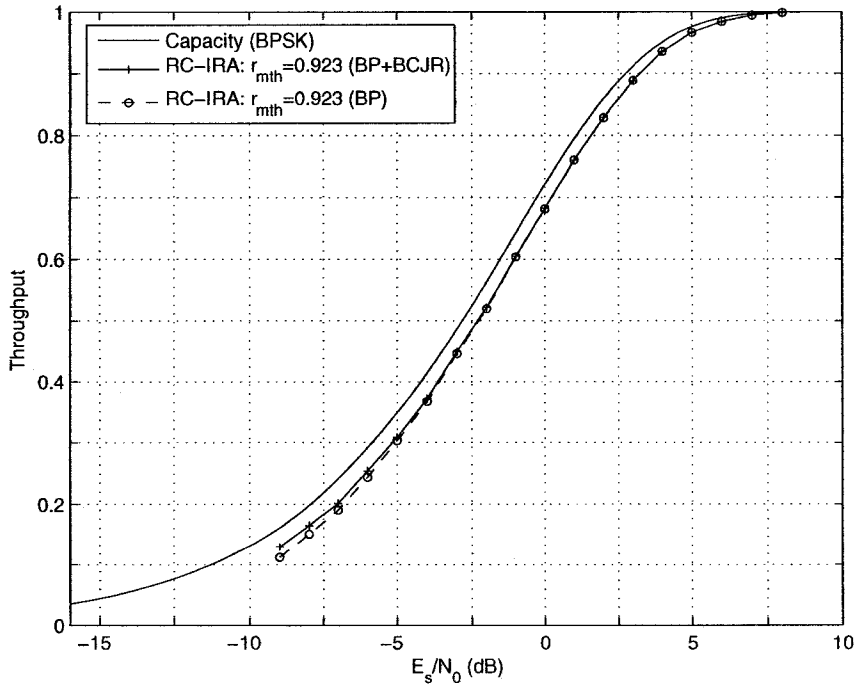

Fig. 4.13: Throughput vs. E_s/N_0 for RC-IRA codes ($N_m = 1528$).

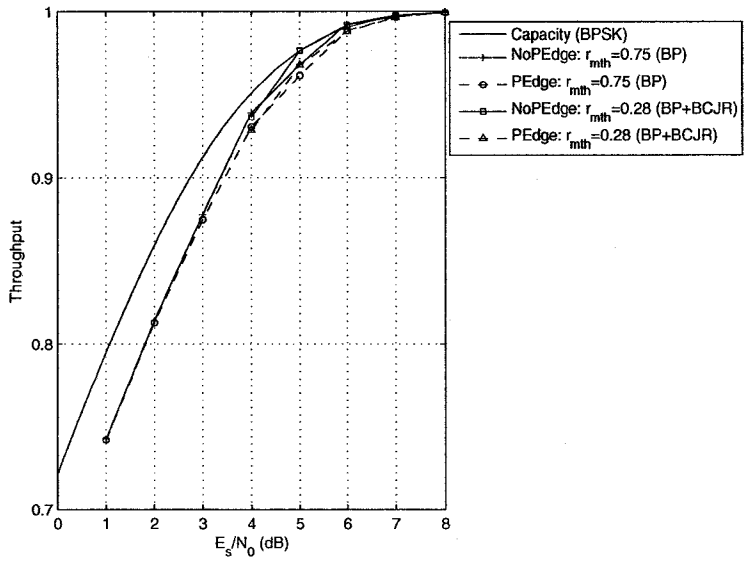

Fig. 4.14: Throughput vs. E_s/N_0 for RC-IRA codes ($N_m = 188$). NoPEdge denotes no parallel edges. PEdge denotes with parallel edges.

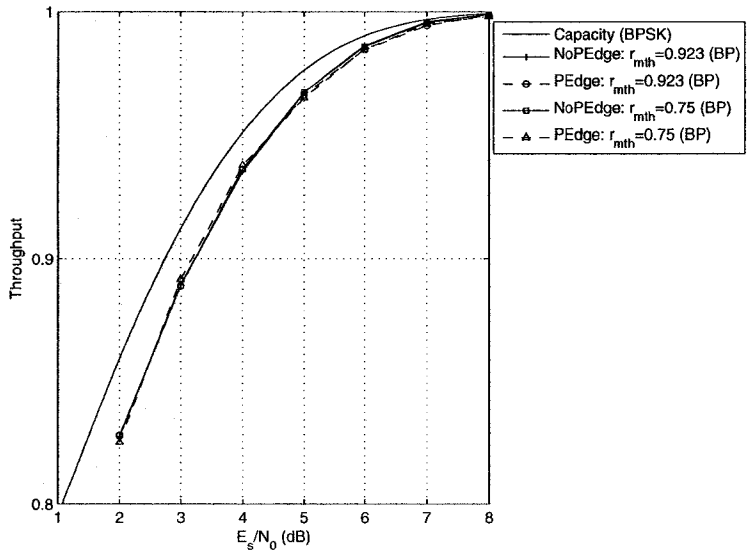

Fig. 4.15: Throughput vs. E_s/N_0 for RC-IRA codes ($N_m = 1528$). NoPEdge denotes no parallel edges. PEdge denotes with parallel edges.

4.3.5 Variable Degree Distributions

To investigate the effects of variable degree distributions, we picked some optimized LDPC variable degree distributions with different average left degree (2.06, 2.67, 3, 3.09, 3.71, 4.67, 5.33, 5.62) from Urbanke's LDPC optimizer [68]. By viewing IRA codes as LDPC codes, we can translate these distributions into the information variable degree distributions of IRA codes. Setting N_m around 188 and using a decoder with the BP and BCJR algorithm, we found, as expected, distributions with lower average left degree provide worse throughput, as shown in Fig. 4.16. Surprisingly, when we use four arbitrary variable degree distributions with the average left degree 5.33, as used in the optimized distribution, there is no noticeable difference in the throughput except one code with concentrated variable degrees: $\lambda(x) = \lambda_5 x^4 + \lambda_6 x^5$, as shown in Fig. 4.17. Variable-concentrated codes are worse due to their similarity to regular LDPC codes; we know irregular LDPC codes perform better than regular ones. For non-optimized codes in Fig. 4.17, we only use two variable degrees such that $d_s^{(avg)} = 5.33$ alone can determine exact distributions. For moderate block lengths, we have similar results shown in Fig. 4.18.

In this section, we discussed some possible improvements on RC-IRA codes and devised some variants of RC-IRA codes. Simulation results on these variants suggest that RC-IRA codes are very robust codes capable of providing consistent throughput performance, regardless of the mother code rate, the BP algorithm scheduling, the existence of parallel edges and the variable degree distribution (as long as the average left degree is not too low).

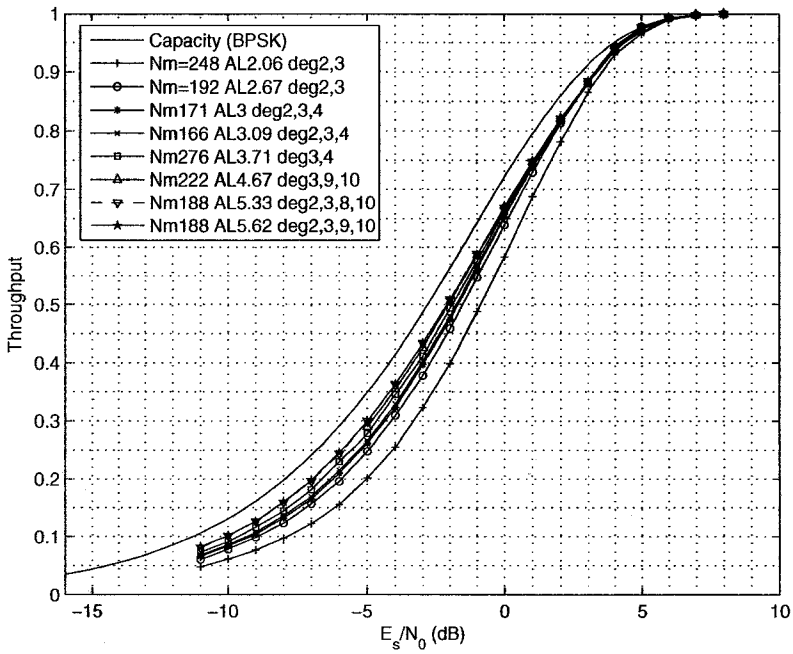

Fig. 4.16: Throughput vs. E_s/N_0 for RC-IRA codes. Nm188 AL5.62 deg2,3,9,10 denotes RC-IRA codes with $N_m = 188, d_s^{(avg)} = 5.62, \lambda(x) = \lambda_2 x + \lambda_3 x^2 + \lambda_9 x^8 + \lambda_{10} x^9$. Other curves follow the same rule.

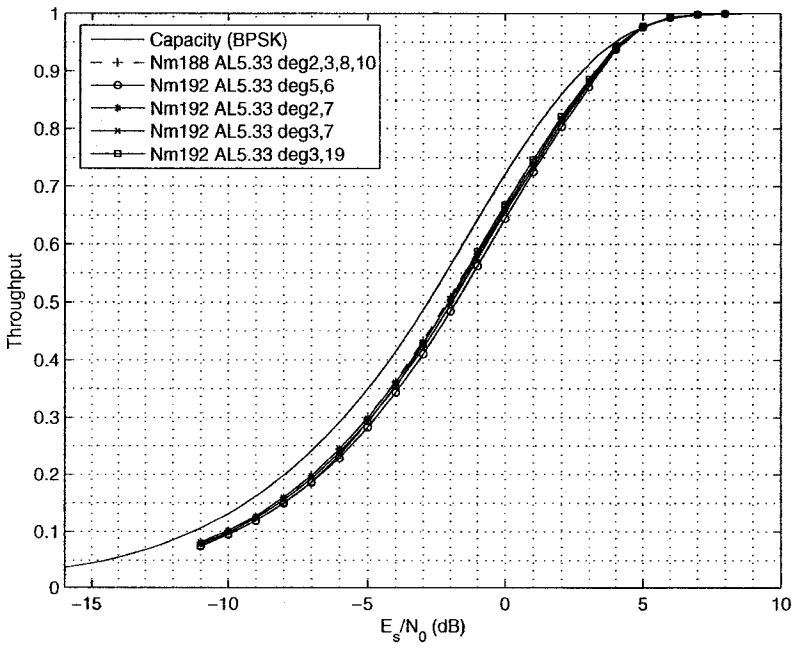

Fig. 4.17: Throughput vs. E_s/N_0 for RC-IRA codes.

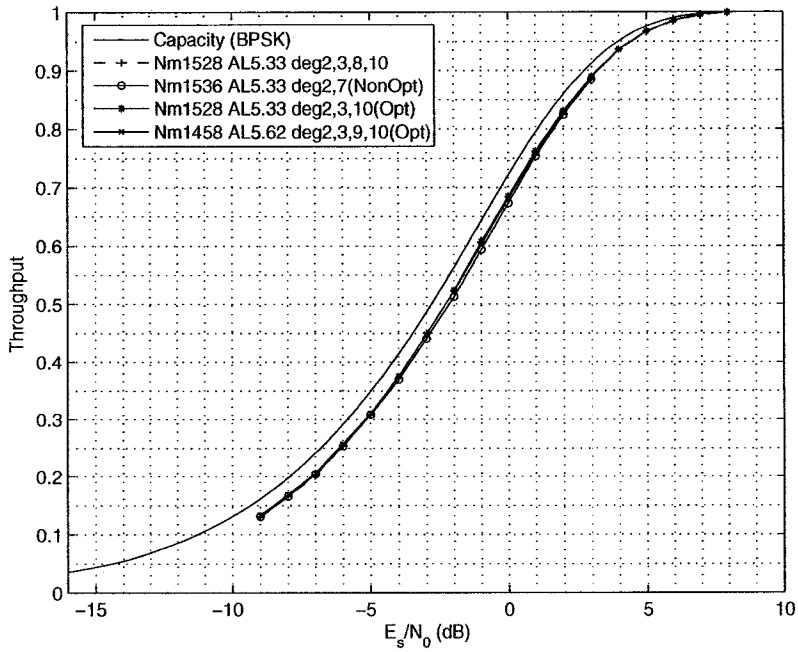

Fig. 4.18: Throughput vs. E_s/N_0 for RC-IRA codes. NonOpt: non-optimized variable degree distribution; Opt: optimized variable degree distribution.

4.4 Conclusions

We showed that both RC/QC-LDPC and Raptor codes have a throughput cap at high SNR's. RC-IRA codes can reach rate 1 but have slightly worse throughput at low SNR's than RC/QC-LDPC codes. We then focused on RC-IRA codes and studied possible directions to improve RC-IRA codes. All these discussions are based on the binary-input AWGN channel. In the next chapter, we extend the discussions to higher-order modulation over the AWGN channel.

Chapter 5

Performance Comparison for Spectrally Efficient Modulation over AWGN Channel

In the previous chapter, we compared three rateless codes over noisy channels, but the discussions are limited to BPSK modulation. As such, the potential of rateless codes has only been demonstrated for fairly low transmission rates. In this chapter we extend the analysis to spectrally efficient 16-symbol quadrature amplitude modulation (16-QAM) and 64-QAM. Doing so allows very high rates to be realized when the channel is good, while still providing acceptable, although lower, rates when the channel is poor, without requiring channel knowledge at the transmitter. We first present simulation results on Raptor codes, showing the throughput cap still exists for 16-QAM and 64-QAM over AWGN channels, regardless of N_m and the pre-coding rate. Following discussions are then focused on RC-IRA codes and RC/QC-LDPC codes with short and moderate block lengths.

5.1 System and Simulation Model

The discrete-time system model is the same as in Fig. 4.1. Descriptions can be found in the previous chapter, except the following parts.

5.1.1 Signal Mapping

The output of the encoder, codeword $\mathbf{c}$ is passed through a M-ary QAM signal mapper to form the transmitted symbol vector $\mathbf{s}$, composed of N_s complex symbols. Fig. 5.1 shows an example of the constellation and labeling scheme for 64-QAM. In this thesis we focus on square QAM constellations, and the Gray labeling we are using is similar to the one in Fig. 5.1. The reasons will be discussed in Section 5.1.3.

5.1.2 AWGN Channel

The transmitted symbol vector, $\mathbf{s}$, is corrupted by discrete-time white Gaussian noise sequence, $\mathbf{n}$, to form the received sample vector, $\mathbf{r} = \mathbf{s} + \mathbf{n}$, where each element of $\mathbf{n}$ is modeled as a statistically independent complex Gaussian random variable with zero mean and variance N_0.

5.1.3 Initial LLR Calculation

The only difference between a BP decoder for QAM and for BPSK is the calculation of the initial LLR's. In following discussions, we will use 64-QAM as an example to show how to derive the initial LLR calculation formulae for M-ary QAM. The derivation method is the same as the one used for BPSK.

At the receiver, we first estimate the SNR of the received signal. If the channel is only slowly-varying, we only conduct this estimation occasionally. For each received sample, the receiver calculates an initial LLR for each code bit, using the estimated

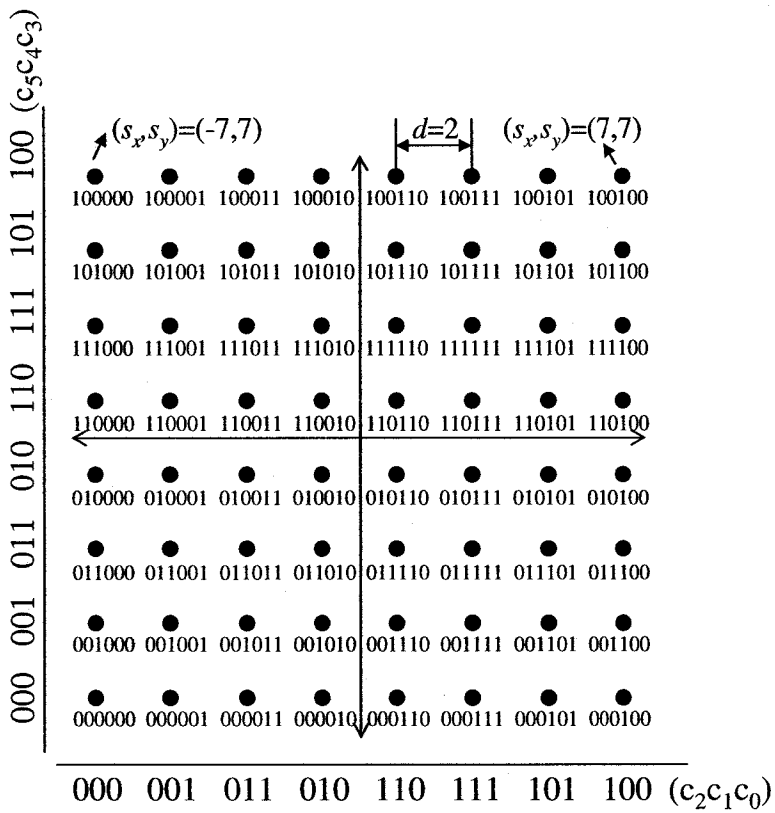

Fig. 5.1: Constellation of 64-QAM with Gray labeling. 8-PAM symbols in X-axis correspond to code bits $\{c_2c_1c_0\}$. 8-PAM symbols in Y-axis correspond to code bits $\{c_5c_4c_3\}$.

SNR and the received sample. Since we choose a square constellation, M-ary QAM can be viewed as two $\sqrt{M}$-ary PAM symbols; one corresponds to the real part of the QAM symbol while another corresponds to the imaginary part. By using Gray labeling similar to the one shown in Fig. 5.1, we decompose the LLR calculation of an M-ary QAM symbol into the LLR's of two $\sqrt{M}$-ary PAM symbols. Noticing the labeling symmetry in Fig. 5.1, it is evident that the LLR formulae of the real part also applies to the imaginary part.

Now let us assume the mapper in Fig. 5.1 is used. Given a received 64-QAM sample, r, corresponding to the transmitted s, its real and imaginary parts, r_x and r_y, can be viewed as 8-PAM symbols belonging to the alphabet $\{\pm 1, \pm 3, \pm 5, \pm 7\}$

and representing code bits $\{c_0, c_1, c_2\}$ and $\{c_3, c_4, c_5\}$, respectively. Taking c_0 as an example, we could express the initial LLR of c_0 as

$$\begin{aligned} \mathrm{LLR}_{ini}(c_0) &= \ln \frac{\mathrm{P}_{r_x|c_0}(r_x|c_0 = 0)}{\mathrm{P}_{r_x|c_0}(r_x|c_0 = 1)} \\ &= \ln \frac{\sum_{i\in\{\pm1,\pm7\}} \mathrm{P}_{r_x|s_x}(r_x|s_x = i)}{\sum_{i\in\{\pm3,\pm5\}} \mathrm{P}_{r_x|s_x}(r_x|s_x = i)}, \end{aligned} \tag{5.1}$$

where s_x is the real part of s. Note that the second equality holds only when the 64-QAM symbols are equally likely (for a large codeword length, this assumption is reasonable). For simplicity of expression, we denote $\mathrm{P}_{r_x|s_x}(r_x|s_x = i)$ as $\mathrm{P}(r_x|i)$, so Eq.(5.1) becomes

$$\begin{aligned} \mathrm{LLR}_{ini}(c_0) &= \ln \frac{\sum_{i\in\{\pm1,\pm7\}} \mathrm{P}(r_x|i)}{\sum_{i\in\{\pm3,\pm5\}} \mathrm{P}(r_x|i)} \\ &= \ln \frac{e^{-\frac{14r_x+49}{N_0}} + e^{-\frac{-14r_x+49}{N_0}} + e^{-\frac{2r_x+1}{N_0}} + e^{-\frac{-2r_x+1}{N_0}}}{e^{-\frac{10r_x+25}{N_0}} + e^{-\frac{-10r_x+25}{N_0}} + e^{-\frac{6r_x+9}{N_0}} + e^{-\frac{-6r_x+9}{N_0}}}, \end{aligned} \tag{5.2}$$

The initial LLR's of c_1 to c_5 can be derived in a similar manner. To simplify numerical implementation of Eq.(5.2), the receiver can store a pre-computed table of $\ln(1+e^{-x})$ for $x \geq 0$. Since the function $\ln(1 + e^{-x})$ is bounded by $(0, \ln 2]$ for $x \geq 0$, one could obtain satisfactory precision using a table with a small number of entries.

5.1.4 Simulation Model

The simplification on the simulation model for 16-QAM and 64-QAM is almost the same as for BPSK, except simulations using all-zero codewords. Taking the constellation shown in Fig. 5.1 as an example, all-zero codewords will be mapped into s composed of only one symbol $(s_x, s_y) = (-7, -7)$, which has the highest energy and therefore best protects code bits. Since N_0 in simulations is calculated by a given

SNR and the average symbol energy $E_s^{(avg)}$, using the symbol (-7,-7) certainly provides better simulation results. Even if we manage to make the all-zero code bits mapped into a symbol that happens to have the energy equal to $E_s^{(avg)}$, there is still a problem of unequal protection over code bits (this issue will be elaborated on in Section 5.5). This breaks the symmetric channel condition in [66] and therefore all-zero codeword cannot represent other codewords. A safe way is to ensure each QAM symbol is equally likely transmitted, so simulations can represent practical communication systems. In particular, we place a code bit randomizer before QAM mapper at the transmitter and a corresponding code bit de-randomizer after the LLR calculator at the receiver. The de-randomizer switches the sign of $\text{LLR}_{ini}(v_i)$ whenever the i^{th} bit in the output of the randomizer is 1. This technique is called a channel adapter in [69].

5.2 Simulation Results on Raptor Codes

Raptor codes with 16-QAM are simulated for the SNR range spanning from -5 dB to 20 dB, and the SNR range for 64-QAM is broader: -5 dB to 30 dB. The performance of 4-QAM is also simulated for comparison. The method of generating Raptor codes is the same as described in the previous chapter. The message word length N_m is 9500 or 9800 bits. In the Raptor decoder, the maximum number of iterations per decoding attempt, L, is set to 200.

Considering the above simulation parameters, each simulation scenario is characterized by the SNR, constellation (4-QAM, 16-QAM, 64-QAM), pre-coding rate (r_p = 0.95 or 0.98), and d_L (the variable degree of left-regular LDPC pre-coding). To avoid length-4 cycles in the factor graph, lower d_L is required for the higher pre-coding rate. In our simulations, there are no length-4 cycles for the parameter set

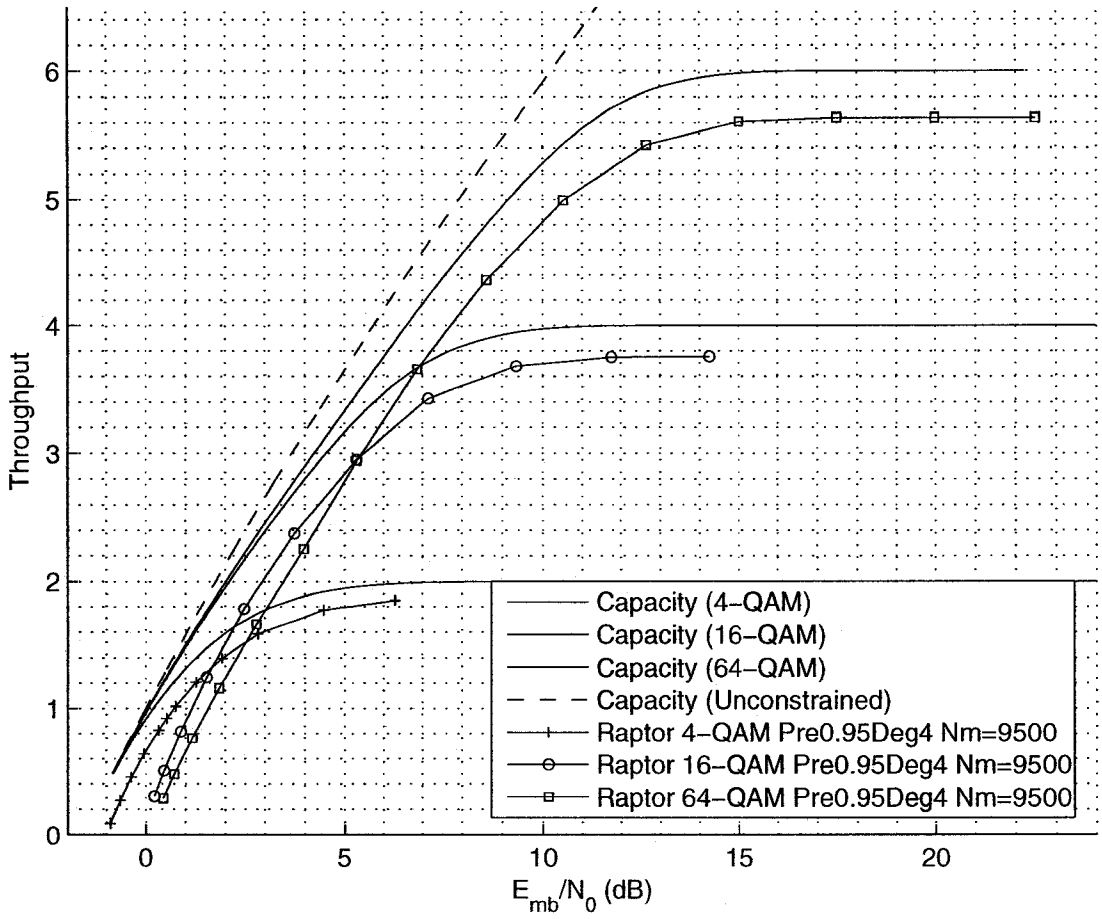

Fig. 5.2: Throughput vs. E_{mb}/N_0 ($N_m = 9500$). Pre0.95Deg4 Nm=9500 denotes $r_p = 0.95, d_L = 4$ for LDPC pre-coding and $N_m = 9500$. Others follow the same rule.

$\{d_L = 2, r_p = 0.98\}$ or $\{d_L = 4, r_p = 0.95\}$, while some length-4 cycles exist for the set $\{d_L = 4, r_p = 0.98\}$. Our simulation results show that given $r_p = 0.98$, $d_L = 4$ outperforms $d_L = 2$. Therefore, we only present results on $d_L = 4$. At least 512 codewords are run for each simulation scenario.

Fig. 5.2 shows the system throughput as a function of the transmitted energy per message bit (E_{mb}) over N_0. We observe that Raptor codes with 16-QAM or 64-QAM work still perform worse in high capacity channels because the LT G-graph does not cover all pre-code bits in a timely manner. Comparing pre-coding rate 0.98 with 0.95 for all constellations, we found the capacity of $r_p = 0.98$ is slightly higher than for

$r_p = 0.95$ at each SNR, as shown in Fig. 5.3 and 5.4. The reason could be rate 0.98 provides lower redundancy than rate 0.95 or we happen to have selected better $\mathbf{H}_p$ and $\mathbf{G}_c$ in the rate 0.98 case than in the rate 0.95. Further study is needed to identify the reason.

Notice that 64-QAM performs worse than 16-QAM when E_{mb}/N_0 is lower than 5 dB and 16-QAM performs worse than 4-QAM when E_{mb}/N_0 is lower than 1.2 dB. To exploit this, we suggest that combining Raptor codes with adaptive constellation selection such that larger throughput could be obtained over a wide SNR range. To make this proposal work, the additional requirement at the transmitter is knowing a rough SNR range instead of knowing the exact SNR, as required in ACM.

We also compared Raptor codes with a simple ACM scheme that uses three optimized irregular LDPC codes with fixed rate 0.5, 0.75 and 0.98, respectively. The purpose of this investigation is to show the advantage of rateless codes over fixed-rate codes in terms of throughput performance. A similar message word length ($N_m \approx 9500$) is used for the ACM scheme. From [68], we obtained three optimized left degree distributions for three irregular LDPC codes, and then use the PEG algorithm to generate the parity check matrix of each LDPC code. The Throughput vs. E_s/N_0 curves under 4-QAM, 16-QAM and 64-QAM are shown in Fig. 5.5, 5.6, and 5.7, respectively. Supposing N_w codewords are transmitted and N_{crt} codewords are successfully decoded, the ARR of a fixed-rate code is defined in the same way as the ARR of a rateless code:

$$\begin{aligned} \mathrm{ARR}_{\mathrm{FIXED}} &= \frac{N_{crt}N_m}{\sum_{i=1}^{N_w} N_s} = \frac{N_{crt}N_m}{N_w N_s} = \frac{N_{crt}}{N_w} \cdot \frac{N_m}{N_s} \\ &= (1 - \mathrm{WER})R_{nom}. \end{aligned} \tag{5.3}$$

where WER is word error rate and R_{nom} is the nominal rate of the LDPC code.

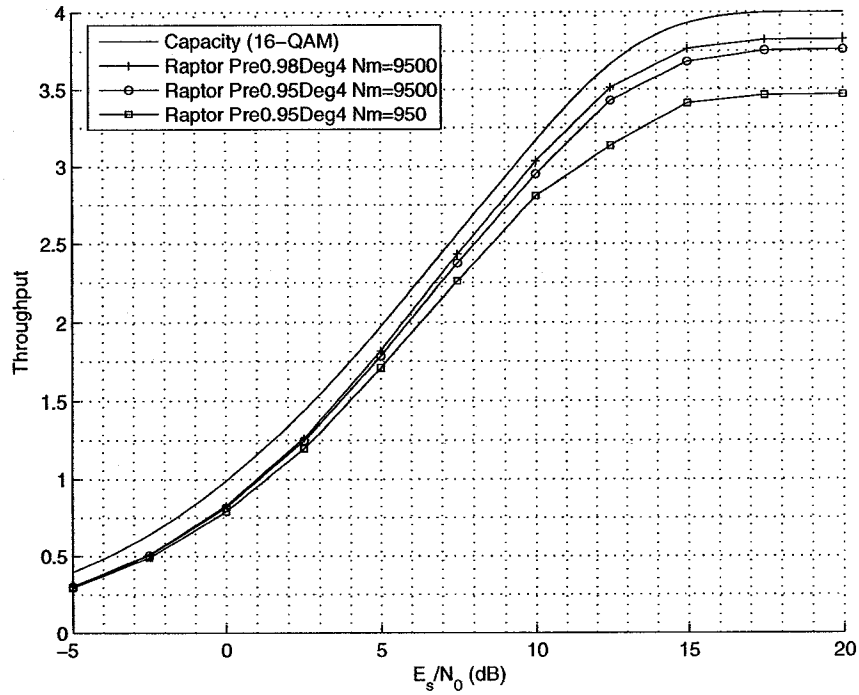

Fig. 5.3: Throughput vs. E_s/N_0 ($N_m = 9500$ and 950), for 16-QAM.

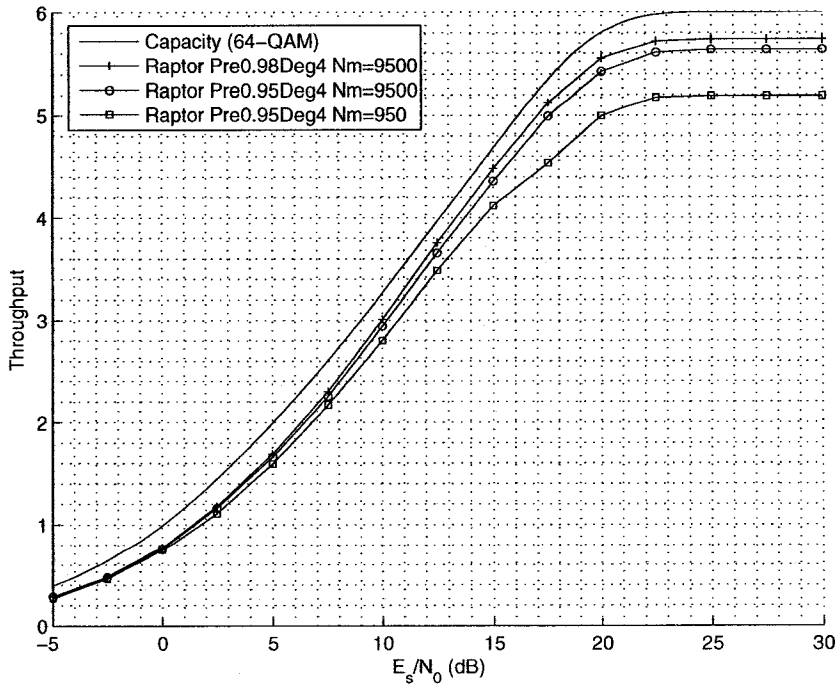

Fig. 5.4: Throughput vs. E_s/N_0 ($N_m = 9500$ and 950), for 64-QAM.

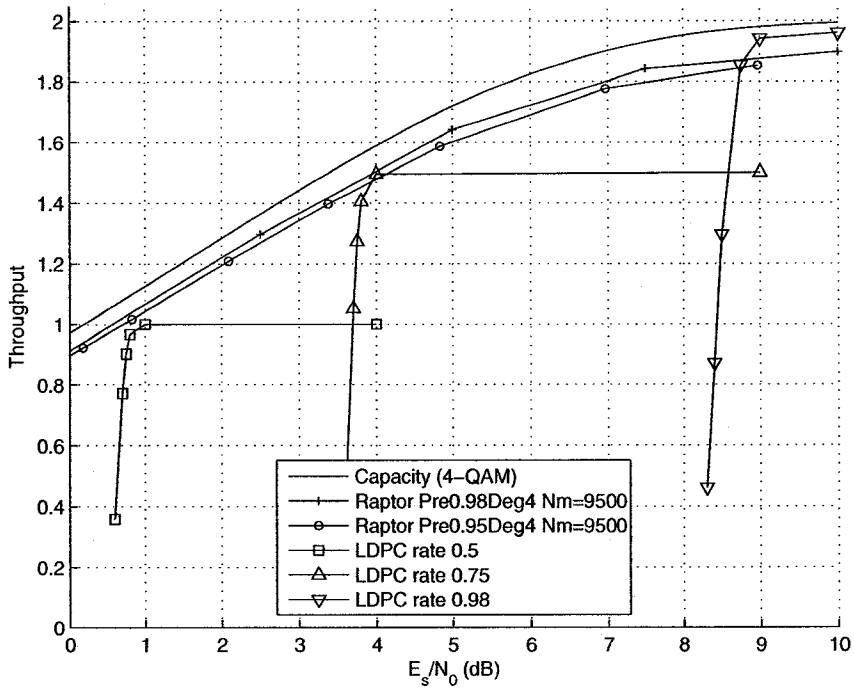

Fig. 5.5: 4-QAM throughput vs. E_s/N_0 ($N_m \approx 9500$).

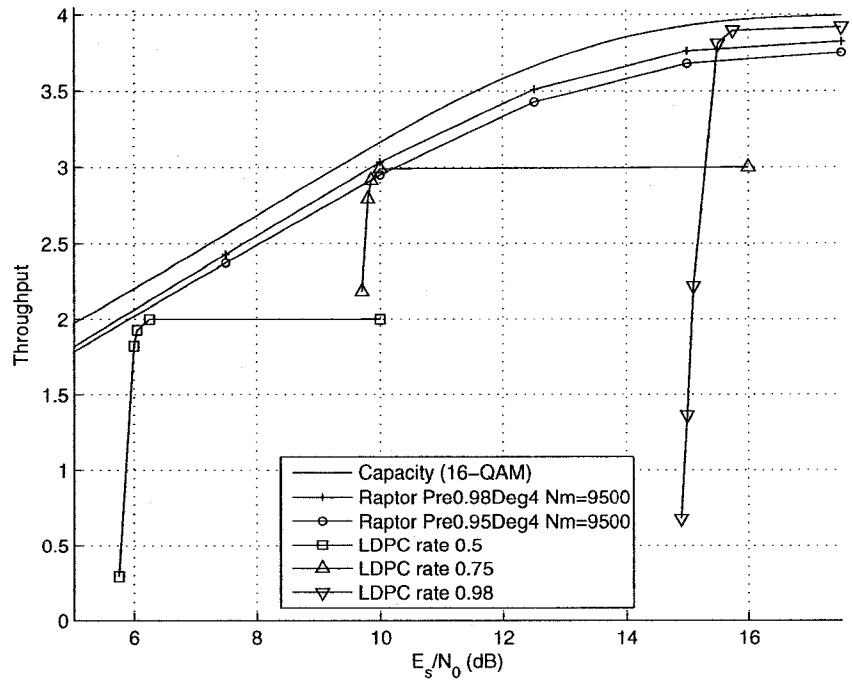

Fig. 5.6: 16-QAM throughput vs. E_s/N_0 ($N_m \approx 9500$).

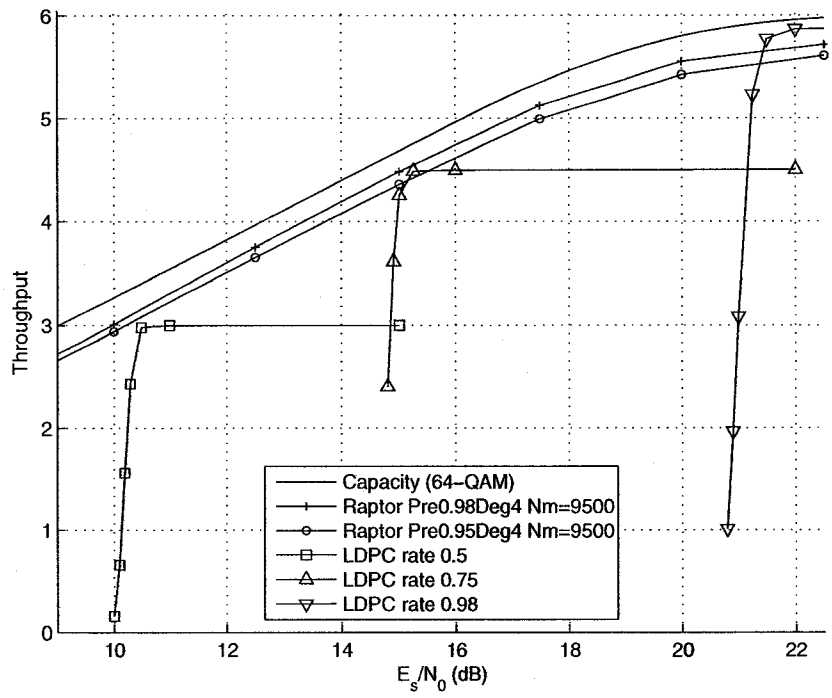

Fig. 5.7: 64-QAM throughput vs. E_s/N_0 ($N_m \approx 9500$).

As expected, Raptor codes have smooth throughput curve across E_s/N_0, unlike the staircase-like curve in the ACM scheme. Note that if ACM uses rate 0.5 LDPC or rate 0.75 LDPC, the throughput of ACM is worse than Raptor codes at all SNR's. This does not imply that the H matrix protection capability of the fixed-rate LDPC code is worse than the Raptor code with a similar codeword length. It is the rateless nature of Raptor codes that makes the ARR higher. Rateless codes like Raptor codes have an ability to vary the number of required symbols in much smaller increments (T_s symbols) than traditional codes like LDPC codes where a whole codeword is used as an increment (in the case where the codeword is received incorrectly). This ability gives rateless codes a huge advantage on ARR performance.

Raptor codes require no channel knowledge at the transmitter and only require noise level estimation at the receiver for an AWGN channel, while ACM requires the transmitter to have accurate channel knowledge. Therefore, we prefer Raptor codes

to ACM when good channel knowledge is not available at the transmitter. However, it must be clarified that with perfect channel knowledge at the transmitter, an ACM scheme using dedicated LDPC codes should give better throughput performance across a wide SNR range than rateless codes. Certainly, the price we will pay is much higher implementation cost due to the requirements of precise channel estimation and large memory required to store enough different H matrices, to reduce the capacity loss caused by the rate quantization effect in ACM.

We demonstrated that Raptor codes with 16-QAM and 64-QAM work well over a very wide SNR range but still perform worse in high capacity channels. We also show than the ACM scheme is worse than Raptor codes in terms of ARR. Thus, the following comparison is focused on RC-IRA codes and RC/QC-LDPC codes.

5.3 Throughput of RC-IRA Codes and RC/QC-LDPC Codes

The throughput of RC-IRA codes and RC/QC-LDPC codes are compared in the E_s/N_0 range from 2 dB to 22 dB. We focus on short and moderate message word length: $N_m = 188$ and 1528 for RC-IRA codes, and $N_m = 192$ and 1536 for RC/QC-LDPC codes (to avoid zero-padding). The mapping is 16-QAM with Gray labeling, shown in Fig. 5.8.

In previous chapter we discussed variants of RC-IRA codes. In this section, unless otherwise indicated, RC-IRA codes have mother code rate 0.923, the variable degree distribution given by Eq. (4.2), no parallel edges, and a BP decoder using BP and BCJR algorithms with Turbo-like scheduling.

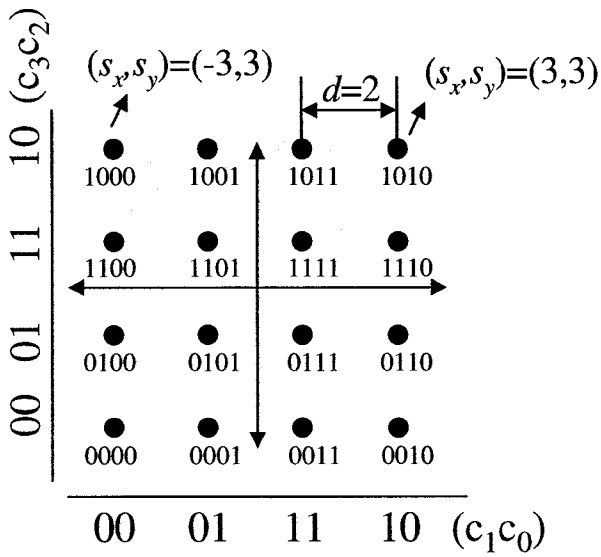

Fig. 5.8: Constellation of 16-QAM with Gray labeling. 4-PAM symbols in X-axis correspond to code bits $\{c_1c_0\}$. 4-PAM symbols in Y-axis correspond to code bits $\{c_3c_2\}$. .

5.3.1 Throughput

Fig. 5.9 and 5.10 show the system throughput against E_s/N_0 under short and moderate message word lengths. Similar to BPSK, RC/QC-LDPC codes still suffer the throughput cap at higher SNR's because the large number of punctured code bits paralyzed the message passing. At lower SNR's, the throughput of RC-IRA codes is around 0.2 dB worse than RC/QC-LDPC codes. Their ARR curves cross at 9.85 dB (ARR/4 $\approx$ 0.73) for short block length and at around 8.76 dB (ARR/4 $\approx$ 0.68) for moderate block length. The cross points in BPSK are similar: ARR is 0.75 for short blocks and 0.72 for moderate blocks.

Further comparisons between RC-IRA and RC/QC-LDPC codes under fading channels will be given in Chapter 6. Following sections in this chapter are solely contributed to how to improve RC-IRA codes under 16-QAM modulation. Particularly, we explore the effects of iterative de-mapping, labeling, and reliability-based coded modulation (RBCM).

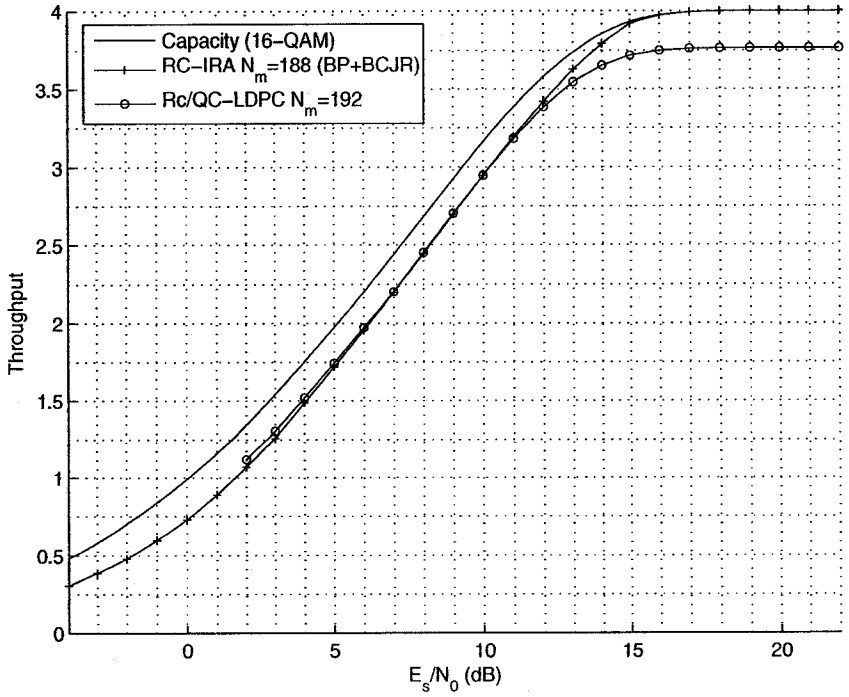

Fig. 5.9: Throughput vs. E_s/N_0 ($N_m \approx 188$).

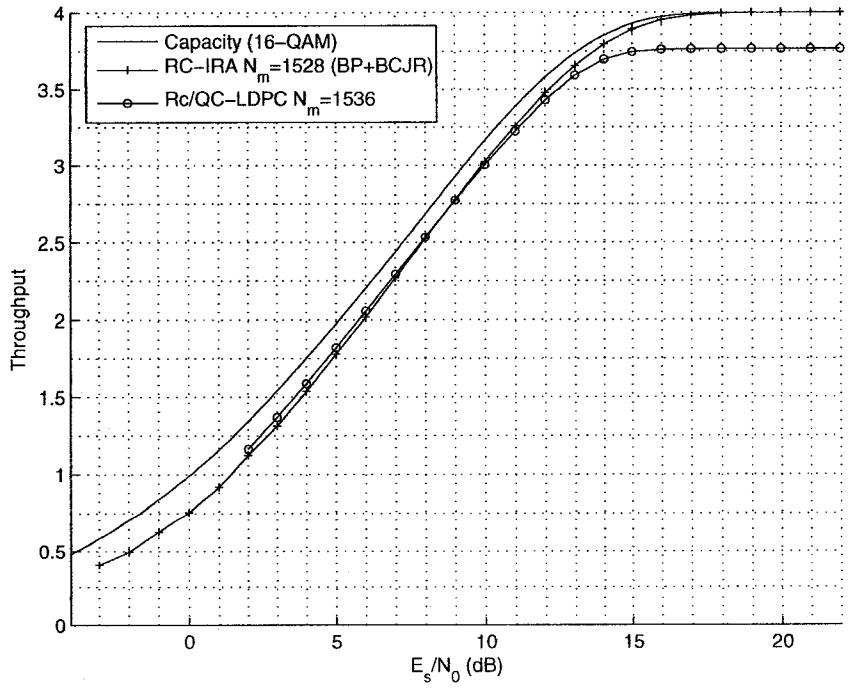

Fig. 5.10: Throughput vs. E_s/N_0 ($N_m \approx 1528$).

5.4 Iterative De-mapping

During the initialization of BP decoding, the LLR calculator estimates LLR_{ini}'s for each code bit. We were assuming, for a given c_i equal to either 0 or 1, eight 16-QAM symbols were equally likely sent. Thus, for the 16-QAM constellation with Gray labeling shown in Fig. 5.8, we have the $\mathrm{LLR}_{ini}(\mathrm{c}_0)$ given by

$$\begin{aligned}\mathrm{LLR}_{ini}(\mathrm{c}_0) &= \ln \frac{\mathrm{P}_{r_x|\ \mathrm{c}_0}(r_x|\ \mathrm{c}_0 = 0)}{\mathrm{P}_{r_x|\ \mathrm{c}_0}(r_x|\ \mathrm{c}_0 = 1)} \\ &= \ln \frac{\sum_{i\in\{\pm 3\}} \mathrm{P}_{r_x|s_x}(r_x|s_x = i)}{\sum_{i\in\{\pm 1\}} \mathrm{P}_{r_x|s_x}(r_x|s_x = i)},\end{aligned} \tag{5.4}$$

where the second equality is based on the assumption that symbols containing c_0 equal to either 0 or 1 are equally likely sent. Final results of $\mathrm{LLR}_{ini}(\mathrm{c}_i)$ $(i \in \{0,1,2,3\})$ are given by

$$\begin{aligned}\mathrm{LLR}_{ini}(\mathrm{c}_0) &= \ln\left(\frac{e^{-4\frac{r_x+2}{N_0}} + e^{8\frac{r_x-1}{N_0}}}{1 + e^{4\frac{r_x}{N_0}}}\right) \\ \mathrm{LLR}_{ini}(\mathrm{c}_1) &= \ln\left(\frac{e^{-8\frac{r_x}{N_0}} + e^{-4\frac{r_x-2}{N_0}}}{e^{8N_0{}^{-1}} + e^{4\frac{r_x}{N_0}}}\right) \\ \mathrm{LLR}_{ini}(\mathrm{c}_2) &= \ln\left(\frac{e^{8\frac{r_y-1}{N_0}} + e^{-4\frac{r_y+2}{N_0}}}{e^{4\frac{r_y}{N_0}} + 1}\right) \\ \mathrm{LLR}_{ini}(\mathrm{c}_3) &= \ln\left(\frac{e^{-4\frac{r_y-2}{N_0}} + e^{-8\frac{r_y}{N_0}}}{e^{4\frac{r_y}{N_0}} + e^{8N_0{}^{-1}}}\right).\end{aligned} \tag{5.5}$$

At the end of each iteration, variable nodes make hard-decisions based on real-valued messages from factor nodes and LLR_{ini}'s. If the de-mapper uses the messages from factor nodes as *a priori* knowledge on symbols, we can have better throughput. The idea of iterative de-mapping originates from iterative multi-stage decoding (MSD) in multi-level codes (MLC) [70], where soft feedback between component codes are

used to reduce error propagation from lower level component codes to higher level component codes in conventional MSD [71–73]. Like in MLC where component codes have different protections over code bits, different bit levels in multilevel modulation also have different protections over code bits. Thus, motivated by iterative MSD in MLC, Li and Ritcey proposed in [74] iterative decoding for bit-interleaved coded modulation (BICM) [75], where hard feedback between bit levels in multilevel modulation is introduced for better de-mapping. Soft feedback between bit levels was proposed by S. ten Brink *et al* [76] and Li *et al* [77]. Iterative de-mapping (IDM) was extended to LDPC codes in [78, 79]. Iterative decoding in BICM in the above literatures is referred to as BICM-ID. To be more specific in this document, we use the name iterative de-mapping to refer to the message exchanging between the de-mapper and the decoder.

Note that the throughput of using IDM is bounded by the capacity of coded modulation, rather than the BICM capacity. Provided Gray labeling is used, the difference between the two capacities is larger at lower SNR's than at higher SNR's. For example, the difference at rate 1/2 for a Gray-labeled 4-PAM on an AWGN channel is only 0.16 dB [69, Figure 4]. So it is expected that IDM can increase the throughput at low SNR's but not at high SNR's. We now give the detailed description on how to apply IDM for 16-QAM with a BP decoder with flooding and a BP and BCJR decoder with Turbo-like scheduling.

5.4.1 BP with IDM in 16-QAM

The BP decoder with flooding is described in Section 2.4. We add the IDM operation into each iteration, and schedule it right after step 4 (variable node operation). IDM operations are carried out by symbols to update LLR_{ini}'s. Take c_0 as an example,

$\mathrm{LLR}_{ini}(\mathrm{c}_0)$ is updated by

$$\begin{aligned}\mathrm{LLR}_{ini}(\mathrm{c}_0) &= \ln \frac{\mathrm{P}_{r_x|\ \mathrm{c}_0}(r_x|\ \mathrm{c}_0 = 0)}{\mathrm{P}_{r_x|\ \mathrm{c}_0}(r_x|\ \mathrm{c}_0 = 1)} \\ &= \ln \frac{\mathrm{P}_{r_x|s_x}(r_x|s_x = 3)\mathrm{P}(s_x = 3) + \mathrm{P}_{r_x|s_x}(r_x|s_x = -3)\mathrm{P}(s_x = -3)}{\mathrm{P}_{r_x|s_x}(r_x|s_x = 1)\mathrm{P}(s_x = 1) + \mathrm{P}_{r_x|s_x}(r_x|s_x = -1)\mathrm{P}(s_x = -1)} \\ &= \ln \frac{\mathrm{P}_{r_x|s_x}(r_x|s_x = 3)\mathrm{P}(\mathrm{c}_1 = 1) + \mathrm{P}_{r_x|s_x}(r_x|s_x = -3)\mathrm{P}(\mathrm{c}_1 = 0)}{\mathrm{P}_{r_x|s_x}(r_x|s_x = 1)\mathrm{P}(\mathrm{c}_1 = 1) + \mathrm{P}_{r_x|s_x}(r_x|s_x = -1)\mathrm{P}(\mathrm{c}_1 = 0)},\end{aligned} \tag{5.6}$$

The sum of incident messages to the variable node c_1 is a symbol-by-symbol MAP estimation on c_1, including the $\mathrm{LLR}_{ini}(\mathrm{c}_1)$ used in previous iteration and the messages from neighbor check nodes. We only use the sum of messages from neighbor check nodes of c_1, denoted as μ_1, to derive $\mathrm{P}(\mathrm{c}_1 = 0)$ and $\mathrm{P}(\mathrm{c}_1 = 1)$ in Eq. (5.6). By $\mu_1 = \ln[\mathrm{P}(\mathrm{c}_1 = 0)/\mathrm{P}(\mathrm{c}_1 = 1)]$, we have

$$\begin{aligned}\mathrm{P}(\mathrm{c}_1 = 0) &= \frac{e^{\mu_1}}{1 + e^{\mu_1}}, \\ \mathrm{P}(\mathrm{c}_1 = 1) &= \frac{1}{1 + e^{\mu_1}}.\end{aligned} \tag{5.7}$$

Plugging Eq. (5.7) into Eq. (5.6), the denominator $(1 + e^{\mu_1})$ will be canceled out and we have

$$\mathrm{LLR}_{ini}(\mathrm{c}_0) = \ln \left(\frac{e^{\frac{-4\,rx-8+m1\,N0}{N0}} + e^{8\,\frac{rx-1}{N0}}}{e^{m1} + e^{4\,\frac{rx}{N0}}} \right). \tag{5.8}$$

The IDM operations in 16-QAM are then summarized as

$$\begin{aligned}
\mathrm{LLR}_{ini}(\mathrm{c}_0) &= \ln\left(\frac{e^{\frac{-4\,r_x-8+\mu_1\,N_0}{N_0}}+e^{8\,\frac{r_x-1}{N_0}}}{e^{\mu_1}+e^{4\,\frac{r_x}{N_0}}}\right),\\
\mathrm{LLR}_{ini}(\mathrm{c}_1) &= \ln\left(\frac{e^{-\frac{8\,r_x-\mu_0\,N_0}{N_0}}+e^{-4\,\frac{r_x-2}{N_0}}}{e^{8\,{N_0}^{-1}}+e^{\frac{4\,r_x+\mu_0\,N_0}{N_0}}}\right),\\
\mathrm{LLR}_{ini}(\mathrm{c}_2) &= \ln\left(\frac{e^{8\,\frac{r_y-1}{N_0}}+e^{\frac{-4\,r_y-8+\mu_3\,N_0}{N_0}}}{e^{4\,\frac{r_y}{N_0}}+e^{\mu_3}}\right),\\
\mathrm{LLR}_{ini}(\mathrm{c}_3) &= \ln\left(\frac{e^{-4\,\frac{r_y-2}{N_0}}+e^{\frac{-8\,r_y+\mu_2\,N_0}{N_0}}}{e^{\frac{4\,r_y+\mu_2\,N_0}{N_0}}+e^{8\,{N_0}^{-1}}}\right),
\end{aligned} \tag{5.9}$$

where μ_i is the sum of messages emanated from neighbor check nodes of c_i.

Eq. (5.9) is almost the same as Eq. (5.5), except some additional μ_i's. If we store LLR_{ini}'s before the first iteration, one IDM operation on one code bit only involves one addition and one CHK operation. The additional complexity brought by IDM is thus acceptable for 16-QAM with Gray labeling. We can also decrease the frequency of LLR updating to further reduce the complexity of IDM, termed belief propagation coded modulation with iterative decoding (BPCM-ID) in [78].

From the form of Eq. (5.7), we can expect that the similarity between LLR updating equations and LLR initializing equations, holds for any labeling. LLR updating equations for other labeling will be discussed in Section 5.4.4.

5.4.2 BP and BCJR with IDM in 16-QAM

The BP and BCJR algorithm with Turbo-like scheduling, described in Section 3.3.3, can be also combined with IDM. We also add IDM operation into each iteration, scheduled after step 4 (continue on BP decoding). IDM operation are carried out by symbols to update the values of LLR_{ini}'s.

5.4.3 Throughput with IDM

Fig. 5.11 and 5.12 show that IDM can increase the throughput at lower SNR's for both short and moderate message word size. We notice that the gain of IDM decreases from 0.45 dB at rate 1/4 to 0.1 dB at rate 1/2. And when the capacity of the channel is larger than 2, IDM cannot noticeably change the throughput. The reason is that Gray labeling can largely reduce the uncoded BER at high SNR's, providing reliable LLR_{ini}'s for BP decoding. There is not much room left for IDM to enhance BP decoding performance when the SNR is high. We also observe the frequency of LLR updating can be reduced to every 20 iterations without much loss of performance.

5.4.4 Choices of Labeling

Gray labeling in M-ary modulation with large M has multiple nonequivalent choices [80]. We have been focusing on binary reflected Gray labeling (BRGL) originally proposed by Gray in his patent [81, column 6]. Choosing BRGL is backed by the analysis given by Agrell *et al* in [82,83]. In [83], the optimality of BRGL in the sense of BER was proved for uncoded systems over AWGN channel with M-ary PSK, M-ary PAM and M-ary QAM, provided the receiver uses minimum Euclidean distance detection [19, page 243, 265] and the SNR is higher than some threshold determined by the modulation scheme and M. The optimality of BRGL in M-ary QAM actually follows from the optimality of BRGL in M-ary PAM, since Gray labeling of rectangular 2^{m_x}-by-2^{m_y} QAM has to be a product of one 2^{m_x}-point Gray-labeled PAM and one 2^{m_y}-point Gray-labeled PAM [80]. We notice that Agrell *et al* used a receiver equipped with minimum Euclidean distance detection, which is designed to minimize symbol error rate (SER), while a BP decoder is designed to minimize BER and therefore LLR_{ini}'s are calculated based on bit likelihood rather than Euclidean distance.

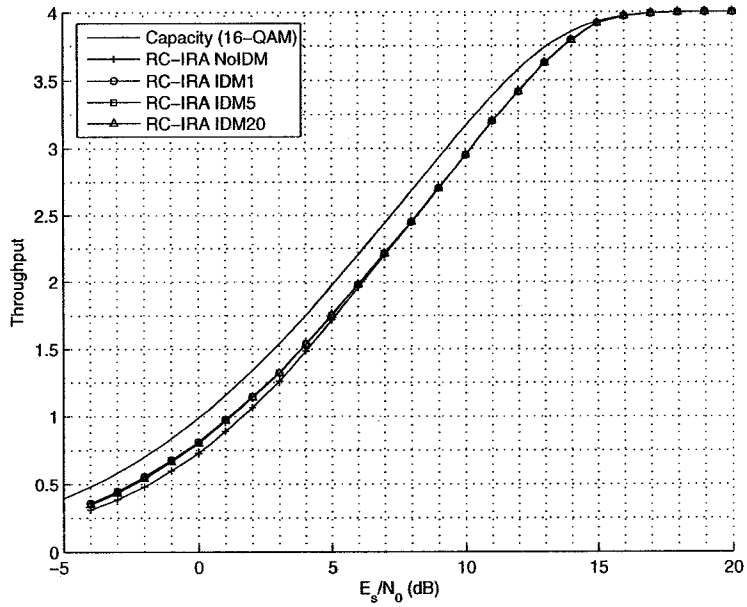

Fig. 5.11: Throughput vs. E_s/N_0 ($N_m = 188$). IDM1 denotes updating LLR every iteration. IDM5 denotes updating LLR every 5 iterations. IDM20 denotes updating LLR every 20 iterations.

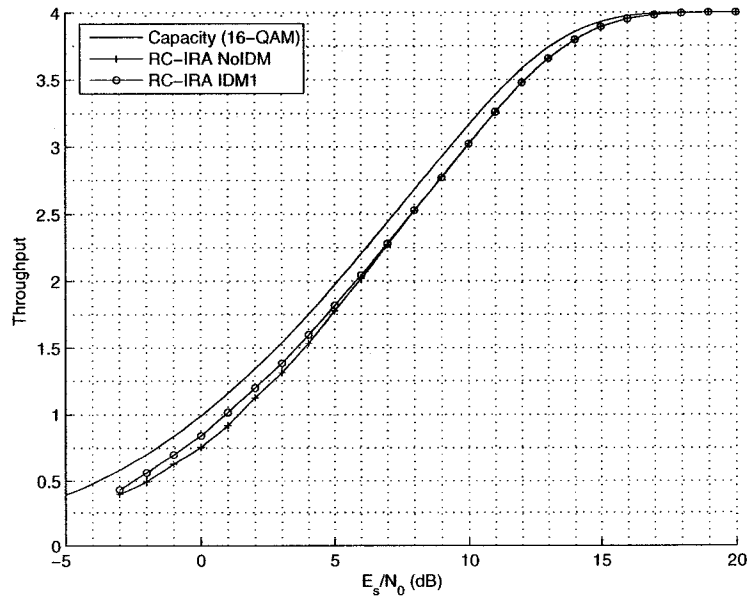

Fig. 5.12: Throughput vs. E_s/N_0 ($N_m = 1528$). IDM1 denotes updating LLR every iteration.

Therefore, we compared the BER difference between two receivers for uncoded systems over AWGN channel with BRGL-labeled 4-PAM and 8-PAM. Fig. 5.13 and 5.14 show that when Gray labeling is used, the BER performance between minimizing BER and minimizing SER is very small even at low SNR's. Thus, we conjecture that BFGC is also optimum for BP decoders.

However, the above analysis is for uncoded systems. The optimality of BRGL for coded systems still remain unanswered [80]. Other than Gray labeling, there are other labeling available for 16-QAM, namely, set-partitioning (SP) labeling by Ungerboeck [84], modified set-partitioning (MSP) and mixed labeling by Chindapol and Ritcey [85], , and M16a labeling by Schreckenbach *et al* [86]. Rui *et al* investigated the performance of a rate 1/2 (3,6)-regular LDPC code with BPCM-ID (a reduced-complexity version of BICM-ID) over Rayleigh fading channel with 16-QAM under different labeling [78]. Their results showed that BER of MSP, SP and mixed labeling become smaller than BER of Gray labeling when the latter is smaller than 10^{-1}; while BER of Gray labeling is larger than 10^{-1}, Gray labeling provides the smallest BER. Similar results were reported for rate 1/2 convolutional codes [86]. We now investigate the effects of these labeling on the throughput of RC-IRA codes. The labeling scheme is shown in Fig. 5.15.

Fig. 5.16 show the throughput of RC-IRA codes with SP, MSP, mixed, M16a, and Gray labeling. Although IDM provides significant gains for SP, MSP, M16a and mixed labeling, Gray labeling still performs best. This confirms the conjecture given in Section 4.3.1 that the methods to improve the throughput of rateless codes should focus more on how to reduce the WER at lower SNR's. Results in [86] and [78] both showed that Gray labeling provides the lowest BER at lower SNR's. It is not unexpected that the throughput of RC-IRA codes is maximized by Gray labeling.

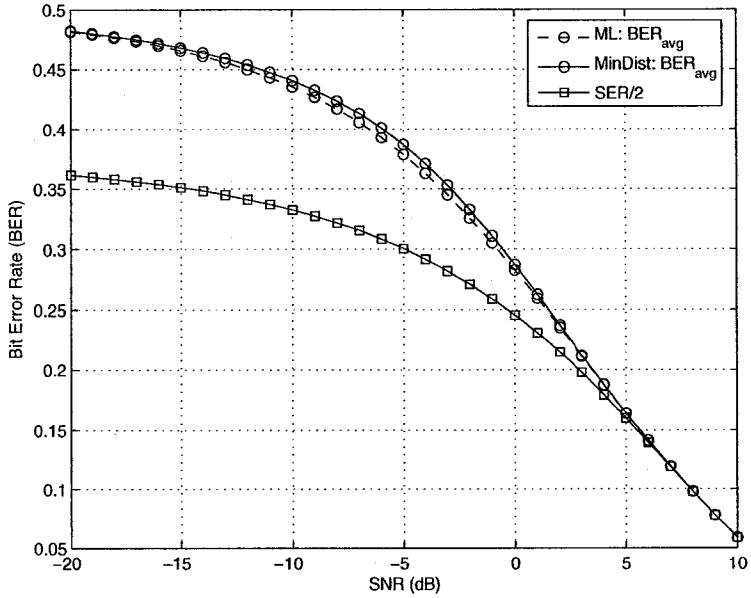

Fig. 5.13: BER vs. SNR for uncoded systems with Gray-labeled 4-PAM over AWGN channel. BER_{avg} denotes the average BER over all code bits, assuming they are equally likely sent.

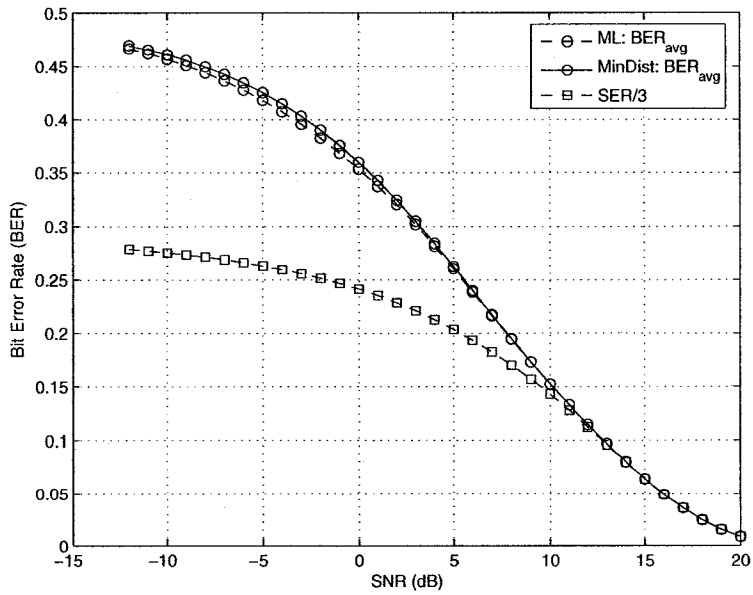

Fig. 5.14: BER vs. SNR for uncoded systems with Gray-labeled 8-PAM over AWGN channel.

Actually, Caire *et al* conjectured in [75] that Gray labeling maximizes the BICM capacity.

Comparing Fig. 5.16 with the BICM capacity of SP labeling shown in [75, Figure 4], we notice that with the help of IDM, the throughput of SP labeling can be larger than the BICM capacity of SP labeling, since the calculation of BICM capacity does not consider *a priori* knowledge of transmitted symbols.

In Fig. 5.16, the throughput of mixed labeling comes the closest to Gray labeling. The reason might be their similarity - both can be viewed as a product of two PAM. Gary labeling is a product of two Gray labeled 4-PAM: 00, 01, 11, 10. The mixed labeling is a product of two 4-PAM: 00, 11, 01, 10. Compared to SP, MSP and M16a labeling, mixed and Gray labeling have very simple LLR calculation and LLR updating equations. This brings us least additional complexity due to IDM. Therefore, we only use Gray labeling for the following discussions due to its good throughput performance and simplicity. By Gray labeling, we always mean binary reflected Gray labeling.

It is rather complex to derive the LLR formulae for labeling other than Gray or mixed labeling. A safe and efficient way is using the symbolic toolbox provided by MATLAB® software, especially when a higher-order modulation like 64-QAM or 256-QAM is used.

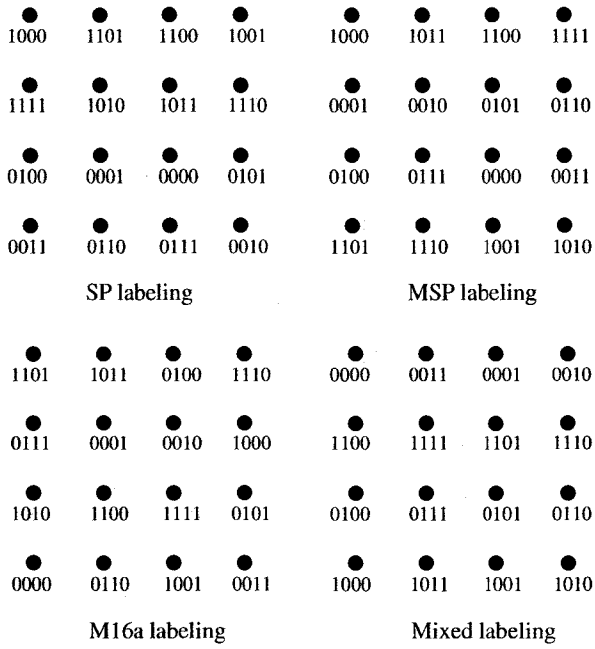

Fig. 5.15: Constellation of 16-QAM with other labeling.

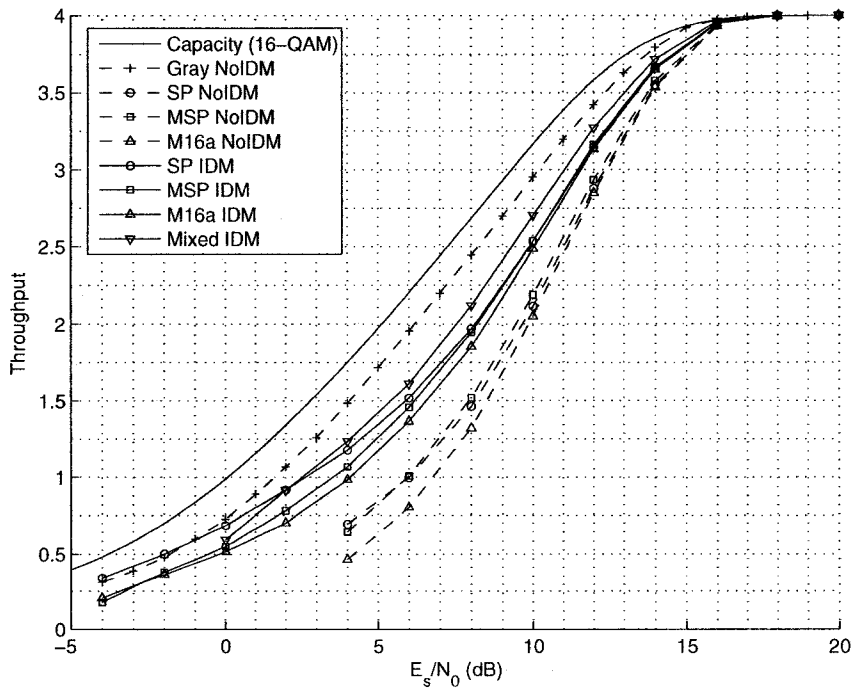

Fig. 5.16: Throughput vs. E_s/N_0 (RC-IRA codes: $N_m = 188$). NoIDM denotes without IDM.

5.5 Reliability-based Coded Modulation

In the previous section, the receiver can use IDM to increase the throughput by combining de-mapping and decoding. We now investigate how a transmitter can combine mapping and coding to increase the throughput of RC-IRA codes.

5.5.1 Unequal Error Protection in Multilevel Modulation

Fig. 5.17, 5.18 and 5.19 show BER against SNR for uncoded Gray-labeled 4-PAM and 8-PAM over AWGN channel. Two types of receivers are investigated. One detects the mostly likely transmitted symbols using minimum distance detection, labeled "MinDist" in the figures. Another detects the mostly likely transmitted bits using LLR's, labeled "ML" in the figures. We can observe that unequal error protection (UEP) over different bit levels exists for both receivers and both modulations. For example, in 8-PAM, c_1 is better protected than c_0 and c_2 is even better protected. This can be easier seen at higher SNR's. In this thesis, bit levels like c_0 are defined as unreliable bit levels and the rest are defined as reliable bit levels. Since Gray-labeling is used, SER/2 and SER/3 provide accurate estimation for BER of 4-PAM and 8-PAM at high SNR's.

Knowing the unequal error protection (UEP) in multilevel modulation, the transmitter can combine coding and mapping to increase the throughput. This is named reliability-based coded modulation (RBCM) in [87], where the authors permutated columns of a regular $\mathbf{H}$ such that neighboring variable nodes of every check node include at most one unreliable bit. We denote this requirement as $N_{UBL}(f_i) \leq 1$ for any i, where $N_{UBL}(f_i)$ indicates the number of unreliable bit levels connected to f_i. Given an LDPC code modulated with Gray-labeled 16-QAM, at most we can tolerate $N_c - N_m$ unreliable bits. This requires $N_c/2 \leq N_c - N_m$, i.e. $r \leq 1/2$. For Gray-labeled 64-QAM, we need $N_c/3 \leq N_c - N_m$, i.e. $r \leq 2/3$.

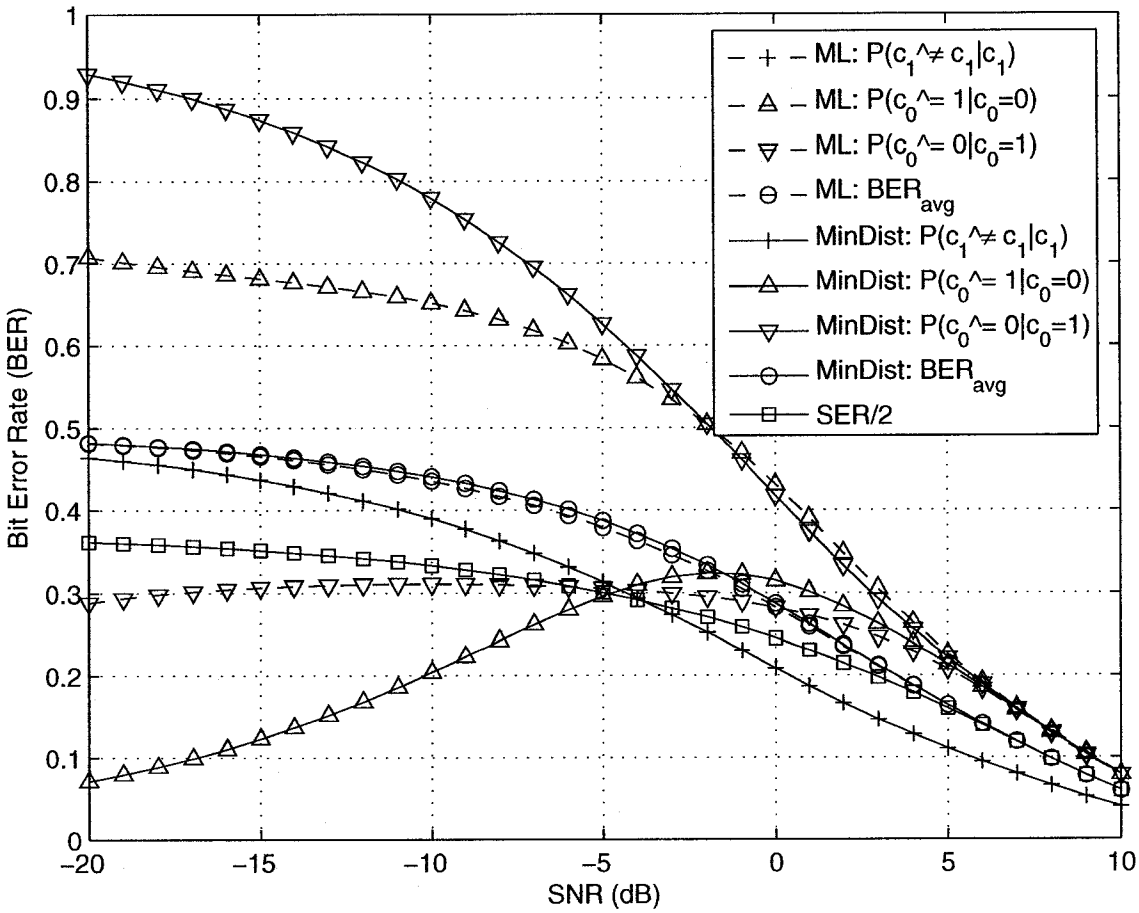

Fig. 5.17: BER vs. SNR for uncoded systems with Gray-labeled 4-PAM over AWGN channel. $\hat{c}_i$ denotes the estimated bit c_i. BER_{avg} denotes the average BER over all code bits, assuming they are equally likely sent.

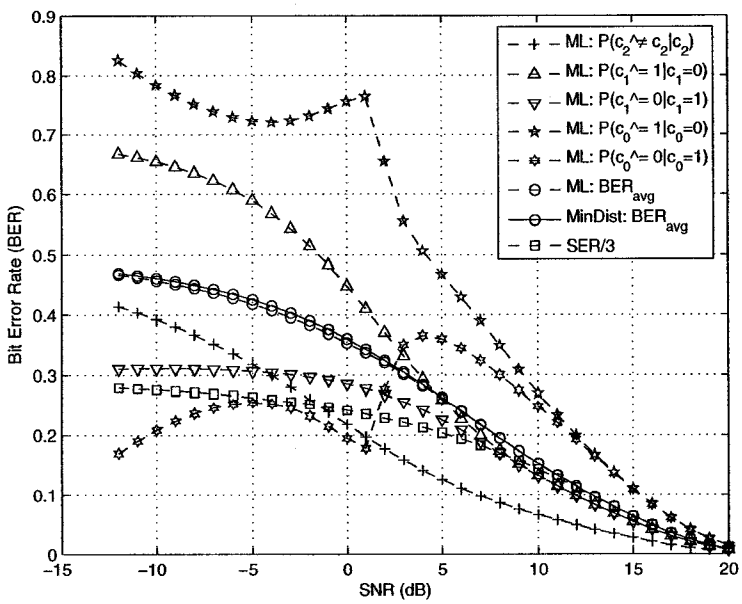

Fig. 5.18: BER vs. SNR for uncoded systems with Gray-labeled 8-PAM over AWGN channel. The receiver uses LLR of bits to detect mostly likely bits.

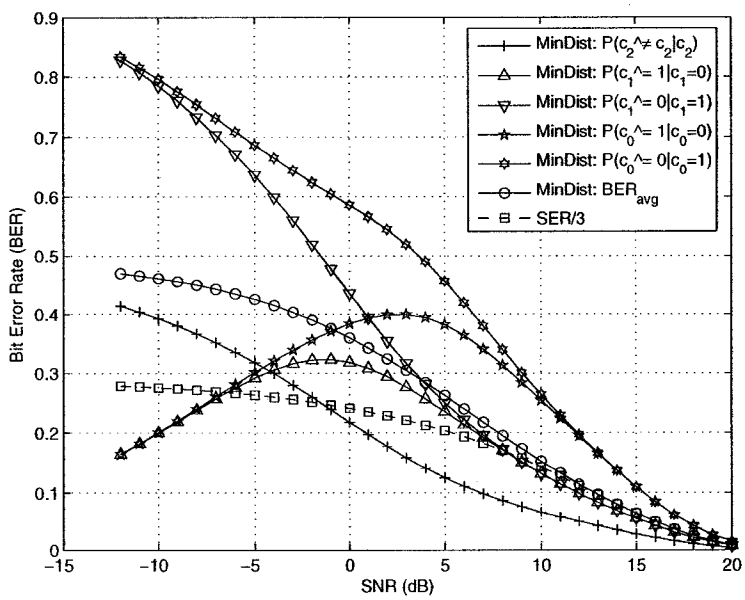

Fig. 5.19: BER vs. SNR for uncoded systems with Gray-labeled 8-PAM over AWGN channel. The receiver uses minimum distance to detect mostly likely symbols.

There are other ways to apply RBCM for LDPC codes. Li and Ryan map bits with high d_s to reliable bit levels and bits with low d_s to unreliable bit levels [88]. Since variable nodes with higher d_s emanate more edges than nodes with lower d_s, assigning reliable bit levels for nodes with higher d_s leads to more reliable edges in the factor graph. Therefore, this scheme can be viewed as a simplified guide to satisfy the requirement of check nodes in [87]. Authors in [89] map systematic bits to reliable bit levels. All three RBCM schemes are simulated for fixed-rate LDPC codes and obtained around 0.2 to 0.4 dB gain in BER performance. We now investigate the effect of RBCM on RC-IRA codes mapped by Gray-labeled 16-QAM over the AWGN channel.

5.5.2 RC-IRA Codes with RBCM

As a rateless code, we cannot map systematic bits only to reliable bit levels. Therefore two RBCM schemes are left for us. Compared to fixed-rate codes, we have less freedom to control the minimum number of unreliable bits for each check node of RC-IRA codes. Thus, we use the RBCM scheme proposed in [88] due to its simplicity to implement and its compatibility with the RBCM scheme in [87]. And we only control the mapping of systematic bits. The Gray-labeled 16-QAM shown in Fig. 5.8 is used throughout this section. Thus, we have c_3, c_1 as reliable bits and c_2, c_0 as unreliable bits. The mother code rate we chose is 0.923. Using the variable degree distribution given in Section 4.1, $\lambda(x) = \Sigma\lambda_i x^{i-1} = 0.0599x + 0.2411x^2 + 0.2653x^7 + 0.4337x^9$, we have the variable degree distribution from the node perspective of $\Lambda(x) = \Sigma\Lambda_i x^i = 0.1437x^2 + 0.4337x^3 + 0.1833x^8 + 0.2391x^{10}$. Three simple RBCM modes are defined as follows.

Mode 0

For variable nodes with the same d_s, one half are assigned to reliable bit levels and the other half are unreliable. Before discussing RBCM, all simulations in 16-QAM used mode 0.

Mode 1

Mode 1 is supposed to provide the worst throughput. We assign reliable bit levels to variable nodes with degree 2 or 3 and unreliable bit levels to variable nodes with degree 8 or 10. For $N_m = 188$, we have 94 reliable and 94 unreliable bit levels. However, we have 25 degree-2 and 84 degree-3 variable nodes. Therefore, 15 degree-3 variable nodes will be assigned to unreliable bit levels.

Mode 2

Mode 2 is supposed to provide the best throughput. We assign reliable bit levels to variable nodes with degree 8 or 10 and unreliable bit levels to variable nodes with degree 2 or 3. For $N_m = 188$, 15 degree-3 variable nodes will be assigned to reliable bit levels.

Table 5.1 shows the distribution of N_{UBL} for a punctured strictly-right-regular RC-IRA code with $N_m = 188$ and $r = 0.423$. Table 5.2 shows the distribution for a longer punctured RC-IRA code with $N_m = 1528$ and $r = 0.427$. At such rates, $d_c = 6$ for all check nodes except one check node at one end of the accumulator. Mode 2 has the best N_{UBL} distribution for both codes.

Table 5.1: The distribution of N_{UBL} for RC-IRA code with $N_m = 188$ and $r = 0.423$.

N_{UBL}	6	5	4	3	2	1	0
Mode 0	5	19	73	71	58	26	4
Mode 1	18	74	83	54	24	3	0
Mode 2	1	4	16	57	91	70	17

Table 5.2: The distribution of N_{UBL} for RC-IRA code with $N_m = 1528$ and $r = 0.427$.

N_{UBL}	6	5	4	3	2	1	0
Mode 0	30	176	477	692	454	192	27
Mode 1	164	499	756	451	159	18	1
Mode 2	1	22	185	428	719	518	175

5.5.3 Throughput of RC-IRA Codes with RBCM

Fig. 5.20 and 5.21 compare the throughput of RC-IRA codes in the three different RBCM modes. IDM is enabled at the receiver. Compared to mode 1, mode 2 has around 0.2 dB gain at high SNR's for both N_m. However, at low SNR's mode 2 perform worse than mode 1 and mode 0. The tie point between mode 2 and mode 1 is 3 dB for $N_m = 188$ and 4 dB for $N_m = 1528$. Similar results are observed in Fig. 5.22, where IDM is disabled. In all three figures, mode 2 has no noticeable gain over mode 0. Note in our simple RBCM scheme for RC-IRA codes, we do not control parity bits. This might be the reason that mode 2 does not work as we expected. With or without IDM, mode 0 appears a robust choice for applying such a simple RBCM scheme in RC-IRA codes.

A more efficient way to increase throughput of rateless codes in multilevel modulation should be re-considering $\lambda(x)$ and $\rho(x)$, which were only optimized for the binary-input AWGN channel. In Gray-labeled multilevel modulation, component channels for each bit level are not symmetric (except the mostly protected bit level). If the symmetric condition breaks, the all-zero codeword cannot represent all codewords for density evolution analysis. Therefore, $\lambda(x)$ and $\rho(x)$ optimized for the binary-input

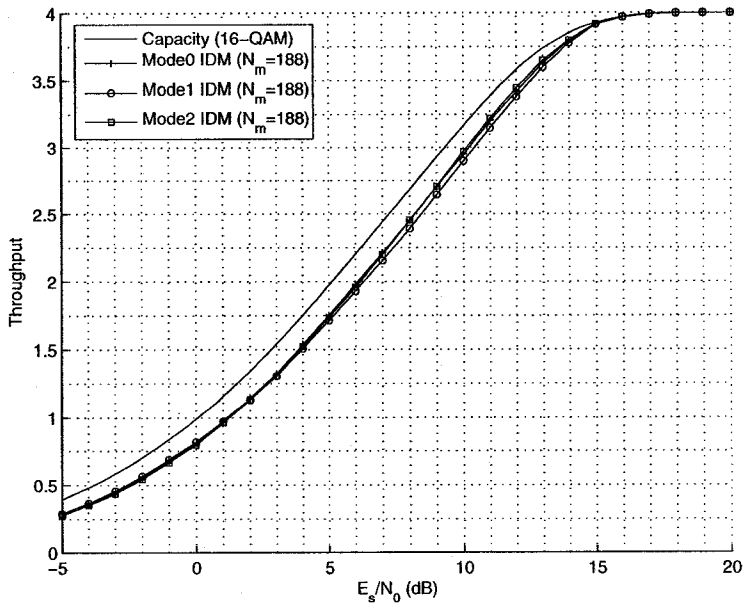

Fig. 5.20: RC-IRA codes with RBCM over 16-QAM AWGN: throughput vs. E_s/N_0 ($N_m = 188$). IDM is enabled.

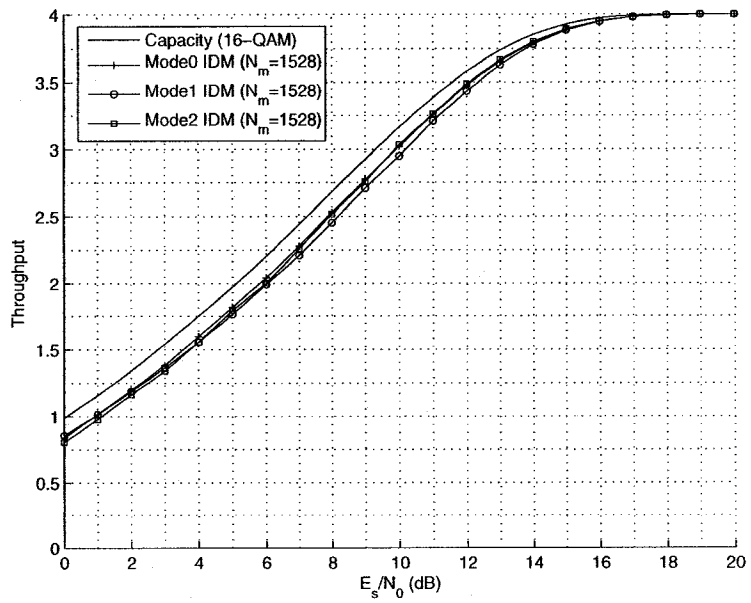

Fig. 5.21: RC-IRA codes with RBCM over 16-QAM AWGN: throughput vs. E_s/N_0 ($N_m = 1528$). IDM is enabled.

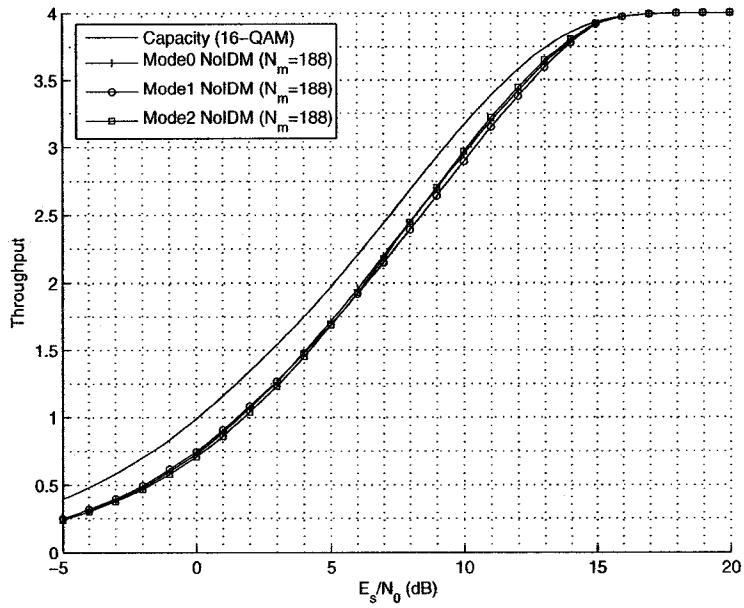

Fig. 5.22: RC-IRA codes with RBCM over 16-QAM AWGN: throughput vs. E_s/N_0 ($N_m = 188$). IDM is disabled.

AWGN channel are not necessarily optimal for multilevel modulations. This gives us the opportunity to jointly design irregular LDPC codes and multilevel modulation. A comprehensive treatment on this topic can be found in [69]. A simpler optimization method targeting shorter block length was given by Sankar *et al* in 2004 [90]. Durisi *et al* optimized eIRA codes for 4-PAM and 8-PSK in 2006, taking into account the systematic property and UEP [91]. While attempting to design RBCM schemes, we should note that a very tight coupling between coding and modulation might not be pragmatic since we cannot arbitrarily match different codes with different modulation like what conventional BICM did.

5.6 Conclusions

In this chapter, we first showed that the cap effect of Raptor codes still exists over higher-order modualtion, and then compared the throughput of RC-IRA and RC/QC-LDPC codes. Similar results are observed as in the binary-input AWGN channel -

RC-IRA codes have higher throughput than RC/QC-LDPC codes at high SNR's but slightly lower throughput at low SNR's. We then use RC-IRA codes to examine how iterative de-mapping (IDM), labeling and reliability-based coded modulation (RBCM) affect rateless codes. IDM can improve the throughput at low capacity channels (below rate 1/2). Our simulation results suggest that binary reflected Gray labeling is the best choice for RC-IRA codes. However, our simple RBCM scheme does not work as expected. A robust choice of RBCM is suggested: mapping half the bits with lower variable degree to reliable bit levels and another half to unreliable bit levels (mode 0).

Chapter 6

Frequency Flat Fading Channel

In wireless communication systems, the transmitted signal will typically propagate over multiple paths to the receiver, with the propagation delay along each path differing slightly. At the receiver, the signals along the different paths are combined, perhaps constructively, perhaps destructively, depending on the positions of the transmitter and receiver and the surrounding environment. Over time, the received signal strength varies in a phenomenon known as multipath fading.

When the spread of the delays along the different paths (the channel delay spread) is small relative to the symbol duration, frequency-flat (frequency-nonselective) fading occurs, and the received sample can be modeled as

$$r_i = h_i s_i + n_i, \tag{6.1}$$

where h_i is a random process in terms of the sample index i, and h_i, s_i, n_i are assumed independent. The AWGN channel can be actually viewed as a sub-class of the fading channel by setting h_i as 1. For better estimation on s_i, we assume the exact values of h_i are available at the receiver (and not necessarily at the transmitter). This assumption

is defined as perfect channel state information (CSI) at the receiver. By application of the central limit theorem, h_i is modeled as a wide-sense stationery (WSS) complex Gaussian random process. Channels with such h_i are termed Rayleigh fading channels. With Clarke's model, the autocorrelation function of h_i is zeroth-order Bessel function of the first kind [92, page 44-48].

6.1 Capacity of Rayleigh Fading Channel

The capacity in the Shannon sense means any rate lower than the capacity can be transmitted error-free. In a Rayleigh fading channel, each symbol experiences different instantaneous SNR, γ_i, which is an exponential random variable with expectation $\bar{\gamma}$. For a channel with varying SNR to have a capacity under which any codeword can pass through the channel error-free, the channel has to change fast enough and each codeword has to be long enough to escape from occasional deep fading. These two conditions compose our ergodicity assumption and the capacity under such assumptions is defined as the *ergodic capacity* of a fading channel. Since γ_i experienced in one codeword can reasonably well represent an ensemble of a Rayleigh random variable, the ergodic capacity of a Rayleigh fading channel, C_{erg} is simply given by taking the expectation over γ_i for the capacity of an AWGN channel with the same input alphabet, either unconstrained or constrained. For example, if we use $C_{UC}(\gamma_i)$ to denote the capacity of an AWGN channel with unconstrained input alphabet and SNR γ_i, C_{erg} with unconstrained alphabet is given by

$$\begin{aligned} C_{erg}(\bar{\gamma}) =& \mathbf{E}[C_{UC}(\gamma_i)] = \mathbf{E}[\log_2(1+\gamma_i)] \\ =& \mathbf{E}[\log_2(1+|h_i|^2\bar{\gamma})] \qquad \text{(bit/dimension pair)}, \end{aligned} \tag{6.2}$$

where $\mathbf{E}[.]$ indicates the expectation operation and $|h_i|$ is a Rayleigh random variable. By a simple change of variable, Eq. (6.2) can be related to the exponential integral, $E_1(x) := \int_x^\infty \frac{e^{-t}}{t}\, dt$ [93], by the following equality,

$$C_{erg}(\bar{\gamma}) = \frac{1}{\ln 2}\, e^{1/\bar{\gamma}}\, E_1(\frac{1}{\bar{\gamma}}). \tag{6.3}$$

For a constrained alphabet, the ergodic capacity can be evaluated via numerical integration or the Monte-Carlo method. Note we have to numerically evaluate a double integral here. Under BPSK modulation, we can avoid this by using a simpler and more accurate method with infinite series from [94].

For a non-ergodic Rayleigh fading channel, γ_i does not change so fast and a fixed-rate codeword may be stuck in a deep fading and cause a word error. An extreme example of non-ergodic fading channel is block-fading channel, where h_i remains constant for all symbols in any block. Now i becomes the block index. For a block-fading channel, given any positive capacity value C' and any codebook with a rate $r' < C'$, there always exists a positive $\gamma_{out}(r')$ such that when $0 \leq \gamma_i \leq \gamma_{out}(r')$, codewords will still cause word errors. Since the probability of the event $\{0 \leq \gamma_i \leq \gamma_{out}\}$ is not zero for a positive $\gamma_{out}(r')$, the concept of capacity is not suitable for a non-ergodic Rayleigh fading channel. Instead, we study outage probability $\mathrm{P}_{out}(\rho, \bar{\gamma})$ for a given *minimal rate*, ρ, and an average SNR, $\bar{\gamma}$ (or equivalently, we can study *outage capacity* for a given outage probability, ε, and $\bar{\gamma}$). The *outage probability* is defined as

$$\mathrm{P}_{out}(\rho, \bar{\gamma}) := \mathrm{P}(\mathrm{C}_{\mathrm{AWGN}}(\gamma_i) \leq \rho), \tag{6.4}$$

where $\mathrm{C}_{\mathrm{AWGN}}(\gamma_i)$ is the capacity of the i^{th} block over the AWGN channel. Here we require ρ for each block. If we relax the requirement to requiring ρ for the average of

multiple blocks, a lower P_{out} is expected [95, page 105]. The simulation results shown in this chapter are based on one-block requirement and therefore represent the largest outage probability. For such a definition of P_{out}, we can easily use the capacity over the AWGN channel to evaluate the theoretic $P_{out}(\rho, \bar{\gamma})$ over block-fading channels.

Given a mobile communication system with only two users communicating to each other, it appears that fast time-varying channels are preferable to slowly time-varying channels, since the former allows us to achieve the ergodic capacity. However, the receiver needs to know exact values of all h_i's to achieve this capacity. This requirement is not practical, not to mention that fast fading channels are rare in most mobile communication systems. For a practical transmitter, we usually put an interleaver after the encoder such that the risk of deep fading is shared by multiple blocks. Now each block can have the chance to achieve the ergodic capacity, provided the channel estimation at the receiver is good enough and the depth of interleaver is much larger than the *coherence time* of the channel. We refer to such a situation as a fully-interleaved fading channel. This ideal channel incurs increased delay and requires large memory. So in a practical mobile system, we still need to deal with slow-fading channels, where an interleaver with finite-depth can improve the throughput to some extent. Therefore, in this chapter we will present simulation results of rateless codes in fast-fading channels as well as slow-fading channels. Readers are referred to [95, chapter 4.3] and [96, page 2627-2633] for more discussions on the capacity of fading channels.

6.2 RC-IRA and RC/QC-LDPC codes in Rayleigh Fading Channel

There is no difference between applying rateless codes over a Rayleigh fading channel and over an AWGN channel. However, we assumed that perfect CSI is available at the receiver. So the BP decoder can take advantage of the CSI knowledge to improve the performance by changing the calculation of LLR_{ini}'s. Let us assume 16-QAM is used for modulation. Given the i^{th} complex-valued received sample $r_i = h_i s_i + n_i$, we multiply both sides by $h_i^*/|h_i|$ to compensate the phase rotation of the i^{th} transmitted symbol s_i, where h_i^* is the conjugate and $|h_i|$ is the modulus of h_i. Now we have

$$r_i' = r_i h_i^*/|h_i| = |h_i| s_i + n_i h_i^*/|h_i|.$$

Letting $h_i = |h_i| e^{j\theta_i}$, we have

$$\begin{aligned} r_i' &= |h_i| s_i + n_i e^{-j\theta_i} \\ &= |h_i| s_i + (n_{xi} \cos\theta_i + n_{yi} \sin\theta_i) + j(-n_{xi} \sin\theta_i + n_{yi} \cos\theta_i) \\ &= |h_i| s_i + n_{xi}' + j n_{yi}', \end{aligned} \tag{6.5}$$

where n_{xi}' denotes the real part of the noise component in r_i' and n_{yi}' is the imaginary part. When the receiver calculates LLR_{ini} for a given r_i, $|h_i|$ and θ_i are taken as known due to perfect CSI. Thus, we can easily show that both n_{xi}' and n_{yi}' are still zero-mean Gaussian random variable with the same variance, $N_0/2$, as n_{xi} and n_{yi}. Also, it can be shown that n_{xi}' and n_{yi}' are uncorrelated. Since n_{xi} and n_{yi} are jointly normal, we know their linear combinations are also jointly normal [32, page 202]. It follows that n_{xi}' and n_{yi}' are independent due to the zero autocorrelation coefficient. With

these results, simply replacing $|h_i|s_i$ into s_i, we can transform the LLR formulae for the AGWN channel into the LLR formulae for the Rayleigh fading channel. Eq. (6.6) gives the LLR formulae for the Rayleigh fading channel with Gray-labeled 16-QAM.

$$\begin{aligned}
\mathrm{LLR}_{ini}(\mathrm{c}_0) &= \ln\left(\frac{e^{-4\frac{h(r_x+2h)}{N_0}}+e^{-8\frac{h(-r_x+h)}{N_0}}}{1+e^{4\frac{r_x h}{N_0}}}\right),\\
\mathrm{LLR}_{ini}(\mathrm{c}_1) &= \ln\left(\frac{e^{-8\frac{r_x h}{N_0}}+e^{4\frac{h(-r_x+2h)}{N_0}}}{e^{8\frac{h^2}{N_0}}+e^{4\frac{r_x h}{N_0}}}\right),\\
\mathrm{LLR}_{ini}(\mathrm{c}_2) &= \ln\left(\frac{e^{-8\frac{h(-r_y+h)}{N_0}}+e^{-4\frac{h(r_y+2h)}{N_0}}}{e^{4\frac{r_y h}{N_0}}+1}\right),\\
\mathrm{LLR}_{ini}(\mathrm{c}_3) &= \ln\left(\frac{e^{4\frac{h(-r_y+2h)}{N_0}}+e^{-8\frac{r_y h}{N_0}}}{e^{4\frac{r_y h}{N_0}}+e^{8\frac{h^2}{N_0}}}\right),
\end{aligned} \tag{6.6}$$

where h denotes $|h_i|$, r_x denotes $\mathbf{Re}\{r_i'\}$, and r_y denotes $\mathbf{Im}\{r_i'\}$. The symbol index i is dropped for more elegant expressions.

If BPSK modulation is used, r_i is equal to $h_i s_i + n_i$, where $s_i \in \{1, -1\}$ and others are complex values. Given h_i and s_i, r_i is a complex Gaussian random variable with the mean $h_i s_i$. Thus, we have

$$\mathrm{LLR}_{ini}(\mathrm{c}_i) = \ln\frac{\mathrm{P}(r_i|s_i=-1)}{\mathrm{P}(r_i|s_i=1)} = \ln\frac{e^{-\frac{|r_i+h_i|^2}{N_0}}}{e^{-\frac{|r_i-h_i|^2}{N_0}}} = -\frac{4}{N_0}\mathbf{Re}\{r_i h_i^*\}. \tag{6.7}$$

If we take the real part of r_i' in Eq. (6.5) to calculate the LLR, we have $\mathrm{LLR}_{ini}(\mathrm{c}_i) = -\frac{4}{N_0}\mathbf{Re}\{r_i'\}|h_i|$. This is the same as in Eq. (6.7).

One advantage of phase compensation is to avoid calculating $h_i s_i$ for each given h_i and all signal points. Gray-labeled square-QAM can still be decomposed into two PAM constellations. Compared to Eq. (5.5), the LLR formulae in Eq. (6.6) almost do not change. To a certain extent, we have reduced the fading-incurred complexity in the LLR calculations.

6.3 Simulation Results

6.3.1 Simulation Models

The colored complex Gaussian random process h_i is generated from the original Jakes method [97], as given by

$$h_i = K\left\{\frac{1}{\sqrt{2}}e^{j\alpha}\cos(iB_dT_s+\theta_0)+\sum_{n=1}^{N_r}[e^{j\beta_n}\cos(iB_dT_s\cos\alpha_n+\theta_n)]\right\}, \tag{6.8}$$

where B_d is the *maximum Doppler shift*, i is the symbol index, T_s is the symbol duration, θ_0 and θ_n are initial phases. We use K as a normalization constant such that $\mathbf{E}[|h_i|^2]=1$. The number of rays is equal to $4N_r+2$; the uniformly distributed arrival angle α_n is given by $2n\pi/(4N_r+2)=n\pi/(2N_r+1)$. In our simulations, N_r is 8. We choose $\alpha=0$ and $\beta_n=n\pi/(N_r+1)$ to ensure zero cross-correlation between real and imaginary parts of h_i. This gives us $K=2/\sqrt{2N_r+1}$. Since we only need one h_i per symbol by assuming no ISI, our simulations are not affected by the difficulty of generating multiple uncorrelated waveforms in this deterministic model [98].

It can be observed in Eq. (6.8) that an infinite B_dT_s simulates a fully-interleaved fading channel, while $B_dT_s=0$ provides a block-fading channel - h_i varies with blocks instead of symbols. Neither case is realistic. Supposing a mobile terminal moving at 100 km/hr is receiving signals with the frequency 2.4 GHz and the symbol rate is 1 MBaud, B_dT_s is given by $\frac{2.4\times10^9\times(10^5/3600)}{3\times10^8}\times10^{-6}=2.2\times10^{-4}$. If the speed decreases to 5 km/hr, B_dT_s becomes 10^{-5}. So in our simulations, we use $B_dT_s=10^{-4}$ and 10^{-5} to simulate slow-fading channels, and $B_dT_s=10^{-1}$ to approximate fully-interleaved fading channels. The latter setting gives us $T_s=0.1/B_d\approx0.1T_{coh}$, where T_{coh} is the channel coherence time.

Simulations of rateless codes in slow-fading channels are very time-consuming. We use two techniques to increase simulation efficiency. The first one is estimating average SNR by calculating the average of $|h_i|^2$ over all received samples. We then use the Shannon capacity for the AWGN channel to determine the decoding start point. Another one is the early-abort (EA) technique [60], which is based on the assumption that if a given block can be decoded successfully in a BP decoder, the sum of absolute values of soft decisions for all information bits, $\mathrm{AS_{SD}}$, will monotonically increase while iterating. Denoting $\mathrm{AS_{SD}}$ after l iterations as $\mathrm{AS_{SD}^{(l)}}$, for each iteration we compare the $\mathrm{AS_{SD}^{(l)}}$ with $\mathrm{AS_{SD}^{(l-L_{EA})}}$. If the former is smaller, the decoding aborts and waits for the next decoding attempt. L_{EA} is a constant chosen as 10 as used in [60]. Although the EA technique provides slightly worse simulation results than without EA, it can greatly accelerate the simulations. We use EA for $N_m \approx 1528$ over slow-fading channels and disable EA for other simulation scenarios.

Simulations in fast-fading channels start at the Shannon capacity point in AWGN channel. If we assume the ARR over fast-fading channels is close to the ARR over ergodic Rayleigh fading channels, it is possible to use the ARR in AWGN channels to estimate ARR in fast-fading channels without simulations. The principle is similar to calculating the ergodic capacity from the capacity in AWGN channel, which was shown in Eq. (6.2). The estimation of ARR is given by

$$\begin{aligned} \hat{\mathrm{ARR}}_{erg}(\bar{\gamma}) &= \mathbf{E}[\mathrm{ARR_{AWGN}}(\gamma)] \\ &= \int_0^\infty \frac{1}{\bar{\gamma}}\, e^{-\gamma/\bar{\gamma}} \mathrm{ARR_{AWGN}}(\gamma) \mathrm{d}\gamma, \end{aligned} \tag{6.9}$$

where $\mathrm{ARR_{AWGN}}(\gamma)$ is approximated by applying linear interpolation on the ARR curves we have simulated in AWGN channels (Fig. 4.2, 4.3, 5.9 and 5.10).

In the following, we present the comparison between RC-IRA codes and RC/QC-LDPC codes over Rayleigh fading channels. Modulation schemes include BPSK and Gray-labeled 16-QAM.

6.3.2 Throughput over Fast Rayleigh Fading Channels

We use $B_d T_s = 10^{-1}$ to represent fast Rayleigh fading channels. Fig. 6.1, 6.2, 6.3 and 6.4 show the throughput against average E_s/N_0 with BPSK and Gray-labeled 16-QAM. The throughput of RC-IRA codes is higher than RC/QC-LDPC codes at high SNR's. The reason is the throughput cap of RC/QC-LDPC codes. At low SNR's, RC-IRA codes perform slightly worse than RC/QC-LDPC codes. However, if we only use the BP algorithm to decode RC-IRA codes, the difference becomes around 1.2 dB at rate 1/2. The similar results are observed in 16-QAM.

We also observe that for RC-IRA codes modulated with 16-QAM, little gain is obtained by enabling IDM and using BP and BCJR algorithms. If BPSK is used, however, BP and BCJR algorithms can help RC-IRA codes achieve higher throughput when $E_s/N_0 < 10$dB.

Fig. 6.5 shows the difference between $\hat{\text{ARR}}_{erg}$, estimated from ARR_{AWGN} by Eq. (6.9), and ARR_{erg}, calculated from simulations in fast-fading channels. Codes with $N_m = 188$ or 192 are plotted. It can be observed that $\hat{\text{ARR}}_{erg}$ fits ARR_{erg} very well at low SNR's. However, at high SNR's $\hat{\text{ARR}}_{erg}$ is higher than ARR_{erg} because the ergodicity assumption cannot hold - it takes only a small number of symbols to successfully decode at high SNR's.

6.3.3 Outage Probability over Slow Rayleigh Fading Channels

Using WER to approximate P_{out}, we plot $P_{out}(\rho, \bar{\gamma})$ against ρ and $\bar{\gamma}$ ranging from 2 to 18dB for BPSK, and 4 to 16 dB for 16-QAM. RC-IRA codes have $N_m = 188$ and 1528; RC/QC-LDPC codes have $N_m = 192$ and 1536. The channels have $B_dT_s = 10^{-4}$ or 10^{-5}. Totally we have eight simulation scenarios shown from Fig. 6.6 to Fig. 6.13.

The outage probability when $B_dT_s = 10^{-5}$ are shown from Fig. 6.6 to 6.9. Regardless of the modulation and the message word length, P_{out} of RC-IRA codes is slightly higher than RC/QC-LDPC codes; however, the latter still suffers the throughput cap. Note that block-fading channels have $B_dT_s = 0$, which is close to the B_dT_s values we are using. We therefore use the theoretic P_{out} of block-fading channels as an approximate lower bound of P_{out} in slow-fading channels. Both codes are closer to the P_{out} of BPSK block-fading channels than the P_{out} of 16-QAM block-fading channels. This results from using similar SNR's in both modulations - codes in BPSK achieve the higher rate than in 16-QAM. Another reason is that BPSK uses more symbols than 16-QAM and appears more like fast-fading channels.

Since a larger B_dT_s always produces a lower P_{out}, it is expected that simulated P_{out} can go below the theoretic P_{out} of block-fading channels, as shown in Fig. 6.7. We also notice that disabling the BCJR algorithm or IDM in RC-IRA decoding only causes a small penalty on the P_{out}.

When we change B_dT_s as 10^{-4}, as shown from Fig. 6.10 to Fig. 6.13, we observe similar results as when B_dT_s is 10^{-5}. Note that Fig. 6.10 and Fig. 6.7 are very similar. So are Fig. 6.12 and Fig. 6.9. The reason is that a larger N_m can shift a slow-fading channel towards a fast-fading channel. If changing N_m and M does not significantly change code performance, we can use $\frac{B_dT_sN_m}{log_2M}$ to classify the fading channels we are using, in terms of time-varying property. The P_{out} values can then be expected in the

following ascending order: $\{N_m = 1528, B_dT_s = 10^{-4}\}$, $\{N_m = 188, B_dT_s = 10^{-4}\}$, $\{N_m = 1528, B_dT_s = 10^{-5}\}$, and $\{N_m = 188, B_dT_s = 10^{-5}\}$. We can observe that simulated P_{out}'s in first two categories, shown in Fig. 6.10 to 6.13, often go below the theoretic P_{out} of block-fading channels, provided $\bar{\gamma}$ is high. The channel with parameter set $\{N_m = 1528, B_dT_s = 10^{-4}, \text{BPSK}\}$ is expected with lowest P_{out}, as shown in Fig. 6.11.

6.3.4 Discussions on ARR over Slow Rayleigh Fading Channels

Previously we mentioned that ARR is not suitable for the evaluation of code performance over slow-fading channels. In fact, if we use Eq. (4.3) to evaluate ARR over Rayleigh block-fading channels, the ARR would be asymptotically zero because the denominator in Eq. (4.3) would approach infinity. This can be easily shown below. If M-ary modulation is used, the denominator is given by

$$\begin{aligned}
\frac{1}{N_w}\sum_{i=1}^{N_w} N_s(i) &= \frac{1}{N_w}\sum_{i=1}^{N_w}\frac{\log_2 M}{\mathrm{RR}(i)} \geq \frac{1}{N_w}\sum_{i=1}^{N_w}\frac{\log_2 M}{C_{UC}(\gamma_i)} \\
&\approx \int_0^{\infty}\frac{\log_2 M}{\log_2(1+\gamma)}\frac{1}{\bar{\gamma}}\,e^{-\gamma/\bar{\gamma}}\,\mathrm{d}\gamma > \int_0^{\sigma}\frac{\log_2 M}{\log_2(1+\gamma)}\frac{1}{\bar{\gamma}}\,e^{-\sigma/\bar{\gamma}}\,\mathrm{d}\gamma \\
&= \frac{\ln M}{\bar{\gamma}}\,e^{-\sigma/\bar{\gamma}}\int_0^{\sigma}\frac{1}{\ln(1+\gamma)}\,\mathrm{d}\gamma > \frac{\ln M}{\bar{\gamma}}\,e^{-\sigma/\bar{\gamma}}\int_0^{\sigma}\frac{1}{\gamma}\,\mathrm{d}\gamma \\
&= \infty,
\end{aligned} \tag{6.10}$$

where $\mathrm{RR}(i)$ denotes the realized rate for i^{th} block, and σ is a finite positive number. Motivated by the above analysis, we know the difficulty of using Eq. (4.3) to calculate the ARR for slow-fading channels arises when we have very large N_c. Since in simulations we have to set a finite $N_c^{(max)}$ to ensure simulation efficiency, decoding failures will occur if the $\bar{\gamma}$ is not high enough. Therefore, Eq. (4.3) can be modified to accommodate decoding failures. One solution is to calculate *conditional ARR* (CARR):

only blocks with realized rates higher than a pre-set rate are calculated towards ARR. Certainly, the pre-set rate, r_{fail} should be larger than r_{min} $(= N_m/N_c^{(max)})$. However, this solution gave overly optimistic results due to completely ignoring the effect of decoding failures on averaged throughput. We use another solution to provide more sensible averaged throughput, *modified ARR* (MARR). Denoting the *decoding failure rate* as DFR, MARR is given by

$$\text{MARR} = \frac{N_w(1-\text{DFR})\cdot N_m}{\sum_{i=1}^{N_w(1-\text{DFR})} \frac{N_m}{\text{RR}(i)} + N_w\cdot\text{DFR}\cdot\frac{N_m}{r_{fail}}} = \frac{(1-\text{DFR})}{\frac{1}{N_w}\sum_{i=1}^{N_w(1-\text{DFR})}\frac{1}{\text{RR}(i)} + \frac{\text{DFR}}{r_{fail}}}, \quad (6.11)$$

where N_w is the number of blocks transmitted and $\text{RR}(i)$ denotes the realized rate of the i^{th} codeword. Using Eq. (6.11), we plot the modified throughput against average E_s/N_0 in Fig. 6.14, 6.15, 6.16 and 6.17. We observe that RC-IRA codes provide larger MARR at high $\bar{\gamma}$'s but lower MARR at low $\bar{\gamma}$'s, regardless of N_m and modulations. The SNR difference between two codes over low capacity channels are marked in the figures. The difference increases when B_dT_s decreases because RC/QC-LDPC codes have less decoding failures than RC-IRA codes at slow-fading channels. We also found that the difference in fast-fading channel is not sensitive on N_m while the difference in slow-fading channel increases with N_m. This suggests RC/QC-LDPC codes can approach the capacity better than RC-IRA codes at low SNR's, so when N_m increases, RC/QC-LDPC codes improve better than RC-IRA codes.

6.4 Conclusions

In this chapter, we compared the throughput and outage probability of RC-IRA codes and RC/QC-LDPC codes over frequency-nonselective Rayleigh fading channels. Similar to what we observed in AWGN channel, in either fast-fading or slow-fading chan-

nels, RC-IRA codes outperform RC/QC-LDPC codes at high SNR's and perform worse at low SNR's.

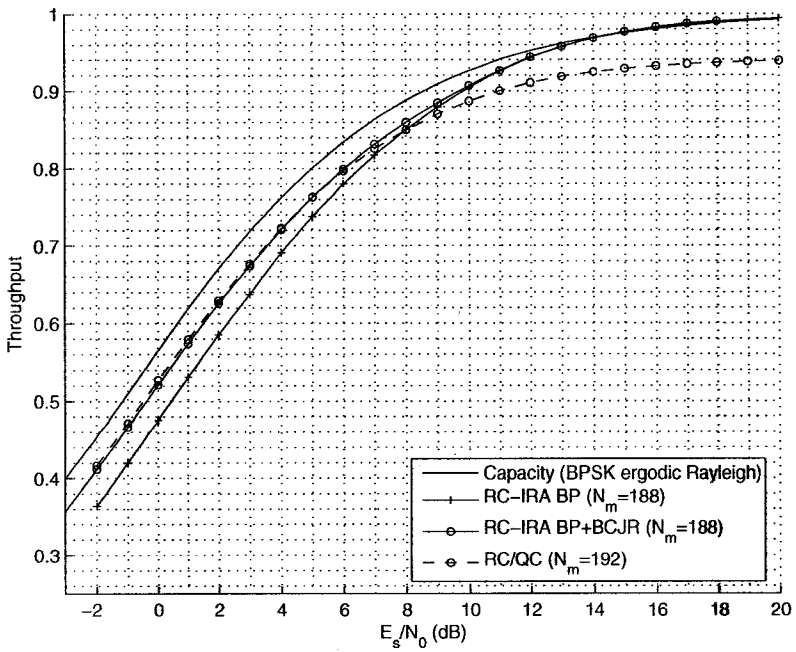

Fig. 6.1: Throughput vs. E_s/N_0 over Rayleigh fading channel with BPSK modulation. $B_dT_s = 10^{-1}$. $N_m \approx 188$.

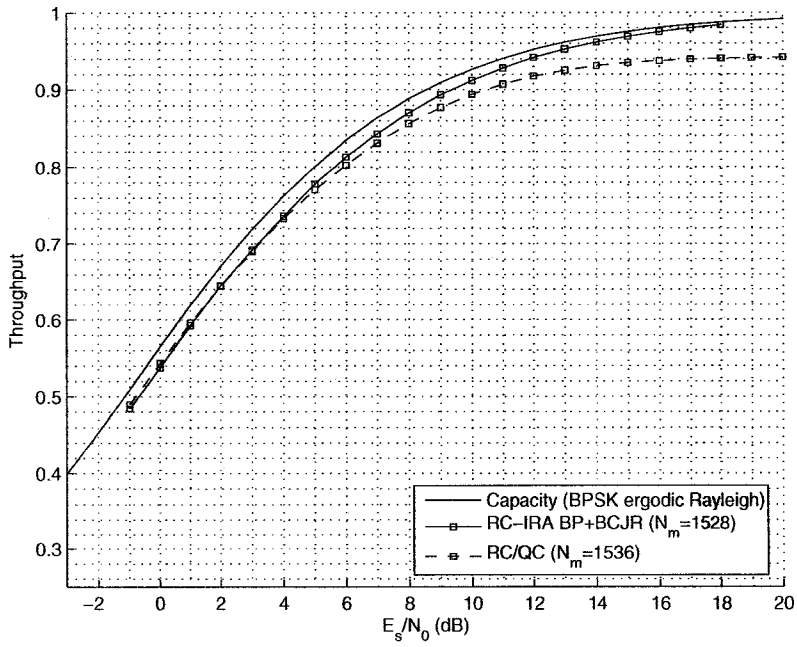

Fig. 6.2: Throughput vs. E_s/N_0 over Rayleigh fading channel with BPSK modulation. $B_dT_s = 10^{-1}$. $N_m \approx 1528$.

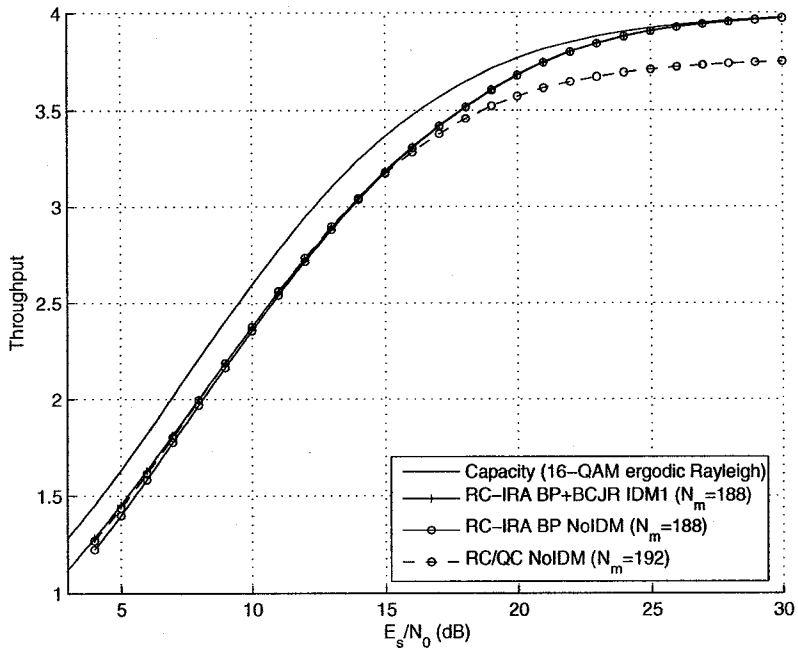

Fig. 6.3: Throughput vs. E_s/N_0 over Rayleigh fading channel with Gray-labeled 16-QAM modulation. $B_dT_s = 10^{-1}$. $N_m \approx 188$.

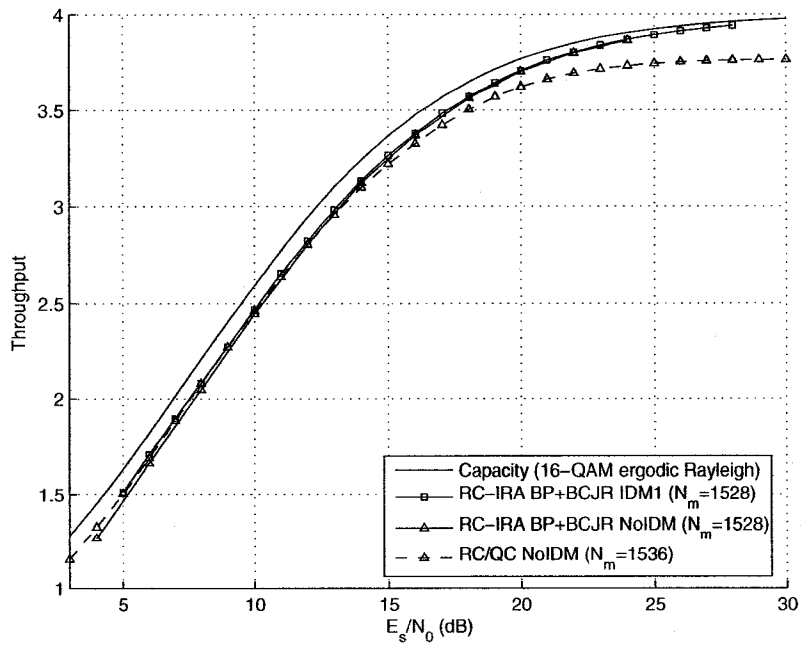

Fig. 6.4: Throughput vs. E_s/N_0 over Rayleigh fading channel with Gray-labeled 16-QAM modulation. $B_dT_s = 10^{-1}$. $N_m \approx 1528$.

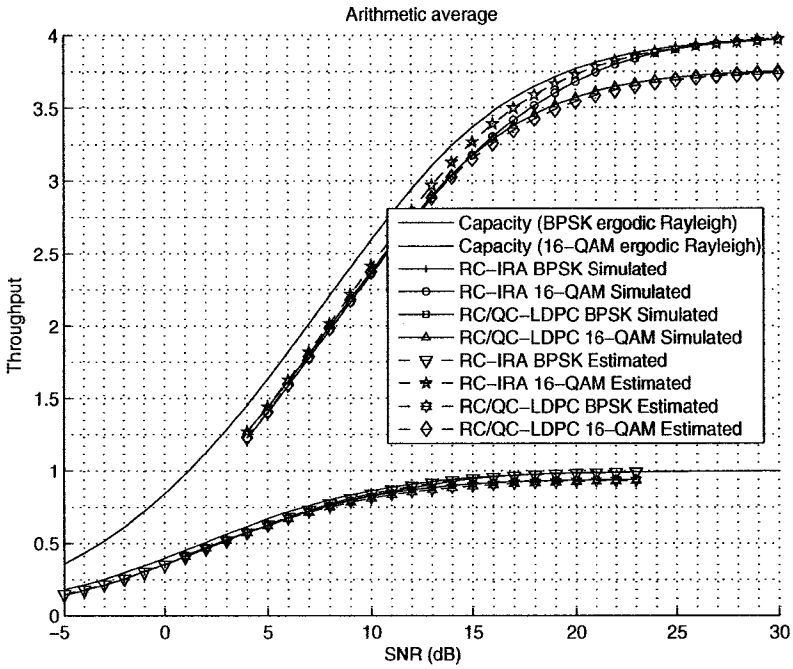

Fig. 6.5: Comparison of $A\hat{R}R_{erg}$ and ARR_{erg}. $B_dT_s = 10^{-1}$. $N_m \approx 188$.

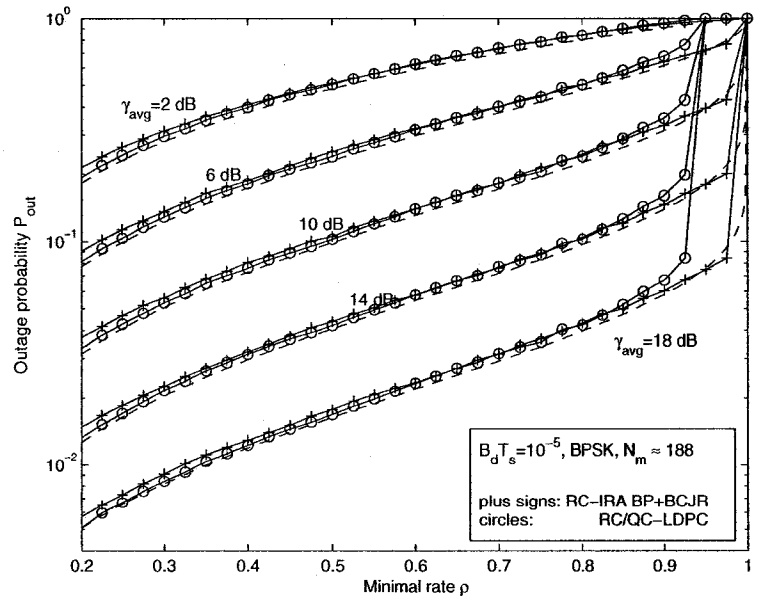

Fig. 6.6: Outage probability P_{out} vs. minimal rate ρ. $B_dT_s = 10^{-5}$. $N_m \approx 188$. The modulation is BPSK. Dash lines are theoretic P_{out} for block-fading channels.

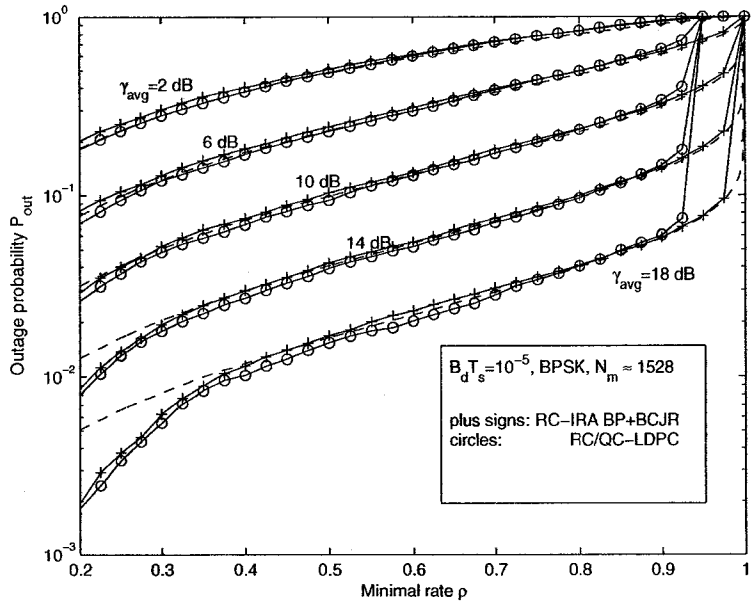

Fig. 6.7: Outage probability P_{out} vs. minimal rate ρ. $B_dT_s = 10^{-5}$. $N_m \approx 1528$. The modulation is BPSK. Dash lines are theoretic P_{out} for block-fading channels.

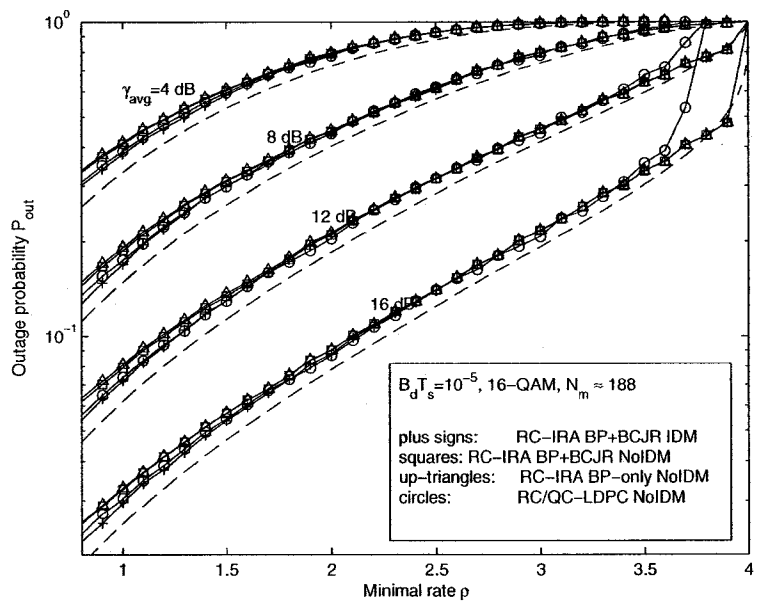

Fig. 6.8: Outage probability P_{out} vs. minimal rate ρ. $B_dT_s = 10^{-5}$. $N_m \approx 188$. The modulation is 16-QAM. Dash lines are theoretic P_{out} for block-fading channels.

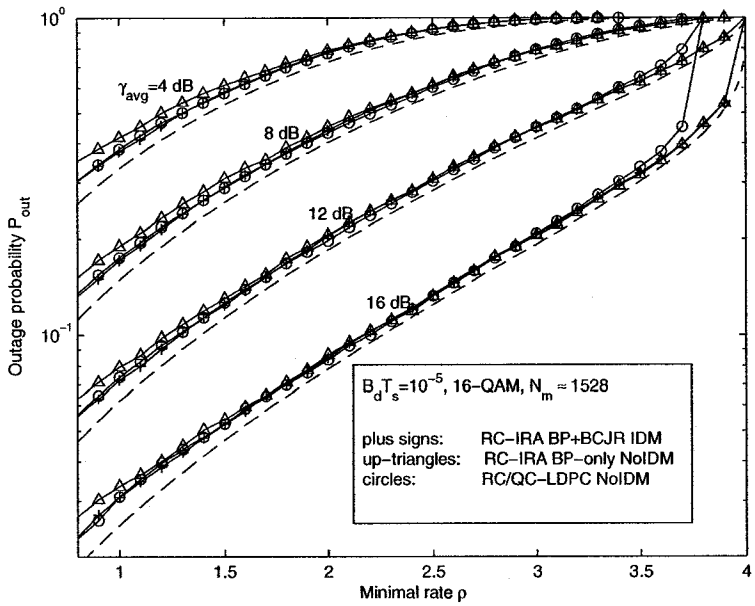

Fig. 6.9: Outage probability P_{out} vs. minimal rate ρ. $B_dT_s = 10^{-5}$. $N_m \approx 1528$. The modulation is 16-QAM. Dash lines are theoretic P_{out} for block-fading channels.

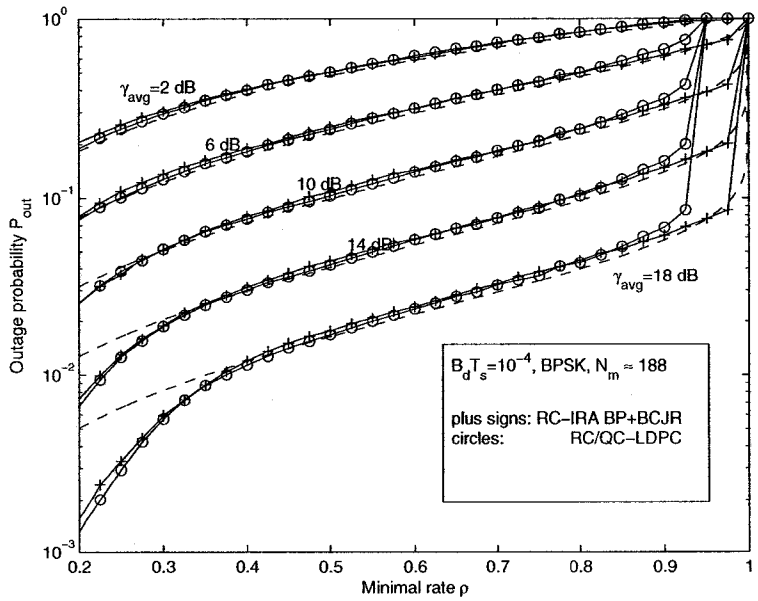

Fig. 6.10: Outage probability P_{out} vs. minimal rate ρ. $B_dT_s = 10^{-4}$. $N_m \approx 188$. The modulation is BPSK. Dash lines are theoretic P_{out} for block-fading channels.

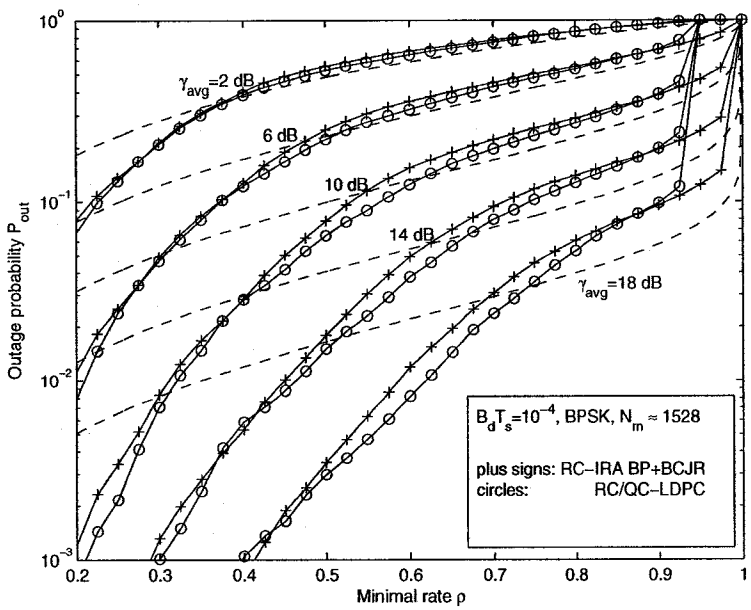

Fig. 6.11: Outage probability P_{out} vs. minimal rate ρ. $B_dT_s = 10^{-4}$. $N_m \approx 1528$. The modulation is BPSK. Dash lines are theoretic P_{out} for block-fading channels.

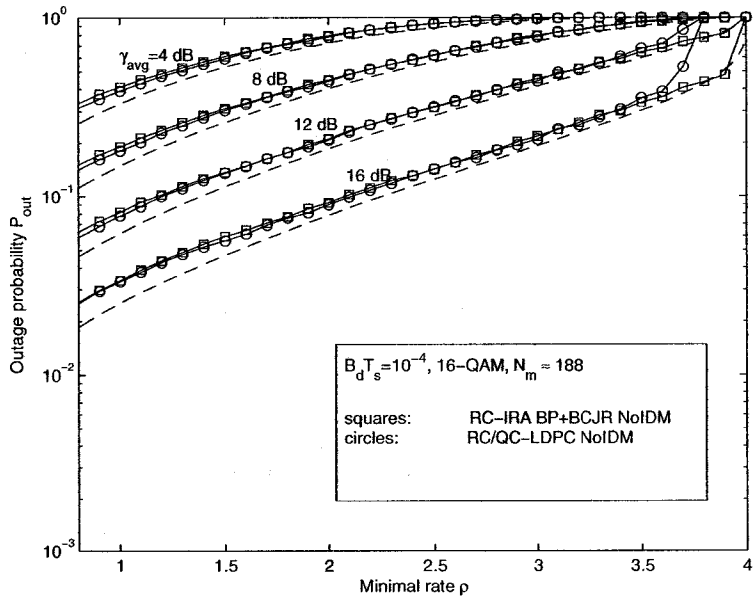

Fig. 6.12: Outage probability P_{out} vs. minimal rate ρ. $B_dT_s = 10^{-4}$. $N_m \approx 188$. The modulation is 16-QAM. Dash lines are theoretic P_{out} for block-fading channels.

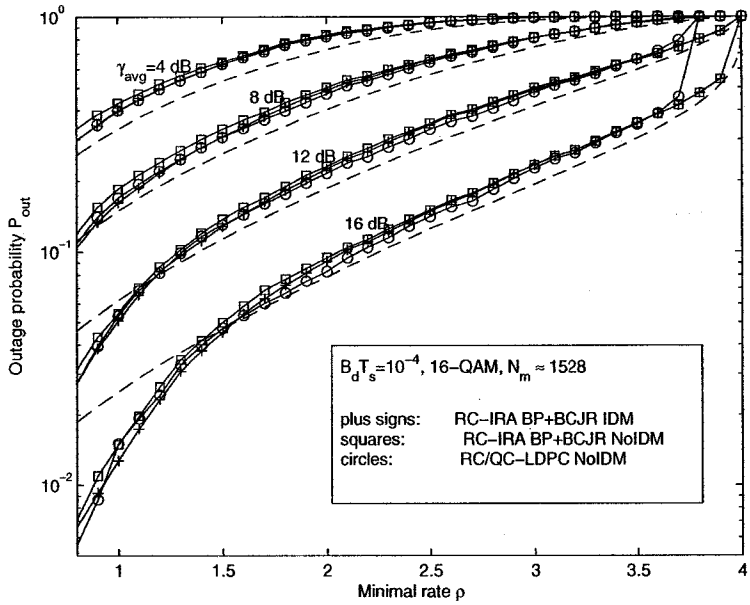

Fig. 6.13: Outage probability P_{out} vs. minimal rate ρ. $B_dT_s = 10^{-4}$. $N_m \approx 1528$. The modulation is 16-QAM. Dash lines are theoretic P_{out} for block-fading channels.

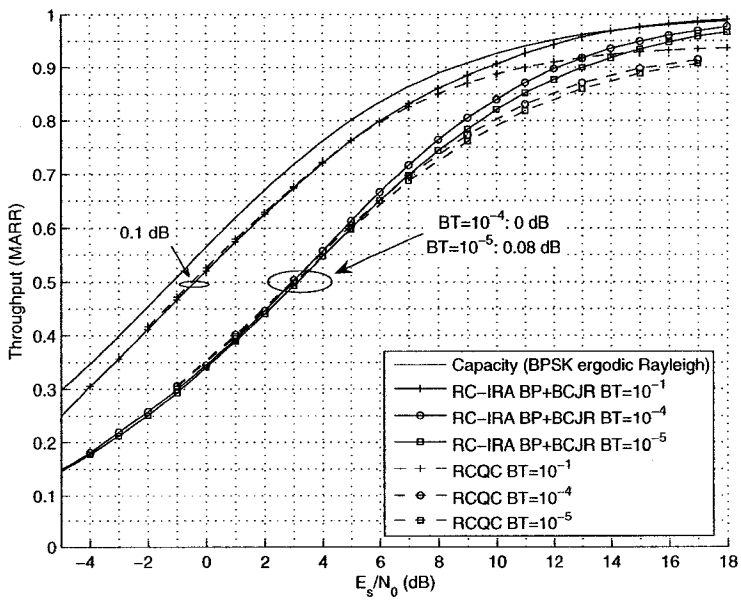

Fig. 6.14: Throughput (MARR) vs. E_s/N_0 over Rayleigh fading channel with BPSK modulation ($N_m \approx 188$).

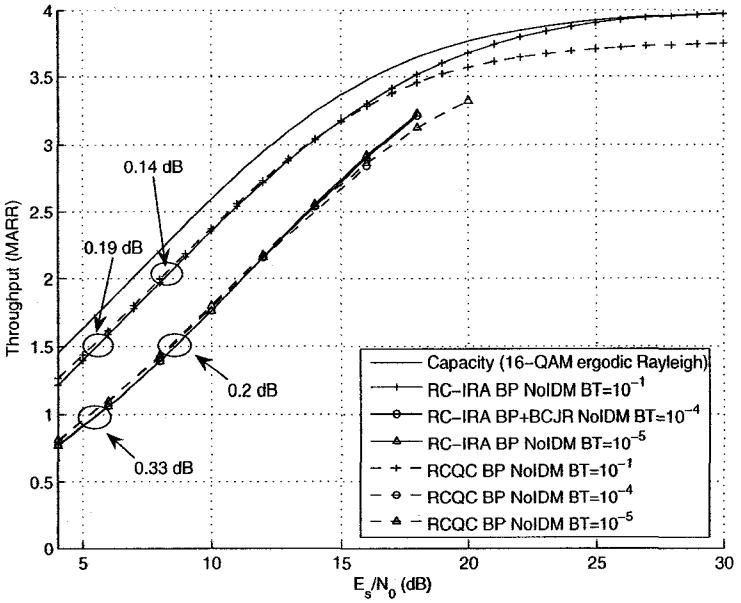

Fig. 6.15: Throughput (MARR) vs. E_s/N_0 over Rayleigh fading channel with 16-QAM modulation ($N_m \approx 188$).

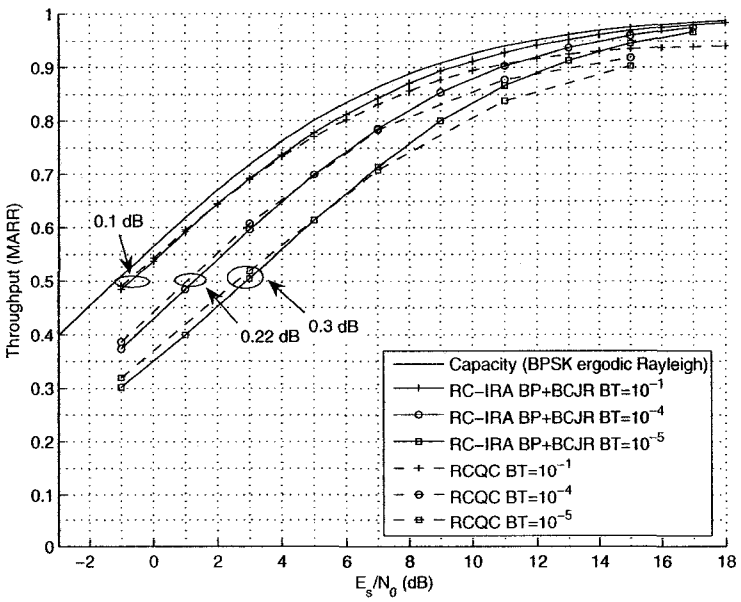

Fig. 6.16: Throughput (MARR) vs. E_s/N_0 over Rayleigh fading channel with BPSK modulation ($N_m \approx 1528$).

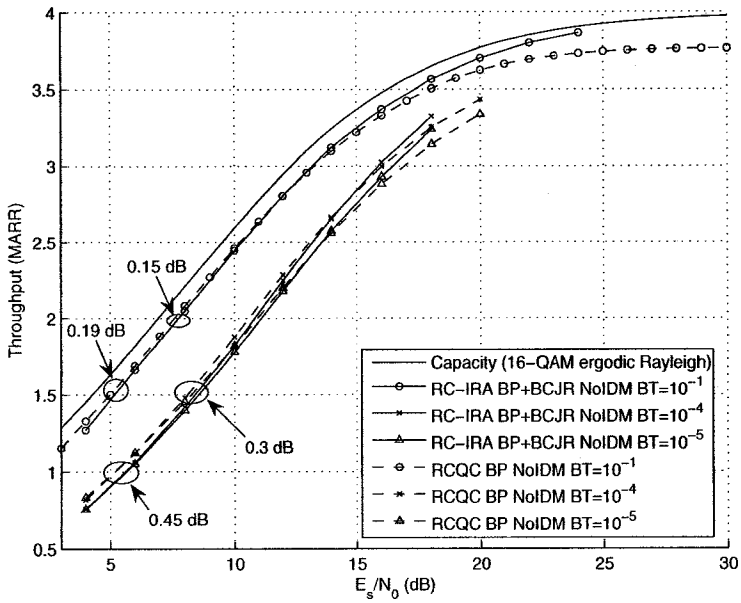

Fig. 6.17: Throughput (MARR) vs. E_s/N_0 over Rayleigh fading channel with 16-QAM modulation ($N_m \approx 1528$).

Chapter 7

Conclusions

7.1 Contributions

Our contribution is comparing the performance of three rateless codes, namely, Raptor codes, RC-IRA codes, and RC/QC-LDPC codes proposed in 3GPP2 and 802.20 standards. The comparison is focused on short and moderate message word lengths under BPSK and higher-order modulations. We first investigated the throughput of Raptor codes and RC-IRA codes with short, moderate and large block lengths over the binary-input AWGN channel. We also compared them with RC/QC-LDPC codes in short and moderate block lengths, since algebraically constructed codes should work better at short or moderate block lengths. The simulation results show that RC-IRA and RC/QC-LDPC codes have higher throughput than Raptor codes at moderate to high SNR's but RC-IRA codes have slightly worse throughput at low SNR's, regardless of the message word size. The throughput of RC-IRA codes is higher than RC/QC-LDPC codes at high SNR's but slightly lower at low SNR's. Raptor codes perform worse in high capacity channels because the LT G-graph does not cover all pre-code bits in a timely manner. RC/QC-LDPC codes proposed in 3GPP2 and

802.20 standards are punctured from a low rate 0.5 and likely have bigger gap from the capacity at high SNR's than at low SNR's.

The slight advantage of RC-IRA and RC/QC-LDPC codes over RC-IRA codes at low SNR's is related to the particular dual-diagonal structure in the parity-check matrix of RC-IRA codes, $\mathbf{H}_{IRA}$. At low SNR's, forcing the dual-diagonal causes lower density of $\mathbf{H}_{IRA}$ than RC/QC-LDPC codes, and therefore less protection over code bits. With the message word lengths we are using, the deterministic construction of the parity-check matrix of RC/QC-LDPC codes, $\mathbf{H}_{RCQC}$, also contributes the good performance of their mother codes.

To increase the throughput of RC-IRA codes at low SNR's, we attempted selective check splitting and observed that the error floor of RC-IRA codes can be lowered. The throughput, however, is not noticeably increased. This suggests that to increase the throughput of rateless codes, we should focus on how to reduce the word error rate (WER) when only a small number of code bits have been transmitted. We also show that for short block lengths, the throughput of RC-IRA codes is not particularly sensitive to the mother code rate, the BP algorithm scheduling, the existence of parallel edges during check node combining, and the variable degree distribution (as long as the average left degree is not too low).

In AWGN channel with higher-order modulations, the throughput of Raptor codes is first investigated. We demonstrated that Raptor codes with 16-QAM and 64-QAM work well over a wide range of SNR, but still have the throughput cap in high capacity channels, even when the pre-coding rate is increased from 0.95 to 0.98. Due to the throughput cap of Raptor codes at high SNR's, we focused on RC-IRA and RC/QC-LDPC codes for the followed discussions. The throughput of both codes with short and moderate message word length is simulated in the AWGN channel with 16-QAM.

We observed similar results as in the binary-input AWGN channel - RC-IRA codes have higher throughput than RC/QC-LDPC codes at high SNR's but slightly lower throughput at low SNR's. We then use RC-IRA codes to examine how iterative de-mapping (IDM), labeling and reliability-based coded modulation (RBCM) affect rateless codes. All of these discussions are based on 16-QAM and can be easily extended on higher-order QAM. We found that the gain of IDM decreases from 0.45 dB at rate 1/4 to 0.1 dB at rate 1/2. Above the rate 1/2, no noticeable gain of IDM can be observed for Gray-labeled 16-QAM. For other previously published labeling schemes, although IDM can significantly improve their throughput, our simulation results suggest that binary reflected Gray labeling is the best choice for RC-IRA codes. This result is different from the previously published results on fixed rate codes, where the BER using labeling other than Gray was observed lower than using Gray when the BER is smaller than 10^{-1}. As an attempt to apply RBCM to rateless codes, we applied a very simple RBCM scheme on RC-IRA codes by mapping information bits with different variable degrees into bit levels with different protection capabilities. Parity bits are left uncontrolled. Mapping bits with lower variable degree to reliable bit levels has around 0.2 dB penalty at moderate SNR's. We recommended a robust choice - mapping half bits with lower variable degree to reliable bit levels and another half to unreliable bit levels.

Other than the AWGN channel, we also investigated RC-IRA and RC/QC-LDPC codes in fast and slow Rayleigh fading channels, characterized by different B_dT_s values: $10^{-1}, 10^{-4}, 10^{-5}$. Using average realized rate (ARR), outage probability, and modified ARR as criteria, we showed in either fast-fading or slow-fading channels, RC-IRA codes outperform RC/QC-LDPC codes at high SNR's and perform worse at low SNR's.

Our results suggest that RC-IRA codes and RC/QC-LDPC codes are better than Raptor codes in terms of performance. The advantage of RC-IRA codes over RC/QC-LDPC codes is the ability to achieve rate 1 at high SNR's. At low SNR's, however, RC/QC-LDPC codes provides slightly higher throughput than RC-IRA codes, either in AWGN or Rayleigh fading channels. On the other hand, the complexity (number of edges) is lower for RC-IRA codes.

7.2 Future Work

To compensate for the lower throughput of RC-IRA codes at low SNR's, a possible future work is to combine the high throughput of RC/QC-LDPC codes at low SNR's and the high throughput of RC-IRA codes at high SNR's. We could use PEG to construct a RC-IRA code with rate 0.75 and then expand it by both check splitting and adding more edges. The latter is to ensure the average left degree of RC-IRA codes does not drop too fast due to check splitting. Cares need to be taken while expanding to ensure the variable degree distribution evolves with the decreased rate. The increased difficulty of hardware implementation also needs to be considered.

In this thesis, only 16-QAM is investigated for IDM, labeling, and RBCM. Since the gap between BICM capacity and CM capacity increases with larger signal sets, the gain of IDM might be larger in Gray-labeled 64-QAM or 256-QAM. We can also apply RBCM to 64-QAM or 256-QAM, where we have more bit levels than 16-QAM and therefore have more freedoms in designing RBCM schemes. Another possible direction on RBCM is to specially design a RBCM scheme for punctured codes. The method might involve jointly designing variable degree, girth distribution, and puncturing order. Note that the gain of jointly optimizing coding and modulation must overweigh the convenience of separating coding and modulation.

References

[1] R. G. Gallager, *Information Theory and Reliable Communication.* New York, NY: John Wiley & Sons, Inc., 1968.

[2] R. J. McEliece, *The Theory of Information and Coding*, 2nd ed. Cambridge, England: Cambridge University Press, 2002.

[3] D. M. Mandelbaum, "An adaptive-feedback coding scheme using incremental redundancy," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 388–389, May 1974.

[4] A. Shokrollahi, "Raptor codes," in *Proc. IEEE Int. Symp. Information Theory (ISIT) 2004*, Chicago, USA, June 2004, p. 36.

[5] M. Good and F. R. Kschischang, "Incremental redundancy via check splitting," in *23rd Biennial Symposium on Communications*, Kingston, Canada, May 2006, pp. 55–58.

[6] 3GPP2. (2007) C.p0084-001-0 v1.088 Physical Layer for Ultra Mobile Broadband (UMB) Air Interface Specification (_c.p0084-001 v1.088 phy clean.pdf). [Online]. Available: http://www.3gpp2.org/Public_html/Misc/C.P0084-001_v1.088_PHY_clean_VV_Due_9_Aug-2007.pdf

[7] S.-E. Park, S. Choi, T. Lestable, and A. Tee. (2007) LDPC code proposal. [Online]. Available: http://www.ieee802.org/20/Contribs/C802.20-07-15.pdf, http://www.ieee802.org/20/Contribs/C802.20-07-16.pdf, http://www.ieee802.org/20/Contribs/C802.20-07-17.pdf

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

[9] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1988.

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

[11] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, 1981.

[12] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. A. Spielman, "Analysis of low-density codes and improved designs using irregular graphs," in *Proc. 30th Annual ACM Symp. on Theory of Computing*, Dallas, TX, USA, 1998, pp. 249–258.

[13] T. Richardson and R. L. Urbanke. (2007) Modern coding theory. [Online]. Available: http://lthcwww.epfl.ch/mct/index.php

[14] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, pp. 657–670, Feb. 2001.

[15] 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, pp. 619–637, Feb. 2001.

[16] P. Oswald and A. Shokrollahi, "Capacity achieving sequences for the erasure channel," in *Proc. IEEE Int. Symp. Information Theory (ISIT) 2001*, Washington D.C., USA, 2001, pp. 152–166.

[17] W. W. Peterson and J. E. J. Weldon, *Error-Correcting Codes.* Cambridge, MA: MIT Press, 1972.

[18] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[19] J. G. Proakis, *Digital Communications*, 4th ed. New York, NY: The McGraw-Hill Companies, Inc., 2001.

[20] E. R. Berlekamp, R. J. McEliece, and H. C. A. V. Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 384–386, May 1978.

[21] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 49, pp. 2809–2825, Nov. 2003.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms.* Cambridge, MA: MIT Press, 2001.

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

[24] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[25] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[26] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?" *IEEE Trans. Inform. Theory*, vol. 45, pp. 2173–2181, June 1999.

[27] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköing Univ., Linköing, Sweden, 1996.

[28] F. Zarkeshvari, "Implementation of iterative decoding algorithms on digital VLSI platform," Master's thesis, Carleton Univ., Ottawa, Canada, 2002.

[29] E. Eleftheriou, T. Mittelholzer, and A. Dholakia, "Reduced-complexity decoding algorithm for low-density parity-check codes," *Electronics Letters*, vol. 37, pp. 102–104, Jan. 2001.

[30] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, pp. 1288–1299, Aug. 2005.

[31] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1570–1579, June 2002.

[32] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*, 4th ed. New York, NY: The McGraw-Hill Companies, Inc., 2002.

[33] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaving laws to parallel Turbo and LDPC decoder architectures," *IEEE Trans. Inform. Theory*, vol. 50, pp. 2002–2009, 2004.

[34] W. T. Tutte, *Graph Theory*. Cambridge, United Kingdom: Cambridge University Press, 2001.

[35] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE GLOBECOM 2001*, San Antonio, USA, Nov. 2001, pp. 995–1001.

[36] ——, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inform. Theory*, vol. 51, pp. 386–398, Jan. 2005.

[37] T. Tian, C. Jones, J. Villasenor, and R. D. Wesel, "Construction of irregular LDPC codes with low error floors," in *Proc. IEEE International Conference on Communications (ICC) 2003*, Anchorage, USA, May 2003, pp. 3125–3129.

[38] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1757–1766, Nov. 1997.

[39] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711–2736, Nov. 2001.

[40] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," in *Proc. 6th Int. Symp. Commun. Theory and Applications*, Ambleside, UK, July 2001, pp. 365–370.

[41] M. P. C. Fossorier, "Quasi-cyclic low density parity check codes from circulant permutation matrices," *IEEE Trans. Inform. Theory*, vol. 50, pp. 1788–1794, Aug. 2004.

[42] S. B. Wicker, *Error Control Systems for Digital Communication and Storage.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1995.

[43] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and J. D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inform. Theory*, vol. 50, pp. 2966–2984, Dec. 2004.

[44] N. Miladinovic and M. P. C. Fossorier, "Systematic recursive construction of LDPC codes," *IEEE Commun. Lett.*, vol. 8, pp. 302–304, May 2004.

[45] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms*, 1st ed. Cambridge, England: Cambridge University Press, 2002.

[46] M. Luby, "LT-codes," in *Proc. IEEE 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS) 2002*, Vancouver, Canada, Nov. 2002, pp. 271–280.

[47] O. Etesami, M. Molkaraie, and A. Shokrollahi, "Raptor codes on symmetric channels," in *Proc. IEEE Int. Symp. Information Theory (ISIT) 2004*, Chicago, USA, June 2004, p. 39.

[48] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in *Proc. IEEE Int. Symp. Information Theory (ISIT) 2004*, Chicago, USA, June 2004, p. 38.

[49] M. Yazdani and A. H. Banhashemi, "On construction of rate-compatible low-density parity-check codes," in *Proc. IEEE International Conference on Communications (ICC) 2004*, Paris, France, June 2004, pp. 430–434.

[50] J. Li and K. R. Narayanan, "Rate-compatible low density parity check codes for capacity-approaching ARQ scheme in packet data communications," in *Proc. Int. Conf. Communications, Internet, and Information Technology (CIIT)*, Virgin Islands, USA, Nov. 2002, pp. 201–206.

[51] J. Ha and S. W. McLaughlin, "Optimal puncturing distributions for rate-compatible low-density parity-check codes," in *Proc. IEEE Int. Symp. Information Theory (ISIT) 2003*, Yokohama, Japan, June 2003, p. 233.

[52] T. Tian, C. Jones, and J. D. Villasenor, "Rate-compatible low-density parity-check codes," in *Proc. IEEE Int. Symp. Information Theory (ISIT) 2004*, Chicago, USA, June 2004, p. 153.

[53] A. I. V. Casado, W. Weng, and R. D. Wesel, "Multiple rate low-density parity-check codes with constant block length," in *Asilomar Conf. on Signals, Systems and Computers (ACSSC)*, Pacific Grove, USA, Nov. 2004, p. 153.

[54] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *IEEE Trans. Inform. Theory*, vol. 52, pp. 728–738, Feb. 2006.

[55] J. Ha, J. Kim, and S. W. McLaughlin, "Puncturing for finite length low-density parity-check codes," in *Proc. IEEE Int. Symp. Information Theory (ISIT) 2004*, Chicago, USA, June 2004, p. 152.

[56] M. Yanga, W. E. Ryan, and Y. Li, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Trans. Commun.*, vol. 52, pp. 564–571, Apr. 2004.

[57] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "Design of efficiently-encodable rate-compatible irregular LDPC codes," in *Proc. IEEE International Conference on Communications (ICC) 2006*, Istanbul, Turkey, June 2006, pp. 1131–1136.

[58] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proc. 2nd Int. Symp. Turbo Codes and Related Topics*, Brest, France, 2000, pp. 1–8.

[59] A. I. V. Casado, S. Valle, W.-Y. Weng, and R. D. Wesel, "Constant-blocklength multiple-rate LDPC codes for analog-decoding implementations," in *Proceedings Analog Decoding Workshop*, Torino, Italy, June 2006.

[60] M. Good, "Incremental redundancy via check splitting," Master's thesis, Univ. of Toronto, Toronto, Canada, 2006.

[61] G. D. Forney, "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, pp. 520–548, Feb. 2001.

[62] A. Roumy, S. Guemghar, G. Caire, and S. Verdú, "Design methods for irregular repeat-accumulate codes," *IEEE Trans. Inform. Theory*, vol. 50, pp. 1711–1727, Aug. 2004.

[63] E. Choi, S.-B. Suh, and J. Kim, "Rate-compatible puncturing for low-density parity-check codes with dual-diagonal parity structure," in *IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Berlin, Germany, 2005, pp. 2642–2646.

[64] S. Myung, K. Yang, and J. Kim, "Quasi-cyclic LDPC codes for fast encoding," *IEEE Trans. Inform. Theory*, vol. 51, pp. 2894–2901, Aug. 2005.

[65] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, pp. 2551–2567, June 2006.

[66] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, Feb. 2001.

[67] Z. Cheng, J. Castura, and Y. Mao, "On the design of Raptor codes binary-input Gaussian channels," in *Proc. IEEE Int. Symp. Information Theory (ISIT) 2007*, Nice, France, 2007, pp. 426–430.

[68] R. L. Urbanke. (2001) LDPC distribution optimizer. [Online]. Available: http://lthcwww.epfl.ch/research/ldpcopt/

[69] J. Hou, P. H. Siegel, L. B. Milstein, and H. D. Pfister, "Capacity approaching bandwidth-efficient coded modulation scheme based on low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 49, pp. 2141–2155, 2003.

[70] U. Wachsmann, R. F. H. Fischer, and J. B. Huber, "Multilevel codes: Theoretical concepts and practical design rules," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1361–1391, 1999.

[71] T. Woerz and J. Hagenauer, "Iterative decoding for multilevel codes using reliability information," in *Proc. IEEE GLOBECOM 1992*, Orlando, FL, USA, Dec. 1992, pp. 1779–1784.

[72] N. Seshadri and C.-E. W. Sundberg, "Multilevel trellis coded modulations for the Rayleigh fading channel," *IEEE Trans. Commun.*, vol. 41, pp. 1300–1310, 1993.

[73] M. Isaka and H. Imai, "On the iterative decoding of multilevel codes," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 935–943, May 2001.

[74] X. Li and J. A. Ritcey, "Bit-interleaved coded modulation with interative decoding," *IEEE Commun. Lett.*, vol. 1, pp. 169–171, Nov. 1997.

[75] G. T. G. Caire and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Trans. Inform. Theory*, vol. 44, pp. 927–946, May 1998.

[76] S. ten Brink, J. Speidel, and R.-H. Yan, "Iterative demapping and decoding for multilevel modulation," in *Proc. IEEE GLOBECOM 1998*, Sydney, Australia, Nov. 1998, pp. 579–584.

[77] X. Li, A. Chindapol, and J. A. Ritcey, "Bit-interleaved coded modulation with iterative decoding and 8PSK signaling," *IEEE Trans. Commun.*, vol. 50, pp. 1250–1257, Aug. 2002.

[78] H. Rui, X. Zhang, and D. Xu, "Performance of belief propagation coded modulation with iterative decoding," in *International Conference on Communications, Circuits and Systems (ICCCAS) 2004*, Chengdu, China, June 2004, pp. 71–74.

[79] Y. Nana, E. Sharon, and S. Litsyn, "Improved decoding of LDPC coded modulation," *IEEE Commun. Lett.*, vol. 10, pp. 375–377, May 2006.

[80] R. D. Wesel, X. Liu, J. M. Cioffi, and C. Komninakis, "Constellation labeling for linear encoders," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2417–2431, 2001.

[81] F. Gray, "Pulse code communication," Mar. 1953, U. S. Patent 2632058.

[82] E. Agrell, J. Lassing, E. G. Ström, and T. Ottosson, "On the optimality of the binary reflected Gray code," *IEEE Trans. Inform. Theory*, vol. 50, pp. 3170–3182, Dec. 2004.

[83] ——, "Gray coding for multilevel constellations in Gaussian noise," *IEEE Trans. Inform. Theory*, vol. 53, pp. 224–235, Jan. 2007.

[84] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. 28, pp. 56–67, Jan. 1982.

[85] A. Chindapol and J. A. Ritcey, "Design, analysis, and performance evaluation for BICM-ID with square QAM constellations in Rayleigh fading channels," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 944–957, May 2001.

[86] F. Schreckenbach, N. Görtz, J. Hagenauer, and G. Bauch, "Optimization of symbol mappings for bit-interleaved coded modulation with iterative decoding," *IEEE Commun. Lett.*, vol. 7, pp. 593–595, Dec. 2003.

[87] R. D. Maddock and A. H. Banihashemi, "Reliability-based coded modulation with low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, pp. 403–406, Mar. 2006.

[88] Y. Li and W. E. Ryan, "Bit-reliability mapping in LDPC-coded modulation systems," *IEEE Commun. Lett.*, vol. 9, pp. 1–3, Jan. 2005.

[89] Y. Li, C. K. Ho, Y. Wu, and S. Sun, "Bit-to-symbol mapping in LDPC coded modulation," in *Proc. IEEE Vehicular Technology Conference (VTC) 2005*, Dallas, TX, USA, May 2005, pp. 683–686.

[90] H. Sankar, N. Sindhushayana, and K. R. Narayanan, "Design of low-density parity-check (LDPC) codes for high order constellations," in *Proc. IEEE GLOBECOM 2004*, Dallas, TX, USA, Dec. 2004, pp. 3113–3117.

[91] G. Durisi, L. Dinoi, and S. Benedetto, "eIRA codes for coded modulation systems," in *Proc. IEEE International Conference on Communications (ICC) 2006*, Istanbul, Turkey, June 2006, pp. 1125–1130.

[92] S. Haykin and M. Moher, *Modern Wireless Communications.* Upper Saddle River, NJ: Prentice Hall, 2005.

[93] E. W. Weisstein. (2007) Exponential integral. From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/ExponentialIntegral.html

[94] T. F. Wong, "Numerical calculation of symmetric capacity of Rayleigh fading channel with BPSK/QPSK," *IEEE Commun. Lett.*, vol. 5, pp. 328–330, Aug. 2001.

[95] E. Biglieri, *Coding for Wireless Channels.* New York, NY: Springer, 2005.

[96] E. Biglieria, J. Proakis, and S. Shamai, "Fading channels: Information theoretic and communications aspects," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2619–2692, Oct. 1998.

[97] W. C. Jakes, *Microwave Mobile Communications.* Wiley, 1974.

[98] P. Dent, G. E. Bottomley, and T. Croft, "Jakes fading model revisited," *Electronics Letters*, vol. 29, pp. 1162–1163, June 1993.