Graph-based Codes and Iterative Decoding

Thesis by

Aamod Khandekar

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2002

(Submitted June 10, 2002)

Acknowledgements

I would like to begin by expressing my heartfelt gratitude to my advisor, Prof. Robert McEliece, for his active guidance and support throughout my stay at Caltech. His infectious enthusiasm and his many insights have been invaluable for the progress of my research career.

I would like to thank Prof. Robert McEliece, Prof. P. P. Vaidyanathan, Prof. Jehoshua (Shuki) Bruck, Prof. John Preskill, Prof. Babak Hassibi and Dr. Dariush Divsalar for serving on my candidacy and defense committees, and for their advice on this thesis.

I am grateful to all the current and former occupants of Moore 155, and to current and former residents of Braun graduate house, for making my stay at Caltech a very pleasant one.

Abstract

The field of error correcting codes was revolutionized by the introduction of turbo codes [7] in 1993. These codes demonstrated dramatic performance improvements over any previously known codes, with significantly lower complexity. Since then, much progress has been made towards understanding the performance of these codes, as well as in using this understanding to design even better codes.

This thesis takes a few more steps in both these directions. We develop a new technique, called the typical set bound, for analyzing the asymptotic performance of code ensembles based on their weight enumerators. This technique yields very tight bounds on the maximum-likelihood decoding threshold of code ensembles, and is powerful enough to reproduce Shannon's noisy coding theorem for the class of binary-input symmetric channels.

We also introduce a new class of codes called irregular repeat-accumulate (IRA) codes, which are adapted from the previously known class of repeat-accumulate (RA) codes. These codes are competitive in terms of decoding performance with the class of irregular low-density parity-check (LDPC) codes, which are arguably the best class of codes known today, at least for long block lengths. In addition, IRA codes have a significant advantage over irregular LDPC codes in terms of encoding complexity.

We also derive an analytical bound regarding iterative decoding thresholds of code ensembles on general binary-input symmetric channels, an area in which theoretical results are currently lacking.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

In his seminal paper in 1948, Shannon [45] derived the fundamental limits on the rate of communication on a noisy channel. Shannon's "noisy coding theorem" states that every channel has a capacity C, which is the highest rate (in bits per channel use) at which reliable communication is possible. Shannon also showed that for a long enough block length, almost any block code of rate $R < C$, with optimal decoding, provides reliable communication over this channel. This scheme, however, is not a practical one, since both encoding and decoding are prohibitively expensive.

Ever since Shannon proved his noisy coding theorem, the construction of practical capacity-achieving schemes has been the supreme goal of coding theory. The classical approaches to this problem included algebraic block codes and convolutional codes. The field was, however, revolutionized by the introduction of turbo codes by Berrou, Glavieux, and Thitimajshima [7]. The performance of turbo codes was much closer to capacity than that of any previous codes, and with significantly lower complexity.

The power of turbo codes comes not only from the code construction, but also from the iterative decoding algorithm used. The code construction consists of simple blocks connected by a pseudorandom interleaver. The interleaver introduces enough randomness into the code to ensure good performance, yet keeps enough structure to allow simple encoding and decoding algorithms. The invention of turbo codes led to an explosion of interest in the field of codes on graphs and iterative decoding, which led among other things to the rediscovery of low-density parity-check (LDPC) codes [17, 37], and the invention of repeat-accumulate (RA) codes [15].

While the turbo-decoding algorithm worked extremely well in practice, there was very little theory available to explain its dramatic performance. It was soon discov-

ered that the turbo-decoding algorithm was an instance of Pearl's belief propagation algorithm over loopy graphs [39], and several frameworks were developed to formalize this notion [39, 4, 31]. The belief propagation algorithm is known to terminate in finite time in the case of non-loopy graphs (trees) and is optimal in this case. Its behavior on general graphs was, however, not known. Some results in this direction were presented in [2, 51, 52]. Luby et al. [33, 34], followed by Richardson and Urbanke [42], showed how LDPC codes could be approximated by the cycle-free case, and were able to prove asymptotic results about their iterative decoding performance. Luby et al. also introduced the concept of irregularity, which seems to provide hope of operating arbitrarily close to channel capacity in a practical manner, on a wide class of channel models. More recently, connections have been discovered between coding theory and statistical physics [54, 55, 5], which show some hope of providing insight into the general problem of decoding on loopy graphs.

All the codes mentioned in the preceding paragraph are so-called "graphical code ensembles." Most of this thesis deals with the analysis and design of such code ensembles, and hence we devote the rest of this chapter to some basic background material concerning them.

1.1 Some Basic Concepts

1.1.1 Channel Models

A canonical communication system is depicted in Figure 1.1. The objective is to communicate an input string $\mathbf{U}$ across a "noisy"channel. The encoder converts this string into another string $\mathbf{X}$ of symbols over the channel's input alphabet. A string $\mathbf{Y}$ is seen at the other end of the channel, which is a non-deterministic function of the channel input $\mathbf{X}$, and the decoder tries to reconstruct the input string $\mathbf{U}$ based on the knowledge of $\mathbf{Y}$. To analyze such a system, we need to have a model for the

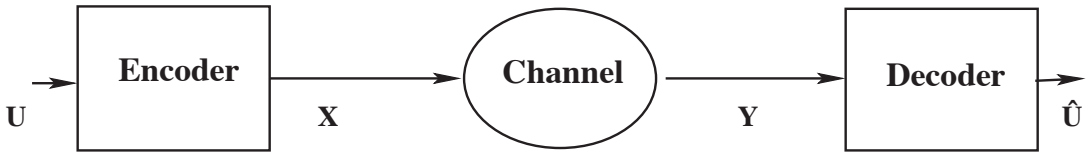

Figure 1.1: A canonical communication system.

channel. Typically, we assume that the output string $\mathbf{Y}$ of the channel has the same length as the input string $\mathbf{X}$, and depends on $\mathbf{X}$ via a conditional probability density function (pdf) $p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$. This is still an extremely general model and we therefore define several special cases.

Definition 1.1 A channel is called *memoryless* if the channel output at any time instant depends only on the input at that time instant, i.e., if $\mathbf{y} = y_1 y_2 \dots y_n$ and $\mathbf{x} = x_1 x_2 \dots x_n$, then $p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} p_{Y|X}(y_i|x_i)$. In this case, the channel is completely described by its input and output alphabets, and the conditional pdf $p_{Y|X}(y|x)$ for one time instant.

Definition 1.2 (Capacity) Let the input to a memoryless channel be generated by independent and identically distributed (iid) copies of a random variable X. Then the output of the channel will be iid copies of some random variable Y whose pdf can be computed. Clearly, $I(X;Y)$, the information between random variables X and Y (see [10] for a definition), is a function of the pdf of X. The *capacity* of the memoryless channel is then defined as $\sup_{p_X(x)} I(X;Y)$. The motivation behind this definition will be clear in Section 1.1.2.

Example 1.1 A very commonly used channel model is the *additive Gaussian noise* channel. This channel has $\mathbb{R}$ as its input and output alphabets, and is parametrized by a non-negative real number σ. The channel output Y is given by $X + N$, where X is the channel input and N is a Gaussian random variable (random variable) with mean 0 and variance σ^2. The conditional pdf $p_{Y|X}(y|x)$ is therefore a Gaussian pdf

with mean x and variance σ^2.

Definition 1.3 A *binary input channel* is merely a channel with a binary input alphabet. We will interchangeably use the sets $\{0, 1\}$ and $\{+1, -1\}$ for the input alphabet with 0 mapping to +1 and 1 to -1.

Example 1.2 (The Z-Channel) This is a binary input channel with a parameter p and output alphabet $\{0, 1\}$, in which a 0 is always received as a 0, but a 1 could be received as a 0 with probability p. That is, $p_{Y|X}(0|0) = 1$, $p_{Y|X}(1|0) = 0$, $p_{Y|X}(0|1) = p$, and $p_{Y|X}(1|1) = 1 - p$.

Suppose we want to compute the input distribution of a binary-input channel conditioned on the knowledge of the received value y, i.e., we want to compute the a posteriori probabilities $\Pr(X = 0|Y = y)$ and $\Pr(X = 1|Y = y)$, which sum to 1. This knowledge can be efficiently packaged into one number, for example their ratio. By Bayes' rule, we have

$$\frac{\Pr(X = 0|Y = y)}{\Pr(X = 1|Y = y)} = \frac{p_{Y|X}(y|0)}{p_{Y|X}(y|1)} \frac{\Pr(X = 0)}{\Pr(X = 1)}. \tag{1.1}$$

Hence the quantity $\frac{p_{Y|X}(y|0)}{p_{Y|X}(y|1)}$ is a sufficient statistic for estimating the input to the channel. So, of course, is any invertible function of it. This leads us to the following definition.

Definition 1.4 The quantity $\frac{p_{Y|X}(y|0)}{p_{Y|X}(y|1)}$ corresponding to the output y of a binary-input channel is called its *likelihood ratio*. Its logarithm $\log \frac{p_{Y|X}(y|0)}{p_{Y|X}(y|1))}$ is called the *log-likelihood ratio (LLR)*.

Notice that there does not have to be an explicit channel for a (log)-likelihood ratio to be defined. It is well defined in the context of a noisy (probabilistic) estimate of a binary random variable.

Definition 1.5 A *binary input symmetric channel (BISC)* is a memoryless binary input channel with $\mathbb{R}$ as its output alphabet, satisfying $p_{Y|X}(y|0) = p_{Y|X}(-y|1)$. Thus a BISC is completely described by the conditional pdf $p_{Y|X}(y|0)$.

In the BISC case, by symmetry, the optimizing pdf in Definition 1.2 is uniform, i.e., $(1/2, 1/2)$, and hence the capacity computation is much simplified.

The BISC assumption leads to many such simplifications in analysis and design, and most of the work presented in this thesis will be for this case. Also, many natural channel models do fall under this class. We present some of the most important ones here.

Example 1.3 (The binary symmetric channel (BSC)) This is a binary-input, binary-output channel with parameter p. To view it as a BISC, it is convenient to let the input and output alphabets be $\{+1, -1\}$. Then the output is equal to the input with probability $1 - p$, and is the negative of the input with probability p. p is called the crossover probability of the channel. We will omit writing the conditional pdf explicitly. The capacity of this channel is given by $1 - H(p)$, where $H(\cdot)$ is the entropy function.

Example 1.4 (The binary erasure channel (BEC)) This channel, with parameter p, has input alphabet $\{+1, -1\}$ and output alphabet $\{+1, 0, -1\}$. The output symbol 0 is also called an erasure. The output is equal to the input with probability $1 - p$ and is 0 with probability p. p is called the probability of erasure. This channel is arguably the simplest nontrivial channel model, and will be one of the focal points of this thesis. The capacity of this channel is $1 - p$.

Example 1.5 (The binary input additive Gaussian noise (BIAGN) channel) This is the additive Gaussian noise channel restricted to inputs +1 and -1. The expression for the capacity of a general BISC is derived in Chapter 2, and is given by

eq. (2.12). For the BIAGN channel, this expression simplifies to

$$C = 1 - \frac{1}{\sqrt{2\pi\sigma^2}} \int_{\mathbb{R}} H\left(\frac{1}{1+e^{2x/\sigma^2}}\right) e^{\frac{(x-1)^2}{2\sigma^2}} dx, \tag{1.2}$$

where $H(\cdot)$ is again the entropy function.

1.1.2 Codes and Code Ensembles

The basic idea in coding theory is to add redundancy to the transmitted information in order to help combat channel errors. Thus, in Figure 1.1, we restrict the set of strings $\mathbf{X}$ that can be used, so that no two legal strings are "close to" one another. In this way, a channel error is unlikely to cause confusion between two legal strings.

Definition 1.6 A *block code* of length n over an alphabet $\mathcal{A}$ is a subset of $\mathcal{A}^n$, the set of n-length strings over $\mathcal{A}$. The elements of this subset are called *codewords*.

We typically need to introduce more structure into our codes to aid in both analysis and design. Most codes used in practice are so-called *linear codes*, which are defined when the elements of the alphabet $\mathcal{A}$ form a field.

Definition 1.7 An (n, k) *linear code* over a field $\mathcal{F}$ is a k-dimensional vector subspace of $\mathcal{F}^n$. k is called the dimension of the code, $r = n - k$ its redundancy, and $R = k/n$ its rate. A *binary linear code* is merely a linear code over the binary field.

Linear codes have several nice properties, for example, they look exactly the same around any codeword. That is, if $\mathcal{C}$ is a linear code and $\mathbf{c} \in \mathcal{C}$ is a codeword, then the set $\mathcal{C} - \mathbf{c}$ is identical to $\mathcal{C}$. Also, in order to describe a linear code, we don't have to list all its elements, but merely a basis. Such a description is called a *generator matrix* representation.

Definition 1.8 A *generator matrix* for an (n, k) linear code $\mathcal{C}$ is a $k \times n$ matrix G

whose rows form a basis for $\mathcal{C}$. As $\mathbf{u}$ varies over the space $\mathcal{F}^k$, $\mathbf{u}G$ varies over the set of codewords. Thus, the matrix G provides an encoding mechanism for the code.

Another useful representation of a linear code is a *parity-check matrix* representation.

Definition 1.9 A *parity-check matrix* for an (n, k) linear code $\mathcal{C}$ is an $(n-k) \times n$ matrix H whose rows form a basis for the space of vectors orthogonal to $\mathcal{C}$. That is, H is a full rank matrix s.t. $H\mathbf{c} = \mathbf{0} \iff \mathbf{c} \in \mathcal{C}$.

To formalize the notion of codewords being "close to" each other, we will make use of the Hamming metric.

Definition 1.10 The *Hamming distance* between two vectors is the number of components in which they differ. The *minimum distance* of a code is the smallest Hamming distance between two distinct codewords. For a linear code, this is the same as the least weight of any nonzero codeword.

Another useful notion is that of an *ensemble* of codes, which is used when we wish to average some quantity over a set of codes.

Definition 1.11 An ensemble of codes is a set of codes of the same length, and typically having approximately the same rates, with an associated pdf. We use the same term for sequences of such sets, often with length going to infinity.

Example 1.6 (The ensemble of random codes) To construct this ensemble, fix a length n and a rate R. Then the number of codewords should be 2^{nR}. Also fix a pdf $p_X(x)$ over the code alphabet, and pick each element of each codeword independently according to this pdf. This ensemble is known as the ensemble of random codes.

For a memoryless channel, Shannon [45] showed that if $p_X(x)$ was chosen to be the optimizing pdf in Definition 1.2, then for any rate R less than the capacity of the

channel, the average probability of error for the ensemble of random codes tends to zero as n tends to infinity. He also showed that the probability of error for a code whose rate exceeded the capacity was bounded away from zero.

If $p_X(x)$ is uniform over the input alphabet, then this ensemble is very similar to the one in which we pick any code of the correct length and the correct rate with equal probability, and we use the term "random codes" for this ensemble also.

Example 1.7 (The ensemble of random linear codes) This is the set of all linear codes of some fixed rate R (and length n), each selected with equal probability. For a BISC, this ensemble is known to achieve capacity [18] (i.e., the average probability of error tends to zero as n tends to infinity for any rate less than the capacity of the channel). This is another reason why the BISC assumption is so useful. One proof of this fact will be given in Chapter 2.

This ensemble is sometimes defined by saying that every entry in the generator (or parity check) matrix is picked independently with a uniform pdf. While the ensembles so defined are not identical to the first definition, most properties for large n (in particular the *weight enumerator*, to be defined in Chapter 2) are indeed the same. We will use the term "random linear codes" for either description.

1.1.3 Decoding Algorithms

Given the output of a noisy channel, we need to form an estimate of the transmitted codeword. We now define notions of "optimal" decoding algorithms.

Given a received vector $\mathbf{y}$, the codeword most likely to have been transmitted is the one that maximizes $p_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y})$. If the channel is memoryless and each of the codewords is equally likely, then this reduces to the codeword $\mathbf{x}$ which maximizes $p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ (which factorizes if the channel is memoryless). This is known as the *maximum likelihood (ML)* estimate of the transmitted codeword.

Definition 1.12 Given a code $\mathcal{C}$ and a received vector $\mathbf{y}$, the *maximum likelihood* decoder has as its output

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}\in\mathcal{C}} p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}). \tag{1.3}$$

Clearly this decoder has the least possible (word) probability of error.

If, on the other hand, we wish to minimize the *bit error probability*, then we need to maximize $\Pr(x_i = x|\mathbf{y})$ over all x.

Definition 1.13 Given a received vector $\mathbf{y}$, the *MAP (maximum a posteriori)* decoder has as its output in the ith position

$$\hat{x}_i = \arg\max_{x\in\mathcal{A}} \Pr(x_i = x|\mathbf{y}), \tag{1.4}$$

where the maximization is over the input alphabet of the channel.

In the case of a binary code, the maximization is over a binary alphabet. Taking the maximum of two values is equivalent to comparing their ratio to 1, or the log of the ratio to 0. This latter quantity is nothing but the LLR. Therefore, MAP decoding in the case of a binary code is equivalent to taking the sign of the a posteriori LLR.

1.2 Some Graphical Code Ensembles

In this section, we will introduce some important graphical code ensembles, which is a generic term we will use to describe all "turbo-like" codes, or codes amenable to iterative decoding. All of these codes can be viewed under a unified graphical framework. Several such frameworks have been proposed, for example, see [4], [31], and [16]. In all these cases, the iterative decoding algorithm reduces to an instance of Pearl's belief propagation algorithm [41] on loopy graphs. Such a general view will, however, not be necessary for the purposes of this thesis, and we will describe each

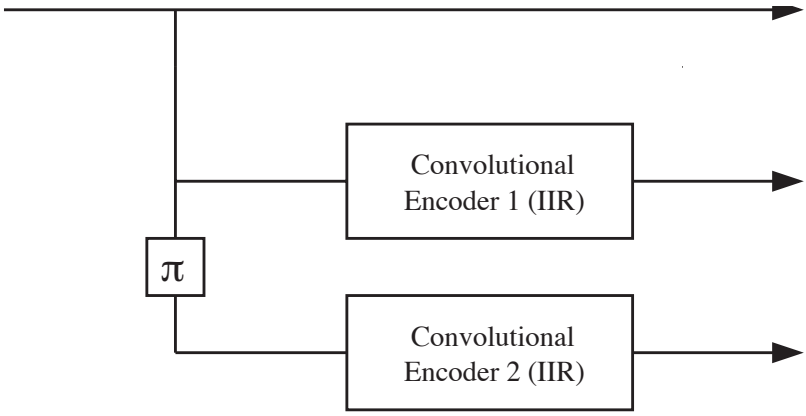

Figure 1.2: Parallel concatenation of two convolutional codes, connected through a random interleaver (denoted by π).

ensemble separately with its own decoding algorithm. The specific decoding algorithm we will describe is called the sum-product algorithm, which aims to minimize the bit error probability. The idea in every case is that we use "long, random-like codes" as suggested by Shannon, which is possible through the existence of a simple though suboptimal decoding algorithm.

1.2.1 Parallel Concatenation of Convolutional Codes (PCCC)

These codes are also known as "parallel turbo codes." The original turbo code introduced by Berrou et al. [7] was a code of this type. The general structure is shown in Figure 1.2. As mentioned earlier, it consists of two relatively simple constituent codes, more specifically truncated binary IIR convolutional codes with a short constraint length. They are connected by an interleaver (labeled π in the figure), which is merely a pseudo-random permutation of the information bits. In the figure, there is also a systematic (information) bit stream, which could be absent. There could also be more than two constituent codes, each with its own interleaver. Here, we briefly

describe the decoding algorithm in the case when there are two constituent encoders and a systematic data stream.

The aim of the sum-product algorithm is to approximate MAP decoding, as defined in Definition 1.13, or equivalently to compute the a posteriori log-likelihoods of the individual transmitted bits given the received vector. The MAP decoding algorithm for the constituent convolutional codes can be implemented with the well known forward-backward or BCJR [6] algorithm, which is feasible in this case because these codes have a short constraint length. Given an a priori estimate (or LLR) on each information bit and an LLR for each transmitted bit, the BCJR algorithm outputs the correct a posteriori LLR for each information bit.

The turbo-decoding algorithm iterates between the MAP decoders corresponding to the two constituent codes. The received values corresponding to the systematic bits are used to initialize the a priori LLR's for the information bits. One of the constituent decoders then outputs the a posteriori LLR's by running the BCJR algorithm, the idea being to use these as a priori LLR's for the other decoder. However, in order not to form short loops in the so-called "computation tree," the *difference* between the a posteriori and the a priori LLR's (this is known as *extrinsic information*) is fed to the other decoder as a priori LLR's, and the same operation is repeated over and over again. Various stopping rules are used to decide on convergence and guard against limit-cycles.

1.2.2 Serial Concatenation of Convolutional Codes (SCCC)

These codes are also known as "serial turbo-codes." In this case each convolutional encoder acts on the interleaved output of the previous one, instead of on the information stream directly. The general structure is shown in Figure 1.3. In this case also, the sum-product algorithm iterates between the decoders corresponding to the constituent codes. Some slight modifications are needed from the PCCC case, but

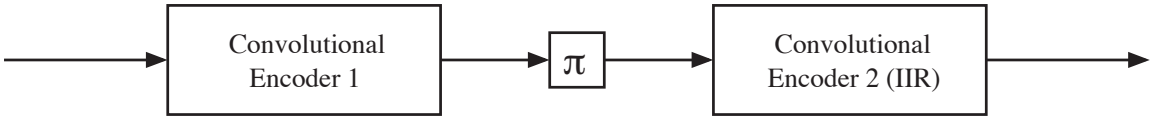

Figure 1.3: Serial concatenation of two convolutional codes, connected through a random interleaver (denoted by π).

the basic idea is the same, and we will omit giving a detailed description here.

One example of the SCCC case is the ensemble of repeat-accumulate (RA) codes introduced in [15]. An RA code is shown in Figure 1.4. It is the concatenation of two particularly simple constituent codes, an outer "repeat by q" code and an inner "accumulate" code. This simple structure was intended to make analysis possible, but their performance under iterative decoding is surprisingly good, especially considering that constituent decoders have extremely low complexity. These codes play an important role in this thesis, particularly in Chapter 3, and we will give an alternative description of them in Section 1.2.3.

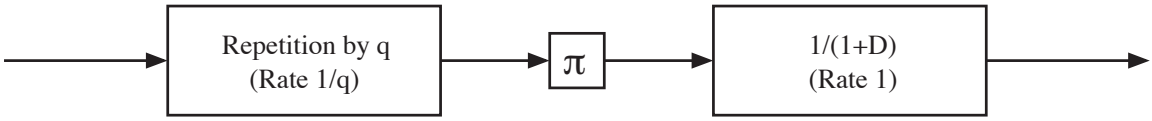

Figure 1.4: A repeat-accumulate (RA) code.

1.2.3 Codes Defined on Tanner Graphs

A *Tanner graph* is a general way of representing any linear code. An example is shown in Figure 1.5. A Tanner graph has two kinds of nodes, called *variable nodes*, represented by hollow circles in the figure, and *check nodes*, represented by filled circles. The graph is bipartite between these two types of nodes, i.e., every edge has

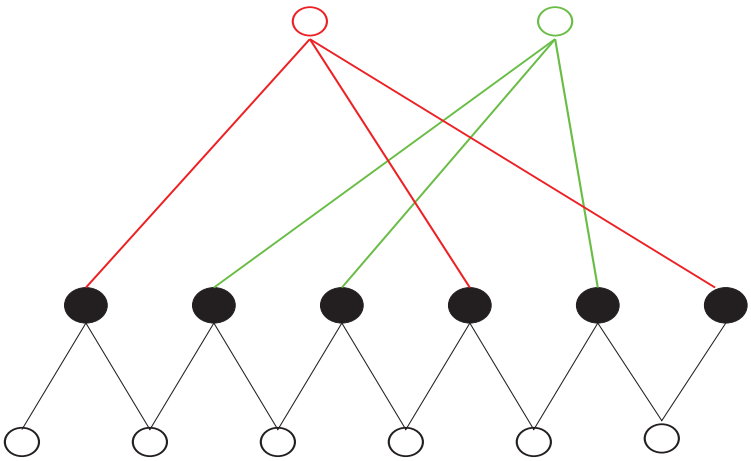

Figure 1.5: A small Tanner graph.

a variable node at one end and a check node at the other end. The variable nodes represent actual variables, for example, elements of the codeword. The check nodes represent constraints among these variables. All the graphs we will look at will be for binary linear codes, in which case the variable nodes represent binary variables, and a check node says that the binary sum of all its neighbors is 0. Clearly, any linear code may be represented in this manner, by directly transcribing its parity-check matrix. In fact, it will have many possible representations, because it has many possible parity-check matrices, and also because we can add dummy or state variables to the graph, as we shall soon see in the case of RA codes.

1.2.4 Decoding on Tanner Graphs

The sum-product algorithm takes a particularly elegant form on a Tanner graph. It was first described in this case by Gallager [17] in the context of LDPC codes. It is a completely distributed algorithm, with each node acting as an independent entity, communicating with other nodes through the edges. The message sent by a variable node to a check node, say in LLR form, is its estimate of its own value. The message sent by a check node to a variable node is its estimate of the variable node's

value. The update rules at the nodes are essentially MAP estimators, given that the incoming messages along the different edges are independent. Again, in order not to form short cycles in the computation tree, the output along any edge is based only on input from the other edges. At a variable node of degree j, if l_1, l_2, ..., l_{j-1} denote the incoming LLR's along $j-1$ edges, and l_0 the LLR corresponding to the channel evidence, then the outgoing LLR l_{out} along the jth edge is merely the MAP estimate of the underlying binary random variable given j independent estimates of it, and is given by

$$l_{\text{out}} = l_0 + \sum_{i=1}^{j-1} l_i. \tag{1.5}$$

At a check node, the situation is similar, though the update rule is more complicated. If l_1, l_2, ..., l_{k-1} denote the incoming LLR's at a check node of degree k, then the outgoing LLR l_k along the kth edge corresponds to the pdf of the binary sum of $j-1$ independent random variables, and works out to be

$$\tanh(l_{\text{out}}/2) = \prod_{i=1}^{k-1} \tanh(l_i/2). \tag{1.6}$$

(For a derivation of eqs. (1.5) and (1.6), see [42, Section 3.2].)

Given these update rules, we only need a schedule for updating the various messages to complete the description of the decoding algorithm, but this schedule varies from code to code, and sometimes there are many reasonable schedules even for a single code. There is one canonical schedule, however, which is to update all variable nodes together, followed by all check nodes, followed again by the variable nodes etc. In practice, for this algorithm to work well, a Tanner graph should have few short cycles. It is not hard to see that if it didn't have any cycles at all, then the independence assumption that we used to derive the message update rules actually holds, which gives an indication why not having short cycles is important. In the next few sections, we give some examples of codes that have natural (and useful) Tanner graph

representations.

1.2.5 Low-Density Parity-Check (LDPC) Codes

LDPC codes were invented by Gallager [17] in 1962, but did not receive much attention until they were rediscovered independently by MacKay [37] following the invention of turbo codes.

Definition 1.14 (LDPC Codes) The ensemble of (j, k) LDPC codes is defined by the set of parity-check matrices with exactly j ones in each column and k ones in each row, with each such matrix being picked with equal probability. Alternatively, it is the set of Tanner graphs in which every variable node has degree j and every check node has degree k, also with a uniform pdf.

The codes are so named because as the length of the code increases, for fixed j and k, the parity-check matrix has very few ones, or equivalently, the Tanner graph is very sparse. Counting edges coming out of variable nodes and check nodes in this graph, we see that $jn = kr$, where n is the length of the code and r its redundancy. Therefore the rate R of the ensemble is given by $1 - r/n = 1 - j/k$. (Here we have assumed that the parity-check matrix is full-rank, but this formula indeed holds for large n, and in any case represents a lower bound on the rate.) For decoding, we use the sum-product decoding algorithm described in the previous section, with the canonical scheduling of messages.

1.2.6 Repeat Accumulate (RA) Codes

The ensemble of RA codes has already been defined as a special case of an SCCC. A Tanner graph representation of this ensemble is shown in Figure 1.6. The nodes on the top represent the information bits, and are not elements of the codeword. (They are examples of the "dummy variables" mentioned in Section 1.2.3.) Each of these

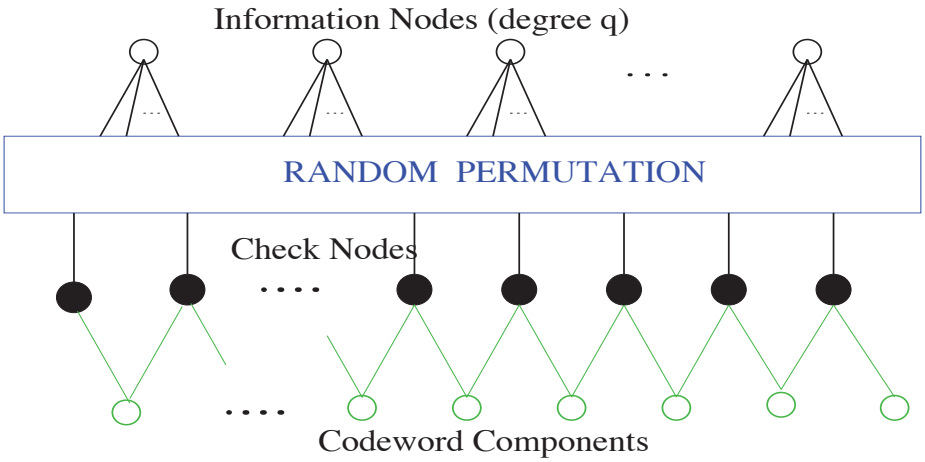

Figure 1.6: The Tanner graph of an RA code.

has q edges coming out of it, corresponding to the outer repeat code. The random permutation corresponds to the interleaver, and the section below it corresponds to the accumulate code. The nodes at the bottom are actually components of the codeword, and it is clear that each element is the sum of the previous element with an information bit, thus showing that the bottom code is actually an accumulator. The graph shown in Figure 1.5 is an example of a small $q = 3$ RA code.

The decoding algorithm described earlier for RA codes can in fact be seen as an instance of the sum-product algorithm on Tanner graphs with appropriate scheduling of message updates. It is not hard to see that the BCJR algorithm for the accumulate code corresponds to messages being passed all the way from left to right and then back from right to left in the part of the graph below the permutation (also updating the edges connecting the information and the check nodes), while the BCJR algorithm on the repeat code is equivalent to the information nodes being updated in any order.

Another possible scheduling of the messages is the following: First pass all messages upwards, i.e., from the lowermost nodes to the check nodes and then form check nodes to information nodes, and then pass all messages downwards, i.e., from information nodes to check nodes followed by messages from check nodes to the nodes rep-

resenting the elements of the codeword. The algorithm given by the second scheduling turns out to be more tractable to analysis, and is the one we will consider most of the time. For most analytical purposes, it is equivalent to the canonical scheduling.

1.3 Density Evolution

In this section, we briefly describe *density evolution*, which is a method for analyzing iterative decoding on many graphical code ensembles. It was first introduced by Gallager [17] in a heuristic form for the analysis of LDPC codes. More recently, it was re-introduced rigorously in the case of the BEC by Luby et al. [33, 34], and was later elaborated on and presented in a more general setting by Richardson and Urbanke [42].

Consider the ensemble of (j, k) LDPC codes of length n. Define the depth-l neighborhood of a node as the set of nodes in the Tanner graph which are a distance at most l from it (with the distance between two nodes being the length of the shortest path between them). Notice that since all nodes in the graph have bounded degree, this neighborhood has bounded size independent of n, and hence covers an arbitrarily small fraction of nodes in the graph as $n \to \infty$. Moreover, since the elements of this neighborhood are picked essentially at random (because of the randomness in the ensemble), with high probability there are no repetitions, and the neighborhood is cycle free.

Definition 1.15 We say that an ensemble of codes satisfies the *RU condition* if for any l, the probability of the depth-l neighborhood of a randomly selected edge containing a cycle goes to 0 as $n \to \infty$.

We just saw that LDPC codes satisfy this condition, and it is not hard to see that so do RA codes. Now, if the depth-l neighborhood of an edge is indeed cycle-free, then it is also not too hard to see that up to l iterations, the messages received at any

node will be independent. (This is true if the canonical message-update schedule is used, but not, for instance, if the SCCC decoding algorithm is used in the case of RA codes.) On a BISC, we can assume that the all-zeros codeword was transmitted (since the code is linear, and the decoder is symmetric between codewords, the probability of error is independent of which codeword was transmitted), which then tells us the pdf of the received messages (since we know the channel). These are also equal to the first set of transmitted messages. Thus we know the pdf's of the received messages at the next iteration, we know the update rule, and hence with the independence assumption, we can compute the pdf's of the transmitted messages. This process can be continued till l iterations. The pdf of a message will also give the corresponding probability of error, and Richardson and Urbanke [42] show that the number so computed is accurate in the limit of large n, if the ensemble of codes satisfies the RU condition.

An interesting thing to check is whether the probability of error so computed tends to 0 as l increases. For any family of channels characterized by a single parameter (we will sometimes call this a *one-parameter family* of channels), the worst parameter for which this happens is called the *iterative decoding threshold* of the code over that family of channels. This quantity is known to be well-defined for any family of channels ordered by *physical degradation* [43], i.e., if the channel with the worse parameter is a degraded version of the channel with the better parameter (i.e., that channel concatenated with another channel). Most of the channels we have defined, including the AGN channel, the BIAGN channel, the BSC, the BEC, and the Z-channel, have this property.

1.3.1 Density Evolution on the BEC

In this section, we give one illustration of how the BEC is a particularly simple channel to analyze. First, notice that on the BEC, with the sum-product decoding algorithm,

there are exactly three distinct possible values among the messages passed, namely 0, $+\infty$ and $-\infty$. To see this, we first note that these are the three values corresponding to the three channel outputs, with $+\infty$ corresponding to a received 0 (in which case we are completely sure that a 0 was transmitted), a -∞ corresponding to a received 1 (in which case we are sure that a 1 was transmitted), and a 0 corresponding to an erasure. Also notice that none of the decoding steps ever causes an error, at most erasures are removed. That is, we never receive both a $+\infty$ and a $-\infty$ at a variable node. With this condition, the set $\{0, +\infty, -\infty\}$ is preserved by the update rules, and hence these are the only messages passed. This is already one indication of the simplicity of the channel.

Under the further assumption that the all-zeros codeword is transmitted, the set of transmitted messages is further reduced to $\{0, +\infty\}$. Hence density evolution on this channel does not involve updating whole pdf's, but merely a single probability. This is the reason why analytical results on the BEC with iterative decoding are possible. Using this technique, Luby et al. [33, 34] demonstrated an ensemble of codes that achieves capacity on the BEC, i.e., has a threshold arbitrarily close to capacity. (Further details can be found in Section 3.2.)

1.4 Thesis Outline

In this chapter, we have outlined some of the developments in coding theory since the invention of turbo codes, and have presented material that will be required in the rest of the thesis. The next chapter deals with the typical set decoder, which is a technique for getting lower bounds on the maximum-likelihood performance of code ensembles on BISC's. This method provides the best known bounds on the ML decoding thresholds of many code ensembles on many standard channel models. Just like the classical union bound, this method decouples the code ensemble from the

channel, but unlike the union bound, it is powerful enough to reproduce Shannon's theorem on general BISC's.

In Chapter 3, we introduce a new class of codes which we call irregular repeat-accumulate (IRA) codes. Like the class of irregular LDPC codes introduced by Luby et al. [33, 34], these codes are able to achieve capacity on the BEC with iterative decoding, and have thresholds extremely close to capacity on the BIAGN channel. In addition, they have the advantage of having a natural linear-time encoding algorithm. We also present some analysis on the near-capacity decoding complexity of these and other codes in this chapter.

While irregular LDPC codes and IRA codes both demonstrate extremely good near-capacity performance on a variety of channel models, this has to be checked either by simulation (for particular codes) or by density evolution (for ensembles) on a case by case basis. Except in the case of the BEC, there are few known analytical results regarding thresholds for iterative decoding. We take a step in this direction in Chapter 4 by deriving a lower bound on the iterative decoding threshold of a code ensemble on any BISC based on its BEC threshold.

In Chapter 5, we consider the question of whether turbolike codes are effective on channels that are not BISC's. In particular, we investigate the performance of IRA codes on some non-binary channel models, including the 2-D Gaussian channel and a simple multi-access channel.

Finally, we present some conclusions and suggestions for future work in Chapter 6.

Chapter 2 The Typical Set Bound

In this chapter, we will try to analyze the performance of code ensembles under optimal, or maximum likelihood (ML) decoding, on a general BISC. This allows us to study the intrinsic "goodness" of the code independent of the decoding algorithm, and thus also lets us measure the suboptimality of the particular decoding algorithm used. We will be interested in the asymptotic performance of code ensembles in the limit of large block length, and hence can define the *ML decoding threshold* of an ensemble of codes in the same way we defined its iterative decoding threshold in the previous chapter. The objective will be to find tight lower bounds on this threshold.

One easy way to bound the ML decoding performance of an ensemble is to use the classical union bound, but this technique gives very loose bounds unless the noise is very weak. Since it is, however, tight when the noise is low, it can be used to prove the existence of (nonzero) thresholds for many code ensembles. This has been done for the SCCC and PCCC cases in [14, 22]. Several techniques have been proposed that improve on the union bound, such as the Viterbi-Viterbi bound [50] and the Divsalar bound [13], but none of these techniques are powerful enough to demonstrate the capacity-achieving nature of random codes. Gallager [17] also gave a variational method for upper-bounding the ML decoded error probability for a code ensemble, but his method is very complex and does not easily yield threshold values.

However, any decoding algorithm that we can define provides a lower bound on the performance of ML decoding. Here, we define an auxiliary decoding algorithm called the *typical set decoder*, which though suboptimal, is easier to provide tight bounds for. This method is inspired by Shannon's proof of the channel coding theorem, but was used for the first time to analyze ensembles other than that of random codes by

MacKay [37], who used it to show that the ensemble of LDPC codes had a nonzero ML decoding threshold. The general method was subsequently developed in [3, 23]. A lot of the material presented in this chapter can be found in [20], but it also contains some previously unpublished material, especially regarding typical set decoding on the BEC and typical set decoding of cycle codes.

2.1 The Union Bound

In this section, we give a brief introduction to the classical union bound, primarily in order to demonstrate techniques for bounding the performance of linear codes on BISC's in terms of their *weight enumerators*.

Definition 2.1 The *weight enumerator* of a linear code of length n is an $n+1$-length list $\{A_0, A_1, \ldots, A_n\}$, where A_h is the number of codewords of weight h in the code. The *average weight enumerator* of an ensemble of linear codes is a list $\{\bar{A}_0, \bar{A}_1, \ldots, \bar{A}_n\}$, where $\bar{A}_h$ is the average number of codewords of weight h in a code belonging to the ensemble.

Consider the performance of a binary linear code $\mathcal{C}$ on some BISC under ML decoding. Let $\mathbf{C} = (C_1, C_2, \ldots, C_n)$ be a random variable representing the transmitted codeword (think of the elements as 0s and 1s) and $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_n)$ a random variable representing the corresponding channel output. Suppose that every codeword is a priori equiprobable. Then, because of the linearity of the code and the symmetry of the channel, it is easy to see that the joint distribution $p_{\mathbf{C},\mathbf{Y}}(\mathbf{c}, \mathbf{y})$ is invariant under translation by a codeword. That is, for any codeword $\mathbf{c}' = (c'_1, c'_2, \ldots, c'_n)$, if we define $\mathbf{c}^* = \mathbf{c} + \mathbf{c}'$ and $\mathbf{y}^* = ((-1)^{c'_1} y_1, (-1)^{c'_2} y_2, \ldots, (-1)^{c'_n} y_n)$, then $p_{\mathbf{C},\mathbf{Y}}(\mathbf{c}, \mathbf{y}) = p_{\mathbf{C},\mathbf{Y}}(\mathbf{c}^*, \mathbf{y}^*)$. As a consequence, the ML error probability given that a particular codeword was transmitted is the same as the overall probability of error, and hence, for

the purpose of computing the probability of error, we can assume that the all-zeros codeword was transmitted.

Under this assumption, a decoding error occurs if some codeword $\mathbf{c}$ is more likely than than the all-zeros codeword $\mathbf{0}$ given the channel output $\mathbf{y}$. This probability is upper bounded by the sum $\sum_{\mathbf{c}\in\mathcal{C}\setminus\{\mathbf{0}\}} \Pr(\Pr(\mathbf{C}=\mathbf{c}|\mathbf{Y}=\mathbf{y}) \geq \Pr(\mathbf{C}=\mathbf{0}|\mathbf{Y}=\mathbf{y}))$, which is the same as $\sum_{\mathbf{c}\in\mathcal{C}\setminus\{\mathbf{0}\}} \Pr(p_{\mathbf{Y}|\mathbf{C}}(\mathbf{y}|\mathbf{c}) \geq p_{\mathbf{Y}|\mathbf{C}}(\mathbf{y}|\mathbf{0}))$. Note that the only randomness now remaining is in the channel output $\mathbf{y}$, conditional on the transmitted codeword being $\mathbf{0}$. By symmetry, the terms in the summation can only depend on the Hamming weight of the codeword $\mathbf{c}$, and hence we can bound the probability of error P_e by $P_e \leq \sum_{h=1}^{n} A_h p_h$, where p_h is the probability that a given codeword of weight h is more likely than the all-zeros codeword given the output. It is well known that $p_h \leq \gamma^h$ (for a proof, see [38, Theorem 7.5]), where γ is the Bhattacharya parameter of the channel defined by

$$\gamma = \int_y \sqrt{p_{Y|X}(y|0)p_{Y|X}(y|1)}dy. \tag{2.1}$$

We can now write the classical union bound in its final form

$$P_e \leq \sum_{h=1}^{n} A_h \gamma^h. \tag{2.2}$$

In order to bound the probability of error of an ensemble of codes, we simply replace A_h by $\bar{A}_h$ in eq. (2.2). To use this bound to compute asymptotic results, we need a characterization of the weight enumerator as the length of the code approaches infinity. Notice that if we fix the rate, the number of codewords increases exponentially in the length, which motivates the following definitions:

Definition 2.2 The spectral shape $r_n(\delta)$ of an ensemble of length n linear codes is

defined as

$$r_n(\delta) = \frac{1}{n} \log \bar{A}_{\lfloor \delta n \rfloor}, \quad 0 < \delta < 1. \tag{2.3}$$

Here and subsequently in this chapter, all logarithms and entropy functions are assumed to have base 2.

Definition 2.3 The asymptotic spectral shape $r(\delta)$ of a sequence of ensembles with length going to ∞ is defined as

$$r(\delta) = \lim_{n \to \infty} r_n(\delta), \quad 0 < \delta < 1, \tag{2.4}$$

when the limit exists.

When the asymptotic spectral shape is well defined, we have $\bar{A}_{\delta n} \sim 2^{nr(\delta)}$. If we substitute this into the union bound, the r.h.s. becomes a sum of exponentials. If all of these have negative exponents, then $P_e \to 0$ as $n \to \infty$, while if any one exponent is positive, the bound diverges to ∞ (clearly, P_e cannot). There is a sort of "phase transition" between these two scenarios as we increase the noise, and this defines the union bound threshold of the ensemble. We shall not be more explicit here, but the idea will become clearer when we define a similar threshold for the typical set decoder in the next section.

2.2 The Typical Set Decoder

As mentioned earlier, the typical set decoder is inspired by Shannon's proof of the noisy coding theorem. The basic idea is that the decoder declares an error if the received vector is not jointly typical with any codeword (in particular with the transmitted codeword). This event has negligible probability in the limit of large length, so we do not expect any asymptotic performance degradation over the ML decoder be-

cause of this restriction. On the other hand, this restriction reduces double-counting in the union bound dramatically, and thus gives very tight lower bounds on the ML decoding threshold of various code ensembles.

The notion of typicality is usually defined only for finite (or countably infinite) alphabets, and hence we will initially restrict our attention to BISC's with finite output alphabets. Also, the definition of typicality we will use is somewhat stricter than the one encountered in most textbooks (eg., [10]).

Definition 2.4 Let $\mathbf{x}$ be a vector of length n with entries from some finite alphabet $\mathcal{A}$, and let X be a random variable over this alphabet with $p_a \triangleq \Pr(X = a),\ a \in \mathcal{A}$. Let n_a be the number of positions in $\mathbf{x}$ having entry a, and $f_a = n_a/n$ the corresponding fractional value. Then for any $\epsilon > 0$, we say that the vector $\mathbf{x}$ is ϵ-typical with respect to the pdf of X, or is a typical realization of X, if $|f_a - p_a| < \epsilon$ for every $a \in \mathcal{A}$.

The typical set decoder works by looking for codewords that are *jointly typical* with the received vector. Here, we use a simplified notion of joint typicality that works only for BISC's. Basically, we think of the channel as multiplying the input (thought of as ± 1s) by a "noise" random variable, which is distributed according to $p_{Y|X}(y|0)$, and we check for the typicality of this noise vector.

Definition 2.5 On a BISC, a received vector $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ is said to be ϵ-jointly typical with a codeword $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ if the "translated" received vector $((-1)^{c_1} y_1, (-1)^{c_2} y_2, \ldots, (-1)^{c_n} y_n)$ is ϵ-typical with respect to the conditional distribution $p_{Y|X}(y|0)$ of the channel output given input 0.

Definition 2.6 (The typical set decoder) Given a vector $\mathbf{y}$ received as the output of a BISC, the typical set decoder (with parameter $\epsilon > 0$) computes the set A of codewords which are ϵ-jointly typical with the received vector. If A is empty, the

decoder declares an error. If A contains exactly one codeword, the decoder decodes to that codeword. If A has more than one element, then the decoder decodes to the codeword $\mathbf{c}$ in A that maximizes $\Pr(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{c})$, where $\mathbf{X}$ and $\mathbf{Y}$ are random variables representing the transmitted and received vectors respectively. (Often, we will talk of using this decoder for an ensemble of codes containing codes of different lengths. In such a case, we will allow a different value of ϵ for each length n, i.e., the decoder will take a sequence ϵ_n as a parameter.)

By the weak law of large numbers applied to the noise vector, we know that for fixed ϵ, an $n \to \infty$, the probability that the received vector is ϵ-jointly typical with the transmitted codeword tends to 1. By a straightforward extension of the weak law, this statement still holds if ϵ is allowed to be a function of n, so long as ϵ decays slower than $1/\sqrt{n}$, i.e., if $\epsilon\sqrt{n} \to \infty$. Thus we have the following lemma:

Lemma 2.1 *Let ϵ_n be a sequence of positive numbers such that $\epsilon_n\sqrt{n} \to \infty$ as $n \to \infty$. Then for any code ensemble, for any BISC, as the length n of the ensemble tends to ∞, the probability that there is no codeword ϵ_n-jointly typical with the received vector tends to 0.*

This lemma basically confirms our idea of the notion of typicality, and says that in the limit of large n, the only errors that occur are the ones due to the noise vector being jointly typical with multiple codewords.

Consider the event that the typical set decoder makes an error. In this case, at least one of the following events must have occurred: 1) The received vector was not jointly typical with respect to the transmitted codeword, or 2) There was another codeword (in fact, another codeword jointly typical with the received vector) which was more likely to cause the received vector. In the second case, the ML decoder also makes an error, while the probability of the first event tends to zero with increasing n if the ϵ_n's do not decay too fast. This proves the following theorem:

Theorem 2.2 *Fix a sequence ϵ_n satisfying $\lim_{n\to\infty} \epsilon_n\sqrt{n} \to \infty$. If the probability of error of a code ensemble under ML decoding tends to 0 as n increases, then so does the probability of error under typical set decoding (with parameter ϵ_n at length n).*

The converse to this theorem is of course true because of the optimality of ML decoding. Another way to state this theorem is that the typical set decoder has the same *threshold* as the ML decoder. The notion of a threshold was first introduced in Chapter 1 in the context of iterative decoding. Let us formally define this notion here.

Definition 2.7 For a given decoding algorithm, a channel is said to be within the decoding threshold of an ensemble of codes if the decoded probability of error of the ensemble tends to 0 in the limit of increasing length. For a one-parameter family of channels, if the probability of error is monotone in the channel parameter, we often call the worst channel parameter for which the decoded probability of error tends to 0, the threshold of the code.

Although the typical set decoder is as good as the ML decoder in terms of threshold values, the bounds we will derive will for the most part assume that an error happens (or equivalently, is declared) whenever there is more than one codeword jointly typical with the received vector, and it is not clear whether a decoder with this property still has the same threshold as the ML decoder. In fact, Shannon's original typical set decoder did declare an error any time such an event occurred. The reason we need the stronger definition is to take care of certain technical problems regarding low-weight codewords (a problem not arising in the case of random codes).

2.3 The Typical Set Bound

By an argument that is entirely similar to the one in the union bound case, the typical set probability of error for a linear code $\mathcal{C}$ is independent of the codeword transmitted. Therefore assume that the all-zeros codeword is transmitted and consider again the event that the typical set decoder makes an error. In this case, either the received vector is not jointly typical with the all-zeros codeword, which happens with probability $o(1)$ (i.e., with probability tending to 0 with increasing n), or there is another codeword jointly typical with the received vector that is more likely than the transmitted codeword. Just as in the union bound case, we can bound the probability of this second event as a sum over all (nonzero) codewords, with each term being the probability that a specific codeword is both jointly typical with the received vector and more likely than the all-zeros codeword given the received vector. By symmetry, each term clearly depends only on the weight h of the codeword. We have already seen while deriving the union bound that the probability of being more likely than the all-zeros codeword is bounded by γ^h. Let us denote by P_h the probability of being jointly typical with the received vector. Then the probability of error P_e of the typical set decoder satisfies

$$P_e \leq \sum_{h=1}^{n} A_h \min(\gamma^h, P_h) + o(1). \tag{2.5}$$

As in the union bound case, for an ensemble of codes, we need to replace A_h by $\bar{A}_h$. We saw earlier that when the asymptotic spectral shape $r(\delta)$ of a code was well defined, $A_h \sim 2^{nr(\delta)}$. If we replace P_h by a similar exponential, then each term in the summation over h becomes an exponential in n, and goes to 0 if the exponent is negative. With this objective in mind, define the function $K(\delta)$ as

$$K(\delta) = \lim_{n\to\infty} \frac{1}{n} \log P_{\lfloor \delta n \rfloor} \tag{2.6}$$

under the assumption that the sequence of ϵ_n's that serves as a parameter to the decoder tends to 0 with increasing n, but $\epsilon_n\sqrt{n} \to \infty$. It is not hard to show that this limit exists and is uniform (i.e., $P_h = 2^{-n(K(\delta)+o(1))}$) for any BISC with a finite output alphabet, and to derive an explicit expression for $K(\delta)$ in terms of the channel transition probabilities. This has been done in [20] and the expression is reproduced in eq. (2.7) in Section 2.3.1.

Having defined $K(\delta)$ in this manner, we now have $A_h P_h \sim 2^{-n(K(\delta)-r(\delta))}$. Therefore, if $K(\delta) > r(\delta) \ \ \forall \delta \in (0,1)$, then we would expect that $P_e \to 0$ as $n \to \infty$. This is indeed true under some added technical conditions, and the formal statement is given by Theorem 2.3 in Section 2.3.2.

2.3.1 Properties of $K(\delta)$

In order to write down the expression for $K(\delta)$, we will need a concrete description of our channel. Let the channel be a BISC taking outputs $\{y_K, y_{K-1}, \ldots, y_1, y_0 = 0, y_{-1} = -y_1, \ldots, y_{-(K-1)} = -y_{K-1}, y_{-K} = -y_K\}$ with corresponding probabilities $\{p_K, p_{K-1}, \ldots, p_1, p_0, p_{-1}, \ldots, p_{-(K-1)}, p_{-K}\}$, given the channel input 0. For this channel, define $\delta_{\max} = p_0 + 2\sum_{i=1}^{K} \min(p_i, p_{-i})$. Then $K(\delta) = 0$ for $\delta > \delta_{\max}$ (in fact, it is not hard to see that $P_h = 0$ in this case), while for $\delta < \delta_{\max}$ it is given by

$$K(\delta) = H(\delta) - \sup_{\sum_{i=0}^{K} \delta_i = \delta} \left[p_0 H\left(\frac{\delta_0}{p_0}\right) + \sum_{i=1}^{K} \left(p_i H\left(\frac{\delta_i}{2p_i}\right) + p_{-i} H\left(\frac{\delta_i}{2p_{-i}}\right)\right)\right] \quad (2.7)$$

where the maximization is over all δ_i's for which the expression makes sense, i.e., satisfying $0 \le \delta_0 \le p_0$ and $0 \le \delta_i \le \min(2p_i, 2p_{-i}) \ \ \forall 1 \le i \le K$. The optimum δ_i's in eq. (2.7) are also computed in [20] using Lagrange multipliers.

Theorem A.1 in Appendix A tells us that $K(\delta)$ is a convex function for any BISC (in the interval $(0, \delta_{\max})$). It is therefore also continuous in this range, and has well defined left and right derivatives at every point. Moreover, by eq. (2.7), it is bounded

above by $H(\delta)$, and hence tends to 0 as $\delta \to 0$. Using the explicit characterization of the optimum δ_i's in equ(2.7), it is shown in [20] that $K'(0) = \lim_{\delta \to 0} K(\delta)/\delta = -\log \gamma$, where γ is the Bhattacharya parameter of the channel. It is not hard to see that $-\log \gamma = 0$ iff the BISC has zero capacity (in which case $K(\delta) \equiv 0$), and is positive otherwise. Thus, $K(\delta)$ is a convex, increasing function on $(0, \delta_{\max})$, and is strictly increasing unless the channel has zero capacity.

Another interesting property of $K(\delta)$ is that it is monotone under physical degradation of channels, i.e., if BISC 2 is a symmetrically degraded version of BISC 1 (by symmetrically degraded we mean that the degrading channel satisfies $p(y|x) = p(-y|-x)$), then $K_2(\delta) \leq K_1(\delta) \ \ \forall\ 0 < \delta < 1$. We will just indicate a proof of this fact in a very simple case, viz., when two outputs of a channel are combined into a single output. We will call this process "binning" a channel. Let BISC 1 have the description we have been using thus far, with $K(\delta)$ being given by eq. (2.7), and let δ_i^* denote the optimizing value of δ_i for $0 \leq i \leq K$. Suppose the outputs y_1 and y_2 are combined into a single output with probability $p_1 + p_2$, with of course the same thing happening to outputs y_{-1} and y_{-2}. If we now use the value $\delta_1^* + \delta_2^*$ for the combined output and keep the other δ_i^*'s unchanged, it is not hard to see that the value of the expression inside the supremum in eq. (2.7) increases. (This is a consequence of the fact that conditioning reduces entropy.) Therefore, the value of the supremum definitely increases and the value of $K(\delta)$ decreases.

2.3.2 The Main Theorem

We are now ready to state and prove the typical set bound.

Theorem 2.3 (The typical set bound) *On any BISC, for an ensemble of codes with spectral shape $r_n(\delta)$ at length n and asymptotic spectral shape $r(\delta)$, suppose that the following conditions hold.*

1. $K(\delta) > r(\delta) \ \ \forall\ 0 < \delta < 1$. *Moreover, for any* $\alpha > 0$, $\inf_{\delta>\alpha}(K(\delta) - r(\delta)) > 0$, *i.e.,* $K(\delta)$ *and* $r(\delta)$ *do not touch except at* $\delta = 0$.

2. $\limsup_{\delta\to 0} r(\delta)/\delta < \lim_{\delta\to 0} K(\delta)/\delta = -\log\gamma$. *(Note that the previous condition already implies that* $\limsup_{\delta\to 0} r(\delta)/\delta \leq -\log\gamma$. *This condition merely asks that the inequality be strict.)*

3. *For some sequence* d_n *of natural numbers tending to* ∞ *with increasing* n, $\sum_{h=1}^{d_n} \bar{A}_h \to 0$. *(This is in some sense saying that the code ensemble has minimum distance* $\geq d_n$*.)*

4. $r_n(\delta) = r(\delta) + o(d_n/n)$, *i.e.,* $r_n(\delta)$ *converges uniformly in* δ *to* $r(\delta)$ *at a fast enough rate.*

Then the channel lies within the typical set (and hence ML) decoding threshold of the code.

Proof:

Beginning with eq. (2.5), we get

$$\begin{aligned} P_e &\leq \sum_{h=1}^{n} \bar{A}_h \min(\gamma^h, P_h) + o(1) \\ &\leq \sum_{h=1}^{d_n} \bar{A}_h + \sum_{h=d_n+1}^{\alpha n} \bar{A}_h \gamma^h + \sum_{h=\alpha n}^{n} \bar{A}_h P_h + o(1) \qquad \text{for any } \alpha > 0 \\ &\leq o(1) + \sum_{h=d_n+1}^{\alpha n} 2^{h(r_n(\delta)/\delta + \log\gamma)} + \sum_{h=\alpha n}^{n} 2^{-n(K(\delta) - r(\delta) + o(1))} \end{aligned}$$

The last term in the above equation clearly goes to 0 for any $\alpha > 0$ because of condition 1. Let us look at the exponent in the second term. By condition 4, this exponent is bounded above by $h(\log\gamma + r(\delta)/\delta + o(d_n/h))$. Since the summation has only terms corresponding to $h > d_n$, $o(d_n/h) \leq o(1)$. Finally, by condition 2, for small enough α, the exponent is negative and bounded away from 0, say by $-\theta_0$,

where $\theta_0 > 0$. Then

$$\sum_{h=d_n+1}^{\alpha n} 2^{h(r_n(\delta)/\delta + \log\gamma)} \leq \sum_{h=d_n+1}^{\alpha n} 2^{-h\theta_0} \leq \frac{2^{-d_n\theta_0}}{1-2^{-\theta_0}},$$

which tends to 0 with increasing n because $d_n \to \infty$. Therefore $P_e \to 0$, which is the statement of the theorem. ■

Condition 3 in Theorem 2.3 asked that the code ensemble should have minimum distance going to ∞. If this condition is not satisfied, then it is not too hard to see that the ensemble has no ML decoding threshold. However, we can replace this condition by a slightly weaker one and ensure that the *bit error probability (BER)* goes to zero.

Theorem 2.4 *If, instead of condition 3 in Theorem 2.3, the ensemble has to satisfy only* $\sum_{h=1}^{d_n} \frac{h}{n}\bar{A}_h \to 0$ *for some sequence* $d_n \to \infty$*, and all the other conditions in Theorem 2.3 are satisfied, then the ensemble lies within the* BER threshold *of the channel under ML decoding.*

The proof of this theorem requires only a minor modification to the proof of Theorem 2.3, and will be omitted.

Having proved the typical set bound in its general form, we will now try and see what it tells us in the case of specific channel models and specific code ensembles.

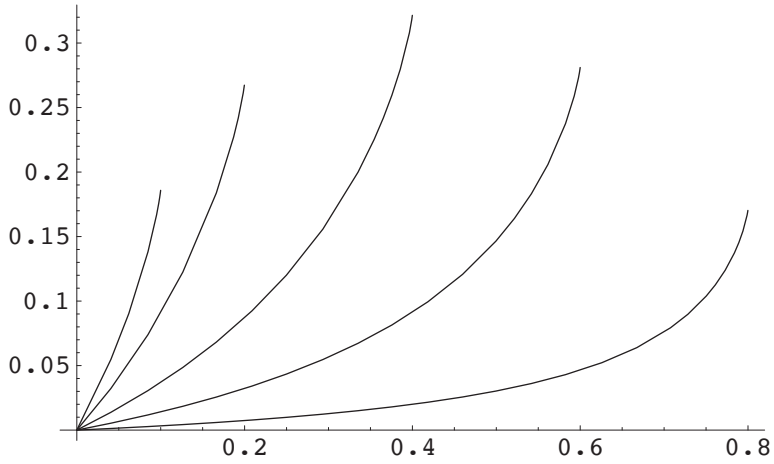

Figure 2.1: The function $K_p(\delta)$ for the BSC with $p = 0.05$, $p = 0.1$, $p = 0.2$, $p = 0.3$ and $p = 0.4$ (going from left to right).

2.4 The Typical Set Bound for Specific Channel Models

2.4.1 The Binary Symmetric Channel

For a BSC with crossover probability p, the expression in eq. (2.7) simplifies to

$$K_p(\delta) = H(\delta) - pH\left(\frac{\delta}{2p}\right) - (1-p)H\left(\frac{\delta}{2(1-p)}\right), \quad 0 < \delta < 2\min(p, 1-p). \quad (2.9)$$

Figure 2.1 shows the $K_p(\delta)$ curve for different values of p.

Since the BSC is a channel family ordered by physical degradation, by our previous observations, $K(\delta)$ decreases monotonically with p for $0 \leq p \leq 1/2$ for every δ. It is easy to check directly that this decrease is strictly monotonic. Another important property is that $K_p'(0) = -\log\gamma$ is strictly monotone in p. Using these properties, we can define the BSC threshold of an ensemble of codes as the largest p in $[0, 1/2]$ for which $K_p(\delta) \geq r(\delta) \ \ \forall\ 0 < \delta < 1$, assuming that the ensemble satisfies conditions 3 and 4 in Theorem 2.3. Theorem 2.3 tells us that under these conditions, the ensemble

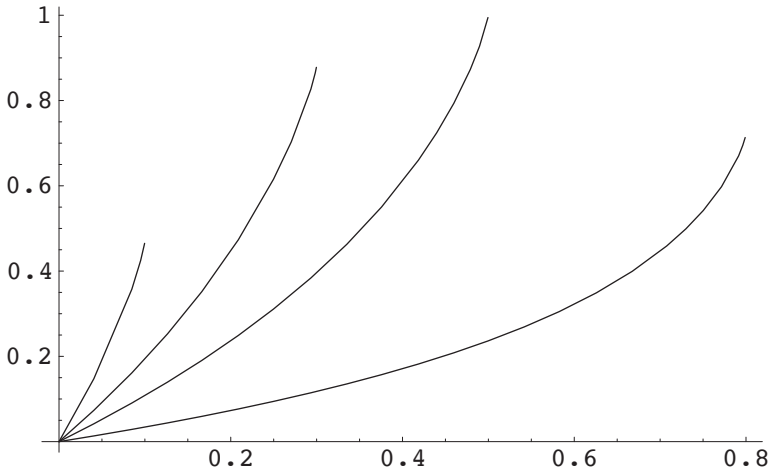

Figure 2.2: The function $K_p(\delta)$ for the BEC with $p = 0.1$, $p = 0.3$, $p = 0.5$ and $p = 0.8$ (going from left to right).

of codes has a probability of error going to 0 on any BSC with crossover probability strictly less than the threshold value.

2.4.2 The Binary Erasure Channel

On a BEC with erasure probability p, the expression for $K(\delta)$ takes the form

$$K_p(\delta) = H(\delta) - pH(\delta/p), \quad 0 < \delta < p. \tag{2.10}$$

Figure 2.2 shows the $K_p(\delta)$ curve for different values of p.

The BEC is also a family of channels ordered by physical degradation, so $K_p(\delta)$ is monotonically decreasing in p for every δ. It can be checked by differentiation that the increase is strictly monotonic in the region that $K_p(\delta)$ is finite. $K_p'(0) = -\log\gamma = -\log p$ is clearly strictly monotone in p. Therefore, exactly as in the BSC case, we can define the BEC threshold of a code ensemble to be the largest value of p for which $K_p(\delta) - r(\delta) \geq 0 \ \ \forall\ 0 < \delta < 1$. Assuming again that the ensemble of codes satisfies conditions 3 and 4 in Theorem 2.3, its probability of error goes to 0 with increasing

n on any BEC with erasure probability smaller than the threshold value.

Recall Theorem 2.2, which said that the typical set decoder has the same threshold as the ML decoder on any BISC with a finite output alphabet. However, according to our definition, the typical set decoder is just an ML decoder restricted to jointly typical codewords. We could also define a "reduced" typical set decoder, which declares an error whenever there is more than one codeword jointly typical with a given received vector. However, our proof of the typical set bound does not extend to this decoder owing to technical difficulties regarding low-weight codewords. On the BEC, however, the situation is somewhat simpler. Notice that if there are multiple codewords consistent with the received vector (i.e., equal to it on the non-erased positions), their a posteriori probabilities given the received vector are equal, and the ML decoder makes an error at least half the time in such a situation. Joint typicality in this case reduces to the codeword being consistent with the received vector, which in turn should have approximately np erasures. In this case, the reduced typical set decoder, which declares an error if there is more than one codeword consistent with the received vector, is at most twice as bad as the ML decoder (in the typical situations) and hence has the same threshold.

Theorem 2.5 *On the BEC, the reduced typical set decoder, which declares an error unless there is a unique codeword jointly typical with the received vector, and decodes to this unique jointly typical codeword in this case, has the same decoding threshold as the ML decoder.*

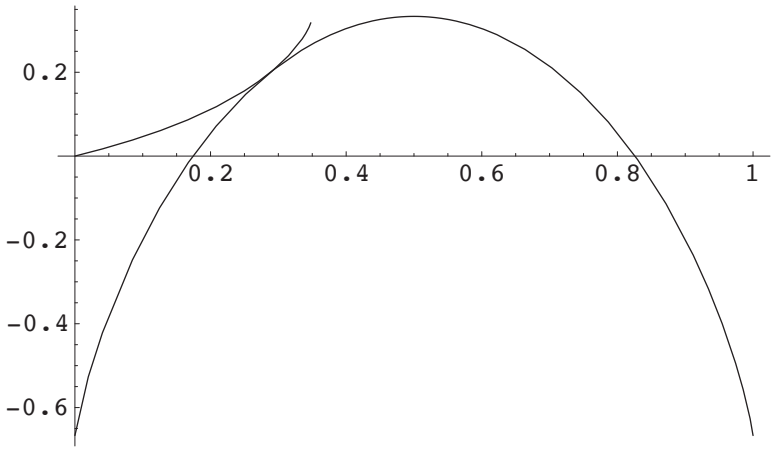

Figure 2.3: The asymptotic spectral shape of the ensemble of rate 1/3 random linear codes, together with the $K(\delta)$ curve for the BSC with crossover probability $p = 0.174$, which is the BSC capacity at this rate.

2.5 The Typical Set Bound for Specific Code Ensembles

2.5.1 The Ensemble of Random Linear Codes

It is well known that random linear codes achieve capacity on BISC's [18]. In this section we show that the typical set bound is strong enough to reproduce this result. The asymptotic weight spectrum of the ensemble of random linear codes of rate R is known to be the same as that of rate R random codes, and is given by

$$r(\delta) = H(\delta) - (1 - R). \tag{2.11}$$

Let us now compute the capacity of the BISC described in Section 2.3.1. Let X denote its input random variable and Y its output random variable. By symmetry, the capacity achieving distribution on X is the uniform one. For this distribution, $H(X) = 1$. With this input distribution, the output y_i of the BISC occurs with probability $(p_i + p_{-i})/2$, and the a posteriori probabilities of 0 and 1 given the output

y_i are $p_i/(p_i+p_{-i})$ and $p_{-i}/(p_i+p_{-i})$, respectively. Therefore the capacity C is given by

$$C = 1 - \sum_{i=-K}^{K} \frac{p_i + p_{-i}}{2} H\left(\frac{p_i}{p_i + p_{-i}}\right) = 1 - p_0 - \sum_{i=1}^{K} (p_i + p_{-i}) H\left(\frac{p_i}{p_i + p_{-i}}\right). \quad (2.12)$$

Now look at the expression for $K(\delta)$ given in eq. (2.7). Clearly, we can give a lower bound on $K(\delta)$ by replacing the constrained maximum by an unconstrained maximum. This can be done separately for each term in the summation, and the optimizing values of the δ_i's are $\delta_0 = p_0/2$ and $\delta_i = 2p_i p_{-i}/(p_i + p_{-i})$ for $i > 0$. Substituting these values, we get

$$\begin{aligned} K(\delta) &\geq H(\delta) - p_0 - \sum_{i=1}^{K} \left[p_i H\left(\frac{p_i}{p_i + p_{-i}}\right) + p_{-i} H\left(\frac{p_{-i}}{p_i + p_{-i}}\right) \right] \\ &= H(\delta) - p_0 - \sum_{i=1}^{K} (p_i + p_{-i}) H\left(\frac{p_i}{p_i + p_{-i}}\right) \\ &= H(\delta) - (1 - C). \end{aligned}$$

Comparing this with the expression for $r(\delta)$ that we had earlier, $K(\delta) - r(\delta)$ is bounded below by $C - R$ for any $R < C$, thus satisfying condition 1 of Theorem 2.3. Condition 2 is true because $r(\delta)$ is negative near $\delta = 0$. Moreover, the ensemble of random linear codes is known to have minimum distance growing linearly with n, i.e., d_n in condition 3 can be chosen to be αn for some $\alpha > 0$. We then require that $\theta_n = o(1)$, i.e., that $r_n(\delta)$ converges uniformly to $r(\delta)$, which is known to be true. Having thus verified all the conditions in Theorem 2.3, we have

Theorem 2.6 *On any BISC with a finite output alphabet, the ensemble of random linear codes of any rate less than capacity has probability of error going to 0 with increasing n, i.e., the ensemble of random linear codes achieves capacity on such a channel.*

Even though we have formulated our bound for ensembles of linear codes, it clearly applies to the ensemble of random binary codes with a uniform distribution on 0 and 1, because the probability of error is independent of the transmitted codeword for this ensemble as well. The above argument then shows that the ensemble of random codes also achieves capacity on a BISC, which is a special case of Shannon's celebrated result.

Finally, as an aside, for the ensemble of random linear codes with $R = C$, the curves $K(\delta)$ and $r(\delta)$ touch at the point where the unconstrained maximization and the constrained maximization yield the same results, i.e., for $\delta = \frac{p_0}{2} + \sum_{i=1}^{K} \frac{2p_i p_{-i}}{p_i + p_{-i}}$. Figure 2.3 shows the asymptotic spectral shape of the ensemble of rate 1/3 random linear codes, together with the $K(\delta)$ function for the BSC with crossover probability $p = 0.174$, which is the BSC capacity at this rate.

2.5.2 LDPC Codes

The ensemble of (j, k) LDPC codes was defined in Example 1.14. In order to compute typical set thresholds for this ensemble, we need to know its asymptotic spectral shape. Gallager [17] derived the asymptotic spectral shape of a modified ensemble of LDPC codes in parametric form, and it was shown in 2001 by Litsyn and Shevelev [32] that the ensemble as we have defined it here also has the same $r(\delta)$ function. It is given in parametric form by the following equations:

$$\delta_{j,k}(s) = \frac{u'_k(s)}{k} \tag{2.13}$$

$$r_{j,k}(s) = \frac{1}{\ln 2}\frac{j}{k}\left[u_k(s) - s u'_k(s)\right] - (j-1)H\left(\frac{u'_k(s)}{k}\right), \tag{2.14}$$

where the parameter s varies over $\mathbb{R}$, $H(\cdot)$ is of course the entropy function, and $u_k(s)$ is defined by

$$u_k(s) = \ln \frac{(1+e^s)^k + (1-e^s)^k}{2}. \tag{2.15}$$

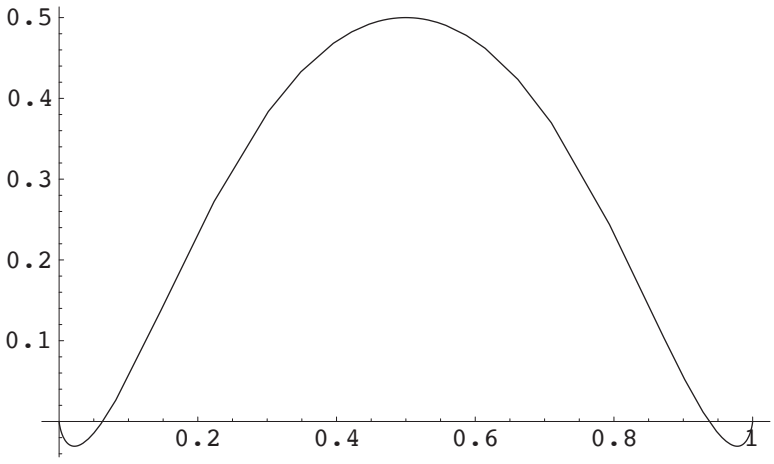

Figure 2.4: The asymptotic spectral shape of the ensemble of $(4, 8)$ LDPC codes.

(In the case when k is odd, as s varies over $\mathbb{R}$, $\delta_{j,k}(s)$ only takes values in $(0, \frac{k-1}{k})$, and $r(\delta)$ is $-\infty$ outside this range). The resulting $r(\delta)$ function is plotted in Figure 2.4 for the ensemble of $(4, 8)$ LDPC codes.

Furthermore, it is known that for $j \geq 3$, LDPC codes have minimum distance growing linearly with n, i.e., we can choose $\alpha > 0$ s.t. condition 3 of Theorem 2.3 is satisfied with $d_n = \alpha n$. This is reflected in the fact that the $r(\delta)$ function is negative for small δ (see Figure 2.4). This also implies that $\limsup_{\delta \to 0} r(\delta)/\delta \leq 0$, so that condition 2 of Theorem 2.3 is satisfied. Moreover, for our choice of d_n, as in the case of random linear codes, condition 4 reduces to uniform convergence of the spectral shape, which is also known to be true. Therefore, we only have to check condition 1 to verify that a BISC lies within the decoding threshold of an ensemble of LDPC codes with $j \geq 3$. Table 2.1 gives thresholds on the BSC and the BEC computed in this manner for different values of j and k. For comparison, this table also lists the iterative decoding (sum-product algorithm) thresholds (labeled RU thresholds in Tables 2.1, 2.2 and 2.3) of these ensembles computed using density evolution in [42].

If we let j and k go to ∞ in such a manner that the rate of the ensemble $1 - j/k$ tends to some positive constant R, then it can be shown that the asymptotic spectral

(j,k)	Rate	BSC			BEC		
		Cap.	Typ. Set Thresh.	RU Thresh.	Cap.	Typ. Set Thresh.	RU Thresh.
$(3,6)$	$1/2$	0.109	0.0915	0.084	0.5	0.483	0.429
$(3,5)$	$2/5$	0.145	0.129	0.113	0.6	0.587	0.517
$(4,6)$	$1/3$	0.174	0.170	0.116	0.667	0.665	0.506
$(3,4)$	$1/4$	0.214	0.205	0.167	0.75	0.744	0.647
$(2,3)$	$1/3$	0.174	0.067	0.067	0.667	0.5	0.5
$(2,4)$	$1/2$	0.109	0.0286	0.0286	0.5	0.333	0.333

Table 2.1: Comparison of capacity, typical set threshold and iterative decoding (RU) threshold for different ensembles of LDPC codes on the BSC and the BEC.

shape $r_{j,k}(\delta)$ converges to $H(\delta) - (1 - R)$ for $0 < \delta < 1$, which is the asymptotic spectral shape for the ensemble of random linear codes of rate R. Moreover, this convergence is uniform on any closed subinterval of $(0, 1)$. Together with the fact that the ensemble has minimum distance that grows linearly in n, this shows that if a BISC lies within the ML decoding threshold of the ensemble of random linear codes of rate R, then it also lies within the decoding threshold of the ensemble of (j, k) LDPC codes for large enough j and k with rate $1 - j/k$ being arbitrarily close to R. In other words, the ensemble of (j, k) LDPC codes under ML decoding achieves capacity on any BISC as j and k tend to ∞.

2.5.3 Cycle Codes

In the previous section, we saw how the conditions in Theorem 2.3 were satisfied by LDPC codes with $j \geq 3$. The case $j = 2$ is somewhat trickier. These codes are called *cycle codes* because their codewords can be viewed as the cycles of an undirected graph. It is not hard to see that in this case, for any fixed h, as $n \to \infty$, $\bar{A}_h$ tends to a nonzero limit, so that condition 3 in Theorem 2.3 is not satisfied for any sequence d_n going to ∞. However, the conditions of Theorem 2.4 are satisfied, so that the BER of the ensemble goes to 0 whenever conditions 1 and 2 of Theorem 2.3 are satisfied.

Moreover, it turns out that $r(\delta)$ is a concave function in the case of cycle codes, with $r(0) = 0$ and $r'(0) = \log(k-1)$. Since Theorem A.1 in Appendix A tells us that that $K(\delta)$ is convex for any BISC, we see that $K(\delta) - r(\delta)$ is a convex function. Since we also have $K(0) - r(0) = 0$, we see that $K(\delta) - r(\delta) > 0 \; \forall \delta \iff K'(0) - r'(0) > 0$, i.e., condition 1 in Theorem 2.3 is equivalent to condition 2. Therefore, the typical set bound for cycle codes says that a BISC lies within the decoding threshold of an ensemble of cycle codes (in the BER sense), if $-\log \gamma > \log(k-1)$, i.e., if

$$\gamma < \frac{1}{k-1}. \tag{2.16}$$

Let us use this formula to explicitly compute the thresholds in the case of the BSC and the BEC. The BEC threshold is simply given by

$$p^* = \frac{1}{k-1}, \tag{2.17}$$

where p^* is of course the threshold channel erasure probability. On the BSC, the equation is $2\sqrt{p^*(1-p^*)} = 1/(k-1)$, i.e., $4p^*(1-p^*) = 1/(k-1)^2$, which is a quadratic equation. Its solution is given by

$$p^* = \frac{1}{2}\left(1 - \sqrt{1 - \frac{1}{(k-1)^2}}\right), \tag{2.18}$$

where p^* is the threshold crossover probability.

The numerical values of these thresholds for some values of k are also shown in Table 2.1. We can see from this table that the typical set bound seems to coincide with the iterative decoding threshold. We will give a proof of this fact (i.e., that a BISC lies within the iterative decoding threshold of an ensemble of cycle codes iff it satisfies eq. (2.16)) in Chapter 4. Moreover, for the BSC, Decreusefond and Zémor [11] have shown that the typical set bound, given by eq. (2.18), is also the

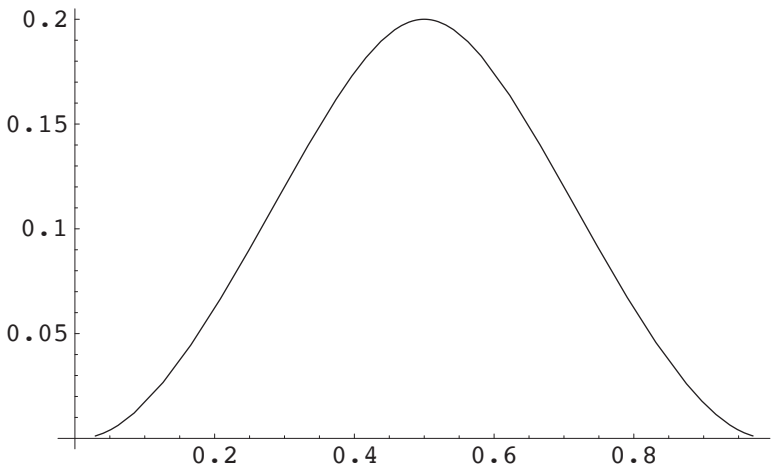

Figure 2.5: The asymptotic spectral shape of the ensemble of $q = 5$ RA codes.

exact ML decoding threshold for an ensemble of expurgated cycle codes. Cycle codes are thus among the most well-understood code ensembles, and it appears that in this case, the typical set bound, the ML threshold and the iterative decoding threshold coincide.

2.5.4 RA Codes

In this section, we apply the typical set bound to the ensemble of repeat-accumulate codes defined in Section 1.2.6. The weight enumerator of this ensemble was derived in [15], and the asymptotic spectral shape $r(\delta)$ can be found in [3]. It is given by

$$r_q(\delta) = \sup_{0 \leq x \leq 2\min(\delta, 1-\delta)} \left[-\frac{q-1}{q} H(x) + (1-\delta) H\left(\frac{x}{2(1-\delta)}\right) + \delta H\left(\frac{x}{2\delta}\right) \right]. \quad (2.19)$$

Figure 2.5 shows a plot of $r(\delta)$ for the case $q = 5$.

It can be shown that this ensemble satisfies conditions 3 and 4 of Theorem 2.3 with $d_n = \log^2 n$ for $q \geq 3$. Furthermore, $r'(0)$ can be shown to be 0, so that condition 2 is automatically satisfied. Therefore, exactly as in the LDPC case, we only need to check condition 1 to see whether a BISC lies within the decoding threshold of

q	Rate	BSC			BEC		
		Cap.	Typ. Set Thresh.	RU Thresh.	Cap.	Typ. Set Thresh.	RU Thresh.
3	1/3	0.174	0.132	0.142	0.667	0.629	0.617
4	1/4	0.215	0.191	0.188	0.75	0.735	0.703
5	1/5	0.243	0.228	0.216	0.8	0.792	0.75
6	1/6	0.264	0.254	0.235	0.833	0.829	0.78

Table 2.2: Comparison of capacity, typical set threshold and iterative decoding (RU) threshold for different ensembles of RA codes on the BSC and the BEC.

this ensemble. Numerical thresholds for the BSC and the BEC for some values of q are shown in Table 2.2. The case $q = 3$ on the BSC is of special interest, because in this case the iterative decoding threshold is actually greater than the typical set bound. This is the only such example that we know of, and the only evidence that the typical set bound is not tight. The same phenomenon is observed on the BIAGN channel. (We will see in the next section how the typical set bound can be applied to continuous output channels). As seen from the table, however, no such reversal is observed on the BEC, and we are hopeful that the typical set bound actually turns out to be tight on this channel.

The case $q = 2$ poses similar problems to the $j = 2$ LDPC case. $q = 2$ RA codes also do not possess a word error rate threshold due to the presence of low-weight codewords. However, in this case too we can show that the conditions of Theorem 2.4 are satisfied, thus allowing us to compute a lower bound on the BER threshold.

2.6 Generalization to Arbitrary BISC's

In Section 2.3.1, we observed that the function $K(\delta)$ decreased pointwise under physical degradation of channels. In particular, it decreased under the process of "binning" which consists of grouping a set of outputs into a single output. An arbitrary BISC can be "binned" in many ways into a BISC with a finite number of outputs. We de-

Code Ensemble	Rate	Capacity	Typical Set Threshold	RU Threshold
$(3,6)$ LDPC	1/2	0.187dB	0.673dB	1.11dB
$(4,6)$ LDPC	1/3	-0.495dB	-0.423dB	1.674dB
$(3,4)$ LDPC	1/4	-0.794dB	-0.510dB	1.003dB
$q=3$ RA	1/3	-0.495dB	0.739dB	0.479dB
$q=4$ RA	1/4	-0.794dB	-0.078dB	0.106dB
$q=5$ RA	1/5	-0.963dB	-0.494dB	0.044dB
$q=6$ RA	1/6	-1.071dB	-0.742dB	0.085dB

Table 2.3: Comparison of capacity, typical set threshold and iterative decoding (RU) threshold for different ensembles of LDPC and RA codes on the BIAGN channel.

fine $K(\delta)$ for an arbitrary BISC to be the supremum over all its binned finite-output versions. An important point to observe is that $K(\delta)$ can be approximated uniformly by a fine enough binning of the channel. Therefore, for a BISC with an arbitrary output set, if $K(\delta)$ is separated from the $r(\delta)$ of some code ensemble away from 0, then we can find a degraded finite-output BISC, whose $K(\delta)$ is also larger than the $r(\delta)$ away from 0. Applying Theorem 2.3 to the new channel tells us that it lies within the ML decoding threshold of the code ensemble under consideration, and hence so does the original channel. Thus we have the following theorem:

Theorem 2.7 *For an arbitrary BISC and an arbitrary code ensemble, if conditions 1–4 of Theorem 2.3 are satisfied, then the BISC lies within the ML decoding threshold of the code ensemble.*

An exactly analogous statement is true regarding Theorem 2.4 and BER thresholds. Theorem 2.6 also holds in the continuous output case. Notice that we don't define a typical set decoder in the continuous output case, but merely use it on a suitably constructed finite-output channel.

For channels having a continuous density function for the output (given input 0), it is easy to write down an explicit expression for $K(\delta)$ by replacing the summations

in eq. (2.7) by integrals, and the maximization inside the expression can again be done explicitly using Lagrange multipliers. Table 2.3 has some numerical thresholds computed using this method for the BIAGN channel. Note again that for $q = 3$ RA codes, the iterative decoding threshold is greater than the typical set bound, demonstrating that the typical set bound is not tight at least in this case.

Chapter 3 Irregular Repeat-Accumulate Codes

In the previous chapter, we looked at bounds on the performance of code ensembles under ML decoding. We will now shift our attention to their iterative decoding performance, and in particular, to finding ensembles with good iterative decoding performance.

We described several graphical code ensembles in Chapter 1, which are known to have extremely good iterative decoding performance. Several variants of these have been introduced in the literature demonstrating improvements in performance, complexity, error floor, or some other characteristic. A major breakthrough, however, was the introduction of irregular LDPC codes by Luby et al. [33, 34], who showed that in the limit of large length, these codes could achieve capacity on the BEC. Richardson et al. [43] then generalized their techniques to general BISC's, and found irregular LDPC code ensembles with thresholds extremely close to capacity on several channel models, including the BSC and the BIAGN channel. Ensembles with even better threshold values were found later by Chung [8]. This suggests that irregular LDPC codes might actually be able to achieve capacity on all BISC's, though there is no proof yet of this fact on any channel other than the BEC.

In spite of their impressive performance, irregular LDPC codes have certain disadvantages. As in the case of "regular" LDPC codes, these codes have a sparse parity-check matrix and a generator matrix that is typically not sparse. Since encoding is based on the generator matrix, both regular and irregular LDPC codes have a natural quadratic-time encoding algorithm. Richardson and Urbanke [44] introduced

an encoding algorithm for these codes that is "almost" linear-time (i.e., there is a quadratic term but with a small coefficient), and in fact exactly linear-time for some ensembles. This improved algorithm, however, is still quite complicated, and poses an obstacle to their implementation.

To get around this problem, we will apply the concept of irregularity to the ensemble of RA codes described in Section 1.2.6, and show that this modified ensemble has most of the desirable properties of irregular LDPC codes. In addition, these codes, which we call irregular repeat-accumulate (IRA) codes, also have a straightforward linear-time encoding algorithm.

In general, the analysis of IRA codes mirrors that of irregular LDPC codes, with the individual steps being somewhat more complicated. Hence, for the sake of simplicity, we will sometimes use irregular LDPC codes to illustrate properties of IRA codes. We will therefore begin by defining the ensemble of irregular LDPC codes, and then go on to define the ensemble of IRA codes. We will then show that they achieve capacity on the BEC and review their performance on the BIAGN channel.

3.1 Irregular LDPC Codes: Definition

The ensemble of (j, k) LDPC codes was defined as the ensemble of all Tanner graphs, where every variable node has degree j and every check node has degree k (see Definition 1.14). In the case of irregular LDPC codes, the variable and check node degrees are no longer all the same. Instead, a fraction f_1 of variable nodes have degree 1, a fraction f_2 have degree 2, and so on up to some maximum degree N. The check node degrees have a similar variation with associated fractions. If we number the nodes and fix the degree of each node, we can think of i sockets coming out of a variable node of degree i, and similarly in the case of check nodes. Of course, the total number of sockets coming out of variable and check nodes has to be the same, and the ensemble

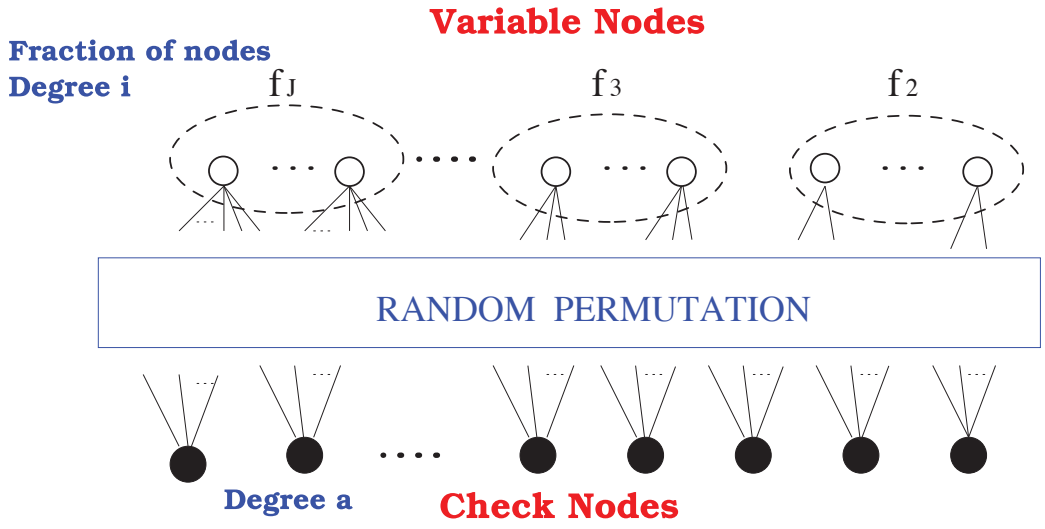

Figure 3.1: The Tanner graph of an irregular LDPC code.

is defined formally by connecting the two sets of sockets via a random permutation in order to form edges. Figure 3.1 shows an irregular LDPC code with varying variable node degrees and constant check node degree a. The code is not shown to have any degree 1 variable nodes, since such codes turn out to have no iterative decoding threshold.

Following [33, 34], let λ_i be the fraction of edges adjacent to a variable node of degree i, and ρ_i the fraction of edges adjacent to a check node of degree i. For purposes of analysis, it is more convenient to deal with these "edge-fractions" rather than the corresponding node fractions, though it is easy to convert between the two representations. (We will see the explicit conversion formulae in the case of IRA codes.) The set of fractions λ_i and ρ_i are called a *degree distribution*. It is also convenient to represent the degree distributions in polynomial form as

$$\lambda(x) = \sum_i \lambda_i x^{i-1}, \quad \rho(x) = \sum_i \rho_i x^{i-1}. \tag{3.1}$$

$\lambda(x)$ is called the variable node degree polynomial, and $\rho(x)$ is called the check node

degree polynomial. The ensemble given by this degree distribution is called the ensemble of (λ, ρ) LDPC codes. Clearly, $\lambda(1) = \rho(1) = 1$, since the λ_i's and the ρ_i's sum to 1. It is an easy exercise to see that the rate of this ensemble is given by

$$R = 1 - \frac{\sum_i \rho_i / i}{\sum_i \lambda_i / i} = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}. \tag{3.2}$$

(For a proof of this formula, see [47].)

3.2 Irregular LDPC Codes on the BEC

In Section 1.3.1, we described how the sum-product algorithm and the technique of density evolution simplify in the case of the BEC. In fact, the sum-product algorithm has very intuitive update rules in this case. At a variable node, the outgoing message is an erasure (i.e., has log-likelihood 0) only if all the incoming messages (including the channel LLR) are erasures, else it is equal to the non-erasure incoming messages. (This is well defined since there are never any clashes.) At a check node, the outgoing message is an erasure if any one of the incoming messages is an erasure, else it corresponds to the binary sum of the incoming messages (thinking of them as certain 0's or certain 1's instead of LLR's). Therefore, if the x_i's are the probabilities of erasure of the incoming messages at a variable node, and p is the channel probability of erasure, then the probability of erasure of the outgoing message is $p \prod_i x_i$. Similarly, if the x_i's are the probabilities of erasure of the incoming messages at a check node, then the probability of erasure of the outgoing message is $1 - \prod_i (1 - x_i)$.

Now suppose that at some iteration, the probability of message erasure for messages coming out of variable nodes is x. Then, at a check node of degree i, the probability of erasure of the outgoing message is $1 - (1 - x)^{i-1}$. Because of the random permutation, the probability of erasure of the incoming message at any variable node is this quantity averaged over the check node degrees, i.e., $\sum_i \rho_i \left[1 - (1 - x)^{i-1}\right] =$

$1-\rho(1-x)$. By the same argument, the probability of erasure of the incoming message at a check node at the next iteration is $p\lambda(1-\rho(1-x))$. Therefore if we have

$$p\lambda(1-\rho(1-x)) < x \quad \forall x > 0, \tag{3.3}$$

then the probability of message erasure goes to 0 in the number of iterations, and the BEC with probability of erasure p lies within the decoding threshold of the ensemble of (λ, ρ) LDPC codes.

Given this simple criterion, we would like to maximize the rate of the ensemble while treating eq. (3.3) as a constraint. There are several known sequences of degree distributions (also called *degree sequences*) whose rate in fact tends to capacity (i.e., $1-p$) while satisfying this constraint. (For a comprehensive treatment of capacity-achieving sequences, see [48].) Here, we will describe the one introduced in [47], because it is similar to the approach we will take for IRA codes.

Firstly, we choose the sequence to be *right-regular*, i.e., we assume that every check node has the same degree a. Therefore $\rho(x) = x^{a-1}$, and eq. (3.3) can be transformed to

$$\lambda(x) < \frac{1}{p}\left[1-(1-x)^{1/(a-1)}\right]. \tag{3.4}$$

The r.h.s. can be expanded explicitly in a power series around $x=0$, whose coefficients turn out to be non-negative. (See Lemma B.1 in Appendix B.) We then choose $\lambda(x)$ simply by truncating this power series (with the number of terms to be kept given by $\lambda(1)=1$), which certainly satisfies the required constraint. Notice that substituting the entire r.h.s. of the above equation for $\lambda(x)$ (which would however not satisfy $\lambda(1)=1$) into eq. (3.2) would make the rate equal to $1-p$, which is the capacity of the BEC. It is shown in [47] that the discarded terms contribute negligible rate loss in the limit of large a. (In fact, [47] shows that the difference between the rate and the capacity dies exponentially with a.)

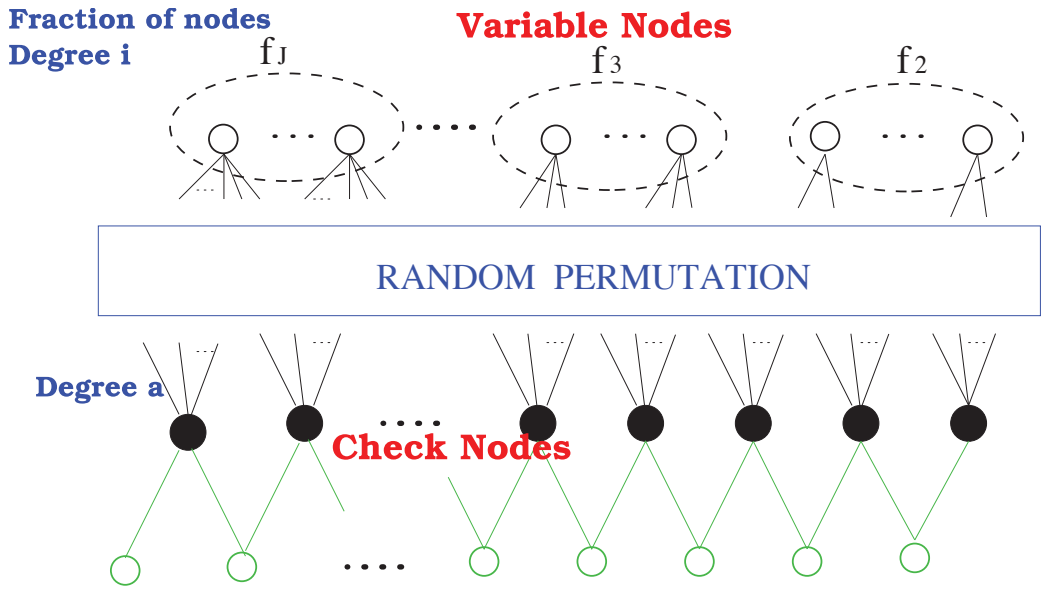

Figure 3.2: The Tanner graph of an IRA code.

3.3 IRA Codes: Definition and Notation

Having reviewed the basic properties of irregular LDPC codes, let us now apply the concept of irregularity to the ensemble of RA codes defined in Section 1.2.6 to get the ensemble of irregular RA codes. The Tanner graph of an irregular RA code is shown in Figure 3.2. IRA codes differ form RA codes in three ways (for comparison, look at Figure 1.6):

1. The variable nodes on the top, which represent information bits and are called *information nodes*, now have variable degrees. A fraction f_2 of them have degree 2, a fraction f_3 have degree 3, up to some maximum degree N. The degree of an information node is the number of times it is repeated. (We neglect codes having information nodes of degree 1 for the same reason as in the irregular LDPC codes case, i.e., they have no iterative decoding threshold.)

2. The check nodes now have multiple edges adjacent from above. In the figure, the number of such edges is a constant a, but these could be varied just as the

information node degrees.

3. IRA codes are systematic, i.e., the information nodes also represent elements of the codeword.

Allowing multiple edges to be adjacent to a check node from above allows us to construct IRA codes of arbitrary rates, as opposed to RA codes, which could only have rates of the form $1/q$ for integer q. On the other hand, this forces us to make the code systematic, because if every check node were to have more than one edge adjacent to it from above and the code were non-systematic, it is not hard to see that iterative decoding would stall at the first iteration.

Just as in the case of RA codes, we can still read out an encoding algorithm from the Tanner graph. Each information bit is first repeated (the number of repetitions being different for each bit) and the resulting bits are permuted randomly. They are then "collated," i.e., groups of bits are replaced by their binary sum. Finally, the resulting sequence of bits is passed through an accumulator to get the parity bits. (We will call the nodes at the bottom *parity nodes*.) This algorithm clearly has linear complexity in the length of the code.

Let k be the number of information bits for the IRA code shown in Figure 3.2. Then the number of information nodes of degree i is $f_i k$, and the total number of edges connecting the information nodes to the check nodes is $k \sum_i i f_i$. Since each check node has degree a in the figure, the number of check nodes, which is equal to the number of parity nodes, is $(k \sum_i i f_i)/a$. The rate of the ensemble is therefore given by

$$R = \frac{k}{k + (k \sum_i i f_i)/a} = \frac{a}{a + \sum_i i f_i}. \tag{3.5}$$

As in the case of irregular LDPC codes, it is more convenient for the purposes of analysis to deal with edge fractions rather than node fractions. Therefore, define λ_i to be the fraction of edges (of the edges connecting information nodes and parity nodes)

adjacent to an information node of degree i. Also, to be general, define ρ_i to be the fraction of edges (again, of the edges connecting information nods and parity nodes) adjacent to a check node of degree $i+2$ (i.e., connected to i information nodes and 2 parity nodes). Also define the variable node degree polynomial $\lambda(x) = \sum_i \lambda_i x^{i-1}$ and the check node degree polynomial $\rho(x) = \sum_i \rho_i x^{i-1}$. Just as in the case of irregular LDPC codes, we will refer to this ensemble as the ensemble of (λ, ρ) IRA codes. It is easy to compute the f_i's from the λ_i's and vice versa. For example, f_i is given in terms of the λ_i's by

$$f_i = \frac{\lambda_i / i}{\sum_j \lambda_j / j}. \tag{3.6}$$

An identical equation holds for the check node degree fractions. From this equation and our previous expression for the rate, it is easy to see that the rate is given in terms of the λ_i's and ρ_i's by

$$R = \left(1 + \frac{\sum_i \rho_i / i}{\sum_i \lambda_i / i}\right)^{-1} = \left(1 + \frac{\int_0^1 \rho(x) dx}{\int_0^1 \lambda(x) dx}\right)^{-1}. \tag{3.7}$$

We would like to study the performance of IRA codes under iterative decoding. The decoding algorithm we will use is the sum-product algorithm for a Tanner graph with canonical scheduling (see Section 1.2.4). It is easy to see that IRA codes satisfy the RU condition stated in Definition 1.15, and we can therefore use density evolution to analyze this ensemble. In practice, we can also use a "turbo-style" decoding algorithm, which would only be a different scheduling of the messages. However, we would no longer be able to use density evolution to rigorously analyze this algorithm.

3.4 IRA Codes on the BEC

3.4.1 Fixed Point Analysis of Iterative Decoding

We will now derive a sufficient condition similar to eq. (3.3) for the bit error probability under iterative decoding on the BEC to go to 0 with the number of iterations. To this end, we first assume that density evolution has reached a fixed point, and analyze the various probabilities of message erasure under this assumption. (It is shown in [43] that density evolution for the sum-product algorithm always converges to a fixed point.)

At this fixed point, let x_0 be the probability of message erasure along an edge from an information node to a check node, x_1 the probability of message erasure along an edge from a check node to a parity node, x_2 the probability of message erasure along an edge from a parity node to a check node, and x_3 the probability of message erasure along an edge from a check node to an information node. Also, let p be the channel probability of erasure, which is of course also the probability that the prior value at a variable node is an erasure. By the same arguments used to derive eq. (3.3), we now get the following equations relating these quantities:

$$x_1 = 1 - (1 - x_2)R(1 - x_0), \tag{3.8}$$

$$x_2 = px_1, \tag{3.9}$$

$$x_3 = 1 - (1 - x_2)^2\rho(1 - x_0), \text{ and} \tag{3.10}$$

$$x_0 = p\lambda(x_3), \tag{3.11}$$

where the $R(x)$ is the polynomial in which the coefficient of x^i denotes the fraction of check *nodes* of degree i. These coefficients are given by an equation similar to

eq. (3.6), and the polynomial $R(x)$ can be written as

$$R(x) = \frac{\int_0^x \rho(t)dt}{\int_0^1 \rho(t)dt}. \tag{3.12}$$

We eliminate x_1 from the eqs. (3.8) and (3.9) to get

$$x_2 = \frac{p(1 - R(1 - x_0))}{1 - pR(1 - x_0)}. \tag{3.13}$$

Substituting this into eq. (3.10), we get

$$x_3 = 1 - \left[\frac{1-p}{1 - pR(1 - x_0)}\right]^2 \rho(1 - x_0). \tag{3.14}$$

Finally, substituting this into eq. (3.11), we get

$$x_0 = p\lambda\left(1 - \left[\frac{1-p}{1 - pR(1 - x_0)}\right]^2 \rho(1 - x_0)\right). \tag{3.15}$$

This equation contains only one variable, namely x_0, and is necessarily satisfied at a fixed point of density evolution. If we choose our degree polynomials such that the above equation has no fixed points other than $x_0 = 0$, then density evolution necessarily converges to $x_0 = 0$ (which also implies $x_1 = x_2 = x_3 = 0$), and the BEC with probability of erasure p lies within the decoding threshold of the corresponding code ensemble. We can ensure this property by imposing the following constraint on the degree polynomials:

$$p\lambda\left(1 - \left[\frac{1-p}{1 - pR(1 - x)}\right]^2 \rho(1 - x)\right) < x \quad \forall x > 0. \tag{3.16}$$

3.4.2 Capacity Achieving Sequences of Degree Distributions

Having derived a sufficient condition for density evolution to converge to a decoded erasure probability of 0, we now proceed to derive sequences of degree distributions whose rate tends to the capacity of the BEC while satisfying this condition. First, we restrict attention to the "right-regular" case, i.e., $\rho(x) = x^{a-1}$ for some $a \geq 1$, since it turns out that we can achieve capacity with this restriction. In this case, $R(x) = x^a$, and the condition for convergence to zero erasure probability now becomes

$$p\lambda\left(1 - \left[\frac{1-p}{1-p(1-x)^a}\right]^2 (1-x)^{a-1}\right) < x, \quad \forall x > 0 \tag{3.17}$$

Let us denote by $f_p(x)$ the argument to $\lambda(\cdot)$ in the above equation, i.e.,

$$f_p(x) \triangleq 1 - \left[\frac{1-p}{1-p(1-x)^a}\right]^2 (1-x)^{a-1}. \tag{3.18}$$

Ideally, we would like to repeat the procedure we followed in the case of irregular LDPC codes, i.e., we would like to expand $f_p^{-1}(x)$ in a power series and choose $\lambda(x)$ to be a suitably truncated version of this power series. In general, however, $f_p^{-1}(x)$ does not have non-negative power series coefficients. We therefore define the auxiliary function $h_p(x)$ as

$$h_p(x) \triangleq 1 - \left[\frac{1-p}{1-p(1-x)^a}\right]^2 (1-x)^{a}. \tag{3.19}$$

Notice that $1 - h_p(x) = (1-x)(1-f_p(x)) < 1 - f_p(x)$ for $x > 0$, i.e., $h_p(x) > f_p(x)\ \forall x > 0$. Theorem B.2 in Appendix B shows that $g_p(x) \triangleq h_p^{-1}(x)$ has non-negative power series coefficients, when expanded around $x = 0$. Let this expansion be $g_p(x) = \sum_{i=1}^{\infty} g_{p,i} x^i$. We now choose $\lambda(x)$ as a truncated version of this power series, i.e.,

$$\lambda(x) = \frac{1}{p}\left(\sum_{i=1}^{N-1} g_{p,i} x^i + \epsilon x^N\right), \tag{3.20}$$

where N and ϵ are fixed by $0 < \epsilon < g_{p,N}$ and $\sum_{i=1}^{N-1} g_{p,i} + \epsilon = p$. For this choice of $\lambda(x)$, we have

$$p\lambda(x) < g_p(x) = h_p^{-1}(x) < f_p^{-1}(x) \quad \forall x > 0, \tag{3.21}$$

where the last inequality follows because $f_p(x) < h_p(x)\ \forall x > 0$. We can rewrite the above inequality as

$$p\lambda(f_p(x)) < x \quad \forall x > 0, \tag{3.22}$$

which is exactly the condition imposed by eq. (3.17), i.e., the ensemble defined by this degree distribution has a BEC threshold of at least p.

Let us now evaluate the rate of this ensemble, which is given by eq. (3.7). In the right-regular case, this expression simplifies to

$$R = \left(1 + \frac{1}{a\sum_i \lambda_i/i}\right)^{-1}. \tag{3.23}$$

We wish to compute this rate in the limit $a \to \infty$. In order to do so, we need to evaluate $\lim_{a\to\infty} a\sum_i \lambda_i/i$, which is given by

$$\lim_{a\to\infty} a\sum_i \frac{\lambda_i}{i} = \lim_{a\to\infty} \frac{a}{p}\left(\sum_{i=1}^{N-1} \frac{g_{p,i}}{i} + \frac{\epsilon}{N}\right) = \lim_{a\to\infty} \frac{a}{p}\sum_{i=1}^{\infty} \frac{g_{p,i}}{i} - \lim_{a\to\infty} \frac{a}{p}\left(\sum_{i=N}^{\infty} \frac{g_{p,i}}{i} - \frac{\epsilon}{N}\right). \tag{3.24}$$

The second term can be bounded as

$$0 \le \lim_{a\to\infty} \frac{a}{p}\left(\sum_{i=N}^{\infty} \frac{g_{p,i}}{i} - \frac{\epsilon}{N}\right) \le \lim_{a\to\infty} \frac{a}{pN}\sum_{i=N}^{\infty} g_{p,i} \le \frac{1}{p}\lim_{a\to\infty} \frac{a}{N} = 0, \tag{3.25}$$

where the last equality is a property of the function $g_p(x)$ and follows from Theorem B.3 in Appendix B. We now have

$$\lim_{a\to\infty} a\sum_i \frac{\lambda_i}{i} = \lim_{a\to\infty} \frac{a}{p}\sum_{i=1}^{\infty} \frac{g_{p,i}}{i} = \lim_{a\to\infty} \frac{a}{p}\int_0^1 g_p(x)dx. \tag{3.26}$$

Now $\int_0^1 g_p(x)dx + \int_0^1 h_p(1-x)dx = 1$. The easiest way to see this is to note that the first term is the area below the graph of $y = g_p(x)$, while the second term is the area above the graph. (Algebraically, this is a simple integration by parts exercise.) Therefore,

$$a\int_0^1 g_p(x)dx = a\left(1 - \int_0^1 h_p(1-x)dx\right) = a\int_0^1 \left(\frac{1-p}{1-px^a}\right)^2 x^a dx. \tag{3.27}$$

The integrand on the right can be expanded in a power series with non-negative coefficients, with the first nonzero coefficient being that of x^a. Keeping in mind that we are integrating this power series, it is easy to see that

$$\begin{aligned} \frac{a}{a+1}\int_0^1 \left(\frac{1-p}{1-px^a}\right)^2 x^{a-1}dx &\leq \int_0^1 \left(\frac{1-p}{1-px^a}\right)^2 x^a dx \\ &\leq \int_0^1 \left(\frac{1-p}{1-px^a}\right)^2 x^{a-1}dx. \end{aligned} \tag{3.28}$$

Therefore,

$$\begin{aligned} \lim_{a\to\infty} a\int_0^1 \left(\frac{1-p}{1-px^a}\right)^2 x^a dx &= \lim_{a\to\infty} a\int_0^1 \left(\frac{1-p}{1-px^a}\right)^2 x^{a-1}dx \\ &= \int_0^1 \left(\frac{1-p}{1-py}\right)^2 dy. \end{aligned} \tag{3.29}$$

The last integral is easy to evaluate and equals $1-p$. Substituting this value backwards through eqs. (3.27), (3.26), and (3.23), we find that the rate R tends to $1-p$ as $a \to \infty$, which is the capacity of the BEC. Thus we have

Theorem 3.1 (IRA codes achieve capacity on the BEC)

Given a BEC with probability of erasure p, we can find a sequence of degree distributions (λ_i, ρ_i), such that the BEC threshold of the ensemble of (λ_i, ρ_i) IRA codes is at least p for every i, and the rate of the ensemble tends to capacity, i.e., $1-p$, as $i \to \infty$.

3.4.3 Some Numerical Results

We have seen that the condition for the BEC threshold of an ensemble of IRA codes being at least p is $p\lambda(x) < f_p^{-1}(x)\ \forall x > 0$. We later enforced a stronger condition, namely, $p\lambda(x) < h_p^{-1}(x) = g_p(x)\ \forall x > 0$ and derived capacity-achieving degree sequences satisfying this condition. The reason we needed to enforce the stronger condition was that $h_p^{-1}(x) = g_p(x)$ has non-negative power-series coefficients around $x = 0$, while the same cannot be said for $f_p^{-1}(x)$. However, from eq. (3.28) we see that enforcing this stronger condition costs us to the extent of a fraction of $1 - a/(a+1) = 1/(a+1)$ in the rate. This is an extremely slow rate of decay (compare it to the degree-sequences for irregular LDPC codes in [47]), and therefore the resulting codes are not very good.

If, however, $f_p^{-1}(x)$ were to have non-negative power series coefficients, then we could use it to define a degree distribution and we would no longer lose this fraction of $1/(a+1)$. We found through direct numerical computation in all cases that we tried, that enough terms in the beginning of this power series are non-negative to enable us to define $\lambda(x)$ by an equation analogous to eq. (3.20), replacing $g_p(x)$ by $f_p^{-1}(x)$. Of course, the resulting code is not theoretically bound to have a BEC threshold $\geq p$, but again numerical computation showed that the threshold is either equal to or very marginally less than p.

This design turns out to yield very powerful codes, in particular degree distributions whose performance is comparable to the irregular LDPC codes listed in [47] as far as decoding threshold is concerned. The performance of some of these distributions is listed in Table 3.1. The threshold values δ are the same as those in [47] for corresponding values of a (IRA codes with right degree $a+2$ should be compared to irregular LDPC codes with right degree a, so that the decoding complexity is about the same), so as to make comparison easy. The degree distributions listed in [47] were shown to have certain optimality properties w.r.t. the tradeoff between $\delta/(1-R)$ (dis-

a	δ	N	$1-R$	$\delta/(1-R)$
4	0.20000	1	0.333333	0.6000
5	0.23611	3	0.317101	0.7448
6	0.28994	6	0.329412	0.8802
7	0.31551	11	0.336876	0.9366
8	0.32024	16	0.333850	0.9592
9	0.32558	26	0.334074	0.9744
4	0.48090	13	0.502141	0.9577
5	0.49287	28	0.502225	0.9814

Table 3.1: Performance of some IRA code ensembles designed using the procedure described in Section 3.4.3 at rates close to 2/3 and 1/2. δ is the code threshold (maximum allowable value of p), N the degree of $\lambda(x)$, and R the rate of the code.

tance from capacity) and a (decoding complexity), so it is very heartening to note that the codes we have designed are comparable to these.

Let us now briefly discuss the case $a = 1$. In this case, it turns out that $f_p^{-1}(x)$ does indeed have non-negative power-series coefficients. The resulting degree sequences yield codes that are better than conventional RA codes at small rates. An entirely similar exercise can be carried out for the case of non-systematic RA codes with $a = 1$ and the codes resulting in this case are significantly better than conventional RA codes for most rates. However, as we have mentioned earlier, non-systematic RA codes turn out to be useless for higher values of a.

Finally, notice that the condition required of $\lambda(x)$ for density evolution to converge to 0 decoded erasure probability, given by eq. (3.17), is linear in the λ_i's. Our aim is to maximize the rate while satisfying this constraint. However, it is clear from eq. (3.23) for the rate that this is equivalent to maximizing $\sum_i \lambda_i/i$, which is also a linear function of the λ_i's. We thus have a linear programming problem, making it very easy to optimize the rate numerically. (Of course, since in practice we can only enforce a finite number of constraints, we need to pick a finite number of values x for which to impose eq. (3.17). However, the results do not seem to be very sensitive to

the choice of these points.) The degree distributions thus obtained seem to have very similar performance in terms of decoding thresholds to those described in Table 3.1. However, they seem to have far fewer nonzero terms (though the largest degrees are about the same), which is a big advantage for constructing practical finite-length codes.

3.5 IRA Codes on the BIAGN Channel

In this section, we will consider the behavior of IRA codes on the BIAGN channel, which was defined in Example 1.5. Given a noise variance σ, our aim will be to find degree distributions with rates as large as possible whose BIAGN thresholds are at least σ. Unlike the BEC, where density evolution involved updating only a single probability, here we must deal with probability densities. This complicates the analysis, and forces us to resort to approximate design methods.

3.5.1 Gaussian Approximation

Wiberg [53] has shown that the LLR messages passed in iterative decoding on the BIAGN channel can be well approximated by Gaussian random variables. In [9], this approximation was used to design good irregular LDPC codes for the BIAGN channel.

Here, we use this Gaussian approximation to design good IRA codes for the BIAGN channel. Specifically, we approximate the pdf's of the messages from check nodes to variable nodes (both information and parity) as Gaussian at every iteration (under the assumption that the all-zeros codeword is transmitted). The channel evidence (i.e., the quantity l_0 in eq. (1.5)) does in fact have a Gaussian pdf. Therefore, for a given variable node, because of the update rule given by eq. (1.5), if all the incoming messages have Gaussian densities, then so do all the outgoing messages. If

we average over nodes of varying degrees, then the outgoing message densities are mixtures of Gaussians.

A pdf $f(x)$ is called *consistent* [43] if $f(-x) = e^{-x} f(x)\ \forall x$. For a Gaussian density with mean μ and variance σ, this condition reduces to $\sigma^2 = 2\mu$. Thus, a *consistent Gaussian density* with mean μ is given by

$$G_\mu(z) = \frac{1}{\sqrt{4\pi\mu}} e^{-(z-\mu)^2/4\mu}. \tag{3.30}$$

It has been shown in [43] that during density evolution for the sum-product algorithm, all the densities encountered are in fact consistent. Thus if we assume Gaussian message densities, and require consistency, we only need to keep track of the means of the densities. Let us define $\phi(\mu)$ to be the expected value of $\tanh(Z/2)$ for a consistent Gaussian random variable Z with mean μ, i.e.,

$$\phi(\mu) \triangleq E[\tanh(Z/2)] = \int_{-\infty}^{+\infty} G_\mu(z) \tanh\frac{z}{2} dz. \tag{3.31}$$

It is easy to see that $\phi(u)$ is a monotone increasing function of u; we denote its inverse function by $\phi^{-1}(y)$. As we did in the case of the BEC, let us assume that density evolution has reached a fixed point. At this fixed point, let μ_L and μ_R be the means of the (consistent Gaussian) messages from check nodes to information nodes and parity nodes respectively. A message from a degree-i information node to a check node is therefore Gaussian with mean $(i-1)\mu_L + \mu_0$, where μ_0 is the mean of the channel evidence (i.e., the mean of the quantity l_0 in eq. (1.5)). Hence if v_L denotes the message on a randomly selected edge from an information node to a check node, then its pdf is given by $\sum_i \lambda_i G_{(i-1)\mu_L+\mu_0}(z)$. Substituting this in eq. (3.31), we get

$$E[\tanh\frac{v_L}{2}] = \sum_i \lambda_i \phi((i-1)\mu_L + \mu_0). \tag{3.32}$$

Similarly, if v_R denotes the message on a randomly selected edge from a parity node to a check node, we have

$$E[\tanh \frac{v_R}{2}] = \phi(\mu_R + \mu_0). \tag{3.33}$$

Let u_L and u_R denote messages from a check node to an information node and a parity node respectively. Then eqs. (3.32) and (3.33), together with the check node update rule given by eq. (1.6), imply

$$\begin{aligned} E[\tanh \frac{u_L}{2}] &= E[\tanh \frac{v_l}{2}]^{a-1} E[\tanh \frac{v_R}{2}]^2 \\ &= \left[\sum_i \lambda_i \phi((i-1)\mu_L + \mu_0)\right]^{a-1} \phi(\mu_R + \mu_0)^2, \quad \text{and} \qquad (3.34) \\ E[\tanh \frac{u_R}{2}] &= E[\tanh \frac{v_L}{2}]^{a} E[\tanh \frac{v_R}{2}] \\ &= \left[\sum_i \lambda_i \phi((i-1)\mu_L + \mu_0)\right]^{a} \phi(\mu_R + \mu_0). \qquad (3.35) \end{aligned}$$

Since we have assumed that u_L and v_L have consistent Gaussian pdf's, the left-hand sides of these equations are nothing but $\phi(\mu_L)$ and $\phi(\mu_R)$, which gives us the following implicit equations for μ_L and μ_R:

$$\begin{aligned} \phi(\mu_L) &= \left[\sum_i \lambda_i \phi((i-1)\mu_L + \mu_0)\right]^{a-1} \phi(\mu_R + \mu_0)^2, \quad \text{and} \qquad (3.36) \\ \phi(\mu_R) &= \left[\sum_i \lambda_i \phi((i-1)\mu_L + \mu_0)\right]^{a} \phi(\mu_R + \mu_0). \qquad (3.37) \end{aligned}$$

Let us denote $\sum_i \lambda_i \phi((i-1)\mu_L + \mu_0)$ by x. From the definition of $\phi(\cdot)$, we can see that $0 < x < 1$ and $x \to 1 \iff \mu_L \to \infty$. Eq. (3.37) then becomes an implicit equation for μ_R in terms of x, which can be solved numerically given a value of x. Let us denote its solution by f, i.e., $\mu_R = f(x)$. Then, dividing eq. (3.36) by the square

of eq. (3.37) gives

$$\phi(\mu_L) = \frac{\phi(\mu_R)^2}{x^{a+1}} = \frac{\phi(f(x))^2}{x^{a+1}}. \tag{3.38}$$

Therefore we can replace μ_L by $\phi^{-1}(\phi(f(x))^2/x^{a+1})$ into the definition of x, to obtain the following implicit equation for x:

$$x = \sum_i \lambda_i \phi\left(\mu_0 + (i-1)\phi^{-1}\left(\frac{\phi(f(x))^2}{x^{a+1}}\right)\right). \tag{3.39}$$

We would like the BER to go to 0 with the number of iterations, which is equivalent to the condition $\mu_L \to \infty$, or $x \to 1$. Just as in the BEC case, we enforce this condition by not allowing fixed points of (the Gaussian approximation to) density evolution, i.e., by not allowing any solution to eq. (3.39). Specifically, we require

$$F(x) \triangleq \sum_{i=1}^{J} \lambda_i \phi\left(\mu_0 + (i-1)\phi^{-1}\left(\frac{\phi(f(x))^2}{x^{a+1}}\right)\right) > x \ \forall x \in [0,1). \tag{3.40}$$

Notice that just as in the BEC case, the above equation is linear in the λ_i's. We would like to maximize the rate, which is given by eq. (3.23), subject to this constraint. This is equivalent to maximizing $\sum_i \lambda_i/i$, which is also linear in the λ_i's. The problem of finding good degree sequences is thus converted into a linear programming problem under the Gaussian approximation, which is easy to solve numerically.

3.5.2 Numerical Results

We used the linear programming technique described in the previous section to design some good degree sequences for IRA codes. The results are presented in Tables 3.2 (code rate $\approx 1/3$) and 3.3 (code rate $\approx 1/2$). After using the heuristic Gaussian approximation method to design the degree sequences, we used exact density evolution to determine the actual noise threshold. (In every case, the true iterative decoding threshold was better than the one predicted by the Gaussian approximation.)

a	2	3	4
λ_2	0.139025	0.078194	0.054485
λ_3	0.222155	0.128085	0.104315
λ_5		0.160813	
λ_6	0.638820	0.036178	0.126755
λ_{10}			0.229816
λ_{11}			0.016484
λ_{12}		0.108828	
λ_{13}		0.487902	
λ_{14}			
λ_{16}			
λ_{27}			0.450302
λ_{28}			0.017842
rate	0.333364	0.333223	0.333218
σ_{GA}	1.1840	1.2415	1.2615
σ^*	1.1981	1.2607	1.2780
$(\frac{E_b}{N_0})^*$(dB)	0.190	-0.250	-0.371
S.L.(dB)	-0.4953	-0.4958	-0.4958

Table 3.2: Good degree sequences yielding codes of rate approximately 1/3 for the BIAGN channel and with $a = 2, 3, 4$. For each sequence, the Gaussian approximation noise threshold σ_{GA}, the actual sum-product decoding threshold σ^*, and the corresponding $(\frac{E_b}{N_0})^*$ in dB are given. Also listed is the Shannon limit (S.L.).

For example, consider the "$a = 3$" column in Table 3.2. We adjust the Gaussian approximation noise threshold σ_{GA} to be 1.2415 so that the returned optimal sequence has rate 0.333223. Then applying the exact density evolution algorithm to this sequence, we obtain the actual sum-product decoding threshold $\sigma^* = 1.2607$, which corresponds to $E_b/N_0 = -0.250$dB. This should be compared to the Shannon limit for the ensemble of all linear codes of the same rate, which is -0.4958dB. As we increase the parameter a, the ensemble improves. For $a = 4$, the best code we have found has iterative decoding threshold $E_b/N_0 = -0.371$dB, which is only 0.12dB above the Shannon limit.

The above analysis is for BER's. We have already seen in Chapter 2 that $q = 2$ RA codes and $j = 2$ LDPC codes have no threshold in terms of word error probability,

a	8	8
λ_2		0.0577128
λ_3	0.252744	0.117057
λ_7		0.2189922
λ_8		0.0333844
λ_{11}	0.081476	
λ_{12}	0.327162	
λ_{18}		0.2147221
λ_{20}		0.0752259
λ_{46}	0.184589	
λ_{48}	0.154029	
λ_{55}		0.0808676
λ_{58}		0.202038
rate	0.50227	0.497946
σ^*	0.9589	0.972
$(\frac{E_b}{N_0})^*$(dB)	0.344	0.266
S.L.(dB)	0.197	0.178

Table 3.3: Two degree sequences yielding codes of rate $\approx 1/2$ with $a = 8$. For each sequence, the actual sum-product decoding threshold, and the corresponding $(\frac{E_b}{N_0})^*$ in dB are given. Also listed is the Shannon limit.

even with ML decoding. The argument used there easily extends to the case of IRA (or irregular LDPC) codes with a nonzero fraction of degree 2 nodes. Therefore it is desirable to find degree sequences with $\lambda_2 = 0$. In Table 3.3, we compare the noise thresholds of codes with and without $\lambda_2 = 0$.

We chose rate 1/2 because we wanted to compare our results with the best irregular LDPC codes obtained in [43]. Our best IRA code has threshold 0.266 dB, while the best rate 1/2 irregular LDPC code found in [43] has threshold 0.25 dB. These two codes have roughly the same decoding complexity, but unlike LDPC codes, IRA codes have a simple linear encoding algorithm.

We simulated the rate 1/2 code with $\lambda_2 = 0$ from Table 3.3. Figure 3.3 shows the performance of that particular code, with information block lengths 10^3, 10^4, and 10^5. For comparison, we also show the performance of the best known rate 1/2 turbo

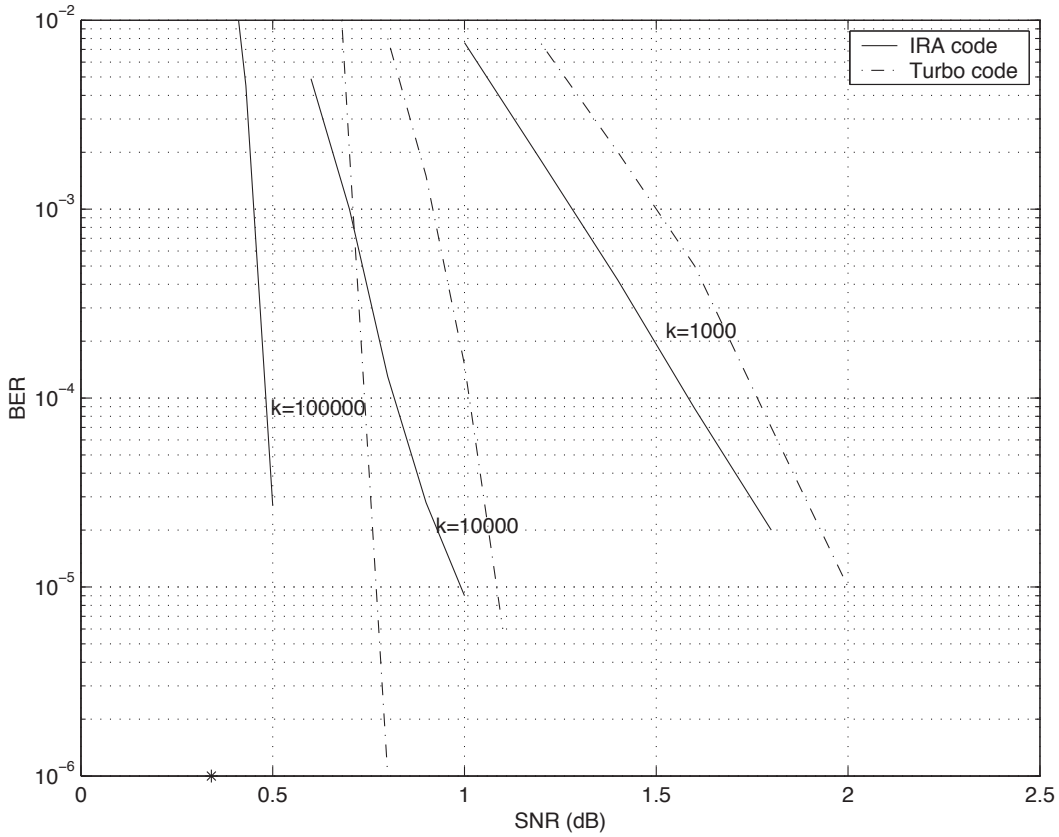

Figure 3.3: Comparison between turbo codes (dashed curves) and IRA codes (solid curves) of lengths $n = 10^3$, 10^4, and 10^5 on the BIAGN channel. All codes are of rate 1/2. The asterisk denotes the threshold of the degree distribution for the IRA code.

code for the same block length.

3.6 Complexity Analysis

We have seen that both irregular LDPC codes and IRA codes achieve capacity on the BEC, and are able to operate extremely close to capacity on the BIAGN channel. Let us now try to study the growth in complexity as we get closer and closer to capacity. We can hope to do this rigorously only on the BEC, since this is the only channel on which these codes have been shown to achieve capacity.

Let us fix a target decoded BER π. For any code (together with a decoding

algorithm) that achieves this BER on a given channel, let ϵ be the fractional difference between its rate R and the capacity C of the channel, i.e., $\epsilon = 1 - R/C$. Let $\chi_D(\epsilon, \pi)$ denote the decoding complexity per decoded bit for an ensemble of codes of rate $R = (1 - \epsilon)C$ and decoded error probability π. Our measure of complexity will be the number of messages passed during decoding.

The sum-product algorithm as we have defined it has decoding complexity proportional to the density of the graph (i.e., the number of edges divided by the length of the code) times the number of decoding iterations. On the BEC, however, this algorithm has the property that once the message on an edge takes on a non-erasure value, it never changes. Using this property, it is possible to reformulate this algorithm in a form so that the decoding complexity is no longer dependent on the number of iterations (and hence the decoded BER π), but is only proportional to the density of the graph. (This modified algorithm can be found in [47].)

In the case of right-regular degree sequences, the density of the graph is asymptotically proportional to the constant check node degree a. It was shown by Shokrollahi [47] that for irregular LDPC codes, ϵ went to 0 exponentially in a, which gives us the following theorem:

Theorem 3.2 (Shokrollahi [47]) *For the ensemble of irregular LDPC codes on the BEC, we can find a sequence of degree distributions such that*

$$\lim_{\pi \to 0} \chi_D(\epsilon, \pi) = O(\log 1/\epsilon). \tag{3.41}$$

We would like to prove a similar result for the ensemble of IRA codes. In Section 3.4.2, we derived a sequence of degree distributions with BEC threshold at least p, and rate going to $1 - p$, i.e., capacity. Let us examine how fast the difference between the rate and the capacity decays. Recall that the rate is given by eq. (3.23), and its computation essentially requires us to estimate the quantity $a \sum_i \lambda_i / i$, which

tends to $(1-p)/p$ as $a \to \infty$. Let us look at the difference between $a\sum_i \lambda_i/i$ and $(1-p)/p$, which is what contributes to the rate loss. One of the loss terms is given by eq. (3.25), and Theorem B.3 in Appendix B shows that this term decays exponentially with a. A term of this type is the dominant loss term in the case of irregular LDPC codes, which is why Theorem 3.2 holds. In the case of IRA codes, however, the dominant loss term is given by eq. (3.28), whose lower bound states that the quantity we are interested in is at least $a/(a+1)$ times its limiting value. The fractional loss here is $1/(a+1)$, which by far dominates over the exponentially decaying loss term given by eq. (3.25). Substituting this loss term into the expression for the rate tells us that ϵ decays as $1/a$, giving us the following theorem:

Theorem 3.3 *For the ensemble of IRA codes on the BEC, we can find a sequence of degree distributions such that*

$$\lim_{\pi \to 0} \chi_D(\epsilon, \pi) = O(1/\epsilon). \tag{3.42}$$

This theorem validates our remarks in Section 3.4.3, where we said that the capacity-achieving degree sequences are not very good in practice. The modified degree sequences we introduced in that section get rid of the dominant loss term of eq. (3.28) and should in principle reduce the rate loss back to $O(\log 1/\epsilon)$. However, we cannot prove that these sequences always exist, or that they have the required threshold values. Our numerical results, however, indicate that they do have these properties, leading us to make the following conjecture:

Conjecture 3.4 *For the ensemble of IRA codes on the BEC, we can find a sequence of degree distributions such that*

$$\lim_{\pi \to 0} \chi_D(\epsilon, \pi) = O(\log 1/\epsilon). \tag{3.43}$$

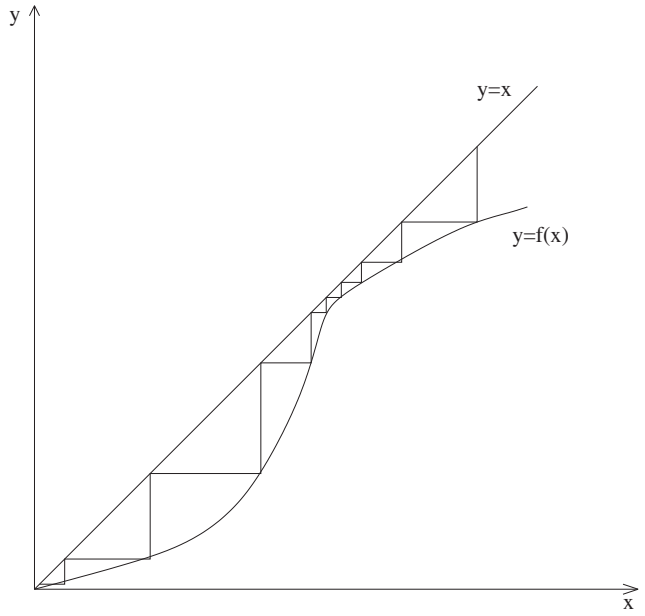

Figure 3.4: Variation of decoded BER with the number of iterations. The function $f(x)$ represents $p\lambda(1-\rho(1-x))$ in the case of irregular LDPC codes and $p\lambda(f_p(x))$ (defined in eq. (3.18)) in the case of IRA codes. The lines in between the two curves represent the variation in the decoded BER with the number of iterations.

In the preceding analysis, we have ignored the number of iterations because they do not play a role in decoding complexity on the BEC. This situation is, however, not indicative of the general case. To get a feel for what happens on more general channel models, let us analyze the number of iterations needed to achieve a decoded BER of π on the BEC.

For a given degree distribution of irregular LDPC codes, the BEC threshold p is given by the smallest value of the channel erasure probability for which eq. (3.3) is not satisfied, i.e., when the curve given by the l.h.s. just touches the curve $y = x$. An analogous statement is true for IRA codes with eq. (3.17) instead of eq. (3.3). The variation of the decoded BER with the number of iterations is shown in Figure 3.4. A point on the $y = x$ line denotes (through both its coordinates) the message erasure probability at the corresponding iteration. At every iteration, the message erasure probability reduces by an amount corresponding to the vertical difference between

the two curves. Thus, one vertical step and one horizontal step together denote one iteration.

Let us consider two cases, one in which the two curves touch at $x = 0$ at the threshold value (i.e., the derivatives at 0 are equal), and the second in which they touch at some other point. Also, suppose that we are operating at a channel erasure probability of $(1 - \epsilon')p$. In the latter case, the vertical distance between the two curves at the point where they touch is proportional to ϵ' for small ϵ'. The number of iterations needed to cross this gap is therefore proportional to $1/\epsilon'$. Near 0, on the other hand, the message erasure probability decays exponentially with the number of iterations. Therefore, the number of iterations needed to achieve a decoded error probability of π grows as $O(1/\epsilon' + \log(1/\pi))$.

On the other hand, if the two curves touch at $x = 0$ at the threshold value p, then at an operating point $(1 - \epsilon')p$, the derivative of the difference at 0 is proportional to ϵ' for small ϵ'. Therefore the message erasure probability decays as $e^{-c\epsilon' l}$ for some c, where l is the number of iterations. Therefore, the number of iterations needed to reach a message erasure probability π grows as $O(\log(1/\pi)1/\epsilon')$. In either case, the number of iterations grows inversely with ϵ'.

Theorem 3.5 *Consider an ensemble of irregular LDPC or IRA codes with BEC threshold p. The number of iterations of the sum-product algorithm needed to achieve a fixed decoded BER of π on a BEC with erasure probability $(1 - \epsilon')p$ grows as $1/\epsilon'$.*

Consider now an ensemble of irregular LDPC codes of rate R with threshold $(1 - \epsilon'')(1 - R)$. It is a simple consequence of Theorem 3.2 that the density of the graph grows as $O(\log 1/\epsilon'')$. Consider the performance of these codes on a BEC with erasure probability $(1 - \epsilon)(1 - R)$, where $\epsilon = \epsilon' + \epsilon''$. Assuming ϵ' and ϵ'' are small, Theorem 3.5 implies that the number of iterations required to achieve a fixed decoded BER grows as $O(1/\epsilon')$. The naive measure of decoding complexity, i.e., graph density

times the number of iterations, therefore grows as $O((1/\epsilon')\log(1/\epsilon''))$. Optimizing for ϵ' and ϵ'' under the constraint $\epsilon' + \epsilon'' = \epsilon$, we see that the naive measure of decoding complexity grows as $O((1/\epsilon)\log(1/\epsilon))$. Based on this evidence, and some other numerical evidence on the BIAGN channel, we advance the following conjecture regarding the complexity of the sum-product algorithm on general BISC's:

Conjecture 3.6 *On any BISC of capacity C, for the ensemble of irregular LDPC or IRA codes, let $\epsilon = 1 - R/C$, R being the rate of the code. Then*

$$\chi_D(\epsilon, \pi) = O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right). \tag{3.44}$$

3.7 Conclusions

We have introduced a class of codes, the IRA codes, that combines many of the favorable attributes of turbo codes and LDPC codes. Like turbo codes (and unlike LDPC codes), they can be encoded in linear time. Like LDPC codes (and unlike turbo codes), they are amenable to an exact Richardson-Urbanke style analysis. In simulated performance they appear to be slightly superior to turbo codes of comparable complexity, and just as good as the best known irregular LDPC codes. We have also presented some analysis on the complexity of iterative decoding close to capacity. In our opinion, the important open problem is to prove (or disprove) that irregular LDPC codes or IRA codes can be decoded reliably in linear time at rates arbitrarily close to channel capacity on channel models other than the BEC. A proof of Conjecture 3.6 can be contemplated only after this problem has been solved.

Chapter 4 A Lower Bound on Iterative Decoding Thresholds for General BISC's

4.1 Introduction

We have seen in Chapter 3 that the technique of density evolution is extremely successful in numerically optimizing ensembles of both irregular LDPC and IRA codes on many BISC's, including the BEC, the BSC and the BIAGN channel. On the BEC, density evolution reduces to an extremely simple one-dimensional evolution, enabling us to design capacity-achieving degree sequences analytically. The general situation is, however, significantly more complicated, and not much is known analytically regarding thresholds of these codes on other channel models. In this chapter, we take a step in this direction by deriving a general lower bound on the threshold of a code ensemble on any BISC, given its BEC threshold.

4.1.1 The Consistency Condition

For any BISC, let Z be a random variable denoting a channel output in log-likelihood form, given that the channel input was 0. For future convenience, let us also define the random variable $Z' = \tanh(Z/2)$. It is shown in [43] that the pdf of Z satisfies the *consistency condition*

$$p_Z(-x) = e^{-x} p_Z(x). \tag{4.1}$$

Moreover, [43] also shows that this condition is preserved during density evolution by both the variable node and check node updates given by eqs. (1.5) and (1.6) (i.e., if all the input random variables are consistent and independent, then the output

random variable is also consistent). Also, it is clear that the consistency condition is preserved under averaging. Therefore any pdf p_X (corresponding to a random variable X) passed at any stage of density evolution satisfies the consistency condition

$$p_X(-x) = e^{-x} p_X(x). \tag{4.2}$$

(We have already encountered this condition in Chapter 3 in the context of the BIAGN channel.) As a simple consequence, the random variable $X' = \tanh(X/2)$ satisfies the consistency condition

$$p_{X'}(-x') = \frac{1 - x'}{1 + x'} p_{X'}(x'). \tag{4.3}$$

4.1.2 The Stability Condition

The Bhattacharya parameter γ of a BISC was defined by eq. (2.1) in the context of the union bound. It is also used in [43], where it is called the *stability function* of the BISC, to derive a necessary condition for the BISC to lie within the decoding threshold of an irregular LDPC code ensemble. In terms of the random variables Z and Z' introduced in Section 4.1.1, γ is given by

$$\gamma = E[e^{-Z/2}] = E\left[\sqrt{\frac{1 - Z'}{1 + Z'}}\right]. \tag{4.4}$$

The first equality can be proved by expanding $E[e^{-Z/2}]$ into an integral over the channel output y, while the second holds because the expressions inside the expectation are identical.

It is shown in [43] that if a BISC with Bhattacharya parameter γ lies within the iterative decoding threshold of the ensemble of (λ, ρ) LDPC codes, then

$$\lambda'(0)\rho'(1) < \frac{1}{\gamma}. \tag{4.5}$$

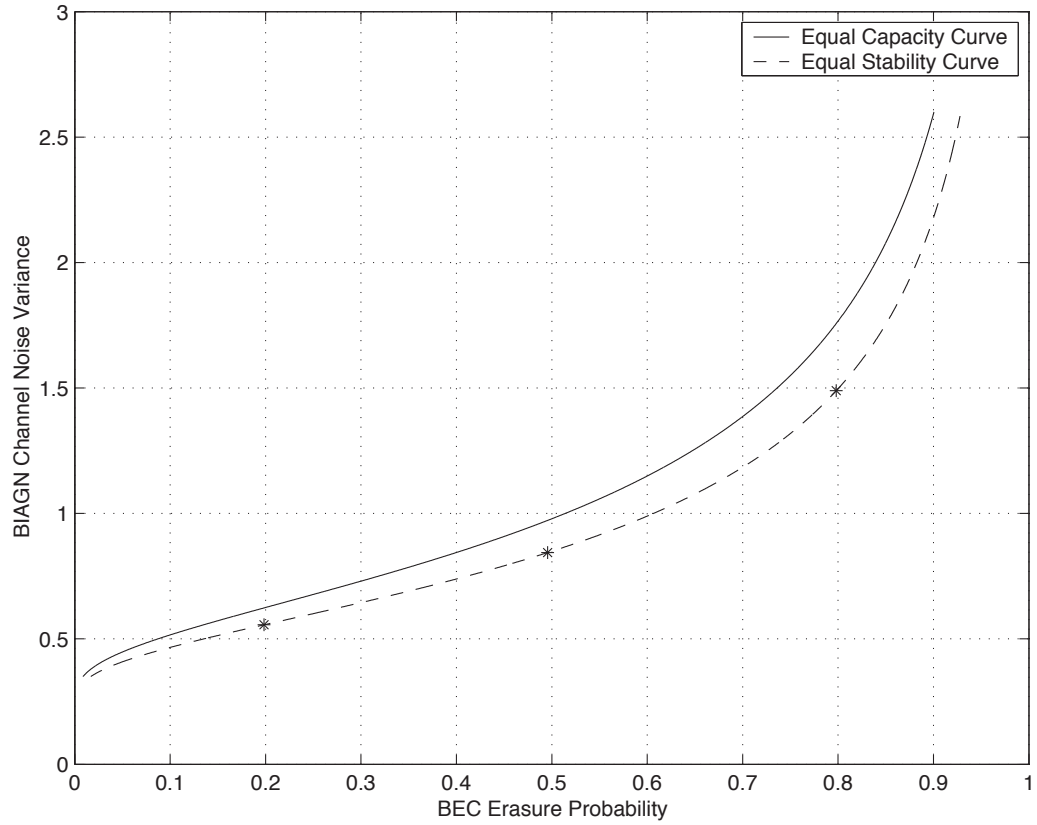

Figure 4.1: BIAGN channel thresholds of codes optimized for the BEC.

The authors call this condition the *stability condition*. Moreover, they also show that if the stability condition is satisfied, and density evolution is initialized with a consistent density having a small enough probability of error (i.e., small enough mass on the negative reals), then the probability of error converges to 0 under density evolution.

4.2 The Main Result

The main result of this chapter is motivated by some observations made in [8]. These observations are illustrated in Figures 4.1 and 4.2, which are adapted from Figures 6.4 and 6.5 in [8]. Figure 4.1 has the BEC (parametrized by the channel erasure proba-

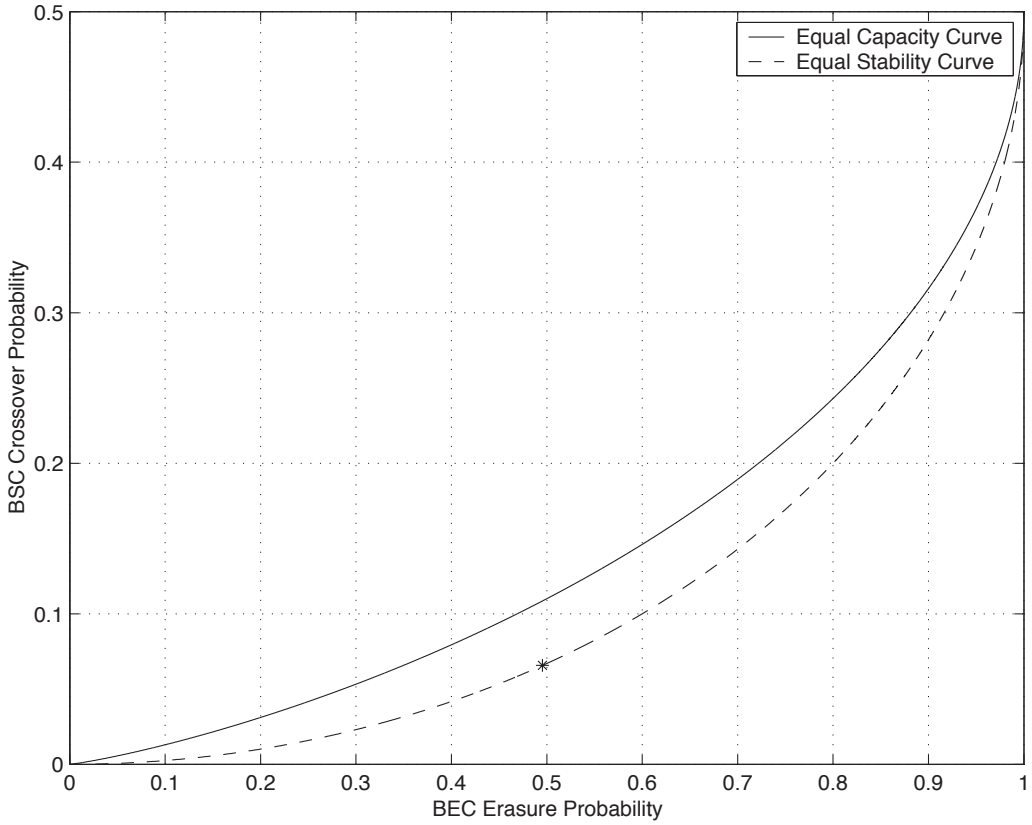

Figure 4.2: BSC thresholds of codes optimized for the BEC.

bility) as the x-axis, and the BIAGN channel (parametrized by the noise variance) as the y-axis. The figure contains two curves, the equal-capacity curve and the equal-stability curve. The equal-capacity curve is defined by the property that the channels given by the two coordinates of any point lying on it have the same capacity. Similarly, the equal-stability curve is defined by the property that the channels given by the two coordinates of any point lying on it have equal Bhattacharya parameters.

Points in this graph can represent ensembles of codes, with their x-coordinate being the BEC (iterative decoding) threshold of the ensemble, and the y-coordinate being the BIAGN channel threshold. The asterisks in the figure, in particular, represent degree distributions of irregular LDPC codes optimized for the BEC, i.e., whose BEC threshold is very close to the capacity of the channel. We can see that all the

degree distributions seem to lie exactly on the equal-stability curve. Figure 4.2 is very similar to Figure 4.1, except that the BIAGN channel is replaced by the BSC. Again, the degree distribution optimized for the BEC is seen to lie on the equal-stability curve. We will prove this observation here by way of a general lower bound on the threshold of a code ensemble on any BISC family in terms of its BEC threshold.

During the course of density evolution, let X_1, X_2, ..., X_{j-1} denote the incoming messages along the first $j-1$ edges adjacent to a variable node of degree j, and let Z denote the channel evidence. By eq. (1.5), the outgoing message X_{out} along the remaining edge is given by $Z + \sum_{i=1}^{j-1} X_i$. Since density evolution assumes that these variables are independent, we have

$$E[e^{-X_{out}/2}] = E\left[e^{-Z/2}\prod_{i=1}^{j-1} e^{-X_i/2}\right] = E[e^{-Z/2}]\prod_{i=1}^{j-1} E[e^{-X_i/2}]. \tag{4.6}$$

Compare this equation to density evolution on the BEC, where the probability of message erasure on the outgoing edge is the product of the corresponding probabilities for the incoming messages and the channel erasure probability (see Section 3.2). The two update equations are identical, with the quantity $E[e^{-Z/2}]$ performing the role of the channel erasure probability, and the quantities $E[e^{-X_i/2}]$ performing the role of the message erasure probabilities.

To complete the comparison, we would like to show a similar equation at the check-node end. But that would be too optimistic since then density evolution on the two channels would be completely equivalent. Instead, we prove an inequality that serves our purpose equally well.

Lemma 4.1 *At a check-node of degree k, if X_1, X_2, ..., X_{k-1} denote the incoming messages along the first $k-1$ edges at some stage of density evolution, and X_{out} the*

outgoing message along the remaining edge, then we have

$$E[1 - e^{-X_{\text{out}}/2}] \geq \prod_{i=1}^{k-1} E[1 - e^{-X_i/2}]. \tag{4.7}$$

Proof:

Define $X'_i = \tanh(X_i/2)$ for $1 \leq i \leq k-1$ and $X'_{\text{out}} = \tanh(X_{\text{out}}/2)$. By Lemma C.1 in Appendix C, and the fact that the X_i's and X_{out} satisfy the consistency condition, the above inequality can be written in the following equivalent form:

$$E\left[1 - \sqrt{1 - X'^2_{\text{out}}}\right] \geq \prod_{i=1}^{k-1} E\left[1 - \sqrt{1 - X'^2_i}\right]. \tag{4.8}$$

The check node update rule given by eq. (1.6) tells us that $X'_{\text{out}} = \prod_{i=1}^{k-1} X'_i$, which implies $X'^2_{\text{out}} = \prod_{i=1}^{k-1} X'^2_i$. Therefore it suffices to prove that given arbitrary independent random variables Y_1, Y_2, ..., Y_{k-1} (to be thought of as X'^2_i's) taking values in the interval $[0, 1]$,

$$E[1 - \sqrt{1 - Y_1 Y_2 \ldots Y_{k-1}}] \geq \prod_{i=1}^{k-1} E[1 - \sqrt{1 - Y_i}]. \tag{4.9}$$

By induction, it is enough to prove the above inequality in the case of two variables, in which case it follows by taking expectations around Lemma C.2 in Appendix C. ■

Eq. (4.7) says that the quantity $E[e^{-X_{\text{out}}/2}]$ is always less than what it would be in the case of the BEC. Together with eq. (4.6), this implies that for any code ensemble (on which density evolution works), if the message erasure probability given by density evolution tends to zero on a BEC with channel erasure probability $E[e^{-Z/2}]$, then so does the quantity $E[e^{-X/2}] = E[\sqrt{1 - X'^2}]$ on the channel under consideration (represented by the distribution of Z). Of course, if the quantity $E[\sqrt{1 - X'^2}]$ tends to 0, then the distribution of X' tends to a delta function at 1, and the decoded probability of error tends to 0 with the number of iterations. Thus, we have proved

the following:

Theorem 4.2 *If a BEC with channel erasure probability p lies within the decoding threshold of an ensemble of codes for which the probability of error can be determined by density evolution (in particular any ensemble of irregular LDPC or IRA codes), then so does any other BISC with the same Bhattacharya parameter, i.e., s.t. $E[e^{-Z/2}] = p$.*

In the case of capacity-achieving degree sequences of irregular LDPC codes for the BEC, we can also prove the converse. Shokrollahi [46] has shown that such a sequence has to be marginally stable, i.e., have $\lambda'(0)\rho'(1)$ tending to $1/p$, where p is the channel erasure probability. Clearly, therefore, any channel within the decoding threshold of this ensemble has to have stability function at most p, else the stability condition will not be satisfied. On the other hand, we have shown that channels with stability function p are within the decoding threshold. Therefore, on any family of channels characterized by a single parameter and having a monotone increasing value of γ, the threshold of this sequence is given by the parameter for which the Bhattacharya parameter of the channel is p, which is exactly the observation that we set out to prove.

Another example for which our bound is tight is the ensemble of cycle codes. Recall that these are nothing but $(2, k)$ LDPC codes for some $k > 2$. Using eq. (3.3), we see that a BEC having erasure probability p lies within the decoding threshold of this ensemble iff $p(1 - (1 - x)^{k-1}) - x < 0 \; \forall x > 0$. Notice (by direct differentiation) that the expression on the l.h.s. is concave, and therefore this inequality holds for all x iff the derivative at 0 is negative. Therefore the given BEC lies within the decoding threshold of the ensemble iff $p(k-1)-1 < 0$, i.e., $p < 1/(k-1)$. Therefore Theorem 4.2 tells us that a BISC with Bhattacharya parameter γ lies within the decoding threshold of the ensemble of $(2, k)$ cycle codes if $\gamma < 1/(k-1)$. On the other hand, the stability

criterion given by eq. (4.5) tells us that this is also a necessary condition. Thus, we see that a BISC with Bhattacharya parameter γ lies within the decoding threshold of this ensemble iff $\gamma < 1/(k-1)$.

Recall from Section 2.5.3 that this is exactly the lower bound on the ML decoding threshold of cycle codes given by the typical set method. Here we have a more powerful result, namely that the iterative decoding threshold is given by the same expression. As we mentioned before, upper bounds on the ML decoding threshold of expurgated cycle code ensembles shown in [11] lead us to believe that in this case, the iterative decoding threshold is the same as the exact ML decoding threshold and is given by $\gamma < 1/(k-1)$.

4.3 Conclusions

As a consequence of what we said in the previous section, we see that any channel with the same stability function as a BEC with parameter p must have a higher capacity, since this is just a way of saying that for a given rate R, the capacity of any one-parameter family of channels is bigger than the threshold C_{BEC} achieved by codes optimized for the BEC. This is in fact proved from first principles in [8]. Unfortunately this difference in capacities is rather significant, as illustrated in Table 4.1, and hence optimizing codes on the BEC for use on other channels is not a very good idea. The main significance of this result is that to the best of our knowledge, it is the first theoretical result about iterative decoding thresholds on a class of general channels.

Another interesting fact is that the threshold achieved on a general BISC by degree sequences optimized for the BEC, while not being close to capacity by current standards, nevertheless beats the so-called *computational cutoff rate* R_0 which was conjectured to be a limit for "practical communication" before the advent of turbo codes and iterative decoding. To see this, note that the computational cutoff rate

Rate	BEC Cap.	BSC		BIAGN Channel	
		C	C_{BEC}	C	C_{BEC}
1/3	0.67	0.174	0.127	-0.495dB	0.851dB
1/2	0.50	0.11	0.067	0.187dB	1.419dB
2/3	0.33	0.061	0.029	1.059dB	2.169dB

Table 4.1: Comparison between capacity C and threshold C_{BEC} achieved by codes optimized for the BEC at different rates, for the BSC (in terms of the crossover probability) and the BIAGN channel (in terms of E_b/N_0).

R_0 of a channel is also defined in terms of its Bhattacharya parameter γ as $R_0 = 1 - \log_2(1+\gamma)$. Because of the concavity of the log function, we can easily see that $R_0 \leq 1 - \gamma$. But the r.h.s. of this equation is the capacity of the BEC with Bhattacharya parameter equal to the channel under consideration, and we have just shown that this channel will lie inside the decoding threshold of a capacity-achieving sequence on this BEC. We thus conclude with the following theorem:

Theorem 4.3 *For any BISC, there exists a degree distribution of irregular LDPC codes with rate greater than the computational cutoff rate R_0 of the channel, such that the BISC lies within its iterative decoding threshold. In other words, rates above R_0 can be achieved in a practical manner on any channel.*

Chapter 5 IRA Codes on Non-Binary Channels

In the last two chapters, we have been concerned with the performance of irregular LDPC codes and IRA codes on BISC's. We have seen that these codes achieve capacity on the BEC and have thresholds very close to capacity on the BIAGN channel. Degree sequences of irregular LDPC codes optimized for various other BISC models, including the BSC and the Laplace channel, can be found in [43] and [8]. The performance of IRA codes on the Rayleigh fading channel, with and without side information, is considered in [24]. The thresholds obtained are very close to capacity in all these cases, and the performance curves are also encouraging.

In this chapter, we will consider the performance of these codes on a couple of channel models that do not fit into the BISC framework. The first is the two-dimensional additive Gaussian noise channel with different constellations, while the second is a very simple multi-access channel called the binary adder channel. The results indicate that turbo-like codes can be adapted to a variety of different channel models.

5.1 The 2-D Gaussian Channel

The BIAGN channel we considered in Chapter 3 is essentially an additive Gaussian noise channel constrained to a BPSK (Binary Phase Shift Keying) constellation. An interesting question to ask is whether IRA codes perform equally well on the Gaussian channel with larger constellations. In many practical situations, the constellation consists of points in a plane, and the additive noise is a two-dimensional circularly symmetric Gaussian random variable characterized by its variance in either dimen-

sion.

Assume that the size of the constellation is 2^M. To use binary codes on such a channel, the elements of the codeword are collected into groups of M, and each group is then mapped to an element of the constellation according to a fixed rule. The resulting scheme is easily represented in a graphical format by adding a set of "modulator" nodes to the graph of the binary code. Each modulator node is connected to the set of codeword components which form its input. If we use IRA codes, then we need to introduce a random permutation between the binary encoder and the modulator in order to avoid too many short cycles in the resulting graph.

In order to extend the sum-product algorithm to this setup, we need to have an update rule for the modulator nodes, which is easily accomplished by means of an a posteriori probability calculation. The communication between nodes in the graph is still in the form of LLR's, but the modulator node update turns out to have complexity proportional to the size of the constellation. This scheme is therefore infeasible for very large constellation sizes; in such cases, more sophisticated techniques like multilevel coding [19] can be used. Many constellations used in practice, however, have small values of M, and do not pose much of a problem.

Another problem that arises while using this scheme is the lack of available techniques to optimize degree distributions over such a channel. For BISC's, Chung [8] observed that a degree distribution optimized for one BISC performs reasonably well on another if the optimization is performed with the additional constraint that it satisfy the stability condition of the latter. Recall that the IRA code we simulated for the BIAGN channel (see Figure 3.3) had no information nodes of degree 2, i.e., has $\lambda'(0) = 0$, and hence satisfies the consistency condition for any BISC. In fact, the same code was simulated in [24] for the Rayleigh fading channel with encouraging results. Here, we show simulation results for IRA codes with no degree 2 information nodes, designed either for the BEC or the BIAGN channel, on 2-D Gaussian channels.

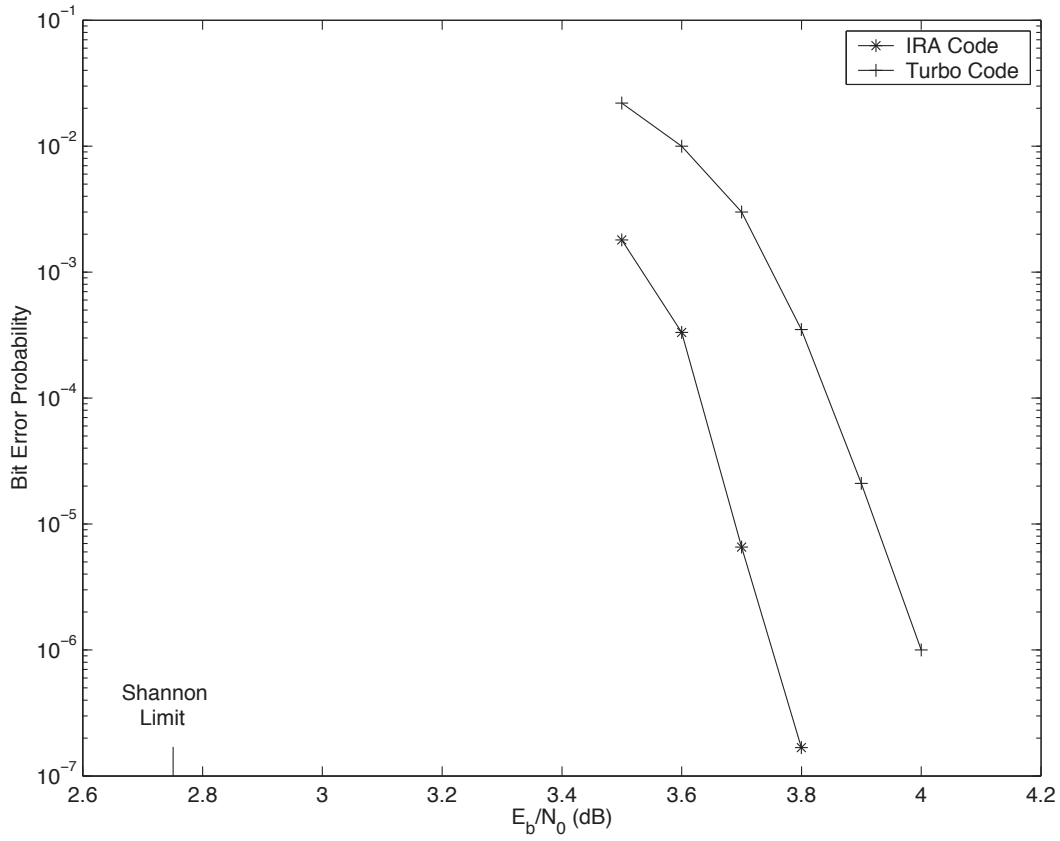

Figure 5.1: Performance of an IRA code vs. a turbo code on the 2-D Gaussian channel with 8-PSK modulation. Both codes have rate 2/3 and 10000 information bits.

Figure 5.1 shows the performance of a rate 2/3 IRA code with 8920 information bits on the 2-D Gaussian channel with 8-PSK modulation. The overall rate of the scheme is 2 bits per channel use. The IRA code in question had $a = 4$, no degree 2 information nodes, and was designed for the BEC. The performance of a turbo code with the same parameters and the capacity of 8-PSK modulation on the 2-D Gaussian channel are also shown for comparison. We can see that the IRA code has a similar advantage relative to the turbo code as it did in the case of the BIAGN channel. The distance from capacity at a given BER is also similar to the BIAGN case for comparable block lengths.

We previously mentioned that the sum-product algorithm has complexity propor-

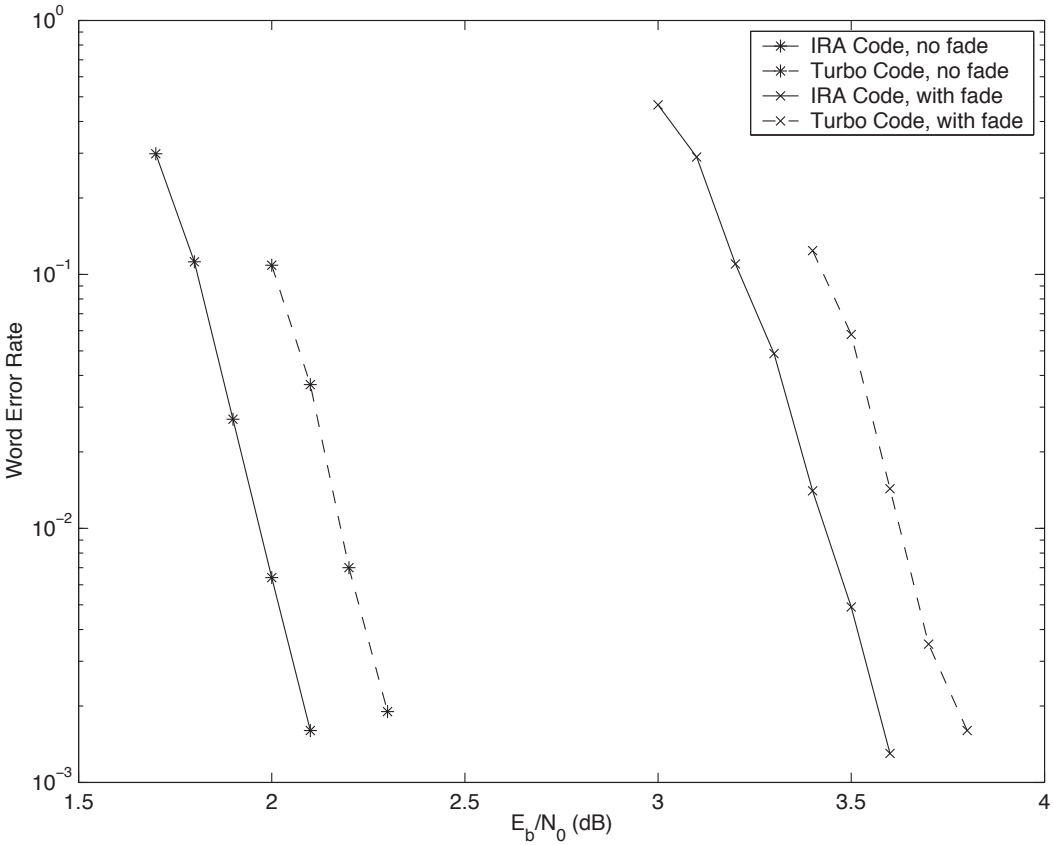

Figure 5.2: Performance of an IRA code vs. a turbo code on the 2-D Gaussian channel with 16-QAM modulation, when bitwise LLR's are marginalized out of the received channel values. Both codes have rate 1/3 and 4096 information bits. Also shown is their performance with an independent Rayleigh fade on each bit.

tional to the size of the constellation, and is hence infeasible for large constellation lengths. One suboptimal option in such a case is to marginalize out the bitwise LLR's from the received channel values, and then use the decoder for the binary code. (Naturally, this involves a loss in capacity.) Figure 5.2 shows the performance using this scheme of a rate 1/3 IRA code having 4096 information bits over a 2-D Gaussian channel with 16-QAM modulation, compared to a turbo code having the same parameters. The IRA code in question had no degree 2 information nodes, had $a = 4$, and was designed for the BIAGN channel. The turbo code had two constituent 8 state convolutional codes. The performance of both codes with an independent Rayleigh

fade on each bit is also shown. (This simulation is one of several done based on parameters taken from one of the 3G wireless protocols.) We see from the figure that the IRA code again maintains a similar advantage over the turbo code as in the case of the BIAGN channel. (The same trend is observed over a range of code rates and constellations, at least for comparable block lengths.)

Further results regarding the performance of IRA codes on 2-D Gaussian channels using multilevel coding can be found in [30]. The techniques described in this section have been used for other channels with non-binary input alphabets, like the 16-ary symmetric channel, in [27, 28].

5.2 The Binary Adder Channel

The binary adder channel (BAC) is a simple example of a multiple-access channel (MAC). It is a two-user channel, with both users having an input alphabet $\{0, 1\}$. The channel output is the real (as opposed to binary) sum of the two inputs. This channel is very closely related to the BEC, and hence permits theoretical analysis.

To see the relation with the BEC, notice that if the channel output is 0 (resp. 2), the receiver knows with perfect certainty that both users transmitted 0 (resp. 1). These two cases correspond to a 0 or a 1 being received on the BEC. On the other hand, if the channel output is 1, the receiver knows that one of the users transmitted a 0 and the other a 1, but cannot decide between the two possibilities. This is analogous to the case in which an erasure is received on the BEC.

The capacity region of a general MAC is well known (see [10, Section 14.3]). In the case of the BAC, the rates R_1 and R_2 of the two users have to satisfy the conditions $R_1 < 1$, $R_2 < 1$ and $R_1 + R_2 < 1.5$. The corner points $(R_1, R_2) = (1, 0.5)$ and $(R_1, R_2) = (0.5, 1)$ of this region are easy to achieve using capacity-achieving BEC codes. Suppose that user 1 transmits uncoded information, i.e., $R_1 = 1$. The

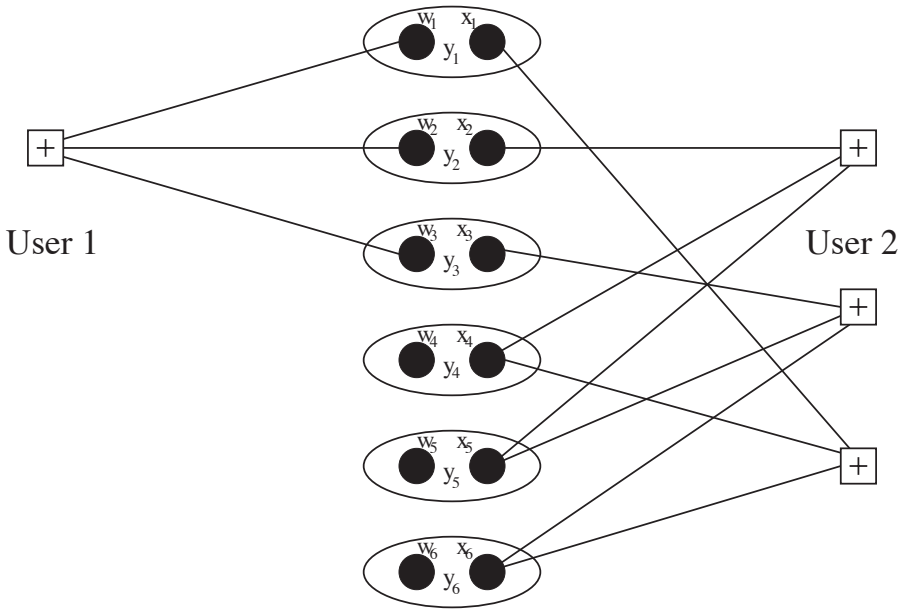

Figure 5.3: Graphical representation of a BAC coding scheme.

receiver can determine what symbol user 2 transmitted if both users transmit the same symbol, i.e., user 2 sees a BEC with erasure probability 0.5. Therefore he can achieve his optimal rate 0.5 using a code that achieves capacity on this channel. Once both the corner points are attained, the remaining points on the boundary of the capacity region can be attained by time-sharing between the two corner points. In this section, we will see another way of attaining an arbitrary point within the capacity region without using time-sharing.

Suppose that each user is encoding his information by means of a binary code represented by a Tanner graph. Then the entire scheme can be represented in a graphical manner as shown in Figure 5.3. The w_i's and the x_i's represent the transmitted values of each of the users, and y_i represents the received value. The parities on the left and right represent the codes used by users 1 and 2 respectively. On a general MAC, the received channel value y_i gives an a priori probability distribution on pairs (w_i, x_i), and thus the node containing these two variables behaves exactly like a modulator

node in the case of the 2-D Gaussian channel in the previous section. Let us call this node a modulator node in this case as well. Therefore, the version of the sum-product algorithm used there can be used for a general MAC as well.

In the specific case of the BAC, the enforced a priori probability distribution consists either of knowing w_i and x_i with certainty, or knowing that w_i is the complement of x_i. As we saw earlier, the latter case corresponds to an erasure on the BEC, and has probability 0.5. Now, if all the incoming messages coming in to the modulator node are erasures and so is the channel prior, then the outgoing message is also an erasure. On the other hand, if either one of these messages is not an erasure, then neither is the outgoing message. (To see this, notice that if the w_i is known, then so is the x_i.) Therefore, as far as the probability of message erasure is concerned, the modulator node update is exactly analogous to the variable node update in the BEC case. The behavior at the check nodes is of course identical to the BEC case.

Therefore, iterative decoding is successful using this code on the BAC, if a BEC with probability of erasure 1/2 lies within the decoding threshold of the overall binary code (after replacing the modulator nodes by variable nodes). Since the capacity of this BEC is 0.5, therefore the overall binary code must have rate at most 0.5. Therefore the sum of the rates of the two constituent codes (of each user) is at most 1.5. (On splitting the parities between the two users, the number of parities remains the same but the number of variable nodes doubles, causing the rate to go up by 1.) This is exactly the capacity of the BAC. To achieve this capacity, all we need to do is start with a capacity-achieving code on a BEC with erasure probability 0.5, and divide its parities among the two users to get an appropriate rate split. This can be done starting with either an irregular LDPC code or an IRA code.

The time-sharing approach derived previously is a special case of this "graph-splitting" approach. In this case, for the first few channel uses, all the parities go to user 1 (achieving one corner point), and after that all the parities are assigned to

user 2 (achieving the other point). However, heuristically speaking, compared to the graph-splitting approach, the time-sharing approach requires a higher block-length to achieve the same probability of error.

The reason the above technique works is that the BAC is "noiseless" in the sense that if you succeed in decoding one of the users, you decode the other automatically. However, it has the property that some variable nodes are completely unprotected either for user 1 or user 2, and therefore the probability of error is always positive if we introduce some additive noise into the channel. [40] gives some coding techniques for communicating effectively on a noisy version of the BAC using IRA codes.

5.3 Conclusions

We have seen that irregular LDPC and IRA codes can be adapted to many different channel models. We have seen this for many different BISC models, as well as 2-D Gaussian channels with different modulation schemes, with and without fading, as well as some simple multiple-access channels. Several other channel models have also been studied in the literature.

Chapter 6 Conclusions

In the preceding chapters, we have considered several problems regarding the analysis and design of graphical code ensembles. In this chapter, we will present a brief summary of the results obtained, together with a discussion of open problems.

The typical set bound was derived in Chapter 2, which is a lower bound on the maximum-likelihood decoding threshold of a code ensemble based on its weight enumerator. We showed that this bound was powerful enough to reproduce Shannon's coding theorem in the case of BISC's, i.e., that the ensemble of random linear codes achieves capacity under maximum-likelihood decoding. We also saw some evidence suggesting that the typical set bound threshold was equal to the actual ML decoding threshold for the ensemble of cycle codes. Though this bound seems to be extremely tight in many cases, we saw that for $q = 3$ RA codes on the BSC and the BIAGN channel, the iterative decoding threshold obtained by density evolution is higher than the threshold obtained by the typical set bound, thus proving that it is not tight in general. In our opinion, the important open problem in this chapter is to determine if there are general conditions under which the typical set bound threshold is the same as the actual ML threshold.

An important contribution of this thesis was the introduction of IRA codes in Chapter 3, which were shown to achieve capacity on the BEC, and have thresholds extremely close to capacity on the BIAGN channel. These codes appear to match irregular LDPC codes in performance, while having an edge over them in terms of encoding complexity. We also analyzed the growth in decoding complexity while approaching capacity on the BEC, and extrapolated these results to make a conjecture for other channels. On the BEC, the provable complexity of approaching capacity (as

given by Theorem 3.3) is seen to be much higher than the observed one (as given by Conjecture 3.4). A proof of the latter would be much appreciated.

In Chapter 4, we derived a general lower bound on the iterative decoding threshold of an ensemble of codes on any BISC based on its BEC threshold. Using this bound, we also showed that it was possible to beat the so-called computational cutoff rate on any BISC using iterative decoding. An important open problem in this regard would be to improve this bound, ideally to get a bound powerful enough to show that irregular LDPC and/or IRA codes achieve capacity on some channel other than the BEC. Only once a proof of this result is available can a resolution of Conjecture 3.6 be attempted.

Finally, we presented some results on the performance of IRA codes on some non-binary channels in Chapter 5. We show IRA codes to be effective on the 2-D Gaussian channel with different input constellations. Particularly interesting is the analysis of the binary adder channel, which seems to be the multiple-access analogue of the BEC, and on which we are able to construct explicit capacity-approaching schemes. Though we have shown some simple techniques for using IRA codes on non-binary channels, more work is required to construct efficient schemes for general discrete channels, especially ones with large input alphabets.

Appendix A Miscellaneous Derivations for Chapter 2

Theorem A.1 *For any BISC, $K(\delta)$ is a convex function in the region where it is finite, i.e., over the interval $(0, \delta_{\max})$, (where $\delta_{\max}$ is as defined in Section 2.3.1).*

Proof:

Consider the BISC described in Section 2.3.1, for which $K(\delta)$ is given by eq. (2.7). Let us define the function $L(\delta_0, \delta_1, \ldots, \delta_K)$ as

$$L(\delta_0, \delta_1, \ldots, \delta_K) \triangleq H(\delta) - \left[p_0 H\left(\frac{\delta_0}{p_0}\right) + \sum_{i=1}^{K} \left(p_i H\left(\frac{\delta_i}{2p_i}\right) + p_{-i} H\left(\frac{\delta_i}{2p_{-i}}\right)\right)\right], \tag{A.1}$$

so that $K(\delta)$, when it is finite, is given by

$$K(\delta) = \inf_{\sum_{i=0}^{K} \delta_i = \delta} L(\delta_0, \delta_1, \ldots, \delta_K), \tag{A.2}$$

where the constraints $0 \leq \delta_0 \leq p_0$ and $0 \leq \delta_i \leq min(2p_i, 2p_{-i})$ are implicitly assumed.

We wish to prove that for any $\delta^{(1)}$ and $\delta^{(2)}$, and any λ between 0 and 1,

$$K(\delta^{(0)}) \leq \lambda K(\delta^{(1)}) + (1-\lambda) K(\delta^{(2)}), \tag{A.3}$$

where $\delta^{(0)} = \lambda\delta^{(1)} + (1-\lambda)\delta^{(2)}$. Let $\delta_i^{(1)}$ and $\delta_i^{(2)}$, $0 \leq i \leq K$ be the optimizing δ_i's in eq. (A.2) at $\delta = \delta^{(1)}$ and $\delta = \delta^{(2)}$ respectively. Then eq. (A.3) holds iff there exists a valid breakup $\delta^{(0)} = \sum_{i=0}^{K} \delta_i^{(0)}$ of $\delta^{(0)}$ satisfying

$$L(\delta_0^{(0)}, \delta_1^{(0)}, \ldots, \delta_K^{(0)}) \leq \lambda L(\delta_0^{(1)}, \delta_1^{(1)}, \ldots, \delta_K^{(1)}) + (1-\lambda) L(\delta_0^{(2)}, \delta_1^{(2)}, \ldots, \delta_K^{(2)}). \tag{A.4}$$

(This follows directly from eq. (A.2).) We claim that eq. (A.4) is satisfied for the choice $\delta_i^{(0)} = \lambda\delta_i^{(1)} + (1-\lambda)\delta_i^{(2)}$. (It is easy to check that this choice satisfies the necessary constraints.) This is equivalent to showing the convexity of the $K+1$-dimensional function $L(\delta_0, \delta_1, \ldots, \delta_K)$.

To this end, consider the random variable X taking integer values between $-K$ and K. Let X take the value i with probability p_i. Let Y be a binary random variable, i.e., taking values 0 and 1. The joint distribution $\Pr(X = x, Y = y)$ is determined by the conditional distribution $\Pr(Y = y|X = x)$. Let this conditional distribution be defined by $\Pr(Y = 0|X = 0) = \frac{\delta_0}{p_0}$ and $\Pr(Y = 0|X = i) = \frac{\delta_{|i|}}{2p_i}$ for $i \neq 0$. Clearly, $\Pr(Y = 0) = \sum_{i=0}^{K} \delta_i = \delta$. Therefore, from eq. (A.1),

$$L(\delta_0, \delta_1, \ldots, \delta_K) = H(Y) - H(Y|X) = I(X;Y), \tag{A.5}$$

the mutual information between X and Y. It is a well-known fact that this quantity is a convex function of the vector of transition probabilities $\Pr(Y = y|X = x)$. (For a proof, see [10, Theorem 2.7.4].) Since the δ_i's are linear functions of this vector, the function $L(\delta_0, \delta_1, \ldots, \delta_K)$ is also convex, thus completing the proof. ■

Appendix B Miscellaneous Derivations for Chapter 3

Lemma B.1 *Let $x = f(y)$ be the solution to the equation $y = 1 - (1-x)^{a-1}$, where both x and y lie in $[0,1]$, and a is an integer greater than 1. Then $f(y)$ has a power series expansion around $y = 0$ with non-negative coefficients.*

Proof:

$$y = 1-(1-x)^{a-1} \iff (1-x)^{a-1} = 1-y \iff x = 1-(1-y)^{1/(a-1)} = f(y). \quad \text{(B.1)}$$

Let $\alpha = 1/(a-1)$. Then $0 < \alpha \leq 1$. We can expand the above expression for $f(y)$ by the binomial theorem as

$$f(y) = 1 - \sum_{i=0}^{\infty} (-1)^i \binom{\alpha}{i} y^i = \sum_{i=1}^{\infty} (-1)^{i+1} \binom{\alpha}{i} y^i. \quad \text{(B.2)}$$

However, for $0 < \alpha \leq 1$, it is easily seen from its definition that $\binom{\alpha}{i}$ is positive for odd i and negative for even i (except for $i = 0$), which together with the $(-1)^i$ factor ensures that each coefficient in the above power series expansion is positive. ■

Theorem B.2 *Let $x = f(y)$ be the solution to the equation*

$$y = 1 - \left[\frac{1-p}{1-p(1-x)^a}\right]^2 (1-x)^a, \quad \text{(B.3)}$$

where both x and y lie in $[0,1]$, and a is a positive integer. Then $f(y)$ has a power series expansion around $y = 0$ with non-negative coefficients.

Proof:

We introduce the intermediate variable $z = 1 - (1-x)^a$. If $x = g(z)$ is the solution to this equation, then by Lemma B.1, $g(z)$ has a power series expansion with non-negative coefficients around $z = 0$. Eq. (B.3) can now be rewritten as

$$y = 1 - \left[\frac{1-p}{1-p(1-z)}\right]^2 (1-z). \tag{B.4}$$

Let $z = h(y)$ denote the solution to this equation. Since $f(y) = x = g(z) = g(h(y))$, and we know that $g(z)$ has a power series expansion with non-negative coefficients around $z = 0$, it suffices to show that $h(y)$ has a power series expansion with non-negative coefficients around $y = 0$.

Now, multiplying both sides of eq. (B.4) by $(1-p(1-z))^2$ and bringing all the terms to one side, we get the following quadratic equation for z:

$$p^2(1-y)z^2 + (1-p)(1+p-2py)z - (1-p)^2 y = 0. \tag{B.5}$$

The non-negative root of this equation is given by

$$\begin{aligned} z &= \frac{-(1-p)(1+p-2py) + \sqrt{(1-p)^2(1+p-2py)^2 + 4p^2(1-y)(1-p)^2 y}}{2p^2(1-y)} \\ &= \frac{(1-p)}{2p^2(1-y)}\left[-(1+p-2py) + \sqrt{(1+p)^2 - 4py}\right] \\ &= \frac{1-p^2}{2p^2}\left[-1 + \frac{2p}{1+p}y + \sqrt{1 - \frac{4p}{(1+p)^2}y}\right](1 + y + y^2 + \ldots). \end{aligned} \tag{B.6}$$

The term inside the square-root can be expanded into a power series using the binomial theorem. Let us define the function $c(y)$ together with its power series expansion as

$$c(y) \triangleq \sum_i c_i y^i \triangleq -1 + \frac{2p}{1+p}y + \sqrt{1 - \frac{4p}{(1+p)^2}y}. \tag{B.7}$$

Expanding the r.h.s. and comparing terms, we see that $c_0 = -1 + 1 = 0$, while c_1 is

given by

$$c_1 = \frac{2p}{1+p} - \frac{2p}{(1+p)^2} = \frac{2p^2}{(1+p)^2} \geq 0. \tag{B.8}$$

For $i > 1$, we have

$$c_i = (-1)^i \binom{1/2}{i} \left(\frac{4p}{(1+p)^2} \right)^i \leq 0, \tag{B.9}$$

because the binomial coefficient $\binom{1/2}{i}$ is positive for odd i and negative for even i (except $i = 0$). Now, if $h(y)$ has a power series expansion $h(y) = \sum_i h_i y^i$, then eq. (B.6) tells us that h_i is given by

$$h_i = \frac{1-p^2}{2p^2} \sum_{j=0}^{i} c_j. \tag{B.10}$$

Therefore $h_0 = 0$ and $h_1 = c_1(1-p^2)/(2p^2) = (1-p^2)/(1+p)^2 \geq 0$. Since $c_i \leq 0$ for $i \geq 2$, the h_i's form a non-increasing sequence for $i \geq 1$. Also $\lim_{i\to\infty} h_i = \frac{1-p^2}{2p^2} \sum_{i=0}^{\infty} c_i = \frac{1-p^2}{2p^2} c(1) = 0$, since $c(1) = 0$. This shows that the h_i's are all non-negative, and completes the proof. ■

Theorem B.3 *Let $h_p(x)$ and $g_p(x)$ be defined as in Section 3.4.2, i.e.,*

$$h_p(x) \triangleq 1 - \left[\frac{1-p}{1-p(1-x)^a} \right]^2 (1-x)^a, \tag{B.11}$$

and $g_p(x) \triangleq \sum_{i=1}^{\infty} g_{p,i} x^i \triangleq h_p^{-1}(x)$. Let N be the smallest integer such that $\sum_{i=1}^{N} g_{p,i} \geq p$. Then, for fixed $p > 0$, and any $c < 1/(1-p)$, there exists a constant k such that $N > kc^a$. In particular, N grows exponentially in a.

Proof: We begin by bounding $g_p(x)$ from below as follows:

$$g_p(x) = \sum_{i=1}^{\infty} g_{p,i} x^i \geq \sum_{i=1}^{N} g_{p,i} x^i \geq \left(\sum_{i=1}^{N} g_{p,i} \right) x^N \geq p x^N. \tag{B.12}$$

This gives us the following lower bound on N:

$$N \geq \frac{\ln(1/g_p(x)) - \ln(1/p)}{\ln(1/x)}. \tag{B.13}$$

Substituting $h_p(x)$ for x in this equation, we get

$$N \geq \frac{\ln(1/x) - \ln(1/p)}{\ln(1/h_p(x))}. \tag{B.14}$$

It is easy to see from eq. (B.11) that $h_p(x) \geq 1 - (1-x)^a$. Therefore $\ln(h_p(x)) \geq \ln(1-(1-x)^a) \geq -(1-x)^a$, i.e., $\ln(1/h_p(x)) \leq (1-x)^a$. Substituting this in eq. (B.14) gives

$$N \geq \frac{\ln(1/x) - \ln(1/p)}{(1-x)^a}. \tag{B.15}$$

Therefore, as long as the numerator of the r.h.s. is positive, i.e., $x < p$, N grows faster than a constant times $(1/(1-x))^a$. This is exactly the statement of the theorem. ■

Appendix C Miscellaneous Derivations for Chapter 4

Lemma C.1 *Given random variables X and $X' = \tanh(X/2)$ satisfying the consistency conditions given by eqs. (4.2) and (4.3), respectively, we have*

$$E\left[e^{-X/2}\right] = E\left[\sqrt{\frac{1-X'}{1+X'}}\right] = E\left[\sqrt{1-X'^2}\right]. \tag{C.1}$$

Proof:

The first equality is true simply because the random variables on both sides are identical. The second needs an application of eq. (4.3). Firstly, let us use eq. (4.3) to find the pdf of $|X'|$.

$$p_{|X'|}(x) = p_{X'}(x) + p_{X'}(-x) = \left(1 + \frac{1-x}{1+x}\right) p_{X'}(x) = \frac{2}{1+x} p_{X'}(x) \tag{C.2}$$

Therefore, for $0 \leq x \leq 1$, we have

$$p_{X'}(x) = \frac{1+x}{2} p_{|X'|}(x), \quad \text{and} \tag{C.3}$$

$$p_{X'}(-x) = \frac{1-x}{2} p_{|X'|}(x). \tag{C.4}$$

Now,

$$\begin{aligned} E\left[\sqrt{\frac{1-X'}{1+X'}}\right] &= \int_{-1}^{1} \sqrt{\frac{1-x}{1+x}} p_{X'}(x) dx \\ &= \int_{0}^{1} \sqrt{\frac{1-x}{1+x}} \frac{1+x}{2} p_{|X'|}(x) dx + \int_{0}^{1} \sqrt{\frac{1+x}{1-x}} \frac{1-x}{2} p_{|X'|}(x) dx \end{aligned}$$

$$= \int_0^1 \sqrt{1-x^2} p_{|X'|}(x) dx$$
$$= E\left[\sqrt{1-X'^2}\right], \tag{C.5}$$

which completes the proof. ■

Lemma C.2 *For any* $0 \leq y_1, y_2 \leq 1$*, we have*

$$1 - \sqrt{1-y_1 y_2} \geq (1-\sqrt{1-y_1})(1-\sqrt{1-y_2}). \tag{C.6}$$

Proof:

$$\begin{aligned}
& 1 - \sqrt{1-y_1y_2} \geq (1-\sqrt{1-y_1})(1-\sqrt{1-y_2}) \\
\iff \quad & \sqrt{1-y_1y_2} \leq \sqrt{1-y_1} + \sqrt{1-y_2} - \sqrt{(1-y_1)(1-y_2)} \\
\iff \quad & 1 - y_1y_2 \leq (1-y_1) + (1-y_2) + (1-y_1-y_2+y_1y_2) \\
& \qquad - 2(1-y_1)\sqrt{1-y_2} - 2(1-y_2)\sqrt{1-y_1} \\
& \qquad + 2\sqrt{(1-y_1)(1-y_2)} \\
\iff \quad & 0 \leq (1-y_1)(1-y_2) - (1-y_1)\sqrt{1-y_2} \\
& \qquad - (1-y_2)\sqrt{1-y_1} + \sqrt{(1-y_1)(1-y_2)} \\
\iff \quad & 0 \leq \sqrt{(1-y_1)(1-y_2)} - \sqrt{1-y_1} - \sqrt{1-y_2} + 1 \\
\iff \quad & 0 \leq (1-\sqrt{1-y_1})(1-\sqrt{1-y_2}),
\end{aligned}$$

which is true. This proves the lemma. ■

Bibliography

[1] S. Aji, "Graphical models and iterative decoding," Ph.D. Thesis, California Institute of Technology, Pasadena, 2000.

[2] S. Aji, G. Horn and R. J. McEliece, "Iterative decoding on graphs with a single cycle," *Proc. ISIT 1998* (Ulm, Germany, 1998).

[3] S. Aji, H. Jin, A. Khandekar, D. J. C. MacKay and R. J. McEliece, "BSC thresholds for code ensembles based on 'typical pairs' decoding," *Proc. IMA Workshop on Codes and Graphs*, (August 1999), pp. 195–210.

[4] S. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Info. Theory*, March 2000, vol. 32, no. 1, pp. 325–343.

[5] S. Aji and R. J. McEliece, "The generalized distributive law and free energy minimization," *Proc. 39th Allerton Conf. on Communication, Control and Computing* (Allerton, Illinois, Oct. 2001), pp. 672–681.

[6] L. Bahl, J. Cocke, F. Jelenik and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Info. Theory*, March 1974, vol. 20, pp. 284–287.

[7] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," *Proc. ICC 1993* (Geneva, Switzerland), pp. 1064–1070.

[8] S.-Y. Chung, "On the construction of some capacity-approaching coding schemes," Ph.D. Thesis, Dept. of Elec. Engg. and Comp. Sc., MIT, 2000.

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

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

[11] L. Decreusefond and G. Zémor, "On the error-correcting capabilities of cycle codes on graphs," *Combinatorics, Probability and Computing*, vol. 6, 1997, pp. 1–35.

[12] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative turbo decoder analysis based on Gaussian density evolution," submitted to *IEEE J. Selected Areas in Comm.*

[13] D. Divsalar, "A simple tight bound on error probability of block codes with application to turbo codes," *JPL TMP Progress Report42-239*, Nov. 1999, pp. 1–35.

[14] D. Divsalar, S. Dolinar, H. Jin and R. J. McEliece, "AWGN coding theorems from ensemble weight enumerators," Proc. ISIT 2000, p. 458.

[15] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for 'turbo-like' codes," *Proc. 36th Allerton Conf. on Communication, Control, and Computing* (Allerton, Illinois, Sept. 1998), pp. 201–210.

[16] G. D. Forney, Jr., "Codes on graphs: normal realizations," *IEEE Trans. Info. Theory*, Feb. 2001, vol. 47, pp. 520–548.

[17] R. Gallager, *Low-Density Parity-Check Codes.* Cambridge, Massachusetts: MIT Press, 1963.

[18] R. Gallager, *Information theory and reliable communication.* New York: McGraw Hill, 1968.

[19] H. Imai and S. Hirakawa, "A new multilevel coding method using error correcting codes," *IEEE Trans. Info. Theory*, May 1977, vol. 23, pp. 371–377.

[20] H. Jin, "Analysis and design of turbo-like codes," Ph.D. Thesis, California Institute of Technology, 2001.

[21] H. Jin, A. Khandekar and R. J. McEliece, "Irregular repeat-accumulate codes," *Proc. 2nd International Symposium on Turbo Codes* (Brest, France, September 2000), pp. 1–8.

[22] H. Jin and R. J. McEliece, "AWGN coding theorems for serial turbo codes," *Proc. 37th Allerton Conf. on Communication, Computation and Control* (Allerton, Illinois, Sept. 1999), pp. 893–894.

[23] H. Jin and R. J. McEliece, "Typical pairs decoding on the AWGN channel," *Proc. 2000 International Symp. on Info. Theory and its Applications* (Hawaii, Nov. 2000), pp. 180–183.

[24] H. Jin and R. J. McEliece, "Performance of IRA codes on Rayleigh fading channels," *Proc. CISS 2001* (Baltimore, MD, Jan. 2001).

[25] H. Jin and R. J. McEliece, "Coding theorems for turbo code ensembles," *IEEE Trans. Info. Theory*, June 2002, vol. 48, pp. 1451–1461.

[26] A. Khandekar and R. J. McEliece, "On the complexity of reliable communication on the erasure channel," *Proc. ISIT 2001* (Washington D.C.), p. 1.

[27] A. Khandekar and R. J. McEliece, "Are turbolike codes effective on nonstandard channels?" *Proc. 2001 International Symp. on Communication Theory and its Applications* (Ambleside, U.K., July 2001), pp. 293–298.

[28] A. Khandekar and R. J. McEliece, "Are turbolike codes effective on nonstandard channels," *IEEE Info. Theory Society Newsletter*, Dec. 2001, vol. 51, pp. 1–8.

[29] A. Khandekar and R. J. McEliece, "A lower bound on the iterative decoding threshold of irregular LDPC code ensembles," *Proc. CISS 2002* (Princeton, NJ, March 2002).

[30] A. Khandekar and R. Palanki, "Irregular repeat-accumulate codes for non-binary modulation schemes," *Proc. ISIT 2002* (Lausanne, Switzerland, July 2002).

[31] F. R. Kschischang, B. J. Frey and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Info. Theory*, Feb. 2002, vol. 47, pp. 498–519.

[32] S. Litsyn and V. Shevelev, "On ensembles of low-density parity-check codes: asymptotic distance distributions," *IEEE Trans. Info. Theory*, April 2002, vol. 48, pp. 887–908.

[33] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," *Proc. 29th ACM Symp. on the Theory of Computing* (1997), pp. 150–159.

[34] M. Luby, M. Mitzenmacher, A. Shokrollahi and D. Spielman, "Analysis of low-density codes and improved designs using irregular graphs," *Proc. 30th ACM Symp. on the Theory of Computing* (1998), pp. 249–258.

[35] M. Luby, M. Mitzenmacher, A. Shokrollahi and D. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Info. Theory*, Feb. 2001, vol. 47, pp. 569–584.

[36] M. Luby, M. Mitzenmacher, A. Shokrollahi and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Info. Theory*, Feb. 2001, vol. 47, pp. 585–598.

[37] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Info. Theory*, March 1999, vol. 45, pp. 399–431.

[38] R. J. McEliece, *The theory of information and coding*. Cambridge, U.K.: Cambridge University Press, 2002.

[39] R. J. McEliece, D. J. C. MacKay and J.-F. Cheng, "Turbo decoding as an instance of Pearl's 'Belief Propagation' algorithm," *IEEE J. Selected Areas in Comm.*, Feb. 1998, vol. 16, no. 2, pp. 140–152.

[40] R. Palanki, A. Khandekar and R. J. McEliece, "Graph-based codes for synchronous multiple access channels," *Proc. 39th Allerton Conf. on Communication, Control and Computing* (Allerton, Illinois, Oct. 2001), pp. 1263–1271.

[41] J. Pearl, *Probabilistic reasoning in intelligent systems*. San Mateo, California: Morgan Kauffman, 1988.

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

[43] T. J. Richardson, A. Shokrollahi and R. Urbanke, "Design of provably good low-density parity-check codes," *IEEE Trans. Info. Theory*, Feb. 2001, vol. 47, pp. 619–637.

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

[45] C. E. Shannon, *The Mathematical Theory of Communication* . Urbana, Illinois: University of Illinois Press, 1963, 1998.

[46] M. A. Shokrollahi, "Capacity-achieving sequences," *Proc. of the 1999 IMA workshop on Codes, Systems and Graphical Models*, pp. 153–166.

[47] M. A. Shokrollahi, "New sequences of linear time erasure codes approaching channel capacity," *Proc. 1999 AAECC* (Honolulu, Hawaii, November 1999) pp. 65–76.

[48] M. A. Shokrollahi, "Capacity-achieving sequences on the erasure channel," preprint, available at `http://shokrollahi.com/amin/pub.html#ldpc`, 2000.

[49] M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Info. Theory*, Sept. 1981, vol. 27, pp. 533–547.

[50] A. J. Viterbi and A. M. Viterbi, "Improved union bound on linear codes for the binary-input AWGN channel, with applications to turbo decoding," *Proc. Winter 1998 Info. Theory Workshop* (San Diego, California, Feb. 1998), pp. 72.

[51] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEE Trans. Info. Theory*, Feb. 2001, vol. 47, pp. 736–744.

[52] Y. Weiss and W. T. Freeman, "Correctness of belief-propagation in Gaussian graphical models of arbitrary topology," in *Advances in Neural Information Processing Systems 12*, eds. S. Solla, T. K. Leen and K. R. Muller, 2000.

[53] N. Wiberg, *Codes and decoding on general graphs*. Linköping Studies in Science and Technology, Dissertation no. 440. Linköping, Sweden, 1996.

[54] J. S. Yedidia, W. T. Freeman and Y. Weiss, "Generalized belief propagation," in *Advances in Neural Information Processing Systems 13*, eds. T. K. Leen, T. G. Diettrich and V. Tresp, 2000.

[55] J. S. Yedidia, W. T. Freeman and Y. Weiss, "Bethe free energy, Kikuchi approximations, and belief propagation algorithms," available at `www.merl.com/papers/TR2001-16`.