Iterative Algorithms For Lossy Source Coding

by

Venkat Chandar

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 26, 2006

Author ....
Department of Electrical Engineering and Computer Science
May 26, 2006

Certified by.... ....
Gregory Wornell
Professor of Electrical Engineering
Thesis Supervisor

Certified by.... ....
Emin Martinian
Research Scientist, Mitsubishi Electric Research Labs
Thesis Supervisor

Accepted by......... ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Iterative Algorithms For Lossy Source Coding

by

Venkat Chandar

Submitted to the
Department of Electrical Engineering and Computer Science

May 26, 2006

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis explores the problems of lossy source coding and information embedding. For lossy source coding, we analyze low density parity check (LDPC) codes and low density generator matrix (LDGM) codes for quantization under a Hamming distortion. We prove that LDPC codes can achieve the rate-distortion function. We also show that the variable node degree of any LDGM code must become unbounded for these codes to come arbitrarily close to the rate-distortion bound.

For information embedding, we introduce the double-erasure information embedding channel model. We develop capacity-achieving codes for the double-erasure channel model. Furthermore, we show that our codes can be efficiently encoded and decoded using belief propagation techniques. We also discuss a generalization of the double-erasure model which shows that the double-erasure model is closely related to other models considered in the literature.

Thesis Supervisor: Gregory Wornell
Title: Professor of Electrical Engineering

Thesis Supervisor: Emin Martinian
Title: Research Scientist, Mitsubishi Electric Research Labs

Acknowledgments

First, I would like to thank my advisors, Greg Wornell and Emin Martinian. Greg first introduced me to lossy source coding and information embedding, and both of them have taught me so much about how to do research over the last year. Our weekly meetings helped me flesh out many of the ideas in this thesis. Also, their detailed feedback on earlier versions of this thesis greatly improved the clarity of this document. I look forward to working with both of them in the future.

I would also like to thank the other members of the Signals, Information, and Algorithms Laboratory for providing a stimulating work environment. The weekly group meetings were very interesting. Special thanks goes to Charles Swannack and Pascal Vontobel. Charles was a great office mate over the last year, and I could ask him about research problems any time. Pascal brought a fresh perspective and gave me many valuable ideas during the last stages of preparing this thesis. I am also thankful to Tricia for all of her help during my time here. Thanks also to our system administrator, Giovanni Aliberti.

Finally, I would like to thank my wonderful parents and my brother Vijay. Their constant support and encouragement have been instrumental in helping me achieve my goals.

Contents

List of Figures

Chapter 1

Introduction

This thesis explores some aspects of the complexity inherent in source coding. Source coding is one of the fundamental problems in digital communications. In information theory, we model a source of information as a random variable, or more generally as a stochastic process. We would like to find a compact representation of this process, for example using only binary digits (0 and 1). Finding a compact representation is useful because this allows fewer bits to be transmitted when the source is sent to a receiver.

The standard version of source coding is lossless source coding. In this framework, an encoder maps the source into bits, or more generally into some alphabet. The receiver sees the encoding, and needs to reconstruct the source exactly. One of the classic results of information theory is the source coding theorem [26], which characterizes the best possible compression we could ever hope to achieve. Lossless source coding has been studied thoroughly, and several efficient solutions are known, including Huffman codes, arithmetic codes, and the Lempel-Ziv algorithm.

In many applications, it may not be necessary for the decoder to reconstruct the source exactly from the compressed version. For example, complex real-world data such as audio and video can typically be distorted slightly without affecting the quality substantially. The popular JPEG 2000 image compression algorithm takes advantage of fact that the human eye is not very sensitive to the high frequency components of an image in order to achieve better compression. By using fewer bits to encode the high frequency components, the decoder cannot exactly recover the original image, but the reconstruction still looks good. The MPEG standard uses similar techniques which do not allow exact reconstruction at the decoder, but maintain high-quality nonetheless.

Information theory provides a theoretical basis for analyzing algorithms like JPEG 2000. Specifically, we can study lossy source coding. In lossy source coding, the decoder does not have to reconstruct the source exactly. Instead, there is a distortion measure which is a function of the particular source realization and the reconstruction. This distortion measure quantifies how good a particular reconstruction is for a source realization. The fundamental result in this area is the rate-distortion theorem [26]. This theorem gives a characterization of the optimal tradeoff between compression rate and (average) distortion.

From a theoretical standpoint, modelling a distortion measure that captures how humans perceive signals such as audio or video is difficult. Thus, currently there is no way of determining how well an algorithm like JPEG is performing compared to an optimal lossy compression algorithm. Instead, in order to gain insight into the problem, we will consider simple distortion measures and analyze the performance of compression algorithms under these distortion measures. In particular, this thesis focuses on the binary symmetric source. This source outputs i.i.d. symbols that are 0 with probability .5 and 1 with probability .5. One of the simplest distortion measures that we can consider is the Hamming distortion. The Hamming distortion between a source and a reconstruction is simply the number of positions where the source and the reconstruction differ, i.e., the Hamming distance between the source and the reconstruction. The rate-distortion theorem can be used to show that for a binary symmetric source under Hamming distortion, if we allow a relative distortion of δ, [1] then the smallest possible compression rate is $1 - h_b(\delta)$. [2]

Although the rate-distortion theorem gives us a simple formula for the optimal

[1]Assume the source is n symbols long. Then, the relative distortion is the Hamming distance between the source and the reconstruction divided by n. Thus, the relative distortion is always between 0 and 1.

[2]$h_b()$ is the binary entropy function. For a more formal discussion of the rate-distortion theorem see, e.g., [26].

compression rate for a binary symmetric source under Hamming distortion, the result is not very useful for designing efficient algorithms for lossy source coding. The standard proof of the rate-distortion theorem uses a randomly generated quantizer. The result is that complexity of compression and decompression scales exponentially with the source block length if we simply try to use the random quantizer used in the proof. To develop a practical solution for the binary symmetric source, a quantizer with more structure is needed. Based on the close relationship between lossy source coding and channel coding, it is natural to examine good channel codes for potential as lossy source codes.

Recently, low density parity check (LDPC) codes [13] have been the object of much study. These codes are based on sparse bipartite graphs. Even though LDPC codes have much more structure than random codes, they can come arbitrarily close to channel capacity for many channel models. More importantly, LDPC codes are equipped with an efficient decoding algorithm, belief propagation (BP) [11, 12, 28]. Belief propagation has applications in many disciplines, and has been used to attack problems in machine learning, statistical inference, and error-correcting codes. Briefly, this algorithm works by passing messages along the edges of a graph. By iteratively updating these messages, BP computes approximations to certain marginal probabilities. For example, when applied to LDPC codes, belief propagation passes messages along the Tanner graph the code. This can be used to find the most probable codeword given a particular received channel output.

Based on the excellent performance of LDPC codes and BP decoding, we will examine LDPC codes for lossy source coding. In [3], the connection between lossy source coding and channel coding is developed for the binary erasure channel (BEC) and an appropriate distortion model. The authors are able to modify BP in order to develop efficient encoding algorithms for their simple distortion measure. In Chapter 2, we present some results on LDPC codes and other related codes which generalize some of the results in [3] to the case of Hamming distortion. While we are unable to develop efficient algorithms [3] for encoding and decoding, we prove several results

[3]Using linear codes, the time required for decoding a lossy source code is reduced to polynomial

that suggest that LDPC codes have potential as low-complexity lossy source codes. In particular, one of our results is that if we use an optimal encoding algorithm, LDPC codes can come arbitrarily close to the rate-distortion function for a binary symmetric source under Hamming distortion.

In addition to lossy source coding, this thesis examines information embedding. The information embedding problem, also known as channel coding with transmitter side information, combines aspects of both channel coding and lossy source coding. At a high level, the setup for information embedding is similar to standard channel coding - a transmitter would like to send data to a receiver over a noisy channel. The difference in information embedding is that the channel is now parameterized by some state information. Given this channel state information, the transmitter can potentially send data at higher rates than he could without the state information.

This general framework arises in many applications, including coding for a memory with defects, broadcast channels, inter-symbol interference channels, multi-access channels and digital watermarking; see, e.g., [15, 16]. One of the fascinating results is that in many cases, it suffices for only the transmitter to know the channel state information. Specifically, for many models, the capacity does not increase if the receiver also knows the channel state information. One method for proving results of this nature uses random codes and a binning argument. As with the rate-distortion theorem, this has the drawback that such constructions have very high complexity. Thus, developing low-complexity schemes for binning could lead to efficient solutions for many problems.

In Chapter 3, we examine the information embedding problem for a particular state and channel model. This model is very closely related to the model in [15]. For our model, we are able to develop a capacity-achieving coding scheme that can be efficiently encoded and decoded using BP. The analysis of this scheme shows the interplay between lossy source coding and information embedding.

time. The time for encoding remains exponential using naive algorithms, and for many linear codes, computing the optimal encoding is NP-complete. Our measure of an efficient algorithm is a linear-time algorithm, i.e., $O(n)$ if the block length is n.

Chapter 2

Lossy Source Coding

2.1 Introduction

In this chapter we present several results on lossy source coding. As discussed in the introduction, this problem has many practical applications. Also, as we will see in Chapter 3, to solve the problem of coding for a memory with defects [15], it seems necessary to have a code that performs well for lossy source coding under a Hamming distortion. Thus, there are many reasons to look for efficient solutions to the problem of compression under a Hamming distortion.

Because of their effectiveness in channel coding, LDPC codes are quite attractive as candidates for lossy source codes. Our focus in this chapter is to give some general performance bounds for LDPC codes and their duals, low density generator matrix (LDGM) codes. As part of our analysis, we will revisit the simpler distortion model considered in [3]. It is shown in [3] that LDPC codes are in some sense bad for this distortion model, but that LDGM codes can achieve the appropriate rate-distortion bound with efficient encoding and decoding algorithms. The results in this chapter (Section 2.2 and Section 2.3) show that LDPC codes are actually not as bad as one might think, in that these codes can come arbitrarily close to the rate-distortion functions for the distortion considered in [3], and for quantization under a Hamming distortion.

From the point of view of graphical complexity (as measured by the number of edges in the graph), [24] shows that certain codes with bounded graphical complexity per bit can come arbitrarily close to the rate-distortion function for quantization under a Hamming distortion. In Section 2.4, we prove that while LDGM codes can come arbitrarily close to the rate-distortion function, the variable degree of these codes must become unbounded in order to get closer to the rate-distortion function. Thus, other constructions have the potential for better performance than LDGM codes, if efficient algorithms can be found for encoding.

2.2 LDPC Codes Are Good For BEQ

To start our study of lossy source codes, we will reexamine the binary erasure quantization (BEQ) problem considered by [3]. The BEQ problem is defined as follows. We are given a string X of n symbols from the alphabet $\{0, 1, *\}$. We would like to compress X. The constraint is that the reconstructed version of X generated by the decoder must match the original source at all of the positions where X is 0 or 1. We can reconstruct $*$ symbols to 0 or 1. The following lemma, from [3], gives a useful relationship between good codes for the BEC and good codes for BEQ.

Lemma 1. *A linear code C with block length n can recover from a particular erasure sequence (under ML decoding) iff the dual code $C^{\perp}$ can quantize the dual sequence, i.e., the sequence where all the erased symbols have been turned into unerased symbols and vice versa. Also, if C can recover from an erasure sequence using BP decoding, then $C^{\perp}$ can quantize the dual sequence using a dualized form of BP. (For the details of how to dualize BP to work for BEQ, see [3].)*

Based on Lemma 1, one can show that LDGM codes that are duals of capacity-achieving LDPC codes for the BEC will be capacity-achieving for BEQ. On the other hand, in [3] the authors prove that low complexity LDPC codes are not effective for this problem. Specifically, they prove that the check node degree of a capacity-achieving LDPC code for BEQ must be at least $\Omega(\log n)$. This result would suggest that LDPC codes are bad quantizers, because even for the simple distortion in BEQ, these codes have at least logarithmic complexity.

As we will demonstrate, there is a simple way around the lower bound proved in [3]. The bound in [3] is reminiscent of the $\Omega(\log n)$ lower bound for LT codes on the BEC. By combining LT codes with a precode, Raptor codes are able to beat the lower bound for LT codes and achieve constant complexity encoding and decoding per bit. [1] So, let us consider the dual of Raptor code. Figure 2-1 shows the structure of a Raptor code, and figure 2-2 shows the structure of the dual of a Raptor code.

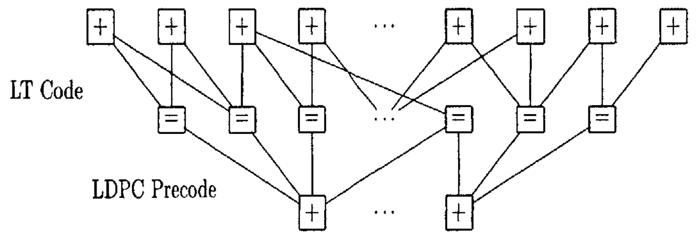

Figure 2-1: Raptor Code - This code performs well on the BEC

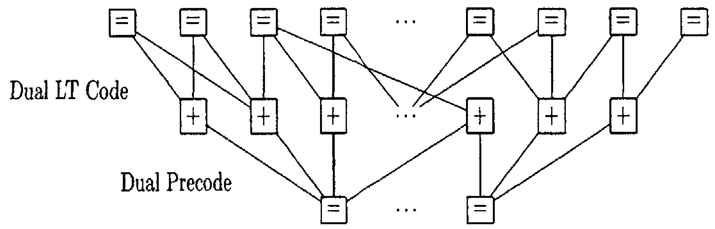

Figure 2-2: Dual Raptor Code - This code performs well for BEQ

Figure 2-2 warrants some explanation. If we define G to be a generator matrix for the LT part of a Raptor code, and H to be a parity check matrix for the precode, then a Raptor code is the set $\{c|\exists x \text{ s.t. } xG = c, Hx^T = 0\}$. Therefore, the dual is the set $\{y|\exists z \text{ s.t. } Gy^T + (zH)^T = 0\}$. To see this, note that for any words c and y satisfying these constraints, $cy^T = xGy^T = x(zH)^T = xH^T z^T = (zHx^T)^T = 0$. Thus, the set of y's defined by our constraint must be contained in the dual code. Assuming that G and H have full rank, it is easy to see that the dimension of our set of y's

[1]In chapter 3, we will discuss LT codes in more detail. For now, it suffices to know that LT codes are one class of LDGM codes, and that a Raptor code is just an LT code with an LDPC precode. The only property used in this section is that Raptor codes can achieve capacity for the BEC using BP decoding.

is the maximum possible value, which shows that the constraint exactly specifies the dual. Translating the constraint into normal graph notation [6], computing Gy^T can be accomplished by dualizing the LT part of the Raptor code, i.e., switch the variable and check nodes as in the top part of figure 2-2. [2] To compute $(zH)^T$ we just dualize the precode, i.e., switch the variable and check nodes as in the bottom part of figure 2-2. Thus, the constraint $Gy^T + (zH)^T = 0$ is exactly captured by the graph in figure 2-2. Because the dual constraint only asks for some z to exist, to interpret figure 2-2 properly, the codeword is placed on the top set of variable nodes, and the bottom variables nodes are all erased, i.e., we can set them arbitrarily in order to satisfy the check nodes.

Because Raptor codes are capacity-achieving on the BEC, it follows from lemma 1 that the duals of Raptor codes are capacity-achieving for BEQ. Now, imagine re-drawing figure 2-2 with all the variable nodes on the top, i.e., we move the variables nodes corresponding to the dual precode to the top of the figure. Then, we see that a dual Raptor code is really just an LDPC code. The important point is that the variable nodes corresponding to the precode are always erased. Thus, a dual Raptor code is an LDPC code which has been padded so that a few variable nodes are always erased. [3]

The precode is the crucial component that allows a Raptor code to beat the lower bound for LT codes. The dual of the precode is an LDGM code. Thus, dual Raptor codes get around the lower bound for BEQ by padding the input with a small number of erased symbols, and these erased symbols form an LDGM code over the check nodes of the dualized LT code. This means that no check node is connected to only unerased symbols, so the lower bound from [3] does not apply. Just as the LDPC code in a

[2]For LDGM and LDPC codes drawn using Forney's normal graph notation [6], the dual code can be obtained by swapping the variables nodes with the check nodes. More generally, swapping the variables nodes with the check nodes allows us to compute Gy^T if we do not view the check nodes as constraints, but instead think of them as producing outputs.

[3]Interpreting a dual Raptor code as an LDPC code suggests that a Raptor code can be interpreted as an LDGM code. This is true in the sense that if we move the check nodes of the precode in figure 2-1 to the top of the figure, then the result is an LDGM code where the last set of code bits must always be 0. The dual result to our claim that LDPC codes are good for BEQ is something of the form "LDGM codes can be modified slightly to perform well on the BEC."

Raptor code "fixes" the few variable nodes we could not recover by decoding the LT code, the LDGM part of a dual Raptor code fixes the few parity checks we could not satisfy using the dual LT code. The simple modification of designating a set of positions as padding symbols allows LDPC codes to achieve capacity for BEQ. This suggests that we should not give up on LDPC codes so quickly, and that they might be useful for coding under a Hamming distortion.

2.3 LDPC Codes Are Good Under Hamming Distortion

In this section we prove that LDPC codes are good under a Hamming distortion. We will actually prove a more general statement which, at a high level, means that good error-correcting codes are also good lossy source codes. To make this precise, we have the following theorem, specialized to codes for the BSC.

Theorem 1. *Consider a sequence of codes $\{C_n\}$, where the n^{th} code has block length n. Define the rate R of the sequence to be $\lim_{n\to\infty} R_n$, where R_n is the rate of the n^{th} code and it is assumed that this limit exists. Assume that for some $p, E > 0$, and sufficiently large n,*

$$\Pr[\textit{Error when using } C_n \textit{ over BSC}(p)] < 2^{-En}.$$

Let δ_{opt} be the optimal distortion for rate R, i.e., $R + h_b(\delta_{opt}) = 1$. Then, for sufficiently large n, the expected (normalized) Hamming distortion when we use C_n to quantize a binary symmetric source is less than $\delta_{opt} + \sqrt{2(h_b(\delta_{opt}) - h_b(p))}$.

Theorem 1 implies that an ensemble of codes that has a positive error exponent at all rates below capacity can achieve distortions arbitrarily close to the rate-distortion bound. For example, Theorem 1 proves that many LDPC ensembles are good for compressing a binary symmetric source under a Hamming distortion. Of course, Theorem 1 does not say anything about the computational complexity of using a channel code for lossy source coding.

We will need the following lemma in order to prove Theorem 1.

Lemma 2. *Consider a (binary) code C with codeword length n (C need not be linear). Let $p_1, \dots, p_n$ be probability distributions over $\{0,1\}$, and let $P = \prod_{i=1}^n p_i$ be a probability distribution over $\{0,1\}^n$ (P generates each source bit independently). Let D be the normalized distortion when an optimal quantizer (i.e., encode to the nearest codeword) is used. Then,* $\Pr[|D - E[D]| > \epsilon] < e^{-\frac{\epsilon^2 n}{2}}$.

Proof. We apply Azuma's inequality. Form random variables $Y_i = E[D|\text{first } i \text{ bits of source}]$. Then, the sequence $Y_i, i \in \{0, 1, \dots, n\}$ is a Doob martingale (see, e.g., [25] for a discussion of martingales). Also, by construction $Y_0 = E[D]$ and $Y_n = D$. Now, $|Y_i - Y_{i-1}| \le \frac{1}{n}, \forall i$. To see this, consider how the distortion changes when we reveal the next bit of the source. Let

$$Y_{i0} = E[D|\text{first } i \text{ bits of source and that the } i^{th} \text{ bit is } 0].$$

Similarly, let $Y_{i1} = E[D|\text{first } i \text{ bits of source and that the } i^{th} \text{ bit is } 1]$. Then, $Y_{i-1} = p_i(0)Y_{i0} + p_i(1)Y_{i1}$. Also, $Y_i = Y_{i0}$ with probability $p_i(0)$ and $Y_i = Y_{i1}$ with probability $p_i(1)$. Therefore, to show $|Y_i - Y_{i-1}| \le \frac{1}{n}$ it suffices to show $|Y_{i0} - Y_{i1}| \le \frac{1}{n}$.

To prove $|Y_{i0} - Y_{i1}| \le \frac{1}{n}$, let us expand

$$Y_{i0} = \sum_{\text{assignments to last } n-i \text{ bits}} \left(\prod_{k=i+1}^{n} p_k \right) \cdot \text{optimal distortion for this assignment},$$

and similarly for Y_{i1}. The key point is that each bit of the source is independent because of the form of P. So, the probability of any assignment to the last $n - i$ bits is the same whether we assign bit i to 0 or 1. Also, for a particular assignment to the last $n - i$ bits, the difference in distortion between sources with the i^{th} bit set to 0 or 1 is at most $\frac{1}{n}$. This is because the distortion for the optimal codeword for the source with the i^{th} bit set to 0 goes up by $\frac{1}{n}$ when used for the source with the i^{th} bit set to 1. The optimal codeword for the source with i^{th} bit set to 1 has to be at least this good. Thus, for corresponding terms in the sums for Y_{i0} and Y_{i1}, the difference in distortion is at most $\frac{1}{n}$. Therefore, the difference between the sums is also at most $\frac{1}{n}$, i.e., we have $|Y_{i0} - Y_{i1}| \leq \frac{1}{n}$. Plugging into Azuma's inequality completes the proof. $\square$

The proof of Theorem 1 is extremely simple. Roughly speaking, first we will show that if we draw balls of radius pn around the codewords of C_n, then these balls do not overlap much, i.e., the fractional volume of the intersection is negligible. This is intuitively obvious, since these balls correspond to the typical set of the noise, and a code achieving small probability of error on the BSC cannot have the noise balls overlap significantly. Next, we use lemma 2 to prove that if the balls do not have significant overlap, then the code is good under a Hamming distortion.

Proof of Theorem 1. Consider the code C_n for sufficiently large n. Let k be the information length of the code. Define the numbers $x_1, x_2, \ldots, x_{2^k}$ as follows. Let c be a codeword of C_n, and let $\epsilon_1 > 0$ be a parameter we will set later. Define $|\cdot|_H$ to be the Hamming weight of a string. Consider the Hamming "shell" of strings $w \in \{0,1\}^n$ satisfying $(p-\epsilon_1)n \leq |c+w|_H \leq (p+\epsilon_1)n$, where addition is performed in GF(2). We can draw these shells around every codeword, and potentially the shells may overlap. We define x_i to be the number of words w with the property that w lies in exactly i Hamming shells. Then, $\sum_i ix_i$ counts the number of (word, codeword) pairs such that the Hamming distance between the word and the codeword is within $(p \pm \epsilon_1)n$. We can evaluate this sum by summing from the point of view of codewords instead of words. This gives $\sum_i ix_i = 2^k \text{Vol(shell)}$, where Vol(shell) is the number of words within distance $(p \pm \epsilon_1)n$ of some fixed center (because of the symmetry of Hamming space, Vol(shell) does not change no matter what point is chosen as the center of the shell). Using Stirling's approximation, $\text{Vol(shell)} \geq \frac{2^{h_b(p)n}}{\sqrt{2\pi n}}$, so $\sum_i ix_i \geq \frac{2^{k+h_b(p)n}}{\sqrt{2\pi n}}$.

Now, consider ML decoding of C_n over the BSC. This corresponds to drawing a Voronoi region in Hamming space around each codeword. Consider a word w that contributes to x_i, i.e., w is contained in i Hamming shells. We can assume that w is assigned to a particular codeword, that is to say we can use a deterministic ML decoder without increasing the error probability (for example, we can break ties by assigning a word to the lexicographically first codeword that it is closest to.)

Given this deterministic rule, consider what happens when we transmit one of the i codewords that w is in the Hamming shell of. Out of these i codewords, w gets decoded to at most 1 since our decoding rule is deterministic. Thus, for the remaining $i-1$ codewords, if we receive w then we will make an error with probability 1. From the AEP, we know that $\Pr[\text{receive } w|c] \geq 2^{-(h_b(p)+\epsilon_1)n}$, where c is one of the $i-1$ remaining codewords.

Armed with this observation, we can lower bound the probability of error by $\sum_i (i-1)x_i 2^{-k-(h_b(p)+\epsilon_1)n}$, since all codewords have probability 2^{-k} of being transmitted. By assumption, this quantity is less than 2^{-En}. Rearranging terms, $\frac{2^{k+h_b(p)n}}{\sqrt{2\pi n}} \leq \sum_i i x_i \leq 2^{k+(h_b(p)+\epsilon_1-E)n} + \sum_i x_i$. From the AEP, we can choose $\epsilon_1 < E$ for sufficiently large n, since Theorem 1 assumes that $E > 0$ is a constant independent of n. Comparing exponents then gives $\frac{2^{k+h_b(p)n}}{2\sqrt{2\pi n}} \leq \sum_i x_i$, for sufficiently large n. But $\sum_i x_i$ is exactly the number of words in the Hamming shells when we count without multiplicity. Therefore, we have shown that for sufficiently large n, at least $2^{(R+h_b(p)-o(1))n} = 2^{(1+h_b(p)-h_b(\delta_{opt})-o(1))n}$ words are within distance $p+\epsilon_1$ of some codeword (The $o(1)$ is because R is a limit, so the R_n may be slightly less than R, but we can make the difference as small as we like by making n larger). Let D_n be the expected distortion when we use C_n to quantize a binary symmetric source. Then, we have shown $\Pr[D_n < p+\epsilon_1] \geq 2^{-(h_b(\delta_{opt})-h_b(p)+o(1))n}$.

Recalling that lemma 2 applies to any code, we can conclude that $E[D_n] \leq \delta_{opt} + \sqrt{2(h_b(\delta_{opt}) - h_b(p))} - (\delta_{opt} - p) + o(1)$, where we have absorbed terms involving ϵ_1 into the $o(1)$ term. □

2.3.1 Comments

1. The proof above appears to be quite general. As mentioned earlier, the small overlap property is quite intuitive, and should be quite simple to generalize to other channel models. Even the constraint of asking for a positive error exponent can probably be relaxed by looking at the AEP more carefully, but we have not tried this. The proof of lemma 2 is also quite general, and only requires a bounded distortion metric. Thus, it seems that for a broad class of

channels, we should be able to show that channel codes are automatically good for properly dualized distortion metrics.

2. A natural question is whether the correspondence goes the other way, i.e., are good quantizers also good channel codes? The short answer is no, because if we have a good quantizer, we can add words next to each quantizer word, resulting in a quantizer with essentially the same rate but terrible error-correction properties. However, it may be possible to show that good quantizers can be turned into good error-correcting codes with small modifications. In particular, it seems that given a good quantizer, it should be possible to perform a small amount of expurgation so that the expurgated code is a good channel code. [4] The intuition for why expurgation should be enough to turn a good quantizer into a good channel code is that although the block error probability is high when we add words next to each quantizer word, the conditional entropy of the channel input is small.

3. We can also ask for codes that come close to the rate-distortion function for quantization under a Hamming distortion in the worst-case. To see this, say we have a code with expected distortion δ. From lemma 2, with high probability the observed distortion is at most $\delta + \epsilon$. So, to create a worst-case quantizer, repeatedly try to quantize a given source by adding independent, uniformly chosen strings to the given source. A simple union bound argument shows that after a constant number of trials (the number is a function of ϵ), with high probability the worst-case distortion is at most $\delta + \epsilon$. If we want to restrict ourselves to linear codes, the arguments in [20] show that we can append a few extra columns (or rows for LDGM codes) so that the resulting linear code works well in the worst-case. The basic idea is that by adding $O(\log n)$ suitably chosen columns, we can "fix" the original code to cover the small fraction of words that previously would have been quantized with high distortion. The drawback of

[4] By small, we mean that the rate loss due to the expurgation should be at most ϵ for some small ϵ. ϵ should be a function of the quantizer, i.e., better quantizers should have a smaller value of ϵ, and ϵ should approach 0 for quantizers that approach the rate-distortion bound.

this method is that the resulting code may no longer be sparse.

4. The authors of [19] analyze the performance of MacKay's ensemble for quantization under a Hamming distortion. One might hope for a more direct proof that LDPC codes are good for quantization by strengthening their approach. The authors use a fairly complicated proof, but the same result can be derived using Chebyshev's inequality and bounding the covariance term. However, even with this method, choosing codes from MacKay's ensemble still gives a logarithmic degree bound. Intuitively, the reason MacKay's ensemble does not give a better bound is that it takes $n \ln n$ steps before a random walk on a hypercube has mixed well enough for the covariance term to be small.

   The analysis in [19] is based on trying to lower bound the probability that all syndromes can be written using only a few columns of the parity check matrix. This approach appears to have a fundamental limit that makes it impossible to prove any bound of better than logarithmic degree for MacKay's ensemble. Essentially, if we want to be able to write all syndromes, the matrix we look at must have full rank. However, one can easily show that if we use a constant (or sublogarithmic) degree in MacKay's ensemble, then with high probability the matrix will have an all 0 row. Thus, with high probability the matrix does not have full rank. The probability of every row having at least one entry set to 1 does not become large until the degree becomes logarithmic, which explains why the analysis in [19] only produces a logarithmic degree bound.

2.4 A Lower Bound On The Degree Of LDGM Codes

Recently, several authors have considered a compound code formed by concatenating LDGM and LDPC codes. It has been shown that this compound construction has remarkable graph complexity properties. In particular, bounded degree compound codes can come arbitrarily close to channel capacity under ML decoding for any memoryless binary input output-symmetric channel [23]. Also, bounded degree compound codes under optimal encoding can come arbitrarily close to the rate-distortion bound for a binary symmetric source under Hamming distortion [24]. Finally, several classes of compound codes have been found that achieve capacity on the BEC using message-passing with bounded complexity, i.e., we can come arbitrarily close to capacity without the running time increasing as a function of the gap to capacity.

These results beg the question of whether LDGM or LDPC codes can achieve similar performance, i.e., come arbitrarily close to either channel capacity or the rate-distortion function while keeping the variable and check node degrees bounded. For channel coding, bounded degree LDGM codes have terrible minimum distance, and hence can never achieve small block error rates on the BSC. Also, there is a well-known bound due to Gallager that says that bounded degree LDPC codes cannot come arbitrarily close to channel capacity, i.e., on the BSC achieving smaller gaps requires increasing the degree. This shows that compound constructions outperform[5] pure LDGM or LDPC constructions for channel coding.

We will prove that for quantization under a Hamming distortion, the variable node degree of any LDGM code must become unbounded. Note that this applies to any code, not codes drawn from some specific random ensemble.

Theorem 2. *Let d be the maximum degree of any variable node in some LDGM code. Let R be the rate of the code, and let δ be the expected distortion. Define $\epsilon = \delta - \delta_{opt}$, where δ_{opt} is the optimal distortion for rate R, i.e., $R + h_b(\delta_{opt}) = 1$. Then, $d = \Omega(\log \frac{1}{\epsilon})$.*

Proof. This is another simple application of lemma 2. If we have a short code, we can repeat the code many times to get a long code with the same expected distortion. Therefore, consider a long code with rate R and maximum variable degree d. We will prove Theorem 2 by proving a lower bound for ϵ in terms of d. Let D be a random variable denoting the observed normalized distortion. From lemma 2, we

[5]All of these results assume maximum-likelihood decoding. From the point of view of iterative decoding, it is still unclear whether the new constructions improve the number of BP iterations compared to irregular LDPC codes.

know that $\Pr[D > \delta_{opt} + 2\epsilon] < e^{-\frac{\epsilon^2 n}{2}}$. Thus, since the source is uniform, at least $2^n - 2^{(1-\frac{\log(e)\epsilon^2}{2})n} \doteq 2^n$ words have distortion less that $\delta_{opt} + 2\epsilon$.

Now, pick some ϵ_1. If we flip $\frac{\epsilon_1 n}{d}$ information bits of the code, the distortion increases by at most $\epsilon_1 n$, using the triangle inequality. So, we will prove the theorem by counting the number of (codeword, source) pairs such that the relative distance between the codeword and the source is $\leq \delta_{opt} + 2\epsilon + \epsilon_1$ in two different ways. First, from the point of view of codewords, this sum is clearly $\doteq 2^{(R+h_b(\delta_{opt}+2\epsilon+\epsilon_1))n}$. Now, we have already observed that almost every source has a codeword within distance $\delta_{opt} + 2\epsilon$. Therefore, using our observation about flipping information bits, the sum is $\dot{\geq} 2^{(1+Rh_b(\frac{\epsilon_1}{Rd}))n}$. Because a good short code can be repeated to form a good long code, we can compare the exponents in the two expressions to obtain $R + h_b(\delta_{opt} + 2\epsilon + \epsilon_1) \geq 1 + Rh_b(\frac{\epsilon_1}{Rd})$. Now, $h_b()$ is a concave function, so $h_b(\delta_{opt} + 2\epsilon + \epsilon_1) \leq h_b(\delta_{opt}) + h_b'(\delta_{opt})(2\epsilon + \epsilon_1)$. So, we get $h_b'(\delta_{opt})(2\epsilon+\epsilon_1) \geq Rh_b(\frac{\epsilon_1}{Rd})$, i.e., $\epsilon \geq \frac{1}{2}(\frac{Rh_b(\frac{\epsilon_1}{Rd})}{h_b'(\delta_{opt})} - \epsilon_1)$. Note that our entropy bounds work for any $\epsilon_1 \leq \frac{Rd}{2}$.

To complete the proof, we optimize over ϵ_1. Taking the derivative with respect to ϵ_1, we find that

$$\frac{d}{d\epsilon}\left(\frac{Rh_b(\frac{\epsilon_1}{Rd})}{h_b'(\delta_{opt})} - \epsilon_1\right) = \frac{\log(e)}{dh_b'(\delta_{opt})}\log\left(\frac{Rd}{\epsilon_1} - 1\right) - 1$$

Setting this expression to 0, we find that the optimal value is $\epsilon_1^* = \frac{Rd}{1+2^{dh_b'(\delta_{opt})}}$. Because h_b' is nonnegative over the range of interest ($\delta_{opt} \leq \frac{1}{2}$), we see that $\epsilon_1^* \leq \frac{Rd}{2}$

as required. Substituting in ϵ_1^*, we find that

$$
\begin{aligned}
\epsilon &\geq \frac{1}{2}\left(\frac{Rh_b(\frac{\epsilon_1^*}{Rd})}{h_b'(\delta_{opt})} - \epsilon_1^*\right) \\
&= \frac{1}{2}\left(\frac{R}{h_b'(\delta_{opt})}\frac{1}{1+2^{dh_b'(\delta)opt}}\log\left(1+2^{dh_b'(\delta_{opt})}\right) + \frac{R}{h_b'(\delta_{opt})}\frac{2^{h_b'(\delta_{opt})}}{1+2^{dh_b'(\delta)opt}}\log\left(\frac{1+2^{dh_b'(\delta_{opt})}}{2^{dh_b'(\delta_{opt})}}\right)\right. \\
&\quad \left. -\frac{Rd}{1+2^{dh_b'(\delta_{opt})}}\right) \\
&\geq \frac{R}{2h_b'(\delta_{opt})}\frac{2^{dh_b'(\delta_{opt})}}{1+2^{dh_b'(\delta_{opt})}}\log\left(1+2^{-dh_b'(\delta_{opt})}\right) \\
&\geq \frac{R}{2h_b'(\delta_{opt})}\frac{\left(1-2^{-1-dh_b'(\delta_{opt})}\right)\ln(2)}{1+2^{dh_b'(\delta_{opt})}} \\
&\geq \frac{R}{8h_b'(\delta_{opt})}\frac{1}{1+2^{dh_b'(\delta_{opt})}}.
\end{aligned}
$$

To obtain the second to last inequality, we have used the inequality $x\ln(1+\frac{1}{x}) \geq 1 - \frac{1}{2x}$. In the last inequality, we have used the fact that $h_b'(\delta_{opt})$ is nonnegative, so the first term in the numerator is at least $\frac{1}{2}$, and $\ln(2) > \frac{1}{2}$. If we invert the last inequality, we see that $d \geq \frac{\log\left(\frac{R}{8h_b'(\delta_{opt})\epsilon}-1\right)}{h_b'(\delta_{opt})}$, proving that d must be $\Omega(\log(\frac{1}{\epsilon}))$. □

2.4.1 Comments

This bound is in some sense tight. We can prove that if $d = O(\log\frac{1}{\epsilon})$, then the gap can be reduced to ϵ. One way to see this is to consider a dualized form of MacKay's ensemble. We cannot use the simpler Poisson ensemble considered in [24] because a few variable nodes might have large (logarithmic) degree. Of course, looking at the proof of Theorem 2, we see that having a few nodes with large degree does not affect the proof since we can always flip the other information bits. However, to prove the bound is tight for the case of the maximum degree being d, it is easier to use a dualized form of MacKay's ensemble because this ensemble is guaranteed to have a constant maximum variable node degree. [6]

[6]By"dualized" form of MacKay's ensemble, we mean an ensemble that selects each row of the generator matrix independently using a random walk. This is not quite the dual of MacKay's ensemble, since the dual would select each column independently. However, selecting rows independently is more convenient since this guarantees a bounded maximum variable degree. The proof that the

Also, we note that in another sense, the bound in Theorem 2 is quite loose. The bound says $\epsilon = \Omega(\frac{1}{2^{\alpha d}})$ for some constant α, while the random ensembles have a much larger value of ϵ. (Note : we can optimize the bound without using the simplification of approximating h_b by a first order Taylor series expansion. However, this does not improve the bound significantly.)

dualized form of MacKay's ensemble works is similar to [24], but involves quite a bit of algebra.

Chapter 3

Information Embedding

3.1 Introduction

In this chapter we consider information embedding, a more complicated coding problem which has both channel and source coding aspects. As discussed in the introduction, this problem arises in several contexts. There is a growing interest in understanding the complexity required to approach capacity for information embedding, and how to design codes with such complexity.

Low density codes on graphs are compelling candidates for the information embedding problem. As was mentioned in chapter 2, LDPC codes and LDGM codes present efficient solutions for the BEC and for BEQ. In closely related work, such codes have been used to approach capacity of the noiseless broadcast channel [17, 18], but some difficulties remain. For example, [17] requires logarithmic (as opposed to constant) density in the block length while [18] uses an $O(n^2)$ algorithm. Furthermore, it is unclear how those approaches fare in the presence of channel noise.

In this chapter, we analyze what may be the simplest information embedding channel model whose source and channel coding aspects are both nontrivial. The model we consider is a variant of the channel analyzed in [15]. The precise model is discussed in Section 3.2. For our simpler channel, we construct a class of capacity-achieving linear complexity codes.

The chapter is organized as follows. The information embedding channel of in-

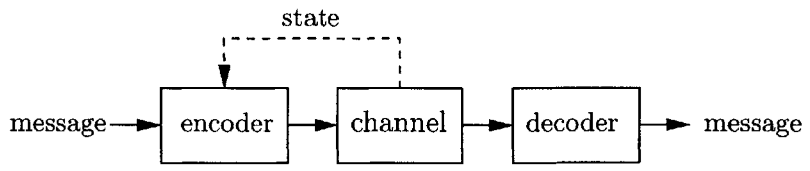

Figure 3-1: The information embedding model. The encoder embeds a message into the channel state, and the decoder recovers the embedded message from the channel output.

terest is described in Section 3.2. Two simpler cases are analyzed in Section 3.3 and Section 3.4. A code construction for the general case is developed in Section 3.5. Our main results are stated in Section 3.6, with the associated proofs summarized in Section 3.7. A discussion of an alternative channel model is given in Section 3.8, and some possible extensions of our results are given in Section 3.9. Finally, Section 3.10 discusses some points related to developing a practical implementation our codes.

3.2 Channel Model

The general information embedding problem is illustrated in Fig. 3-1. A channel state vector $\mathbf{s}$ consisting of n symbols from the alphabet Σ_s is selected according to the probability law $p(\mathbf{s})$. The encoder takes as input $\mathbf{s}$ as well as a k-bit message $\mathbf{m}$, and produces a channel input vector $\mathbf{x}$ consisting of n symbols from the alphabet Σ_x. The channel takes $\mathbf{x}$ as input and produces a channel output vector $\mathbf{y}$ consisting of n symbols from the alphabet Σ_y according to the probabilistic channel law $p(\mathbf{y}|\mathbf{x},\mathbf{s})$. Finally, the decoder receives $\mathbf{y}$, (which we sometimes denote as $\mathbf{y}(\mathbf{x},\mathbf{s})$ to indicate the dependence on the channel input and state), and attempts to determine the message $\mathbf{m}$. The goal of the information embedding problem is to construct systems operating at rates near capacity with low complexity encoders and decoders, with the probability of decoding error vanishing as $n \to \infty$.

Our "double-erasure" information embedding channel of interest is a variant of the "memory with defects" channel model [15]. Specifically,

$$\Sigma_x = \{0, 1\}, \quad \Sigma_s = \Sigma_y = \{0, 1, *\}, \tag{3.1}$$

the state s is independent and identically distributed (i.i.d.) with

$$p_S(s) = \begin{cases} (1 - e_s)/2 & s = 0 \text{ or } s = 1 \\ e_s & s = *. \end{cases} \tag{3.2}$$

and the channel law is i.i.d. with

$$p_{Y|S,X}(y|s, x) = \begin{cases} \frac{e_c}{e_s}, & y = * \text{ and } s = * \\ 1 - \frac{e_c}{e_s}, & y = x \text{ and } s = * \\ 1, & y = s \text{ and } s \neq *. \end{cases} \tag{3.3}$$

The channel is almost identical to the channel considered in [15]. Instead of allowing the channel to introduce errors, we only allow erasures. The erasures introduced by the channel can only occur in positions where the state was erased, i.e., unerased state positions are never changed by the channel. Also, we define the channel erasure probability as $\frac{e_c}{e_s}$ so that the expected fraction of erasures introduced by the channel is e_c. It may seem that allowing the channel to only introduce erasures makes the problem too simple. In Section 3.8, we will show that a small change in the statistical distribution of the erasures introduced by the channel makes our model essentially as difficult as the original channel in [15].[1]

The capacity of the information embedding problem for the state and channel models given by (3.2) and (3.3) is $e_s - e_c$. To see that $e_s - e_c$ is an upper bound, consider the case where the encoder and decoder both know the state. In this case, the channel is essentially a BEC with erasure probability $\frac{e_c}{e_s}$, but with the added constraint that out of n channel uses, only $e_s n$ are actually usable because the rest

[1]It turns out that in some sense, changing the statistics of the channel erasures forces a good encoder to deliberately introduce errors. The random coding argument in [15] also relies on the encoder to deliberately add errors.

will be hardcoded positions. Thus, when the encoder and decoder both know the state, the capacity is $e_s(1-\frac{e_c}{e_s}) = e_s - e_c$. Because the decoder does not see the state, the capacity can only decrease. In Section 3.5, we show that the capacity is in fact $e_s - e_c$ by constructing codes that achieve rates arbitrarily close to this upper bound.

3.3 A Simple Scheme For Large Alphabets

To develop some intuition for code design, let us consider a simpler case. Assume that the channel state is drawn from a large alphabet of size q, and assume the state block length $n \leq q$. Assume that the channel state has $\geq e_s n$ erasures with high probability, and that the channel introduces $\leq e_c n$ erasures with high probability. Then, there is a simple coding scheme using Reed-Solomon (RS) codes that can transmit at rate $e_s - e_c$. The precise statistical model for the state and the channel does not affect the scheme, provided that the right number of erasures occur. Thus, our scheme could be useful in situations where there is no information about the distribution of the state or channel erasures.

Treat the state $s = s_1 s_2 \ldots s_n$ as an RS codeword, i.e., the evaluations of some polynomial at n points $x_1, \ldots, x_n \in$GF(q). Then, we can transmit a message $m = m_1 m_2 \ldots m_{n(e_s-e_c)}$ of length $n(e_s - e_c)$ as follows. Form a string $t = t_1 t_2 \ldots t_{n(1-e_s)}$ of $n(1-e_s)$ unknown symbols, and append the $n(e_s - e_c)$ symbols of m to get the string $tm = t_1 \ldots t_{n(1-e_s)} m_1 \ldots m_{n(e_s-e_c)}$, with total length $n(1-e_c)$.

For encoding, m is known, and s has $n(1-e_s)$ or fewer unerased symbols. Therefore, we can find an assignment to t such that the encoding of tm matches s at all unerased positions. Specifically, we want to find t such that for

$$\sum_{i=1}^{n(1-e_s)} t_i x_j^{i-1} + \sum_{i=1}^{n(e_s-e_c)} m_i x_j^{n(1-e_s)+i-1} = s_j$$

for all j where the state is unerased, i.e., $s_j \neq *$. The second term on the left hand side is known because the message is an input to the encoder, so we can subtract to get a system of equations for the t_i. The fact that there are at most $n(1-e_s)$

unerased symbols in s means that there are at most $n(1-e_s)$ equations constraining the t_i. Since t has length $n(1-e_s)$, it would appear that this system must always have a solution. This is true, and can be proved in many ways (for example, the Vandermonde matrix is always invertible; also, this is a simple consequence of the fact that RS codes are MDS). Furthermore, there are efficient algorithms for finding the required t. If there are fewer than $n(1-e_s)$ unerased symbols in s, assign values to some of the erased symbols such that the new string s' has exactly $n(1-e_s)$ erasures. Then, our previous arguments show that there exists a unique t such that the RS encoding of tm matches s' at all unerased positions. To find this t, we can apply any standard decoding algorithm for RS codes. Then, we apply an RS encoder to find the final encoding, i.e. assign values to any remaining erased positions in s'.

To decode, we receive at least $n(1-e_c)$ unerased symbols. Therefore, we can solve for the length $n(1-e_c)$ input tm that was used during encoding, again by applying an RS decoding algorithm. Then, the last $n(e_s-e_c)$ bits are m, the message we wanted to transmit. Thus, we can reliably communicate using this simple scheme. The reason that this method works is that RS codes are MDS codes, which implies that they are simultaneously optimal for both channel coding and erasure quantization. [2] To design a scheme that works for the binary alphabet, we need to use binary codes that are simultaneously close to optimal for both channel coding and erasure quantization. In Section 3.5 we will develop a scheme that works over the binary alphabet by proving that certain binary codes are nearly MDS codes, and have nearly MDS duals. This fact allows us to construct codes for information embedding with rates arbitrarily close to $e_s - e_c$.

Finally, note that the reason that the above method can only transmit at rate $(e_s - e_c)$ is because our coding scheme does not take into account the statistical models for the erasures in the state and the channel, so it has to work even in the worst case placement of the erasures. In particular, it is obvious that if the state

[2]To see that an MDS code is optimal for erasure quantization, recall that the Singleton bound implies that given any set of $n(1-e_c)$ output positions, the codewords take on all possible $n(1-e_c)$-length strings when restricted to these positions, and each string occurs exactly once. Therefore, we can quantize optimally. Alternatively, note that the dual of an MDS code is MDS, and apply Lemma 1.

always has the first $e_s n$ positions erased, and the channel always erases the first $e_c n$ positions, then the maximum rate of transmission $e_s - e_c$. Thus, without using the statistics of the state and channel, we cannot do better than our RS scheme. For the state and channel models given in Section 3.2, this is not a problem since $e_s - e_c$ is the capacity. However, in a situation where the erasure model is unknown, we need to bear in mind that some erasure models have a higher capacity. For example, the model considered in Section 3.8 has a higher capacity. Thus, to transmit at higher rates for these different channels, we need to take into account the models for state and channel erasures.

3.4 Simple Coding Schemes for the case $e_c = 0$

In this section we consider the information embedding problem when $e_c = 0$, i.e., the channel introduces no erasures. The state is drawn from (3.2), i.e., the alphabet is binary. In this case, the capacity is just e_s, and there is a particularly simple way to get arbitrarily close to capacity. Based on the results from the last section, we would think that if there are no channel erasures, then we only need a code that has a nearly MDS dual. As we now show, this intuition is correct.

To give a more precise definition of what it means for a code to have a nearly MDS dual, we will use the BEQ problem. The information embedding problem with $e_c = 0$ essentially boils down to finding a code C_1 that is good at BEQ.[3] Therefore, using Lemma 1, if C_1 is the dual of a good code for the BEC, C_1 should work well for this setup. As a concrete example, consider taking C_1 to be the dual of a nonsystematic irregular repeat-accumulate (IRA) code [5]. Figure 3-2 shows the normal graph [6] representation of a nonsystematic IRA code.

These codes are the same as traditional IRA codes, except that the systematic bits have been punctured. It is shown in [5] that these codes can achieve capacity on the BEC for all erasure probabilities $p \in [0, .95]$, and it is conjectured that these codes

[3]We also need C_1 to be invertible. Specifically, we would like a linear-time algorithm that takes as input a codeword and outputs the information bits that produce this codeword.

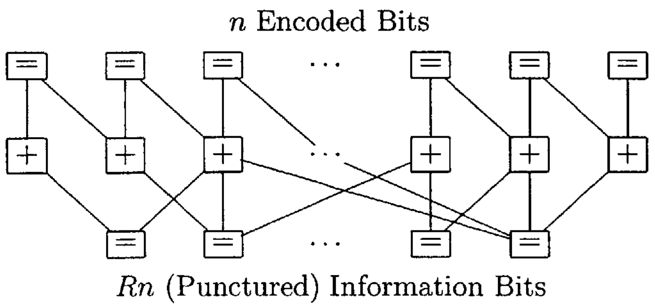

Figure 3-2: Nonsystematic Irregular Repeat-Accumulate Code

achieve capacity $\forall p \in [0,1)$. Furthermore, these codes achieve capacity with *bounded complexity*. This means that unlike many other known codes, as we approach capacity, the complexity does not become an unbounded function of the gap to capacity.

Figure 3-3 shows how the dual of a nonsystematic IRA code looks.

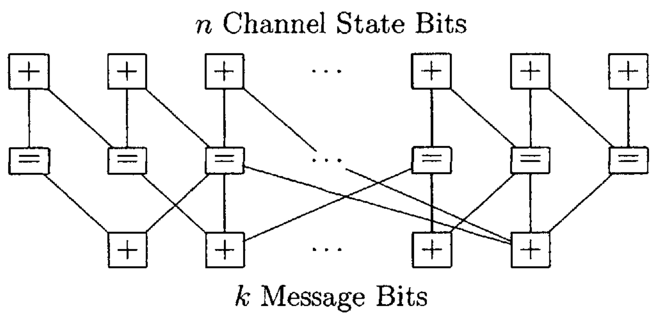

Figure 3-3: Dual Nonsystematic IRA Code - this code is capacity-achieving when $e_c = 0$

Dual nonsystematic IRA codes can achieve capacity for information embedding when $e_c = 0$. Encoding proceeds as follows. The channel state is placed on the top n check nodes of the code, as shown in figure 3-3. The message is placed along the bottom k check nodes. Using the dualized version of BP for BEQ, we calculate values for the n intermediate variable nodes such that all unerased checks are satisfied. In order to use the algorithm given in [3] to calculate the values for the variable nodes, one cannot simply apply the message passing rules. Instead, one needs to pretend that the variable nodes have "hidden" inputs which need to be determined, and then use the ERASURE-QUANTIZE algorithm given in [3] to determine the values of these inputs. Finally, given the values of the variable nodes, the encoder calculates the values of the erased symbols in the state, i.e the erased check nodes. The output is the top n check nodes. Figure 3-4 gives an example of what we mean by "hidden" inputs.

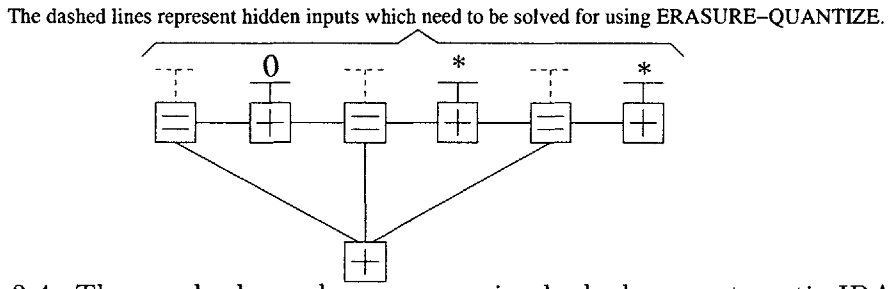
The dashed lines represent hidden inputs which need to be solved for using ERASURE–QUANTIZE.
0
*
*

Figure 3-4: The graph above shows a very simple dual nonsystematic IRA code. If we pretend the variable nodes have "hidden" inputs, the code can be thought of as an LDGM code. Then, ERASURE-QUANTIZE [3] can be used to solve for the variable nodes. Finally, values for the erased checks can be determined by adding the appropriate variable node values together (the message-passing rules in [3] will not give the correct answer on this example).

Since $e_c = 0$, the decoder sees the top n check nodes as calculated by the encoder. Now, the variable nodes are simply the check nodes accumulated, i.e., $v_i = \sum_{k=1}^{i} c_i$, so we can calculate the values of all the variable nodes in $O(n)$ time. Then, we can calculate the values of the bottom k check nodes by summing the appropriate variable nodes, as determined by the graph. Since the graph has $O(n)$ edges, this step also takes $O(n)$ time.

Thus, dual nonsystematic IRA codes appear to present a computationally efficient solution to the information embedding problem when $e_c = 0$, and also appear capable of achieving rates arbitrarily close to capacity, i.e., e_s. Technically, we still have not given a precise definition of what it means for a sequence of codes to achieve capacity. A formal definition which also works for the general case where $e_c \geq 0$ will have to wait until the next section. However, for the sake of having a precise result, we summarize our reasoning so far in the following lemma, which verifies that dual nonsystematic IRA codes achieve capacity. The proof assumes knowledge of the definitions given in the next section.

Lemma 3. *There exists a sequence of dual nonsystematic IRA codes which achieves capacity for information embedding when $e_c = 0$ and $e_s \in [.05, 1]$.*

Proof. We need to prove that dual nonsystematic IRA codes have the encoding and decoding properties. From Lemma 1 and [5], we conclude that there exist dual nonsystematic IRA codes which are capacity achieving for BEQ, provided that the erasure probability (e_s in our case) is $\geq .05$. This means that for these dual nonsystematic IRA codes, our encoding procedure succeeds with high probability as long as $e_s \geq \frac{k}{n}$. The decoding property is trivial to verify since if encoding succeeds, then our decoder recovers the message bits with probability 1. By taking $\frac{k}{n}$ arbitrarily close to e_s, we can construct capacity-achieving ensembles, since we observed earlier that the capacity is e_s. (Note: the encoding property requires that encoding works always. So, we can modify the encoding algorithm as follows. If BP fails, the encoding is just the state with erased positions set to 0. Then, the decoder fails with a small probability, namely the probability that BP failed during the encoding step.) $\square$

3.5 A Coding Scheme For The Binary Alphabet When $e_c \geq 0$

In this section we describe a capacity-achieving coding scheme for the general case where $e_c \geq 0$. A coding scheme consists of a sequence of encoding functions[4] $E_n : \Sigma_s^n \times \{0,1\}^k \mapsto \Sigma_x^n$ and decoding functions $D_n : \Sigma_y^n \mapsto \{0,1\}^k$ for $n = 1, 2, \ldots$.

Definition 1. *A coding scheme is* admissible *for the double-erasure information embedding channel if i) $E_n(\mathbf{s}, \mathbf{m})_i = \mathbf{s}_i$ whenever $\mathbf{s}_i \neq *$ for all messages $\mathbf{m}$ (cf.* (3.2)*); and ii) for a message $\mathbf{m}$ drawn at random, and any $\epsilon > 0$, there are infinitely many n such that* $\Pr[D_n(\mathbf{y}(E_n(\mathbf{s}, \mathbf{m}), \mathbf{s})) \neq \mathbf{m}] \leq \epsilon$. *The* rate *of an admissible coding scheme is defined as $R = \limsup(k/n)$.*

Our encoder, illustrated in Fig. 3-5, is formed by combining an $(n + \tilde{n}, \tilde{k})$ LDGM

[4]We use the notation A^b to denote the b-fold Cartesian product of a set with itself and $\mathbf{c}_i$ to denote the ith component of a vector $\mathbf{c}$.

code $\mathcal{C}_1$ and an $(\tilde{n}, k)$ LDPC code $\mathcal{C}_2$.[5] A k-bit message $\mathbf{m}$ is encoded into an n-bit channel input $\mathbf{x}$ as a function of the n-bit channel state $\mathbf{s}$ as follows:

1. Encode $\mathbf{m}$ using $\mathcal{C}_2$, obtaining $\mathbf{w} = \mathbf{G}_2 \cdot \mathbf{m}$, where $\mathbf{G}_2$ is the generator matrix for $\mathcal{C}_2$.

2. Use a modified version of belief propagation (BP) [3, 11, 12] to find a codeword of $\mathcal{C}_1$ denoted $(\tilde{\mathbf{s}}, \tilde{\mathbf{w}})$ such that $(\tilde{\mathbf{s}}, \tilde{\mathbf{w}})$ matches $(\mathbf{s}, \mathbf{w})$ in as many non-$*$ positions as possible. For example, this can be implemented by directly applying the ERASURE-QUANTIZE algorithm of [3]. If ERASURE-QUANTIZE fails, randomly assign values to all of the so-called unreserved variables [3], thereby incurring some small number of errors. Then, solve for the reserved variables as if ERASURE-QUANTIZE did not fail.

3. The channel input is the n-bit vector $\mathbf{x}$ where $x_i = s_i$ if $s_i \neq *$, and $x_i = \tilde{s}_i$ if $s_i = *$.

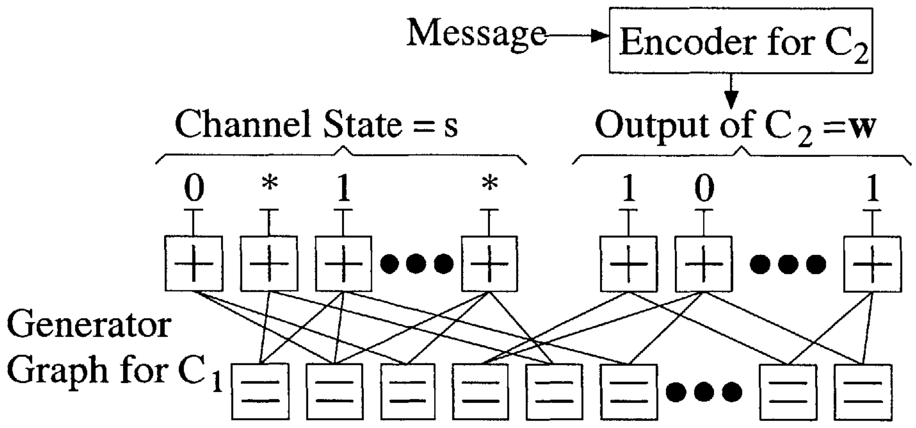

Figure 3-5: Encoder structure. The message consists of k bits, which are encoded by code $\mathcal{C}_2$ to produce $\tilde{n}$ outputs labeled $\mathbf{w}$. After concatenating $\mathbf{w}$ onto the n-bit channel state $\mathbf{s}$, the encoder finds a codeword of the main code $\mathcal{C}_1$ that matches $(\mathbf{s}, \mathbf{w})$ in as many non-$*$ positions as possible.

[5]By LDGM and LDPC codes, we mean codes with a graphical representation having $\mathcal{O}(r)$ edges, where r denote the block length. In particular, this definition allows codes that have unbounded maximum degree, as long as the average degree is bounded by a constant independent of r.

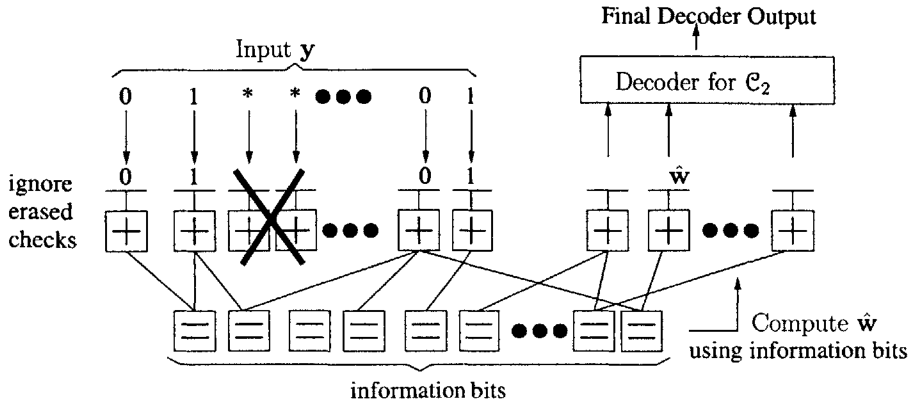

Figure 3-6: Decoder Structure. The arrows indicate the flow of information through the decoder from the channel output y to the final decoder output.

To understand the encoding algorithm, it helps to contrast the ideal case where there exists a codeword of $\mathcal{C}_1$ that exactly matches $(\mathbf{s}, \mathbf{w})$ in all non-$*$ positions with what actually happens. Usually, there will be at least a few positions of $(\mathbf{s}, \mathbf{w})$ that cannot be exactly matched by a codeword of $\mathcal{C}_1$. The encoding algorithm accounts for this in step 3 by changing the positions of $\tilde{\mathbf{s}}$ that do not match the non-$*$ positions of $\mathbf{s}$. The decoder will need to correct these errors in addition to the erasures in the channel output.

Fig. 3-6 illustrates the decoder for our codes. Decoding a received n-bit channel output $\mathbf{y}$ proceeds as follows:

1. Form the subgraph of $\mathcal{C}_1$ obtained by ignoring the erased positions of the received signal $\mathbf{y}$ and the last $\tilde{n}$ positions of $\mathcal{C}_1$.

2. Use BP on this subgraph as if the vector $\tilde{\mathbf{s}}$ was corrupted by a binary symmetric channel (BSC) to estimate the information bits of $\mathcal{C}_1$. Then, use the information bits to compute an estimate $\hat{\mathbf{w}}$ of $\mathbf{w}$.

3. Decode $\mathcal{C}_2$ to recover the message $\mathbf{m}$ from $\hat{\mathbf{w}}$. [6]

We first discuss the required properties of $\mathcal{C}_1$.

[6]In practice, one would probably want to use a code $\mathcal{C}_2$ that would allow BP decoding.

Definition 2. *A $\mathcal{C}_1$ code ensemble is* good *if for some choice of $\epsilon_s, \epsilon_c, \delta_s, \delta_c$ it is $(\epsilon_s, \epsilon_c, \delta_s, \delta_c)$-good. The latter is a family of $\mathcal{C}_1$ codes, with a probability distribution over the family, mapping $\tilde{k}$ information bits to $n + \tilde{n}$ code bits, where members $\mathcal{C}_1$ of the ensemble have the following two properties:*

1. *Erasure Quantization: Let $\mathbf{t} \in \Sigma_s^{n+\tilde{n}}$ be arbitrary. If the number of $*$ symbols in $\mathbf{t}$ exceeds $n + \tilde{n} - \tilde{k}(1 - \epsilon_s)$, then with high probability, there exists a codeword $\mathbf{c} \in \mathcal{C}_1$ such that $|\{i : c_i \neq \mathbf{t}_i \text{ and } \mathbf{t}_i \in \{0,1\} \text{ and } i < n\}| \leq \delta_s n$ and $|\{i : c_i \neq \mathbf{t}_i \text{ and } \mathbf{t}_i \in \{0,1\} \text{ and } i > n-1\}| \leq \delta_s \tilde{n}$.*

2. *Erasure Correction: Let $\tilde{\mathcal{C}}_1$ denote a punctured version of $\mathcal{C}_1$ that keeps only the first n code bits of every codeword, and let $\tilde{\mathbf{c}} \in \tilde{\mathcal{C}}_1$ correspond to a codeword $\mathbf{c} \in \mathcal{C}_1$. Form $\mathbf{t}$ by changing $\leq n - \tilde{k}(1 + \epsilon_c)$ positions of $\tilde{\mathbf{c}}$ to $*$ symbols. With high probability, we can compute a reconstruction $\mathbf{w} = f(\mathbf{t})$ such that $|\{i : \mathbf{c}_i \neq \mathbf{w}_i \text{ and } i > n-1\}| \leq \delta_c \tilde{n}$.*

One class of codes $\mathcal{C}_1$ that can meet the conditions of Definition 2 is an LT code [14], to which we restrict our attention for the remainder of this chapter. Following the notation from [1], a $(\tilde{k}, \Omega(z))$ LT code is one with $\tilde{k}$ information bits and output degree distribution given by the generator polynomial[7] $\Omega(z)$. For our construction, we use, as in [1], a modified version of the ideal soliton distribution. Specifically, our distribution has generator polynomial

$$\Omega_{\mu,D}(z) = \frac{1}{\mu + 1}\left(\mu z + \sum_{i=2}^{D-1} \frac{z^i}{i(i-1)} + \frac{z^D}{D}\right),$$

where we have made the parameters μ and D explicit. We truncate this LT code so that only $n + \tilde{n}$ outputs are produced.

For $\mathcal{C}_2$, we require only that the code 1) be of high-rate, 2) have efficient (linear complexity) encoding and decoding algorithms, and 3) be a good error-correction code. With respect to the latter, we require that the code be capable of correcting

[7]Recall that the probability of a degree i node is specified by the coefficient of z^i in a generator polynomial $\Omega(z)$. Thus the expected degree is given in terms of this polynomial by $\Omega'(1)$.

a fixed fraction r of errors regardless of the locations of these errors in the received signal. However, we do not require that the code be capacity-achieving. As an example, one class of codes $\mathcal{C}_2$ that meets these requirements is that due to Spielman [8].

3.6 Main Theorem

We will choose the following parameters for $\mathcal{C}_1$. Let $\epsilon_s > 0$ be arbitrary, and let $\epsilon_c = (2\ln(1/\epsilon_s))^{-1/\epsilon_s}$. In turn, set the parameters of $\mathcal{C}_1$ according to $\mu = \epsilon_s/2 + (\epsilon_s/2)^2$ and $D = \lceil 1/\rho \rceil$, where $\rho = \epsilon_c/(4(1+\epsilon_c))$. Furthermore, let $\delta_s = 10/\Omega'_{\mu,D}(1) < 10/\ln(1/\epsilon_c)$, and let $\delta_c = 2(\tilde{k}/\tilde{n})\rho\Omega'_{\mu,D}(1)$. Then we have the following:

Lemma 4. *The ensemble of $(\tilde{k}, \Omega_{\mu,D}(z))$ LT codes truncated to length $n + \tilde{n}$ is an $(\epsilon_s, \epsilon_c, \delta_s, \delta_c)$-good code for $\mathcal{C}_1$.*

This $\mathcal{C}_1$ code, when combined with a suitably parameterized $\mathcal{C}_2$ code, yields an admissible coding scheme for our channel in the sense of Definition 1. Specifically, we have the following as our main result.

Theorem 3. *Suppose $\mathcal{C}_1$ is chosen as in Lemma 4, and $\mathcal{C}_2$ is capable of correcting a fraction $r = \delta_s + \delta_c$ of errors. Then for the double-erasure information embedding channel with*

$$e_s \geq 1 - \frac{\tilde{k} - \tilde{n}}{n} + \frac{\tilde{k}}{n}\epsilon_s,$$

and

$$e_c \leq 1 - \frac{\tilde{k}}{n}\left(1 + \epsilon_c + \sqrt{\frac{160}{(1-e_c)\ln(1/\epsilon_c)}}\right),$$

our construction produces an admissible coding scheme with rate $\limsup k/n$.

Our proof of Theorem 3 is actually a bit more general, in that our coding scheme can handle other channel erasure models. We will discuss this more in Section 3.8. It follows immediately from Theorem 3 that our coding scheme is able to achieve rates close to $e_s - e_c$, i.e., close to capacity. Specifically, we have:

Corollary 1. *For a double-erasure information embedding channel with parameters* e_s *and* e_c*, we can choose* $k, \tilde{k}, n, \tilde{n}$ *to obtain an admissible coding scheme with rate arbitrarily close to* $e_s - e_c$.

3.7 Proofs

In this section, we prove Lemma 4 and Theorem 3.

3.7.1 Proof of Lemma 4

To prove Lemma 4, we verify the erasure quantization and erasure correction properties separately.

The erasure quantization property is so named because it requires the code ensemble to be good for BEQ. To prove that $(k, \Omega_{\mu,D}(x))$ LT codes satisfy the erasure quantization property, we will use lemma 1.

Lemma 5. *For any* $\epsilon_s > 0$*, let* ϵ_c*,* μ*, and* D *be defined as in Section 3.6 . Then, a* $(\tilde{k}, \Omega_{\mu,D}(x))$ *LT code truncated to produce* n *output symbols can match all but a* δ_s *fraction of any subset of* $\tilde{k}(1-\epsilon_s)+1$ *unerased output symbols with high probability.*

Proof. From Lemma 1, it follows that to prove Lemma 5 we only need to show that the dual of a truncated $(\tilde{k}, \Omega_{\mu,D}(x))$ LT code is good on the BEC. [8] More precisely, we must show that if all the inputs to the dual code are erased, we can recover all but a δ_s fraction of the erased symbols.

The analysis of the dual code is similar to the proof of [1, Lemma 4]. Let $\omega(x)$ and $\ell(x)$ be the generating functions for the *edge* degree distributions with respect to the variable and check nodes of the dual LT code. From [1], $\omega(x) = \frac{\Omega'_{\mu,D}(x)}{\Omega'_{\mu,D}(1)}$, and $\ell(x) = (1 - \frac{\Omega'_{\mu,D}(1)(1-x)}{\tilde{k}})^{\tilde{k}(1-\epsilon_s)}$. Using the density evolution method, to prove Lemma 5 we must show that $\omega(1-\ell(1-x)) < x, \forall x \in [\delta_s, 1]$. (The only difference between

[8] In Lemma 1, "recover" includes the case where BP decoding can only determine some of the information bits. For a particular erasure sequence, suppose BP decoding can recover l information bits. Then, the dualized form of BP applied to $\mathcal{C}^{\perp}$ and the dual sequence can quantize the dual sequence such that at least l unerased positions are matched.

our argument and the argument in [1] is that in [1] the author proves $\ell(1-\omega(1-x)) < x, \forall x \in [\delta_s, 1]$, i.e., our argument proves that we can interchange ω and ℓ and preserve the inequality.)

Now, $\omega(1-\ell(1-x)) < x$ reduces to $\Omega'(1-\ell(1-x)) < \Omega'(1)x$. Using the formula for $\Omega(x)$ given in [1], we obtain

$$\begin{aligned}
\Omega'(1-\ell(1-x)) &= \frac{1}{\mu+1}(\mu - \ln(1-\ell(1-x)) + (1-\ell(1-x))^D - \sum_{d=D+1}^{\infty} \frac{(1-\ell(1-x))^d}{d}) \\
&\leq \frac{1}{\mu+1}(\mu + \tilde{k}(1-\epsilon_s)\ln \frac{1}{1-\frac{\Omega'(1)x}{\tilde{k}}} + (1-\ell(1-x))^D) \\
&\leq \mu + \tilde{k}(1-\epsilon_s)\frac{\frac{\Omega'(1)x}{\tilde{k}}}{1-\frac{\Omega'(1)x}{1-\frac{\Omega'(1)x}{\tilde{k}}}} + (1-\ell(1-x))^D.
\end{aligned}$$

Now, choose $\tilde{k}$ so that $\tilde{k} > \frac{10\Omega'(1)}{\epsilon_s}$. This implies that $\frac{1}{1-\frac{\Omega'(1)x}{n}} \leq 1 + \frac{\epsilon_s}{5} \forall x \in [0,1]$. So, we get

$$\begin{aligned}
\Omega'(1-\ell(1-x)) &\leq \mu + \Omega'(1)(1-\epsilon_s)(1+\frac{\epsilon_s}{5})x + (1-\ell(1-x))^D \\
&\leq \mu + \Omega'(1)x - \frac{3\epsilon_s}{10}\Omega'(1)x + (1-\ell(1-x))^D.
\end{aligned}$$

If $x > \frac{5\mu}{\Omega'(1)\epsilon_s}$, then the RHS is $\leq \Omega'(1)x - \frac{\epsilon_s}{10}\Omega'(1)x + (1-\ell(1-x))^D$. So, to complete the proof, we must show $(1-\ell(1-x))^D < \frac{\epsilon_s}{10}\Omega'(1)x$.

$$\begin{aligned}
(1-\ell(1-x))^D &\leq e^{-D\ell(1-x)} \\
&\leq e^{-De^{-\frac{\Omega'(1)x(1-\epsilon_s)}{1-\frac{\Omega'(1)x}{k}}}}
\end{aligned}$$

because $\ell(1-x) > e^{-\frac{\Omega'(1)x(1-\epsilon_s)}{1-\frac{\Omega'(1)x}{k}}}$, using the inequality $1-x > e^{-\frac{x}{1-x}}$. So, taking logarithms twice, we find

$$x < \frac{\ln D - \ln\ln\frac{10}{\epsilon_s\Omega'(1)x}}{\Omega'(1)(1-\epsilon_s)}.$$

We want this inequality to be true for the interval $[\delta_s, 1]$. If $\delta_s = \frac{10}{\Omega'(1)}$ then

$$\frac{\ln D - \ln\ln \frac{10}{\epsilon_s \Omega'(1) x}}{\Omega'(1)(1-\epsilon_s)} \geq \frac{\ln D - \ln\ln \frac{1}{\epsilon_s}}{\Omega'(1)(1-\epsilon_s)},$$

so it suffices to show

$$x < \frac{\ln D - \ln\ln \frac{1}{\epsilon_s}}{\Omega'(1)(1-\epsilon_s)},$$

which in turn is true iff it is true at $x = 1$. This reduces to

$$1 - \epsilon_s < \frac{\ln D - \ln\ln \frac{1}{\epsilon_s}}{\Omega'(1)}.$$

Now, $\ln D = \Omega'(1) - O(1)$, so the RHS

$$= 1 - \frac{\ln\ln \frac{1}{\epsilon_s} + O(1)}{\Omega'(1)},$$

so we need to choose $\Omega'(1)$ so that

$$\frac{\ln\ln \frac{1}{\epsilon_s} + O(1)}{\Omega'(1)} < \epsilon_s.$$

$\Omega'(1) = \ln \frac{1}{\epsilon_c} + O(1)$, so we find

$$\epsilon_c < (\ln \frac{1}{\epsilon_s} + O(1))^{-\frac{1}{\epsilon_s}}.$$

This proves the lemma. □

Lemma 5 shows that $(\tilde{k}, \Omega_{\mu,D}(x))$ LT codes satisfy the erasure quantization property because the LT code generates every output bit independently and identically. Thus, the unmatched positions are evenly distributed throughout the n output positions, and we can consider the first n positions separately from the last $\tilde{n}$ positions. (In fact, since all the erasures are in the first n positions, the fraction that are incorrect in the first n positions is upper bounded by $\delta_s(1 - e_s)$).

The erasure correction property follows from the following lemma, given in [1, Lemma 4].

Lemma 6. *For any $\epsilon_s > 0$, let ϵ_c, μ, D, and ρ be as defined in Section 3.6. Then, the LT code with distribution $\Omega_{\mu,D}(x)$ can recover all but a ρ fraction of the $\tilde{k}$ inputs from any subset of $\tilde{k}(1+\frac{\epsilon_c}{2})+1$ output symbols with high probability.*

Proof of Lemma 4. Lemma 5 proves the erasure quantization property. To complete the proof of Lemma 4, we need to turn the bound on the number of unrecovered inputs given in Lemma 6 into a bound on the number of unrecovered outputs in the last $\tilde{n}$ positions. With high probability at most $2\tilde{k}\rho$ variable nodes of $\mathcal{C}_1$ are unrecovered (we need the 2 for Lemma 8 to come later). These unrecovered variable nodes induce at most $2\tilde{k}\rho\Omega'_{\mu,D}(1)$ unrecovered check nodes in the last $\tilde{n}$ positions of $\mathcal{C}_1$ with high probability. This is because the number of unrecovered check nodes in the last $\tilde{n}$ positions is upper bounded by $\sum_i \deg(i)$, where $\deg(i)$ is the degree of node i in the subgraph induced by the last $\tilde{n}$ check nodes of $\mathcal{C}_1$, and the sum ranges over all the variables nodes that are in error. Because the check nodes choose their neighbors independently at random, this sum is tightly concentrated around its expected value, which is $\Omega'_{\mu,D}(1)\tilde{n}\rho$. Thus, with high probability we do not see more than $2\tilde{k}\rho\Omega'_{\mu,D}(1)$ unrecovered check nodes in the last $\tilde{n}$ positions.[9] Note that this analysis would hold even if we made errors in the variable nodes instead of just not recovering certain nodes. This is important when we prove Lemma 8 to come. □

Note that the proofs of Lemmas 5 and 6 show that BP can be used to find the strings guaranteed by the good code property.

3.7.2 Proof of Theorem 3

We prove, in order, that our construction satisfies both the encoding and decoding properties of an admissible coding scheme.

The former is established by the following Lemma.

[9]We assume $\tilde{k} \geq \tilde{n}$. This is reasonable because the proof of corollary 1 sets $\tilde{k} \approx n(1-e_c)$, and $\tilde{n} \approx n(e_s - e_c)$.

Lemma 7. *For the choices of $\mathcal{C}_1$, $\mathcal{C}_2$, state, and channel distributions given in Theorem 3, our construction satisfies the encoding property of Definition 1.*

Proof. The encoding algorithm given in Section 3.5 guarantees that we satisfy the encoding property, since step 3 of the algorithm ensures that the encoding matches s at all non-$*$ positions. However, we can make a stronger statement than this. Lemma 4 guarantees that a large fraction of the state positions are matched after step 2 on the encoding algorithm. Thus, step 3 only changes a small (δ_s) fraction of unmatched positions to get the final encoding. This will be important when we analyze the decoder. □

In the sequel, we refer to the encoding computed after step 2 as the *preliminary encoding.*

Now we prove that our construction satisfies the decoding property. Specifically, we have the following result, whose proof requires us to show that our code can correct the erasures made by the channel, and the errors introduced by the preliminary encoding.

Lemma 8. *For the choices of $\mathcal{C}_1$, $\mathcal{C}_2$, state, and channel distributions given in Theorem 3, our construction satisfies the decoding properties of Definition 1.*

To prove Lemma 8, we first need the following Lemma, which implies that a truncated version of $\mathcal{C}_1$ can be decoded reliably over $\text{BSC}(\frac{\delta_s}{1-e_c})$. [10]

Lemma 9. *For the choice of parameters given in Theorem 3, assume that the first n bits of a codeword of $\mathcal{C}_1$ are sent over $\text{BSC}(\frac{\delta_s}{1-e_c})$, and then over the erasure channel specified in Theorem 3. Then, BP can be used to recover the variable nodes of C_1 with high probability, in the sense that at most a fraction ρ of the nodes are not recovered correctly.*

[10]The reason we divide by $(1 - e_c)$ is that although at most $\delta_s n$ errors are introduced by the encoder, the channel only erases positions which were not in error. This is because the channel only erases positions which were erased by the state. This means that there are $\delta_s n$ errors in only $(1 - e_c)n$ received positions, so the relevant fraction of errors is $\frac{\delta_s}{1-e_c}$.

In order to prove Lemma 9, we will make use of the following result from [4, Thm. 4.2] relates the performance of a code on the BEC to its performance on any binary input-symmetric channel (BISC). The Bhattacharya parameter of a BISC is defined as $\lambda = E[e^{-L/2}]$, where L is the log-likelihood ratio of the input given the channel output.

Lemma 10. *Let $\lambda(\mathcal{C})$ be the Bhattacharya parameter of an arbitrary BISC $\mathcal{C}$. If BP can decode an ensemble of codes over* $\mathrm{BEC}(\lambda(\mathcal{C}))$, *then BP can also decode reliably over $\mathcal{C}$.*

We remark that the proof of Lemma 10 given in [4] actually proves the stronger statement that if the fraction of unrecovered inputs over $\mathrm{BEC}(\lambda(\mathcal{C})) < \delta$, then the fraction of inputs which are recovered incorrectly over $\mathcal{C}$ is also less than δ. [11]

Proof of Lemma 9. We prove that the subgraph of $\mathcal{C}_1$ formed by only considering the positions that were not erased by the erasure channel is good for $\mathrm{BSC}(\frac{\delta_s}{1-e_c})$. Let $\tilde{\epsilon} > 0$ be a parameter we determine later. Say we receive $\tilde{k}(1+\tilde{\epsilon})$ bits, but a $\frac{\delta_s}{1-e_c}$ fraction of these bits are incorrect. From Lemma 6, we know that this subcode of $\mathcal{C}_1$ can recover from erasures provided that $\tilde{k}(1+\epsilon_c/2)$ unerased outputs are available. Thus, we can tolerate an erasure probability of $\tilde{e} = (\tilde{\epsilon}-\epsilon_c)/(1+\tilde{\epsilon})$. Applying Lemma 10, it follows that if δ_s satisfies $\lambda(\frac{\delta_s}{1-e_c}) = 2\sqrt{\frac{\delta_s}{1-e_c}(1-\frac{\delta_s}{1-e_c})} < \tilde{e}$, then we can correct a $\frac{\delta_s}{1-e_c}$ fraction of errors. This inequality is satisfied if we choose $\tilde{\epsilon} = \epsilon_c + \sqrt{160/((1-e_c)\ln(1/\epsilon_c))}$. The reader can verify that the choices of Theorem 3 give at least this value of $\tilde{\epsilon}$. □

It remains to confirm that $\mathcal{C}_1$ can correct enough of the errors from the preliminary encoding that $\mathcal{C}_2$ can correct those that remain.

Proof of Lemma 8. We first define a new channel $\mathcal{C}_e$, which models the positions whose bits we need to change after the preliminary encoding in order to satisfy the encoding property. Let Γ be the fraction of unmatched positions after the preliminary

[11]Density evolution typically looks at the values passed along edges of the graph. To turn this into a bound on inputs, it suffices to pretend that each variable nodes has an "extra" edge leaving which is not attached to any other nodes. The value of this edge is updated using the same density evolution equations, and the value on this edge determines the value of the associated variable.

encoding, so that $\mathcal{C}_e$ introduces Γn errors to form the final encoding. Because the LT code generates each output symmetrically, and because the state distribution (3.2) is symmetric, it follows that given $\Gamma = \gamma$, the γ positions flipped by $\mathcal{C}_e$ are equally likely to be any γ positions. Thus, conditioned on the number of errors, $\mathcal{C}_e$ has the same distribution as a BSC.

Lemma 4 guarantees that $\Gamma \leq \frac{\delta_s}{1-e_c}$ with high probability. Let $\overline{\gamma}$ be the expected value of Γ (our proof of Lemma 5 shows $\overline{\gamma} \leq \delta_s$, but we can calculate $\overline{\gamma}$ to any desired accuracy using density evolution). Then, a standard martingale argument [7] shows that there exists a constant β such that for any $\epsilon > 0$,

$$\Pr[|\Gamma - \overline{\gamma}| > \epsilon] < e^{-\beta\epsilon^2 n}.^{12} \tag{3.4}$$

Let $D(\cdot\|\cdot)$ denote the Kullback-Leibler distance between two Bernoulli random variables. For any particular value γ and sufficiently large n, we know that the probability $P_{\gamma,\epsilon}$ that a realization BSC(γ) of the BSC makes, for some $\epsilon > 0$, a fraction $\gamma + \epsilon$ errors in n transmissions is lower bounded by

$$P_{\gamma,\epsilon} \geq \frac{e^{-nD(\gamma+\epsilon\|\gamma)}}{\sqrt{2\pi n}}, \tag{3.5}$$

and a similar statement is true for $\gamma - \epsilon$.

We say that BP decoding of (a truncated version) of $\mathcal{C}_1$ fails if the fraction of level 1 variable nodes that are not recovered correctly is greater than 2ρ. Using martingale arguments, one can show that Lemma 9 implies

$$\Pr[\text{BP decoding fails for BSC}(\delta_s)] \leq e^{-\alpha\rho^2 n}, \tag{3.6}$$

[12] In deriving equation 3.4, it is important that the encoder uses the algorithm specified in Section 3.5. Specifically, the unreserved variables need to be assigned randomly. This allows us to conclude that about half of the unreserved checks are not satisfied. Then, we can multiply the density evolution value for the number of unreserved checks by .5 to get $\overline{\gamma}$. The concentration result in equation 3.4 follows easily by combining the facts that the fraction of erasures introduced by the channel concentrates around e_c, the number of unreserved checks concentrates around the density evolution value, and the number of errors introduced by the encoder concentrates around .5*the density evolution value since we assign the unreserved variables randomly.

for a suitable constant $\alpha > 0$. Choosing ϵ such that $\sigma = D(\overline{\gamma} \pm \epsilon || \overline{\gamma}) - \alpha \rho^2 < 0$, then combining (3.4), (3.5), (3.6), and the fact that conditioned on the number of errors, the distribution of a BSC and $\mathcal{C}_e$ is the same, we obtain

$$\Pr[\text{BP decoding fails for } \mathcal{C}_e] \leq e^{-\sigma n}\sqrt{2\pi n} + e^{-\beta\epsilon^2 n}.$$

Thus, the probability that BP decoding of $\mathcal{C}_1$ fails is exponentially small even when the errors are introduced by $\mathcal{C}_e$.

To complete the proof, we need to correct the small fraction of check nodes which are not recovered properly after decoding $\mathcal{C}_1$. There are two sources of error for the last $\tilde{n}$ check nodes: errors caused because our definition of failure allows for a 2ρ fraction of errors in the variable nodes, and errors caused because during encoding we may not be able to match a δ_s fraction of the check nodes. In total, with high probability the two sources of error induce at most a fraction $\delta_s + \delta_c$ of errors in the last $\tilde{n}$ check nodes, which can be corrected given the choice of $\mathcal{C}_2$. Note that we have not shown that the errors introduced to the level 2 variable nodes occur independently for each variable. This is not necessary since we assume that C_2 can correct a small fraction of errors in the worst case, so the probabilistic model for the errors is not important. $\square$

We have proved that our construction has the encoding and decoding properties, so this proves Theorem 3. Encoding and decoding take time $O(n + \tilde{n})$. The rate of our ensemble is $e_s - e_c - O(\sqrt{\epsilon_s})$. However, because ϵ_c is so small compared to ϵ_s, the number of edges in the graph in figure 3-5, and hence the complexity of encoding and decoding, scales with $\ln(\frac{1}{\epsilon_c}) = O(\frac{1}{\epsilon_s}\ln\ln(\frac{1}{\epsilon_s}))$. Thus, our choice of $\Omega_{\mu,D}(x)$ allows us to prove the asymptotic result, but the dependence on ϵ_s makes it difficult to get close to $e_s - e_c$ with this scheme. However, in section 3.10, we will see that the performance can be improved by finding choices for $\Omega(x)$ with a lower value of $\Omega'(1)$, which still perform well simultaneously for the BEQ and BEC.

3.8 Information Embedding Capacity

As mentioned earlier, our coding scheme can be used for certain other channel erasure models. In this section we analyze one such model, and then discuss some extensions of Theorem 3.

Consider the information embedding problem where the state is still drawn from (3.2). The channel model is no longer restricted to only erase positions which were unerased in the state. Instead, we take the channel as a BEC with parameter e_c. Thus, the channel still introduces an e_c fraction of erasures, but these positions are evenly distributed now, as opposed to being concentrated on the positions where the state was erased.

To calculate the capacity for this new channel model, we use the Gel'fand-Pinkser [10] formula. This formula gives the channel capacity as

$$C = \max_{p(u|s),p(x|u,s)} I(U;Y) - I(U;S),$$

where U is an additional random variable. We can solve this optimization problem numerically using a version of the Blahut-Arimoto algorithm [9]. However, we can determine the capacity analytically for this particular channel.

Lemma 11. *Assume that the state is drawn from (3.2), and that the channel is a BEC with erasure probability e_c. Let $h_b(\omega)$ be the binary entropy function. Then, the capacity of the information embedding problem is*

$$C = e_s - e_c + (1 - e_s)\, h_b(\omega^*) - (1 - e_c)\, h_b(\omega^*(1 - e_s)),$$

where ω^* satisfies $\frac{1-\omega^*}{\omega^*} = \left(\frac{1-\omega^*(1-e_s)}{\omega^*(1-e_s)}\right)^{(1-e_c)}$.

We give a simple method that obtains a lower bound to the capacity. Appendix A uses a brute-force approach to prove that the lower bound is actually tight. Consider forming the random variable U in the Gel'fand-Pinkser formula as follows. With probability 2ω, set U to be 0 or 1 with equal probability, independent of S. With probability $1 - 2\omega$, set $U = S$ if $S \in \{0, 1\}$, and set U to be 0 or 1 with equal probability if $S = *$. Take $X = U$. Then, optimizing over ω gives a lower bound of

$$C_- = e_s - e_c + (1 - e_s)h_b(\omega^*) - (1 - e_c)h_b(\omega^*(1 - e_s)),$$

where ω^* satisfies $\frac{1-\omega^*}{\omega^*} = (\frac{1-\omega^*(1-e_s)}{\omega^*(1-e_s)})^{(1-e_c)}$. Thus, Lemma 11 shows that this simple choice of U and X gives the correct formula for capacity, i.e., $C_- = C$.

Figure 3-7 shows the capacity for several choices of e_s and e_c.

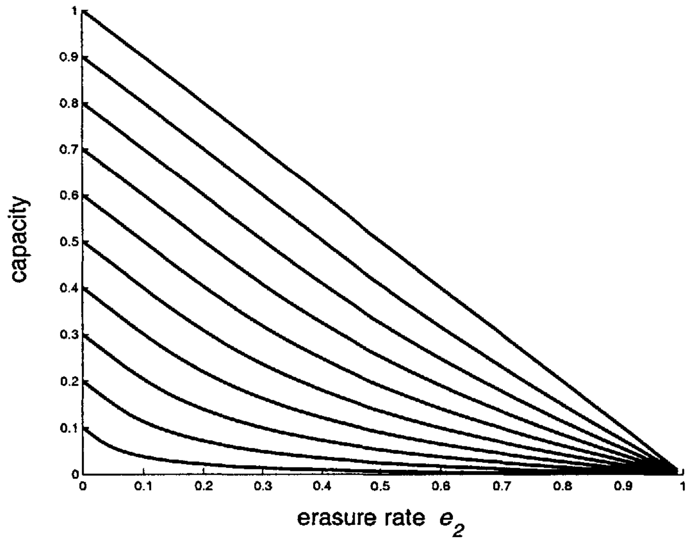

Figure 3-7: Capacity for Information Embedding Problem

3.9 Remarks

1. The proof of Theorem 3 does not require much knowledge of the statistical model for the channel. In particular, no matter how the channel erasures are placed, there will be less than $\delta_s n$ errors with high probability. There will also be about $n(1 - e_c)$ received symbols with high probability. Thus, the fraction of errors is concentrated in the interval $[0, \frac{\delta_s}{1-e_c}]$ with high probability, since the worst kind of erasure noise would be noise that avoided the mismatched symbols

entirely, i.e., the channel given by 3.3.

Based on this fact, we can show that the result of Theorem 3 holds even for the new channel considered in this section. To see this, note that our new channel introduces the same fraction of erasures as the old channel. The only other property we need from the channel is that the fraction of errors seen at the decoder is concentrated around some value. This is true for our new model, so Theorem 3 still holds. (Basically, since the new channel model is a BEC, every bit is equally likely to be erased, which means we do not need to change δ_s to $\frac{\delta_s}{1-e_c}$ in the proof of lemma 8 anymore.) Note that although our construction works for the new channel, it does not achieve capacity. From figure 3-7, we see that when e_c is small compared to e_s, the capacity is very closely approximated by $e_s - e_c$. Thus, our coding scheme does come close to capacity for a reasonable range of values for e_s and e_c.

2. The capacity-achieving distribution for U described after Lemma 11 essentially quantizes the state under a Hamming distortion. Going through a random coding argument shows that U achieves capacity because the associated random code has the property that it is simultaneously good for the BSC and for quantization under a Hamming distortion. Thus, it seems that the new channel model is just as hard as the original coding for a memory with defects problem, even though the channel is an erasure channel rather than a BSC. It is interesting that even though the state and channel models only introduce erasures, quantization under a Hamming distortion and error correction have come into play under our new channel model.

3. Note that Lemma 5 does not assume anything about the statistical model of the erasures in the state, i.e., Lemma 5 says that LT codes can match the state provided the right number of erasures occur, regardless of how the erasures are placed. Lemma 6 gives a similar result for channel decoding. Thus, it is natural to wonder why Theorem 3 needs to assume that the state sequence is generated from the distribution given by (3.2). Recall that in the proof of Theorem 3,

we modelled the unmatched check nodes as errors from an additional channel. When the state is generated from (3.2), we can use symmetry to argue that every check node is equally likely to be unmatched. This would not be true for an arbitrary state distribution. We could prove that Theorem 3 is true for any state distribution if we could make a statement of the form, "If BP works over BSC(δ), then BP also works over a channel where different bits are flipped with different probabilities $\leq \delta$." Intuitively, it seems like this statement should be true, since we are essentially switching between BSCs with different parameters; however, we have not tried to prove this statement rigorously.

4. We note that the LT code analyzed in section 3.7 is in some sense optimal. Specifically, the LT code satisfies $\omega(1-\ell(1-x)) < x, \forall x \in [\delta_s, 1]$ and $\ell_1(1-\omega(1-x)) < x \forall x \in [\rho, 1]$ for arbitrarily small δ_s, ρ. Here, $\ell_1(x)$ is the generating function for the edge degree distribution of the variable nodes of C_1 when we only look at the subgraph induced by the unerased check nodes received over a BEC. Thus, $\ell(x)$ and $\ell_1(x)$ are only different in that the exponent in the distribution changes by ϵn. Now, using a message-passing decoder, no LT code can satisfy the BEQ and BEC constraints better than our choice $\Omega_{\mu,D}(x)$. This is because the BEC constraints imply that $\omega(0) > 0$, while the BEQ constraints imply $\omega(0) = 0$ if we want the BEQ constraints to hold on $(0, 1]$. Similarly, the BEC constraints imply that $\ell_1(0) = 0$, and the BEQ constraints imply $\ell(0) > 0$. Because the only difference between $\ell(x)$ and $\ell_1(x)$ is the exponent, we see that it is impossible to satisfy the constraints better than the LT code above without introducing some fixed points into the decoder. The problem basically boils down to the fact that if we want BEQ (or BEC) to work all the way down to 0, then there can't be degree 1 check (or variable) nodes, and this will cause the decoder for the other problem to get stuck and not even start. Of course, this simple analysis says nothing about the rate of convergence in δ_s, ρ to 0, which could be improved greatly by using a different code. To this end, we note that in [1], it is shown that the dependence between ρ and $\Omega'_{\mu,D}(1)$ is nearly

optimal for the BEC. Thus, the source of our bad dependence between encoding and decoding complexity and the gap to capacity is the slow convergence of δ_s to 0. The rate of convergence can probably be drastically improved, especially in light of the fact that there exist constant-degree LDGM codes which achieve capacity for BEQ.

5. Related to the last remark, consider the inequalities $\omega(1-\ell(1-x)) < x, \forall x \in [\delta_s, 1]$ and $\ell_1(1-\omega(1-x)) < x \forall x \in [\rho, 1]$. By inverting the first inequality, we see that intuitively, a capacity-achieving ensemble for the BEC, where $\omega(1-\ell(1-x)) \approx x$, will probably satisfy the dual condition. This "symmetry" is discussed in more detail in [22]. Thus, it seems that most capacity-achieving codes for the BEC should work as a C_1 code in our construction, provided that we use random puncturing techniques to make the information/code bits symmetric. Also, note that even though we haven't drawn C_1 this way in the figure, really C_1 can be thought of as an LDPC-GM code, i.e., C_1 has the same structure as the compound code considered by [23], [24] (Imagine drawing the last $\tilde{n}$ nodes of C_1 on the bottom. This looks like an LDPC precode).

3.10 Some Notes On Practical Schemes For Information Embedding

In this section we make a few observations that may be relevant for actually implementing the codes describes earlier. First we examine the case considered in Section 3.4, where $e_c = 0$. Figure 3-8 shows a code construction based on Raptor codes.

Essentially the code in figure 3-8 is formed by taking the dual of the LT part of a Raptor code. This looks like a parity check code. Now, instead of dualizing the precode, we keep the precode the same. More precisely, we use a low density parity check (LDPC) [13] precode capable of correcting a small fraction of errors. The details are similar to the construction based on nonsystematic IRA codes given in Section 3.4, so we will be brief. To encode a message, we first encode the message

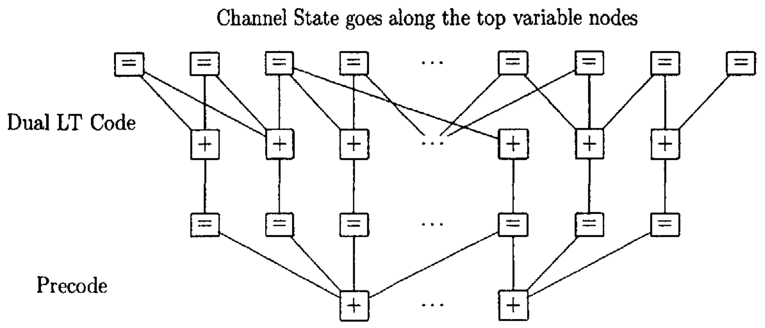

Figure 3-8: Code Based on Raptor Codes - Another capacity-achieving construction for the case $e_c = 0$. The variable nodes in the middle represent the message after it has been encoded with the precode.

with the precode. Then, we use the dualized form of BP to assign values to the variable nodes of the dual LT code. From Lemmas 1 and 6, we know that this algorithm will compute an assignment to the erased state nodes which satisfies all but a small linear fraction of the check nodes. So, the decoder sees the encoded state, calculates the values at the check nodes, and corrects this small fraction of errors using BP or some other algorithm capable of correcting a small fraction of errors. This scheme has the advantage that it is easily proved to work for all values of e_s, while the dual IRA code is only conjectured to work for all values. Also, experiments need to be carried out to determine which scheme would work better for short block lengths. We know that for long block lengths the dual IRA codes are good since they have bounded complexity, but in [5] the authors find that for short block lengths other codes perform better than nonsystematic IRA codes. However, the construction based on Raptor codes has the drawback that we need to use an error-correcting code rather than an erasure-correcting code. This makes it harder to have a small gap to capacity.

Now we deal with the case $e_c \geq 0$. To design distributions $\Omega(x)$ for C_1 that exhibit better performance than the choice $\Omega_{\mu,D}$ used to prove Theorem 3, we can use many of the same design techniques outlined in [1]. From [1], we know that the following constraint works well for finding good distributions for the BEC:

$$\Omega'(x) \geq \frac{-\ln\left(1 - x - c\sqrt{\frac{1-x}{n(e_s - e_c)}}\right)}{1 + \frac{\epsilon_c}{2}},$$

where c is some constant. We discretize this inequality by evaluating it at many points in $[0, 1 - \rho]$. Each of these evaluations gives a linear constraint on the unknown coefficients of $\Omega(x)$.

We need to add constraints that will make $\Omega(x)$ good for BEQ. Recall from Lemma 5 that the condition for quantization success is $\omega(1 - \ell(1 - x)) < x, \forall x \in [\delta_s, 1]$. Going through the same derivation as in [1], we find that

$$\Omega'(x) \leq \frac{-\ln\left(1 - x + c\sqrt{\frac{1-x}{n(e_s - e_c)}}\right)}{1 - \epsilon_s}, \forall x \in [\Omega'(1)\delta_s, 1]$$

should be a good constraint on $\Omega(x)$ to ensure good performance at BEQ. We can discretize this inequality, and add the resulting constraints to get the final LP. We note that to get close to capacity, it is crucial that δ_s, ρ be very small. This is because the fraction of errors C_2 needs to correct scales with these parameters, and the rate of C_2 will be $< 1 - \Omega(\sqrt{\text{fraction of errors corrected}})$. This rate loss will be the dominant source of the gap to capacity for reasonable choices of ϵ_s and ϵ_c.

Chapter 4

Conclusion

This thesis has examined the problems of lossy source coding and information embedding. For lossy source coding, we proved several results showing that LDPC codes are effective for this problem, at least if we allow optimal (high complexity) encoding algorithms. We also saw that LDGM codes cannot achieve constant graphical complexity per bit, unlike the compound codes in [24].

For information embedding, we developed a coding scheme with efficient algorithms for encoding and decoding over the double erasure information embedding channel. The key ingredient in our construction was an LT code that was simultaneously optimal for BEQ and BEC. We also showed that changing the erasure model creates an information embedding channel that is essentially as hard as coding for a memory with defects. Our code construction, although not capacity-achieving, still works for the new model.

The results of Chapter 2 leave many questions unanswered. Our goal in studying lossy source codes was to develop codes for the binary symmetric source under Hamming distortion with efficient algorithms for encoding and decoding. While we have shown that LDPC codes should be considered along with LDGM codes and compound constructions, our proof techniques rely on optimal encoding algorithms, so the obvious question is whether any of these sparse graph codes has an efficient encoding algorithm. Towards this end, we note that density evolution shows that standard BP cannot work as an encoding algorithm, but there is evidence that BP can be modified to work for lossy source coding [3,27].

A simpler problem mentioned earlier is to determine whether the correspondence between error-correcting codes and quantizers proved in Section 2.3 goes the other way. Also, a problem suggested by our result in Section 2.4 is to prove a similar lower bound on the degree of either the variable nodes or the check nodes of good LDPC codes. Based on the fact that LDPC codes with bounded degree cannot come arbitrarily close to capacity for the BSC, we would be surprised if LDPC codes with bounded degree could come arbitrarily close to the rate-distortion function for a binary symmetric source under Hamming distortion. A lower bound for LDPC codes would be important conceptually because this would show that the compound construction of [24] somehow greatly improves over using LDGM codes or LDPC codes separately.

For information embedding, the natural problem to consider is how to generalize our construction to other source and channel models. In particular, a better C_1 code is needed in order to achieve capacity for the channel considered in Section 3.8. It is easily shown that if C_1 is simultaneously good for the BSC and for quantization under a Hamming distortion, then our construction can come arbitrarily close to capacity for the channel in Section 3.8. Therefore, if we proceed along these lines, finding a good lossy source code is a prerequisite for finding a C_1 code that is good enough for our construction. For example, taking C_1 to be a random linear code allows us to achieve capacity for arbitrary e_s and e_c, but practical algorithms generally do not work for random linear codes. It would be interesting to see if another code construction is possible which bypasses the need for such strong properties from the code. We believe that this is probably not possible. Intuitively, it seems that modifying the encoder for a good code for coding for a memory with defects ought to produce a good lossy source code, i.e., we should be able to prove a reduction from coding for a memory with defects to lossy source coding.

Appendix A

Proof of Lemma 11

This appendix contains a proof of Lemma 11.

To prove that the formula for capacity is correct, consider the following equations given in [9]:

$$q(t|s) = \frac{\prod_y Q(t|y)^{p(y|x=t(s),s)}}{\sum_t \prod_y Q(t|y)^{p(y|x=t(s),s)}}$$

$$Q(t|y) = \frac{\sum_s p(s)q(t|s)p(y|x=t(s),s)}{\sum_{s,t'} p(s)q^*(t'|s)p(y|x=t'(s),s)}.$$

Here t ranges over all functions from $\{0,1,*\} \to \{0,1\}$, and $p(y|x = t(s), s)$ is the distribution of the channel conditioned on the state s and the input x. The fixed point of these equations defines the optimal distributions $q^*(t|s)$ and $Q^*(t|y)$, which can be used to calculate the capacity. We will show that the following choices for q^* and Q^* are a fixed point:

$$q^*(t|s) = \frac{1-\omega^*}{4}\delta(s-t(*)) + \frac{\omega^*}{4}\delta(s-1+t(*)) + \frac{1}{8}\delta(s-*)$$

$$Q^*(t|y) = \frac{1-(1-e_s)\omega^*}{4}\delta(y-t(*)) + \frac{(1-e_s)\omega^*}{4}\delta(y-1+t(*)) + \frac{1}{8}\delta(y-*)$$

To show that these choices are a fixed point, we will start by verifying that $q^*(t|s)$ can be calculated from $Q^*(t|y)$. We break down the verification by s. First, consider the case $s = 0$. Then, if $t(*) = 0$,

$$\begin{aligned}\prod_y Q(t|y)^{p(y|x=t(s),s)} &= \left(\frac{1-(1-e_s)\omega^*}{4}\right)^{1-e_c}\left(\frac{(1-e_s)\omega^*}{4}\right)^0\left(\frac{1}{8}\right)^{e_c}\\ &= \left(\frac{1-(1-e_s)\omega^*}{4}\right)^{1-e_c}\left(\frac{1}{8}\right)^{e_c}.\end{aligned}$$

Similarly, if $t(*) = 1$,

$$\prod_y Q(t|y)^{p(y|x=t(s),s)} = \left(\frac{(1-e_s)\omega^*}{4}\right)^{1-e_c}\left(\frac{1}{8}\right)^{e_c}.$$

There are four functions with $t(*) = 0$ and four functions with $t(*) = 1$. So, if $t(*) = 0$,

$$\frac{\prod_y Q^*(t|y)^{p(y|x=t(s),s)}}{\sum_t \prod_y Q^*(t|y)^{p(y|x=t(s),s)}}$$

$$\begin{aligned}&= \frac{1}{4}\frac{(1-(1-e_s)\omega^*)^{1-e_c}}{(1-(1-e_s)\omega^*)^{1-e_c}+((1-e_s)\omega^*)^{1-e_c}}\\ &= \frac{1}{4}\frac{\left(\frac{1-\omega^*(1-e_s)}{\omega^*(1-e_s)}\right)^{(1-e_c)}}{\left(\frac{1-\omega^*(1-e_s)}{\omega^*(1-e_s)}\right)^{(1-e_c)}-1}\\ &= \frac{1}{4}\frac{\frac{1-\omega^*}{\omega*}}{\frac{1-\omega*}{\omega*}-1}\\ &= \frac{1-\omega*}{4}\\ &= q^*(t|s),\end{aligned}$$

where we have used the fact that $\frac{1-\omega^*}{\omega^*} = \left(\frac{1-\omega^*(1-e_s)}{\omega^*(1-e_s)}\right)^{(1-e_c)}$. Thus, we have verified the case $s = 0, t(*) = 0$. The case $s = 0, t(*) = 1$ follows immediately because our calculations imply that the probability in this case must be $\frac{1}{4}-$ the probability of the case $s = 0, t(*) = 0$.

The case $s = 1$ also follows easily from our analysis so far because essentially all that changes is that we flip the $t(*) = 0$ and $t(*) = 1$ formulas for $\prod_y Q(t|y)^{p(y|x=t(s),s)}$. This flip carries throughout the rest of the derivation, so we have verified the case

$s = 1$. Finally, for $s = *$,

$$\prod_y Q(t|y)^{p(y|x=t(s),s)} = \left(\frac{1-(1-e_s)\omega^*}{4}\right)^{1-e_c} \left(\frac{1}{8}\right)^{e_c}$$

for all t. Thus, normalizing will give $q^*(t|s) = \frac{1}{8}$ when $s = *$, so we have verified the first fixed point condition.

Now, we verify the second condition. Again, we will break the analysis down by cases. First, consider the case $y = 0$. Then, if $t(*) = 0$,

$$\begin{aligned}\sum_s p(s)q^*(t|s)p(y|x = t(s), s) &= \frac{1-e_s}{2}\frac{1-\omega^*}{4}(1-e_c) + \frac{1-e_s}{2}\frac{\omega^*}{4}(0) + e_s\frac{1}{8}(1-e_c) \\ &= (1-e_s)\omega^*\frac{1-e_c}{8} + e_s\frac{1-e_c}{8}.\end{aligned}$$

Similarly, if $t(*) = 1$, $\sum_s p(s)q^*(t|s)p(y|x = t(s), s) = (1-e_s)\omega^*\frac{1-e_c}{8}$. So, normalizing and recalling that four functions have $t(*) = 0$ and four have $t(*) = 1$, we obtain

$$\begin{aligned}\frac{\sum_s p(s)q^*(t|s)p(y|x = t(s), s)}{\sum_{s,t'} p(s)q^*(t'|s)p(y|x = t'(s), s)} &= \frac{1}{4}\frac{1-\omega^*(1-e_s)}{1-\omega^*(1-e_s)+\omega^*(1-e_s)} \\ &= \frac{1-\omega^*(1-e_s)}{4} \\ &= Q^*(t|y)\end{aligned}$$

for the case $y = 0, t(*) = 0$. The cases $y = 0, t(*) = 1$ and $y = 1$ follow using the same arguments as we used for the first condition. Finally, if $y = *$, $\sum_s p(s)q^*(t|s)p(y|x = t(s), s) = \frac{1-e_s}{2}\left(\frac{1-\omega^*}{4} + \frac{\omega^*}{4}\right)e_c + e_s\frac{1}{8}e_c = \frac{1}{8}$ for all t. $Q^*(t|y) = \frac{1}{8}$ when $y = *$, so we have verified both fixed point conditions.

To complete the proof, we must calculate the capacity from q^* and Q^*. In [9], the following formula is given for the capacity:

$$C = \sum_{s,y,t} p(s)q^*(t|s)p(y|t,s)\log\frac{Q^*t|y}{q^*(t|s)}.$$

This sum resembles the sum in the second fixed point condition, so we will use the same approach and consider each value of y separately. The terms corresponding to $y = 0, t(*) = 0$ are

$$\frac{1-e_s}{2}\frac{1-\omega^*}{4}(1-e_c)\log\frac{\frac{1-\omega^*(1-e_s)}{4}}{\frac{1-\omega^*}{4}} + e_s\frac{1}{8}(1-e_c)\log\frac{\frac{1-\omega^*(1-e_s)}{4}}{\frac{1}{8}}.$$

The terms for $y = 0, t(*) = 1$ are

$$\frac{1-e_s}{2}\frac{\omega^*}{4}(1-e_c)\log\frac{\frac{\omega^*(1-e_s)}{4}}{\frac{\omega^*}{4}}.$$

To get the total contribution from the $y = 0$ terms, we need to add these two expressions and multiply by 4 since there are four functions t of each type. Factoring out $\frac{1-e_c}{2}$ and cancelling out common terms in the fractions, we get a total of

$$\frac{1-e_c}{2}\left((1-e_s)(1-\omega^*)\log\frac{1-\omega^*(1-e_s)}{1-\omega^*} + e_s\log 2(1-\omega^*(1-e_s)) + (1-e_s)\omega^*\log(1-e_s)\right).$$

Now, using the same argument that $y = 1$ just flips some terms around, it is easy to see that the total contribution from $y = 1$ terms must be the same as the contribution from $y = 0$ terms. So, the total from $y = 0$ and $y = 1$ terms is

$$(1-e_c)\left((1-e_s)(1-\omega^*)\log\frac{1-\omega^*(1-e_s)}{1-\omega^*} + e_s\log 2(1-\omega^*(1-e_s)) + (1-e_s)\omega^*\log(1-e_s)\right).$$

The contribution from $y = *$ terms will be the same for all t. Multiplying an individual term by 8 gives us a total contribution of

$$(1-e_s)(1-\omega^*)e_c\log\frac{\frac{1}{8}}{\frac{1-\omega^*}{4}} + (1-e_s)\omega^*e_c\log\frac{\frac{1}{8}}{\frac{\omega^*}{2}} + e_se_c\log\frac{\frac{1}{8}}{\frac{1}{8}}.$$

Note that the last term is 0 since $\log 1 = 0$. Adding together the contributions from $y = 0, 1$ and $*$ gives

$$C = (1-e_c)\left((1-e_s)(1-\omega^*)\log\frac{1-\omega^*(1-e_s)}{1-\omega^*} + e_s\log 2(1-\omega^*(1-e_s)) + (1-e_s)\omega^*\log\frac{\omega^*(1-e_s)}{\omega^*}\right)$$

$$
\begin{aligned}
& +e_c\left((1-e_s)(1-\omega^*)\log\frac{1}{2(1-\omega^*)}+(1-e_s)\omega^*\log\frac{1}{2\omega^*}\right) \\
= \; & (1-e_c)(1-e_s)(1-\omega^*)\log\frac{1}{1-\omega^*}+(1-e_c)(1-e_s)(1-\omega^*)\log\left(1-\omega^*(1-e_s)\right) \\
& +(1-e_c)e_s\log\left(1-\omega(1-e_s)\right)+(1-e_c)e_s\log 2+(1-e_c)(1-e_s)\omega^*\log\frac{1}{1-\omega^*} \\
& +(1-e_c)(1-e_s)\omega^*\log\left(\omega^*(1-e_s)\right)+e_c(1-e_s)(1-\omega^*)\log\frac{1}{1-\omega^*} \\
& +e_c(1-e_s)log\frac{1}{2}+e_c(1-e_s)\omega^*\log\frac{1}{\omega^*}+e_c(1-e_s)\omega^*\log\frac{1}{2} \\
= \; & (1-e_s)h_b(\omega^*)-(1-e_c)h_b(\omega^*(1-e_s))+e_s(1-e_c)\log 2+e_c(1-e_s)\log\frac{1}{2} \\
= \; & e_s-e_c+(1-e_s)h_b(\omega^*)-(1-e_c)h_b(\omega^*(1-e_s)).
\end{aligned}
$$

This completes the proof.

Bibliography

[1] A. Shokrollahi, "Raptor Codes." Available at http://www.inference.phy.cam.ac.uk/MacKay/dfountain/RaptorPaper.pdf.

[2] O. Etesami, M. Molkaraie, and A. Shokrollahi. "Raptor Codes on Symmetric Channels." Available at http://www.cs.berkeley.edu/ etesami/raptor.pdf.

[3] E. Martinian and J.S. Yedidia, "Iterative Quantization Using Codes on Graphs," *Allerton Conference on Communications, Control, and Computing*, October 2003.

[4] A. Khandekar, *Graph-based Codes and Iterative Decoding.* PhD thesis, California Intstitute of Technology, 2002.

[5] H.D. Pfister, I. Sason, and R. Urbanke. "Capacity-Achieving Ensembles for the Binary Erasure Channel With Bounded Complexity," *IEEE Trans. on Inform. Theory*, Vol 51 (7), pp. 2352-2379, July 2005.

[6] G.D. Forney, "Codes on Graphs: Normal Realizations," *IEEE Trans. Inform. Theory*, vol. 47, pp. 520-548, February 2001.

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

[8] D. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1723-1731, November 1996.

[9] F. Dupuis, W. Yu, F.M.J. Willems, "Blahut-Arimoto Algorithms for Computing Channel Capacity and Rate-Distortion With Side Information," *IEEE International Symposium on Information Theory (ISIT)*, 2004

[10] S.I. Gel'fand and M.S. Pinsker. Coding for Channel with Random Parameters. *Problems of Control and Information Theory*, vol. 9, no. I, pp. 19-31, 1980.

[11] S. M. Aji and R. J. McEliece, "The Generalized Distributive Law," *IEEE Trans. Inform. Theory*, vol. 46, pp. 325-343, March 2000.

[12] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger, Factor Graphs and the Sum-Product Algorithm, *IEEE Trans. Inform. Theory*, vol. 47, pp. 498-519, February 2001.

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

[14] M. Luby, "LT-codes," in *Proceedings of the 43rd Annual IEEE Symposium on the Foundations of Computer Science (STOC)*, 2002, pp. 271-280.

[15] A. El Gamal and C. Heegard, "On the Capacity of Computer Memory with Defects," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 731-739, September 1983.

[16] U. Erez, S. Shamai, and R. Zamir, "Capacity and lattice strategies for canceling known interference," *IEEE Trans. Inform. Theory*, vol. 51, no. 11, pp. 3820-3833, November 2005.

[17] T. P. Coleman, E. Martinian, M. Effros, and M. Medard, "Interference management via capacity-achieving codes for the deterministic broadcast channel" *Proc. IEEE Information Theory Workshop*, pp. 23-27, September 2005.

[18] W. Yu and M. Aleksic, "Coding for the Blackwell channel: a survey propagation approach," *Proc. IEEE International Symposium on Information Theory*, pp. 1583-1587, September 2005.

[19] Y. Matsunaga and H. Yamamoto, "A Coding Theorem for Lossy Data Compression by LDPC Codes," *IEEE Trans. Inform. Theory*, vol. 49, pp. 2225-2229, September 2003.

[20] A. Barg, "Complexity Issues in Coding Theory," in book: *Handbook of Coding Theory*, vol. 1, Elsevier Science, 1998

[21] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399431, Mar. 1999. See also the correction: *IEEE Trans. Inform. Theory*, vol. 47, pp. 2101, July, 2001.

[22] H.D. Pfister and I. Sason, "Accumulate-Repeat-Accumulate Codes: Systematic Codes Achieving the Binary Erasure Channel Capacity with Bounded Complexity," *Allerton Conference on Communications, Control, and Computing*, September 2005.

[23] C. Hsu and A. Anastasopoulos, "Capacity-Achieving Codes with Bounded Graphical Complexity on Noisy Channels," *Allerton Conference on Communication, Control and Computing*, September 2005.

[24] E. Martinian and M. Wainwright, "Low Density Codes Achieve the Rate-Distortion Bound," dcc, pp. 153-162, Data Compression Conference (DCC'06), 2006.

[25] R. Motwani and P. Raghavan, *Randomized Algorithms.* Cambridge, U.K.: Cambridge Univ. Press, 1995.

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

[27] M. Wainwright and E. Maneva, "Lossy source encoding via message-passing and decimation over generalized codewords of LDGM codes," *Proceedings of the International Symposium on Information Theory*, Adelaide, Australia; September, 2005

[28] J.S. Yedidia, W.T. Freeman, and Y. Weiss, "Understanding Belief Propagation and Its Generalizations," Exploring Artificial Intelligence in the New Millennium, ISBN 1558608117, Chap. 8, pp. 239-236, January 2003 (Science & Technology Books).

3233-15