Searching for Good Low-Density Parity-Check Codes
Theoretical and Practical Approaches

A Thesis presented

by

Mihaela Enachescu

to

Computer Science

in partial fulfillment of the honors requirements

for the degree of

Bachelor of Arts

Harvard College

Cambridge, Massachussets

April 8, 2002

Abstract

This work discusses the problem of designing codes with good performance, discussing both theoretical and practical aspects of this problem. We first present historical results from coding theory, by deriving lower and upper bounds for error-correcting codes and presenting one of the first theoretical constructions approaching these bounds. We then focus on low-density parity-check codes, which approach the lower bound we derive and have very efficient encoding and decoding schemes. We conclude by presenting our results on finding good short LDPC codes based on a heuristic method, based on the distribution of cycles in the underlying graphs representing the codes.

Acknowledgements

My greatest debt of gratitude goes to my advisor, Michael Mitzenmacher. During the past three years he has taken the time to teach me a great deal of theoretical and practical approaches in computer science. Last summer he introduced me to coding theory, low-density parity-check codes in particular, and since then he has constantly helped me to do research in this area. I am extremely grateful to him for his invaluable guidance and support.

Thanks to Professors Avi Pfeffer, Michael Rabin, and Salil Vadhan for the excellent lectures and advice that they have given me throughout my years at Harvard.

Thanks to all my friends, and especially to Ciprian Manolescu, Andreea Balan, Florin Niculescu, and Bogdan Grigorescu who have been extremely helpful through our numerous discussions and as draft readers of this thesis.

Last but not least, I wish to express my gratitude to my parents, Maria Nicola and Marian Enachescu, and to my dear Theo. Throughout my college years, their love, understanding, and continuous emotional support have been essential. Without them, the valleys would have been much lower and the mountains not as high.

Table of Contents

1. Introduction 5

2. Capabilities of codes 7

2.1 The Gilbert-Varshamov Bound 9

2.2 The McEliece Bound 11

2.3 An example: the Justesen's Codes 13

3. Low-Density Parity-Check Codes 16

3.1 Asymptotic behavior of LDPC Codes 18

3.2 Decoding efficiency for the LDPC codes 21

3.3 Decoding of LDPC codes for the binary erasure channel 22

3.4 Encoding algorithms for LDPC codes 24

4. Search for good LDPC codes at short block length 26

4.1 Girth distribution 27

4.2 Experimental Design and Implementation 28

4.3 Results 29

4.4 Conclusions 31

References 32

1. Introduction

Communicating information is an essential process in today's society. However, most means of data transmission involve imperfect channels that may corrupt part of the data. Coding theory is concerned with detecting and correcting in the most efficient way possible the errors that occur in the data transmission. The theoretical origins of coding theory are marked by Shannon's 1948 paper. Recognized as "father of modern digital communications and information theory" [23], Shannon proved the existence of families of error-correcting codes that can perform with arbitrarily small probability of error, up to the rate given by the capacity of the channel [5]. He also showed the converse – that we cannot find codes that do better than the channel capacity. This provided an initial upper bound for the codes we can hope to find.

Shannon's Theorem pioneered the work on the theoretical bounds for codes. Gilbert and Varshamov found a very natural lower bound shortly after. This was the best bound known for 30 years [6], and it presents a challenge for code designers even today. We will derive and present this bound in chapter 2. Different methods were applied for deriving upper bounds. One of the most effective was the linear programming method that led to a series of upper bounds. This method and the McEllies upper bound—which was derived using it—are presented in section 2.2. One of the first families of codes that asymptotically met the Gilbert-Varhamov lower bound were algebraic codes. One example, the Justesen codes, are discussed in detail in section 2.3. While these codes are a very elegant theoretical construction, their lack of an efficient encoding algorithm renders them impractical.

The initial focus was on discovering codes that get arbitrarily close to the capacity of the channel, and which have polynomial time encoding and decoding algorithms. The most popular example is the family of Reed-Solomon (RS) codes that provide optimal information recovery for sufficiently large block lengths, which were discovered and studied in the 1960s. However, for practical values of the block length, their best encoding and decoding algorithms are quadratic in time. Although algorithms with lower asymptotic bounds were discovered and applied to RS codes, the overhead of the constant factor hidden by the big-O notation makes them less efficient for most practical block lengths [7].

The necessity of finding codes with more efficient encoding and decoding schemes has prompted a great deal of attention on finding good low-density parity-check (LDPC) codes. These were first introduced by Gallager [3] in 1963, forgotten, and then rediscovered three decades later. We talk about these codes in the remaining two chapters. In chapter 3 we present some general results and algorithms. These codes have very efficient encoding and decoding algorithms (due to the low density requirement) for a variety of channels. A number of techniques have been developed for constructing capacity-approaching LDPC codes of large (but still practical, for some applications) block lengths [7], [16], [20], [21]. The general idea is to analyze the performance of *ensembles* of codes, and find ones with good average performance. Then, based on

concentration results, which were proven to hold for sufficiently large block lengths [20], random codes are ensured to have good performance as well. This is because the performance is concentrated around the average performance, and hence a good code of large enough length can be generated with high probability by randomly selecting any code in the ensemble.

However, the concentration results are not useful for small block lengths. In fact, empirical results show that for small block lengths (under 10,000), the variations in performance can be quite significant [20]. This variation motivated Mao and Banihashemi to introduce a heuristic method for comparing codes based on the average value of their cycle distribution [12], [13]. The precise definition and the way to efficiently compute this cycle distribution are presented in chapter 4. The method was successfully applied for the binary symmetric channel and a message-passing decoding. In this work we apply this method to the binary erasure channel, in order to search for good low-density parity-check codes of small block lengths. Note that a given ensemble of codes contains a large number of codes even for small block lengths. While evaluating the performance for all the codes in order to select the best ones is a more accurate mean of classifying the codes, it is too inefficient, and thus not a practical way of searching the large space of possible codes. Prior to Mao's and Banihashemi's work very little progress was made in the direction of developing an efficient and close to accurate method of evaluating and selecting small block lengths codes. Nevertheless, short codes are very important for applications that require the exchange of short and frequent messages, which need to be encoded through such codes to avoid unnecessary overheads and delays. The bloom of such applications in the context of digital and satellite communication, some of which need to transmit data through erasure channels, motivates our idea to apply these authors' method for finding good codes to the erasure channel.

The design and implementation of the testing system, as well as the tests we run and the results are presented in chapter 4. While the evaluation of codes seems to be useful in the case of regular-degree codes, the results are less conclusive for the irregular codes tested. The results suggest that the method applies best to the binary symmetric channel and its corresponding algorithm, but it can also be used with some success for the erasure channel and the iterative decoding algorithm we implemented in our simulations.

2. Capabilities of codes

This chapter explores the question of what we can expect from codes. We give a lower and an upper bound on the information rate of codes. We discuss the problem of constructing and representing a code efficiently, and give one of the first examples of codes that have a nice algebraic representation and good asymptotic properties: the Justesen code. However, while the underlying idea of this code is very elegant, the construction is based on the existence of a primitive element, which cannot be efficiently discovered, and thus this code remains hard to construct in general.

Before proceeding with the derivation of the bounds, we will first introduce some notations and basic mathematical results that will be used in the paper. We will use **Q** to denote the alphabet, containing q symbols, which we use to code the information. A *block code* **C** *of length n* is a subset of $\mathbf{Q^n}$. Each element in **C** is a *codeword*, while the elements of $\mathbf{Q^n}$ are called *words*. The sender always sends the message using the codewords. At the other end, a non-codeword may be received instead of the original codeword, and the receiver needs to decode it. The decoding is usually done by computing the most likely origin. For the channel models considered here (the memoryless symmetric binary-input channels) the closest codeword is the one at smallest *Hamming distance*, i.e. with the smallest number of digits differing from the original received word. More formally, we can write the Hamming-distance as $d(x,y) = |\{i : 1 \le i \le n, x_i \ne y_i\}|$. We define the *minimum distance d* of a code **C** as $d = \min\{d(x,y) : x \in C, y \in C, x \ne y\}$. If **C** is a length n code, with minimum distance d, and M codewords, we call **C** a (n, M, d) code. Note that if $d = 2e+1$, then using **C** the receiver can successfully decode from e or fewer errors.

One of the most important properties of a code **C** is its *information rate R* defined as:

$$R = \frac{\log_q M}{n}$$

Here q is the size of the alphabet **Q**, and so $\log_q M$ represents the length of a code containing no redundant information that would be sufficient to encode all the M codewords, while n is the actual length. When talking about families of codes, another important parameter is the *relative minimum distance* $\delta = \frac{d}{n}$. When the communication is performed through a channel with given probability of error p, the expected number of errors is np, increasing proportionally with n. Thus, the minimum distance has to increase proportionally with n as well, in order to maintain the same error-correcting capabilities. In order to maximize the rate of information of a code we want to maximize M, the number of codewords in **C**. For given n and d, we denote by $A_q(n,d)$ the maximum value of M, for which an (n, M, d) code exists. It is then useful to consider the information rate as a function of the parameter δ, and we define the following:

$$\alpha(\delta) = \limsup_{n\to\infty} \frac{\log_q A_q(n,[n\delta])}{n} \qquad (2.1.1)$$

The bounds we will derive in the next two sections are given in terms of the above parameter.

2.1 The Gilbert-Varshamov Bound

Discovered in 1952, the Gilbert-Varshamov bound was the best bound for 30 years. It is based on the basic fact that no two codewords can be at distance *d*-1 or less from each other. Thus, each codeword has a ball around it of size *d*-1 (according to the metric defined by the Hamming distance) which is free of other codewords. We will use the notation $V_q(n,r)$ to denote the cardinality of the ball of radius *r* in $\mathbf{Q^n}$. The number of words in that ball is then given by the sum of all words at distance less or equal to *r* from the center. Now, the number of words **y** at distance exactly *i* from a given word **x** is: $\binom{n}{i}(q-1)^i$, so $V_q(n,r) = \sum_{i=0}^{r}\binom{n}{i}(q-1)^i$. In the worst case the balls around the codewords of size d-1 are non-overlapping in $\mathbf{Q^n}$. Thus, a code of length *n* and distance *d* could have at least $\frac{q^n}{V_q(n,d-1)}$ codewords, otherwise we could find a non-codeword at distance at least *d* of all other codeword and we could add it to the code **C** . In fact, we can show that there exist linear codes that achieve the Gilbert-Varshamov bound.

Definition 2.1.1 *A **linear code** is a code with the property that the sum of any two codewords is a codeword, so the code forms a linear subspace of* $\mathbf{Q^n}$*, of size k= $\log_q$ M. Such a code is denoted as an [n,k,d]-code, where d is the distance.*

Because of the linearity, for such codes the minimum distance *d* equals the minimum weight of the non-zero codewords. To show that we can get linear codes achieving the Gilbert bound it is enough to prove the following result:

Theorem 2.1.1 [18] *If **C** is a linear code of size n, minimum distance d, and,* $|C| < \frac{q^n}{V_q(n,d-1)}$ *then there is a linear code **C'** of size n, distance d, containing **C**, such that |**C'**|>|**C**|.*

Proof The ball of radius d-1 around the codewords in C do not cover the space $\mathbf{Q^n}$, so there exist some word *v* not in **C**, who is in neither of those balls, i.e. it is at distance at least *d* from all codewords in **C**. Then we define **C'** to be all codewords of the form $u - av$, where $u \in C, a \in Q$. Taking $a = 0$ we can see that all codewords in C are also in **C'**. Taking *a*=-1 and $u = 0$, we get $v \in C'$, and by construction $v \notin C$. So the size and containment requirements of the theorem are satisfied. It is easy to observe that, because

C is linear, **C'** is linear as well. We need to show that **C'** still has minimum distance *d*. It is enough to show that the minimum weight of $u - av$ is *d*. When *a*=0, this is true because $u \in C,$ and **C** has minimum distance *d*. When $a \neq 0,$ *a* has a (non-zero) inverse a^{-1}. Then we have: wt($u - av$) = wt($a^{-1}(u - av)$)=wt($a^{-1}u - v$) =d($a^{-1}u,$ v). Since C is linear $a^{-1}u \in C$, and by construction, v is at distance at least d from it. Thus, we can conclude that $\text{wt}(u - av) \geq d, \forall u, a.$ So C' has all the desired properties. □

Our goal is to derive an expression for $\alpha(\delta)$ using the simple Gilbert bound, and obtain the asymptotic version of the Gilbert bound. This bound turns out to be easily expressed in terms of the q-ary entropy function $H_q(x)$.

Definition 2.1.2 *The **q-ary entropy function** is the function:*

$$H_q(x) = x\log_q(q-1) - x\log_q x - (1-x)\log_q(1-x).$$

Theorem 2.1.2 ***(The Asymptotic Gilbert bound, 1952) [18]*** *For all* $\delta \leq \frac{q-1}{q}$*, there is a family of codes* $\mathbf{C_n}$ *of size n, relative minimum distance converging to* δ *such that* $\alpha(\delta) \to 1 - H_q(\delta)$.

Proof Let *d* be the largest integer less than $n\delta$. From the Gilbert bound, we know there is a code, call it $\mathbf{C_n}$ with rate $R = \frac{\log_q M}{n} = \frac{\log_q \frac{q^n}{V_q(n, d-1)}}{n} = \frac{n - \log_q V_q(n, d-1)}{n}$

$$= 1 - \frac{\log_q V_q(n, d-1)}{n}$$

We want to show that $\limsup_{n\to\infty} R = 1 - H_q(\delta)$, or, by simplifying the constant factor we want to show that $\lim_{n\to\infty}\left(\frac{\log_q V_q(n, d-1)}{n}\right) = H_q(\delta)$. To clarify our explanation, we will use *L* to denote the limit on the left-hand side from now on.

We saw that $V_q(n,r) = \sum_{i=0}^{r}\binom{n}{i}(q-1)^i$, so in particular $V_q(n,r) < \binom{n}{r}(q-1)^r$. Taking the log of both sides, and dividing by n we get a lower bound for *L*, namely $\lim_{n\to\infty}\frac{\log_q\binom{n}{r}(q-1)^r}{n}$. We can simplify the above expression by using Stirling's formula (in the log form), which we will write for simplicity without the subscript *q* as the base of the logs:

$$\log(n!)-\frac{1}{12n}\le\left(n+\frac{1}{2}\right)\log(n)-n+\left(\frac{\log(2\pi)}{2}\right)\le\log(n!)$$

Since we divide by n and take the limit we can ignore the factors: $1/12n$, $1/2\log(n)$ and $\log(2\pi)/2$, when we apply the above inequalities to $\log(n!)$, $\log(r!)$ and $\log((n\text{-}r)!)$
We get that:

$$\lim_{n\to\infty}\frac{\log_q\binom{n}{r}(q-1)^r}{n}\ge\lim_{n\to\infty}\frac{r\log(q-1)+n\log(n)-n-r\log r+r-(n-r)\log(n-r)+n-r}{n}$$

We now use the fact that $r\to n\delta$, and so all terms on the numerator have a factor n, which we simplify with the denominator to get $H_q(\delta)$ on the right hand side of the above equation. After we find that L is bounded below by the entropy function at δ, it is also easy to show that $nH_q(\delta)\ge\log_q V_q(n,d-1)$, by playing with the Binomial expansion of $(\delta+(1-\delta))^n$, so in the limit the entropy function is an upper bound as well as a lower bound for L. Thus we get that $L=H_q(\delta)$, which concludes the proof of Theorem 2.1.2.

□

2.2 The McEliece Bound

In the previous section we presented the fundamental lower bound in coding theory, showing what a random code might achieve. Now, we will present an upper bound as well. This bound was derived using the linear programming method, which was first introduced by Delsarte in 1973 [17]. McEliece adopts the method and applies it to different topics in coding theory [15]. In particular, together with Rodermich, Rumsey, and Welch, he comes up in 1977 with the best upper bound known up to present.

Linear programming is an efficient tool, which gives the optimal result for a linear function of n variables, subject to any number of linear constraints. For codes, the linear constrains are the Delsarte-MacWilliams inequalities, that have as coefficients values of the Krawtchouk polynomials. Let us first state two relevant definitions before stating the theorem:

Definition 2.2.1 *The* ***Krawtchouk polynomial*** $K_j(i)$ *is the coefficient of* y^j *in the polynomial* $(1-y)^i(1+y)^{n-i}$.

Definition 2.2.2 *The average number of codewords at distance i from a given word is given by*

$$a_i=\frac{1}{M}|\{(u,v):d(u,v)=i,u\in C,v\in C\}|$$

Theorem 2.2.1 (The Delsarte-MacWilliams inequalities) [5] *If* $(a_i)_{i=0}^n$ *is the distance distribution of the code* **C** *of length n, then for all* $0 \le k \le n$ *we have*:

$$\sum_{i=0}^{n} a_i K_k(i) \ge 0$$

Proof Let <*x, y*> denote the inner product of *x, y,* elements of $\mathbf{Q^n}$. Fix *x* to be a certain word of weight *i*. Our first claim is that $\sum_{\substack{y \in Q^n \\ wt(y)=k}} \omega^{<x,y>} = K_k(i)$, where ω is a primitive qth root of unity in the complex numbers. We can prove this by a simple counting argument: let $x_{c_1}, x_{c_2} \ldots x_{c_i}$ be the *i* non-zero positions in *x*. Now choose *k* (ordered) positions that overlap with exactly *j* of the c_s's and let D be the set of all words that are non-zero in those positions. There are $\binom{i}{j}\binom{n-i}{k-j}$ choices for D, and for each choice we have:

$$\sum_{y \in D} \omega^{,<x,y>} = \sum_{\substack{h_1, 1st \\ position}} \ldots \sum_{\substack{h_k, kth \\ position}} \prod_{l=1}^{k} \omega^{x_{h_l} y_{h_l}} = (q-1)^{k-j} \prod_{l=1}^{j} \sum_{y \in Q \setminus \{0\}} \omega^{x_{h_l} y} = (-1)^j (q-1)^{k-j}$$

In the above, the second product is taken over the *j*th indices at which both x and y are non-zero. Overall, we obtain exactly $K_k(i) = \sum_{i=0}^{k} (-1)^j \binom{i}{j}\binom{n-i}{k-j}$, which completes the proof of our first claim. We then use this claim to conclude the proof of our theorem by observing that:

$$M \sum_{i=0}^{n} a_i K_k(i) = M \sum_{i=0}^{n} \sum_{\substack{(x,y) \in C^2 \\ d(x,y)=i}} \sum_{\substack{z \in Q^n \\ wt(z)=k}} \omega^{<x-y,z>} = \sum_{\substack{z \in Q^n \\ wt(z)=k}} \left| \sum_{x \in C} \omega^{<x,z>} \right|^2 \ge 0 .$$

□

Now we can define a linear program that maximizes $\sum_{i=0}^{n} a_i$, with the following constraints: i) $a_0 = 1$, ii) $a_j = 0, \forall 1 \le j \le d-1$ iii) $a_i \ge 0, \forall d \le i \le n$, and last but not least iv) $\sum_{i=0}^{n} a_i K_k(i) \ge 0$ (true by the above theorem). (2.2.0)

All (*n, M, d*)-codes have to satisfy the constraints of this linear program, but some solution to the problem above might not have a valid code corresponding to it. If a code exists, then by definition, $M = \sum_{i=0}^{n} a_i$ so the result of the maximization above provides an upper limit for $A_q(n,d)$. However, since we lack a general formula for the result of a linear program, we need to do some more manipulations in order to get a result for our upper bound. We will use the duality principle to transform the problem into a minimization problem. We get the following:

Theorem 2.2.3 [4] *Let* $\gamma(x) = 1 + \sum_{k=1}^{n} \beta_k K_k(x)$ *be a polynomial with* $\beta_k \geq 0, \forall 1 \leq k \leq n$, *and such that* $\gamma(j) \leq 0, \forall d \leq j \leq n$, *where d and n are fixed. Then* $A_q(n,d) \leq \gamma(0)$.

Proof Let $(a_i)_{i=0}^{n}$ be the distance distribution of a (n,M,d)-code. As we saw, they must satisfy the conditions of the initial linear program. Combining i), ii) and iv) we get the following condition of the (a_i)'s: $K_k(0) + \sum_{i=d}^{n} a_i K_k(i) \geq 0$. (2.2.1)

Let us add $a_i \gamma(i)$ for all $d \leq i \leq n$. Since the (a_i)'s are non-negative, from the hypothesis of the theorem we get that the sum of these terms is less or equal to 0. Using the definition of γ, we get that $\sum_{i=d}^{n} a_i (1 + \sum_{k=1}^{n} \beta_k K_k(i)) \leq 0$, or

$$\sum_{i=d}^{n} a_i \leq -\sum_{k=1}^{n} \beta_k \sum_{i=d}^{n} a_i K_k(i)) \overset{from\ (3.1)}{\leq} -\sum_{k=1}^{n} \beta_k (-K_k(0)) = \gamma(0) - 1 \qquad (2.2.2)$$

Since $A_q(n,d) = 1 + \sum_{i=d}^{n} a_i$, from the conditions i) and ii) in (3.0), we get that γ(0) is an upper bound for $A_q(n,d)$ [5]. □

In their 1977 paper, McEliece et al discover that $\gamma(x) = \frac{(\gamma^*(x))^2}{a - x}$ for $a \leq d$ satisfies the **4** conditions of Theorem 2.2.3, where $\gamma^*(x) = \frac{2(a-x)}{t+1}\binom{n}{t}\sum_{k=0}^{t} \frac{K_k(x)K_k(a)}{\binom{n}{a}}$ [14]. Using this function they derived the following bound for $\alpha(\delta)$ for **Q**=**F**^{2}:

$$\alpha(\delta) \leq \min_{0 \leq u \leq 1-2\delta} (1 + g(u^2) - g(u^2 + 2u\delta + 2\delta)) \qquad (2.2.4)$$

In the above $g(x) = H_2\left(\frac{1 - \sqrt{1-x}}{2}\right)$, again the entropy function which played a very important role in coding theory from the very beginning[1]. For u=0, we recover on the right-hand side of (2.2.4) an earlier bound due to Elias. Hence McEliece's result is an improvement of Elias's bound. For $0.273 \leq \delta \leq 1/2$, the minimum in (2.2.4) is achieved for u=1-2δ.

For a general field Q, the MRRW bound equals $H_q(\gamma_q(\delta))$, where

[1] Shannon's definition of the capacity of a channel is 1-H_q(x), and thus his 1948 theorem involves the entropy function, which appears in the GV bound as well (see section 2).

$$\gamma_q(x) = \frac{1}{q}(q-1-(q-2)x - 2\sqrt{(q-1)(x(1-x))}) .$$

2.3 An example: the Justesen's Codes

The lower bound given by Gilbert, while very natural and elegant, does not provide a practical construction to finding a good code. A code discovered by randomly selecting its codewords until we get the tightest packing possible, would likely lack structure, and could only be represented by a table. The number of possible codewords gets very large even for relatively small block lengths. For example, for a block length equal to 500 and a linear code with information rate 1/5, the linear subspace is of dimension 100, so there are 2^{100} possible codewords. Thus it becomes impractical to construct such a table, and even to list the codewords.

Some mathematical structure to enable nice representation and an efficient decoding and encoding scheme is necessary. The first emphasis was to find codes whose basic structure is algebraic. Mathematicians started searching for *good* codes, for which both δ and R are bound away from zero. By this definition, a code may be good even if it is far from meeting the asymptotic version of the basic Gilbert-Varshamov bound presented in section 2. In fact, the Gilbert-Varshamov bound was the best bound known for 30 years, until 1982 when it was exceeded by the Goppa codes. The Goppa codes are algebraic codes with *very good* asymptotic behavior (i.e. are able to get arbitrarily close to the information rate given by Shannon's Theorem). However, they are not yet very practical because of the relative high-cost and complexity of their decoding algorithms [6].

In this section we will present Justesen's Codes, as an example of good codes with a simple algebraic definition. They were defined in the 1970s, almost a decade before the Goppa codes, and represented a major advancement in coding theory at the time, although their definition is non-constructive and also the bound they achieve is below the GV bound [5].

Justesen's Codes are a type of concatenated codes, which are, as the name suggests, constructed from two codes: the *inner-code* C_1, which is the alphabet of the *outer code* C_2. More formally:

Definition 2.3.1 *The **concatenated code** C is given by:*

$$\{(a(b_0),...,a(b_{N-1})) \mid (b_0,...,b_{N-1}) \in C_2, a(b)_i \in C_i\} \qquad (2.3.1)$$

where $a : \{1,..,n\} \to C_1$ *is a one-to-one mapping.*

From the above definition, if C_1 is an (n, m, d) q-ary code, and C_2 is an (N, M, D) Q-ary code with $|Q| \leq m$, then C is a (nN, M, D) q-ary code.

Justesen's construction uses a generalization of the basic definition for concatenated codes. He uses *different* inner codes on the different positions of the outer code. He shows that we can construct codes with relative minimum distance converging to a non-zero value, and with non-zero asymptotic rates bounded by the so-called *Justesen bound*.

Theorem 2.3.2 *For any code rate R, there exist families of concatenated (nN, RnN, D) binary codes C_N, with $N \to \infty$ which meet the Justesen bound*

$$R \geq \max_{\frac{1}{2} \leq r < 1} (1 - \frac{R}{r}) H_2^{-1}(1-r) \qquad (2.3.2)$$

Proof [2] Let $r \geq \max(R, \frac{1}{2})$, and take $k = \lfloor nr \rfloor$. Now construct a (*N, kK, d*) code *C*, where $N = 2^k - 1, K = \left\lfloor \frac{NR}{r} \right\rfloor$, and $d = 2^k - K$ as follows:

1. Pick first the outer code C_{RS}, a (*N, K, d*) Reed-Solomon code[2] over an alphabet of size 2^k (note that the given properties are valid RS parameters).
2. Pick a primitive element α of F_{2^k}, and construct the following N linear codes: $W_j = \{(b, \alpha^j b) \mid b \in F_{2^k}\}, 0 \leq j < N$, where we view the elements of F_{2^k} as binary strings of length 2^k. These (2^k, 2*k*, 2) codes, of rate ½ are also known as the *Wolzencraft set* or randomly shifted codes [2].
3. Construct the inner codes (call them I_j) by deleting the last 2*k-n* digits in W_j's.
4. Our code is now obtained by mapping a word in C_{RS} to a word in *C*, using the *i*th inner code for encoding the *i*th letter of the Reed-Solomon codeword.

We want to derive a bound for D, the minimum distance of the resulting code We want to derive a bound for *D*, the minimum distance of the resulting code *C*. We make the following observations:

i) Because of the minimum distance *d*, every non-zero codeword *c* in C_{RS} has weight at least *d*, so there are at least *d* non-zero positions c_i in *c*.
ii) If we fix the codeword *c*, and look at the non-zero c_j s, then the length 2^k $(c_j, \alpha^j c_j)$ s are all distinct, either because $c_{j_k} \neq c_{j_l}$, or because $\alpha^{j_k} c_j \neq \alpha^{j_l} c_j$, for the same c_j.
iii) From ii), after puncturing in 2k-n positions, there are at most 2^{2k-n} pre-images for a given $m_j(c_j)$, and each of the pre-images appear at most one, so there are at most 2^{2k-n} repetitions of a given $m_j(c_j)$ substring in a codeword.
Using the above, we can see that the difference between two codewords contains a certain substring at most 2^{2k-n} times, so the minimum distance D is at least $2^{2k-n} \sum_{i=0}^{\left\lfloor \frac{D}{2^{2k-n}} \right\rfloor} w_i$,

[2] Reed-Solomon codes encode a message by creating a polynomial having the message symbols as its coefficients, and sending the values of the polynomial at various points (as many as the block length of the code).

where the w_i are the least weights possible. Let us introduce the following notation: $L = \left\lfloor \frac{D}{2^{2k-n}} \right\rfloor$ and denote by S the sum of the L weights w_i.

As $n \to \infty$, $D \to 2^k\left(1-\frac{R}{r}\right)$, so $L \to 2^{n-k}\left(1-\frac{R}{r}\right)$. Since there are $\binom{n}{i}$ binary strings of weight i, by a basic property of the entropy function[3] we know that $\sum_{i=0}^{\lambda n}\binom{n}{i} \le 2^{nH(\lambda)}$, and taking $t = \lambda n = \lfloor nH^{-1}(1-r) - \log_2 n \rfloor$, it is easy to show that the number of words of weight less or equal to t is $o(L)$ therefore almost all of the weights in S are at least $t = n\lambda$. Thus, asymptotically, we have $S \succ n\lambda L$, and we get:

$$D \succ n\lambda 2^{n-k} L \approx n(H^{-1}(1-r))(1-\frac{R}{r})2^k . \qquad (2.3.3)$$

But the length of the Justesen code is $nN = n(2^k - 1)$, and we can conclude that the relative minimum distance satisfies (2.3.2). □

Justesen's idea of building concatenated codes with good asymptotic behavior was extended later, by Zybalov, in 1982, leading to better lower bounds, which remained however below the Gilbert-Varshamov bound [2].

[3] See [1], Chapter 1 on Mathematical Preliminaries, p. 20-21.

3. Low-Density Parity-Check Codes

The emphasis in the earlier years of coding theory was on codes whose basic structure was algebraic. A different approach, more difficult to evaluate at the time, was the use of random codes. Shannon's proof of the capacity of a noisy channel, as well as many similar fundamental results in coding theory were derived by analyzing the average properties of a large class of codes. Low-density parity-check codes are based on the intuition that the best of several codes chosen at random from an ensemble of codes will have properties at least as good as the average ones. The crucial innovation was Gallager's introduction of low complexity iterative decoding algorithms, which can decode very close to the channel capacity. LDPC codes can be used for a variety of channels, and recently, the best LDPC code of length one million achieved a bit-error probability that was clearly surpassing the other known codes for the binary-input additive white Gaussian channel [21]. Despite their very attractive properties, LDPC codes were forgotten for three decades, and rediscovered recently. One practical issue, that remained open for a period of time, was the encoding costs. Currently, however, several encoding techniques have been proposed for LDPC codes that lead to linear or nearly linear encoding [22], [7]. In this chapter we will present the capabilities of LDPC codes, efficient encoding algorithms, some decoding algorithms for the binary-erasure channel and the binary-symmetric channel, as well as concentration results that ensure that random LDPC codes of sufficiently large block length come extremely close to capacity on many channels.

The binary-symmetric channel is a channel for which both the input and the output are sequences of the binary digits 0 and 1. The channel is memoryless, meaning that for every input the probability that the output digit is switched is p, and the probability that the output is the same is $1-p$. The channel is entirely specified by this crossover probability.

The binary erasure channel has the same input as the binary-symmetric channel, but the output can be either 0, 1, or *?* (the erasure symbol). The input never gets flipped but, with some fixed probability p, it can get erased, meaning the corresponding output is ? with that probability, or the same as the input with probability $1-p$.

Before proceeding, some general remarks about LDPC codes: LDPC codes are a special type of linear codes (which were defined in section 2.1). One way to describe a linear code is to give its generator matrix, whose rows are codewords that generate all other possible codewords. A code of a certain block length n over a finite field F_q forms a linear subspace. If the dimension of this subspace is k, then there are k linearly independent codewords (rows in the matrix) that can generate this subspace, so the generator matrix is a $k \times n$ matrix over F_q. An equivalent, more useful way (for decoding purposes especially) is to represent the code using its parity-check matrix, defined so that it returns the zero vector when multiplied by any codeword in the code it specifies. For LDPC codes the parity check matrix also satisfies a certain sparsity condition, in that only a certain fraction of the entries are non-zero. This condition might be requiring the matrix

to have a certain number of 1's per column or row. A matrix satisfying the given constraints is then selected at random by a process which we discuss in more detail below.

Example 1 [Parity-Check Matrix of a code of length 12]

$$H = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

As stated above, the code represented by the above matrix is formed of all codewords x satisfying $Hx^T = 0^T$. (*)

When describing and analyzing the different algorithms for encoding and decoding, it is generally useful to look at an LDPC code as a bipartite graph with the message nodes that represent rows in the parity-check matrix, and the check nodes on the left representing the rows. A check node implies a dependence relation on some of the message nodes (see figure 1). The dependence can be extracted from the parity check matrix. Edges connect the check nodes to the message nodes that they depend on. In the above example we would have 12 variable nodes (one per column) and 6 check nodes (one per row). A check node represents the constraint given by the corresponding row of the matrix, namely that the sum of the variable nodes corresponding to non-zero entries is 0. In the bipartite graph we connect these variable nodes with the corresponding check node. The number of non-zero entries in the row is the number of connections of the corresponding check node. Because the matrix is sparse, with a linear fraction of non-zero entries, the resulting bipartite graph contains a number of edges, which is a linear (instead of a possibly quadratic) factor of the number of nodes n. This allows for linear time encoding and decoding algorithms, which, as we mentioned in the introduction, is a great practical advantage of the LDPC codes. Note that the condition of sparsity is an overall requirement, which allows for a high degree for a given node (corresponding to a high number of 1's on a given row/column in the parity-check matrix) as long as the other nodes have low degrees to make up for the high degrees of a few other nodes.

If the number of 1's in the same for each column and for each row, then the code is *regular*. Regular codes were the focus of the first studies, by both Gallager, and later by MacKay and Neal [6] and Wiberg[25], who independently rediscovered LDPC codes. However, the initial attraction to regular designs proved to be misleading. In fact, the best performance codes tend to be of irregular forms, with multiple degree values on the right and left as shown by Luby, Mitzenmacher, Shokrollahi, and Spielman [7]. The underlying degree distribution tends to have a great influence on the performance of the

codes. For large enough block lengths, finding optimal degree sequences[4] plays a major role in the search for good codes.

Once the degree sequence is specified, we can construct a random code with that degree sequence by pairing up the "edge slots" on the left with the "edge slots" on the right in a random manner. Each vertex would have a number of edge slots corresponding to its degree. For example, in figure 1, the first message node x_1 has degree 3 since there are 3 edge slots associated with it; similarly x_2 has degree 4, and x_n degree 5.

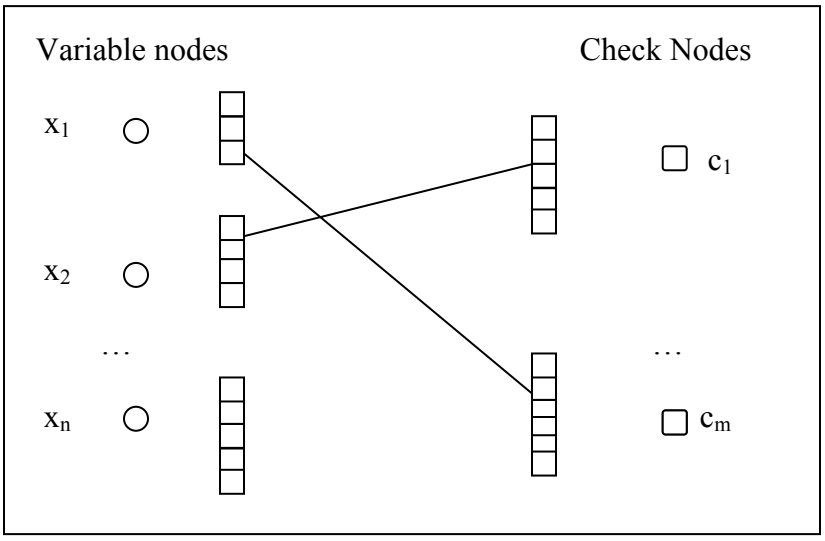

Figure 1. The edge slots on the left (corresponding to different variable nodes) connect to the edge slots on the right (check nodes). In the figure, one of the edge slots corresponding to x_1 connects to an edge slot of c_m. Similarly x_2 connects to c_1. The value of c_1 is the check-sum of the five variable nodes that would connect to it, which is always 0 by (*).

3.1 Asymptotic capabilities of LDPC Codes

Gallager showed that, for sufficiently large block lengths, most codes have minimum relative distance close to the average. His analysis focuses on codes that are regular. Depending on the specific parameters of the codes, namely the left and right degrees, the codes have relative minimum distance approaching that given by the Gilbert bound presented in the previous chapter as one of the fundamental bounds that good families of codes could achieve. The analysis works for a variety of binary-symmetric channels, in particular BSC. Since the Gilbert bound equals the capacity of the channel, this means that LDPC codes have nearly ideal asymptotic performance.

The proof for low-density parity-check codes is similar to that for the general ensemble of parity-check codes of a given rate *R*, with some supplementary analysis relating to the specific sparsity condition considered.

[4] A degree sequence specifies the fraction of edges connected to nodes of each degree both on the right and on the left.

A code of block length n, with rate R, has a fraction of $1-R$ dependent nodes, which gives the fraction of dependent relations the nodes would satisfy, and thus the fraction of check nodes (rows in the parity-check matrix). Consider the ensemble of codes determined by the parity-check matrices with dimensions $n(1-R)\times n$ and 0 and 1 entries, each digit having the same probability for a given entry. This is essentially the ensemble of codes of rate R, except that some codes may have a rate slightly higher, since the rows are not necessarily linearly independent. However, for the sake of simplicity, in the following discussion we assume that the rate is exactly R. Under this assumption, we analyze the relative minimum distance of this ensemble.

Let us define the distance function of a parity-check code to be the number of codewords in the code of weight l and denote it by $W(l)$. Since these codes are linear, the minimum distance D of a code is the same as the smallest distance from the all-zero codeword. Note that $\vec{0}$ is always part of the code, because it satisfies (*) for any party-check matrix H. Thus we can express D in terms of $W(l)$, namely as the smallest value $l>0$ such that $W(l)\neq 0$. The goal is to have a large D, which would give a code with high error-detection and error-correction capabilities.

Theorem 3.1.1 *Define $\overline{W(l)}$ to be the average number of codewords of weight l in the ensemble of codes of length n and rate R defined above. Then, for $l>0$,*

$$\overline{W(l)}=\binom{n}{l}2^{-n(1-R)}\leq\frac{1}{\sqrt{2\pi n\lambda(1-\lambda)}}e^{nH(\lambda)-n(1-R)\ln 2} \qquad (3.1.1)$$

where $\lambda=\frac{l}{n}$, and H is the q-ary entropy function for q=2 (see previous chapter).

Proof Define $P(l)$ to be the probability that a code, chosen at random from the ensemble, contains a codeword of weight l. Note that $P(0)=1$ since the all-zero word is always a codeword, as discussed above. Now consider only the case $l>0$. Since the probability of a given entry of the parity-check matrix to be 0 or 1 is ½, the probability that a given parity-check equation (row) is satisfied is also ½, which we can see by looking only at the entry corresponding to the last position in which the weight l codeword is non-zero. Since a sequence is a codeword if and only if all parity-check equations are satisfied, and there are $n(1-R)$ such equations, it means that the probability $P(l)$ is equal to $2^{-n(1-R)}$.

Since there are $\binom{n}{l}$ words of weight l, the expected number of codewords among these, i.e. the average value of $W(l)$ we are looking for is given by $\binom{n}{l}2^{-n(1-R)}$.

Using Stirling's approximation formula for $n!$, which we encountered before when deriving the asymptotic Gilbert bound, it follows that

$$\binom{n}{l} = \binom{n}{\lambda n} \leq \frac{1}{\sqrt{2\pi n\lambda(1-\lambda)}} e^{nH(\lambda)} \qquad (3.1.2)$$

Combining the above two results we immediately obtain the statement of the theorem. □

Now we will analyze the minimum distance for the ensemble of codes considered, noting that its distribution function satisfies the following theorem.

Theorem 3.1.2 *Over the ensemble of parity-check codes of length l and rate R, the minimum distance distribution function is bounded by the following inequality, where* $\delta < \frac{1}{2}$ *is such that* $n\delta$ *is an integer:*

$$\Pr(D \leq n\delta) \leq \frac{1}{1-2\delta}\sqrt{\frac{1-\delta}{2\pi n\delta}} e^{nH(\delta)-n(1-R)\ln 2}.$$

Proof We will derive a bound on the probability that a non-zero word of weight δn or less is a codeword. This probability is clearly less than the sum of the probabilities that individual words are codewords. Using the probability derived in Theorem 3.1.1, we have:

$$\Pr(D \leq n\delta) \leq \sum_{l=1}^{n\delta} \binom{n}{l} 2^{-n(1-R)} \qquad (3.1.3)$$

We can rewrite the above summation as:

$$\binom{n}{n\delta}\left[1 + \frac{n\delta}{n-n\delta+1} + \frac{n\delta(n\delta-1)}{(n-n\delta+1)(n-n\delta+2)} + \cdots\right]$$

Since $\frac{n\delta-k}{n-n\delta+1+k} < \frac{n\delta}{n-n\delta+1}$ for all $k = \overline{1..n\delta}$, we can upper bound the above summation by the geometric series $1 + x + x^2 + \ldots$, with $x = \frac{n\delta}{n-n\delta+1}$, and obtain that

$$\sum_{l=1}^{n\delta} \binom{n}{l} 2^{-n(1-R)} \leq \binom{n}{n\delta}\frac{1}{1-x} \leq \binom{n}{n\delta}\left(\frac{1-\delta}{1-2\delta}\right).$$

The statement of our theorem immediately follows from substituting this result into (3.1.3). □

Gallager further notes that as n gets large, this bound on $\Pr(D \leq \delta n)$ approaches a step function, with the step at $\delta_0 < \frac{1}{2}$ for which $H(\delta_0) = (1-r)\ln 2$. This relates to the asymptotic Gilbert bound on the minimum distance. While Gilbert proves the existence

of one code with $D > n\delta_0$, the above result shows that most of the codes have distance close to $n\delta_0$. More formally, for any $\varepsilon > 0$, theorem 3.1.2 shows that the probability of a parity-check code to have distance $D < n(\delta_0 - \varepsilon)$ approaches 0 exponentially with *n*. [3]

Note that the above result works for a general parity-check ensemble, with no low-density condition. A similar result holds for low-density parity-check ensembles. Depending on the specific condition, namely the right and left degrees, the threshold δ is slightly less than δ_0. Gallager analyzes these thresholds for regular ensembles only, and he shows that as the values of the right and left degrees in the ensemble increases, these get closer to the value δ_0 [3]. More recent analysis showed that carefully chosen irregular ensembles tend to decode with high probability at rates closer to the capacity of the channel than is the case for the regular ensembles [21]. The specific irregular structures that lead to the best LDPC ensembles depends on both the channel and the specific decoding algorithms used.

Let us explain briefly what makes irregular codes perform better than regular codes. In a regular code, the number of check nodes corresponding to a given message node is constant, i.e. each message node is equally "protected". To increase protection we would need to have a large number of adjacent check nodes. Thus the message nodes would have a larger degree, and so would the check nodes. However, this would make the check nodes less reliable because they would depend on more message nodes to be received correctly. By contrast, irregular codes do not need to balance reliability and protection uniformly. In fact, with irregular codes it is possible to have a "wave effect", in which the nodes with the best protection are corrected first, and then their results are transmitted through check nodes to the less protected ones [8].

3.2 Decoding efficiency for the LDPC codes

In his thesis, Gallager discussed several decoding algorithms that apply to the underlying bipartite graph representing a code. The algorithms work iteratively, and information is exchanged between nodes in the graph by passing messages along the edges connecting them. The messages represent an estimate of the value of the message bit on the left hand side of the edge. In the first round, the message nodes simply send the values initially received on the channel. The check nodes respond with a message dependent on the messages received. The message nodes then combine these responses and their originally received value and compute a new message to send. This process continues, hopefully converging on the maximum-likelihood codeword for the received message (for specific details of the values sent by the algorithms see the next section).

All decoding algorithms we present generate messages based on *extrinsic* information, meaning the messages send to a node should not depend on the messages received from that node. This property, which is important in the proof of the performance bounds for the decoding algorithms, requires the graphs to have no cycles of degree less than the

number of rounds needed for the results to converge. The proofs show that there exist some parameter depending on the degree sequence of the graphs, such that if the initial fraction of errors is below this parameter, then the fraction of incorrect messages passed at each round decreases exponentially with the number of rounds, under the independence assumption [20]. Richardson and Urbanke then analyze the fraction of errors at each step for a general ensemble of codes, looking at the ensembles of all codes with a given degree distribution and of a certain block length. These ensembles would contain codes with small cycles. The authors show, however, that the fraction of incorrect messages passed at some step for a given code is close to the average fraction of incorrect messages. The difference converges to zero exponentially fast in the block length. This result is the *concentration property*. The authors then show that the average fraction of incorrect messages for an ensemble converges to value for the cycle-free codes, but the convergence is much slower in terms of the block length. But for the cycle-free codes, the fraction of incorrect messages decreases to zero, as mentioned above. This shows that for large enough block lengths, there is a high probability that the decoding algorithm successfully converges for any code, as long as the fraction of errors is below the threshold limit.

Richardson and Urbanke also present different methods for determining the threshold value for ensembles of codes, and use them to analyze the thresholds for some ensembles on the binary symmetric channel. They also explore methods for finding good degree distributions that lead to threshold values close to the channel capacity, an exciting research problem of high practical value. The methods depend on the specific channel and decoding algorithms considered. For the binary symmetric channel they conjecture that, as the maximum degree in the distribution increases, the thresholds will converge to the ultimate limit, the channel capacity [21]. For the binary erasure channel Luby, Mitzenmacher, Shokrollahi, and Spielman have produced codes with rate arbitrarily close to the channel capacity, which is $1-p$, as showed by Elias [7]. More specifically, they construct, for all $\varepsilon > 0$ codes of rate $R = 1-p(1+\varepsilon)$ that, using the erasure decoding algorithm presented in the next section, can recover with high probability a message with up to pn erasures in a period of time proportional to $n\ln(1/\varepsilon)$ [7].

In order to make our discussion about decoding more concrete we present and discuss in the next section some specific decoding algorithms for the binary symmetric channel and the binary erasure channel.

3.3 Decoding algorithms for the binary symmetric and binary erasure channels

The following algorithms were the original algorithms proposed and analyzed by Gallager.

Gallager's algorithm A

For each message nodes, the neighboring check nodes send the XOR of the messages received from all their adjacent message nodes other[5] than the receiver of the message. The message node continues to send the originally received bit (if any) unless all messages received from the adjacent check nodes (other than the receiver of the message) disagree with the original value. In the later case he "switches," and sends the value received rather than the original value.

Gallager's algorithm B

Gallager observed that the above algorithm leads to better results when the message nodes switch their value sooner. In this revised algorithm, for each round there is an optimal threshold value, which gives the number of disagreeing message from other check nodes needed for the message node to "switch" the originally received value. The check nodes behave in the same way as in algorithm A.

By extending Gallager's algorithm to allow nodes to be indecisive, Mitzenmacher produced an improved performance algorithm [26], which suggests using larger alphabets to provide a more robust decoding at the expense of increase decoder complexity. The limit, obviously, is a completely continuous value alphabet to be sent between the nodes. This leads to the following decoding algorithm.

Belief propagation

In the previous algorithms, the value sent by the message nodes represented the "best guess" of the nodes's correct value. Using a continuous alphabet, belief propagation is instead able to communicate an approximation of the probability. This is the probability, conditioned on the information received from all other adjacent message nodes, that the check node will be satisfied if message node is 0 for example (for non-binary alphabets, the message node would send multiple values, for all but one of the possible values it could take). Technically, the message sent is the a posteriori probability of the value of the associated variable based on the values of all nodes observed up to and including the last round. Note that both the message and the check nodes compute these probability distributions[6].

The above algorithms apply directly for the binary symmetric channel, and they can be adapted with minor modifications to work for the binary erasure channel as well. For example, the belief propagation algorithm would have to send two different values, since the message alphabet is ternary, not binary, in this model.

Finally, we present a simple decoding algorithm for the binary erasure channel, which we used in our experimental testing for the codes generated in chapter 4.

[5] Recall the independence condition, which requires that the message sent to a node should be independent of the message received from that node.
[6] For the specific details of this computation an excellent reference is [20].

Erasure Decoding
Do the following:

Search for a check node for which all but one message value is known.
Set the missing value to be the XOR of the other (known) values

Repeat until either all nodes are recovered or the search returns no node.

Unlike the previous algorithms, which may run for an undetermined number of rounds until convergence, this algorithm is assured to be linear in the number of nodes and edges. However, despite its low complexity this algorithm was proved to decode with high probability codes of sufficiently large block lengths and rates converging to the channel capacity [7]. Experimental data suggest this simple algorithm can recover a significant fraction of errors, and thus can be used in combination with another algorithm, such as belief propagation to create a more robust decoding scheme, of intermediate[7] complexity.

3.4 Encoding algorithms for LDPC codes

In the previous sections we presented theoretical evidence that LDPC codes could achieve error-correction close to capacity. LDPC codes exhibit asymptotically better performance than other classes of codes, such as turbo codes[8], for example. Moreover, the variety of decoding algorithms devised for LDPC codes, allows for a wide variety of tradeoffs between performance and decoding complexity. However, a major criticism has been the apparent high encoding complexity. While turbo codes, for example, can be encoded in linear time, the most straightforward construction proposed for encoding an LDPC code takes quadratic time in the block length. We will present this construction, and give alternative ideas that reduce the encoding complexity down to linear time.

The basic quadratic algorithm is based on the manipulation of the $m \times n$ parity-check matrix of the code. We assume that the parity-check equations are all linear independent, so that the *rate* of the code (the fraction of non-redundant bits) is exactly $n - m/n$.

I. Preprocessing step: Using Gaussian elimination, transform the matrix H into lower-triangular form, and use the new matrix H' as the parity-check matrix for the encoding. Note that we can now divide the codeword into a *systematic* part (the first non-redundant $n - m$ bits), and a *parity* part, that can be computed using the parity-check matrix from the previous bits.

II. Encoding step: i) Fill in the systematic part with the $n - m$ desired information symbols. ii) Determine the m parity-check bits, noting that, because H' is in lower-

[7] The complexity of the combined scheme would in general be smaller than that of the belief propagation algorithm used by itself, since the lower number of initial errors should lead to faster convergence rate
[8] This is another class of codes that have generated a lot of interest in the recent years and is regarded as one of the most efficient coding schemes [20].

triangular form, each of these bits depends only on the bits at lower index, and each bit can be computed by looking at a single row in the matrix H'.

After the $O(n^3)$ operations required to bring the matrix into lower-triangular form, the time needed to do the actual encoding is proportional to the number of non-zero entries in H', since for each parity bit, which is computed by looking at a given row, we need to add all the previous bits corresponding to non-zero entries in the row. Since after doing Gaussian elimination, in general the matrix will no longer be sparse, there are $O(n^2)$ additions to be performed.

If we can guarantee a sparsity condition on H' similar to that on H the above algorithm would be linear. However, simply forcing the parity-check matrix to have lower-triangular form would, in general, result in some loss of performance. Richardson and Urbanke developed in [22] greedy algorithms to transform a sparse matrix into an equivalent almost lower-triangular *sparse* matrix. They showed that for certain degree sequences these algorithms produce very good results giving rise with high probability to codes that allow both transmission close to the capacity of the channel and linear encoding complexity.

4. Search for good LDPC codes at short block length

The very efficient decoding and encoding algorithms for LDPC codes, discussed above make them especially attractive for a number of practical applications, in particular for online applications that are especially sensitive to the encoding and decoding time complexity. These online applications, such as telnet, require the frequent and reliable exchange of messages of small length. These applications would greatly benefit from the LSPC coding scheme. This motivates our desire to find good short LDPC codes, which could then be used for the above applications.

The challenge for generating good codes of short block length is that, as we discussed in section 3.2 the convergence to the performance approaching the channel capacity is quite slow for two reasons. First, the average performance of the general ensemble is generally lower than the performance of the cycle-free ensemble, and it converges toward the later performance slowly. Secondly, the performance of the cycle-free ensemble is guaranteed to approach the channel capacity only for large block length, and in practical terms these "large enough" values are larger than the ones we would like to consider here.

The encouraging factor is the variation in performance of short codes, which was also mentioned in the introduction. This variation implies that the best of a sample of randomly selected codes can be much better than the average. The idea is to find an efficient method of comparing codes that detects the better ones with high probability. Mao and Banihashemi propose such a method in [12], [13], and show that it performs well for the binary symmetric channel. The idea of the method is based on the intuition that small cycles interfere with the decoding process, because the independence assumption is violated and the errors propagate faster than they can be corrected. They give a method that efficiently compares the codes based on their girth distribution, which is a function of the cycles in the bipartite graph representing the code, and selects the best according to this function. They show that this method works well in practice for the binary symmetric channel.

In the case of the erasure decoding which we use in our work, the propagation of error is avoided by the strong independence condition imposed by the algorithm. However, small cycles increase the probability that there exist a small stopping set, which is a set of errors that the algorithm cannot recover [1]. Thus, if we can select the codes without small cycles, we increase the probability that the erasure algorithm does not get stuck. Thus, there is a good intuitive reason why the same heuristic method proposed by Mao and Banihashemi should also work for the binary erasure channel.

4.1 Girth distribution

Here we give the definition of girth distribution on which Mao and Banihashemi method is based. For a given graph, the *girth* is defined as the smallest cycle in the graph. Mao and Banihashemi extend the definition of girth for a node u in the graph to be the smallest

cycle which includes u. Because our graph is bipartite, all cycles must have even length. This leads to an efficient method of computing the girth of a node. A cycle of length $2l$ passing through u indicates the presence of at least two paths of equal length, l, from u to v, the l'th node in the cycle, starting from u. This node v must be connected to two nodes at distance $l-1$ from u. Thus, to detect the girth of the node u, we proceed as follows: we start at u, and successively construct layers of nodes at distance 1, 2, etc. from u, by adding the neighbours of the nodes from the previous layer. We can easily check for the first layer at which a node is the neighbour of al least two nodes from the previous layer, which indicates a cycle passing through this node and the original node u. At this point we halt and report that the girth is $2\times$(layer number). In O(n^2) time we can thus compute the girth for all nodes in the graph and determine the number of nodes with girth values 2, 4, 6, …l_{max}, This is defined to be the *girth distribution* of the graph. Note that l_{max}, the maximum girth of a node is necessarily smaller than the number of nodes in the graph (in fact it is smaller than twice the number of both message and check nodes of the bipartite graph representing a code).

We can use different measures based on the degree distribution. Mao and Banihashemi use the average girth. Based on the idea that smaller cycles are worst than larger cycles, we want to avoid a higher number of nodes of a given girth, which is worst than a smaller number of nodes of the same girth. Accordingly, we used as our principal comparison measure $\sum_{i=2}^{l_{\max}} g(i)\times\frac{1}{i}$, where $g(i)$ represents the number of nodes of girth i. Thus smaller girths contribute more to the sum, and a smaller value of the sum indicates bigger cycles.

4.2 Experimental Design and Implementation

To test the performance of this method on the binary symmetric channel we implemented several functionalities:

1. A random code generator, which takes in a list of pairs of the form (degree, number of nodes of that degree) for both the message and check nodes. We decided to specify the number of nodes rather than the number of edges (with a certain degree) to avoid round-up errors, and potential mismatches in the number of edge slots on the right and left. As explained in section 2, each node is associated with a number of edge slots corresponding to its degree. A random code is generated by randomly pairing the edge slots of the message nodes with the edge slots of the check nodes. Note that we must have the same number of edges on both sides.

2. Given a code we can find its girth distribution, as described in section 3. We also compute different evaluating functions based on the girth distribution. We have considered the following evaluating functions:

   a. $\sum_{i=2}^{l_{\max}}\frac{1}{i}\times g(i)$

b. $\sum_{i=2}^{l_{\max}} \frac{1}{\sqrt{i}} \times g(i)$, which could potentially reduce the influence of small girths.

c. $\sum_{i=2}^{6} \frac{1}{i} \times g(i)$, focusing on the influence of the small girth values only.

3. To search for the codes with best girth properties we implemented two heuristic search algorithms:

a. Repeated Random, which selects a code with the best score for a given evaluating function, out of a random sample of 1000 randomly generated codes with same number of nodes and a certain degree distribution.
b. Metropolis, which, given a code, switches two random edges, and if the resulting code turns out to have a better score, Metropolis selects it. Otherwise, Metropolis selects it only with a low probability. This procedure is repeated 1000 times, and the encountered code with the best evaluating function is returned.

4. To test the codes, we implemented the iterative decoding algorithm discussed in section 2. A random codeword compatible with the given code is first generated. Then a fixed number of message nodes are erased. The iterative algorithm is then run on the resulting word, and the number of unrecovered message nodes is reported.

We tested two kinds of ensembles:

a. A regular (3,6)-degree ensemble of codes with 600 message nodes, and 300 check nodes. Note that for the regular codes, 3 is the optimal value for the message nodes degree.

b. An irregular ensemble with 7 different degrees on the left, and 3 different degrees on the right. The degrees and numbers of nodes of each degree were chosen to approximate a probability distribution derived in [7], for irregular codes over the erasure channel. Due to problems of matching degrees on the left and right hand side and to our desire to have about 600 message nodes and 300 check nodes (as in a.), the rounding up involved in the calculation of the number of nodes of each degree leads to values not very close to the theoretical ones, especially in the case of the check nodes. We present below the theoretical vs. actual fraction of edges of given left and right degrees.

(a)

Degree	Theoretical	Actual
3	0.430034	0.436179205
13	0.237331	0.241758242
14	0.007979	0.007889546
48	0.119493	0.121724429
49	0.052153	0.055226824
162	0.07963	0.091293322
163	0.07338	0.045928431

(b)

Degree	Theoretical	Actual
10	0.713788	0.859397013
11	0.122494	0.027895182
200	0.163718	0.112707805

Figure 2. Fraction of edges of the given degrees on the right (a) and left (b).

4.3 Results

The three evaluation functions induced the same ordering on the random codes tested. The evaluating functions a. and c. returned the same scores indicating that the score was dominated by the small cycles (of size 6 and lower).

For both the repeated random and the metropolis algorithms, the codes selected had the same score values, so neither method outperformed the other.

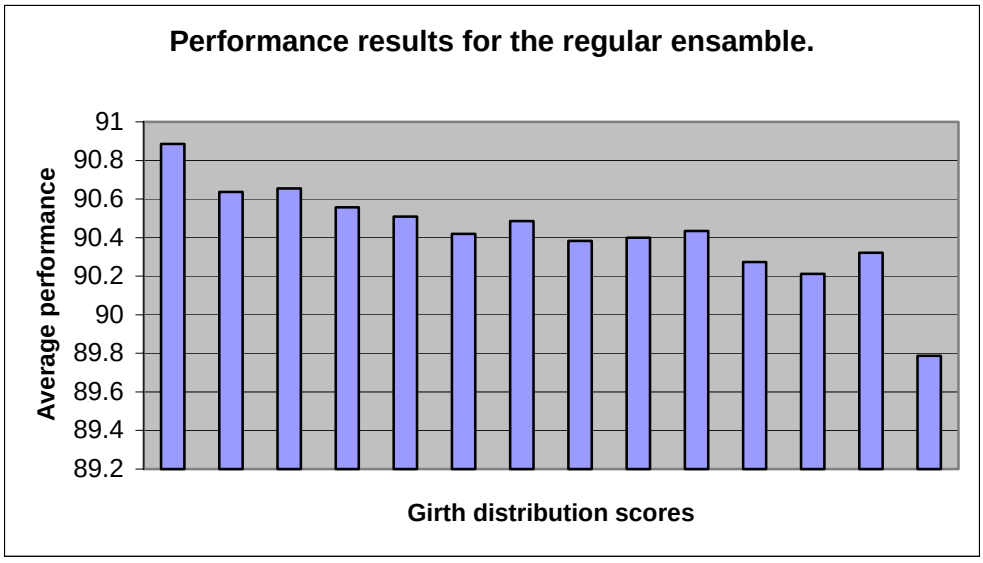

Figure 3.

Figure 3 presents the average performance of the codes for the different score values encountered in a random ensemble of 1000 codes. The performance is given in terms of the fraction of recovered errors from an initial fraction of 33.3% erasures (200 nodes are

erased). The scores are listed in increasing order. As noted in section 2, lower scores indicated girth distributions with fewer smaller cycles, and we expect this property to lead to better performance. The above results confirm this claim. All codes encountered with the best girth score had performance above the average.

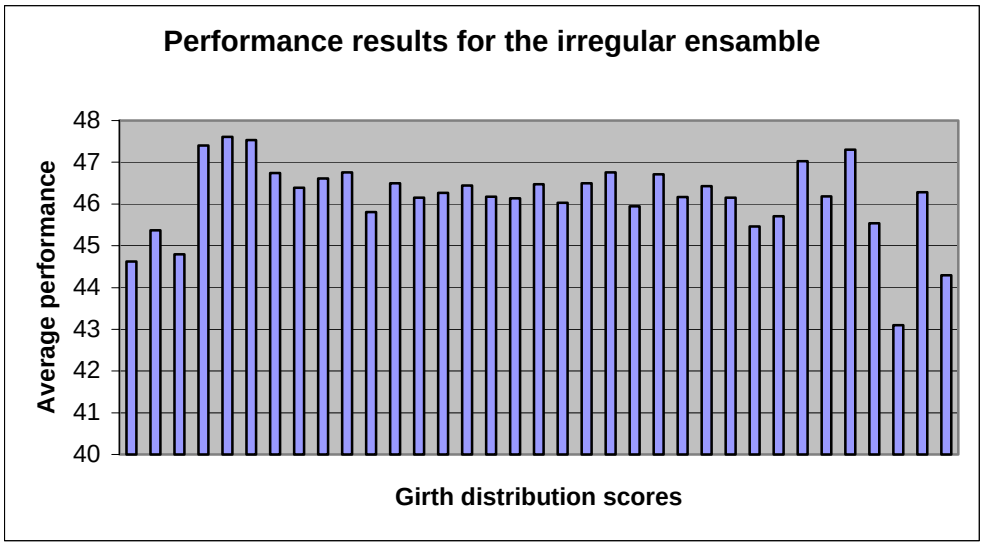

Figure 4.

The score values are more widely distributed in the irregular case. However, the average performance of this particular irregular ensemble we tested (with 600 message nodes and 347 check nodes) is significantly lower than that of the regular ensemble, probably due to our choice of a degree distribution, which is better suited for longer length codes, and despite the fact that it contains a larger number of check nodes. The few nodes with very high degrees probably introduce a large number of small cycles/stopping sets, which are hard to avoid. Indeed, while the score range for the regular codes is 150 to 153.25, the codes do not go below the score of 160.25.

Also, Figure 4 does not indicate the correlation between lower girth distribution scores and high performance that we would have expected. While codes with higher than average performance tend to have lower scores, the lowest scores are not favored. In fact all four codes with the lowest three scores have below-average performance.

Note that there is a concentration of codes with scores in the middle of the range of scores in the ensemble, consistent with the concentration observed in [12] and [13] for the average girth evaluating function.

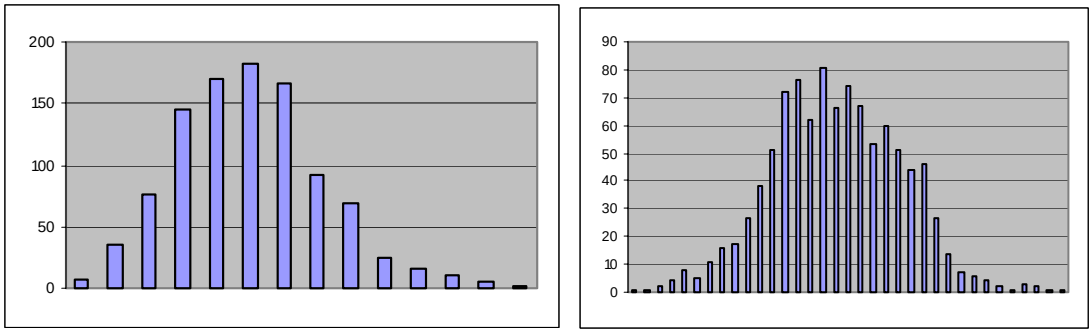

Figure 5. Number of nodes at different girth distribution scores for the regular ensemble (right) and the irregular ensemble (left). Note the higher score variance for the irregular ensemble.

4.4 Conclusions

The results obtained were encouraging in the case of the higher-performance regular codes, but were inconclusive for the lower performance, irregular codes that we tested. One possible explanation is that the method is appropriate for selecting the best codes only in an ensemble with already good average performance. The specific degree sequence selected plays an important role, and sequences that have good asymptotic performance are not necessarily those that give best results for selecting short block-length codes.

Another reason for the lower performance of the irregular code is the presence of a very high degree node, which greatly increases the probability of small cycle in such a small code. Also the rounding errors introduced by the need to have integer solutions, whereas the linear program used gives rational solutions, may lead to a degree sequence that is further from optimal than expected.

Overall, our results indicate that the degree sequence used plays a far more important role in the performance of the code we select, than the condition that the code performs well within the ensemble. The variation of average performances between ensembles is far greater than the variation of performance within the ensemble, even for such short block lengths. Thus one important area for future exploration is to find good degree sequences for short length codes, and good irregular ensembles on which to test this hypothesis.

Another idea is to develop efficient heuristics or greedy methods to combine in the process of generating a random code so that nodes with low girths are avoided.

References

1. Di C., Proietti D., Telatar E., Richardson T., Urbanke R. *Finite Length Analysis of Low-Density Parity-Check Codes for the BEC*. 39th Annual Allerton Conference on Communication, Control, and Computing, Illinois, 2001.

2. Dumer, I. *Concatenated Codes and Their Multilevel generalizations*. Handbook of Coding Theory, (see [18]), Vol. 2, p. 1911-1988.

3. Gallager, R.G. Low Density Parity-Check Codes. Cambridge, MA: MIT Press, 1963.

4. Levenhtein, V.I. *Universal Bounds for Codes and Designs*. Handbook of Coding Theory, Edited by V.S. Pless and W.C. Huffman.Elsevier, 1998. Vol. 1, p. 499-648.

5. van Lint, J.H. Introduction to Coding Theory, 2nd Edition. Springer-Verlag, 1992.

6. van Lint, J.H. *Algebraic Geometric Codes*. Coding Theory and Design Theory, Edited by Dijen Ray-Chaudhuri. Springer-Verlag, 1990.

7. Luby M. G., Mitzenmacher M., Shokrollahi M. A. and Spielman D. *Efficient Erasure Correcting Codes*, IEEE Trans. on Information Theory, Vol. IT-47, pp. 569-584, Feb. 2001.

8. Luby M. G., Mitzenmacher M., Shokrollahi M. A. and Spielman D. *Improved Low Density Parity Check Codes Using Irregular Graphs and Belief Propagation*. IEEE Trans. on Information Theory, Vol. IT-47, pp. 585-598, Feb. 2001.

9. MacKay, D.J.C and Neal, RM. *Good codes based on very sparse matrices*. Cryptography and Coding: Proceedings of the 5th IMA conference. Springer-Verlag, 1995, p. 100-111.

10. Extended version (55 pages) of the above paper available at: http://131.111.48.24/mackay/abstracts/mncN.html

11. MacWilliams, F.J. *An Historical Survey*. Error Correcting Codes, Edited by Henry B. Mann. John Willey & Sons, 1968, p. 3-13.

12. Mao, Y. and Banihashemi, A.H. *A Heuristic Search for Good Low-Density Parity-Check Codes at Short Block Lengths*, presented at IEEE ICC2001, Helsinki, Finland, June 11-14, 2001.

13. Mao, Y. and Banihashemi, A.H. *Design of Good LDPC Codes Using Girth Distribution*, presented at IEEE International Symposium on Information Theory, Italy, June, 2000.

14. McEliece, R. et al. *New Upper Bounds on the Rate of a Code via the Delsarte-MacWilliams Inequalities*. IEEE Trans Inf Th, Vol. IT-23, No2, 1977, p. 157-166.

15. McEliece, R. *The Bounds of Delsarte and Lovasz, and Their Applications to Coding Theory*. Algebraic Coding Theory and Applications, Edited by G. Longo. Springer-Verlag, 1979, p. 107-178.

16. Oswald, P. and Shokrollahi, M.A. *Capacity-achieving sequences on the erasure channel*, Proceedings of the IEEE International Symposium on Information Theory '00, p.5. 2000.

17. Poli, A. and Huguet, L. Error Correcting Codes. Prentice Hall, 1992.

18. Pretzel, O. Codes and Algebraic Curves. Oxford University Press, 1998.

19. Pyndiah, R. A. Brief Historic of Turbo Codes. http://www-sc.enst.bretagne.fr/historic.html

20. Richardson, T.J. and Urbanke, R.L. *The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding*, IEEE Trans. on Information Theory, Vol. IT-47, pp. 599-618, Feb. 2001.

21. Richardson T.J., Shokrollahi M.A. and Urbanke R.L. *Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes*, IEEE Trans. on Information Theory, Vol. IT-47, pp. 619-637, Feb. 2001.

22. Richardson, T.J. and Urbanke, R.L. *Efficient Encoding of Low-Density Parity-Check Codes*, IEEE Trans. on Information Theory, Vol. IT-47, pp. 638-656, Feb. 2001.

23. Sales, Robert J. MIT Professor Claude Shannon: Founder of Digital Communications. http://web.mit.edu/newsoffice/nr/2001/shannon.html

24. Shokrollahi, M.A. *Capacity-Achieving Sequences*, Codes, Systems, and Graphical Models, number 123 of IMA volumes in Mathematics and its Applications, B. Marcus and J. Rosenthal (eds), pp. 153-166, 2000.

25. Wiberg, N. *Codes and Decoding on General Graphs*. Dissertation no. 440, Dept. Elect. Eng. Linkoping Univ., Linkoping, Sweden, 1996.