11

LOW-DENSITY PARITY-CHECK CODES

{ch:LDPC}

Low-density parity-check (LDPC) error correcting codes were introduced in 1963 by Robert Gallager in his Ph.D. thesis. The basic motivation came from the observation that random linear codes, cf. Section **??**, had excellent theoretical performances but were unpractical. In particular, no efficient algorithm was known for decoding. In retrospect, this is not surprising, since it was later shown that decoding for linear codes is an NP-hard problem.

The idea was then to restrict the RLC ensemble. If the resulting codes had enough structure, one could exploit it for constructing some efficient decoding algorithm. This came of course with a price: restricting the ensemble could spoil its performances. Gallager's proposal was simple and successful (but ahead of times): LDPC codes are among the most efficient codes around.

In this Chapter we introduce one of the most important families of LDPC ensembles and derive some of their basic properties. As for any code, one can take two quite different points of view. The first is to study the code performances[27] under *optimal* decoding. In particular, no constraint is imposed on the computational complexity of decoding procedure (for instance decoding through a scan of the whole, exponentially large, codebook is allowed). The second approach consists in analyzing the code performance under some specific, efficient, decoding algorithm. Depending on the specific application, one can be interested in algorithms of polynomial complexity, or even require the complexity to be linear in the block-length.

Here we will focus on performances under optimal decoding. We will derive rigorous bounds, showing that appropriately chosen LDPC ensembles allow to communicate reliably at rates close to Shannon's capacity. However, the main interest of LDPC codes is that they can be decoded efficiently, and we will discuss a simple example of decoding algorithm running in linear time. The full-fledged study of LDPC codes under optimal decoding is deferred to Chapters **??**. A more sophisticated decoding algorithm will be presented and analyzed in Chapter **??**.

After defining LDPC codes and LDPC code ensembles in Section 11.1, we discuss some geometric properties of their codebooks in Section 11.2. In Section 11.3 we use these properties to a lower bound on the threshold for reliable communication. An upper bound follows from information-theoretic considera-

[27] Several performance parameters (e.g. the bit or block error rates, the information capacity, etc.) can be of interest. Correspondingly, the 'optimal' decoding strategy can vary (for instance symbol MAP, word MAP, etc.). To a first approximation, the choice of the performance criterion is not crucial, and we will keep the discussion general as far as possible.

tions. Section 11.4 discusses a simple-minded decoding algorithm, which is shown to correct a finite fraction of errors.

{se:DefLDPC}

11.1 Definitions

11.1.1 *Boolean linear algebra*

Remember that a code is characterized by its codebook $\mathfrak{C}$, which is a subset of $\{0,1\}^N$. LDPC codes are **linear codes**, which means that the codebook is a linear subspace of $\{0,1\}^N$. In practice such a subspace can be specified through an $M \times N$ matrix $\mathbb{H}$, with binary entries $\mathbb{H}_{ij} \in \{0,1\}$, and $M < N$. The codebook is defined as the kernel of $\mathbb{H}$:

$$\mathfrak{C} = \{\, \underline{x} \in \{0,1\}^N \;:\; \mathbb{H}\underline{x} = \underline{0} \,\}\,. \tag{11.1}$$

Here and in all this chapter, the multiplications and sums involved in $\mathbb{H}\underline{x}$ are understood as being computed modulo 2. The matrix $\mathbb{H}$ is called the **parity check matrix** of the code. The size of the codebook is $2^{N-\mathrm{rank}(\mathbb{H})}$, where $\mathrm{rank}(\mathbb{H})$ denotes the rank of the matrix $\mathbb{H}$ (number of linearly independent rows). As $\mathrm{rank}(\mathbb{H}) \leq M$, the size of the codebook is $|\mathfrak{C}| \geq 2^{N-M}$. With a slight modification with respect to the notation of Chapter 1, we let $L \equiv N - M$. The rate R of the code verifies therefore $R \geq L/N$, equality being obtained when all the rows of $\mathbb{H}$ are linearly independent.

Given such a code, encoding can always be implemented as a linear operation. There exists a $N \times L$ binary matrix $\mathbb{G}$ (the generating matrix) such that the codebook is the image of $\mathbb{G}$: $\mathfrak{C} = \{\underline{x} = \mathbb{G}\underline{z}$, where $\underline{z} \in \{0,1\}^L\}$. Encoding is therefore realized as the mapping $\underline{z} \mapsto \underline{x} = \mathbb{G}\underline{z}$. (Notice that the product $\mathbb{H}\,\mathbb{G}$ is a $M \times L$ 'null' matrix with all entries equal to zero).

11.1.2 *Factor graph*

In Example 9.5 we described the factor graph associated with one particular linear code (a Hamming code). The recipe to build the factor graph, knowing $\mathbb{H}$, is as follows. Let us denote by $i^a_1, \dots, i^a_{k(a)} \in \{1, \dots, N\}$ the column indices such that $\mathbb{H}$ has a matrix element equal to 1 at row a and column i^a_j. Then the a-th coordinate of the vector $\mathbb{H}\underline{x}$ is equal to $x_{i^a_1} \oplus \cdots \oplus x_{i^a_{k(a)}}$. Let $P_{\mathbb{H}}(\underline{x})$ be the uniform distribution over all codewords of the code $\mathbb{H}$ (hereafter we shall often identify a code with its parity check matrix). It is given by:

$$P_{\mathbb{H}}(\underline{x}) = \frac{1}{Z} \prod_{a=1}^{M} \mathbb{I}(x_{i^a_1} \oplus \cdots \oplus x_{i^a_k} = 0)\,. \tag{11.2}$$

Therefore, the factor graph associated with $P_{\mathbb{H}}(\underline{x})$ (or with $\mathbb{H}$) includes N variable nodes, one for each column of $\mathbb{H}$, and M function nodes (also called, in this context, **check nodes**), one for each row. A factor node and a variable node are joined by an edge if the corresponding entry in $\mathbb{H}$ is non-vanishing. Clearly this procedure can be inverted: to any factor graph with N variable nodes and M

function nodes, we can associate an $M \times N$ binary matrix $\mathbb{H}$, the **adjacency matrix** of the graph, whose non-zero entries correspond to the edges of the graph.

11.1.3 *Ensembles with given degree profiles*

{se:LDPCegdp}

In Chapter 9 we introduced the ensembles of factor graphs $\mathbb{D}_N(\Lambda, P)$. These have N variable nodes, and the two polynomials $\Lambda(x) = \sum_{n=0}^{\infty} \Lambda_n x^n$, $P(x) = \sum_{n=0}^{\infty} P_n x^n$ define the degree profiles: Λ_n is the probability that a randomly chosen variable node has degree n, P_n is the probability that a randomly chosen function node has degree n. We always assume that variable nodes have degrees ≥ 1, and function nodes have degrees ≥ 2, in order to eliminate trivial cases. The numbers of parity check and variable nodes satisfy the relation $M = N\Lambda'(1)/P'(1)$.

We define the ensemble $\mathrm{LDPC}_N(\Lambda, P)$ to be the ensemble of LDPC codes whose parity check matrix is the adjacency matrix of a random graph from the $\mathbb{D}_N(\Lambda, P)$ ensemble. (We will be interested in the limit $N \to \infty$ while keeping the degree profiles fixed. Therefore each vertex typically connects to a vanishingly small fraction of other vertices, hence the qualification 'low density'). The ratio $L/N = (N-M)/N = 1 - \Lambda'(1)/P'(1)$, which is a lower bound to the actual rate R, is called the **design rate** R_{des} of the code (or, of the ensemble). The actual rate of a code from the $\mathrm{LDPC}_N(\Lambda, P)$ ensemble is of course a random variable, but we will see below that it is in general sharply concentrated 'near' R_{des}.

A special case which is often considered is the one of 'regular' graphs with fixed degrees: all variable nodes have degree l and all functions nodes have degree k, (i.e. $P(x) = x^k$ and $\Lambda(x) = x^l$). The corresponding code ensemble is usually simply denoted as $\mathrm{LDPC}_N(l, k)$, or, more synthetically as (l, k). It has design rate $R_{\mathrm{des}} = 1 - \frac{l}{k}$.

Generating a uniformly random graph from the $\mathbb{D}_N(\Lambda, P)$ ensemble is not a trivial task. The simplest way to by-pass such a problem consists in substituting the uniformly random ensemble with a slightly different one which has a simple algorithmic description. One can proceed for instance as follows. First separate the set of variable nodes uniformly at random into subsets of sizes $N\Lambda_0$, $N\Lambda_1$, ..., $N\Lambda_{l_{\max}}$, and attribute 0 'sockets' to the nodes in the first subset, one socket to each of the nodes in the second, and so on. Analogously, separate the set of check nodes into subsets of size MP_0, MP_1, ..., $MP_{k_{\max}}$ and attribute to nodes in each subset $0, 1, \ldots, k_{\max}$ socket. At this point the variable nodes have $N\Lambda'(1)$ sockets, and so have the check nodes. Draw a uniformly random permutation over $N\Lambda'(1)$ objects and connect the sockets on the two sides accordingly.

Exercise 11.1 In order to sample a graph as described above, one needs two routines. The first one separates a set of N objects uniformly into subsets of prescribed sizes. The second one samples a random permutation over a $N\Lambda'(1)$. Show that both of these tasks can be accomplished with $O(N)$ operations (having at our disposal a random number generator).

This procedure has two flaws: (*i*) it does not sample uniformly $\mathbb{D}_N(\Lambda, P)$, because two distinct factor graphs may correspond to a different number of permutations. (*ii*) it may generate multiple edges joining the same couple of nodes in the graph.

In order to cure the last problem, we shall agree that each time n edges join any two nodes, they must be erased if n is even, and they must be replaced by a single edge if n is odd. Of course the resulting graph does not necessarily have the prescribed degree profile (Λ, P), and even if we condition on this to be the case, its distribution is not uniform. We shall nevertheless insist in denoting the ensemble as $\mathrm{LDPC}_N(\Lambda, P)$. The intuition is that, for large N, the degree profile is 'close' to the prescribed one and the distribution is 'almost uniform', for all our purposes. Moreover, what is really important is the ensemble that is implemented in practice.

Exercise 11.2 This exercise aims at proving that, for large N, the degree profile produced by the explicit construction is close to the prescribed one.

(*i*) Let m be the number of multiple edges appearing in the graph and compute its expectation. Show that $\mathbb{E}\, m = O(1)$ as $N \to \infty$ with Λ and P fixed.

(*ii*) Let (Λ', P') be the degree profile produced by the above procedure. Denote by

$$d \equiv \sum_l |\Lambda_l - \Lambda'_l| + \sum_k |P_k - P'_k|\,, \tag{11.3}$$

the 'distance' between the prescribed and the actual degree profiles. Derive an upper bound on d in terms of m and show that it implies $\mathbb{E}\, d = O(1/N)$.

{se:WELDPC}

11.2 Geometry of the codebook

As we saw in Sec. 6.2, a classical approach to the analysis of error correcting codes consists in studying the 'geometric' properties of the corresponding codebooks. An important example of such properties is the distance enumerator $\mathcal{N}_{\underline{x}_0}(d)$, giving the number of codewords at Hamming distance d from $\underline{x}_0$. In the case of linear codes, the distance enumerator does not depend upon the reference ⋆ codeword $\underline{x}_0$ (the reader is invited to prove this simple statement).It is therefore customary to take the all-zeros codeword as the reference, and to use the denomination **weight enumerator**: $\mathcal{N}(w) = \mathcal{N}_{\underline{x}_0}(d = w)$ is the number of codewords having **weight** (the number of ones in the codeword) equal to w.

In this section we want to estimate the expected weight enumerator $\overline{\mathcal{N}}(w) \equiv \mathbb{E}\mathcal{N}(w)$, for a random code in the $\mathrm{LDPC}_N(\Lambda, P)$ ensemble. In general one expects, as for the random code ensemble of Sec. 6.2, that $\overline{\mathcal{N}}(w)$ grows exponentially in the block-length N, and that most of the codewords have a weight

$w = N\omega$ growing linearly with N. We will in fact compute the exponential growth rate $\phi(\omega)$ defined by

{eq:weightphidef}
$$\overline{\mathcal{N}}(w = N\omega) \doteq e^{N\phi(\omega)} \ . \tag{11.4}$$

Notice that this number is an 'annealed average', in the terminology of disordered systems: in other words, it can be dominated by rare instances in the ensemble. On the other hand, one expects $\log\mathcal{N}(w)$ to be tightly concentrated around its typical value $N\phi_{\rm q}(\omega)$. The typical exponent $\phi_{\rm q}(\omega)$ can be computed through a quenched calculation, for instance considering $\lim_{N\to\infty} N^{-1}\mathbb{E}\log[1+\mathcal{N}(w)]$. Of course $\phi_{\rm q}(\omega) \le \phi(\omega)$ because of the concavity of the logarithm. In this Chapter we keep to the annealed calculation, which is much easier and gives an upper bound. Quenched calculations will be the object of Chapter **???**.

Let $\underline{x} \in \{\mathtt{0},\mathtt{1}\}^N$ be a binary word of length N and weight w. Notice that $\mathbb{H}\underline{x} = 0 \mod 2$ if and only if the corresponding factor graph has the following property. Consider all variable nodes i such that $x_i = \mathtt{1}$, and color in red all edges incident on these nodes. Color in blue all the other edges. Then all the check nodes must have an even number of incident red edges. A little thought shows that $\overline{\mathcal{N}}(w)$ is the number of 'colored' factor graphs having this property, divided by the total number of factor graphs in the ensemble. We shall compute this number first for a graph with fixed degrees, associated with a code in the $\mathrm{LDPC}_N(l,k)$ ensemble, and then we shall generalize to arbitrary degree profiles.

11.2.1 *Weight enumerator: fixed degrees*

In the fixed degree case we have N variables nodes of degree l, M function nodes of degree k. We denote by $F = Mk = Nl$ the total number of edges. A valid colored graph must have $E = wl$ red edges. It can be constructed as follows. First choose w variable nodes, which can be done in $\binom{N}{w}$ ways. Assign to each node in this set l red sockets, and to each node outside the set l blue sockets. Then, for each of the M function nodes, color in red an even subset of its sockets in such a way that the total number of red sockets is $E = wl$. Let m_r be the number of function nodes with r red sockets. The numbers m_r can be non-zero only when r is even, and they are constrained by $\sum_{r=0}^{k} m_r = M$ and $\sum_{r=0}^{k} r m_r = lw$. The number of ways one can color the sockets of the function nodes is thus:

{eq:colsock}
$$\mathcal{C}(k,M,w) = \sum_{m_0,\dots,m_k}^{(e)} \binom{M}{m_0,\dots,m_k} \ \mathbb{I}\Big(\sum_{r=0}^{k} m_r = M\Big) \ \mathbb{I}\Big(\sum_{r=0}^{k} r m_r = lw\Big) \ , \tag{11.5}$$

where the sum $\sum^{(e)}$ means that non-zero m_r appear only for r even. Finally we join the variable node and check node sockets in such a way that colors are matched. There are $(lw)!(F-lw)!$ such matchings out of the total number of $F!$ corresponding to different element in the ensemble. Putting everything together, we get the final formula:

$$\overline{\mathcal{N}}(w) = \frac{(lw)!(F-lw)!}{F!} \binom{N}{w} \mathcal{C}(k,M,w) \ . \tag{11.6}$$

In order to compute the function $\phi(\omega)$ in (11.4), one needs to work out the asymptotic behavior of this formula when $N \to \infty$ at fixed $\omega = w/N$. Assuming that $m_r = x_r M = x_r N l/k$, one can expand the multinomial factors using Stirling's formula. This gives:

{eq:weightphires1}
$$\phi(\omega) = \max_{\{x_r\}}{}^{*} \left[(1-l)\mathcal{H}(\omega) + \frac{l}{k} \sum_r \left(-x_r \log x_r + x_r \log \binom{k}{r} \right) \right] , \qquad (11.7)$$

where the max* is taken over all choices of $x_0, x_2, x_4, \ldots$ in $[0,1]$, subject to the two constraints $\sum_r x_r = 1$ and $\sum_r r x_r = k\omega$. The maximization can be done by imposing these constraints via two Lagrange multipliers. One gets $x_r = C z^r \binom{k}{r} \mathbb{I}(r \text{ even})$, where C and z are two constants fixed by the constraints:

$$C = \frac{2}{(1+z)^k + (1-z)^k} \qquad (11.8)$$

$$\omega = z \frac{(1+z)^{k-1} - (1-z)^{k-1}}{(1+z)^k + (1-z)^k} \qquad (11.9)$$

Plugging back the resulting x_r into the expression (11.10) of ϕ, this gives finally:

{eq:weightphires1}
$$\phi(\omega) = (1-l)\mathcal{H}(\omega) + \frac{l}{k} \log \frac{(1+z)^k + (1-z)^k}{2} - \omega l \log z \ , \qquad (11.10)$$

where z is the function of ω defined in (11.9).

We shall see in the next sections how to use this result, but let us first explain how it can be generalized.

11.2.2 *Weight enumerator: general case*

We want to compute the leading exponential behavior $\overline{\mathcal{N}}(w) \doteq \exp[N\phi(\omega)]$ of the expected weight enumerator for a general $\mathrm{LDPC}_N(\Lambda, P)$ code. The idea of the approach is the same as the one we have just used for the case of regular ensembles, but the computation becomes somewhat heavier. It is therefore useful to adopt more compact notations. Altogether this section is more technical than the others: the reader who is not interested in the details can skip it and go to the results.

We want to build a valid colored graph, let us denote by E its number of edges (which is no longer fixed by w). There are $\mathsf{coeff}[\prod_l (1+xy^l)^{N\Lambda_l}, x^w y^E]$ ways of choosing the w variable nodes in such a way that their degrees add up to E [28]. As before, for each of the M function nodes, we color in red an even subset of its sockets in such a way that the total number of red sockets is E. This can be done in $\mathsf{coeff}[\prod_k q_k(z)^{MP_k}, z^E]$ ways, where $q_k(z) \equiv \frac{1}{2}(1+z)^k + \frac{1}{2}(1-z)^k$. The numbers of ways one can match the red sockets in variable and function

[28] We denote by $\mathsf{coeff}[f(x), x^n]$ the coefficient of x^n in the formal power series $f(x)$.

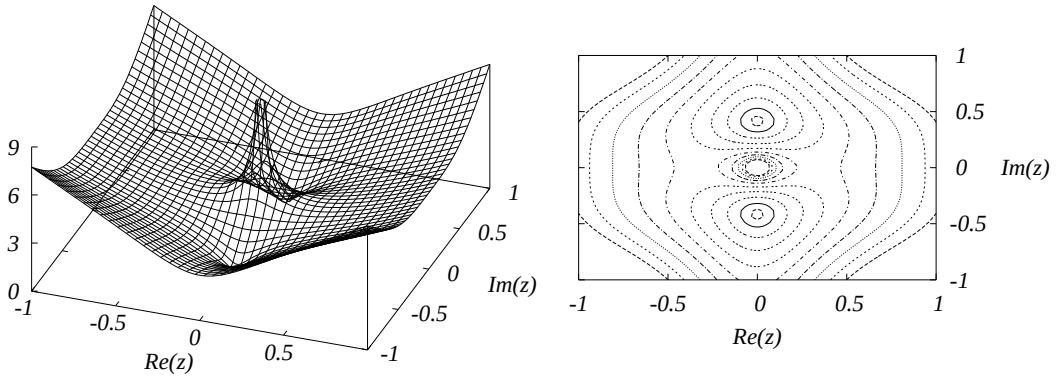

FIG. 11.1. Modulus of the function $z^{-3\xi}\, q_4(z)^{3/4}$ for $\xi = 1/3$. {fig:SaddleWE}

nodes is still $E!(F-E)!$, where $F = N\Lambda'(1) = MP'(1)$ is the total number of edges in the graph. This gives the exact result

$$\overline{\mathcal{N}}(w) = \sum_{E=0}^{F} \frac{E!(F-E)!}{F!} \qquad \text{{eq:WELeading1}}$$
$$\mathsf{coeff}\left[\prod_{l=1}^{l_{\max}} (1+xy^l)^{N\Lambda_l}, x^w y^E\right] \mathsf{coeff}\left[\prod_{k=2}^{k_{\max}} q_k(z)^{MP_k}, z^E\right] . \quad (11.11)$$

In order to estimate the leading exponential behavior at large N, when $w = N\omega$, we set $E = F\xi = N\Lambda'(1)\xi$. The asymptotic behaviors of the $\mathsf{coeff}[\dots,\dots]$ terms can be estimated using the saddle point method. Here we sketch the idea for the second of these terms. By Cauchy theorem

$$\mathsf{coeff}\left[\prod_{k=2}^{k_{\max}} q_k(z)^{MP_k}, z^E\right] = \oint \frac{1}{z^{N\Lambda'(1)\xi+1}} \prod_{k=2}^{k_{\max}} q_k(z)^{MP_k} \frac{\mathrm{d}z}{2\pi i} \equiv \oint \frac{f(z)^N}{z} \frac{\mathrm{d}z}{2\pi i}\,, \quad (11.12)$$

where the integral runs over any path encircling the origin in the complex z plane, and

$$f(z) \equiv \frac{1}{z^{\Lambda'(1)\xi}} \prod_{k=2}^{k_{\max}} q_k(z)^{\Lambda'(1)P_k/P'(1)}\,. \quad (11.13)$$

In Fig. 11.1 we plot the modulus of the function $f(z)$ for degree distributions $\Lambda(x) = x^3$, $P(x) = x^4$ and $\xi = 1/3$. The function has a saddle point, whose

location $z_* = z_*(\xi) \in \mathbb{R}_+$ solves the equation $f'(z) = 0$, which can also be written as

$$\xi = \sum_{k=2}^{k_{\max}} \rho_k \, z \frac{(1+z)^{k-1} - (1-z)^{k-1}}{(1+z)^k + (1-z)^k} \,, \tag{11.14}$$

where we used the notation $\rho_k \equiv kP_k/P'(1)$ already introduced in Sec. 9.5 (analogously, we shall write $\lambda_l \equiv l\Lambda_l/\Lambda'(1)$). This equation generalizes (11.9). If we take the integration contour in Eq. (11.12) to be the circle of radius z_*, the integral is dominated by the saddle point at z_* (together with the symmetric point $-z_*$). We get therefore

$$\mathsf{coeff}\left[\prod_{k=2}^{k_{\max}} q_k(z)^{MP_k}, z^E\right] \doteq \exp\left\{N\left[-\Lambda'(1)\xi\log z_* + \frac{\Lambda'(1)}{P'(1)}\sum_{k=2}^{k_{\max}} P_k \log q_k(z_*)\right]\right\}.$$

Proceeding analogously with the second $\mathsf{coeff}[\dots,\dots]$ term in Eq. (11.11), we get $\overline{\mathcal{N}}(w = N\omega) \doteq \exp\{N\phi(\omega)\}$. The function ϕ is given by

$$\phi(\omega) = \sup_{\xi} \inf_{x,y,z} \Big\{ -\Lambda'(1)\mathcal{H}(\xi) - \omega\log x - \Lambda'(1)\xi\log(yz) + \\ + \sum_{l=2}^{l_{\max}} \Lambda_l \log(1+xy^l) + \frac{\Lambda'(1)}{P'(1)} \sum_{k=2}^{k_{\max}} P_k \log q_k(z) \Big\} \,, \tag{11.15}$$

where the minimization over x, y, z is understood to be taken over the positive real axis while $\xi \in [0,1]$. The stationarity condition with respect to variations of z is given by Eq. (11.14). Stationarity with respect to ξ, x, y yields, respectively

$$\xi = \frac{yz}{1+yz} \,, \qquad \omega = \sum_{l=1}^{l_{\max}} \Lambda_l \frac{xy^l}{1+xy^l} \,, \qquad \xi = \sum_{l=1}^{l_{\max}} \lambda_l \frac{xy^l}{1+xy^l} \,. \tag{11.16}$$

If we use the first of these equations to eliminate ξ, we obtain the final parametric representation (in the parameter $x \in [0,\infty[$) of $\phi(\omega)$:

$$\phi(\omega) = -\omega\log x - \Lambda'(1)\log(1+yz) + \sum_{l=1}^{l_{\max}} \Lambda_l \log(1+xy^l) + \\ + \frac{\Lambda'(1)}{P'(1)} \sum_{k=2}^{k_{\max}} P_k \log q_k(z) \,, \tag{11.17}$$

$$\omega = \sum_{l=1}^{l_{\max}} \Lambda_l \frac{xy^l}{1+xy^l} \,, \tag{11.18}$$

with $y = y(x)$ and $z = z(x)$ solutions of the coupled equations

$$y = \frac{\sum_{k=2}^{k_{\max}} \rho_k \, p_k^-(z)}{\sum_{k=2}^{k_{\max}} \rho_k \, p_k^+(z)}\,, \qquad z = \frac{\sum_{l=1}^{l_{\max}} \lambda_l x y^{l-1}/(1+xy^l)}{\sum_{l=1}^{l_{\max}} \lambda_l/(1+xy^l)\,}, \tag{11.19}$$

where we defined $p_k^\pm(z) \equiv \frac{(1+z)^{k-1} \pm (1-z)^{k-1}}{(1+z)^k + (1-z)^k}$.

Exercise 11.3 The numerical solution of Eqs. (11.18) and (11.19) can be quite tricky. Here is a simple iterative procedure which seems to work reasonably well (at least, in all the cases explored by the authors). The reader is invited to try it with her favorite degree distributions Λ, P.

First, solve Eq. (11.18) for x at given $y \in [0, \infty[$ and $\omega \in [0, 1]$, using a bisection method. Next, substitute this value of x in Eq. (11.19), and write the resulting equations as $y = f(z)$ and $z = g(y, \omega)$. Define $F_\omega(y) \equiv f(g(y, \omega))$. Solve the equation $y = F_\omega(y)$ by iteration of the map $y_{n+1} = F_\omega(y_n)$ Once the fixed point y_* is found, the other parameters are computed as $z_* = g(y_*, \omega)$ and x_* is the solution of Eq. (11.18) for $y = y_*$. Finally x_*, y_*, z_* are substituted in Eq. (11.17) to obtain $\phi(\omega)$.

Examples of functions $\phi(\omega)$ are shown in Figures 11.2, 11.3, 11.4. We shall discuss these results in the next section, paying special attention to the region of small ω.

11.2.3 *Short distance properties*

In the low noise limit, the performance of a code depends a lot on the existence of codewords at short distance from the transmitted one. For linear codes and symmetric communication channels, we can assume without loss of generality that the all zeros codeword has been transmitted. Here we will work out the short distance (i.e. small weight ω) behavior of $\phi(\omega)$ for several LDPC ensembles. These properties will be used to characterize the code performances in Section 11.3.

As $\omega \to 0$, solving Eqs. (11.18) and (11.19) yields $y, z \to 0$. By Taylor expansion of these equations, we get

$$y \simeq \rho'(1) z\,, \qquad z \simeq \lambda_{l_{\min}} x y^{l_{\min}-1}\,, \qquad \omega \simeq \Lambda_{l_{\min}} x y^{l_{\min}}\,, \tag{11.20}$$

where we neglected higher order terms in y, z. At this point we must distinguish whether $l_{\min} = 1$, $l_{\min} = 2$ or $l_{\min} \geq 3$.

We start with the case $l_{\min} = 1$. Then x, y, z all scale like $\sqrt{\omega}$, and a short computation shows that

$$\phi(\omega) = -\frac{1}{2}\,\omega \, \log\left(\omega/\Lambda_1^2\right) + O(\omega)\,. \tag{11.21}$$

In particular $\phi(\omega)$ is strictly positive for ω sufficiently small. The expected number of codewords within a small (but $\Theta(N)$) Hamming distance from a given codeword is exponential in N. Furthermore, Eq. (11.21) is reminiscent of the behavior in absence of any parity check. In this case $\phi(\omega) = \mathcal{H}(\omega) \simeq -\omega \log \omega$.

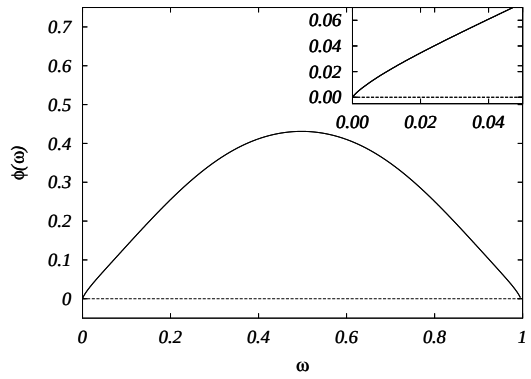

FIG. 11.2. Logarithm of the expected weight enumerator, $\phi(\omega)$, plotted versus the reduced weight $\omega = w/N$, for the ensemble $\mathrm{LDPC}_N(\frac{1}{4}x + \frac{1}{4}x^2 + \frac{1}{2}x^3, x^6)$. Inset: small weight region. $\phi(\omega)$ is positive near to the origin, and in fact its derivative diverges as $\omega \to 0$: each codeword is surrounded by a large number of very close other codewords. This makes it a very bad error correcting code. {fig:WEIRR1}

Exercise 11.4 In order to check Eq. (11.21), compute the weight enumerator for the regular $\mathrm{LDPC}_N(l = 1, k)$ ensemble. Notice that, in this case the weight enumerator does not depend on the code realization and admits the simple representation $\mathcal{N}(w) = \mathsf{coeff}[q_k(z)^{N/k}, z^w]$.

An example of weight enumerator for an irregular code with $l_{\min} = 1$ is shown in Fig. 11.2. The behavior (11.21) is quite bad for an error correcting code. In order to understand why, let us for a moment forget that this result was obtained by taking $\omega \to 0$ *after* $N \to \infty$, and apply it in the regime $N \to \infty$ at $w = N\omega$ fixed. We get

$$\overline{\mathcal{N}}(w) \sim \left(\frac{N}{w}\right)^{\frac{1}{2}w} . \tag{11.22}$$

It turns out that this result holds not only in average but for most codes in the ensemble. In other words, already at Hamming distance 2 from any given codeword there are $\Theta(N)$ other codewords. It is intuitively clear that discriminating between two codewords at $\Theta(1)$ Hamming distance, given a noisy observation, is in most of the cases impossible. Because of these remarks, one usually discards $l_{\min} = 1$ ensembles for error correcting purposes.

Consider now the case $l_{\min} = 2$. From Eq. (11.20), we get

$$\phi(\omega) \simeq A\omega\,, \qquad A \equiv \log\left[\frac{P''(1)}{P'(1)}\frac{2\Lambda_2}{\Lambda'(1)}\right] . \tag{11.23}$$

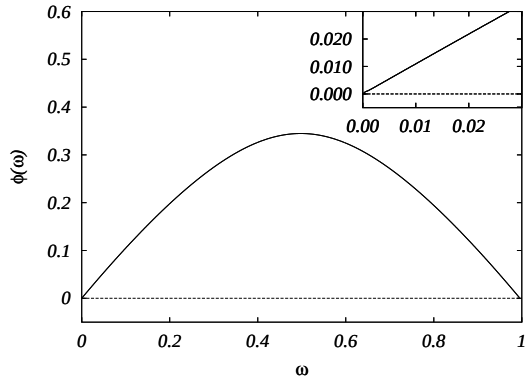

FIG. 11.3. Logarithm of the expected weight enumerator for the $\mathrm{LDPC}_N(2,4)$ ensemble: $\Lambda(x) = x^2$, meaning that all variable nodes have degree 2, and $P(x) = 4$, meaning that all function nodes have degree 4. Inset: small weight region. The constant A is positive, so there exist codewords at short distances {fig:WE24}

The code ensemble has significantly different properties depending on the sign of A. If $A > 0$, the expected number of codewords within a small (but $\Theta(N)$) Hamming distance from any given codeword is exponential in the block-length. The situation seems similar to the $l_{\min} = 1$ case. Notice however that $\phi(\omega)$ goes much more quickly to 0 as $\omega \to 0$ in the present case. Assuming again that (11.23) holds beyond the asymptotic regime in which it was derived, we get

$$\overline{\mathcal{N}}(w) \sim e^{Aw} \,. \tag{11.24}$$

In other words the number of codewords around any particular one is $o(N)$ until we reach a Hamming distance $d_* \simeq \log N/A$. For many purposes d_* plays the role of an 'effective' minimum distance. The example of the regular code $\mathrm{LDPC}_N(2,4)$, for which $A = \log 3$, is shown in Fig. 11.3

If on the other hand $A < 0$, then $\phi(\omega) < 0$ in some interval $\omega \in]0, \omega_*[$. The first moment method then shows that there are no codewords of weight 'close to' $N\omega$ for any ω in this range.

A similar conclusion is reached if $l_{\min} \geq 3$, where one finds:

$$\phi(\omega) \simeq \left(\frac{l_{\min}-2}{2}\right) \omega \, \log\left(\frac{\omega}{\Lambda_{l_{\min}}}\right) \,, \tag{11.25}$$

An example of weight enumerator exponent for a code with good short distance properties, the $\mathrm{LDPC}_N(3,6)$ code, is given in Fig. 11.4.

This discussion can be summarized as:

Proposition 11.1 *Consider a random linear code from the* $\mathrm{LDPC}_N(\Lambda, P)$ *ensemble with* $l_{\min} \geq 2$ *and assume* $\frac{P''(1)}{P'(1)} \frac{2\Lambda_2}{\Lambda'(1)} < 1$. *Let* $\omega_* \in]0, 1/2[$ *be the first non-trivial zero of* $\phi(\omega)$, *and consider any interval* $[\omega_1, \omega_2] \subset]0, \omega_*[$. *With high*

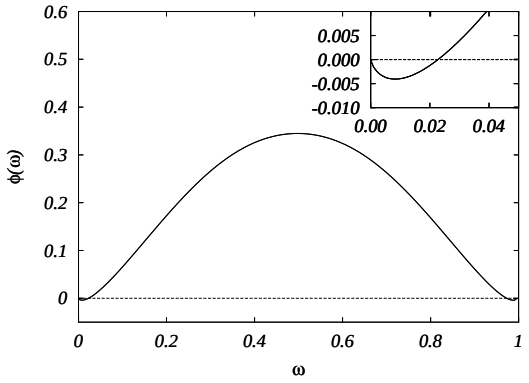

FIG. 11.4. Logarithm of the expected weight enumerator for the $\mathrm{LDPC}_N(3,6)$ ensemble. Inset: small weight region. $\phi(\omega) < 0$ for $\omega < \omega_* \sim .02$. There are no codewords except from the 'all-zeros' one in the region $\omega < \omega_*$.

{fig:WE36}

probability, there does not exist any pair of codewords with distance belonging to this interval.

Notice that our study only deals with weights $w = \omega N$ which grow linearly with N. The proposition excludes the existence of codewords of arbitrarily small ω, but it does not tell anything about possible codewords of sub-linear weight: $w = o(N)$ (for instance, with w finite as $N \to \infty$). It turns out that, if $l_{\min} \geq 3$, the code has with high probability no such codewords, and its minimum distance is at least $N\omega_*$. If on the other hand $l_{\min} = 2$, the code has typically codewords of finite weight. However (if $A < 0$), they can be eliminated without changing the code rate by an 'expurgation' procedure.

11.2.4 *Rate*

The weight enumerator can also be used to obtain a precise characterization of the rate of a $\mathrm{LDPC}_N(\Lambda, P)$ code. For $\omega = 1/2$, $x = y = z = 1$ satisfy Eqs. (11.18) and (11.19); this gives:

$$\phi(\omega = 1/2) = \left(1 - \frac{\Lambda'(1)}{P'(1)}\right) \log 2 = R_{\mathrm{des}} \log 2\,. \tag{11.26}$$

It turns out that, in most[29] of the cases of practical interest, the curve $\phi(\omega)$ has its maximum at $\omega = 1/2$ (see for instance the figures 11.2, 11.3, 11.4). In such cases the result (11.26) shows that the rate equals the design rate:

{prop:Rate}

Proposition 11.2 *Let R be the rate of a code from the $\mathrm{LDPC}_N(\Lambda, P)$ ensemble, $R_{\mathrm{des}} = 1 - \Lambda'(1)/P'(1)$ the associated design rate and $\phi(\omega)$ the function defined in Eqs. (11.17) to (11.19). Assume that $\phi(\omega)$ achieves its absolute maximum*

[29] There exist exceptions though (see the Notes section for references).

over the interval $[0,1]$ *at* $\omega = 1/2$. *Then, for any* $\delta > 0$, *there exists a positive* N*-independent constant* $C_1(\delta)$ *such that*

$$\mathbb{P}\{|R - R_{\rm des}| \geq \delta\} \leq C_1(\delta)\, 2^{-N\delta/2}\,. \tag{11.27}$$

Proof: Since we already established that $R \geq R_{\rm des}$, we only need to prove an upper bound on R. The rate is defined as $R \equiv (\log_2 \mathcal{N})/N$, where $\mathcal{N}$ is the total number of codewords. Markov's inequality gives:

$$\mathbb{P}\{R \geq R_{\rm des} + \delta\} = \mathbb{P}\{\mathcal{N} \geq 2^{N(R_{\rm des}+\delta)}\} \leq 2^{-N(R_{\rm des}+\delta)}\, \mathbb{E}\mathcal{N}\,. \tag{11.28}$$

The expectation of the number of codewords is $\mathbb{E}\mathcal{N}(w) \doteq \exp\{N\phi(w/N)\}$, and there are only $N+1$ possible values of the weight w, therefore:

$$\mathbb{E}\mathcal{N} \doteq \exp\{N \sup_{\omega\in[0,1]} \phi(\omega)\}\,, \tag{11.29}$$

As $\sup\phi(\omega) = \phi(1/2) = R_{\rm des}\log 2$ by hypothesis, there exists a constant $C_1(\delta)$ such that, for any N, $\mathbb{E}\mathcal{N} \leq C_1(\delta) 2^{N(R_{\rm des}+\delta/2)}$ for any N. Plugging this into Eq. (11.28), we get

$$\mathbb{P}\{R \geq R_{\rm des} + \delta\} \leq C_1(\delta)\, 2^{N\delta/2}\,. \tag{11.30}$$

□

11.3 Capacity of LDPC codes for the binary symmetric channel

{se:BoundsLDPC}

Our study of the weight enumerator has shown that codes from the $\mathrm{LDPC}_N(\Lambda, P)$ ensemble with $l_{\rm min} \geq 3$ have a good short distance behavior. The absence of codewords within an extensive distance $N\omega_*$ from the transmitted one, guarantees that any error (even introduced by an adversarial channel) changing a fraction of the bits smaller than $\omega_*/2$ can be corrected. Here we want to study the performance of these codes in correcting *typical* errors introduced from a given (probabilistic) channel. We will focus on the $\mathrm{BSC}(p)$ which flips each bit independently with probability $p < 1/2$. Supposing as usual that the all-zero codeword $\underline{x}^{(0)} = \underline{0}$ has been transmitted, let us call $\underline{y} = (y_1 \dots y_N)$ the received message. Its components are iid random variables taking value 0 with probability $1-p$, value 1 with probability p. The decoding strategy which minimizes the block error rate is word MAP decoding[30], which outputs the codeword closest to the channel output $\underline{y}$. As already mentioned, we don't bother about the practical implementation of this strategy and its computational complexity.

The block error probability for a code $\mathfrak{C}$, denoted by $\mathrm{P_B}(\mathfrak{C})$, is the probability that there exists a 'wrong' codeword, distinct from $\underline{0}$, whose distance to $\underline{y}$ is smaller than $d(\underline{0}, \underline{y})$. Its expectation value over the code ensemble, $\mathrm{P_B} = \mathbb{E}\,\mathrm{P_B}(\mathfrak{C})$,

[30] Since all the codewords are *a priori* equiprobable, this coincides with maximum likelihood decoding.

is an important indicator of ensemble performances. We will show that in the large N limit, codes with $l_{\min} \geq 3$ undergo a phase transition, separating a low noise phase, $p < p_{\rm ML}$, in which the limit of $\mathrm{P_B}$ is zero, from a high noise phase, $p > p_{\rm ML}$, where it is not. While the computation of $p_{\rm ML}$ is deferred to Chapter ??, we derive here some rigorous bounds which indicate that some LDPC codes have very good (i.e. close to Shannon's bound) performances under ML decoding.

{se:LBLDPC}

11.3.1 *Lower bound*

We start by deriving a general bound on the block error probability $\mathrm{P_B}(\mathfrak{C})$ on the BSC(p) channel, valid for any linear code. Let $\mathcal{N} = 2^{NR}$ be the size of the codebook $\mathfrak{C}$. By union bound:

$$\begin{aligned} \mathrm{P_B}(\mathfrak{C}) &= \mathbb{P}\left\{ \exists\, \alpha \neq 0 \quad \text{s.t.} \quad d(\underline{x}^{(\alpha)}, \underline{y}) \leq d(\underline{0}, \underline{y}) \right\} \\ &\leq \sum_{\alpha=1}^{\mathcal{N}-1} \mathbb{P}\left\{ d(\underline{x}^{(\alpha)}, \underline{y}) \leq d(\underline{0}, \underline{y}) \right\} . \end{aligned} \tag{11.31}$$

As the components of $\underline{y}$ are iid Bernoulli variables, the probability $\mathbb{P}\{d(\underline{x}^{(\alpha)}, \underline{y}) \leq d(\underline{0}, \underline{y})\}$ depends on $\underline{x}^{(\alpha)}$ only through its weight. Let $\underline{x}(w)$ be the vector formed by w ones followed by $N-w$ zeroes, and denote by $\mathcal{N}(w)$ the weight enumerator of the code $\mathfrak{C}$. Then

$$\mathrm{P_B}(\mathfrak{C}) \leq \sum_{w=1}^{N} \mathcal{N}(w)\ \mathbb{P}\left\{ d(\underline{x}(w), \underline{y}) \leq d(\underline{0}, \underline{y}) \right\} . \tag{11.32}$$

The probability $\mathbb{P}\left\{ d(\underline{x}(w), \underline{y}) \leq d(\underline{0}, \underline{y}) \right\}$ can be written as $\sum_u \binom{w}{u} p^u (1-p)^{w-u} \mathbb{I}(u \geq w/2)$, where u is the number of $y_i = 1$ in the first w components. A good bound is provided by a standard Chernov estimate. For any $\lambda > 0$:

$$\mathbb{P}\left\{ d(\underline{x}(w), \underline{y}) \leq d(\underline{0}, \underline{y}) \right\} \leq \mathbb{E} e^{\lambda [d(\underline{0}, \underline{y}) - d(\underline{x}(w), \underline{y})]} = \left[(1-p)\, e^{-\lambda} + p\, e^{\lambda} \right]^w .$$

The best bound is obtained for $\lambda = \frac{1}{2} \log(\frac{1-p}{p}) > 0$, and gives

$$\mathrm{P_B}(\mathfrak{C}) \leq \sum_{w=1}^{N} \mathcal{N}(w)\ e^{-\gamma w} . \tag{11.33}$$

where $\gamma \equiv -\log \sqrt{4p(1-p)} \geq 0$. The quantity $\sqrt{4p(1-p)}$ is sometimes referred to as **Bhattacharya parameter**.

Exercise 11.5 Consider the case of a general binary memoryless symmetric channel with transition probability $Q(y|x)$, $x \in \{0, 1\}$ $y \in \mathcal{Y} \subseteq \mathbb{R}$. First show that Eq. (11.31) remains valid if the Hamming distance $d(\underline{x}, \underline{y})$ is replaced by the log-likelihood

$$d_Q(\underline{x}|\underline{y}) = -\sum_{i=1}^{N} \log Q(y_i|x_i)\,. \tag{11.34}$$

[Hint: remember the general expressions (6.3), (6.4) for the probability $P(\underline{x}|\underline{y})$ that the transmitted codeword was $\underline{x}$, given that the received message is $\underline{y}$]. Then repeat the derivation from Eq. (11.31) to Eq. (11.33). The final expression involves $\gamma = -\log B_Q$, where the Bhattacharya parameter is defined as $B_Q = \sum_y \sqrt{Q(y|1)Q(y|0)}$.

Equation (11.33) shows that the block error probability depends on two factors: one is the weight enumerator, the second one, $\exp(-\gamma w)$ is a channel-dependent term: as the weight of the codewords increases, their contribution is scaled down by an exponential factor because it is less likely that the received message $\underline{y}$ will be closer to a codeword of large weight than to the all-zero codeword.

So far the discussion is valid for any given code. Let us now consider the average over $\mathrm{LDPC}_N(\Lambda, P)$ code ensembles. A direct averaging gives the bound:

$$\mathrm{P_B} \equiv \mathbb{E}_{\mathfrak{C}} \mathrm{P_B}(\mathfrak{C}) \leq \sum_{w=1}^{N} \overline{\mathcal{N}}(w)\; e^{-\gamma w} \doteq \exp\left\{ N \sup_{\omega \in]0,1]} [\phi(\omega) - \gamma\omega] \right\}\,. \tag{11.35}$$

As such, this expression is useless, because the $\sup_\omega[\phi(\omega) - \gamma\omega]$, being larger or equal than the value at $\omega = 0$, is positive. However, if we restrict to codes with $l_{\min} \geq 3$, we know that, with probability going to one in the large N limit, there exists no wrong codeword in the ω interval $]0, \omega_*[$. In such cases, the maximization over ω in (11.35) can be performed in the interval $[\omega_*, 1]$ instead of $]0, 1]$. (By Markov inequality, this can be proved whenever $N \sum_{w=1}^{N\omega_*-1} \overline{\mathcal{N}}(w) \to 0$ as $N \to \infty$). The bound becomes useful whenever the supremum $\sup_{\omega\in[\omega_*,1]}[\phi(\omega)-\gamma\omega] < 0$: then $\mathrm{P_B}$ vanishes in the large N limit. We have thus obtained:

{propo:LDPCUnionBound}

Proposition 11.3 *Consider the average block error rate* $\mathrm{P_B}$ *for a random code in the* $\mathrm{LDPC}_N(\Lambda, P)$ *ensemble, with* $l_{\min} \geq 3$*, used over a BSC*(p) *channel, with* $p < 1/2$*. Let* $\gamma \equiv -\log\sqrt{4p(1-p)}$ *and let* $\phi(\omega)$ *be the the weight enumerator exponent, defined in (11.4) [*$\phi(\omega)$ *can be computed using Eqs. (11.17), (11.18), and (11.19)]. If* $\phi(\omega) < \gamma\omega$ *for any* $\omega \in (0, 1]$ *such that* $\phi(\omega) \geq 0$*, then* $\mathrm{P_B} \to 0$ *in the large block-length limit.*

This result has a pleasing geometric interpretation which is illustrated in Fig. 11.5 for the $(3, 6)$ regular ensemble. As p increases from 0 to $1/2$, γ decreases

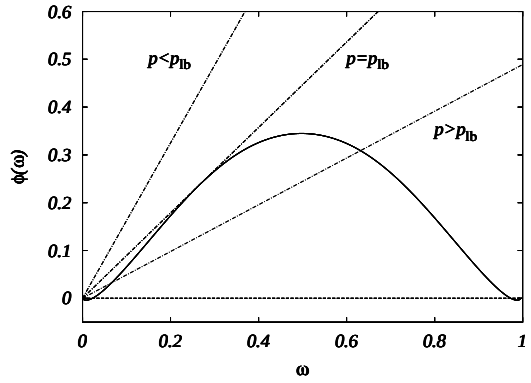

FIG. 11.5. Geometric construction yielding the lower bound on the threshold for reliable communication for the $\mathrm{LDPC}_N(3,6)$ ensemble used over the binary symmetric channel. In this case $p_{\mathrm{LB}} \approx 0.0438737$. The other two lines refer to $p = 0.01 < p_{\mathrm{LB}}$ and $p = 0.10 > p_{\mathrm{LB}}$. {fig:UnionBound36}

from $+\infty$ to 0. The condition $\phi(\omega) < \gamma\omega$ can be rephrased by saying that the weight enumerator exponent $\phi(\omega)$ must lie below the straight line of slope γ through the origin. Let us call p_{LB} the smallest value of p such that the line $\gamma\omega$ touches $\phi(\omega)$.

The geometric construction implies $p_{\mathrm{LB}} > 0$. Furthermore, for p large enough Shannon's Theorem implies that $\mathrm{P_B}$ is bounded away from 0 for any non-vanishing rate $R > 0$. The **ML threshold** p_{ML} for the ensemble $\mathrm{LDPC}_N(\Lambda, P)$ can be defined as the largest (or, more precisely, the supremum) value of p such that $\lim_{N\to\infty} \mathrm{P_B} = 0$. This definition has a very concrete practical meaning: for any $p < p_{\mathrm{ML}}$ one can communicate with an arbitrarily small error probability, by using a code from the $\mathrm{LDPC}_N(\Lambda, P)$ ensemble provided N is large enough. Proposition 11.3 then implies:

$$p_{\mathrm{ML}} \geq p_{\mathrm{LB}} \,. \tag{11.36}$$

In general one expects $\lim_{N\to\infty} \mathrm{P_B}$ to exist (and to be strictly positive) for $p > p_{\mathrm{ML}}$. However, there exists no proof of this statement.

It is interesting to notice that, at $p = p_{\mathrm{LB}}$, our upper bound on $\mathrm{P_B}$ is dominated by codewords of weight $w \approx N\tilde{\omega}$, where $\tilde{\omega} > 0$ is the value where $\phi(\omega) - \gamma\omega$ is maximum (which is larger than ω_*). This suggests that, each time an error occurs, a finite fraction of the bits are decoded incorrectly and this fraction fluctuates little from transmission to transmission (or, from code to code in the ensemble). The geometric construction also suggests the less obvious (but essentially correct) guess that this fraction jumps discontinuously from 0 to a finite value when p crosses the critical value p_{ML}.

Exercise 11.6 Let us study the case $l_{\min} = 2$. Proposition 11.3 is no longer valid, but we can still apply Eq. (11.35). (*i*) Consider the $(2, 4)$ ensemble whose weight enumerator exponent is plotted in Fig. 11.3, the small weight behavior being given by Eq. (11.24). At small enough p, it is reasonable to assume that the block error rate is dominated by small weight codewords. Estimate $\mathrm{P_B}$ using Eq. (11.35) under this assumption. (*ii*) Show that the assumption breaks down for $p \geq p_{\rm loc}$, where $p_{\rm loc} \leq 1/2$ solves the equation $3\sqrt{4p(1-p)} = 1$. (*iii*) Discuss the case of a general code ensemble with $l_{\min} = 2$, and $\phi(\omega)$ concave for $\omega \in [0, 1]$. (*iv*) Draw a weight enumerator exponent $\phi(\omega)$ such that the assumption of low-weight codewords dominance breaks down before $p_{\rm loc}$. (*v*) What do you expect of the average bit error rate $\mathrm{P_b}$ for $p < p_{\rm loc}$? And for $p > p_{\rm loc}$?

Exercise 11.7 Discuss the qualitative behavior of the block error rate for the cases where $l_{\min} = 1$.

11.3.2 *Upper bound*

{se:UBLDPC}

Let us consider as before the communication over a BSC(p), but restrict for simplicity to regular codes $\mathrm{LDPC}_N(l, k)$. Gallager has proved the following upper bound:

{thm:GallUB}

Theorem 11.4 *Let $p_{\rm ML}$ be the threshold for reliable communication over the binary symmetric channel using codes from the $\mathrm{LDPC}_N(l, k)$, with design rate $R_{\rm des} = 1 - k/l$. Then $p_{\rm ML} \leq p_{\rm UB}$, where $p_{\rm UB} \leq 1/2$ is the solution of*

$$\mathcal{H}(p) = (1 - R_{\rm des})\mathcal{H}\left(\frac{1 - (1-2p)^k}{2}\right) , \tag{11.37}$$

We shall not give a full proof of this result, but we show in this section a sequence of heuristic arguments which can be turned into a proof. The details can be found in the original literature.

Assume that the all-zero codeword $\underline{\mathtt{0}}$ has been transmitted and that a noisy vector $\underline{y}$ has been received. The receiver will look for a vector $\underline{x}$ at Hamming distance about Np from $\underline{y}$, and satisfying all the parity check equations. In other words, let us denote by $\underline{z} = \mathbb{H}\underline{x}$, $\underline{z} \in \{\mathtt{0}, \mathtt{1}\}^M$, (here $\mathbb{H}$ is the parity check matrix and multiplication is performed modulo 2), the **syndrome**. This is a vector with M components. If $\underline{x}$ is a codeword, all parity checks are satisfied, and we have $\underline{z} = \underline{\mathtt{0}}$. There is at least one vector $\underline{x}$ fulfilling these conditions (namely $d(\underline{x}, \underline{y}) \approx Np$, and $\underline{z} = \underline{\mathtt{0}}$): the transmitted codeword $\underline{\mathtt{0}}$. Decoding is successful only if it is the unique such vector.

The number of vectors $\underline{x}$ whose Hamming distance from $\underline{y}$ is close to Np is approximatively $2^{N\mathcal{H}(p)}$. Let us now estimate the number of distinct syndromes $\underline{z} = \mathbb{H}\underline{x}$, when $\underline{x}$ is on the sphere $d(\underline{x}, \underline{y}) \approx Np$. Writing $\underline{x} = \underline{y} \oplus \underline{x}'$, this is equivalent to counting the number of distinct vectors $\underline{z}' = \mathbb{H}\underline{x}'$ when the weight

Table 11.1 *Bounds on the threshold for reliable communication over the BSC(p) channel using* $\mathrm{LDPC}_N(l,k)$ *ensembles. The third column is the rate of the code, the fourth and fifth columns are, respectively, the lower bound of Proposition 11.3 and the upper bound of Theorem 11.4. The sixth column is an improved lower bound by Gallager, and the last one is the Shannon limit.*

l	k	$R_{\rm des}$	LB of Sec. 11.3.1	Gallager UB	Gallager LB	Shannon limit
3	4	1/4	0.1333161	0.2109164	0.2050273	0.2145018
3	5	2/5	0.0704762	0.1397479	0.1298318	0.1461024
3	6	1/2	0.0438737	0.1024544	0.0914755	0.1100279
4	6	1/3	0.1642459	0.1726268	0.1709876	0.1739524
5	10	1/2	0.0448857	0.1091612	0.1081884	0.1100279

{TableLDPCBSC}

of $\underline{x}'$ is about Np. It is convenient to think of $\underline{x}'$ as a vector of N iid Bernoulli variables of mean p: we are then interested in the number of distinct *typical* vectors $\underline{z}'$. Notice that, since the code is regular, each entry z'_i is a Bernoulli variable of parameter

$$p_k = \sum_{n\ \mathrm{odd}}^{k} \binom{k}{n} p^n (1-p)^{k-n} = \frac{1-(1-2p)^k}{2}\,. \tag{11.38}$$

If the bits of $\underline{z}'$ were independent, the number of typical vectors $\underline{z}'$ would be $2^{N(1-R_{\rm des})\mathcal{H}(p_k)}$ (the dimension of $\underline{z}'$ being $M = N(1-R_{\rm des})$). It turns out that correlations between the bits decrease this number, so we can use the iid estimate to get an upper bound.

Let us now assume that for each $\underline{z}$ in this set, the number of reciprocal images (i.e. of vectors $\underline{x}$ such that $\underline{z} = \mathbb{H}\underline{x}$) is approximatively the same. If $2^{N\mathcal{H}(p)} \gg 2^{N(1-R_{\rm des})\mathcal{H}(p_k)}$, for each $\underline{z}$ there is an exponential number of vectors $\underline{x}$, such that $\underline{z} = \mathbb{H}\underline{x}$. This will be true, in particular, for $\underline{z} = \underline{0}$: the received message is therefore not uniquely decodable. In the alternative situation most of the vectors $\underline{z}$ correspond to (at most) a single $\underline{x}$. This will be the case for $\underline{z} = \underline{0}$: decoding can be successful.

11.3.3 *Summary of the bounds*

In Table 11.1 we consider a few regular $\mathrm{LDPC}_N(\Lambda, P)$ ensembles over the $\mathrm{BSC}(p)$ channel. We show the window of possible values of the noise threshold $p_{\rm ML}$, using the lower bound of Proposition 11.3 and the upper bound of Theorem 11.4. In most cases, the comparison is not satisfactory (the gap from capacity is close to a factor 2). A much smaller uncertainty is achieved using an improved lower bound again derived by Gallager, based on a refinement of the arguments in the previous Section. However, as we shall see in next Chapters, neither of the bounds is tight. Note that these codes get rather close to Shannon's limit, especially when k, l increase.

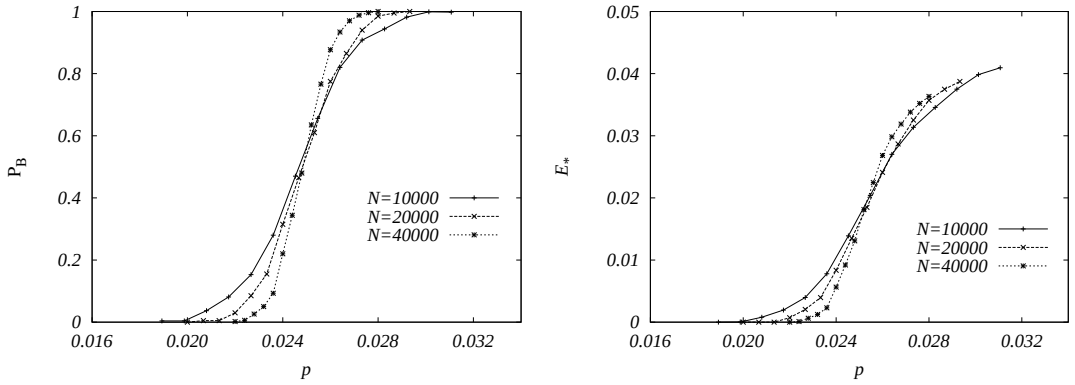

FIG. 11.6. Performances of the bit-flipping decoding algorithm on random codes from the $(5, 10)$ regular LDPC ensemble, used over the BCS(p) channel. On the left: block error rate. On the right residual number of unsatisfied parity checks after the algorithm halted. Statistical error bars are smaller than symbols. {fig:Flip510}

Exercise 11.8 Let $p_{\rm Sh}$ be the upper bound on $p_{\rm ML}$ provided by Shannon channel coding Theorem. Explicitly $p_{\rm Sh} \leq 1/2$ is the solution of $\mathcal{H}(p) = 1 - R$. Prove that, if $R = R_{\rm des}$ (as is the case with high probability for LDPC$_N(l, k)$ ensembles) $p_{\rm UB} < p_{\rm Sh}$.

11.4 A simple decoder: bit flipping {se:BitFlippingLDPC}

So far we have analyzed the behavior of LDPC ensembles under the optimal (ML) decoding strategy. However there is no known way of implementing this decoding with a fast algorithm. The naive algorithm goes through each codeword $\underline{x}^{(\alpha)}$, $\alpha = 0, \dots 2^{NR} - 1$ and finds the one of greatest likelihood $Q(\underline{y}|\underline{x}^{(\alpha)})$ (since all the codeword are *a priori* equiprobable, this is in fact the same as word MAP decoding). However this approach takes a time which grows exponentially with the block-length N. For large N (which is the regime where the error rate becomes close to optimal), this is unpractical.

LDPC codes are interesting because there exist fast sub-optimal decoding algorithms with performances close to the theoretical optimal performance, and therefore close to Shannon's limit. Here we show one example of a very simple decoding method, called the **bit flipping** algorithm. We have received the message $\underline{y}$ and try to find the sent codeword $\underline{x}$ by:

Bit-flipping decoder

0. Set $\underline{x}(0) = \underline{y}$.
1. Find a bit belonging to more unsatisfied than satisfied parity checks.
2. If such a bit exists, flip it: $x_i(t+1) = x_i(t) \oplus 1$. Keep the other bits: $x_j(t+1) = x_j(t)$ for all $j \neq i$. If there is no such bit, return $\underline{x}(t)$ and halt.

`3. Repeat steps 2 and 3.`

The bit to be flipped is usually chosen uniformly at random among the ones satisfying the condition at step `1`. However this is irrelevant in the analysis below.

Exercise 11.9 Consider a code from the (l, k) regular LDPC ensemble (with $l \geq 3$). Assume that the received message differs from the transmitted one only in one position. Show that the bit-flipping algorithm always corrects such an error.

Exercise 11.10 Assume now that the channel has introduced two errors. Draw the factor graph of a regular (l, k) code for which the bit-flipping algorithm is unable to recover such an error event. What can you say of the probability of this type of graphs in the ensemble?

In order to monitor the bit-flipping algorithm, it is useful to introduce the 'energy':

$$E(t) \equiv \text{ Number of parity check equations not satisfied by } \underline{x}(t)\,. \quad (11.39)$$

This is a non-negative integer, and if $E(t) = 0$ the algorithm is halted and its output is $\underline{x}(t)$. Furthermore $E(t)$ cannot be larger than the number of parity checks M and decreases (by at least one) at each cycle. Therefore, the algorithm complexity is $O(N)$ (this is a commonly regarded as the ultimate goal for many communication problems).

It remains to be seen if the output of the bit-flipping algorithm is related to the transmitted codeword. In Fig. 11.6 we present the results of a numerical experiment. We considered the $(5, 10)$ regular ensemble and generated about 1000 random code and channel realizations for each value of the noise in some mesh. Then, we applied the above algorithm and traced the fraction of successfully decoded blocks, as well as the residual energy $E_* = E(t_*)$, where t_* is the total number of iterations of the algorithm. The data suggests that bit-flipping is able to overcome a finite noise level: it recovers the original message with high probability when less than about 2.5% of the bits are corrupted by the channel. Furthermore, the curves for $\mathrm{P_B^{bf}}$ under bit-flipping decoding become steeper and steeper as the system size is increased. It is natural to conjecture that asymptotically, a phase transition takes place at a well defined noise level $p_{\rm bf}$: $\mathrm{P_B^{bf}} \to 0$ for $p < p_{\rm bf}$ and $\mathrm{P_B^{bf}} \to 1$ for $p > p_{\rm bf}$. Numerically $p_{\rm bf} = 0.025 \pm 0.005$.

This threshold can be compared with the one for ML decoding: The results in Table 11.1 imply $0.108188 \leq p_{\rm ML} \leq 0.109161$ for the $(5, 10)$ ensemble. Bit-flipping is significantly sub-optimal, but is still surprisingly good, given the extreme simplicity of the algorithm.

Can we provide any *guarantee* on the performances of the bit-flipping decoder? One possible approach consists in using the expansion properties of the underlying factor graph. Consider a graph from the (l, k) ensemble. We say that it is an (ε, δ)-**expander** if, for any set U of variable nodes such that $|U| \leq N\varepsilon$,

the set $|D|$ of neighboring check nodes has size $|D| \geq \delta|U|$. Roughly speaking, if the factor graph is an expander with a large **expansion constant** δ, any small set of corrupted bits induces a large number of unsatisfied parity checks. The bit-flipping algorithm can exploit these checks to successfully correct the errors.

It turns out that random graphs are very good expanders. This can be understood as follows. Consider a fixed subset U. As long as U is small, the subgraph induced by U and the neighboring factor nodes D is a tree with high probability. If this is the case, elementary counting shows that $|D| = (l-1)|U| + 1$. This would suggest that one can achieve an expansion factor (close to) $l-1$, for small enough ε. Of course this argument have several flaws. First of all, the subgraph induced by U is a tree only if U has sub-linear size, but we are interested in all subsets U with $|U| \leq \varepsilon N$ for some fixed N. Then, while most of the small subsets U are trees, we need to be sure that *all* subsets expand well. Nevertheless, one can prove that the heuristic expansion factor is essentially correct:

Proposition 11.5 *Consider a random factor graph $\mathcal{F}$ from the (l,k) ensemble. Then, for any $\delta < l-1$, there exists a constant $\varepsilon = \varepsilon(\delta; l, k) > 0$, such that $\mathcal{F}$ is a (ε, δ) expander with probability approaching 1 as $N \to \infty$.*

In particular, this implies that, for $l \geq 5$, a random (l,k) regular factor graph is, with high probability a $(\varepsilon, \frac{3}{4}\, l)$ expander. In fact, this is enough to assure that the code will perform well at low noise level:

Theorem 11.6 *Consider a regular (l,k) LDPC code $\mathfrak{C}$, and assume that the corresponding factor graph is an $(\varepsilon, \frac{3}{4}\, l)$ expander. Then, the bit-flipping algorithm is able to correct any pattern of less then $N\varepsilon/2$ errors produced by a binary symmetric channel. In particular $\mathrm{P_B}(\mathfrak{C}) \to 0$ for communication over a BSC(p) with $p < \varepsilon/2$.*

Proof: As usual, we assume the channel input to be the all-zeros codeword $\underline{0}$. We denote by $w = w(t)$ the weight of $\underline{x}(t)$ (the current configuration of the bit-flipping algorithm), and by $E = E(t)$ the number of unsatisfied parity checks, as in Eq. (11.39). Finally, we call F the number of *satisfied* parity checks among the ones which are neighbors of at least one corrupted bit in $\underline{x}(t)$ (a bit is 'corrupted' if it takes value 1).

Assume first that $0 < w(t) \leq N\varepsilon$ at some time t. Because of the expansion property of the factor graph, we have $E + F > \frac{3}{4}\, l\, w$. On the other hand, every unsatisfied parity check is the neighbor of at least one corrupted bit, and every satisfied check which is the neighbor of some corrupted bit must involve at least two of them. Therefore $E + 2F \leq l\, w$. Eliminating F from the above inequalities, we deduce that $E(t) > \frac{1}{2}\, l\, w(t)$. Let $E_i(t)$ be the number of unsatisfied checks involving bit x_i. Then:

$$\sum_{i:x_i(t)=1} E_i(t) \geq E(t) > \frac{1}{2}\, l\, w(t)\,. \tag{11.40}$$

Therefore, there must be at least one bit having more unsatisfied than satisfied neighbors, and the algorithm does not halt.

Let us now start the algorithm with $w(0) \leq N\varepsilon/2$. It must halt at some time t_*, either with $E(t_*) = w(t_*) = 0$ (and therefore decoding is successful), or with $w(t_*) \geq N\varepsilon$. In this second case, as the weight of $\underline{x}(t)$ changes by one at each step, we have $w(t_*) = N\varepsilon$. The above inequalities imply $E(t_*) > Nl\varepsilon/2$ and $E(0) \leq lw(0) \leq Nl\varepsilon/2$. This contradicts the fact that $E(t)$ is a strictly decreasing function of t. Therefore the algorithm, started with $w(0) \leq N\varepsilon/2$ ends up in the $w = 0$, $E = 0$ state. □

The approach based on expansion of the graph has the virtue of pointing out one important mechanism for the good performance of LDPC codes, namely the local tree-like structure of the factor graph. It also provides explicit lower bounds on the critical noise level $p_{\rm bf}$ for bit-flipping. However, these bounds turn out to be quite pessimistic. For instance, in the case of the $(5, 10)$ ensemble, it has been proved that a typical factor graph is an $(\varepsilon, \frac{3}{4}l) = (\varepsilon, \frac{15}{4})$ expander for $\varepsilon < \varepsilon_* \approx 10^{-12}$. On the other hand, numerical simulations, cf. Fig. 11.6, show that the bit flipping algorithm performs well up noise levels much larger than $\varepsilon_*/2$.

Notes

Modern (post-Cook Theorem) complexity theory was first applied to coding by (Berlekamp, McEliecee and van Tilborg, 1978) who showed that maximum likelihood decoding of linear codes is NP-hard.

LDPC codes were first introduced by Gallager in his Ph.D. thesis (Gallager, 1963; Gallager, 1962), which is indeed older than these complexity results. See also (Gallager, 1968) for an extensive account of earlier results. An excellent detailed account of modern developments is provided by (Richardson and Urbanke, 2006).

Gallager proposal did not receive enough consideration at the time. One possible explanation is the lack of computational power for simulating large codes in the sixties. The rediscovery of LDPC codes in the nineties (MacKay, 1999), was (at least in part) a consequence of the invention of Turbo codes by (Berrou and Glavieux, 1996). Both these classes of codes were soon recognized to be prototypes of a larger family: codes on graphs.

The major technical advance after this rediscovery has been the introduction of irregular ensembles (Luby, Mitzenmacher, Shokrollahi, Spielman and Stemann, 1997; Luby, Mitzenmacher, Shokrollahi and Spielman, 1998). There exist no formal proof of the 'equivalence' (whatever this means) of the various ensembles in the large block-length limit. But as we will see in Chapter **??**, the main property that enters in the analysis of LDPC ensembles is the local tree-like structure of the factor graph as described in Sec. 9.5.1; and this property is rather robust with respect to a change of the ensemble.

Gallager (Gallager, 1963) was the first to compute the expected weight enumerator for regular ensembles, and to use it in order to bound the threshold for reliable communication. The general case ensembles was considered in (Litsyn and Shevelev, 2003; Burshtein and Miller, 2004; Di, Richardson and Urbanke,

2004*b*). It turns out that the expected weight enumerator coincides with the typical one to leading exponential order for regular ensembles (in statistical physics jargon: the annealed computation coincides with the quenched one). This is not the case for irregular ensembles, as pointed out in (Di, Montanari and Urbanke, 2004*a*).

Proposition 11.2 is essentially known since (Gallager, 1963). The formulation quoted here is from (Méasson, Montanari and Urbanke, 2005*a*). This paper contains some examples of 'exotic' LDPC ensembles such that the maximum of the expected weight enumerator is at weight $w = N\omega_*$, with $\omega_* \neq 1/2$.

A proof of the upper bound 11.4 can be found in (Gallager, 1963). For some recent refinements, see (Burshtein, Krivelevich, Litsyn and Miller, 2002).

Bit-flipping algorithms played an important role in the revival of LDPC codes, especially following the work of Sipser and Spielman (Sipser and Spielman, 1996). These authors focused on explicit code construction based on expander graph. They also provide bounds on the expansion of random $\mathrm{LDPC}_N(l, k)$ codes. The lower bound on the expansion mentioned in Sec. 11.4 is taken from (Richardson and Urbanke, 2006).