2

Coding for Erasures and Fountain Codes

A coordinate position in a received word is said to be an erasure if the receiver is using a detection algorithm that is unable to decide which symbol was transmitted in that position and outputs an erasure symbol such as E rather than risk making an error, i.e., outputting an incorrect symbol. One might describe an erasure as an error whose position is known. For binary information symbols the two most common discrete memoryless channels are shown in Figure 1.1, the binary erasure channel (BEC) and the binary symmetric channel (BSC), introduced in Chapter 1. The symbols p and δ will generally refer to the channel crossover probability for the BSC and erasure probability for the BEC, respectively. Both symbols sometimes occur with other meanings as will be noted. Each such channel has a capacity associated with it which is the maximum rate at which information (per channel use) can be sent through the channel error-free, as discussed in Chapter 1. The subject of error-correcting codes arose to meet the challenge of realizing such performance.

It is to be emphasized that two quite different channel error models are used in this chapter. The BEC will be the channel of interest in the first part of this chapter. Thus a codeword (typically of a linear code) is received which contains a mix of correct received symbols and erased symbols. The job of the code design and decoder algorithm is then to "fill in" or interpolate the erased positions with original transmitted symbols, noting that in such a model the unerased positions are assumed correct.

Codes derived for the BEC led to the notion of irregular distribution codes where the degrees of variable and check nodes of the code Tanner graph, to be introduced shortly, are governed by a probability distribution. These in turn led to fountain codes, which is the interest of the last section of the chapter. In such channels each transmitted packet is typically a linear combination of information packets over some fixed finite field. The receiver gathers a

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

sufficient number of transmitted packets (assumed without errors) until it is able to retrieve the information packets by, e.g., some type of matrix inversion algorithm on the set of packets received. The retriever then does not care which particular packets are received, just that they receive a sufficient number of them to allow the decoding algorithm to decode successfully. This is often described as a "packet loss" channel, in that coded packets transmitted may be lost in transmission due to a variety of network imperfections such as buffer overflow or failed server nodes, etc. Such a packet loss situation is not modeled by the DMC models considered.

While the two channel models examined in this chapter are quite different, it is their common heritage that suggested their discussion in the same chapter.

2.1 Preliminaries

It is convenient to note a few basic results on coding and DMCs for future reference. Suppose $C = (n,k,d)_q$ is a linear block code over the finite field of q elements $\mathbb{F}_q$, designating a linear code that has k *information symbols* (dimension k) and $(n-k)$ *parity-check symbols* and minimum distance d. Much of this volume is concerned with binary-input channels and $q = 2$.

Suppose a codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ is transmitted on a BEC and the received word is $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$ which has e erasures in positions $\mathcal{E} \subset \{1, 2, \ldots, n\}$, $|\mathcal{E}| = e$. Then $r_i = E$ for $i \in \mathcal{E}$ for E the erasure symbol. The unerased symbols received are assumed correct.

A parity-check matrix of the code is an $(n-k) \times n$ matrix H over $\mathbb{F}_q$ such that

$$\boldsymbol{H} \cdot \boldsymbol{c}^t = \mathbf{0}_{n-k}^t$$

for any codeword $\boldsymbol{c}$ where $\mathbf{0}_{n-k}$ is the all-zero $(n-k)$-tuple, a row vector over $\mathbb{F}_q$. If the columns of $\boldsymbol{H}$ are denoted by $\boldsymbol{h}^{(i)}$, $i = 1, 2, \ldots, n$, then $\boldsymbol{H} \cdot \boldsymbol{c}^t$ is the sum of columns of $\boldsymbol{H}$ multiplied by the corresponding coordinates of the codeword, adding to the all-zero column $(n-k)$-vector $\mathbf{0}^t$. Similarly let $\boldsymbol{H}_e$ be the $(n-k) \times e$ submatrix of columns of $\boldsymbol{H}$ corresponding to the erased positions and $\boldsymbol{c}_e$ be the e-tuple of the transmitted codeword on the erased positions. Then

$$\boldsymbol{H}_e \cdot \boldsymbol{c}_e^t = -\boldsymbol{y}_e^t$$

where $\boldsymbol{y}_e^t$ is the $(n-k)$-tuple corresponding to the weighted sum of columns of $\boldsymbol{H}$ in the nonerased positions, i.e., columns of $\boldsymbol{H}$ multiplied by the known (unerased) positions of the received codeword.

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

As long as $e \leq d-1$ the above matrix equation can be solved uniquely for the erased word positions, i.e., $\boldsymbol{c}_e$. However, this is generally a task of cubic complexity in codeword length, i.e., $O(n^3)$. The work of this chapter will show how a linear complexity with codeword length can be achieved with high probability.

This chapter will deal exclusively with binary codes and the only arithmetic operation used will be that of XOR (exclusive or), either of elements of $\mathbb{F}_2$ or of packets of n bits in $\mathbb{F}_2^n$. Thus virtually all of the chapter will refer to packets or binary symbols (bits) equally, the context being clear from the problem of interest.

It is emphasized that there are two different types of coding considered in this chapter. The first is the use of linear block codes for erasure correction while the second involves the use of *fountain codes* on a packet loss channel.

Virtually all of this chapter will use the notion of a bipartite graph to represent the various linear codes considered, a concept used by Tanner in his prescient works [37, 38, 39]. A bipartite graph is one with two sets of vertices, say U and V and an edge set E, with no edges between vertices in the same set. The graph will be called (c,d)-regular bipartite if the vertices in U have degree c and those of V have degree d. Since $|U| = n$, then $|V| = (c/d)n$. The U set of vertices will be referred to as the left vertices and V the right vertices. Bipartite graphs with irregular degrees will also be of interest later in the chapter.

There is a natural connection between a binary linear code and an $(n-k)\times n$ parity-check matrix and a bipartite graph. Often the left vertices of the bipartite code graph are associated with the entire n codeword coordinate positions and referred to as the *variable* or *information* nodes or vertices. Equivalently they represent the columns of the parity-check matrix. Similarly the $(n-k)$ right nodes or vertices are the *constraint* or *check* nodes which represent the rows of the parity-check matrix. The edges of the graph correspond to the ones in the check matrix in the corresponding rows and columns. Such a graphical representation of the code is referred to as the *Tanner graph* of the code, a notion that will feature prominently in many of the chapters.

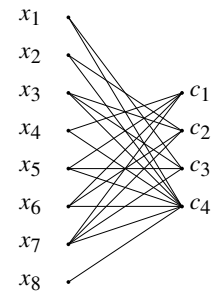
The binary parity-check matrix of the code is an alternate view of the incidence matrix of the bipartite graph. The following illustrates the Tanner graph associated with the parity-check matrix for a Hamming $(8,4,4)_2$ code which is used in Example 2.3 shown also in Figures 2.1 and 2.3.

As a second graph representation of a binary linear code, it is equally possible to have the left nodes of the graph as the k information nodes and the $(n-k)$ right nodes as the check nodes and this is the view for most of the next section. As a matter of convenience this representation is referred to as the

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

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

(a)



(b)

Figure 2.1 (a) The Hamming $(8,4,4)_2$ code and (b) its Tanner graph

normal graph representation of a code in this work, although some literature on coding has a different meaning for the term "normal." The Tanner graph representation seems more common in current research literature.

The next section describes a class of linear binary codes, the *Tornado codes*, which use a cascade of (normal) bipartite graphs and a very simple decoding algorithm for correcting erasures. To ensure the effectiveness of decoding it is shown how the graphs in the cascade can be chosen probabilistically and this development introduced the notion of irregular distributions of vertex/edge degrees of the left and right vertices of each graph in the cascade. This notion has proved important in other coding contexts, e.g., in the construction of LDPC codes to be considered in Chapter 3.

Section 2.3 introduces the notion of *LT codes*, standing for *Luby transform*, the first incarnation of the important notion of a *fountain code* where coded packets are produced at random by linearly XORing a number of information packets, according to a probability law designed to ensure efficient decodability. That section also considers *Raptor* codes, a small but important modification of LT codes that has been standardized as the most effective way to achieve large downloads over the Internet.

The notion of Tornado codes introduced the idea of choosing random bipartite graphs to effect erasure decoding. Such a notion led to decoding algorithms of fountain codes where a file is comprised of randomly linearly encoded pieces of the file. These decoding algorithms achieve linear complexity rather than the normally cubic complexity associated with Gaussian elimination with a certain probability of failure. As noted, there is no notion of "erasure" with fountain codes as there is with Tornado codes. A significant feature of these fountain codes is that they do not require requests for retransmission of missing packets. This can be a crucial feature in some systems since such

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

requests could overwhelm the source trying to satisfy requests from a large number of receivers, a condition referred to as *feedback implosion*. This is the *multicast* situation where a transmitter has a fixed number of packets (binary n-tuples) which it wishes to transmit to a set of receivers through a network. It is assumed receivers are not able to contact the transmitter to make requests for retransmissions of missing packets. The receiver is able to construct the complete set of transmitted information packets from the reception of *any* sufficiently large set of coded packets, not a specific set. Typically the size of the set of received packets is just slightly larger than the set of information packets itself, to ensure successful decoding with high probability, leading to a very efficient transmission and decoding process.

Most of the algorithms in the chapter will have linear complexity (in codeword length or the number of information symbols), making them very attractive for implementation.

2.2 Tornado Codes and Capacity-Achieving Sequences

The notion of Tornado codes was first noted in [7] and further commented on in [2] with a more complete account in [10] (an updated version of [7]). While not much cited in recent works they introduced novel and important ideas that have become of value in LDPC coding and in the formulation of fountain codes. At the very least they are an interesting chapter in coding theory and worthy of some note.

For this section it is assumed transmission is on the BEC. Since only the binary case is of interest the only arithmetic operation will be the XOR between code symbols and thus the code symbols (coordinate positions) can be assumed to be either bits or sequences of bits (packets). Any received packet is assumed correct – no errors in it. Packets that are erased will be designated with a special symbol, e.g., E (either a bit or packet) when needed.

Tornado codes can be described in three components: a cascade of a sequence of bipartite graphs; a (very simple) decoding algorithm for each stage, as decoding proceeds from the right to the left and a probabilistic design algorithm for each of the bipartite graphs involved. As mentioned, the design algorithm has proven influential in other coding contexts.

Consider the first bipartite graph B_0 of Figure 2.2 with k left vertices, associated with the k information packets and βk, $\beta < 1$, right nodes, the *check* nodes (so each code in the cascade is a normal graph – one of the few places in these chapters using normal graphs). It is assumed the codes are binary and, as noted, whether bits or packets are used for code symbols is immaterial.

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

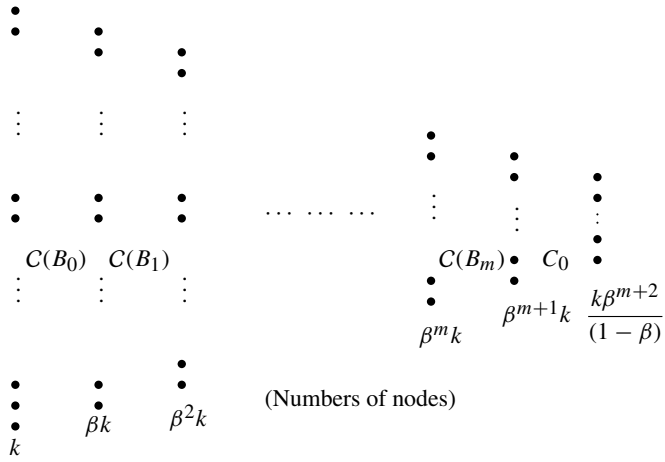


Figure 2.2 The cascade construction of the Tornado code $C(B_0, B_1, \ldots, B_m, C_0)$ – code rate = $1 - \beta$, $\beta \in (0, 1)$, code length $k/(1 - \beta)$. Graph edges are not shown for clarity

For the sake of concreteness packets will be assumed. The parity-check matrix could have been used here but the equivalent graph is more convenient. The terms vertices and nodes are used interchangeably. The term packets will often be used to emphasize the application of the ideas to general networks. For a given set of information packets the check packets are a simple XOR of the connected information packets. Note that the edges have not been included in the figure of this cascade of graphs as it tended to obscure other aspects of importance.

To show how decoding could take place on such a graph, consider the first graph on the right of the cascade, C_0, and assume for the moment that all check packets (the furthest right nodes) have been received without erasures and consider the following simple decoding algorithm [7, 10].

Algorithm 2.1 Given the value of a check packet and all but one of its neighbor information packets, set the missing information packet to the XOR of the check packet and its neighbors.

Thus to decode, one searches for a check node at each stage of the decoding satisfying the criterion – namely that only one of its neighbor variable nodes is erased, and decodes that packet, finding the value of the missing information packet. That information packet is then XORed with its neighbor check packets. Each stage proceeds similarly. For the decoding to be successful, there must exist suitable check nodes of degree one at each stage of the decoding

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

until all variable (information) packets for that stage are found – which are then the check nodes for the next (left) stage of the graph.

This first code in the cascade (on the left) is designated as $C(B_0)$ and consists of the k left information packets and βk right check packets. The next bipartite graph in the cascade will consist of the βk left nodes (the check packets of $C(B_0)$ and the "information packets" of $C(B_1)$) and $\beta^2 k$ check packets on these for some fixed β. This stage of the "code" will be designated as $C(B_1)$. The process of constructing the cascade continues, each graph consisting of the previous check nodes and constructing a new set of check nodes of size β times that of the set of left nodes. The cascade components are designated $C(B_i), i = 0, 1, \ldots, m$. Each stage of the cascade has $k\beta^i$ left nodes and $k\beta^{i+1}$, $i = 0, 1, 2, \ldots, m$ check nodes. The cascade is concluded with a conventional block code C_0 which is assumed to be a simple conventional erasure-correcting code with $k\beta^{m+1}$ left nodes and $k\beta^{m+2}/(1-\beta)$ check nodes, capable of correcting any fraction of β erasures with high probability. This code is not specified further – any of a number of standard codes will suffice. The whole cascade is designated as $C(B_0, B_1, \ldots, B_m, C_0)$ (see Figure 2.2). As mentioned, the edges of the graph have been omitted as they would tend to obscure the simple process involved. The code has k information packets and

$$\sum_{i=1}^{m+1} k\beta^i + k\beta^{m+2}/(1-\beta) = k\beta/(1-\beta), \quad \beta \in (0,1)$$

check packets and hence the overall code has rate $1-\beta$.

To decode this code one starts at the extreme right graph/code C_0. It is assumed this ordinary erasure-correcting code is strong enough to correct all erasures in its parity checks. At each stage of decoding, as it progresses from right to left, it will be assumed the check nodes are all determined. Applying the above decoding algorithm, the variable node values of C_0 can be determined as long as there is a check node satisfying the criterion noted for the decoding algorithm, i.e., that all but one of its neighbor variables are determined. If all such variable nodes of C_0 are determined, one proceeds to decoding $C(B_m)$, all of whose check nodes (variable nodes of C_0) are now known. Applying the decoding algorithm to these determines its variable nodes – assuming the supply of suitable check nodes is not exhausted prior to completion. The process continues to the left. The decoding is successful only if each stage is successful, i.e., at each stage a suitable check node is always available with high probability until all information nodes for that stage are determined.

The problem remains as to how to choose the component graphs in this cascade (i.e., the variable/check node connections) so that the algorithm will complete with high probability.

The above assumed a graph representation for each stage of the code in what has been designated as the normal form with information nodes on the left and check nodes on the right. It is clear that there are equivalent versions of the decoding algorithm for a Tanner graph representation of a code, with the n codeword symbols (information and check symbols) on the left and $(n-k)$ check symbols associated with the nodes on the right (a single stage only), with erasures randomly occurring possibly in both variable and check nodes. Much of the recent work on coding uses the Tanner graph representation and a modification of the above discussion for such graphs might be as follows:

Algorithm 2.2 Identify the received codeword symbols (or packets) (with erasures) with the n variable (left) nodes and associate a register with each of the $(n-k)$ check nodes on the right, each initially set to the all-zero packet. XOR the value of each nonerased variable node to its check neighbors and delete those variable nodes and edges emanating from them – thus only variable nodes that have been erased remain at this stage. If there is a check node of degree one, transfer its contents to its variable node neighbor, say v, and then XOR this value to the check neighbors of v and remove all associated edges. Continue in this manner until there are no check nodes of degree one. If decoding completes, the values of the variable nodes are the decoded word. If at any stage before completion, there are no check nodes of degree one the algorithm fails (there are still edges left in the graph).

This algorithm can be adapted for use with the cascade structure of a Tornado code. In both of the algorithms, decoding will be successful only if there is a sufficient number of suitable right nodes of degree one at each stage until completion. Thus the problem of designing each stage of the decoding algorithm to ensure this condition with high probability is of importance. Henceforth only one stage of the graph is of interest and the Tanner graph description of the binary linear code is assumed. The connection to the normal graph representation of interest above is immediate.

Example 2.3 Consider a parity-check matrix of a $(8,4,4)_2$ extended Hamming code:

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

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

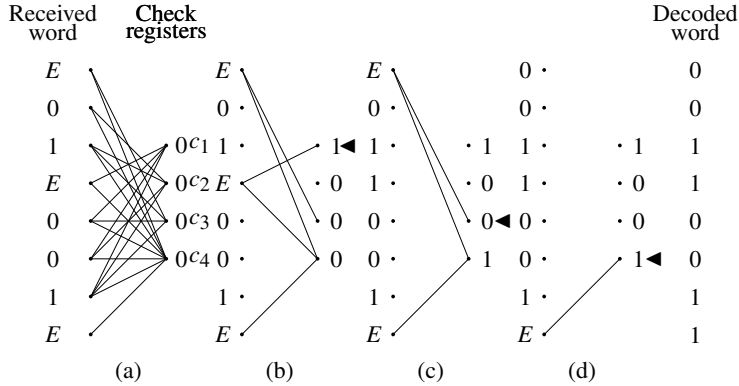


Figure 2.3 Decoding the triple erasure-correcting extended Hamming $(8,4,4)_2$ code

As a code of distance 4 it is capable of correcting three erasures. Suppose the codeword $c = (0,0,1,1,0,0,1,1)$ is transmitted on a BEC and the received word is $y = (E,0,1,E,0,0,1,E)$ is received. The decoding process is shown in Figure 2.3.

Algorithms where messages are passed on graph edges are termed *message-passing algorithms*. When the messages reflect a belief on the bit values involved they are termed *belief propagation* (BP) algorithms. A *Gaussian elimination* (GE) decoding algorithm is where decoding is achieved by solving a matrix equation via matrix reduction, involving received symbols and information symbols (packets). Such decoding algorithms are often optimum achieving a minimum probability of error – hence a *maximum-likelihood* (ML) algorithm – but suffer from a much higher complexity, often $O(n^3)$, than BP. In BP algorithms it is quite possible that the system of equations to be solved is of full rank – hence possesses a unique solution, yet the BP decoding algorithm fails because there is no appropriate check node. Hence BP algorithms are generally not ML.

Decoding algorithms that achieve linear decoding complexity are sought. For successful decoding of the Tornado codes it is necessary for there to be at least one suitable check node at each iteration and the analysis of the algorithm to compute the probability of complete decoding involves computing the probability of there being such a check node at each stage of decoding until completion. The construction of graphs that allow for such an analysis is needed – an overview of the following innovative approach of [11] that achieves this goal is discussed.

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

For a graph edge, define its *left (resp. right) degree* to be i if it is connected to a variable (resp. check) node of degree i. Thus all edges incident with a degree i vertex (either variable or check) are of edge degree i. Define a pair of degree sequences, $(\lambda_1, \ldots, \lambda_m)$ for the left (variable) nodes and right $(\rho_1, \ldots, \rho_m)$ for the right check nodes, with λ_i the fraction of edges with left degree i and ρ_i the fraction of edges with right degree i. It is convenient for the analysis to follow to define two polynomials

$$\lambda(x) = \sum_i \lambda_i x^{i-1} \quad \text{and} \quad \rho(x) = \sum_i \rho_i x^{i-1}.$$

The exponents of $(i-1)$ rather than i in these polynomials are an artifact of the analysis. Graphs with these edge degree fractions will be denoted (λ, ρ) distribution graphs or equivalently $C^n(\lambda, \rho)$. Thus by a $C^n(\lambda, \rho)$ graph is meant some incarnation of a bipartite graph with some n variable nodes with these edge distributions. One can think of choosing a graph uniformly at random from this ensemble of graphs for analysis.

The case of (d_v, d_c) biregular bipartite discussed previously corresponds to $\lambda(x) = x^{d_v-1}$, $\rho(x) = x^{d_c-1}$.

If E is the number of edges in the bipartite graph (previously used to indicate a coordinate position containing an erasure) then the number of left or variable nodes of degree i is given by $E\lambda_i/i$ and the total number of variable nodes is then $E\sum_i \lambda_i/i$, with similar comments for check nodes. The average left degree of nodes in the graph is

$$a_L = 1\Big/\Big(\sum_i \lambda_i/i\Big) = 1\Big/\Big(\int_0^1 \lambda(x)dx\Big)$$

and the average right degree is

$$a_R = 1\Big/\Big(\sum_i \rho_i/i\Big) = 1\Big/\Big(\int_0^1 \rho(x)dx\Big).$$

The code rate is given by

$$R = \frac{k}{n} = 1 - \frac{(n-k)}{n} \geq 1 - \frac{E\sum_i \rho_i/i}{E\sum_i \lambda_i/i} = 1 - \frac{a_L}{a_R} = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}$$

with equality only if the check equations are linearly independent.

Our goal will be to determine conditions on these two-degree distributions that ensure, with high probability, the availability of degree one check nodes at each stage of decoding to allow the decoding algorithm to complete. Before considering this problem, it is noted that given such degree distributions it is

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

not difficult to give a random graph construction with such distributions, as follows.

To construct a Tanner graph corresponding to a code of length $n = k + m$, $m = n - k$ and dimension k the number of left variable nodes is n and the number of check right nodes is m. Decide on an appropriate number of graph edges E which is the total number of 1's in the parity-check matrix and is proportional to the complexity of the proposed decoding algorithm. For a given $(\lambda(x), \rho(x))$ distribution the number of left (resp. right) nodes of degree i is $E\lambda_i/i$ (resp. $E\rho_i/i$). For the sake of argument assume all such quantities are integers. To construct a bipartite graph with the given edge distributions, imagine an array consisting of four columns of nodes, the first column is the set of n variable (codeword nodes) and the fourth column corresponding to the m check nodes. The second and third columns each have E nodes. From each left variable node of degree i, designate, for each i, $E\lambda_i/i$ variable (first column) nodes to be of degree i and generate i edges emanating from each of them terminating in a total of $E\lambda_i$ (disjoint) nodes in the second column. Similarly for each right check node of degree i, designate, for each i, $E\rho_i/i$ nodes to be of degree i and generate i edges from each of them terminating in a total of $E\rho_i$ (disjoint) nodes in the third column. The second and third columns each contain E nodes and so far are of degree one. A random permutation of order E is generated to join nodes in the second and third columns. The nodes in the first and fourth columns are now joined (in the sense of edges between them) and the nodes in the second and third columns can be deleted. There is a small probability that this procedure might generate a bipartite graph with multiple edges between nodes in the first and fourth columns. These may be removed and the effect on the probabilistic analysis is minimal. Similarly there is a probability the associated parity-check matrix will not be of full rank.

It is noted again that for such graphs with edge distributions, the central question is how to choose the distributions $\lambda(x)$ and $\rho(x)$ in order to have the previous algorithm complete the decoding with high probability? Equivalently how to choose the distributions so that, with high probability, there is at least one degree one check node at each stage of the decoding. A formal analysis involving the use of martingales and differential equations is given in [7, 10] and will be commented on later. The following informal argument is instructive and gives an idea as to why the condition on the degree distributions developed arises [9].

Consider an edge e joining variable node v and check node c_{i-1} with left edge degree i and right edge degree j. Consider the graph generated by variable node v and all paths emanating from v with the first edge e for ℓ iterations, with one iteration being one traverse from a variable node to a check

node and back again to another variable node. If the girth of the graph is greater than 4ℓ, this graph will be a tree and all variable nodes encountered when traversing from v to the variable nodes (the leaves of the graph) at level 0 will have been erased on the channel independently, each with probability δ.

Let p_ℓ be the probability that a variable node is not resolved (its value is not known) by level ℓ. A recursion is developed for $p_{\ell+1}$ and a condition found on the two edge distributions that ensures these probabilities are decreasing as the algorithm continues. The implication is that if the conditions stated remain satisfied as the number of iterations increases, the probability will vanish implying successful decoding. Suppose all variable node neighbors of check node c_{i-1} are resolved. Since each such node has the same probability of being resolved at level ℓ, the probability they are resolved is $(1-p_\ell)^{j-1}$. The probability the edge e has right degree j is ρ_j and the probability that at least one such neighbor of c_{i-1} is unresolved is

$$1-\sum_j \rho_j (1-p_\ell)^{(j-1)} = 1-\rho(1-p_\ell)$$

(hence the reason for the $(i-1)$ exponent in the polynomial definition). Now the variable node v at iteration $\ell+1$ will be unresolved only if all $i-1$ check nodes of v at lower level in the tree have at least one lower-level variable node unresolved and since each edge joining v to these lower-level check nodes has left degree i with probability λ_i, the probability the variable node v remains unresolved at level $\ell+1$ is

$$p_{\ell+1} = \delta \sum_i \lambda_i (1-\rho(1-p_\ell))^{(i-1)} = \delta\lambda(1-\rho(1-p_\ell)) \tag{2.1}$$

where $p_0 = \delta$, the erasure probability on the channel, the initial condition that variable node v was unresolved (received as an erasure). Thus [28] successful decoding gives the condition that the probability a variable node being unresolved decreases with the level:

$$p_{\ell+1} = \delta\lambda(1-\rho(1-p_\ell)) < p_\ell. \tag{2.2}$$

Thus as the number of iterations ℓ increases the probabilities will tend to 0 and decoding will be successful if the condition holds. In other words, finding edge distributions $\lambda(x)$ and $\rho(x)$ that satisfy this condition will, with some probability, assure the completion of the decoding algorithm.

Alternatively, replacing p_ℓ by a variable x, if neighborhoods of a variable node of depth ℓ are trees (which assures the independent erasure condition holds), and if

$$\delta\lambda(1-\rho(1-x)) < (1-\epsilon)x, \quad 0 < x < \delta,$$

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

then after ℓ iterations the probability a variable node is unresolved is at most $(1-\epsilon)^{\ell}\delta$. Thus edge distributions $\rho(x)$ and $\lambda(x)$ are sought that, for a given erasure channel δ, satisfy the condition

$$\delta\lambda(1-\rho(1-x)) < x, \quad 0 < x < \delta \tag{2.3}$$

so that with high probability, decoding on a BEC channel with erasure probability at most δ will be successful, i.e., the probabilities the nodes are unresolved decrease with iterations.

Notice that as a polynomial with positive coefficients $\lambda(x)$ is an increasing function on $(0,1)$ and hence is invertible and letting $x = \delta\lambda(1-y)$ the condition can be written as

$$\rho(1-\delta\lambda(1-y)) \geq 1-\lambda^{-1}(\lambda(1-y)) = y \quad \text{on} \quad (0,1) \tag{2.4}$$

and the equivalent (dual) condition to Equation 2.3 is [29]

$$\rho(1-\delta\lambda(1-y)) > y, \quad 0 < y < 1.$$

Equivalently by letting $y = \rho^{-1}(1-x)$ this equation can be written as

$$\delta\lambda(1-\rho(y)) < 1-y, \quad y \in [0,1) \tag{2.5}$$

(which is also obtained by substituting $y = 1-x$ in Equation 2.3). Degree distributions that satisfy these conditions for as large a value of δ as possible are sought, where δ is the probability of erasure on the channel. Thus with high probability, for $(\lambda(x), \rho(x))$ distributions that satisfy these conditions, the decoding will complete for as high a channel erasure probability as possible. Indeed, distributions that result in a δ satisfying a code rate $R = 1-\delta$ achieve capacity on a BEC.

The informal development for the condition in Equation 2.3 can be made formal. The approach in [7, 8, 10, 23, 24] is outlined. Let $\ell_t^{(i)}$ and $r_t^{(i)}$ be the *fraction* of left (resp. right) edges at stage (time) t of the algorithm of degree i, where by fraction is meant the actual number of left (resp. right) divided by the original total number of edges E. A discrete time differential equation is developed in terms of the degree distributions and solved. In particular it is shown ([10], proposition 1) that the fraction of degree one right edges is

$$r_1(x) = \delta\lambda(x)[x-1+\rho(1-\delta\lambda(x))]$$

where x is defined via $dx/d\tau = -x/e(\tau)$ where $e(\tau)$ is the fraction of edges remaining at time τ. Thus the decoding continues as long as $r_1(\tau) > 0$ which leads to the proposition:

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

Proposition 2.4 ([10], proposition 2) *Let B be a bipartite graph that is chosen at random with edge degree distributions $\lambda(x)$ and $\rho(x)$. Let δ be fixed so that*

$$\rho(1 - \delta\lambda(x)) > 1 - x, \quad x \in (0, 1]. \tag{2.6}$$

For all $\eta > 0$ there is some k_0 such that for all $k \geq k_0$, if the k (left) message bits of $C(B)$ are erased independently with probability δ, then with probability at least $1 - k^{2/3}\exp(-\sqrt[3]{k/2})$ the recovery algorithm terminates with at most ηk message bits erased.

Notice the proposition only gives an upper bound on the number of erasures remaining after the algorithm terminates.

The result can be used to determine suitable distributions to ensure there is at least one check node available of degree one for the completion of the algorithm. Returning to Tornado codes, the following conclusion can be shown as a condition on the degree distributions:

Theorem 2.5 ([10], theorem 2) *Let k be an integer and suppose that*

$$C = C(B_0, \ldots, B_m, C_0)$$

is a cascade of bipartite graphs where B_0 has k variable nodes. Suppose each B_i is chosen at random with edge degrees specified by $\lambda(x)$ and $\rho(x)$ such that $\lambda_1 = \lambda_2 = 0$ and suppose that δ is such that

$$\rho(1 - \delta\lambda(x)) > 1 - x, \quad 0 < x < 1.$$

Then, if at most a δ-fraction of the coordinates of an encoded word in C are erased independently at random, the erasure-decoding algorithm terminates successfully with probability $1 - O(k^{-3/4})$ and does so in $O(k)$ steps.

The $O(k)$ complexity follows from the fact the average node degree is a constant. Recall that if the probability of an erasure on the BEC is δ, the capacity of the channel, the maximum rate at which information can be transmitted through the channel reliably, is $R = 1 - \delta$, or conversely, the maximum-erasure rate that a code of rate R may be used reliably on a BEC is $\delta = 1 - R$. The following result shows this:

Theorem 2.6 ([10], theorem 3) *For any rate R with $0 < R < 1$, any ϵ with $0 < \epsilon < 1$ and sufficiently large block length n, there is a linear code and a decoding algorithm that, with probability $1 - O(n^{-3/4})$, is able to correct a random $(1 - R)(1 - \epsilon)$-fraction of erasures in time proportional to $n \ln(1/\epsilon)$.*

Note that this result achieves the goal of a linear-time (in code length) decoding algorithm. Thus the challenge is to devise a distribution pair

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

$(\lambda(x), \rho(x))$ such that the rate R of the corresponding code (the related bipartite graph) is such that the code is able to correct a fraction of $1 - R$ erasures, asymptotically on average. Distribution pairs that achieve this relationship are referred to as *capacity-achieving sequences*.

The average node degrees on the left and right, a_L and a_R, were shown to be

$$a_L = \left(\sum_i \lambda_i / i \right)^{-1} \quad \text{and} \quad a_R = \left(\sum_i \rho_i / i \right)^{-1}.$$

The following theorem is of interest:

Theorem 2.7 ([27], theorem 1) *Let G be a bipartite graph with distributions $(\lambda(x), \rho(x))$ and let δ be a positive number such that*

$$\delta \lambda(1 - \rho(1 - x)) \leq x, \quad 0 < x \leq \delta.$$

Then

$$\delta \leq \frac{a_L}{a_R}(1 - (1 - \delta)^{a_R}).$$

It is also of interest to note ([27], lemma 2) that if λ and ρ are polynomials satisfying the above with the above notation, then $\delta \leq \rho'(1)/\lambda'(1)$. Thus the distribution pair determines the rate of the code and this result determines a bound on how close it will be to achieving capacity in the sense it gives a bound on the erasure-correcting capability of the code. In addition it is of interest to find good degree distributions so as to yield as large a value of erasure probability δ as possible.

We return to the notion of a Tornado code which is a cascade of graphs each of whose distributions satisfy the above conditions. They are referred to as Tornado [2] as in practice it often occurs that as the substitution process progresses (recovery of variable nodes by check nodes of degree one), the decoding process typically proceeds slowly until the resolution of one more variable node results in the whole set of variable nodes being resolved quickly.

To summarize, the arguments that led to the conditions Equations 2.3 and 2.4 say that under certain conditions on the edge distributions, the probability of a node not being resolved by a certain iteration decreases with the number of iterations and hence tends to zero and to complete decoding. The results depend only on edge distributions and hence can apply also to the Tanner graph representation of a code. The decoding algorithm is equivalent to that of Tornado codes and, as noted, the crucial property is to have check nodes of degree one with high probability at each iteration, which the stated edge degree distributions tend to fulfill.

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

The fact that the differential equation/martingale approach to the decoding problem and this rather different approach leads to the same conditions is an interesting confirmation of the results.

Examples of Capacity-Achieving Distributions

Numerous works give examples of pairs of distributions $(\lambda(x), \rho(x))$ that satisfy the above conditions. Recall that the average left and right degrees of a graph with edge distributions $(\lambda(x), \rho(x))$ are $1/\left(\sum_i \lambda_i/i\right)$ and $1/\left(\sum_i \rho_i/i\right)$ $\left(\text{or } 1/\int_0^1 \lambda(x)dx \text{ and } 1/\int_0^1 \rho(x)dx\right)$. The rate R of the code is at least $1-\left(\int_0^1 \rho(x)dx/\int_0^1 \lambda(x)dx\right)$ (depending on the corresponding matrix having linearly independent rows). It can be shown [28] that for given $(\lambda(x), \rho(x))$ distributions satisfying condition of Equation 2.3, δ is always less than or equal to $1-R$ for R the rate of the resulting code derived from the edge distributions. For the formal definition of a capacity-achieving sequence we use the following:

Definition 2.8 ([28], section 3.4) An edge distribution sequence $(\lambda(x), \rho(x))$ is called *capacity achieving* of rate R if

(i) the corresponding graphs give rise to codes of rate at least R;
(ii) for all $\epsilon > 0$ there exists an η_0 such that for all $\eta > \eta_0$ we have

$$\lambda(1-\rho(1-x)) < x \text{ for } x \in (0, (1-R)(1-\epsilon))$$

where η is the length of the probability distributions λ and ρ.

Example 2.9 The first example of such distributions, cited in numerous works (e.g., [2, 10, 27, 28, 29]) is referred to as *heavy-tailed/Poisson* sequences for reasons that will become clear.

Let D be a positive integer (that will be an indicator of how close δ can be made to $1-R$ for the sequences obtained). Let $H(D) = \sum_{i=1}^{D} 1/i$ denote the harmonic sum truncated at D and note that for large D, $H(D) \approx \ln(D)$. The two distributions parameterized by the positive integer D are

$$\lambda_D(x) = \frac{1}{H(D)} \sum_{i=1}^{D} x^i/i \quad \text{and} \quad \rho_D(x) = e^{\mu(x-1)} \ (D \text{ terms}).$$

Here μ is the solution to the equation

$$\frac{1}{\mu}(1-e^{-\mu}) = \frac{1-R}{H(D)}\left(1-\frac{1}{D+1}\right)$$

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

and such edge distributions give a code of rate at least R and note the average left degree is $a_L = H(D)(D+1)/D$ and $\int_0^1 \lambda_D(x)dx = (1/H(D))(1 - 1/(D+1))$. The right degree edge distribution is the truncated Poisson distribution. It can also be established that

$$\begin{aligned}\delta\lambda_D(1-\rho_D(1-x)) &= \delta\lambda_D(1-e^{-\mu x}) \\ &\le \frac{-\delta}{H(D)}\ln(e^{-\mu x}) \\ &= \frac{\delta\mu x}{H(D)}.\end{aligned}$$

For the right-hand side of the last equation to be at most x it is required that

$$\delta \le H(D)/\mu$$

(δ is the largest erasure probability possible on the channel possible for the conditions) which can be shown to be equal to

$$(1-R)(1-1/(D+1))/(1-e^{-\mu}).$$

That this pair of distributions satisfies condition

$$(1-R)(1-1/D)\lambda_D(1-\rho_D(1-x)) < x, \quad 0 < x < (1-R)(1-1/D),$$

is shown in [28]. (Note: Such codes are referred to as Tornado codes there in contrast to the definition of such codes used here.)

Example 2.10 Another example from [27, 28] is the following: Recall the general binomial theorem

$$(x+y)^\alpha = \sum_{j=0}^{\infty}\binom{\alpha}{j}x^j y^{\alpha-j}$$

for α any real or complex number and the fractional binomial coefficients are given by

$$\binom{\alpha}{n} = \alpha(\alpha-1)\cdots(\alpha-(n-1))/n!\ .$$

Consider the right regular graphs with the check nodes all of degree a, for a positive integer $a \ge 2$ and the distributions

$$\lambda_{a,n}(x) = \frac{\sum_{k=1}^{n-1}\binom{\alpha}{k}(-1)^{k+1}x^k}{1-n\binom{\alpha}{k}(-1)^{k+1}} \quad \text{and} \quad \rho_a(x) = x^{a-1}$$

where $\alpha = 1/(a-1)$ is real, positive and noninteger. Notice that the fractional binomial coefficients alternate in sign for $\alpha < 1$ and hence the coefficients of the distribution $\lambda_{a,n}$ are positive. It is convenient to introduce a parameter

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

ν such that $0 < \nu < 1$ by $n = \nu^{-1/\alpha}$ (ignoring integer constraints). It is shown in ([27], proposition 2 and theorem 2), that for the code rate defined as $R = 1 - a_L/a_R$ and $a_R = a = (\alpha + 1)/\alpha$

$$\frac{\delta}{1 - R} \geq 1 - \nu^{\alpha+1}/\alpha = 1 - \nu^{a_R}.$$

As $\nu < 1$ this suggests $\delta \approx 1 - R$ for large a_R, approaching capacity.

The properties of these distributions and the sense in which they are asymptotically optimal and satisfy the conditions such as Equation 2.3 or 2.4 is discussed in [27]. Notice in this case the graphs are right regular – all right check nodes of the same degree.

Example 2.11 A variety of techniques have been developed to determine capacity-achieving sequences, including a linear programming approach and density evolution, a method used effectively in the analysis of LDPC codes (e.g., see [10, 34] and Chapter 3).

As an example, the following distribution pair given in [34] was found using density evolution:

$$\begin{aligned} \lambda(x) &= 0.29730x + 0.17495x^2 + 0.24419x^5 + 0.28353x^{19} \\ \rho(x) &= 0.33181x^6 + 0.66818x^7 \end{aligned}$$

The Example 2.10 is interesting in that one is able to construct good sequence pairs with one of the pair being a monomial, i.e., all right nodes have the same degree (right regular). However, it is shown in [7] that sequence pairs which are both monomials (hence biregular graphs) perform poorly.

The relationship of these results on the BEC to those obtained for other channels, most notably the BIAWGN using belief propagation as discussed in Chapter 3, is of interest. To briefly note the approach taken there, the pdf p_ℓ of the log likelihood ratios (LLRs) (not the same p_ℓ used in the previous analysis) passed during the decoding process with the code graph described by the distribution pair $(\lambda(x), \rho(x))$ satisfied the recursion (Equation 3.28 of Chapter 3) is shown there to satisfy

$$p_\ell = p_0 \otimes \lambda(\Gamma^{-1}(\rho\Gamma(p_{\ell-1}))), \quad \ell \geq 1$$

where $\otimes$ indicates convolution and Γ is an operator introduced in Chapter 3 that gives the pdf of it argument. It should be noted the arguments in that chapter will be likelihood ratios as messages passed on the decoding graph and these are random since they depend on received random messages.

Applying this equation to the erasure channel, the pdf [28] is a two-point mass function with a mass of p_ℓ at 0 and mass $1 - p_\ell$ at ∞. Performing the

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

convolutions with such mass functions in the above equation can be shown to yield the result

$$p_\ell = \delta\lambda(1 - \rho(1 - p_{\ell-1}))$$

where, as before, δ is the channel erasure probability. This is essentially the Equation 2.2 obtained by very different means, an interesting confirmation.

2.3 LT and Raptor Codes

The formulation of the Tornado codes of the previous section introduced (at least) two novel ideas: the notion of deriving edge probability distributions to generate bipartite code graphs where certain simple decoding algorithms could prove effective. In a sense to be discussed these ideas led to the notion of fountain codes.

To initiate the discussion, consider the following simple situation. A server has k information packets of data of some fixed number of bits to be downloaded over a network to a large number of users. The users are interested in receiving a complete set of the k packets. Packets can be XORed. One possibility to achieve this download is to simply forward the packets on the Internet with each user obtaining the packets in some order. However, due to imperfections in the Internet such as buffer overflow or node servers failing, it is likely some users will miss one or several of the packets leading to the requirement of some feedback mechanism to the source where each user is able to request retransmission of their missing packets. This can lead to inefficient operation and congestion (implosion) on the network.

A more interesting possibility is for the server to first *code*, each coded packet consisting of the XOR of a random selection of information packets. This notion will be the basis of fountain codes.

Consider first a (not very efficient) case of random fountain coding where each of the k information packets is included in the formation of a coded packet with a probability of $1/2$ independently and transmitted on the Internet, i.e., the coder chooses uniformly at random a selection of information packets, XORs them together to form a coded packet which is transmitted on the network.

A user now must gather a sufficient number of *any* of the coded packets to allow the solution of the set for the original information packets. In effect the user must gather a set of $(k + m)$ coded packets for m a small positive integer to allow for a full-rank random $k \times (k+m)$ matrix equation to be formed which is necessary to guarantee solution. The fact that any of the coded packets can be used is a positive feature of the scheme. However, solving for the original

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

k information packets involves the solution of a $k \times (k+m)$ binary random matrix, say by Gaussian elimination, an operation that is $O(k^3)$ in complexity which, since k might easily on the order of several thousand be far higher than is typically of interest. By Equation A.8 of Appendix A the probability such a matrix is of full rank, $Q_{k,k+m}$ (hence solvable by Gaussian elimination), is

$$0.999 < Q_{k,k+10} < 0.999 + \epsilon, \quad \epsilon < 10^{-6}.$$

Such codes are generally referred to as *fountain codes* for the obvious reason. They are also referred to as *rateless codes* as their design involves no particular rate in the sense of the design of block codes.

A possible remedy to the large complexity of the Gaussian elimination decoding argument might be as with the Tornado codes and form a coding graph in the obvious manner. The headers of the coded packets contain the information as to which information packets were XORed to form the coded packet allowing the code graph to be formed consisting of information nodes on the left corresponding to information packets (nodes) and code packets (nodes) on the right. There is an edge between an information packet and a code packet if the information packet is involved with the formation of the code packet. The following decoding algorithm is as for Tornado codes.

Algorithm 2.12 If there is a coded node of degree 1, XOR the contents of the coded node to the neighbor information node thus resolving that information node. XOR this information node to its other neighbor coded nodes and delete all edges involved (decreasing the number of unresolved information nodes by one and the number of coded nodes not yet used by at least one).

This algorithm is simple with linear decoding complexity. Notice that if the $k \times (k+m)$ matrix formed, corresponding to the graph, is of full rank (k), Gaussian elimination is guaranteed to provide a solution for the k information packets. On the other hand the decoding algorithm above, while very simple with linear complexity, is unlikely to run to completion and will almost certainly fail.

The remedy for this situation is not to choose the packets for XORing to form a coded packet uniformly at random but rather formulate a probability distribution for the number of information packets to be XORed, much as for a single section of the Tornado code algorithm, so that with high probability, the previous simple decoding algorithm will be successful, i.e., at each stage of decoding find a check node of degree one all the way to completion. With such a distribution the simple (linear complexity) decoding algorithm can be used rather than the computationally expensive Gaussian elimination. This is the genius behind the LT (Luby transform) [6] fountain codes discussed below.

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

LT Codes

The approach of Tornado codes and capacity-achieving sequences of the previous section suggests the following possibility [6, 31]. As above, assume there are k information packets to be coded (to give coded packets). A discrete probability distribution (the literature typically uses $\Omega(x)$ for this distribution, a convention which is followed here)

$$\Omega_i, \;\; i = 1, 2, \ldots, k, \quad \sum_{i=1}^{k} \Omega_i = 1, \quad \Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$$

is to be chosen and used in the following manner to form coded packets: for each coded packet, the distribution is first sampled – if d is obtained (with probability Ω_d, $d \in [k]$) – then d distinct information packets are chosen uniformly at random from among the k and XORed together to form a coded packet to be transmitted on the network. The choice of packets is stored in the coded packet header. The process is repeated as required. As noted, such a coding process is referred to as a fountain code or as a rateless code since there is no specific rate associated with the process of forming coded words. It is the form of the distribution that restores the complexity of decoding to linear and makes the above decoding algorithm effective.

The receiver gathers slightly more than k, say $k + \epsilon(k)$ (to be discussed), coded packets from the network and forms the bipartite graph with k information packet nodes (considered as initially empty registers), on the left, denoted $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ (to be determined by the decoding algorithm) and $k + \epsilon(k)$ received coded packets on the right, denoted $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{k+\epsilon(k)}$.

The problem is to choose the distribution $\Omega(x)$ to have the probability the decoding algorithm is successful – runs to completion. Clearly at least one check node of degree one is required at each stage of decoding until all information symbols are found. If this is not the case the decoding algorithm fails. Clearly a balance is needed in the sense a sufficient supply of degree one coded nodes is required at each stage to make the decoding algorithm robust and successful while too many degree one nodes will make it inefficient and give poor performance. The analysis is somewhat intricate and the reader is referred to [6, 31, 33] for details. The essence of the results is described here without proofs.

To discuss the situation further, the behavior of the algorithm as it iterates – at each stage finding a coded node of degree one, transferring the coded packet associated with such a node to the connecting information node and then XORing that packet to all neighbor check nodes and removing the information node and all its edges. The terms "node," "symbol" and "packet" are used interchangeably.

Some definitions are needed. The number of coded symbols gathered for decoding, beyond the number of information symbols, is termed the *overhead* of the decoding algorithm. Thus if $k(1+\epsilon k)$ coded packets are used the overhead is $k\epsilon$. A simple argument ([33], proposition 2.1) shows that if an ML decoder is to succeed with high probability for an overhead fraction ϵ that tends to zero with k, then Ω_1 has to converge to zero. At stage i of the algorithm define the *output ripple* (or simply ripple) as the set of coded nodes of degree one. At this stage, since one information symbol (packet) is released at each stage of the algorithm, there are $k-i$ undecoded (or unresolved) information nodes remaining at stage i. A coded node is said to be *released* at stage $i+1$ if its degree is greater than one at stage i and equal to one at step $i+1$. To calculate the probability [30, 31, 33] a coded node of initial degree d is released at stage $i+1$ it is assumed the original neighbor nodes of a coded node are chosen with replacement – convenient for the analysis and having little effect on the code performance. The probability a coded node, originally of degree d, has only one neighbor at stage $i+1$ among the $k-(i+1)$ information nodes not yet recovered and that all of its other $(d-1)$ information node neighbors are in the set of recovered information nodes is

$$d\left(\frac{k-(i+1)}{k}\right)\left(\frac{i+1}{k}\right)^{d-1}$$

(this is where the choosing with replacement assumption appears).

Another way of viewing this formula is to reverse the situation and suppose the set of variable (information) nodes resolved at the i-th iteration is V_i, $\mid V_i \mid = i$ and at the $(i+1)$-st iteration is $V_{i+1} \supset V_i$, $\mid V_{i+1} \mid = i+1$. We ask what is the probability the edges of a constraint (coded) node c of original degree d are chosen so it is released at the $(i+1)$-st iteration with these sets of resolved variable nodes. Thus at iteration i there are at least two variable (information) neighbors of c unresolved and at iteration $i+1$ there is only one unresolved – c is in the output ripple at that stage. There are d ways of choosing the edge from c that will be unresolved at the $(i+1)$-st iteration and the probability it is unresolved then is $\left(1-\frac{i+1}{k}\right)$. For the constraint node c to be released at the i-th iteration it has $(d-1)$ edges attached to resolved nodes at the $(i+1)$-st iteration but not all of these were resolved at the i-th iteration. The probability of this is

$$\left(\left(\frac{i+1}{k}\right)^{d-1}-\left(\frac{i}{k}\right)^{d-1}\right).$$

Thus, as in the above formula, the probability a coded node, originally of degree d, is released at stage $i+1$ of the decoding is approximately

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

$$d\left(1-\left(\frac{i+1}{k}\right)\right)\left(\left(\frac{i+1}{k}\right)^{d-1}-\left(\frac{i}{k}\right)^{d-1}\right).$$

Multiplying this expression by Ω_d and summing yields

$$\begin{aligned}&\Pr(\text{output symbol is released at stage } i+1)\\&=\left(1-\frac{i+1}{k}\right)\left(\Omega'\left(\frac{i+1}{k}\right)-\Omega'\left(\frac{i}{k}\right)\right).\end{aligned}$$

Recalling the approximation for a smooth continuous function (like polynomials) $f''(x)\approx\big(f'(x+\Delta)-f'(x)\big)/\Delta$ for small Δ it is clear that

$$\Omega'\left(\frac{i+1}{k}\right)-\Omega'\left(\frac{i}{k}\right)\approx\frac{1}{k}\Omega''\left(\frac{1}{k}\right)$$

and the expected number of coded nodes released at step $i+1$ is n times this amount ($n=k+\epsilon(k)$, the number of coded packets collected)

$$\frac{n}{k}\left(1-\frac{i+1}{k}\right)\Omega''\left(\frac{i}{k}\right). \tag{2.7}$$

If one sets $x=i/k$, if the probability the last coded symbol is released at stage $k+\epsilon(k)=n$ is one, this is equivalent to the equation

$$(1-x)\Omega''(x)=1,\quad 0<x<1$$

which, with the required condition that $\Omega(1)=1$, has the solution

$$\Omega(x)=\sum_{i\geq 2}\frac{x^i}{i(i+1)}.$$

This is referred to as the *limited degree distribution* [33]. Clearly this distribution is useless for our purpose since it produces no coded nodes of degree one and decoding could not start. In addition, the distribution is infinite – the needed distribution should produce no coded nodes of degree greater than k, the number of information nodes assumed. The analysis, however, is instructive. The following modified distribution is suggested [6]:

Definition 2.13 The *(ideal) soliton distribution* is defined as

$$\Omega_i^{sol}=\begin{cases}\frac{1}{k}, & i=1\\ \frac{1}{i(i-1)}, & i=2,3,\ldots,k.\end{cases} \tag{2.8}$$

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

The term soliton arises in a refraction problem in physics with similar requirements. The distribution is associated with the polynomial

$$\Omega^{sol}(x) = \frac{x}{k} + \sum_{i=2}^{k} \frac{x^i}{(i-1)i}.$$

That this is a distribution (probabilities sum to unity) follows readily by an induction argument or by observing that $1/(i(i-1)) = 1/(i-1) - 1/i$.

Notice that this distribution has a coded node of expected degree

$$\sum_{i=1}^{k} i\Omega_i^{sol} = \frac{1}{k} + \sum_{i=2}^{k} i\frac{1}{i(i-1)} = \sum_{i=1}^{k} \frac{1}{i} \approx \ln(k)$$

which is the first k terms of the harmonic series which is well approximated by $\ln(k)$ for large k. This is the minimum possible to give a reasonable possibility of each information node being involved with at least one coded node in the following sense. The problem is equivalent to throwing n balls (the coded nodes) into k cells (the information nodes) and asking for the probability there is no empty cell (although with replacement). This is just one less the probability at most $(k-1)$ cells are occupied which is

$$1 - \binom{k}{k-1}\left(\frac{k-1}{k}\right)^n.$$

If it is wished to have a probability of $1-\alpha$ to have no cell empty (all information nodes covered) is

$$1 - \alpha = 1 - k\left(1 - \frac{1}{k}\right)^n \quad \text{or} \quad \left(1 - \frac{1}{k}\right)^n = \alpha$$

and in the limit

$$\lim_{k \longrightarrow \infty} \left(1 - \frac{1}{k}\right)^{k\cdot(n/k)} \longrightarrow \exp(-n/k) \approx \alpha/k \quad \text{or} \quad n \approx k\log(k/\alpha).$$

Thus for the encoding process to cover each information symbol at least once with probability at least $1-\alpha$, the average degree of the approximately k coded nodes must be at least $\log(k/\alpha)$.

As discussed below, the ideal soliton distribution will prove to be unsatisfactory for several reasons. One problem is that the variance of the number of coded nodes in the ripple at each stage of the decoding is so large that the probability there is no node of degree one at each stage is too high. Nonetheless it does have some interesting properties. If the probability that a coded node of initial degree i is released when there are L information nodes remaining unrecovered is denoted $r(i, L)$ and $r(L)$ is the overall probability of release [6],

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

$$r(L) = \sum_i r(i, L).$$

Then for the ideal soliton distribution it is shown ([6], proposition 10) that $r(L) = 1/k$ and the probability a coded node is released at each stage of decoding is $r(L) = 1/k$, i.e., the probability of release is the same at each stage of decoding.

For the following let η (δ in [6], used for erasure probability here) be the target probability the decoding algorithm fails to complete decoding. In an effort to improve the performance of the ideal soliton distribution the *robust soliton distribution* was proposed, defined as follows ([6], definition 11):

Definition 2.14 The *robust soliton distribution* denoted Ω^{rs} is defined using the ideal Soliton distribution and the function τ_i as follows. Let $R = c\sqrt{k}\ln(k/\eta)$ for a constant $c > 0$ and define

$$\tau_i = \begin{cases} R/ik & \text{for } i = 1, 2, \ldots, k/R - 1 \\ R\ln(R/\eta)/k & \text{for } i = k/R \\ 0 & \text{for } i = k/R + 1, \ldots, k. \end{cases}$$

Then the robust soliton distribution is

$$\Omega_i^{rs} = (\Omega_i^{sol} + \tau_i)/\beta, \quad i = 1, 2, \ldots, k \tag{2.9}$$

for the normalizing constant $\beta = \sum_{i=1}^{k}(\Omega_i^{sol} + \tau_i)$.

The rationale for this distribution is that it tends to maintain the size of the ripple, the number of coded nodes of degree one at each stage of decoding, preventing the size to fall to zero at any stage before the end – which would lead to decoding failure.

The distribution chosen for the formation of the coded packets is critical for performance of the decoding algorithm. The relationship between the distribution and the number of extra coded packets (beyond the number of information packets k) required to achieve a given error probability (the probability the decoding fails to complete) is complex. For the Robust Soliton distribution it is shown ([6], theorems 12 and 17) that for a decoder failure probability of η an overhead of $K = k + O(\sqrt{k} \cdot \ln^2(k/\eta))$ is required and that the average degree of a coded node is $O(\ln(k/\eta))$ and that the release probability when L information nodes have not been covered is of the form $r(L) \geq L/((L - \theta R)K), L \geq R$ for some constant θ and R as in the above definition.

It has been observed above the average degree of a coded node under the Robust Soliton distribution is $O(\log(k))$ (for constant probability of decoding error η) and hence each coded symbol requires $O(\log(k))$ operations to

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

produce it and the decoding operation is of complexity $O(k \log(k))$. For very large k a decoder with linear complexity would be more desirable and this will be possible with the Raptor codes discussed in the next section.

The literature on the analysis of aspects of decoding LT codes is extensive. The references [4, 13] are insightful. A detailed analysis of the decoding error probability is given in ([31], section 3).

Shokrollahi [30] defines a *reliable decoding algorithm* as one that can recover the k information symbols from any set of n coded symbols and errs with a probability that is at most inversely polynomial in k, i.e., a probability of the form $1/k^c$. Recall the overhead of a decoding algorithm is the number of coded symbols beyond k needed to achieve the target probability of error. A *random LT code* is one with a uniform distribution on the choice of information symbols to combine for a coded symbol. This corresponds to the choice of the distribution

$$\Omega(x) = (1 + x)^k \cdot (1/2^k)$$

and corresponds to the random analysis given earlier in this section. It can be shown ([30], propositions 1 and 2) that for *any* LT code with k information symbols there is a constant c such that the graph associated with the decoder has at least $ck \log(k)$ edges and that any random LT code with k information symbols has an encoding complexity of $k/2$ and ML decoding is a reliable decoder with an overhead of $O(\log(k)/k)$. ML is Gaussian elimination for the $k \times (k + O(\log(k)/k))$ matrix formed at the decoder with complexity $O(k^3)$. As noted previously in general BP (the algorithm described above) is not ML – indeed it is possible that if K coded symbols are gathered that the $k \times K$ matrix can be of full rank k (for which ML decoding would be successful) and yet the BP algorithm fail to completely decode (run out of coded nodes of degree one before completing). However, the low computational requirements of the BP decoding algorithm and its error performance are impressive.

The implication of these comments is that it will not be possible to have a reliable LT decoder and achieve linear time encoding and decoding. The development of the Raptor codes of the next section shows how this goal can be achieved by a slight modification of LT codes.

Raptor Codes

The idea behind Raptor codes (the term is derived from RAPid TORnado) is simple. It was observed that decoding failures for LT codes tend to occur toward the end of the decoding process when there are few information symbols left unresolved. This suggests introducing a simple linear block

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

erasure-correcting code to correct the few remaining erasures after the LT decoding fails.

For Raptor codes the k information symbols/packets are first *precoded* with an efficient simple (easy to decode) linear block erasure-correcting $C_n = (n,k,d)_2$ code and then these n packets are LT coded. Note this introduces $(n-k)$ parity packets – i.e., the parities of the code C_n are formed across the k information packets. For decoding, some $n(1+\epsilon)$ of the LT coded packets are gathered for LT decoding, as before. Using the LT decoding on the precoded bits, the decoding might (typically) stall with a few uncoded packets left undecoded (no more coded nodes of degree one in the ripple). These can then be decoded with the linear erasure-correcting code C_n. It is also desired to have linear coding and decoding complexity and because of the reduced requirements on the LT decoding, this will be possible. Recall from the above discussion that a reliable decoding algorithm for LT codes must have at least $O(\log k)$ information node degrees and an overhead of $O(\log(k)/k)$.

Much of the development and commercial deployment of Raptor codes is due largely to the work of Michael Luby and Amin Shokrollahi and their colleagues although the key idea behind them was independently found in the work of Maymounkov [15, 16]. Both of the papers [30] and [15] are important reading.

A Raptor code then has parameters $(k, C_n, \Omega(x))$ where C_n is a binary linear $(n,k,d)_2$ erasure-correcting code and $\Omega(x)$ is a distribution on n letters. For simplicity Ω is used in the remainder of this chapter or Ω_D when the parameter D is needed. The coordinates of the code C_n will be an $(n,k,d)_2$ code and will produce n *intermediate packets* from the k information packets and $(n-k)$ parity packets. This code is not specified further – as noted its requirements are minimal in that a variety of simple codes, capable of correcting a few erasures with linear complexity, can be used. It is used essentially to correct a few erasures if the LT decoding leaves a few nodes unresolved (viewed as erasures) toward the end of its decoding process. That is, if the LT decoding does not quite complete to the end, leaving a few unresolved packets, the code C_n completes the decoding, treating the unresolved packets as erasures. The output of the LT code will be referred to as the coded packets and $(n, \Omega_D(x))$ LT coding, as before.

It will be important to recognize that while LT codes cannot have linear decoding complexity, using the LT decoding process with an appropriate distribution to decode up to a few remaining erasures can have linear complexity.

As before, a reliable decoding algorithm for the Raptor code will have a probability of error of inverse polynomial form, $1/k^c$ for some positive constant c. Such codes will be analyzed with respect to their space requirements,

overhead and cost or complexity. The two extremes of such codes are the LT codes where C is the trivial $(k,k,1)_2$ code and the *precode only* (PCO) code where there is no LT code. The *decoding cost* of such a code is the expected number of arithmetic operations per information symbol. The *overhead* of the code is the number of coded symbols *beyond* k needed to recover all information symbols. The Raptor code $(k,C_n,\Omega_D(x))$ is thought of in terms of three columns of nodes or symbols (the left part of the previous figure):

- a left column of k information packets (associated with information packets);
- a middle column of *intermediate* packets consisting of the k information packets and $(n-k)$ check packets computed using the linear erasure-correcting code $C_n = (n,k,d)_2$;
- a right column of $N = n(1+\epsilon)$ coded packets consisting of LT coding of the intermediate packets using the distribution $\Omega_D(x)$.

A representation of the Raptor encoder/decoder is shown in Figure 2.4.

In analyzing Raptor codes it will be assumed ([30], proposition 3) the precode C_n can be encoded with complexity βk for some constant $\beta > 0$ and that there is an $\epsilon > 0$ such that C_n can be decoded over a BEC with erasure probability $1 - R(1+\epsilon)$ with high probability with cost/complexity γk. With such assumptions Raptor codes will be designed with constant encoding and decoding cost per symbol (hence both of overall complexity $O(k)$), and their space consumption is close to one and their overhead close to zero.

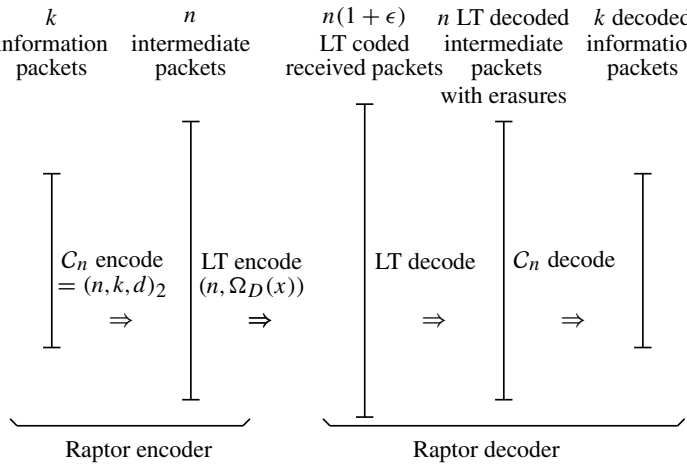


Figure 2.4 Basic structure of a Raptor code $(k,C_n,\Omega_D(x))$ system – encoder/decoder – graph edges not shown for clarity

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

We discuss the section VI of [30] on Raptor codes with good asymptotic performance. The arguments needed to achieve linear encoding and decoding require a delicate balancing of parameters.

The distribution used for the LT coding ([30], section VI) is a slightly modified ideal soliton:

$$\Omega_D(x) = \frac{1}{\mu+1}\left(\mu x + \sum_{i=2}^{D} \frac{x^i}{i(i-1)} + \frac{x^{D+1}}{D}\right) = \sum_{i=1}^{D} \Omega_i x^i \qquad (2.10)$$

where $\mu = (\epsilon/2) + (\epsilon/2)^2$ and $D = \lceil 4(1+\epsilon)/\epsilon \rceil$. That this is in fact a distribution is seen by observing

$$\sum_{i=2}^{D} \frac{1}{i(i-1)} = \sum_{i=2}^{D} \left(\frac{1}{i-1} - \frac{1}{i}\right) = 1 - \frac{1}{D}$$

and hence it is a distribution for any positive number μ and positive integer $D > 2$. It will be shown later that for constant D this distribution will yield coded nodes with constant average degree (as opposed to being a function of k) which allows the overall algorithm to have a linear complexity. With this distribution we have:

Lemma 2.15 ([30], lemma 4) *There exists a positive real number c (depending on ϵ) such that with error probability at most e^{-cn} any set of $(1+\epsilon/2)n+1$ LT coded symbols with parameters $(n, \Omega_D(x))$ are sufficient to recover at least $(1-\delta)n$ intermediate symbols via BP decoding where $\delta = (\epsilon/4)(1+\epsilon)$.*

The proof of this important lemma is straightforward although it requires some interesting manipulations. Some discussions are given to assist – the lemma itself is not proved.

The right-degree node (packet) distribution is by assumption $\Omega_D(x)$ and hence the right edge degree distribution is

$$\omega(x) = \frac{\Omega_D'(x)}{\Omega_D'(1)}.$$

Recall there are n input (intermediate – or left nodes for this argument) nodes to the LT coding process and $N = (1+\epsilon/2)n + 1$ coded (right) nodes for the decoding. To compute the left edge degree distribution $\iota(x)$, let α be the average degree of an output node (LT coded) node which is $\alpha = \Omega_D'(1)$. Let u be a left node and w an output (right) node. The probability that u is a neighbor of w, since on average w has α input neighbors, is α/n. Assuming each output node has this same probability of having u as a neighbor independently, on average the degree of u will have a binomial distribution, i.e., the probability u has degree ℓ is given by

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

$$\binom{N}{\ell}\left(\frac{\alpha}{n}\right)^{\ell}\left(1-\frac{\alpha}{n}\right)^{N-\ell}, \quad N = n(1+\epsilon/2)+1$$

and the generating function of the degree distribution of the left input nodes is thus

$$L(x) = \sum_{\ell=0}^{N}\binom{N}{\ell}\left(\frac{\alpha}{n}\right)^{\ell}\left(1-\frac{\alpha}{n}\right)^{N-\ell}x^{\ell} = \left(1-\frac{\alpha(1-x)}{n}\right)^{N}. \tag{2.11}$$

The edge distribution is $\iota(x)$ where $\iota(x) = L'(x)/L'(1)$ and

$$\iota(x) = \left(1-\frac{\alpha(1-x)}{n}\right)^{(1+\epsilon/2)n}.$$

For these edge distributions Equation 2.5 is equivalent to the condition that

$$\iota(1-\omega(1-x)) < 1-x \quad \text{for} \quad x \in [0, 1-\delta].$$

Since $\lim_{n\to\infty}(1-b/n)^n \longrightarrow \exp(-b)$ and that for $0 < b < m$, $(1-b/m)^m < \exp(-b)$ then

$$\begin{aligned}\iota(1-\omega(1-x)) &= \left(1-\frac{\alpha}{n}\frac{\Omega_D^{'}(1-x)}{\Omega_D^{'}(1)}\right)^{(1+\epsilon/2)n}\\ &< \exp\left(-\frac{\alpha}{n}\frac{\Omega_D^{'}(1-x)}{\Omega_D^{'}(1)}(1+\epsilon/2)n\right)\\ &< \exp(-(1+\epsilon/2)\Omega_D^{'}(1-x)), \quad \text{since } \alpha = \Omega_D^{'}(1).\end{aligned}$$

The above condition then reduces to showing that

$$\exp\Big(-(1+\epsilon/2)\Omega_D^{'}(x)\Big) < 1-x, \quad x \in [0, 1-\delta]. \tag{2.12}$$

To establish this inequality using the form of $\Omega_D^{'}(x)$ of Equation 2.10 is a technical development [10], which completes the proof of Lemma 2.15.

The analysis implies ([31], p. 79) that, asymptotically, the input ripple (the expected fraction of input symbols connected to LT coded symbols of degree one when a fraction of input symbols that have been recovered is x), is given by

$$1-x-\exp\Big(-(1+\epsilon)\Omega_D^{'}(x)\Big). \tag{2.13}$$

The analysis of [9] further gives the fact that if x_0 is the smallest root of Equation 2.13 in $[0,1)$, then the expected fraction of input symbols not recovered at the termination of the decoding process is $1-x_0$. Thus if distributions are designed so that x_0 is maximized then [30] the associated Raptor codes will have an average coded node degree of $O(\log(1/\epsilon))$ a linear

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

decoding complexity of $O(k\log(1/\epsilon))$ and a decoding error probability which decreases inversely polynomial in k.

The conditions on the choice of erasure-correcting code C_n are simple:

(i) The rate of C_n is at least $(1+\epsilon/2)(1+\epsilon)$.
(ii) The BP decoder for C_n used on a BEC can decode to an erasure probability of $\delta = (\epsilon/4)/(1+\epsilon)$ (which is half the capacity) and has linear decoding complexity.

The details are omitted. The key theorem for Raptor codes is the following:

Theorem 2.16 ([30], theorem 5) *Let ϵ be a positive real number, k an integer, $D = \lceil 4(1+\epsilon)/\epsilon \rceil$, $R = (1+\epsilon/2)/(1+\epsilon)$, $n = k/R$ and let C_n be a block erasure code satisfying the conditions above. The Raptor code $(k, C_n, \Omega_D(x))$ has space consumption $1/R$, overhead ϵ and cost $O(\log(1/\epsilon))$ with respect to BP decoding of both the precode C_n and LT code.*

Lemma 2.15 shows the LT code with these parameters is able to recover at least a fraction of $(1-\delta)$ of the intermediate symbols and the erasure code is, by assumption and design, to correct this fraction of erasures to recover the k information symbols. It remains to show the cost is linear in k. The average degree of coded nodes is (from Equation 2.10)

$$\begin{aligned}\Omega_D'(1) &= \frac{1}{\mu+1}\left(\mu + \sum_{i=1}^{D-1}\frac{1}{i} + \frac{D+1}{D}\right)\\ &= 1 + H(D)/(1+\mu) = 1 + \ln\left(\frac{1}{\epsilon}\cdot 4(1+\epsilon)\right) + 1 = \ln\left(\frac{1}{\epsilon}\right) + O(\epsilon)\end{aligned}$$

where the standard upper bound on the truncated Harmonic series $H(D) \leq \ln(D)+1$ has been used. Thus the cost of encoding the LT code is $O(\ln(1/\epsilon))$ per coded symbol which is also the cost of LT decoding and also the cost of decoding the code C_n by assumption. Notice the overhead of the LT code is approximately $\epsilon/2$.

It is noted again that the linear encoding/decoding here is achieved by relaxing the condition that the LT have a high probability of complete decoding – it is sufficient to achieve almost complete decoding and complete the task with the linear code. This allows the average graph degree to be constant rather than linear in $\log(k)$.

It has been observed [11] that if the graph that remains of the LT decoding algorithm above when it stalls (all constraint node degrees greater than 1) is a good enough expander graph, then a hard decision decoding algorithm can proceed and with high probability complete the decoding process – thus all errors can be corrected. A brief informal explanation of this "expander-based completion" argument is given as it will also apply to certain LDPC arguments.

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

A bipartite graph has expansion (α, β) if for all variable node subsets S of size at most α, the set of neighbor constraint nodes is at least $\beta\ |S|$. For such codes/graphs the following two simple decoding algorithms are proposed [36]:

Algorithm 2.17 (Sequential decoding) If there is a variable node that has more unsatisfied constraint neighbors than satisfied, flip the value of the variable node. Continue until no such variable exists.

Algorithm 2.18 (Parallel decoding) In parallel, flip each variable node that is in more unsatisfied than satisfied constraints and repeat until no such nodes exist.

It can be shown ([11], lemma 2, [36], theorems 10 and 22) that if the graph remaining after LT decoding as discussed above is an (α, β) and is a good enough expander ($\beta > 3/4 + \epsilon$), then the above parallel and sequential decoding algorithms will correct up to αn errors in linear time, n the number of variable nodes. Comments on the probability the remaining graph will be such an expander are given in [11] – and hence this switching of decoding algorithms after the LT decoding will lead to complete decoding.

A brief interesting heuristic discussion of the analysis of the decoding performance of Raptor codes from ([30], section VII) is given. The intermediate and coded edge degree distributions have been noted as $\iota(x)$ and $\omega(x)$, respectively. Let p_{i+1} be the probability an edge in the decoding graph is of right degree one at iteration i. The analysis of Section 2.2 shows that

$$p_{i+1} = \omega(1 - \iota(1 - p_i)) \tag{2.14}$$

and it is desired to design the distribution $\Omega(x)$ such that this quantity decreases. Equation 2.11 giving the binomial moment generating function (mgf), for large n, can be approximated with the standard Poisson mgf

$$\iota(x) = \exp(\alpha(x - 1))$$

where α is the average degree of an input node. This function is also the input node degree distribution since it is easily verified that

$$\iota(x) = \frac{\iota'(x)}{\iota'(1)}.$$

This analysis required the tree involved to have no cycles and the variables involved being independent. Let u_i denote the probability that an input symbol is recovered at step i of the LT decoding algorithm. Then given an unresolved input node of degree d (note input node degrees only change when the node is resolved), then

$$u_i = 1 - (1 - p_i)^d$$

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

and averaging over the input degree distribution gives

$$u_i = 1 - \iota(1 - p_i) = 1 - \exp(-\alpha p_i).$$

Hence $p_i = -\ln(1 - u_i)/\alpha$.
From this relation and Equation 2.14 write

$$\begin{aligned} p_{i+1} &= \omega(1 - \exp(1 - (1 - p_i))) = \omega(1 - \exp(-\alpha p_i)) \\ &= \omega(1 - \exp(-\alpha\{-\ln(1 - u_i)\}/\alpha)) = \omega(1 - (1 - u_i)) \\ &= \omega(u_i) \end{aligned}$$

and so, since $\alpha\omega(x) = (1 + \epsilon)\Omega_D'(x)$ it is argued [30, 31, 33] that, for $k(1 + \epsilon)$ coded (output) nodes the expected fraction of symbols in the input ripple is given, when the expected fraction of input nodes that have already been recovered is x, by

$$1 - x - \exp\Big(-(1 + \epsilon)\Omega_D'(x)\Big).$$

It is further argued this fraction should be large enough to ensure continued operation of the decoding algorithm to completion and it is suggested as a heuristic that this fraction satisfy

$$1 - x - \exp\Big(-(1 + \epsilon)\Omega_D'(x)\Big) \geq c\sqrt{\frac{1 - x}{k}}, \quad x \in [0, 1 - \delta],\ \delta > c/\sqrt{k}.$$

Probability distributions that satisfy this criterion are found to be similar to the Soliton distribution for small values of d and give good performance.

This has been a heuristic overview of aspects of the performance of Raptor codes. The reader is referred to the literature for more substantial treatments on both the performance analysis and code design using such techniques as density evolution and linear programming, in [4, 5, 7, 8, 10, 11, 13, 14, 14, 18, 19, 21, 22, 24, 27, 29, 30, 31, 33, 34, 40].

The remainder of the section considers aspects of Raptor codes that prove important in their application, including the notion of inactivation decoding, the construction of systematic Raptor codes and their standardization.

Inactivation Decoding of LT Codes

It has been noted that BP decoding of LT codes is very efficient but is clearly not ML since it is possible to construct a situation where the decoding process will not complete due to the lack of degree one nodes, yet the decoding matrix formed from the information symbols and received coded symbols is of full rank – and hence ML, which is Gaussian elimination, would yield the unique solution. For a $k \times (k + m)$ matrix, ML would have complexity $O(k^3)$

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

which for the file and symbol sizes considered here, would be far too large. Inactivation decoding seeks to combine the two techniques in a manner that can significantly decrease the overhead and probability of failing to decode for a slight increase of decoding complexity. It takes advantage of the efficiency of the BP decoding and the optimality of ML decoding to yield an efficient version of an ML algorithm.

The following description of the inactivation decoding procedure is slightly unconventional but intuitive. Consider a first phase of BP decoding on the bipartite graph representing n coded symbols (received, known) and k information symbols whose values are sought. To start with, the bipartite graph of the set of k unknown information nodes (registers) is on the left and $n = k(1+\epsilon)$ received coded nodes (registers) is on the right. A coded node of degree one is sought and if found, its value is transferred to the attached information node and its value is then added (XORed) to all its right neighbors. The associated edges involved are all deleted. The process continues until no coded node of degree one is available. In this case suppose a coded symbol of degree two is available (in a well-designed LT code this will often be the case). Referring to Figure 2.5, suppose the register associated with the node contains the symbol r_{i_3}. If the information nodes that are neighbors are u and v, then create a variable x_1 (an inactivation variable) and assign a variable $r_{i_3} \oplus x_1$ to u and x_1 to v. The variable value $r_{i_3} \oplus x_1$ is then added symbolically to all coded neighbors of u and x_1 to all neighbors of v and the BP process is continued as though the value of the variables are known. The process is shown in Figure 2.5. The process continues and each time no node of degree one is available, a

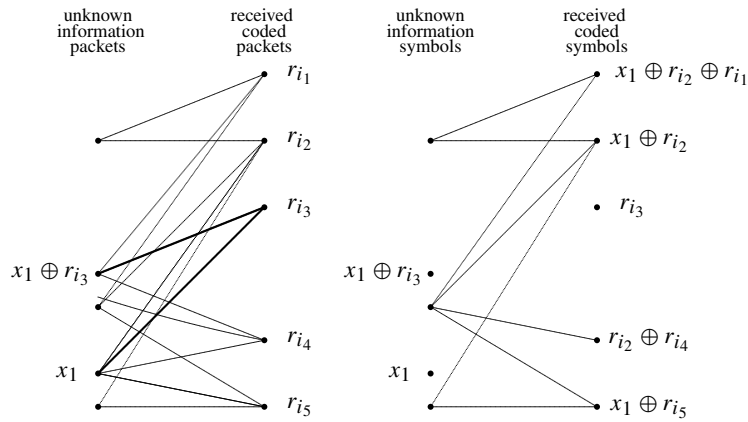


Figure 2.5 One stage of inactivation decoding

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

sufficient number of variables are introduced to allow the process to continue. At the end of the process each information node is "resolved" – although many may contain unknown variables. Typically the number of unknown variables is small. Suppose z inactivation variables have been introduced. Using the original received values of the right nodes and the values assigned to the information nodes, it is possible then to set up a linear relationship between the nodes on the left containing inactivation variables and the known received values on the right. These equations can then be solved, a process equivalent to solving a $z \times z$ matrix equation of complexity $O(z^3)$. If the original received nodes are equivalent to a full-rank system, the process is guaranteed to be successful and is ML. However, since the number of inactivation variables is typically quite small, this use of a combination of BP and ML is typically far more efficient than performing ML on the original received variables.

The above discussion gives an intuitive view of inactivation decoding. More efficient versions are found in the literature, including [1, 12, 20, 33] as well as the original patent description in [26].

Systematic Encoding

There are many situations where a systematic Raptor code is beneficial, i.e. one that with the usual Raptor coding produces the original information symbols among the coded output symbols. A natural technique might be to simply transmit all the information symbols first and then the usual Raptor coded symbols. This turns out not to be a good idea as it gives very poor error performance and requires large overheads to make it work at all. A convincing mathematical proof that this is so is given in [31].

A brief description of the more successful approach of [30] is noted. The terminology of that work is preserved except we retain our notation of information, intermediate and coded symbols/packets for consistency. As before a packet is viewed as a binary word of some fixed length and the only arithmetic operations employed will be the XOR of words or packets of the same length. The approach is to determine a preprocessing of the information packets in such a way as to ensure the original information packets are among the Raptor encoded packets. Thus the Raptor encoding method, and the impact of the distribution derived, is not disturbed ensuring that its promised efficiency is retained. The process is somewhat involved.

First consider the usual Raptor encoding. Let $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ denote the k information packets and consider the Raptor code $(k, C_n, \Omega_D(x))$. Let $\boldsymbol{G}$

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

be a binary (fixed) $n \times k$ generator matrix of the precode C_n assumed to be of full rank.

To initiate the process $k(1+\epsilon)$ LT encoded vectors are generated by sampling $\Omega_D(x)$ this number of times to combine packets (n-tuples chosen randomly from $\mathbb{F}_2{}^n$) to generate the $(k, \Omega_D(x))$ coded vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{k(1+\epsilon)}$. It will be assumed the n-tuples $\boldsymbol{v}_{i_1}, \boldsymbol{v}_{i_2}, \ldots, \boldsymbol{v}_{i_k}$ are linearly independent over $\mathbb{F}_2$. Let $\boldsymbol{A}$ be the $k \times n$ matrix with these independent n-tuples as rows and let $\boldsymbol{R} = \boldsymbol{AG}$, an invertible $k \times k$ matrix.

To encode the k information packets $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$, a vector whose i-th component ([30], algorithm 11) is the information packet $\boldsymbol{x}_i$. Compute the vector $\boldsymbol{y}^t = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k)^t = \boldsymbol{R}^{-1}\boldsymbol{x}^t$ and encode these packets via the C_n code as $\boldsymbol{u} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n)$ where $\boldsymbol{u}^t = \boldsymbol{G} \cdot \boldsymbol{y}^t$. From the packets $\boldsymbol{v}_j$ (vectors) introduced above compute the inner products

$$z_i = \boldsymbol{v}_i \cdot \boldsymbol{u}^t = (v_{i,1}, v_{i,2}, \ldots, v_{i,k}) \cdot \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_k \end{bmatrix}, \quad i = 1, 2, \ldots, k(1+\epsilon)$$

which are the output packets. Notice that from the generating process for the packets $\boldsymbol{v}_j$ this last operation is a $(k, \Omega_D(x))$ LT coding of the coded packets $\boldsymbol{u}_j$. Further packets $z_{k(1+\epsilon)+1}, z_{k(1+\epsilon)+2} \ldots$ are formed by $(k, \Omega_D(x))$ coding of the $\boldsymbol{u}_j$ packets. Thus these output packets are LT codings of the $\boldsymbol{y}_j$ packets.

The point of the process is to notice that the output packet $z_{i_j} = x_j, j = 1, 2, \ldots, k$. To see this ([30], proposition 12) note that since $\boldsymbol{R} = \boldsymbol{A} \cdot \boldsymbol{G}$ and $\boldsymbol{Ry}^t = \boldsymbol{x}^t$. Then $\boldsymbol{A} \cdot \boldsymbol{Gy}^t = \boldsymbol{A} \cdot \boldsymbol{u}^t = \boldsymbol{x}^t$. By construction of the matrix $\boldsymbol{A}$, the j-th row is $\boldsymbol{v}_{i_j}$ and hence this last equation gives the inner product

$$\boldsymbol{v}_{i_j} \cdot \boldsymbol{x}^t = \boldsymbol{x}_j, \quad j = 1, 2, \ldots, k.$$

Thus the Raptor code is systematic.

To decode the $z_j, j = 1, 2, \ldots, k(1+\epsilon)$ one uses the usual decoding procedure for the $(k, C_n, \Omega_D(x))$ code to yield the packets $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k$ and then compute $\boldsymbol{x}^t = \boldsymbol{R} \cdot \boldsymbol{y}^t$ for the original information packets.

It is important to note that the point of this process is to retain the optimality of the distribution of the LT process, yet have certain coded outputs as the input information packets. Of course the process requires the preprocessing of information packets regarding the matrix $\boldsymbol{R}$ and systematic row indices to be known to receivers. The process can be made efficient and is incorporated into many of the standards of Raptor codes. More details on the process are given in [30] which also contains a complexity analysis. Additional information is in the patents regarding these techniques [26, 32, 35].

Standardized Raptor Codes

Raptor codes have been adopted in a large number of mobile and broadcast and multicast standards. These are discussed extensively in the monograph [33]. Two versions of Raptor codes are available commercially, the Raptor version 10, or R10 code and the RaptorQ code. Both are systematic and use versions of inactivation decoding. The R10 code supports blocks of up to 8,192 source symbols per block and up to 65,536 coded blocks. It achieves a probability of failure of 10^{-6} with only a few blocks of overhead across the entire range of block sizes. It is described in detail in [25].

The RaptorQ code is designed for a much wider range of applications and achieves faster encoding and decoding with good failure probability curves. It supports blocks of up to 56,403 source symbols and up to 16,777,216 coded blocks. It uses the observation that the rank properties of random matrices are superior in the sense of achieving full rank for much lower overheads. The field of 256 elements, $\mathbb{F}_{256}$ is used for this code which is described in [17], although the reference [33] gives an excellent description of both of these codes.

Other Aspects of Fountain Codes

The term "online codes" has also been used [15] for the term "fountain codes." Additionally [3] it has been used to describe decoders for fountain codes that adapt to changing conditions of the channel and decoder and are thus able to achieve better performance. It requires some method to measure states and transmission back to the encoder, an important consideration in implementation.

The performance of similar techniques on other channels such as the BSC for error correction remains an interesting possibility.

Comments

The chapter has traced the progress in the remarkable development of erasure-correcting and lost packet codes/fountain codes with linear encoding and decoding complexity. The notion of choosing the parity checks according to a probability distribution in order to ensure, with high probability, a particular decoding algorithm completes, is novel and surprising. The success of both the theoretical and practical developments techniques is impressive.

https://doi.org/10.1017/9781009283403.003 Published online by Cambridge University Press

References

[1] Blasco, F.L. 2017. Fountain codes under maximum likelihood decoding. *CoRR*, abs/1706.08739v1. arXiv:1706.08739v1,2017.

[2] Byers, J.W., Luby, M., Mitzenmacher, M., and Rege, A. 1998. A digital fountain approach to reliable distribution of bulk data. Pages 56–67 of: *Proceedings of the ACM SIGCOMM '98*. ACM, New York.

[3] Cassuto, Y., and Shokrollahi, A. 2015. Online fountain codes with low overhead. *IEEE Trans. Inform. Theory*, **61**(6), 3137–3149.

[4] Karp, R., Luby, M., and Shokrollahi, A. 2004 (June). Finite length analysis of LT codes. Page 39: *Proceedings of the IEEE International Symposium on Information Theory, June 2004*. ISIT.

[5] Kim, S., Lee, S., and Chung, S. 2008. An efficient algorithm for ML decoding of raptor codes over the binary erasure channel. *IEEE Commun. Letters*, **12**(8), 578–580.

[6] Luby, M.G. 2002. LT codes. Pages 271–280 of: *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*.

[7] Luby, M.G., Mitzenmacher, M., Shokrollahi, M.A., Speileman, D., and Stenman, V. 1997. Practical loss-resilient codes. Pages 150–159 of: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*. ACM, New York.

[8] Luby, M.G., Mitzenmacher, M., Shokrollahi, M.A., and Spielman, D. 1998. Analysis of low density codes and improved designs using irregular graphs. Pages 249–258 of: *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*. ACM, New York.

[9] Luby, M.G., Mitzenmacher, M., and Shokrollahi, M.A. 1998. Analysis of random processes via And-Or tree evaluation. Pages 364–373 of: *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1998)*. ACM, New York.

[10] Luby, M.G., Mitzenmacher, M., Shokrollahi, M.A., and Spielman, D.A. 2001. Efficient erasure correcting codes. *IEEE Trans. Inform. Theory*, **47**(2), 569–584.

[11] Luby, M.G., Mitzenmacher, M., Shokrollahi, M.A., and Spielman, D.A. 2001. Improved low-density parity-check codes using irregular graphs. *IEEE Trans. Inform. Theory*, **47**(2), 585–598.

[12] Lzaro, F., Liva, G., and Bauch, G. 2017. Inactivation decoding of LT and Raptor codes: analysis and code design. *IEEE Trans. Commun.*, **65**(10), 4114–4127.

[13] Maatouk, G., and Shokrollahi, A. 2009. Analysis of the second moment of the LT decoder. *CoRR*, abs/0902.3114.

[14] Maneva, E., and Shokrollahi, A. 2006. New model for rigorous analysis of LT-codes. Pages 2677–2679 of: *2006 IEEE International Symposium on Information Theory*.

[15] Maymounkov, P. 2002. Online codes. Technical report. New York University.

[16] Maymounkov, P., and Mazières, D. 2003. Rateless codes and big downloads. Pages 247–255 of: Kaashoek, M.F., and Stoica, I. (eds.), *Peer-to-peer systems II*. Springer, Berlin, Heidelberg.

[17] Minder, L., Shokrollahi, M.A., Watson, M., Luby, M., and Stockhammer, T. 2011 (August). *RaptorQ forward error correction scheme for object delivery*. RFC 6330.
[18] Oswald, P., and Shokrollahi, A. 2002. Capacity-achieving sequences for the erasure channel. *IEEE Trans. Inform. Theory*, **48**(12), 3017–3028.
[19] Pakzad, P., and Shokrollahi, A. 2006 (March). Design principles for Raptor codes. Pages 165–169 of: *2006 IEEE Information Theory Workshop – ITW '06 Punta del Este*.
[20] Paolini, E., Liva, G., Matuz, B., and Chiani, M. 2012. Maximum likelihood erasure decoding of LDPC codes: Pivoting algorithms and code design. *IEEE Trans. Commun.*, **60**(11), 3209–3220.
[21] Pfister, H.D., Sason, I., and Urbanke, R.L. 2005. Capacity-achieving ensembles for the binary erasure channel with bounded complexity. *IEEE Trans. Inform. Theory*, **51**(7), 2352–2379.
[22] Rensen, J.H.S., Popovski, P., and Ostergaard, J. 2012. Design and analysis of LT codes with decreasing ripple size. *IEEE Trans. Commun.*, **60**(11), 3191–3197.
[23] Richardson, T.J., and Urbanke, R.L. 2001. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inform. Theory*, **47**(2), 599–618.
[24] Richardson, T.J., Shokrollahi, M.A., and Urbanke, R.L. 2001. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inform. Theory*, **47**(2), 619–637.
[25] Shokrollahi, A., Stockhammer, T., Luby, M.G., and Watson, M. 2007 (October). *Raptor forward error correction scheme for object delivery*. RFC 5053.
[26] Shokrollahi, A.M., Lassen, S., and Karp, R. 2005 (February). Systems and processes for decoding chain reaction codes through inactivation. US Patent 6856263. www.freepatentsonline.com/6856263.html.
[27] Shokrollahi, M.A. 1999. New sequences of linear time erasure codes approaching the channel capacity. Pages 65–76 of: *Applied algebra, algebraic algorithms and error-correcting codes (Honolulu, HI, 1999)*. Lecture Notes in Computer Science, vol. 1719. Springer, Berlin.
[28] Shokrollahi, M.A. 2000. Codes and graphs. Pages 1–12 of: *In STACS 2000 (invited talk), LNCS No. 1770*.
[29] Shokrollahi, M.A. 2001. Capacity-achieving sequences. Pages 153–166 of: *Codes, systems, and graphical models (Minneapolis, MN, 1999)*. The IMA Volumes in Mathematics and Its Applications, vol. 123. Springer, New York.
[30] Shokrollahi, M.A. 2006. Raptor codes. *IEEE Trans. Inform. Theory*, **52**(6), 2551–2567.
[31] Shokrollahi, M.A. 2009. Theory and applications of Raptor codes. Pages 59–89 of: *MATHKNOW – Mathematics, applied sciences and real life*. Modeling, Simulation & Analysis (MS&A), vol. 3. Springer, Milan.
[32] Shokrollahi, M.A., and Luby, M. 2004 (April). Systematic encoding and decoding of chain reaction codes. US Patent WO 2004/034589 A2. www.freepatentsonline.com/y2005/0206537.html
[33] Shokrollahi, M.A., and Luby, M. 2009. Raptor codes. Pages 213 – 322 of: *Foundations and trends in communications and information theory*, vol. 6. NOW Publishers.

[34] Shokrollahi, M.A., and Storn, R. 2005. *Design of efficient erasure codes with differential evolution*. Springer, Berlin Heidelberg.
[35] Shokrollahi, M.A., Lassen, S., and Karp, R. 2005 (February). Systems and processes for decoding chain reaction codes through inactivation. US Patent 20050206537. www.freepatentsonline.com/y2005/0206537.html.
[36] Sipser, M., and Spielman, D.A. 1996. Expander codes. *IEEE Trans. Inform. Theory*, **42**(6, part 1), 1710–1722.
[37] Tanner, R.M. 1981. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, **27**(5), 533–547.
[38] Tanner, R.M. 1984. Explicit concentrators from generalized N-gons. *SIAM J. Algebraic Discrete Methods*, **5**(3), 287–293.
[39] Tanner, R.M. 2001. Minimum-distance bounds by graph analysis. *IEEE Trans. Inform. Theory*, **47**(2), 808–821.
[40] Xu, L., and Chen, H. 2018. New constant-dimension subspace codes from maximum rank distance codes. *IEEE Trans. Inform. Theory*, **64**(9), 6315–6319.