49

Repeat–Accumulate Codes

In Chapter 1 we discussed a very simple and not very effective method for communicating over a noisy channel: the repetition code. We now discuss a code that is almost as simple, and whose performance is outstandingly good.

Repeat–accumulate codes were studied by Divsalar *et al.* (1998) for theoretical purposes, as simple turbo-like codes that might be more amenable to analysis than messy turbo codes. Their practical performance turned out to be just as good as other sparse-graph codes.

▶ 49.1 The encoder

1. Take K source bits.
$$s_1 s_2 s_3 \ldots s_K$$
2. Repeat each bit three times, giving $N = 3K$ bits.
$$s_1 s_1 s_1 s_2 s_2 s_2 s_3 s_3 s_3 \ldots s_K s_K s_K$$
3. Permute these N bits using a random permutation (a fixed random permutation – the same one for every codeword). Call the permuted string $\mathbf{u}$.
$$u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 \ldots u_N$$
4. Transmit the *accumulated sum*.
$$\begin{aligned} t_1 &= u_1 \\ t_2 &= t_1 + u_2 \ (\mathrm{mod}\, 2) \\ \ldots \quad t_n &= t_{n-1} + u_n \ (\mathrm{mod}\, 2) \quad \ldots \\ t_N &= t_{N-1} + u_N \ (\mathrm{mod}\, 2). \end{aligned} \tag{49.1}$$
5. That's it!

▶ 49.2 Graph

Figure 49.1a shows the graph of a repeat–accumulate code, using four types of node: equality constraints ⊟, intermediate binary variables (black circles), parity constraints ⊞, and the transmitted bits (white circles).

The source sets the values of the black bits at the bottom, three at a time, and the accumulator computes the transmitted bits along the top.

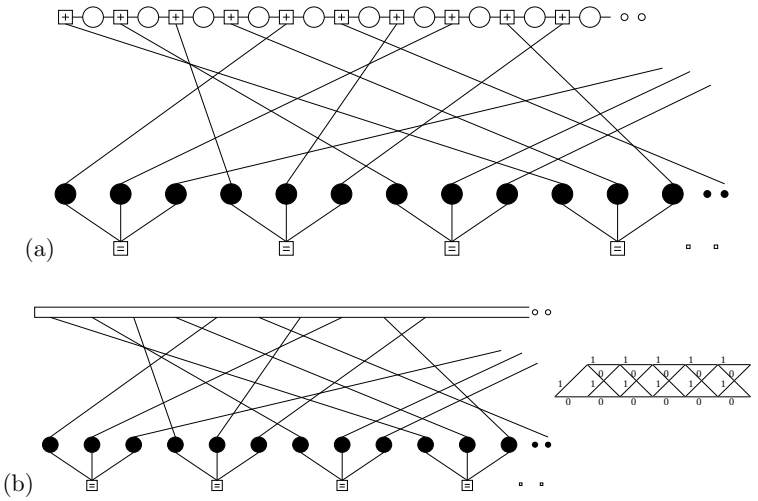


Figure 49.1. Factor graphs for a repeat–accumulate code with rate 1/3. (a) Using elementary nodes. Each white circle represents a transmitted bit. Each ⊞ constraint forces the sum of the 3 bits to which it is connected to be even. Each black circle represents an intermediate binary variable. Each ⊟ constraint forces the three variables to which it is connected to be equal.
(b) Factor graph normally used for decoding. The top rectangle represents the trellis of the accumulator, shown in the inset.

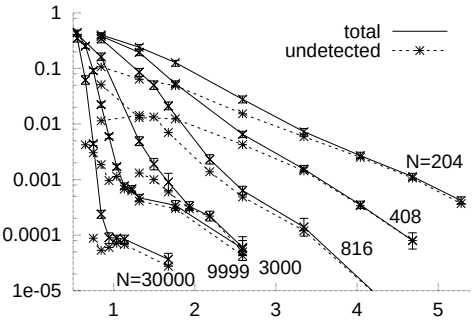


Figure 49.2. Performance of six rate-1/3 repeat–accumulate codes on the Gaussian channel. The blocklengths range from $N = 204$ to $N = 30\,000$. Vertical axis: block error probability; horizontal axis: E_b/N_0. The dotted lines show the frequency of undetected errors.

This graph is a factor graph for the prior probability over codewords, with the circles being binary variable nodes, and the squares representing two types of factor nodes. As usual, each ⊞ contributes a factor of the form $\mathbb{1}[\sum x = 0 \bmod 2]$; each ⊟ contributes a factor of the form $\mathbb{1}[x_1 = x_2 = x_3]$.

▶ 49.3 Decoding

The repeat–accumulate code is normally decoded using the sum–product algorithm on the factor graph depicted in figure 49.1b. The top box represents the trellis of the accumulator, including the channel likelihoods. In the first half of each iteration, the top trellis receives likelihoods for every transition in the trellis, and runs the forward–backward algorithm so as to produce likelihoods for each variable node. In the second half of the iteration, these likelihoods are multiplied together at the ⊟ nodes to produce new likelihood messages to send back to the trellis.

As with Gallager codes and turbo codes, the stop-when-it's-done decoding method can be applied, so it is possible to distinguish between undetected errors (which are caused by low-weight codewords in the code) and detected errors (where the decoder gets stuck and knows that it has failed to find a valid answer).

Figure 49.2 shows the performance of six randomly-constructed repeat–accumulate codes on the Gaussian channel. If one does not mind the error floor which kicks in at about a block error probability of 10^{-4}, the performance is staggeringly good for such a simple code (cf. figure 47.17).

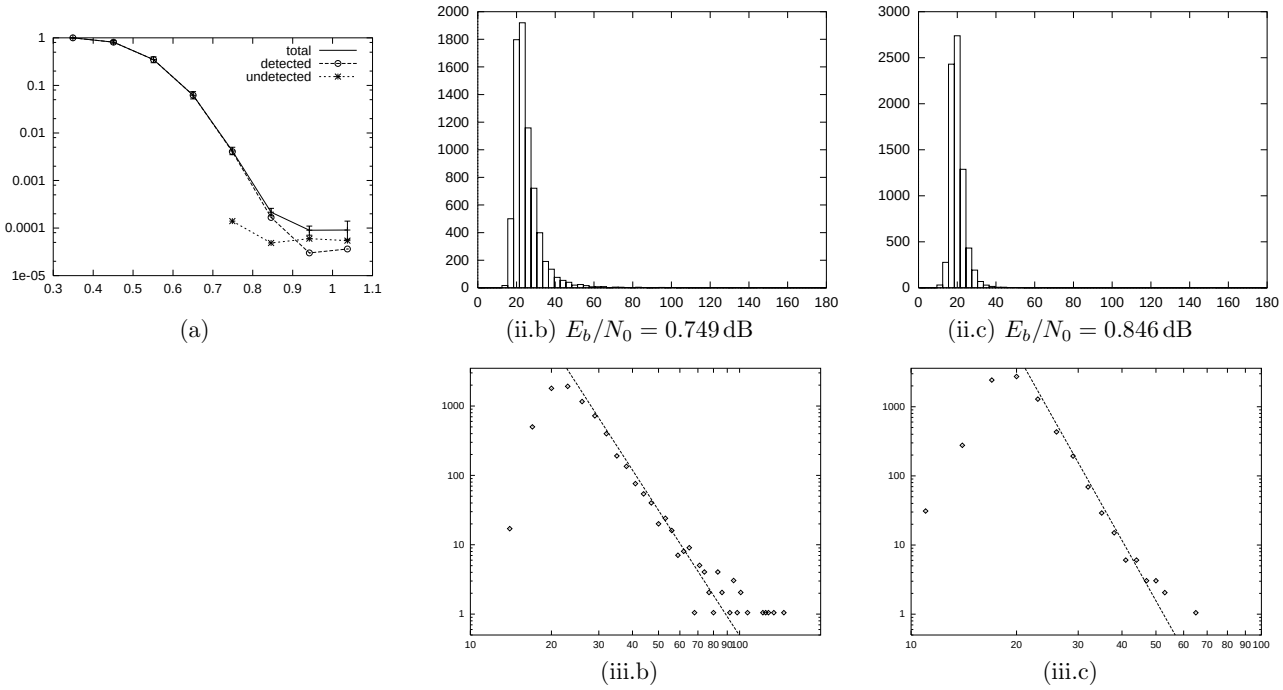


Figure 49.3. Histograms of number of iterations to find a valid decoding for a repeat–accumulate code with source block length $K = 10\,000$ and transmitted blocklength $N = 30\,000$. (a) Block error probability versus signal-to-noise ratio for the RA code. (ii.b) Histogram for $x/\sigma = 0.89$, $E_b/N_0 = 0.749\,\mathrm{dB}$. (ii.c) $x/\sigma = 0.90$, $E_b/N_0 = 0.846\,\mathrm{dB}$. (iii.b, iii.c) Fits of power laws to (ii.b) $(1/\tau^6)$ and (ii.c) $(1/\tau^9)$.

▶ 49.4 Empirical distribution of decoding times

It is interesting to study the number of iterations τ of the sum–product algorithm required to decode a sparse-graph code. Given one code and a set of channel conditions, the decoding time varies randomly from trial to trial. We find that the histogram of decoding times follows a power law, $P(\tau) \propto \tau^{-p}$, for large τ. The power p depends on the signal-to-noise ratio and becomes smaller (so that the distribution is more heavy-tailed) as the signal-to-noise ratio decreases. We have observed power laws in repeat–accumulate codes and in irregular and regular Gallager codes. Figures 49.3(ii) and (iii) show the distribution of decoding times of a repeat–accumulate code at two different signal-to-noise ratios. The power laws extend over several orders of magnitude.

Exercise 49.1.[5] Investigate these power laws. Does density evolution predict them? Can the design of a code be used to manipulate the power law in a useful way?

▶ 49.5 Generalized parity-check matrices

I find that it is helpful when relating sparse-graph codes to each other to use a common representation for them all. Forney (2001) introduced the idea of a *normal graph* in which the only nodes are ⊞ and ⊟ and all variable nodes have degree one or two; variable nodes with degree two can be represented on edges that connect a ⊞ node to a ⊟ node. The *generalized parity-check matrix* is a graphical way of representing normal graphs. In a parity-check matrix, the columns are transmitted bits, and the rows are linear constraints. In a generalized parity-check matrix, additional columns may be included, which represent state variables that are not transmitted. One way of thinking of these state variables is that they are punctured from the code before transmission.

State variables are indicated by a horizontal line above the corresponding columns. The other pieces of diagrammatic notation for generalized parity-

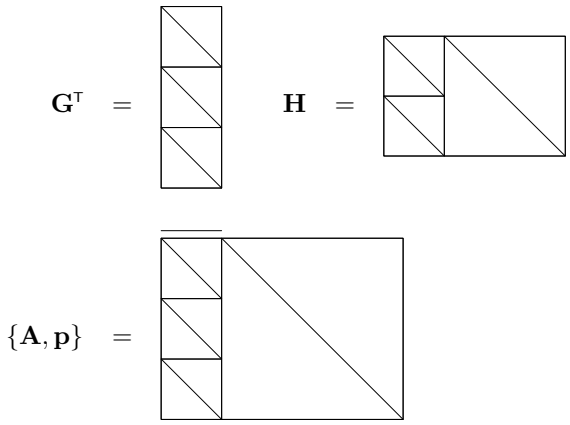


Figure 49.4. The generator matrix, parity-check matrix, and a generalized parity-check matrix of a repetition code with rate $1/3$.

check matrices are, as in (MacKay, 1999b; MacKay *et al.*, 1998):

- A diagonal line in a square indicates that that part of the matrix contains an identity matrix.
- Two or more parallel diagonal lines indicate a band-diagonal matrix with a corresponding number of 1s per row.
- A horizontal ellipse with an arrow on it indicates that the corresponding columns in a block are randomly permuted.
- A vertical ellipse with an arrow on it indicates that the corresponding rows in a block are randomly permuted.
- An integer surrounded by a circle represents that number of superposed random permutation matrices.

Definition. A generalized parity-check matrix is a pair $\{\mathbf{A}, \mathbf{p}\}$, where $\mathbf{A}$ is a binary matrix and $\mathbf{p}$ is a list of the punctured bits. The matrix defines a set of *valid vectors* $\mathbf{x}$, satisfying

$$\mathbf{A}\mathbf{x} = 0; \tag{49.2}$$

for each valid vector there is a codeword $\mathbf{t}(\mathbf{x})$ that is obtained by puncturing from $\mathbf{x}$ the bits indicated by $\mathbf{p}$. For any one code there are many generalized parity-check matrices.

The *rate* of a code with generalized parity-check matrix $\{\mathbf{A}, \mathbf{p}\}$ can be estimated as follows. If $\mathbf{A}$ is $L \times M'$, and $\mathbf{p}$ punctures S bits and selects N bits for transmission ($L = N + S$), then the effective number of constraints on the codeword, M, is

$$M = M' - S, \tag{49.3}$$

the number of source bits is

$$K = N - M = L - M', \tag{49.4}$$

and the rate is greater than or equal to

$$R = 1 - \frac{M}{N} = 1 - \frac{M' - S}{L - S}. \tag{49.5}$$

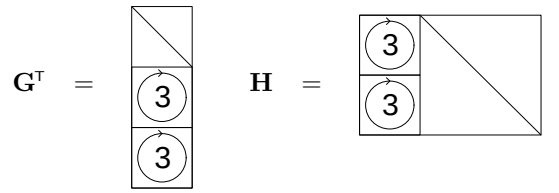


Figure 49.5. The generator matrix and parity-check matrix of a systematic low-density generator-matrix code. The code has rate $1/3$.

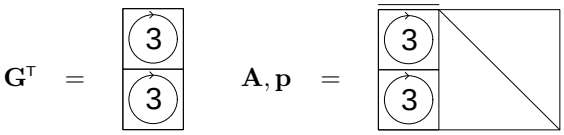


Figure 49.6. The generator matrix and generalized parity-check matrix of a *non-systematic* low-density generator-matrix code. The code has rate $1/2$.

Examples

Repetition code. The generator matrix, parity-check matrix, and generalized parity-check matrix of a simple rate-$1/3$ repetition code are shown in figure 49.4.

Systematic low-density generator-matrix code. In an (N, K) systematic low-density generator-matrix code, there are no state variables. A transmitted codeword $\mathbf{t}$ of length N is given by

$$\mathbf{t} = \mathbf{G}^{\mathsf{T}}\mathbf{s}, \tag{49.6}$$

where

$$\mathbf{G}^{\mathsf{T}} = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{P} \end{bmatrix}, \tag{49.7}$$

with $\mathbf{I}_K$ denoting the $K \times K$ identity matrix, and $\mathbf{P}$ being a very sparse $M \times K$ matrix, where $M = N - K$. The parity-check matrix of this code is

$$\mathbf{H} = [\mathbf{P}|\mathbf{I}_M]. \tag{49.8}$$

In the case of a rate-$1/3$ code, this parity-check matrix might be represented as shown in figure 49.5.

Non-systematic low-density generator-matrix code. In an (N, K) non-systematic low-density generator-matrix code, a transmitted codeword $\mathbf{t}$ of length N is given by

$$\mathbf{t} = \mathbf{G}^{\mathsf{T}}\mathbf{s}, \tag{49.9}$$

where $\mathbf{G}^{\mathsf{T}}$ is a very sparse $N \times K$ matrix. The generalized parity-check matrix of this code is

$$\mathbf{A} = \left[\overline{\mathbf{G}^{\mathsf{T}}}|\mathbf{I}_N\right], \tag{49.10}$$

and the corresponding generalized parity-check equation is

$$\mathbf{A}\mathbf{x} = 0, \quad \text{where } \mathbf{x} = \begin{bmatrix} \mathbf{s} \\ \mathbf{t} \end{bmatrix}. \tag{49.11}$$

Whereas the parity-check matrix of this simple code is typically a complex, dense matrix, the generalized parity-check matrix retains the underlying simplicity of the code.

In the case of a rate-$1/2$ code, this generalized parity-check matrix might be represented as shown in figure 49.6.

Low-density parity-check codes and linear MN codes. The parity-check matrix of a rate-1/3 low-density parity-check code is shown in figure 49.7a.

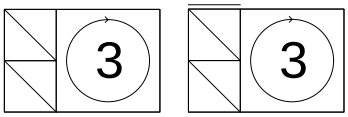
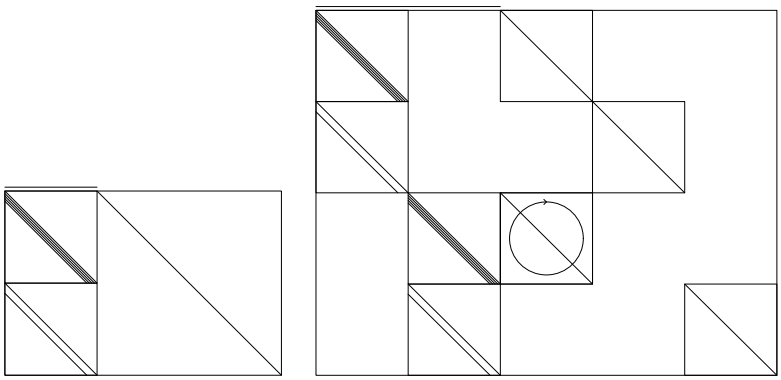


Figure 49.7. The generalized parity-check matrices of (a) a rate-$1/3$ Gallager code with $M/2$ columns of weight 2; (b) a rate-$1/2$ linear MN code.



(a) (b)

Figure 49.8. The generalized parity-check matrices of (a) a convolutional code with rate $1/2$. (b) a rate-$1/3$ turbo code built by parallel concatenation of two convolutional codes.

A linear MN code is a non-systematic low-density parity-check code. The K state bits of an MN code are the source bits. Figure 49.7b shows the generalized parity-check matrix of a rate-1/2 linear MN code.

Convolutional codes. In a non-systematic, non-recursive convolutional code, the source bits, which play the role of state bits, are fed into a delay-line and two linear functions of the delay-line are transmitted. In figure 49.8a, these two parity streams are shown as two successive vectors of length K. [It is common to interleave these two parity streams, a bit-reordering that is not relevant here, and is not illustrated.]

Concatenation. 'Parallel concatenation' of two codes is represented in one of these diagrams by aligning the matrices of two codes in such a way that the 'source bits' line up, and by adding blocks of zero-entries to the matrix such that the state bits and parity bits of the two codes occupy separate columns. An example is given by the turbo code that follows. In 'serial concatenation', the columns corresponding to the *transmitted* bits of the first code are aligned with the columns corresponding to the *source* bits of the second code.

Turbo codes. A turbo code is the parallel concatenation of two convolutional codes. The generalized parity-check matrix of a rate-1/3 turbo code is shown in figure 49.8b.

Repeat–accumulate codes. The generalized parity-check matrices of a rate-1/3 repeat–accumulate code is shown in figure 49.9. Repeat-accumulate codes are equivalent to staircase codes (section 47.7, p.569).

Intersection. The generalized parity-check matrix of the intersection of two codes is made by stacking their generalized parity-check matrices on top of each other in such a way that all the transmitted bits' columns are correctly aligned, and any punctured bits associated with the two component codes occupy separate columns.

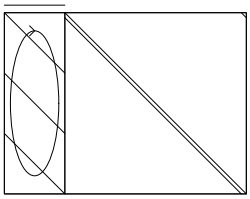


Figure 49.9. The generalized parity-check matrix of a repeat–accumulate code with rate $1/3$.