Relationships between Sparse Graph Codes

David J.C. MacKay *

Abstract: The best error-correcting codes that are known at present are irregular Gallager codes. Finding the best way to construct the parity check matrix of these codes is still an active research area. This note reviews some of the issues, and uses the notation of generalized parity check matrices to relate irregular Gallager codes to turbo codes, repeat-accumulate codes, and the MN codes used by Kanter and Saad.
Nine research questions are posed.

1 Introduction

The central problem of communication theory is to construct an encoding and a decoding system that make it possible to communicate reliably over a noisy channel. The encoding system uses the source data to select a codeword from a set of codewords. The decoding algorithm ideally infers, given the output of the channel, which codeword in the code is the most likely to have been transmitted; for an appropriate definition of distance, this is the 'closest' codeword to the received signal. A good code is one in which the codewords are well spaced apart, so that codewords are unlikely to be confused.

Designing a good and practical error correcting code is difficult because (a) it is hard to find an explicit set of well-spaced codewords; and (b) for a generic code, decoding, *i.e.*, finding the closest codeword to a received signal, is intractable.

However, a simple method for designing good codes, first pioneered by Gallager [1], has recently been rediscovered [2] and generalized. These codes are defined in terms of sparse random graphs. Because the graphs are constructed randomly, the codes are likely to have well-spaced codewords. And because the codes' constraints are defined by a sparse graph, the decoding problem can be solved – almost optimally – by message-passing on the graph. The practical performance of Gallager's codes and their modern cousins [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] is vastly better than the performance of the codes with which textbooks have been filled in the intervening years.

2 Sparse Graph Codes

In a **sparse graph code**, the nodes in the graph represent the transmitted bits and the constraints they satisfy. For a linear code with a codeword length N

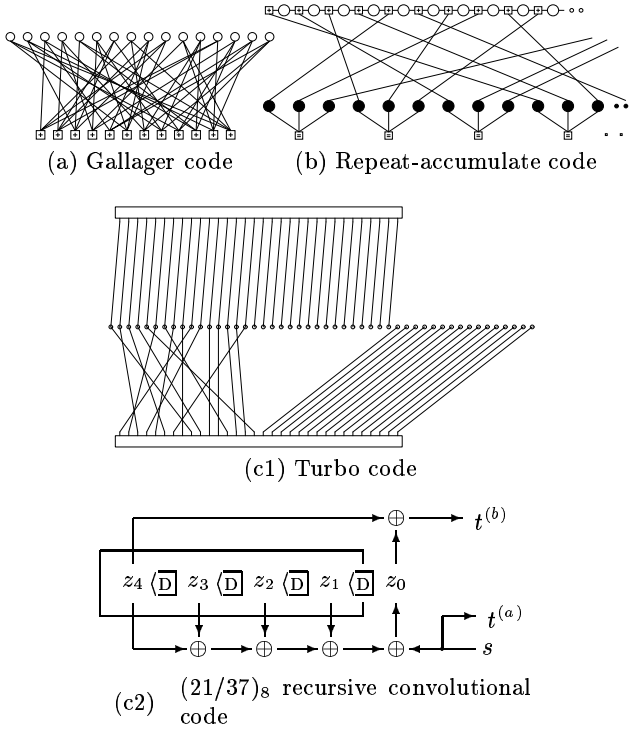


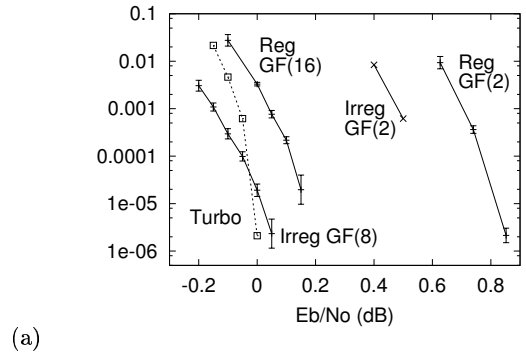
Figure 1. Graphs of three sparse graph codes.

(a) A rate 1/4 low–density parity–check code (Gallager code) with blocklength $N = 16$, and $M = 12$ constraints. Each white circle represents a transmitted bit. Each bit participates in $j = 3$ constraints, represented by $\boxed{+}$ squares. Each $\boxed{+}$ constraint forces the sum of the $k = 4$ bits to which it is connected to be even.

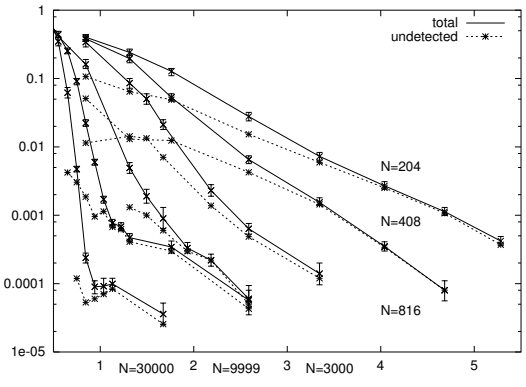
(b) A repeat-accumulate code with rate 1/3. Each white circle represents a transmitted bit. Each black circle represents an intermediate binary variable. Each $\boxed{=}$ constraint forces the variables to which it is connected to be equal.

(c) A turbo code with rate 1/3. (c1) The circles represent the codeword bits. The two rectangles represent rate 1/2 convolutional codes (c2), with the systematic bits $\{t^{(a)}\}$ occupying the left half of the rectangle and the parity bits $\{t^{(b)}\}$ occupying the right half.

*Department of Physics, University of Cambridge, Cavendish Laboratory, Madingley Road, Cambridge, CB3 0HE, United Kingdom. mackay@mrao.cam.ac.uk



(a)



(b)

Figure 2. (a) Bit error probabilities for communication over a Gaussian channel at rate 1/4: left–right: Irregular LDPC, $GF(8)$, transmitted blocklength 24000 bits; JPL turbo, $N = 65536$ bits (dotted line); Regular LDPC, $GF(16)$, $N = 24448$ bits; Irregular LDPC, $GF(2)$, $N = 64000$ bits; Regular LDPC, $GF(2)$, $N = 40000$ bits. [From [17].]

(b) Block error probability of repeat-accumulate codes with rate 1/3 and various blocklengths, versus E_b/N_0. The dotted lines show the frequency of undetected errors.

and rate $R = K/N$, the number of constraints is of order $M = N - K$. [There could be more constraints, if they happen to be redundant.] Any linear code can be described by a graph, but what makes a sparse graph code special is that each constraint involves only a small number of variables in the graph: the number of edges in the graph scales roughly linearly with N, rather than as N^2.

The most famous sparse graph codes are

- Convolutional codes
- Product codes
- Low-density parity-check codes (Gallager codes)
- Turbo codes
- Repeat-accumulate codes

Also worth mentioning are:

- Low-density generator-matrix codes (sometimes called Sourlas codes). These could be split into two categories:

  1. Systematic sparse generator matrix.
  2. Non-systematic sparse generator matrix.

  Both these families of codes have bad distance properties (trivially), and their performance under optimal (maximum likelihood) decoding is bad too.

- MacKay-Neal codes (MN codes) [4, 18]. These are nonlinear codes. A single parameter controls a 'sparsifier', through which the original dense data is passed before encoding. The decoder can then make use of the knowledge that both the source bits and the noise bits are sparse when decoding. The code is defined in terms of a sparse graph such that decoding can be done using the same message-passing algorithms as for Gallager codes, but with both source bits and noise bits being inferred.

  In the special case where an MN code has no sparsfier, the MN code is a linear code.

The objects in the graph

The graph defining a **low–density parity–check code**, or Gallager code [1, 19, 18], contains two types of node: codeword bits, and parity constraints. In a regular (j, k) Gallager code (figure 1a), each codeword bit is connected to j parity constraints and each constraint is connected to k bits. The connections in the graph are made at random.

Repeat-accumulate codes [14] can be represented by a graph with four types of node (figure 1b): equality constraints $\boxed{=}$, intermediate binary variables (black circles), parity constraints $\boxed{+}$, and the transmitted bits (white circles). The encoder sets each group of *intermediate* bits to values read from the source. These bits are put through a fixed random permutation. The transmitted stream is the accumulated sum (modulo 2) of the permuted intermediate bits.

In a **turbo code** [11], the K source bits drive two linear feedback shift registers, which emit parity bits (figure 1c).

All these codes can be decoded by a local message-passing algorithm on the graph, the sum-product algorithm [2, 20], and, while this algorithm is not the optimal decoder, the empirical results are record-breaking. Figure 2 shows the performance of various sparse graph codes on a Gaussian channel. In figure 2(a) turbo codes with rate 1/4 are compared with regular and irregular Gallager codes over GF(2), GF(8) and GF(16). In figure 2(b) the performance of repeat-accumulate codes of various blocklengths and rate 1/3 is shown.

3 Issues in sparse graph codes

In this section we explore a few issues relating to the construction of good sparse graph codes.

3.1 State variables

Both repeat-accumulate (RA) codes and turbo codes have their simplest description in terms of graphs that employ **state variables**. These are variables that do not form part of the transmitted codeword.

Gallager codes do not have state variables. Regular turbo codes outperform regular Gallager codes. Is this because turbo codes have state variables? The best codes known at present are irregular Gallager codes [17, 16]; are even better irregular codes *with* state variables waiting to be created? [Incidentally, it's not essential to have state variables in turbo codes and RA codes: both of these codes can also be represented in terms of low density parity check matrices, as I have pointed out elsewhere.]

Question 1: Are state variables going to be present in the best codes?

3.2 Connectivity

Another distinction can be drawn between Gallager codes, RA codes and turbo codes. In Gallager codes studied to date, apart from those of Kanter and Saad [21], every transmitted bit is connected to *two or more* constraint nodes. In RA codes, every transmitted bit is connected to *two* constraint nodes. In turbo codes, in contrast, all the parity bits have a connectivity of only 1; they dangle from the trellis. So in Gallager codes and RA codes, when messages are passed during decoding, every transmitted bit is a conduit through which decoding information flows; in contrast, turbo codes have backwaters: the message-passing tide laps against the dangling parity bits, but they do not send on messages to anywhere else. I always assumed that these dangling bits of turbo codes were a weakness; and that there would be no advantage to having equivalent bits in Gallager codes, which would correspond to weight-one columns in the parity check matrix. However the results of Kanter and Saad [21] force this assumption to be reassessed.

Question 2: Is there anything intrinsically good or bad about having transmitted bits that dangle from the graph with connectivity one?

One can certainly have too many weight-one columns. Low-density generator-matrix codes are trivially shown to be bad codes, because they have many low weight codewords. Their parity check matrices have M columns of weight one. This is too many. But it is plausible that the best codes might have αM weight-one columns, for some $\alpha < 1$.

Question 3: Do the best Gallager codes have some weight-one columns? If so, how many?

3.3 Weight-two columns

A similar question has been floating around for some time:

Question 4: How many weight-two columns can a Gallager code of rate R have, and still remain a 'good' code?

At one end of the spectrum, Gallager codes having entirely weight-two columns, known as cycle codes, have been studied by Wiberg [22] and others. They are known to have bad distance properties, especially if the weight-2 columns are constructed at random, but this does not immediately imply that they are 'bad' codes in terms of their bit error probability under maximum likelihood decoding. Indeed, for binary symmetric channels with noise level below a threshold, cycle codes are 'good' codes.

At the other extreme, I have for several years imposed on my own Gallager codes the rule that 'the maximum safe number of weight two columns in a binary Gallager code is $M/2$' (where M is the number of rows in the parity check matrix). This rule was motivated by my investigations in 1994 of 'staircase Gallager codes', which have M columns of weight 2 (discussed later in this paper). I observed that such codes often have a small number of low-weight codewords, because the weight-2 columns give a cheap way of stitching up near-codewords created from the other columns of the matrix. I now wonder whether my $M/2$ rule was too strict, and I should have persisted with larger numbers of weight-2 columns.

The automatic profile-optimizing programs of Richardson, Urbanke and Chung often turn out profiles with large fractions of weight-2 columns. Their optimizations are based on belief propagation on infinite graphs, and so do not take into account the worrying issue of low-weight codewords. The optimizations find the threshold at which the *bit error probability* is predicted to go to zero, not the block error probability.

Question 5: Are the optimized profiles found by infinite-graph simulations appropriate for finite graphs, particularly the fraction of low-weight columns? Are there optimization methods that optimize the block error probability instead of the bit error probability?

4 Generalized parity-check matrices

I find that it is helpful when relating sparse graph codes to each other to find *generalized parity-check matrices* for them. 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.

The notation for state variables is a horizontal line above the corresponding columns. The other pieces of diagramatic notation for generalized parity-check matrices are, as in [18, 12]:

- 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.

Mathematically, 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

$$\mathbf{A}\mathbf{x} = 0; \tag{1}$$

for each valid vector there is a codeword $\mathbf{t}(\mathbf{x})$ which is obtained by puncturing from $\mathbf{x}$ the bits indicated by $\mathbf{p}$.

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

$$M = M' - S, \tag{2}$$

the number of source bits is

$$K = N - M = L - M', \tag{3}$$

and the rate is greater than or equal to

$$R = 1 - \frac{M}{N} = 1 - \frac{M' - S}{L - S} \tag{4}$$

4.1 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 3.

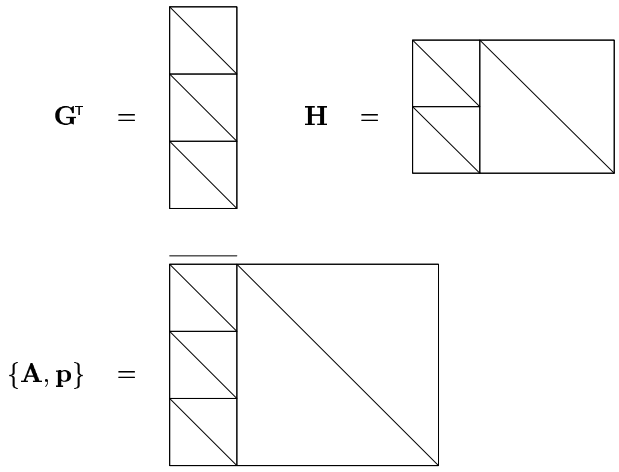


Figure 3. The generator matrix, parity check matrix, and generalized parity check matrix of a repetition code with rate 1/3.

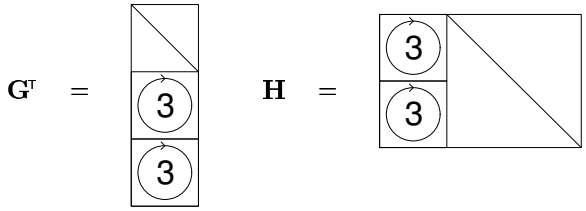


Figure 4. The generator matrix and parity check matrix of a systematic low-density generator-matrix code. The code has rate 1/3.

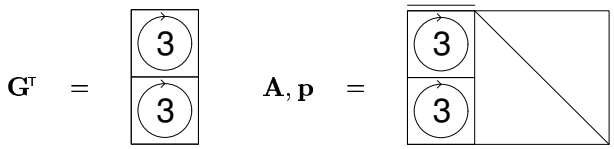


Figure 5. The generator matrix and generalized parity check matrix of a *non-systematic* low-density generator-matrix code. The code has rate 1/2.

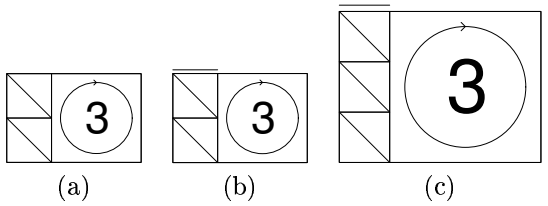


Figure 6. The generalized parity check matrices of (a) a rate-1/3 Gallager code with $M/2$ columns of weight 2; (b) a rate-1/2 linear MN code; (c) a rate-1/3 linear MN code.

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

$$\mathbf{t} = \mathbf{G}^{\mathsf{T}}\mathbf{s}, \tag{5}$$

where

$$\mathbf{G}^{\mathsf{T}} = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{P} \end{bmatrix}, \tag{6}$$

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

$$\mathbf{H} = \begin{bmatrix} \mathbf{P} | \mathbf{I}_M \end{bmatrix}. \tag{7}$$

In the case of a rate 1/3 code, this parity check matrix might be represented as shown in figure 4.

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

$$\mathbf{t} = \mathbf{G}^{\mathsf{T}}\mathbf{s}, \tag{8}$$

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

$$\mathbf{A} = \begin{bmatrix} \overline{\mathbf{G}^{\mathsf{T}}} | \mathbf{I}_N \end{bmatrix}, \tag{9}$$

and the corresponding generalized parity-check equation is

$$\mathbf{A}\mathbf{x} = 0, \tag{10}$$

where $\mathbf{x} = \begin{bmatrix} \mathbf{s} \\ \mathbf{t} \end{bmatrix}$.

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 5.

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 6(a).

A linear MN code is an MN code in which the source bits have density 1/2. Such MN codes can be viewed as non-systematic low-density parity-check codes. The K state bits of an MN code are the source bits. Figure 6(b) shows the generalized parity-check matrix of a rate-1/2 linear MN code, and figure 6(c) shows that of a rate-1/3 linear MN code.

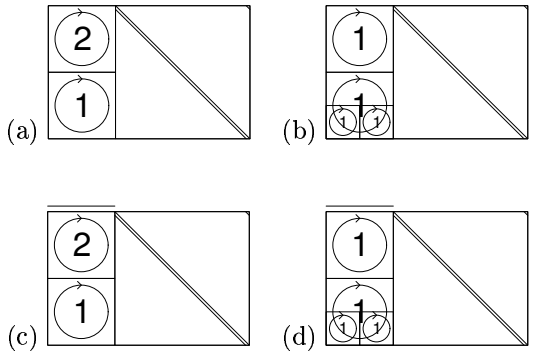


Figure 7. Generalized parity-check matrices for (a) a 'staircase Gallager code' having M columns of weight 2. (b) another staircase Gallager code with identical column-profile and slightly different row-profile. (c,d) Linear MN codes with rate 1/2 derived from the above two rate-1/3 codes.

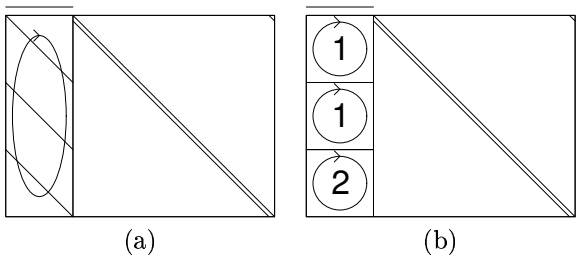


Figure 8. The generalized parity check matrices of (a) a repeat-accumulate code with rate 1/3; (b) an irregular repeat-accumulate code with rate 1/3.

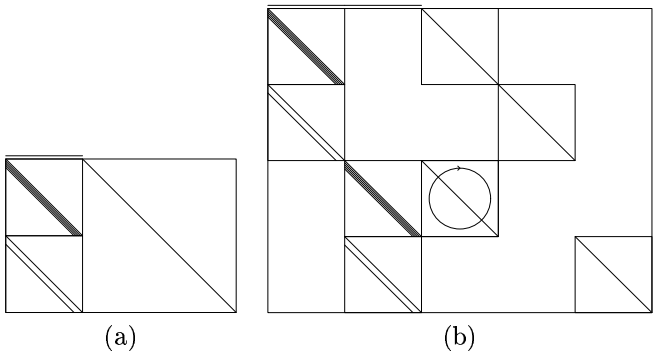


Figure 9. The generalized parity check matrices of (a) a convolutional code with rate 1/2. (b) a rate-1/3 turbo code built by parallel concatenation of two convolutional codes.

'Staircase' MN codes.

MacKay and Neal (unpublished, 1996) studied MN codes in which M' of the columns had weight 2, arranged as shown in figure 7(c). Because such codes were found to have small numbers of low-weight codewords, we did not include them in our reports [2, 18], which focused on codes with generalized parity-check matrices like those in figures 6.

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 9(a), 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.]

In a systematic, recursive convolutional code, the source bits are transmitted, and are fed into a linear-feedback shift-register. The generalized parity-check matrix of such a code is identical to that of the non-systematic, non-recursive convolutional code, and the two codes are identical in that the set of valid codewords are identical; the only difference between the two codes is the mapping of source bits to codewords.

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 below.

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 9(b).

Repeat-accumulate codes.

The generalized parity-check matrices of two rate-1/3 repeat-accumulate codes are shown in figure 8. It is amusing to notice that these repeat-accumulate codes are identical to the staircase linear MN codes shown in figure 7. If we had not been so wary of low-weight codewords, it's possible we would have appreciated the importance of repeat-accumulate codes earlier!

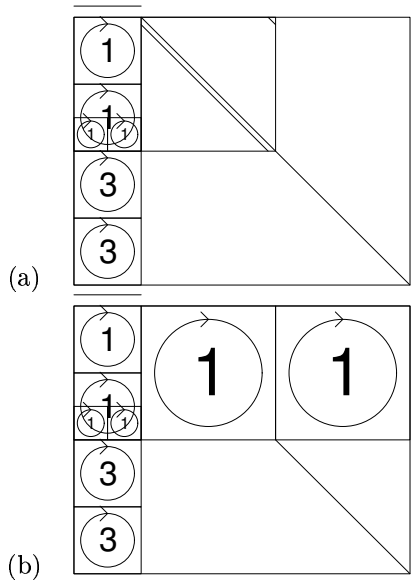


Figure 10. The generalized parity check matrices of (a) the rate 1/4 code of Kanter and Saad [21]; (b) another rate 1/4 code with similar profile that would have many low-weight codewords.

Punctured repeat-accumulate codes.

Puncturing of turbo codes has been extensively studied. I don't know if a similar study of repeat-accumulate codes has been made. It seems plausible that an irregular puncturing of some sort would be useful. Figure 8(b) shows the generalized parity-check matrix of a rate-1/3 code obtained from a rate-1/4 repeat-accumulate code by puncturing K of the transmitted bits.

5 The codes of Kanter and Saad

Some of the questions in section 3 have been prompted by the results reported by Kanter and Saad [21], who describe constructions of irregular MacKay-Neal codes that give promising performance, though having an error-floor at bit-error probabilities of about 10^{-5}.

The construction of the rate 1/4 code of Kanter and Saad [21] is shown in figure 10(a). On a first reading of their paper, I thought the construction was as shown in figure 10(b), which shows a parity check matrix with identical row-weights. [Indeed, I don't think there is sufficient information in the paper to distinguish between these interpretations; but (a) is the correct matrix (David Saad, personal communication).] The matrix (b) has many low-weight codewords.

So, what can we say about these codes? What is different about them, and what is familiar?

- **Use of MN codes rather than Gallager codes.** In general, a matrix that defines an MN code can be used to define Gallager codes. In my experience [18], wherever a matrix makes an impressive MN code, it also makes an even more impressive Gallager code (of higher rate). I am sceptical

about the idea that MN codes could be superior to Gallager codes, but I would be happy to be proved wrong.

- **Use of weight-one and weight-two columns.** This is an innovation, compared with recent work on Gallager codes. There are $M/2$ columns of weight one and $M/2$ of weight two. These columns are arranged in a systematic manner, with the weight-two columns organised in the form of parallel lines of 1s. [Incidentally, this immediately implies that the Gallager code is fast-encoding, which is a nice property.]

- **Arrangement of low weight columns.** None of the weight-one 1s appears in the same row as a weight-two 1. This feature is important to avoid low-weight codewords.

- There are no columns with very high weight (eg 20). The row weights are unequal but not greatly so.

- The systematic construction of the matrix in the form of several blocks is similar to the constructions studied by MacKay, Davey and Wilson [12, 13].

What are some simple ways of describing these codes, in terms of pre-existing codes? Because the right-hand portion of the matrix has a block-diagonal form, we can describe the code as a *parallel concatenation* of two constituent codes. The first code is a 'staircase MN code', whose parity check matrix is shown in figure 7(d). The second is a low-density generator matrix code, whose parity check matrix is shown in figure 5. The parity check matrix in figure 10(a) can be constructed by pasting together those two parity check matrices, with the parity bits appropriately offset.

The staircase code of figure 7(d) is somewhat reminiscent of figure 10. Indeed one way of viewing any staircase code (whether or not the row weight is 3 in all rows) is that it is a repeat-accumulate code in which some of the transmitted bits have been omitted (wherever the row weight is greater than 3, a corresponding number of transmitted bits are omitted). So the transmitted bit is incremented by the sum of several bits, rather than by just one.

Question 6: Are there any advantages in terms of code strength to making the code by parallel concatenation of two or more codes? [I always assumed it would be best to have a single large tangled graph, rather than having the graph separate into two weakly-connected pieces.]

Question 7: Do existing profile-optimizers for Gallager codes include the option of having weight-one columns? Should they?

Question 8: Most existing profile-optimizers for Gallager codes assume that the graph will be constructed at random (the 'Poisson' construction). With large numbers of weight-two and weight-one columns, however, it seems likely that systematic non-Poisson constructions will be needed, to avoid making a code that is infested with low-weight codewords. How should these non-Poisson constructions be optimized?

Question 9: What is the best message-passing schedule for decoding codes that contain the staircase motif? Perhaps the results of Kanter and Saad [21] could be improved by using the sum-product algorithm all the way up and down the trellis of the staircase; this schedule is found to be superior in the case of the repeat-accumulate code, for example (McEliece and students, personal communication).

Acknowledgements

This work was supported by the Gatsby Charitable Foundation and by a partnership award from IBM Research Laboratory, Zürich

References

[1] R. G. Gallager. Low density parity check codes. *IRE Trans. Info. Theory*, IT-8:21–28, Jan 1962.

[2] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 32(18):1645–1646, August 1996. Reprinted *Electronics Letters*, **33**(6):457–458, March 1997.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *Proc. 1993 IEEE International Conference on Communications, Geneva, Switzerland*, pages 1064–1070, 1993.

[4] D. J. C. MacKay and R. M. Neal. Good codes based on very sparse matrices. In Colin Boyd, editor, *Cryptography and Coding. 5th IMA Conference*, number 1025 in Lecture Notes in Computer Science, pages 100–111. Springer, Berlin, 1995.

[5] D. A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6.1):1723–1731, November 1996.

[6] M. Sipser and D. A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6Pt1):1710–1722, 1996.

[7] M. C. Davey and D. J. C. MacKay. Low density parity check codes over GF(q). *IEEE Communications Letters*, 2(6):165–167, June 1998.

[8] M. G. Luby, M. Mitzenmacher, M. Amin Shokrollahi, D. A. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theo ry of Computing (STOC)*, 1997.

[9] B. J. Frey and D. J. C. MacKay. Trellis-constrained codes. In *Proceedings of the 35th Allerton Conference on Communication, Control, and Computing, Sept. 1997*, 1998. Available at `http://www.cs.utoronto.ca/~frey`.

[10] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Improved low-density parity-check codes using irregular graphs and belief propagation. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, page 117, 1998.

[11] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding: Turbo-codes. *IEEE Transactions on Communications*, 44:1261–1271, October 1996.

[12] D. J. C. MacKay, S. T. Wilson, and M. C. Davey. Comparison of constructions of irregular Gallager codes. In *Proceedings of the 36th Allerton Conference on Communication, Control, and Computing, Sept. 1998*, pages 220–229, Monticello, Illinois, 1998. Allerton House.

[13] D. J. C. MacKay, S. T. Wilson, and M. C. Davey. Comparison of constructions of irregular Gallager codes. *IEEE Transactions on Communications*, 47(10):1449–1454, October 1999.

[14] D. Divsalar, H. Jin, and R. J. McEliece. Coding theorems for 'turbo-like' codes. In *Proceedings of the 36th Allerton Conference on Communication, Control, and Computing, Sept. 1998*, pages 201–210, Monticello, Illinois, 1998. Allerton House.

[15] Matthew C. Davey and David J. C. MacKay. Watermark codes: Reliable communication over insertion/deletion channels. In *ISIT 2000*, 2000.

[16] R. Urbanke, T. Richardson, and Amin Shokrollahi. Design of provably good low density parity check codes. Submitted, 1999.

[17] M. C. Davey and D. J. C. MacKay. Low density parity check codes over GF(q). In *Proceedings of the 1998 IEEE Information Theory Workshop*, pages 70–71. IEEE, June 1998.

[18] D. J. C. MacKay. Good error correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, 1999.

[19] R. G. Gallager. *Low Density Parity Check Codes.* Number 21 in Research monograph series. MIT Press, Cambridge, Mass., 1963.

[20] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng. Turbo decoding as an instance of Pearl's 'belief propagation' algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152, February 1998.

[21] I. Kanter and D. Saad. Error-correcting codes that nearly saturate Shannon's bound. *Physics Review Letters*, 83(13):2660–2663, 1999.

[22] N. Wiberg. *Codes and Decoding on General Graphs.* PhD thesis, Dept. of Electrical Engineering, Linköping, Sweden, 1996. Linköping studies in Science and Technology. Dissertation No. 440.

This paper appears in the proceedings of the workshop *Information-Based Induction Sciences (IBIS 2000)*, July 17-18 2000, Shizuoka, Japan.

The author's website can be found at `http://wol.ra.phy.cam.ac.uk/`.