Comparison of Constructions of Irregular Gallager Codes

David J. C. MacKay, Simon T. Wilson and Matthew C. Davey

Department of Physics, University of Cambridge

Cavendish Laboratory, Madingley Road,
Cambridge, CB3 0HE, United Kingdom.

mackay|stw11|mcdavey@mrao.cam.ac.uk

Submitted to IEEE Transactions on Communications 30 July 1998.

Abstract

The low density parity check codes whose performance is closest to the Shannon limit are 'Gallager codes' based on irregular graphs. We compare alternative methods for constructing these graphs and present two results.

First, we find a 'super–Poisson' construction which gives a small improvement in empirical performance over a random construction.

Second, whereas Gallager codes normally take N^2 time to encode, we investigate constructions of regular and irregular Gallager codes which allow more rapid encoding and have smaller memory requirements in the encoder. We find that these 'fast–encoding' Gallager codes have equally good performance.

1 Introduction

Gallager codes [3, 4] are low density parity check codes constructed at random subject to constraints on the weight of each row and of each column. The original *regular* Gallager codes have very sparse random parity check matrices with uniform weight t per column and t_r per row. [We will also use the term 'regular' for codes which have nearly uniform weight columns and rows — for example, codes which have some weight 2 columns and some weight 3 columns.] These codes are asymptotically good, and can be practically decoded with Gallager's sum–product algorithm giving near Shannon limit performance when large block lengths are used [8, 9, 6]. Regular Gallager codes have also been found to be competitive codes for short block-length CDMA applications [11].

Recent advances in the performance of Gallager codes are summarised in figure 1. The rightmost curve shows the performance of a regular binary Gallager code with rate 1/4. The best known binary Gallager codes are *irregular* codes whose parity check matrices have *nonuniform* weight per column [5]; the performance of one such code is shown by the second curve from the right. The best known Gallager codes of all are Gallager codes defined over finite fields $GF(q)$ [2, 1]. The remaining two solid curves in figure 1 show the performance of a regular Gallager code over $GF(16)$ [2] and an irregular code over $GF(8)$ with bit error probability of 10^{-4} at $E_b/N_0 = -0.05$dB [1]. In comparing this code with the rate 1/4 Turbo code shown by the dotted line, the following points

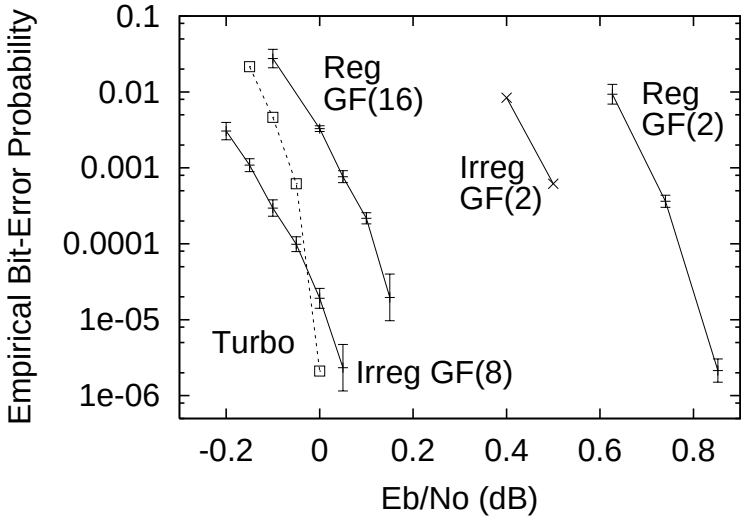


Figure 1: Empirical results for Gaussian Channel, Rate 1/4 Left–Right : Irregular LDPC, $GF(8)$ blocklength 24000 bits; JPL Turbo, blocklength 65536 bits; Regular LDPC, $GF(16)$, blocklength 24448 bits; Irregular LDPC , $GF(2)$, blocklength 64000 bits; Regular LDPC, $GF(2)$, blocklength 40000 bits. (Reproduced from [1].)

should be noted. (1) The transmitted blocklength of the irregular Gallager code is only 24000 bits, whereas that of the Turbo code is 65536 bits. (2) The errors made by the Gallager codes were all detected errors, whereas Turbo codes make undetected errors at high signal to noise ratio. This difference is not caused by a difference in the decoding algorithm: both codes are decoded by the sum–product algorithm [10]. Turbo Codes make undetected errors because *they have low–weight codewords*. For Gallager codes, the rate of occurrence of undetected errors is extremely small because they have good distance properties (the minimum distance scales linearly with the blocklength) [4]. In all our experiments with Gallager codes of block length greater than 1000 and column weight at least 3, undetected errors have never occurred.

The excellent performance of irregular Gallager codes is the motivation for this paper, in which we explore ways of further enhancing these codes.

The irregular codes of Luby, Mitzenmacher, Shokrollahi and Spielman [5] have parity check matrices with both nonuniform weight per row and nonuniform weight per column. It has not yet been established whether both of these non-uniformities are desirable. In our experience with codes for noisy channels, performance is more sensitive to the distribution of column weights. In this paper we concentrate on irregular codes with the weight per row as uniform as possible.

We can define an irregular Gallager code in two steps. First, we select a *profile* that describes the desired number of columns of each weight and the desired number of rows of each weight. The parity check matrix of a code can be viewed as defining a bipartite graph with 'bit' vertices corresponding to the columns and 'check' vertices corresponding to the rows. Each non–zero entry in the matrix corresponds to an edge connecting a bit to a check. The profile specifies the degrees of the vertices in this graph.

Second, we choose a *construction method*, that is, a pseudo–random algorithm for putting edges between the vertices in a way that satisfies the constraints. [In the case of non-binary Gallager codes we also need to choose an algorithm for assigning values to the non–zero entries in the matrix.]

This paper has two parts. In the first part (section 3) we compare alternative construction methods for a fixed profile in order to find out whether the construction method matters. In the second part (section 4) we examine regular and irregular constructions which lend themselves to rapid encoding. One motivation for this second study is that the only drawback of regular Gallager codes compared to Turbo codes for CDMA applications appears to be their greater encoding complexity [11].

In the experiments presented here, we study binary codes with rate 1/2 and blocklength about $N = 10,000$. We simulate an additive white Gaussian noise channel in the usual way [2] and examine the block error probability as a function of the signal to noise ratio. The error bars we show are one standard deviation error bars on the estimate of the logarithm of the block error probability $\hat{p}$, defined thus: when we observe r failures out of n trials, $p_\pm = \hat{p}\exp(\pm\sigma_{\log p})$ where $\sigma_{\log p} = \sqrt{(n-r)/(rn)}$.

2 Constructions

We compare the following methods.

Poisson: The edges are placed 'completely at random' subject to the profile constraints, and the rule that you can't put two edges between one pair of vertices, which would correspond to a double entry in the parity check matrix. One way to implement a Poisson construction is to make a list of all the columns in the matrix, with each column appearing in the list a number of times equal to its weight, then make a similar list of all the rows in the matrix, each row appearing with multiplicity equal to its weight, and then map one list onto the other by a random permutation, taking care not to create duplicate entries [5].

A variation of this construction is to require that no two columns in the parity check matrix have an overlap greater than one, *i.e.,* forbid cycles of length 4 in the graph. [Similar to construction 1A in [9].] A second variation requires that the graph to have no cycles of length less than some l. [Similar to construction 1B in [9].] This constraint can be quite hard to enforce if the profile includes high weight rows or columns.

Permutations. We can build parity check matrices by superposing random permutation matrices [4]. The convenience of this method depends on the profile. There are many ways of laying out these permutation matrices to satisfy a given profile. We will distinguish *'super–Poisson'* and *'sub–Poisson'* constructions.

- In a super–Poisson construction, the distribution of high weight columns per row has greater variance than a Poisson distribution;
- In a sub–Poisson construction, the distribution of high weight columns per row has smaller variance than a Poisson distribution.

Profile 3	Column weight	Fraction of columns	Row weight	Fraction
	3	1	6	1
Profile 93	Column weight	Fraction of columns	Row weight	Fraction
	3	11/12	7	1
	9	1/12		

Table 1: The two profiles studied in this paper

3 Comparing Poisson, 'super–Poisson' and 'sub–Poisson' constructions

3.1 Profiles and constructions studied in this paper

3.1.1 Regular codes: 3 and 33

As our baseline we study regular Gallager codes with weight per column exactly $t = 3$ and weight per row exactly $t_r = 6$. We construct parity check matrices satisfying this profile from permutation matrices in two ways, labelled '3' and '33', shown diagrammatically in the upper panels of figure 2. In the figures, a square containing an integer (for example, '3') denotes the superposition inside that square of that number of random permutation matrices. The matrices are generated at random subject to the constraint that no two non–zero entries coincide.

3.1.2 Irregular codes: 93p, 93a, 93x and 93y

We chose the profile '93' shown in table 1. It has columns of weight 9 and of weight 3; all rows have weight 7. Note that this profile only differs from the regular profile '3' in that some extra 1s are added to $\frac{1}{12}$ of the columns. We emphasize that this profile has not been carefully optimized, so the results of this paper should not be taken as describing the best that can be done with irregular binary Gallager codes. We chose this profile because it lends itself to interesting experiments.

We will refer to the bits that connect to 9 checks as 'elite' bits. We use four different constructions that match this profile, named as follows. These constructions are depicted diagrammatically in the upper panels of figure 2.

Poisson: 93p. In this construction, while most checks will connect to one or two elite bits, a fraction of them will connect to more than two elite bits, and some will connect to none.

Sub–Poisson: 93a. This construction allocates exactly one or two elite bits to each check.

Super–Poisson: 93x and 93y are respectively moderately and very super–Poisson. In 93y, one third of the checks are connected to *four* elite bits, one third are connected to *one*, and one third are connected to *none*.

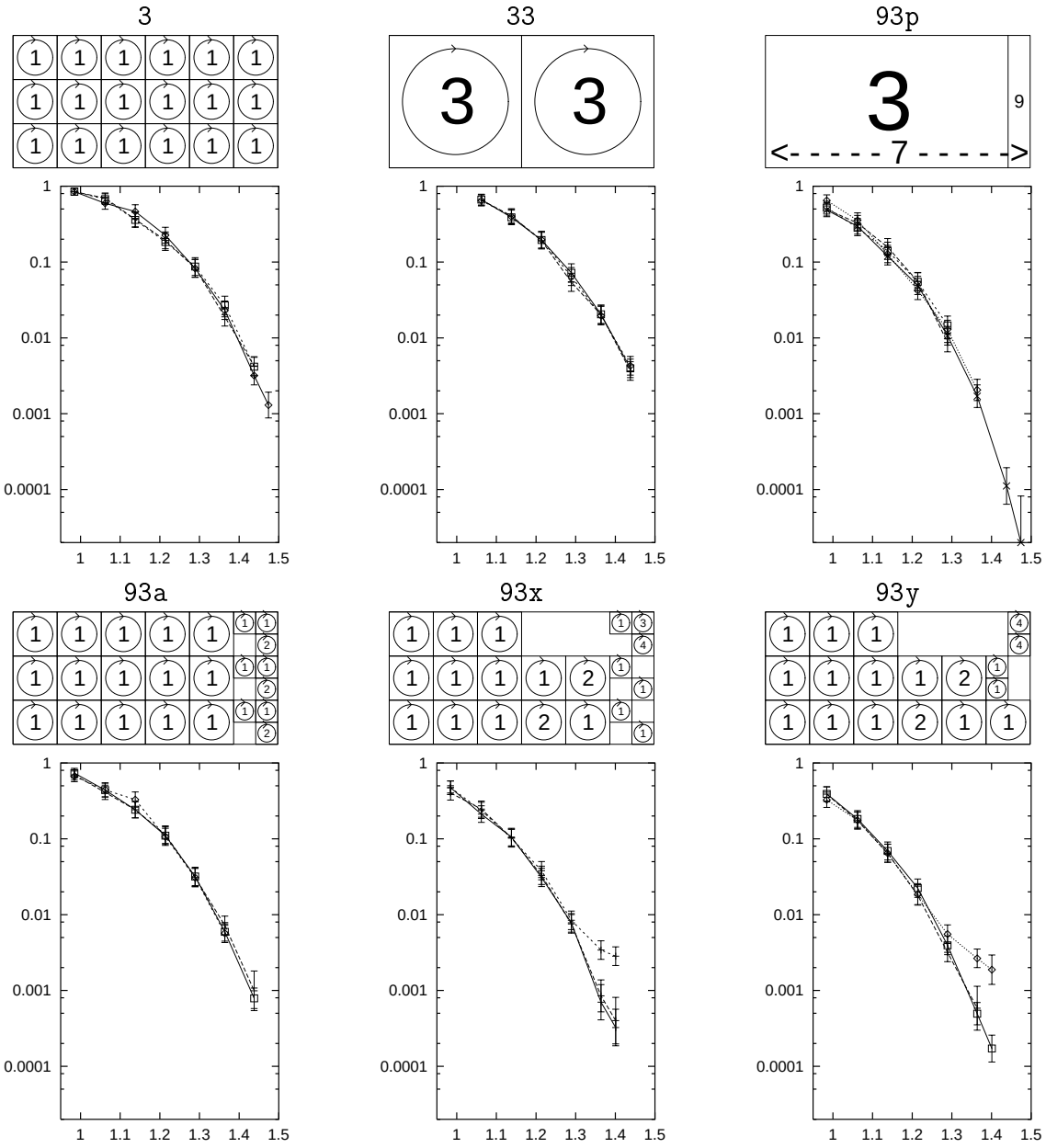


Figure 2: Upper panels: constructions of regular and irregular codes. Lower panels: performance of these codes. The construction types shown are regular, (3, 33), Poisson (93p), sub–Poisson (93a), super–Poisson (93x), and super–Poisson (93y).

Notation for upper panels for all constructions except 93p: an integer represents a number of permutation matrices superposed on the surrounding square. Horizontal and vertical lines indicate the boundaries of the permutation blocks. Notation for the Poisson construction 93p: integers '3' and '9' represent column weights. The integer '7' represents the row weight.

Lower panels show the performance of several random codes of each construction. Vertical axis: block error probability. Horizontal axis: E_b/N_0 in dB. All codes have $N = 9972$, and $K = M = 4986$.

All errors were detected errors, as is usual with Gallager codes.

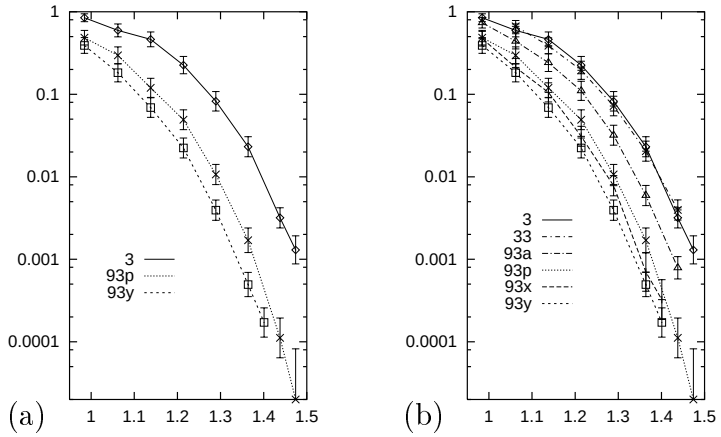


Figure 3: (a) Comparison of one representative of each of the constructions `3` (regular), `93p` (Poisson) and `93y` (super–Poisson). (b) shows representatives of all six constructions in figure 2. Vertical axis: block error probability. Horizontal axis: E_b/N_0 in dB.

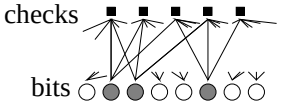
3.2 Results

3.2.1 Variability within each construction

For each construction we created several codes in order to assess the variability of performance within each ensemble. All codes studied were of rate 1/2, with blocklength $N = 4986$. The results are shown in figure 2. We see no significant variability among the `3`, `33`, `93i` (Poisson) or `93a` (sub–Poisson) codes. But among the super–Poisson codes, `93x` and `93y`, there is some variability, with some codes showing an error floor.

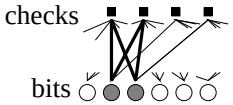
3.2.2 Explanation of error floors

In both cases the error floor has a simple cause. The most frequent error under these conditions is a reconstructed transmission which differs from the correct codeword in exactly three bits — the same three bits every time. These bits, which have weight 3 columns in the parity check matrix, are connected to just five checks with the topology shown below.



(1)

If the three bits shaded grey are flipped into the wrong state then the syndrome vector changes sign in the fifth check only. The sum–product algorithm is unable to extricate itself from this state. As the block length of the code is increased, the probability of this topology's occurrence falls. It is also possible to modify the construction algorithm for Gallager codes such that cycles of length 4, like this



(2)

are forbidden (as in construction 1A of [9]). This modification is sufficient to prevent the topology shown in (1) from occurring. In principle it is possible for a code to have a minimum distance of 4 even when the minimum cycle length is 6. However, for randomly

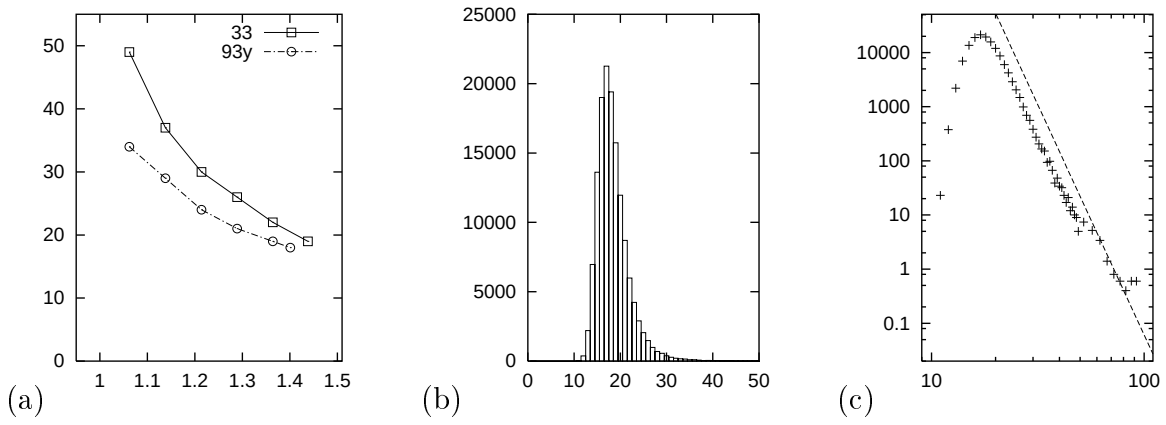


Figure 4: (a) Comparison of irregular 93y with regular 33 code. Vertical axis: median number of iterations. Horizontal: E_b/N_0 in dB. (b) Histogram of number of iterations for 93y code at $E_b/N_0 = 1.4$. (c) Log/log plot of iterations histogram showing that the tail of the distribution is well approximated by a power law. The straight line has slope -8.5. Above 50, iterations were binned into intervals of 5.

constructed codes the minimum distance increases linearly with the blocklength, for almost all codes [4].

We discard the two codes with error floors in the subsequent comparisons.

3.2.3 Comparison of constructions

The six families are compared with each other in figure 3. There are no detectable differences between the regular codes 3 and 33. There is a clear ranking of the other constructions, as follows:

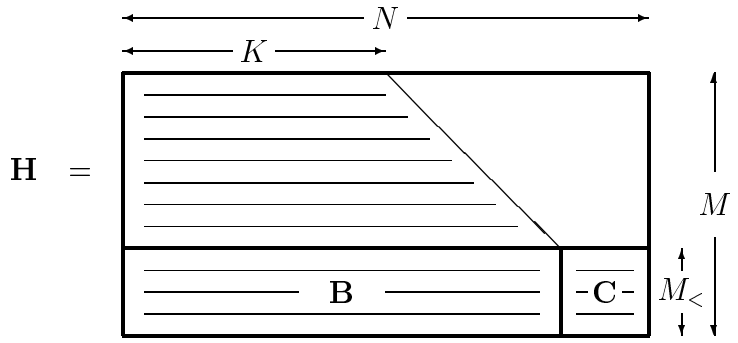
$$\texttt{3} < \texttt{93a} < \texttt{93p} < \texttt{93x} < \texttt{93y}$$

Thus we find that, at least for the 93 profile, sub–Poisson constructions are inferior to Poisson constructions, and super–Poisson constructions are significantly superior. In the case of 93y we see an improvement of about 0.05dB.

3.2.4 Decoding times

Not only do these irregular codes outperform the regular codes, they require fewer iterations as illustrate in figure 4(a) which compares the median number of iterations of the irregular code 93y and the regular code 33. Note that 93y requires 7/6 times more operations per iteration due to the increased weight of the matrix, so the total decoding times are similar.

Figures 4(b) and (c) show that the distribution of decoding times is heavy tailed. At $E_b/N_0 = 1.4$ the tail is well approximated by the power law: $P(\tau) \sim \tau^{-8.5}$, where τ is the number of iterations. At $E_b/N_0 = 1.2$ the distribution is heavier tailed, and we have $P(\tau) \sim \tau^{-5}$.



Encoding procedure:

Bits $t_1 \ldots t_K$ are defined to be source bits.

Bits $t_{K+1} \ldots t_{N-M_<}$ are set in sequence, using the mth parity check to determine t_{K+m}.

This costs $(N - M_<)t_r$ computational operations, where t_r is the typical weight per row.

Bits $t_{N-M_<+1} \ldots t_N$ are set equal to:

$$\mathbf{C}^{-1}\mathbf{B}\mathbf{t}^* \bmod 2$$

where $\mathbf{t}^* = (t_1 \ldots t_{N-M_<})^\intercal$ and $\mathbf{C}^{-1}$ is the inverse of $\mathbf{C}$ in modulo 2 arithmetic.

$\mathbf{C}^{-1}$ can be stored in $M_<^2$ bits of memory. The product $\mathbf{B}\mathbf{t}^*$ can be computed in $M_< t_r$ computational operations, and the multiplication by $\mathbf{C}^{-1}$ takes $M_<^2$ operations.

Figure 5: Upper panel: General form of a fast–encoding Gallager code. Horizontal stripes indicate low–weight rows. The diagonal line is a line of 1s. The matrices $\mathbf{B}$ and $\mathbf{C}$ are of dimension $M_< \times (N - M_<)$ and $M_< \times M_<$ respectively. Lower panel: The fast encoding method to generate a codeword, and its computational cost, assuming an appropriate representation of the sparse matrix.

3.2.5 Unequal error protection

We can compare the bit error–rate of elite bits with that of standard bits. We find that when decoding fails, elite bits are more likely than standard bits to be correctly decoded. In the case of construction `93x` we found that at $E_b/N_0 = 1.3$dB the probability of an elite bit being in error, given that the block was incorrectly decoded, was 0.012 whereas standard bits had an error rate of 0.065. Differences remain at small E_b/N_0. For example, at $E_b/N_0 = 0.7$dB the error rates are 0.035 and 0.097.

4 Fast–encoding Gallager codes

One of the possible drawbacks of Gallager codes is that their encoding time generally scales as N^2. Inspired by Spielman's [12] work, we have investigated constructions of Gallager codes whose profiles are similar to or identical to the `3` and `93` profiles above, but which are fast–encoding. The general form of parity check matrix for a fast–encoding Gallager code is shown in figure 5. The parity check matrix has an almost lower–triangular structure which allows all but a small number $M_<$ of the parity bits to be computed using sparse operations. The final $M_<$ parity bits can be computed in $M_<^2$ binary operations.

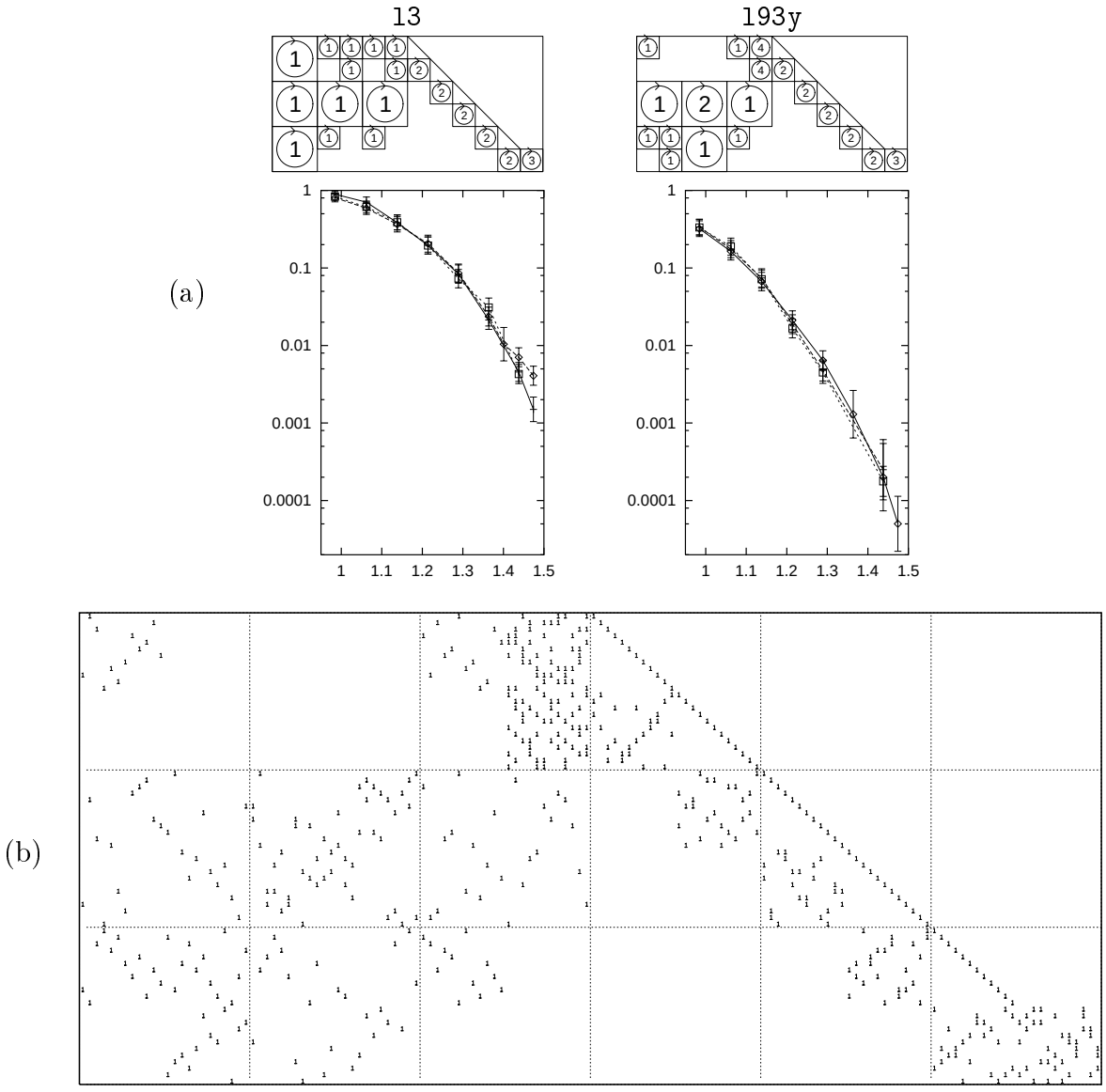


Figure 6: (a) Upper panels: construction methods `13` and `193y`. As in figure 2, an integer represents a number of permutation matrices superposed on the surrounding square. A diagonal line represents an line of 1s. Horizontal and vertical lines indicate the boundaries of the permutation blocks. Lower pictures: Variability of performance among `13` and `193y` codes. Vertical axis: block error probability. Horizontal axis: E_b/N_0 in dB. All codes have $N = 9972$, and $K = M = 4986$.
(b) Example of a parity check matrix with $N = 144$ made using construction `193y`.

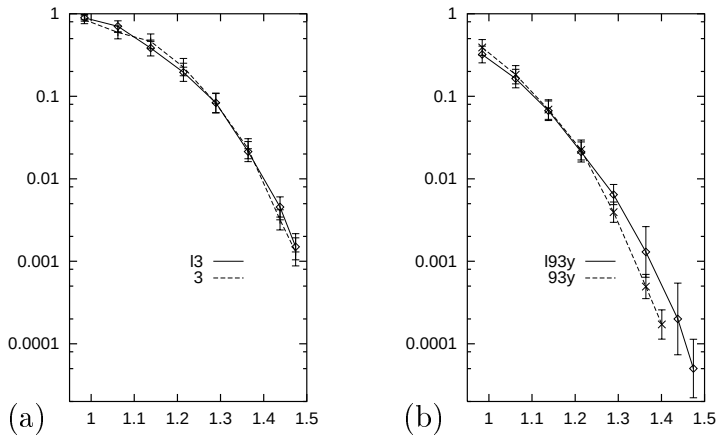


Figure 7: (a) Comparison of `l3` with an ordinary `3` code. (b) Comparison of `l93y` with the best super–Poisson `93y` code. Vertical axis: block error probability. Horizontal: E_b/N_0/dB.

If $M_<$ were as small as $\sqrt{M}$ then the codes would be linear–time encodeable.

We introduce two constructions, `l3` and `l93y` ('`l`' for linear), shown diagrammatically in figure 6(a). Construction `l3` has profile identical to construction `3`. Construction `l93y` has a profile identical to the `93` codes, and is most similar to `93y`. figure 6(b) shows an example of a matrix made using construction `l93y` and Figure 6(a) shows the variability of performance of these codes.

Figure 7 shows that the fast–encoding codes have almost the same performance as the ordinary–encoding codes.

5 Discussion

The detection of error floors in some Gallager codes reminds us that it is a good idea, though not essential, to avoid cycles of length 4 when building these codes, as was standard practice in [3, 9, 6]. This is not so easy to enforce in irregular Gallager codes with high weight columns, but the present results indicate that what is really important is to ensure that no pairs of low weight columns (with weights less than 4) have overlaps greater than one.

The improvement of 0.05dB of the super–Poisson construction compared with a Poisson construction of irregular codes may not be viewed as practically useful, but we think it is important to be aware of all the methods available for enhancing code performance, especially when these methods involve no added cost at the encoder or decoder. It will be interesting to see how beneficial the super–Poisson concept is for other irregular profiles. We are currently investigating the creation of irregular Turbo codes which make use of super–Poisson constructions (work in progress with B. J. Frey).

The discovery that we can make fast–encoding Gallager codes whose performance is just as good as ordinary Gallager codes may have immediate practical importance for mobile communications where computer power consumption must be minimized. Both the memory requirements and the CPU requirements at the encoder of our fast–encoding codes are substantially smaller. The static memory required for encoding is proportional to $M_<^2 + Mt_r$, where t_r is the typical weight per row in the parity check matrix. This com-

pares to $M(N-M)$ for storing the generator matrix. The particular constructions used in this paper would allow encoding using roughly thirty–six times fewer computational operations, since we chose $M_< = M/6$. This reduction would be more than sufficient to cancel out the factor of fourteen encoding complexity disadvantage with respect to Turbo codes of the example mentioned in [11]. The smaller the ratio $M_</M$, the greater the reduction in encoding cost. It will be interesting to investigate how small $M_<$ can be made without deterioration in performance.

References

[1] 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*. IEEE, June 1998.

[2] 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.

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

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

[5] 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 1998 IEEE International Symposium on Information Theory, Boston USA*, page 117.

[6] D. J. C. MacKay. Good error correcting codes based on very sparse matrices. *IEEE transactions on Information Theory*, 1999. In press. Available from `http://wol.ra.phy.cam.ac.uk/`.

[7] D. J. C. MacKay and J. Lafferty. Codes from Cayley graphs. Work in progress, 1997.

[8] 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.

[9] 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.

[10] 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, 1998.

[11] V. Sorokine, F. R. Kschischang, and S. Pasupathy. Gallager codes for CDMA applications II: Implementations, complexity and system capacity. *IEEE Trans. Communications*, 1998. submitted.

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