B. J. Frey and D. J. C. MacKay (1999) In *Proceedings of the* 37^{th} *Allerton Conference on Communication, Control and Computing 1999*, Allerton House, Illinois.

Irregular Turbocodes

Brendan J. Frey
Computer Science, University of Waterloo
Electrical and Computer Engineering, University of Illinois at Urbana
http://www.cs.uwaterloo.ca/~frey

David J. C. MacKay
Department of Physics, Cavendish Laboratories
Cambridge University
http://wol.ra.phy.cam.ac.uk/mackay

Abstract

Recently, several groups have increased the coding gain of iteratively decoded Gallager codes (low density parity check codes) by varying the number of parity check equations in which each codeword bit participates. In regular turbocodes, each "systematic bit" participates in exactly 2 trellis sections. We construct irregular turbocodes with systematic bits that participate in varying numbers of trellis sections. These codes can be decoded by the iterative application of the sum-product algorithm (a low-complexity, more general form of the turbodecoding algorithm). By making the original rate 1/2 turbocode of Berrou *et al.* slightly irregular, we obtain a coding gain of 0.15 dB at a block length of $N = 131,072$, bringing the irregular turbocode within 0.3 dB of capacity. Just like regular turbocodes, irregular turbocodes are linear-time encodable.

1 Introduction

Recent work on irregular Gallager codes (low density parity check codes) has shown that by making the codeword bits participate in varying numbers of parity check equations, significant coding gains can be achieved [1–3]. Although Gallager codes have been shown to perform better than turbocodes at BERs below 10^{-5} [4][1], until recently Gallager codes performed over 0.5 dB worse than turbocodes for BERs greater than 10^{-5}. However, in [3], Richardson *et al.* found irregular Gallager codes that perform 0.16 dB *better* than the original turbocode at BERs greater than 10^{-5} [5] for a block length of $N \approx 131,072$.

[1]Gallager codes to not exhibit decoding errors, only decoding failures, at long block lengths with $N > 5,000$.

In this paper, we show that by tweaking a turbocode so that it is irregular, we obtain a coding gain of 0.15 dB for a block length of $N = 131,072$. For example, an $N = 131,072$ irregular turbocode achieves $E_b/N_0 = 0.48$ dB at BER $= 10^{-4}$, a performance similar to the best irregular Gallager code published to date [3]. By further optimizing the degree profile, the permuter and the trellis polynomials, we expect to find even better irregular turbocodes. Like their regular cousins, irregular turbocodes exhibit a BER flattening due to low-weight codewords.

2 Irregular turbocodes

In Fig. 1, we show how to view a turbocode so that it can be made irregular. The first picture shows the set of systematic bits (middle row of discs) being fed directly into one convolutional code (the chain at the top) and being permuted before being fed into another convolutional code (the chain at the bottom). For a rate 1/2 turbocode, each constituent convolutional code should be rate 2/3 (which may, for example, be obtained by puncturing a lower-rate convolutional code).

Since the order of the systematic bits is irrelevant, we may also introduce a permuter before the upper convolutional code, as shown in the second picture. In the third picture, we have simply drawn the two permuters and convolutional codes side by side.

For long turbocodes, the values of the initial state and the final state of the convolutional chains do not significantly influence performance (*e.g.*, see [6]). So, as shown in the fourth picture, we can view a turbocode as a code that copies the systematic bits, permutes both sets of these bits, and then feeds them into a convolutional code. We refer to this turbocode as "regular", since each systematic bit is copied exactly once.

The final picture illustrates one way the above turbocode can be made irregular. Some of the systematic bits are "tied" together, in effect causing some systematic bits to be replicated more than once. Notice that too keep the rate of the overall code fixed at 1/2, some extra parity bits must be punctured.

More generally, an *irregular turbocode* has the form shown in Fig. 2, which is a type of "trellis-constrained code" as described in [7]. We specify a *degree profile*, $f_d \in [0,1]$, $d \in \{1, 2, \ldots, D\}$. f_d is the fraction of codeword bits that have degree d and D is the maximum degree. Each codeword bit with degree d is repeated d times before being fed into the permuter. Several classes of permuter lead to linear-time encodable codes. In particular, if the bits in the convolutional code are partitioned into "systematic bits" and "parity bits", then by connecting each parity bit to a degree 1 codeword bit, we can encode in linear time.

The average codeword bit degree is

$$\bar{d} = \sum_{d=1}^{D} d \cdot f_d \tag{1}$$

The overall rate R of an irregular turbocode is related to the rate R' of the convolutional code and the average degree $\bar{d}$ by

$$\bar{d}(1 - R') = 1 - R. \tag{2}$$

So, if the average degree is increased, the rate of the convolutional code must also be increased to keep the overall rate constant. This can be done by puncturing the convolutional code or by designing a new, higher rate convolutional code.

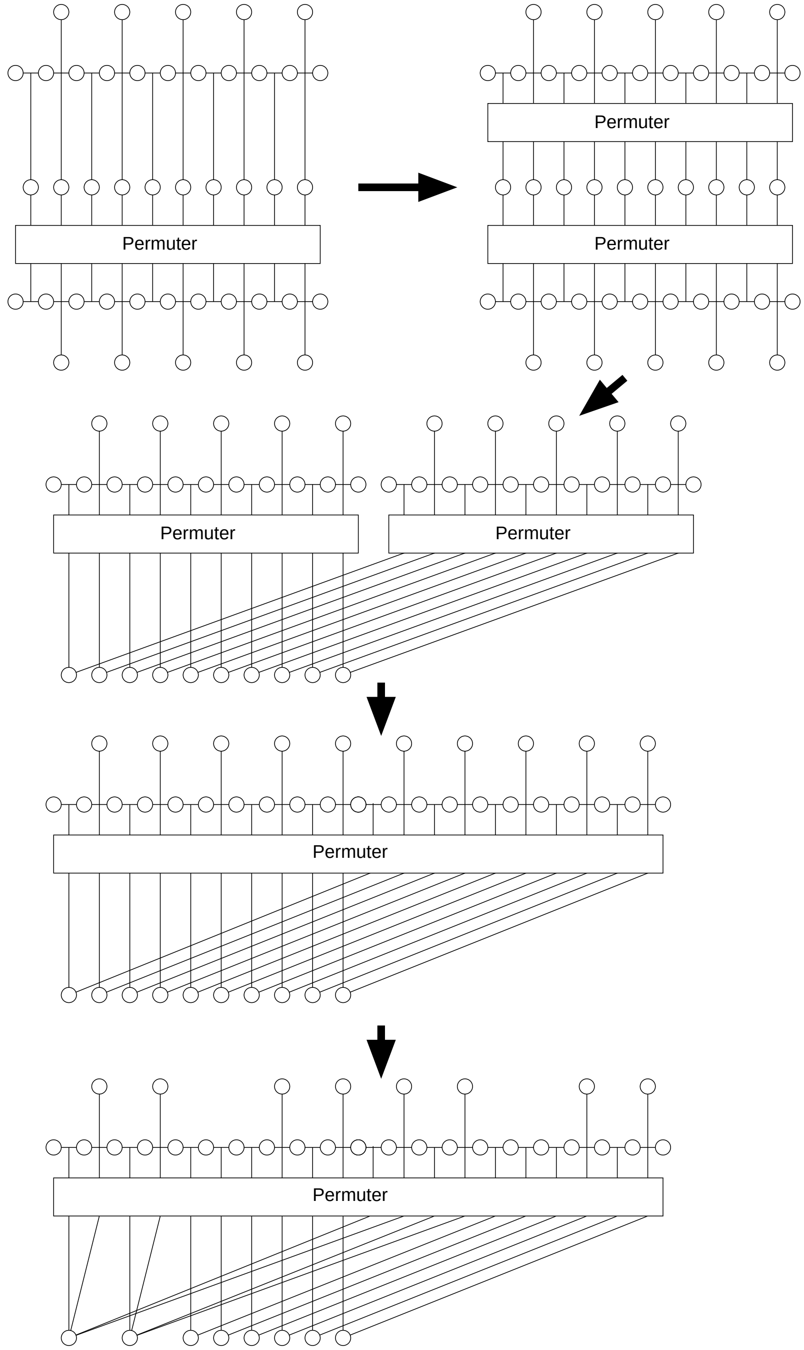

Figure 1: The first 4 pictures show that a turbocode can be viewed as a code that copies the systematic bits, permutes both sets of these bits and then feeds them into a convolutional code. The 5th picture shows how a turbocode can be made irregular by "tying" some of the systematic bits together, *i.e.*, by having some systematic bits replicated more than once. Too keep the rate fixed, some extra parity bits must be punctured. Too keep the block length fixed, we must start with a longer turbocode.

3 Decoding irregular turbocodes

Fig. 2 can be interpreted as the graphical model [6, 8–10] for the irregular turbocode. Decoding consists of the iterative application of the sum-product algorithm (a low-complexity, more general form of turbodecoding) in this graph.

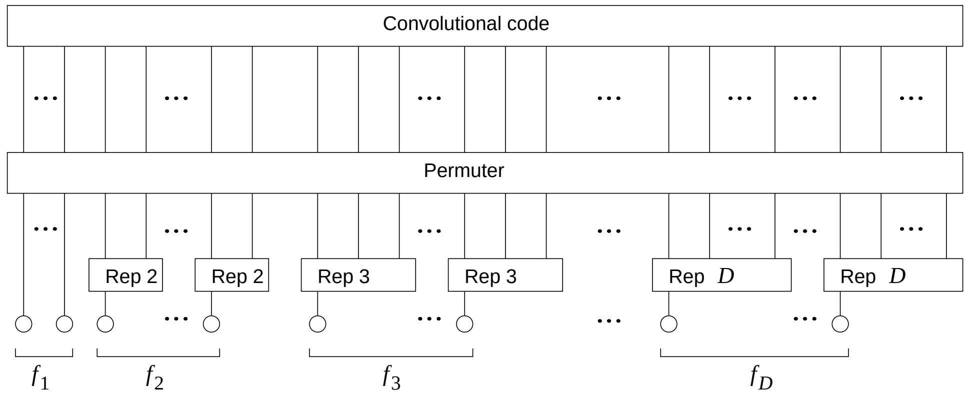

Figure 2: A general *irregular turbocode*. For $d = 1, \ldots, D$, fraction f_d of the codeword bits are repeated d times, permuted and connected to a convolutional code.

After receiving the channel output, the decoder computes the channel output log-likelihood ratios for the N codeword bits,

$$L_1^0, L_2^0, \ldots, L_N^0, \tag{3}$$

and then repeats each log-likelihood ratio appropriately. If codeword bit i has degree d, we set

$$L_{i,1} \leftarrow L_i^0,\ L_{i,2} \leftarrow L_i^0,\ \ldots,\ L_{i,d} \leftarrow L_i^0. \tag{4}$$

Next, the log-likelihood ratios are permuted and fed into the BCJR algorithm [11] for the convolutional code. The BCJR algorithm assumes the inputs are *a priori* log-probability ratios and uses the forward-backward algorithm [12] to compute a set of *a posteriori* log-probability ratios. If codeword bit i has degree d, the algorithm produces d *a posteriori* log-probability ratios,

$$L'_{i,1}, L'_{i,2}, \ldots, L'_{i,d}. \tag{5}$$

For a regular turbocode, there are just two *a posteriori* log-probability ratios, $L'_{i,1}$ and $L'_{i,2}$, for each degree 2 bit and they correspond to the "extrinsic information" produced by each constituent convolutional code.

The current estimate of the log-probability ratio for bit i given the channel output is

$$\hat{L}_i \leftarrow L_i^0 + \sum_{k=1}^{d} (L'_{i,k} - L_{i,k}). \tag{6}$$

To compute the inputs to the BCJR algorithm needed for the next iteration, we subtract off the corresponding outputs from the BCJR algorithm produced by the previous iteration:

$$L_{i,k} \leftarrow \hat{L}_i - L'_{i,k}. \tag{7}$$

So, each iteration consists of computing the inputs to the BCJR algorithm, permuting the inputs, applying the BCJR algorithm, permuting the outputs of the BCJR algorithm, and taking the repetitions into account to combine the outputs to form estimates of the log-probability ratios of the codeword bits given the channel output.

In our simulations, after each iteration, we check to see if the current decision gives a codeword. If it does, the iterations terminate and otherwise, the decoder iterates further until some maximum number of iterations is reached and a decoding failure is declared.

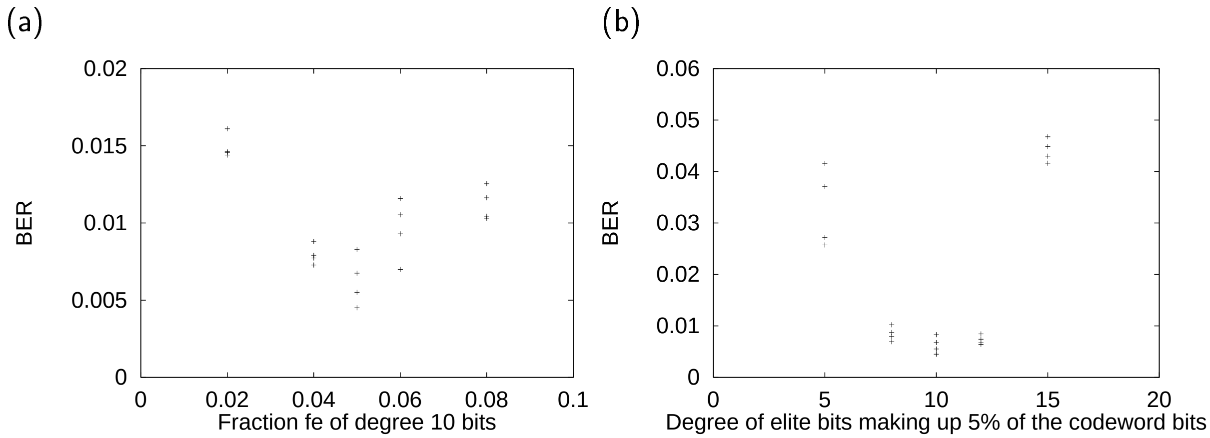

Figure 3: (a) shows the effect of changing the fraction of elite bits on the BER, while keeping the degree of the elite bits fixed at 10. (b) shows the effect of changing the degree of the elite bits on the BER, while keeping the fraction of elite bits fixed at 0.05. For each fraction and degree, the performance of 4 randomly drawn permuters is shown.

4 Selecting the profile

Finding a good profile is not trivial, since the best profile will depend on the parameters of the convolutional code, the permuter and the distortion measure (bit error rate, block error rate, decoding failure rate, high-weight decoding failure rate, *etc.*)

The results we report in this paper were obtained by making small changes to a block length $N = 10,000$ version of the original rate $R = 1/2$ turbocode proposed by Berrou *et al.*. In this turbocode, $f_1 = f_2 = 1/2$ (see Fig. 2) and the convolutional code polynomials are 37 and 21 (octal). The taps associated with polynomial 37 are connected to the degree 2 codeword bits, 1/2 of the taps associated with polynomial 21 are connected to the degree 1 bits, and the remaining 1/2 of the taps associated with polynomial 21 are punctured, giving the required convolutional code rate of $R' = 2/3$.

To simplify our search, we considered profiles where besides degrees 1 and 2, only one other degree, e for "elite", had a nonzero fraction. So, for a code with overall rate R and fraction f_e of degree e elite bits, we have

$$f_1 = 1 - R = 1/2,$$
$$f_2 = 1 - f_1 - f_e = 1/2 - f_e. \quad (8)$$

In this restricted class of codes, irregularity is governed by two parameters, e and f_e.

From (2) it is clear that when the average degree is increased, the rate of the convolutional code must also be increased to keep the overall rate at 1/2. We increased the rate of the punctured convolutional code by further puncturing the taps associated with polynomial 21 to obtain a convolutional code with rate

$$R' = 1 - \frac{1 - R}{\bar{d}} = 1 - \frac{1/2}{1/2 + 2(1/2 - f_e) + ef_e}. \quad (9)$$

So, in the codes we explored, the level of puncturing was quite high and some extra low-weight codewords were introduced.

To begin with, we made an irregular turbocode with $e = 10$ (chosen using intuition) and varied f_e from 0.02 to 0.08 while measuring the BER at $E_b/N_0 = 0.6$ dB. In each

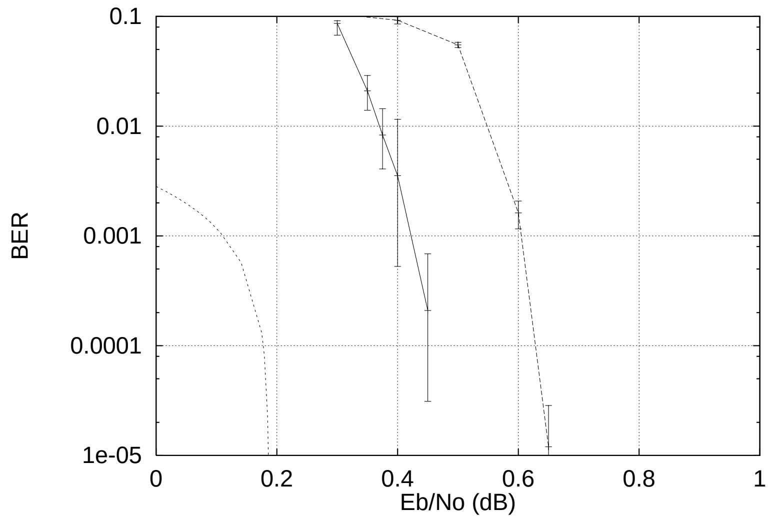

Figure 4: Performance of the original block length $N = 131,072$ turbocode (dashed line) and one of its irregular cousins (solid line). These results are for irregular turbocodes obtained by tweaking the original turbocode – we are currently searching for optimal degree profiles, permuters and trellis polynomials.

experiment, we simulated enough blocks to obtain a relatively small confidence interval. The results are shown in Fig. 3a, which indicates that for degree 10 elite bits, the best fraction is roughly 0.05. Next, we kept the fraction of elite bits fixed at $f_e = 0.05$ and we varied the degree of the elite bits. The results are shown in Fig. 3b, which indicates that for a fraction of 0.05, the best degree is roughly 10.

These results show that for $e = 10$, $f_e = 0.05$ is a good fraction and that for $f_e = 0.05$, $e = 10$ is a good degree. However, values of e and f_e that give good profiles are probably correlated, so we are currently extending our search.

5 Results

Fig. 4 shows the simulated BER-E_b/N_0 curves for the original block length $N = 131,072$ regular turbocode (dashed line) and its irregular cousin (solid line), using profile $e = 10$, $f_e = 0.05$.

The irregular turbocode clearly performs better than the regular turbocode for BER $> 10^{-4}$. At BER $= 10^{-4}$, the $N = 131,072$ irregular turbocode is 0.3 dB from capacity, a 0.15 dB improvement over the regular turbocode.

For high E_b/N_0, most of the errors for the irregular turbocode were due to low-weight codewords. According to preliminary results, the distribution of error weights appears to indicate that the flattening effect for the particular $N = 131,072$ irregular turbocode we constructed occurs at a *higher* BER than it does for the regular turbocode. However, the flattening effect is highly sensitive to the technique used to construct the permuter (we drew it at random) and the design of the convolutional code (we just further punctured the convolutional code used in the original turbocode). We are currently experimenting with techniques for lowering the level of the flattening effect (*e.g.*, see [13]).

6 Conclusions

We have shown that by making the original, regular turbocode irregular, a coding gain of 0.15 dB is obtained, bringing the irregular turbocode within 0.3 dB of capacity at a BER of 10^{-4}. This irregular turbocode performs in the same regime as the best known irregular Gallager code.

We emphasize that we obtained these results by tweaking the regular turbocode originally introduced by Berrou *et al.*. We believe we will be able to improve these performance curves significantly, both by exploring the polynomials used in the convolutional code and by adjusting the degree profile and the permuter structure. (One way to speed up the search is to extend the method of "density evolution" [3] to models with state.) In particular, we are investigating ways to select the permuter and the polynomials to eliminate low-weight codewords, thus reducing the flattening effect [13–15]. We are also studying ways of constraining the degree 1 "parity" bits (*i.e.*, increasing their degree) to eliminate low-weight codewords.

References

[1] 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 IEEE International Symposium on Information Theory*, 1998.

[2] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of constructions of irregular Gallager codes," *IEEE Transactions on Communications*, vol. 47, October 1999.

[3] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low density parity check codes." Submitted to *IEEE Transactions on Information Theory*, July 1999.

[4] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, pp. 399–431, March 1999.

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

[6] B. J. Frey, *Graphical Models for Machine Learning and Digital Communication*. Cambridge MA.: MIT Press, 1998. See `http://www.cs.utoronto.ca/~frey`.

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

[8] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *European Transactions on Telecommunications*, vol. 6, pp. 513–525, September/October 1995.

[9] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 219–230, February 1998.

[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*, vol. 16, February 1998.

[11] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, March 1974.

[12] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *Annals of Mathematical Statistics*, vol. 37, pp. 1559–1563, 1966.

[13] M. Öberg and P. H. Siegel, "Application of distance spectrum analysis to turbocode performance improvement," in *Proceedings of the* 35^{th} *Allerton Conference on Communication, Control and Computing 1997*, 1998.

[14] D. Divsalar and F. Pollara, "Turbo-codes for PCS applications," in *Proceedings of the International Conference on Communications*, pp. 54–59, 1995.

[15] D. Divsalar and R. J. McEliece, "Effective free distance of turbocodes," *Electronics Letters*, vol. 32, pp. 445–446, February 1996.