Sparse Graph Codes

David J. C. MacKay (mackay@mrao.cam.ac.uk)
Cavendish Laboratory, Cambridge, CB3 0HE, United Kingdom.

Abstract

In the last decade remarkable progress has been made towards the Shannon limit, using codes that are defined in terms of sparse random graphs, and which are decoded by message passing.

Designing a good 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 codes, first pioneered by Gallager (1962), has recently been rediscovered and generalized. The practical performance of Gallager's codes and their modern cousins is vastly better than the performance of the codes with which textbooks have been filled in the intervening years.

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 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 only involves a small number of variables in the graph: the number of edges in the graph scales roughly linearly with N.

The graph defining a **low density parity check code** (Gallager code) contains two types of node: codeword bits, and parity constraints. In a regular (j, k) Gallager code, 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 (Divsalar *et al.* 1998) can be represented by a graph with four types of node: 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 $\boxed{+}$ constraints cause the transmitted stream, working from left to right, to be the accumulated sum (modulo 2) of the permuted intermediate bits.

In a **turbo code** (Berrou and Glavieux 1996), the K source bits drive two linear feedback shift registers, which emit parity bits.

All these codes can be decoded by a local message–passing algorithm on the graph, the sum–product algorithm (MacKay and Neal 1996; McEliece *et al.* 1998; MacKay 1999), and, while this algorithm is not the optimal decoder, the empirical results are record–breaking.

Which of the three types of sparse graph code is 'best' depends on the chosen rate and blocklength, the permitted encoding and decoding complexity, and the question of whether occasional undetected errors are acceptable (turbo codes and repeat–accumulate codes both typically make occasional undetected errors because they have a small number of low weight codewords; Gallager codes do not typically show such an error floor). Gallager codes are the most versatile; it's easy to make a competitive Gallager code with almost any rate and blocklength.

The best known binary Gallager codes are *irregular* codes whose parity check matrices have *nonuniform* weight per column (Luby *et al.* 1998; Urbanke *et al.* 1999). The carefully constructed codes of Urbanke, Richardson and Shokrollahi outperform turbo codes at long blocklengths. Turbo codes can also be beaten by irregular Gallager codes defined over finite fields $GF(q)$ (Davey and MacKay 1998). While there is a good theory for Gallager code design (Urbanke *et al.* 1999), there is no comparable theory for the construction of irregular graphs with state variables. Given the ease with which simple repeat–accumulate codes achieve good results, it seems plausible that the best codes should make use of state variables.

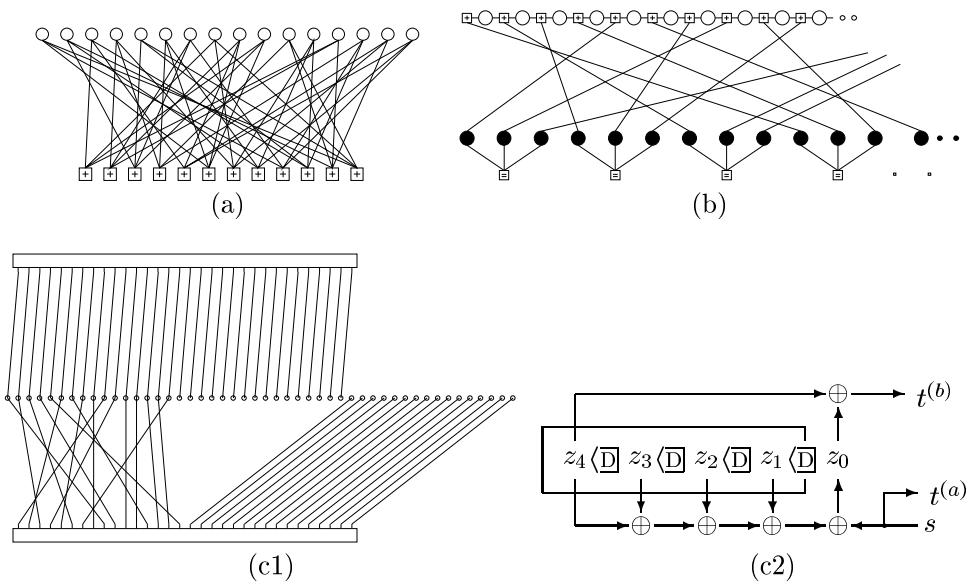

Figure 1: Graphs of three sparse graph codes.
(a) A rate 1/4 low density parity check code (Gallager code) with blocklength $N = 16$, and $M = 12$ constraints. Each white circle represents a transmitted bit. Each bit participates in $j = 3$ constraints, represented by $\boxed{+}$ squares. Each $\boxed{+}$ 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, with the systematic bits occupying the left half of the rectangle and the parity bits occupying the right half. (c2) Each trellis is generated by a $(21/37)_8$ recursive filter which has a state space of 4 bits.

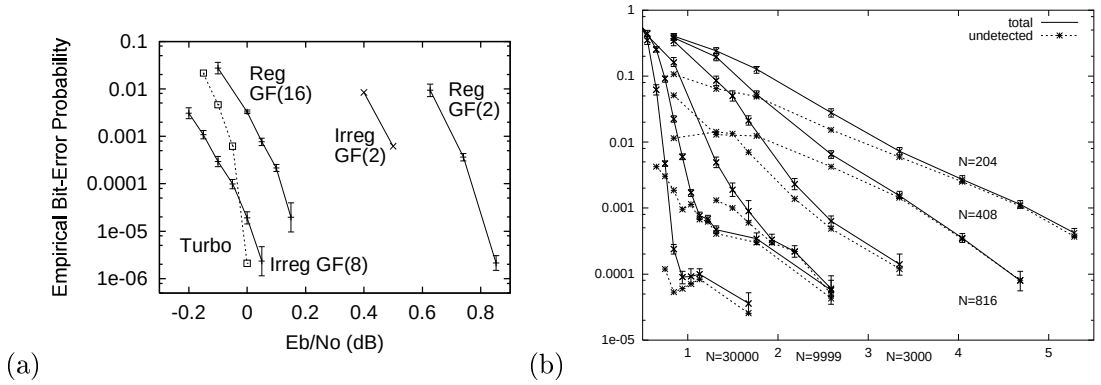

Figure 2: (a) Empirical results for Gaussian Channel, Rate 1/4 Left–Right : Irregular LDPC, $GF(8)$ blocklength 24000 bits; JPL turbo, blocklength 65536 bits; Regular LDPC, $GF(16)$, blocklength 24448 bits; Irregular LDPC , $GF(2)$, blocklength 64000 bits; Regular LDPC, $GF(2)$, blocklength 40000 bits. (Reproduced from (Davey and MacKay 1998).)
(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.

DIFFERENCE SET CYCLIC CODES

N	7	21	73	273	1057	4161
M	4	**10**	**28**	**82**	**244**	**730**
K	3	**11**	45	191	813	3431
d	4	**6**	**10**	**18**	34	66
k	3	**5**	**9**	17	33	65

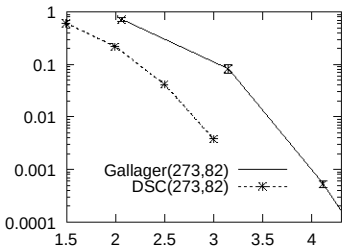

Figure 3: Low density parity check codes satisfying many redundant constraints outperform equivalent Gallager codes. Data on DSC code performance courtesy of R. Lucas and M. Fossorier. The table shows the N, M, K, distance d, and row weight k of some difference set cyclic codes, highlighting the codes that have large d/N, small k, and large N/M. In the comparison the Gallager code had $(j, k) = (4, 13)$, and rate identical to the DSC code.

The performance of Gallager codes can be enhanced by designing the code to have **redundant sparse constraints** (Tanner 1981) (R. Lucas and M. Fossorier, personal communication). There is a difference–set cyclic code, for example, that has $N = 273$, and $K = 191$, but the code satisfies not $M = 82$ but N, *i.e.*, *273* low-weight constraints (figure 3). It is impossible to make random Gallager codes that have anywhere near this much redundancy among their checks.

An open problem is to discover codes sharing the remarkable properties of the difference–set cyclic codes but with different blocklengths and rates. I call this task **the Tanner challenge**.

References

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

Davey, M. C., and MacKay, D. J. C. (1998) Low density parity check codes over GF(q). In *Proceedings of the 1998 IEEE Information Theory Workshop*. IEEE.

Divsalar, D., Jin, H., and McEliece, R. J., (1998) Coding theorems for 'turbo like' codes.

Gallager, R. G. (1962) Low density parity check codes. *IRE Trans. Info. Theory* **IT-8**: 21–28.

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

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

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

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

Tanner, R. M. (1981) A recursive approach to low complexity codes. *IEEE Transactions on Information Theory* **27** (5): 533 547.

Urbanke, R., Richardson, T., and Shokrollahi, A., (1999) Design of provably good low density parity check codes.