Optimizing Sparse Graph Codes over $GF(q)$

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

August 13, 2003

Indexing terms: Error-correction codes, low-density parity-check codes.

Abstract

This note explains the method used by Davey and MacKay to set the non-zero entries in low-density parity-check codes over $GF(q)$, and gives explicit prescriptions.

Introduction

The outstanding codes described in (Hu *et al.*, 2003) have revived interest in low-density parity-check codes over $GF(q)$ for $q > 2$, and demonstrated that as q is increased, steadily improved performance can be achieved on a given fixed channel, as long as the progressive edge-growth method is used to make the code.

In our work on codes over $GF(q)$ (Davey and MacKay, 1998a; Davey and MacKay, 1998b; Davey and MacKay, 1997; MacKay and Davey, 2000; Davey, 1999), Matthew Davey and I always used matrices whose non-zero elements were selected not completely at random but from a special distribution that we found empirically gave slightly improved performance. (Perhaps in the ballpark of 0.1dB.)

Our method was simple. We assumed a particular channel model, such as a binary symmetric channel. Then, for each distinct choice of the k non-zero entries in a row of the parity-check matrix, we examined the marginal entropy of one element of the syndrome vector. We searched for choices of the non-zero entries that maximized this entropy. Then the codes we created had their rows drawn randomly from these choices.

The justification for this procedure is also simple: the greater the entropy of the syndrome, the closer to the Shannon limit an optimal decoder can get.

Examples

As an example, here are the optimal entries, by this criterion, for a row of weight $k = 4$ in a code over $GF(16)$. Obviously, any row may be permuted arbitrarily; also all k elements in a row may be multiplied by any non-zero element of $GF(16)$ to obtain an equally good row. For brevity, we omit most of these equivalent choices and show just four.

14	5	3	1
15	5	3	1
14	11	3	1
14	11	8	1

We would choose at random from these entries, these entries multiplied by constants, and their random permutations.

For the case of $k = 5$, $GF(16)$, the following entries are optimal, or within 5 decimal places of optimal.

14 5 3 1 1	15 5 5 3 1	11 9 7 3 1	14 11 6 4 1	13 10 8 6 1
14 11 3 1 1	15 6 5 3 1	12 9 7 3 1	13 12 7 4 1	15 10 8 6 1
14 5 3 2 1	9 7 5 3 1	13 11 7 3 1	14 12 7 4 1	14 11 8 6 1
14 6 5 2 1	13 7 5 3 1	14 11 7 3 1	15 12 7 4 1	11 9 8 7 1
14 10 6 2 1	15 8 5 3 1	15 11 7 3 1	15 12 10 4 1	13 11 8 7 1
15 10 6 2 1	14 9 5 3 1	14 12 7 3 1	14 12 11 4 1	14 12 8 7 1
14 11 6 2 1	13 10 5 3 1	14 11 10 3 1	15 8 6 5 1	14 12 10 8 1
14 5 3 3 1	14 13 5 3 1	14 12 10 3 1	14 11 8 5 1	15 12 10 8 1
15 5 3 3 1	13 10 6 3 1	14 14 11 3 1	15 11 8 5 1	14 11 11 8 1
15 5 4 3 1	14 10 6 3 1	15 14 11 3 1	14 12 8 5 1	14 12 11 8 1
	15 10 6 3 1	15 10 6 4 1	15 12 8 5 1	14 14 11 8 1

For $k = 4$, $GF(64)$, we find the following rows are optimal (to within 5 decimals):

13	11	7	1
28	23	10	1
46	28	10	1
14	13	11	1
59	22	12	1
53	44	24	1
52	40	29	1
58	52	40	1

For $k = 5$, $GF(64)$, we find the following rows are optimal (to within 5 decimals):

				34 30 27 9 1	48 28 23 19 1
31 11 5 3 1	45 30 10 3 1	52 37 31 3 1	25 21 11 7 1	53 27 19 12 1	62 29 26 19 1
62 11 5 3 1	55 30 10 3 1	52 51 31 3 1	50 21 11 7 1	53 44 19 12 1	62 58 26 19 1
25 15 5 3 1	53 46 13 3 1	53 46 39 3 1	43 25 11 7 1	62 27 20 12 1	38 35 28 23 1
29 15 5 3 1	45 21 15 3 1	59 46 39 3 1	58 18 13 7 1	51 31 20 12 1	48 38 28 23 1
45 15 5 3 1	55 21 15 3 1	62 39 27 5 1	58 36 13 7 1	62 44 20 12 1	56 38 35 23 1
55 15 5 3 1	45 42 15 3 1	62 50 27 5 1	39 29 18 7 1	51 46 20 12 1	44 34 30 24 1
58 15 5 3 1	60 45 20 3 1	62 50 44 5 1	60 54 18 7 1	56 46 20 12 1	51 40 31 24 1
62 22 5 3 1	60 55 20 3 1	62 27 10 6 1	60 27 24 7 1	44 34 30 12 1	43 36 34 25 1
62 27 5 3 1	55 38 21 3 1	62 44 10 6 1	44 30 24 7 1	62 40 29 13 1	62 40 29 26 1
62 44 5 3 1	59 39 23 3 1	45 42 15 6 1	60 44 24 7 1	62 55 26 17 1	62 55 41 26 1
62 43 9 3 1	51 46 26 3 1	53 44 19 6 1	43 36 25 7 1	62 52 45 17 1	50 44 35 31 1
30 29 10 3 1	56 46 26 3 1	59 44 40 6 1	60 47 36 7 1	39 34 29 18 1	62 52 45 34 1
				54 34 30 18 1	56 46 38 35 1
					62 50 44 35 1

Commands used

The perl program that generates these results is available in `http://www.inference.phy.cam.ac.uk/macka`
The program is not particularly efficient. It uses fourier transforms to compute the syndrome prob-

ability, and walks through all possible rows, subject to the constraint that the final entry is 1, and avoiding repetition of any permutations.

The program was used as follows to perform exhaustive searches (rather excessive, but it does allow you to quantify how much better the best rows are).

```
hzFFTe.p tr=3 q=16 exhaustive=1 > tmpe16.3
hzFFTe.p tr=4 q=16 exhaustive=1 > tmpe16.4
hzFFTe.p tr=5 q=16 exhaustive=1 > tmpe16.5
hzFFTe.p tr=6 q=16 exhaustive=1 > tmpe16.6
hzFFTe.p tr=7 q=16 exhaustive=1 > tmpe16.7

hzFFTe.p tr=3 q=64 exhaustive=1 > tmpe64.3
hzFFTe.p tr=4 q=64 exhaustive=1 > tmpe64.4
hzFFTe.p tr=5 q=64 exhaustive=1 > tmpe64.5

tar cvf ~/pub/code/perl/hz.tar tmpe*
```

The default channel is binary symmetric with flip probability 0.1. For historical reasons, the program reports the syndrome entropy on a scale from 0 to 2, *i.e.*, it is the entropy per two bits.

I will make a tar file with all the output files in it and send it to you.

Acknowledgements

This work was supported by a partnership award from IBM Zürich research labs.

References

Davey, M. C. (1999) *Error-correction using Low-Density Parity-Check Codes*. University of Cambridge dissertation.

Davey, M. C., and MacKay, D. J. C., (1997) Monte Carlo simulations of infinite low density parity check codes over $GF(q)$. Available from `http://www.inference.phy.cam.ac.uk/is/papers/`.

Davey, M. C., and MacKay, D. J. C. (1998a) Low density parity check codes over GF(q). In *Proceedings of the 1998 IEEE Info. Theory Workshop*, pp. 70–71. IEEE.

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

Hu, X.-Y., Eleftheriou, E., and Arnold, D.-M., (2003) Regular and irregular progressive edge-growth Tanner graphs. Submitted to IEEE TRANS. on INFORMATION THEORY.

MacKay, D. J. C., and Davey, M. C. (2000) Evaluation of Gallager codes for short block length and high rate applications. In *Codes, Systems and Graphical Models*, ed. by B. Marcus and J. Rosenthal, volume 123 of *IMA Volumes in Mathematics and its Applications*, pp. 113–130. Springer.

Version 1.2