Low Density Parity Check Codes over $GF(q)$

Matthew C. Davey, David J. C. MacKay

Abstract—**Gallager's low density binary parity check codes have been shown to have near Shannon limit performance when decoded using a probabilistic decoding algorithm. We report the empirical results of error-correction using the analogous codes over $GF(q)$ for $q > 2$, with Binary Symmetric channels and Binary Gaussian Channels. We find a significant improvement over the performance of the binary codes, including a rate 1/4 code with bit error probability $< 10^{-5}$ at $E_b/N_0 = 0.2$dB.**

I. Introduction

CODES defined in terms of a non-systematic low density parity check matrix [1], [2] are asymptotically good, and can be practically decoded with Gallager's belief propagation algorithm [3], [4], [5]. Our proof in [5] shows that they are asymptotically good codes for a wide class of channels, not just for the memoryless binary symmetric channel. Results presented in [4] showed these codes (which we call 'LDPC' codes) have near Shannon limit performance when decoded using the belief propagation algorithm.

Binary LDPC codes may be generalised to finite fields $GF(q)$ in a natural way. In the remainder of this paper we use a vector space over the finite field $GF(q)$ where $q = 2^b$. Elements of $GF(q)$ will be called *symbols* and we use the term *bits* when referring to the binary representation of symbols.

Definition 1: The ***weight*** of a vector or matrix is the number of non-zero symbols in it. The ***density*** of a source of random symbols is the expected fraction of non-zero symbols. The ***overlap*** between two vectors is the number of coordinates in which both vectors have non-zero entries.

II. Construction

The code is defined in terms of a very sparse random parity check matrix $\mathbf{H}$. A transmitted block length N and a source block length K are selected. We define $M = N - K$ to be the number of parity checks. We select a *mean column weight* t, which is a number greater than 2. We create a rectangular $M \times N$ matrix [M rows and N columns] $\mathbf{H}$ at random having mean weight t per column with the weight of each column at least 2. The weight per row is made as uniform as possible with the overlap between any two columns being either zero or one. The non-zero elements of $\mathbf{H}$ are selected from a carefully selected random distribution[6]; rather than using the uniform distribution we choose the entries in each row to maximise the entropy of the corresponding bit of the syndrome vector $\mathbf{z} = \mathbf{Hx}$ where $\mathbf{x}$ is a sample from the assumed channel noise model.

Matthew Davey and David MacKay are with the Cavendish Laboratory, Cambridge, United Kingdom. Email: `mcdavey@mrao.cam.ac.uk, mackay@mrao.cam.ac.uk`

To reduce the probability of introducing low weight codewords the weight 2 columns are constructed systematically. To generate codewords, we would use Gaussian elimination to derive the generator matrix.

There is a possibility that the rows of $\mathbf{H}$ are not independent (though for odd t, this has small probability); in this case $\mathbf{H}$ is a parity check matrix for a code with the same N and with smaller M. So $\mathbf{H}$ defines a code with rate of *at least* K/N. Results are quoted here based on the assumption that the rate is equal to K/N.

III. Channel models

We will use these codes to communicate over binary channels, making no special use of the algebraic structure of $GF(q)$. Moving to $GF(q)$ makes the codes more complex while decoding remains tractable. We have applied our codes to the binary symmetric channel (BSC) and the Binary Gaussian Channel with inputs of $\pm s$ and additive noise of variance $\sigma^2 = 1$. If one communicates using a code of rate R then it is conventional to describe the signal to noise ratio (SNR) by $E_b/N_0 = s^2/2R\sigma^2$ and to report this number in decibels as $10\log_{10} E_b/N_0$. We define the received bit to be the sign of the channel output and set the likelihood of the nth noise bit being 1 to $g_n^1 = 1/(1+\exp(2s\,|y_n|\,/\sigma^2))$ where y_n is the output of the channel. We also define $g_n^0 = 1 - g_n^1$.

In the case of the BSC g_n^1 is independent of n.

In $GF(2^b)$ each noise symbol x_n consists of b noise bits $x_{n_1} \ldots x_{n_b}$. Our channel models are memoryless binary channels so we can set the likelihood of the noise symbol x_n being equal to a to $f_n^a := \prod_{i=1}^{b} g_{n_i}^{a_i}$ for each $a \in GF(q)$ where a_i is the ith bit of the binary representation of a.

IV. Decoding

The decoding problem is to find the most probable vector $\mathbf{x}$ such that $\mathbf{Hx} = \mathbf{z}$, with the likelihood of $\mathbf{x}$ determined by the channel model. The decoding algorithm we use is a generalisation of the approximate belief propagation algorithm [7] used by Gallager [1] and MacKay and Neal [3], [4], [5]. The complexity of decoding scales as Ntq^2 per iteration.

We will refer to elements of $\mathbf{x}$ as *noise symbols* and elements of $\mathbf{z}$ as *checks*. Let $\mathcal{N}(m) := \{n : H_{mn} \neq 0\}$ be the set of noise symbols that participate in check m. Let $\mathcal{M}(n) := \{m : H_{mn} \neq 0\}$ be the set of checks that depend on noise symbol n.

With each non-zero entry in the parity check matrix H_{mn} we associate quantities q_{mn}^a, r_{mn}^a for $a \in GF(q)$. The quantity q_{mn}^a is meant to be the probability that symbol n of $\mathbf{x}$ is a, given the information obtained via checks other than check m. The quantity r_{mn}^a is meant to be the probability of check m being satisfied if symbol n of $\mathbf{x}$ is considered fixed at a and the other noise symbols have a separable distribution given by the probabilities $\{q^a_{mn'} : n' \in \mathcal{N}(m)\backslash n, a \in GF(q)\}$.

A. Initialisation

We initialise the values of q^a_{mn} to f^a_n, the likelihood that $x_n = a$ according to the channel model as outlined in section III.

B. Updating r^a_{mn}

We compute the new value of r^a_{mn}:

$$r^a_{mn} = \sum_{\mathbf{x}':x'_n=a} \mathrm{Prob}\left[z_m|\mathbf{x}'\right] \prod_{j\in\mathcal{N}(m)\backslash n} q^{x'_j}_{mj} \tag{1}$$

where $\mathrm{Prob}\left[z_m|\mathbf{x}'\right] \in \{0,1\}$ according to whether or not $\mathbf{x}'$ satisfies check m.

We can calculate r^a_{mn} efficiently by defining the partial sums $\sigma_{mk} := \sum_{j:j\leq k} H_{mj}x'_j$ and $\rho_{mk} := \sum_{j:j\geq k} H_{mj}x'_j$ and calculating $\mathrm{Prob}\left[\sigma_{mk} = a\right]$ for each $a \in GF(q)$ and each $k \in \mathcal{N}(m)$ according to the probabilities given by the q. If i, j are successive indices in $\mathcal{N}(m)$ with $j > i$ then

$$\mathrm{Prob}\left[\sigma_{mj} = a\right] = \sum_{\{s,t:H_{mj}t + s = a\}} \mathrm{Prob}\left[\sigma_{mi} = s\right] q^t_{mj} \tag{2}$$

Similarly we can calculate the distribution of each ρ_{mk}. Now we can update the values of r^a_{mn} using

$$\begin{aligned} r^a_{mn} &= \mathrm{Prob}\left[(\sigma_{m(n-1)} + \rho_{m(n+1)}) = z_m - H_{mn}a\right] \quad (3)\\ &= \sum_{\{s,t:s+t = z_m - H_{mn}a\}} \mathrm{Prob}\left[\sigma_{m(n-1)} = s\right] \mathrm{Prob}\left[\rho_{m(n+1)} = t\right] \quad (4) \end{aligned}$$

C. Updating q^a_{mn}

For each m and n and for $a \in GF(q)$ we update:

$$q^a_{mn} = \alpha_{mn}\, f^a_n \prod_{j\in\mathcal{M}(n)\backslash m} r^a_{jn} \tag{5}$$

where α_{mn} is chosen such that $\sum_{a=1}^{q} q^a_{mn} = 1$.

We then make a tentative decoding $\hat{\mathbf{x}}$:

$$\hat{x}_n = \operatorname*{argmax}_a f^a_n \prod_{j\in\mathcal{M}(n)} r^a_{jn} \tag{6}$$

If $\mathbf{H}\hat{\mathbf{x}} = \mathbf{z}$ then the decoding algorithm halts having identified a valid decoding of the syndrome, otherwise the algorithm repeats. A failure is declared if some maximum number of iterations (*e.g.* 500) occurs without a valid decoding.

V. Results

We compare codes of blocklength N symbols over $GF(2^b)$ with binary codes of length Nb, *i.e.* codes of length 1000 over $GF(8)$ with binary codes of length 3000.

Figure 1 compares the performance of LDPC codes with a fixed column weight of 3 over a BSC using codes over fields $GF(2), GF(4)$, and $GF(8)$. The vertical axis shows the empirical bit-error probability. It should be pointed out that all the errors were *detected* errors: the decoder reported that it had failed. The BSC results show a monotonic improvement as we increase the order of the fields on which the codes are based.

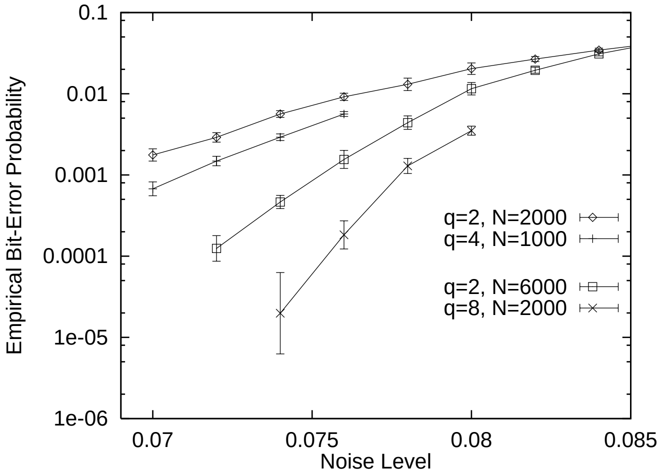

Fig. 1. Comparison of performance of LDPC codes for a BSC over $GF(2), GF(4)$, and $GF(8)$. These codes had column weight 3 and rate 1/2. From top to bottom: (N, q) = (2000,2); (1000,4); (6000,2); (2000,8).

Figure 2 shows the performance of LDPC codes over a Binary Gaussian Channel. Three codes of rate 1/3 with average column weight 2.5 are shown, over fields $GF(2), GF(4)$, and $GF(8)$. Also shown is a code of rate 0.26 with column weight 2.3 over $GF(16)$. We compare the performance of these LDPC codes with recent Turbo code results [8].

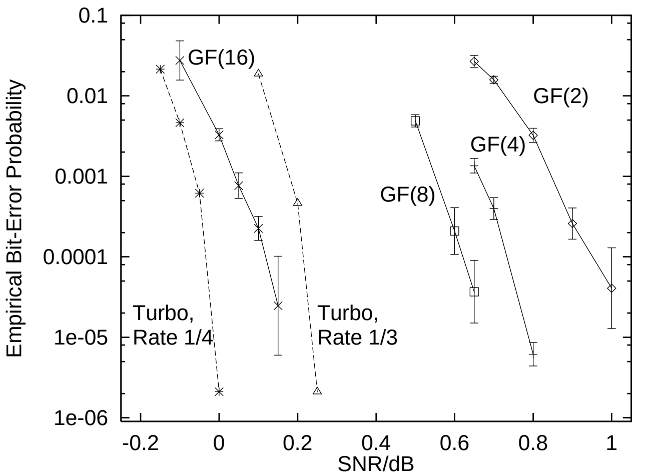

Fig. 2. Comparison of performance of LDPC codes over a Binary Gaussian Channel over $GF(2), GF(4)$, and $GF(8)$. From left to right: JPL Turbo code, rate 1/4, blocklength 65536; LDPC Code with rate 0.26, average column weight 2.3, (N, q) = (6000,16); JPL Turbo code, rate 1/3, blocklength 49152; LDPC codes with rate 1/3, average column weight 2.5: (N, q) = (6000,8), (9000,4), (18000,2).

The Binary Gaussian Channel results show that LDPC codes over higher order fields can significantly outperform the equivalent binary codes. An improvement of 0.3dB is shown for the rate 1/3 code moving from binary to $GF(8)$ construction. Our Monte Carlo simulations of infinite LDPC codes [9] motivated the construction of a rate 0.26 code on $GF(16)$ with average column weight 2.3, whose performance approaches that of the best known Turbo codes.

It is worth noting that investigations of higher weight codes over the Binary Gaussian Channel suggest that there is not always a monotonic improvement with increased field order. For example with column weight 3, rate 1/2 codes, the $GF(4)$ codes outperformed $GF(8)$ codes. It remains an interesting topic for further research to understand the observed dependencies of the decoding algorithm on the parameters of the code.

This work confirms that the near Shannon limit performance of Gallager's binary low density parity check codes can be significantly enhanced by a move to fields of higher order.

References

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

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

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

[4] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes", *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, August 1996, Reprinted *Electronics Letters*, vol 33, no 6, 13th March 1997, p.457–458.

[5] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices", submitted to IEEE transactions on Information Theory. Available from `http://wol.ra.phy.cam.ac.uk/`, 1997.

[6] M. C. Davey and D. J. C. MacKay, "Good codes over $GF(q)$ based on very sparse matrices", In preparation, 1997.

[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, 1988.

[8] JPL, "Turbo codes performance", Available from `http://www331.jpl.nasa.gov/public/TurboPerf.html`, August 1996.

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