Monte Carlo simulations of infinite low density parity check codes over GF(q)

Matthew C. Davey David J. C. MacKay*

March 30, 1998

Abstract

Binary Low Density Parity Check (LDPC) codes have been shown to have near Shannon limit performance when decoded using a probabilistic decoding algorithm. The analogous codes defined over finite fields $GF(q)$ of order $q > 2$ show significantly improved performance. We present the results of Monte Carlo simulations of the decoding of infinite LDPC Codes which can be used to obtain good constructions for finite Codes. Our empirical results for the Gaussian channel include a rate 1/4 code with bit error probability of 10^{-4} at $E_b/N_0 = -0.05$dB.

1 Introduction

We consider a class of error correcting codes first described by Gallager in 1962 [1]. These recently rediscovered low density parity check (LDPC) codes are defined in terms of a sparse parity check matrix and are known to be asymptotically good for all channels with symmetric stationary ergodic noise [2, 3]. Practical decoding of these codes is possible using an approximate belief propagation algorithm and near Shannon limit performance has been reported [4].

We consider the generalisation of binary LDPC codes to finite fields $GF(q)$, $q > 2$, and demonstrate a significant improvement in empirical performance.

Little is known about the theoretical properties of the approximate belief propagation algorithm when applied to this decoding problem. However, we can use Monte Carlo methods to simulate the behaviour of the decoding algorithm applied to an infinite LDPC code in an effort to investigate the effects of changes in field order, code construction and noise level on the decoding performance. These Monte Carlo results have been used to design better codes for practical decoding.

In section 2 we define low density parity check codes and in section 4 we describe the decoding algorithm. Section 5 presents the results of the Monte Carlo simulation and empirical decoding performance is presented in section 5.1.

2 Code construction

The codes are defined in terms of a low density parity check matrix $\mathbf{H}$ as follows. We choose a source block length K, a transmitted block length N and a mean column weight $t > 2$. The *weight* of a vector is the number of non-zero components in that vector. We

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

define $M = (N - K)$ to be the number of parity checks in the code. $\mathbf{H}$ is a rectangular matrix with M rows and N columns. We construct $\mathbf{H}$ such that the weight of each column is at least 2, the mean column weight is t and the weight per row is as uniform as possible.

We fill the non-zero entries in $\mathbf{H}$ from the elements of a finite field $GF(q)$, $q = 2^b$, according to a carefully selected random distribution[5]: 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. Although the code construction is largely random, we may reduce the probability of introducing low weight codewords by constructing the weight 2 columns systematically. To generate codewords we would derive the generator matrix using Gaussian elimination.

If the rows of $\mathbf{H}$ are not independent (for odd t, this has small probability) $\mathbf{H}$ 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.

3 Channel models

We consider these codes over 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.

4 Decoding algorithm

We transmit a vector $\mathbf{t}$ which is received as $\mathbf{r} = \mathbf{t} + \mathbf{n}$ where $\mathbf{n}$ is a sample from the channel noise distribution. An instance of the decoding problem requires finding the most probable vector $\mathbf{x}$ such that $\mathbf{Hx} = \mathbf{z}$, where $\mathbf{z}$ is the syndrome vector $\mathbf{z} := \mathbf{Hr} = \mathbf{Hn}$ and the likelihood of $\mathbf{x}$ is determined by the channel model. The decoding algorithm we use is a generalisation of the approximate belief propagation algorithm [6] used by Gallager [1] and MacKay and Neal [3, 4, 2]. 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 $\{Q_{mn'}^a : n' \in \mathcal{N}(m) \backslash n, a \in GF(q)\}$.

We initialise the values of Q_{mn}^a to f_n^a, the likelihood that $x_n = a$ according to the channel model.

We compute the new value of R_{mn}^a:

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

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

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

$$Q^a_{mn} = \alpha_{mn}\, f^a_n \prod_{j \in \mathcal{M}(n)\backslash m} R^a_{jn} \tag{2}$$

where α_{mn} is chosen such that $\sum_{a=0}^{q-1} Q^a_{mn} = 1$.

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

$$\hat{x}_n = \operatorname*{argmax}_a f^a_n \prod_{j \in \mathcal{M}(n)} R^a_{jn} \tag{3}$$

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.

5 Monte Carlo simulation

The belief propagation algorithm may be viewed as a message passing algorithm on a directed bipartite graph where nodes are associated with checks and noise symbols. Noise symbol n is connected to check m if the corresponding matrix entry H_{mn} is non-zero. The edges of the graph are directed from noise vertices (parents) to check vertices (children), indicating the causal relationship. In this view, the quantities Q are passed from parents to children, and the quantities R are passed from children to parents.

In the absence of undirected loops in the graph, the messages Q are known to converge to the correct posterior probabilities after a number of iterations equal to the width of the graph. In practice, our LDPC codes will have many loops and we do not expect to calculate the true posterior nor (in general) to get convergence.

We can use Monte Carlo methods to simulate an infinite LDPC code in order to investigate the properties of the decoding algorithm in the absence of loops. We would expect the behaviour to be similar to the empirical performance as the blocklength of the codes increased. We form an ensemble of noise symbols $\mathcal{N} := \{n_i, \{Q^a_i\}_{a \in GF(q)}\}_1^N$ emitting initial messages Q^a_i according to our channel model. We then form a fragment of an infinite graph using this ensemble, reflecting our matrix construction, and propagate the Q messages down to a new noise symbol to produce an element of a new ensemble $\mathcal{N}'$. This ensemble $\mathcal{N}'$ will represent an ensemble of noise symbols after one iteration of the decoding algorithm. We iterate this procedure and declare a decoding 'success' if the average entropy of the Q messages in our ensemble drops below some chosen threshold.

With this approach it is possible to investigate the effect of changes in field order, code construction and noise level on the decoding performance. This proves useful in the construction of finite LDPC codes.

5.1 MC Results

Using the Monte Carlo method outlined above we can examine the range of parameters for which the entropy S becomes arbitrarily small. Of interest to us will be the dependence of S on the field order q and the mean column weight t for a given channel. We present results for the Binary Symmetric Channel and the Binary Gaussian Channel.

Although the Monte Carlo analysis is an approximate simulation of infinte LDPC codes, we find that the behaviour is very close to that of finite codes, at least in certain regimes. In figure 1 we plot average entropy per bit as a function of the number of iterations of the decoder. We compare the output of the Monte Carlo simulation with the average of 100 runs of the corresponding finite LDPC code. We find that for higher

weight codes the behaviour is almost identical. As the weight decreases we expect more short cycles to appear in the belief network of the finite code introducing feedback in the decoding algorithm. This can have the effect of making entropy decrease faster than in the infinite case.

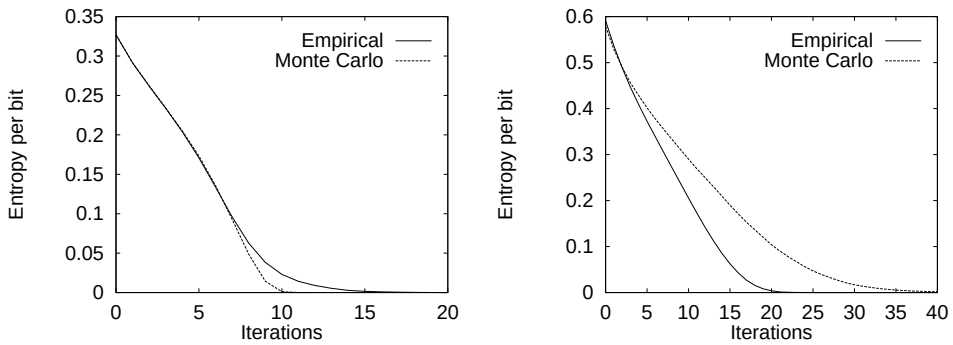

Figure 1: Comparison of Monte Carlo and empirical results. (Left) Codes over $GF(8)$ with mean column weight 3, Gaussian channel, SNR 1.7dB. (Right) Codes over $GF(16)$ with mean column weight 2.3, Gaussian Channel, SNR 0.75dB

5.2 Binary Symmetric Channel

In figure 2 we compare the effectiveness of the decoding for a range of mean column weights, for a rate 1/4 code. The horizontal axis shows the mean column weight and the vertical axis shows the maximum noise level for which the average entropy reached 10^{-5} per bit in less than 80 iterations.

The results suggest that for best performance we should choose the highest order code that is feasible, bearing in mind that the decoding time per iteration scales as q^2. We then choose an appropriate mean column weight, which will be lower as the order increases.

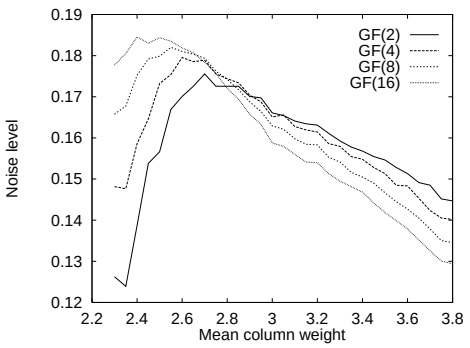

Figure 2: Binary Symmetric Channel: Maximum Noise level for which average bit entropy drops below 10^{-5} within 80 iterations, as a function of the column weight. Rate 1/4 code.

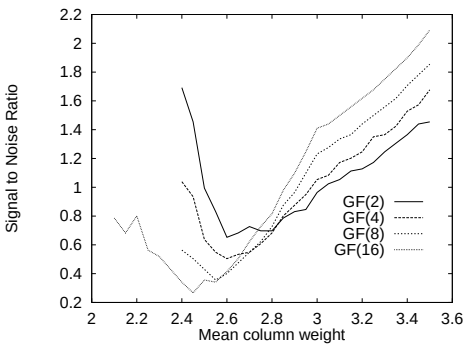

Figure 3: Gaussian Channel: Minimum signal to noise ratio for which average bit entropy reaches 10^{-5} within 80 iterations, as a function of mean column weight. Rate 1/4 code.

5.3 Gaussian Channel

Figure 3 shows similar results for the Gaussian channel. We plot the minimum signal to noise ratio for which the average bit entropy reaches 10^{-5} within 80 iterations as a function of the mean column weight. Again, we see that the best performance is obtained for a code defined over a high order, for an appropriate mean weight.

6 Conclusions

For the channel models we have studied, we find that for a given communication rate, field order and channel there is an optimal mean column weight t^* for which the entropy decreases at a maximum rate. Furthermore we find that for a given rate and channel t^* becomes smaller as the field order increases. Lower weight codes are easier to decode, suggesting improved performance for codes defined over higher order fields.

The Monte Carlo procedure simulates ideal infinite LDPC codes but the results are very close to those for actual finite codes, especially for higher weight ($t > 3$) codes. It is interesting that lower weight codes can show performance better than their infinite counterparts (figure 1). We have used the approximate value of t^* calculated in the MC analysis to construct finite codes with performance very close to the Shannon limit [7]. We have produced rate 1/4 codes with mean column weight 2.3 with bit error probability of 10^{-5} at $E_b/N_0 = 0.2$dB. Using LDPC codes with more complex constructions [8] we have achieved a bit error of 10^{-4} at $E_b/N_0 = -0.05$dB, representing a slight improvement over the best turbo codes this author is aware of. It remains an interesting topic for further research to develop systematic constructions for ultra low weight LDPC codes that can be effectively decoded.

References

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

[2] 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.

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

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

[7] M. C. Davey and D. J. C. MacKay, "Low density parity check codes over GF(q)", Submitted to IEEE Electronics Letters, available from `http://wol.ra.phy.cam.ac.uk/is/papers/`, 1997.

[8] M. G. Luby M. Mitzenmacher M. Amin Shokrollahi D. A. Spielman V. Stemann, "Practical loss-resilient codes", To appear in "The Twenty-Ninth Annual ACM Symposium on Theory of Computing (STOC)".