

# Sparse Low-Density Parity-Check Codes for Channels with Cross-Talk

Edward A. Ratzler and David J.C. MacKay  
 Cavendish Laboratory,  
 Madingley Road, Cambridge CB3 0HE  
 {ear23,mackay}@mrao.cam.ac.uk

**Abstract** — Low-density parity-check codes are modified to produce transmissions in which the symbols 1 and 0 are used with different frequencies. These codes are good candidates for multi-user channels with crosstalk, such as optical channels.

## I. INTRODUCTION

For some noisy channels the noise level depends on the transmitted signal, and to achieve the maximum rate of communication, the symbols should not be used with equal probability. For a single user binary channel the maximum gain in capacity from using an asymmetric distribution is 6% [8], but for a multiuser channel larger gains can be achieved.

In this paper we will look at two simple channel models:

**Parallel Z Channel** Each user's channel is a Z-channel where the probability of corrupting 0s is proportional to the number of 1s being transmitted on the other channels, figure 1(a). This is a model of crosstalk in optical channels.

**Parallel Binary Symmetric Channel (BSC)** Each user's channel is a BSC where the flip probability is proportional to the number of 1s being transmitted on the other channels, figure 1(b). This example is similar to noise on some code-division multiple access (CDMA) channels.

For both channels,  $a$  is used as the constant of proportionality,  $o$  as the number of ones being transmitted on the other channels and  $u$  as the number of users.

We will assume the constraint that we treat each subchannel independently and identically. This constraint leads to each subchannel being a channel with flip probability  $= ap_1(u - 1)$ , where  $p_1$  is the probability of transmitting 1. In general, for the parallel BSC,  $p_1 \leq \frac{1}{2}$  for optimal communication. For the parallel Z channel this same condition will apply for high noise levels.

Some numerical capacity calculations for these channels and a coding solution involving sparse data bits but dense parity bits (hence called Sparse-Dense codes) were presented in [7]. In this paper we extend the work by creating codes where the data bits are dense and all the transmitted bits are sparse.

We started by looking at binary low-density parity-check (LDPC) codes with a mapper similar to McEliece [5]. This mapper took a set of equiprobable bits from a binary LDPC encoder and converted them to an output bit with a biased distribution. Simulations showed that codes constructed in this way did not generally perform well. We decided to investigate an alternative scheme using LDPC codes defined over  $GF(q)$  [1, 2].

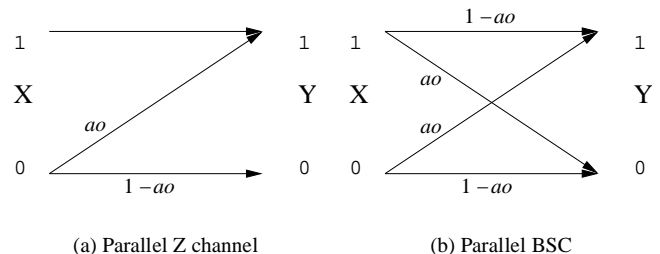


Figure 1: The two channel models studied.  $o$  is the number of ones being transmitted on the other channels at the current time and  $a$  is the noise level parameter.

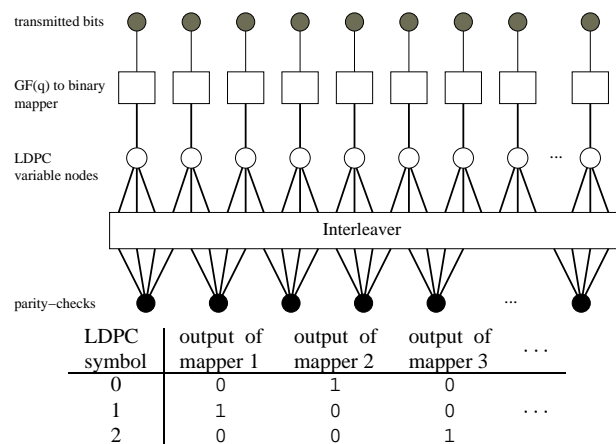


Figure 2: A Sparse LDPC Code. The thick lines represent constraints over  $GF(q)$ , the thin lines represent binary constraints. An example mapper configuration is shown over  $GF(3)$  which will give  $p_1 = 1/3$ .

## II. SPARSE LDPC CODES

LDPC codes over larger finite fields are defined in a similar manner to binary LDPC codes by a sparse parity-check matrix,  $\mathbf{H}$ , but with the symbols chosen from  $GF(q)$ . Techniques from binary LDPC code construction were used to arrange the non-zero entries of the parity-check matrix [4]. The values of the non-zero elements were chosen uniformly at random. To decode a received signal, belief propagation was carried out in a similar manner to binary LDPC codes [6].

### A. Mapper

A time-varying function was used to translate individual code symbols to channel bits, as illustrated in figure 2. To obtain sparse transmissions, most of the symbols in  $GF(q)$  are mapped to 0. The fraction of symbols mapped to 1 is  $p_1$ . We

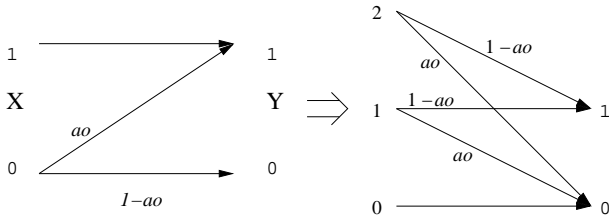


Figure 3: A noise model over  $\text{GF}(3)$  corresponding to the parallel Z channel. The mapper  $\{0\} \rightarrow 1, \{1, 2\} \rightarrow 0$  is used. If applied to a  $\text{GF}(3)$  LDPC code the output bits would have  $p_1 = 1/3$ .

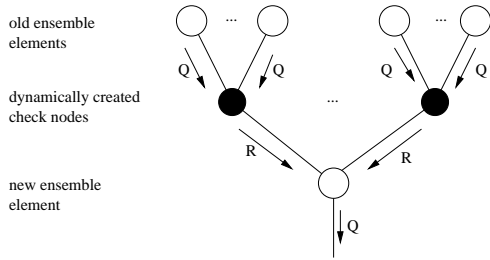


Figure 4: An example of the dynamically created tree above an element of the new ensemble. The downward messages from elements of the old ensemble are connected via dynamically created check nodes to the new ensemble element. In general all the  $Q$  and  $R$  messages illustrated are different.

hope that, with a function chosen at random for each symbol node, all the codewords of the code are mapped to distinct sparse sequences.

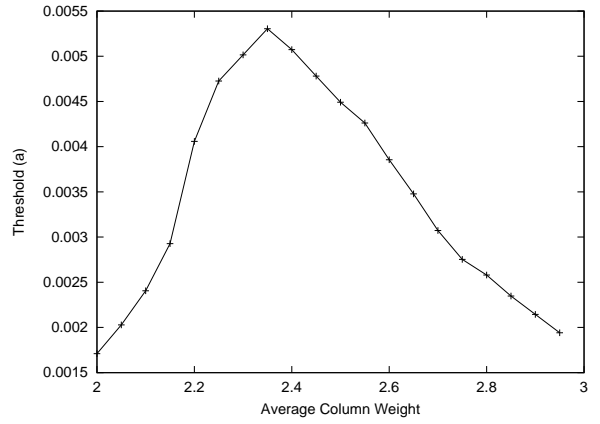
The decoder can view the binary channel and mapper as a single noisy channel, as illustrated in figure 3.

### B. Degree sequence optimization

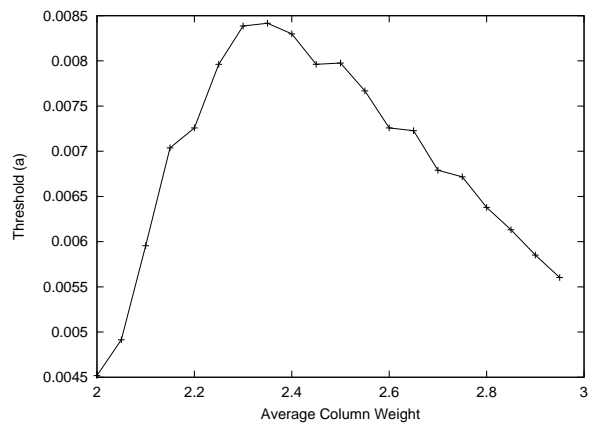
Binary LDPC code constructions with a fixed column weight, called regular codes, are popular. It is known that a column weight of 3 performs well on many standard channels. For LDPC codes over larger finite fields a column weight of three does not lead to optimal code performance. A simple mixture of weight 2 and weight 3 columns can perform a lot better [1].

To optimize the profile of the irregular code, we used the Monte Carlo approach of [1], modified to include the non-linear channel. We started with an ensemble of variable nodes that represented the state of variable nodes of an infinite loop-free code in a particular decoding iteration. We then created a new ensemble of variable nodes from the former ensemble to represent the state one iteration later. This process was repeated for a fixed number of iterations to see whether the decoding converges. By binary division, we found a noise threshold below which decoding converges and above which it does not.

In detail, we associated each variable node with a “correct”  $\text{GF}(q)$  symbol, and a vector,  $Q$ , of messages from it. As the infinite code had a tree structure each symbol node had one message coming out of it. To create an element of the new ensemble we first chose a “correct” symbol from  $\text{GF}(q)$  and used the channel noise model to simulate the received signal.



(a) Parallel BSC,  $\text{GF}(5)$ ,  $R = 0.2$ ,  $p_1 = 1/5$



(b) Parallel Z channel,  $\text{GF}(3)$ ,  $R = 0.3$ ,  $p_1 = 1/3$

Figure 5: Threshold values of  $a$  for semi-regular Sparse LDPC codes with 128 users assuming a belief propagation decoder.

Then we created a tree fragment above it, for example figure 4, by first choosing a column weight,  $c$ , as a sample from the degree distribution. Then we created  $c - 1$  incoming check nodes. For each of these we chose a row weight,  $r$  by sampling from the row distribution weighted by the row weight. We then connected it to  $r - 1$  incoming symbol nodes from the old ensemble. These nodes were sampled from a column distribution weighted by column weight with the constraint that the check node was satisfied using the correct transmissions. Belief propagation was then carried out down the tree to evaluate  $Q$  for the new node. This process was repeated for each element in the new ensemble.

For each of the two channel models we assumed 128 users and a line search was carried out to find the best combination of weight 2 and weight 3 columns for particular code parameters, figure 5. The best average column weight was 2.35 for the parallel BSC channel for codes defined over  $\text{GF}(5)$  with  $p_1 = 1/5$  and  $R = 0.2$ . Similarly 2.35 was the best average column weight for the parallel Z channel with codes defined over  $\text{GF}(3)$  with  $p_1 = 1/3$  and  $R = 0.3$ .

A search for better fully irregular degree sequences using DIRECT [3] was carried out. Some degree sequences for the Z channel are shown in table 1.

The thresholds found in figure 5 may be compared with that of a simple time-sharing scheme: each user transmits using dense data a certain fraction of the time and is silent otherwise. The Shannon limit was optimized over the fraction for which each transmitter was active. The Shannon limit for time-sharing is  $a = 0.00463$  with each transmitter active for 54% of the time for the above parallel BSC. Similarly the Shannon limit is  $a = 0.00832$  with each transmitter active for 81% for the parallel Z channel case. Both of these Shannon limits are less than the respective Sparse LDPC code thresholds found above.

### C. Code Creation

Randomly arranged weight 2 columns can give rise to low weight codewords. For GF(2), if any cycle can be made of weight 2 columns then a codeword of weight half the cycle length exists [9]. For larger finite fields, with randomly chosen matrix elements, a cycle in columns of weight 2 will cause a low-weight codeword of weight half the cycle length with probability  $\frac{1}{q-1}$ .

To avoid such low-weight codewords, we created the weight 2 columns so they are loop-free. First  $\frac{M}{2}$  non-overlapping weight 2 columns were created, then, if more columns were required, we created up to  $\frac{M}{2} - 1$  further columns that connected those columns in a single chain. We also ensured that weight 2 check nodes were not connected solely to weight 2 variable nodes.

## III. SIMULATION RESULTS

An  $N = 10000$ ,  $R = 0.2$ ,  $p_1 = 1/5$  code was created for testing on the parallel BSC. These parameters were chosen to match simulations in [7]. The base code was defined over GF(5) with  $N = 10000$ ,  $M = 9139$  with mean column weight 2.4 constructed using Construction 1A [4] for the weight 3 columns. Random data symbols were chosen, encoded, transmitted and then decoded. The results are compared with a normal LDPC code, a timesharing scheme (with the optimum time sharing fraction) and a Sparse-Dense code [7] in figure 6(a).

Then two  $N = 10000$ ,  $R = 0.3$ ,  $p_1 = 1/3$  codes were created for testing on the parallel Z channel. The base codes over GF(3) had  $N = 10000$ ,  $M = 8107$  and degree sequences from table 1 (the ‘‘semi-regular’’ degree sequence with only weight 2 and weight 3 columns and the degree sequence with weight 2, 3 and 10 columns were used). Simulation results with the parallel Z channel are shown in figure 6(b).

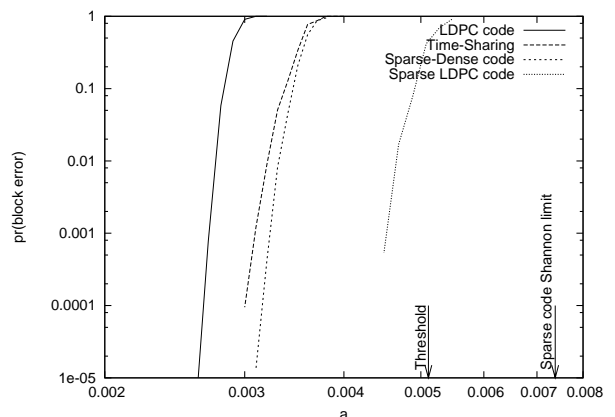
For the two channels studied above, the Sparse LDPC codes can be seen to outperform the other coding schemes tested by 1.6dB and 1.1dB respectively.

## IV. DISCUSSION

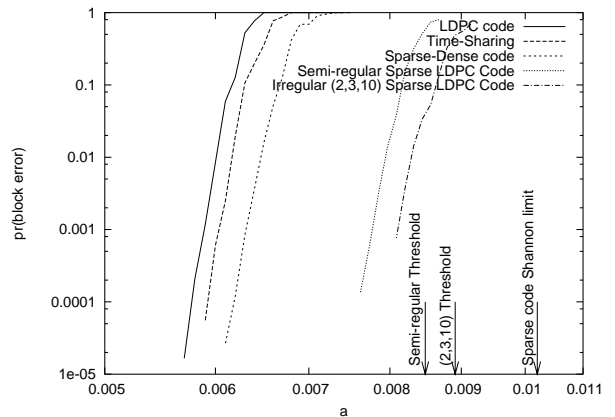
Even in the event of an uncorrupted transmission being received, a decoding is required to recover the original GF( $q$ ) symbols. This approach is therefore best suited to use with good, large block-length codes. For small block-length codes, errors might still be possible at zero noise level.

Col Weight	Proportions		
2	0.65	0.715	0.635
3	0.35	0.193	0.302
5		0.092	
10			0.063
Threshold/0.001	8.48	8.74	8.91

Table 1: Thresholds of three degree sequences for codes defined over GF(3) with  $p_1 = 1/3$  and  $R = 0.3$  over the parallel Z channel with 128 users.



(a) Parallel BSC,  $R = 0.2$



(b) Parallel Z Channel,  $R = 0.3$

Figure 6: Codes with  $N = 10000$  on 128 user multi-user channels. The LDPC codes and Sparse-Dense codes are from [7]. The Sparse-Dense code is a code with sparsified systematic bits but dense parity bits. The time-sharing schemes are LDPC codes being used with the optimum time-sharing fraction.

The complexity of decoding at the check nodes increases with the size of the finite field used [6]. If a value of  $p_1$  is needed that can not be obtained with smallish fields then the coding technique presented may become computationally infeasible. We have used a uniform value of  $p_1$  throughout a codeword – it is possible that using a different value for each codeword symbol might help the decoding process and lead to better code performance.

The method of code construction could be studied further. It may be possible to choose the non-zero values in  $\mathbf{H}$  to create a code with better distance properties.

The performance on single-user asymmetric channels could be tested – Sparse LDPC codes, unlike Sparse-Dense codes, might be able to outperform linear codes on these channels.

## V. CONCLUSION

The codes presented here show a substantial gain over linear codes for a multi-user channel. The codes have a modified linear encoder and a message-passing decoder and thus are easier to implement than many non-linear codes.

## REFERENCES

- [1] Matthew C. Davey. *Error-Correction Using Low-Density Parity-Check Codes*. PhD thesis, University of Cambridge, 1999. Available on the web at <http://www.inference.phy.cam.ac.uk/mcdavey>.
- [2] Matthew C. Davey and David J.C. MacKay. Low-density parity check codes over GF(q). *IEEE Communication Letters*, 2(6):165–167, June 1998.
- [3] Joerg M. Gablonsky and C. Tim Kelley. A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization*, 21:27–37, 2001.
- [4] David J.C. MacKay. Good error correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, 1999.
- [5] Robert J. McEliece. Are turbo-like codes effective on non-standard channels? *IEEE Information Theory Society Newsletter*, 51(4):1–8, December 2001. Based on 2001 ISIT Plenary Lecture.
- [6] Edward A. Ratzert. Complexity analysis of Fourier-transform decoding of LDPC codes over GF(q). Technical Report RZ 3459, IBM Research, 2000.
- [7] Edward A. Ratzert. Sparse data blocks and multi-user channels. Technical report, University of Cambridge, June 2002.
- [8] Richard A. Silverman. On binary channels and their cascades. *IEEE Transactions on Information Theory*, 1(3):19–27, December 1955.
- [9] Niclas Wiberg. *Codes and Decoding on General Graphs*. PhD thesis, Linköping University, 1996.