Sparse Data Blocks and Multi-User Channels

Edward A. Ratzer
Cavendish Laboratory, University of Cambridge
Madingley Road, Cambridge CB3 0HE, UK
e-mail: ear23@mrao.cam.ac.uk

Abstract — **An efficient algorithm for creating sparse data blocks of fixed size is presented. The use of sparse data blocks with a systematic error-correcting code is shown to increase the communication rate on multi-user channels.**

I. A variable-to-fixed sparsifier

An algorithm for mapping a dense data stream to a sparse data stream (consisting of i.i.d. bits with $p_1 \equiv \Pr(\text{digit} = 1) \neq 1/2$) is presented in [1], but it does not address termination and hence the possibility of catastrophic error propagation in a communications context. Our new algorithm creates blocks of fixed size, hence avoiding catastrophic error propagation.

A sparse data block is indistinguishable from a block created by a fixed weight block generator, if the fixed weight is sampled from the correct binomial distribution. We can therefore split the sparsifier into two steps – choosing a weight and generating blocks of that weight.

If we take a prefix code decoder and feed it a stream of i.i.d. bits it will output members of the code's alphabet with probabilities $q_i = 2^{-l_i}$ (where l_i is the codeword length of alphabet member i). A Huffman code was constructed with an alphabet consisting of a subset of the weights. This subset was chosen to be as large as possible, centred around Np_1, such that the q_i were within a factor of two of the binomial distribution. This code's decoder was used to choose the weight of a block.

To generate a block of a particular weight a constant weight code (also known as an M-out-of-N code) was used with the efficient algorithms from [2].

To create a sparse block, our sparsifier first reads off a Huffman codeword from the dense data stream to set the weight. Then it reads off from the dense stream one input word of the corresponding constant weight encoder; its output is the output of our sparsifier. The efficiency of this algorithm is illustrated in figure 1.

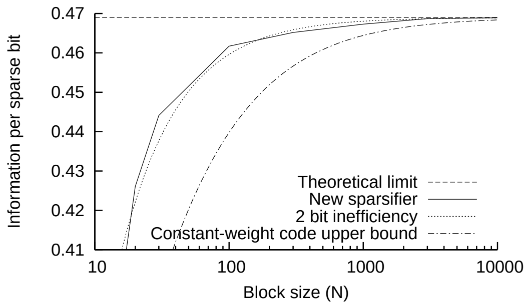

Figure 1: Efficiency of the proposed sparsification algorithm for $p_1 = 0.1$ assuming the worst case performance of the constant weight code. Also shown are $H_2(p_1)$ (a large blocksize limit), $H_2(p_1) - 2/N$ (performance with a 2 bit inefficiency per block) and the performance achievable with just a constant-weight code (of weight Np_1).

II. Sparse-Dense codes

Some multi-user noisy channels can benefit substantially from using a code where zeros and ones are not equiprobable. Using a variable-to-fixed sparsifier followed by a systematic error-correcting code encoder leads to a codeword with low weight, though the parity bits are dense. We call this construction a Sparse-Dense code.

As an example we looked at a simple multi-user channel model from [4]. Each user's channel model is a binary symmetric channel for which the flip probability is proportional to the number of 1s being transmitted on the other channels (with constant of proportionality a).

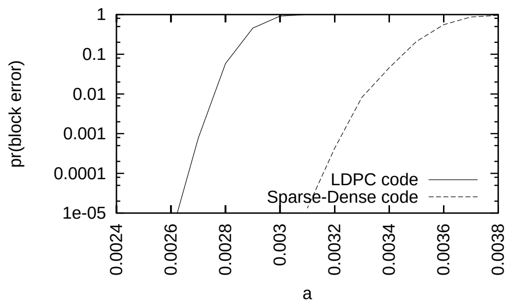

Figure 2: $R = 0.2, N = 10000$ codes on a 128-user parallel BSC as a function of the noise level parameter a.

Two $N = 10000$, $R = 0.2$ codes were constructed. The first was a low-density parity-check (LDPC) code [1]. The second was a Sparse-Dense code created using a $p_1 = 0.11$ sparsifier combined with an $R = 0.4$ LDPC code so that the overall code rate was 0.2. Simulation results are plotted in figure 2. It can be seen that the Sparse-Dense code outperforms a traditional LDPC code.

A technical report on this work is available [3].

References

[1] D.J.C. MacKay. Good error correcting codes based on very sparse matrices. *IEEE Trans. Info. Th.*, 45(2):399–431, 1999.

[2] T.V. Ramabadran. A coding scheme for M-out-of-N codes. *IEEE Trans. Comm.*, 38(8):1156–1163, August 1990.

[3] E.A. Ratzer. Sparse data blocks and multi-user channels. http://www.inference.phy.cam.ac.uk/ear23/papers/.

[4] E.A. Ratzer and D.J.C. MacKay. Sparse low-density parity-check codes for channels with cross-talk. In *IEEE Information Theory Workshop 2003*, pages 127–130, April 2003.