International Symposium on Information Theory and its Applications, ISITA2008
Auckland, New Zealand, 7-10, December, 2008

Improvement of the Davey-MacKay Construction

Johann A. Briffa and Hans Georg Schaathun

University of Surrey, Dept. of Computing
GU2 7XH Guildford, England
E-mail: j.briffa@surrey.ac.uk and h.schaathun@surrey.ac.uk

Abstract

The Davey-MacKay construction is a Deletion-Insertion Correcting Code scheme consisting of an inner code that functions as a pilot sequence to which the receiver seeks to synchronize, and an outer code that provides error protection. We analyse the performance of the inner code in isolation, arguing that these codes provide unequal protection, and demonstrate empirically that the error rate is dependent on the data symbol values. We also propose modifications to the code construction that alleviate this asymmetry. Simulation results show that these codes have an improved performance with no penalty.

1. Introduction

In [1], Davey and MacKay (DM) proposed a Deletion-Insertion Correcting Code (DICC) and applied it to a Binary Substitution, Insertion, and Deletion (BSID) channel that extends the Binary Symmetric Channel (BSC) with the possibility of deletion or unbounded insertions at every timestep. Our initial interest in the DM construction is in applying DICC for image watermarking, as suggested in [2]. This may prove a suitable counter-measure against local geometrical distortion, as in the jitter and StirMark attacks [3].

The DM construction consists of an inner code that functions as a pilot sequence to which the receiver seeks to synchronize, and an outer code that provides error protection. In prior work, we have already improved performance by replacing the LDPC outer code with a q-ary turbo code, while keeping the same inner code [4]. We extend that work here by considering the design of the inner code.

Davey did not address codebook selection for the inner code. In this paper we show that for a sequential codeword selection, the DM inner codes provide unequal protection; simulation results confirm that the error rate is dependent on the data symbol values. We also propose modifications to the code construction that alleviate this asymmetry, and show experimentally that this reduces the error rate of the concatenated code. It is worth clarifying that this improvement is achieved with no penalty.

2. Background

Three different kinds of errors may occur on the BSID channel, namely *deletions*, *insertions*, and *substitutions*. The errors occur with probability P_d, P_i, and P_s respectively. The channel is memory-less, so errors are independent. Substitutions are the kind of error most frequently studied in coding theory, where a single symbol is replaced by a wrong one. Deletions and insertions are known as desynchronisation errors, as they will cause a displacement of the following sequence.

The Davey-MacKay inner code [1] uses a binary sequence of length n called the *watermark*. The watermark is drawn (independently) at random for each n-bit block. A q-ary data symbol, where $q = 2^k$, is encoded as a sparse binary vector of length n, which is added to the watermark. We define a code as *optimally-spread* if it uses all available sequences up to some weight w. The inner code is decoded using a modified Forward-Backward (FB) algorithm, which determines at every block the likelihoods $P(d)$ of each possible transmitted symbol d. The $P(d)$ are then used as soft input for the outer decoder.

It can be shown experimentally that the inner decoder is very effective in maintaining synchronisation over long sequences of blocks, in spite of the 'errors' caused by adding the sparse vector. However, considering the output from the inner decoder, there will be a number of substitution errors which will have to be corrected by another coding layer. Davey and MacKay used a q-ary LDPC code as the outer code.

3. Results

We consider the BSID channel with $P_d = P_i =: p$ and $P_s = 0$. Davey has shown empirically how performance degrades smoothly as P_s is increased, and is not significantly affected when P_s is much less than the

Table 1: Performance of unprotected DM inner codes

k/n	N	α
4/5	3 840	13.94
4/6	3 200	8.18
4/8	2 400	6.29
4/10	1 920	5.49
4/12	1 600	5.08
4/15	1 280	4.60
3/6	3 200	4.64
3/7	2 743	4.24

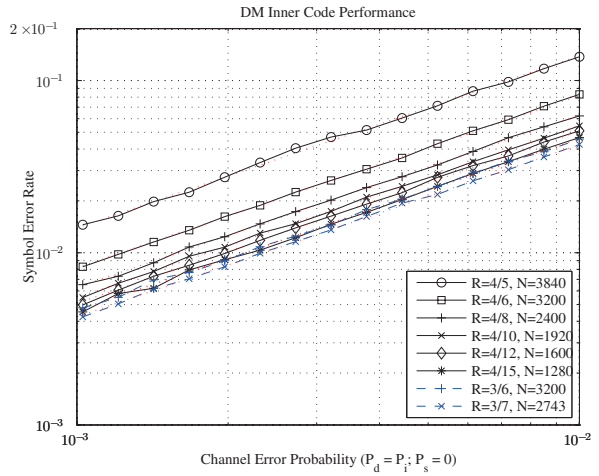

Figure 1: Error performance of DM inner codes

density of the sparse code. While that claim may need to be revisited in light of the changes introduced here, the performance for $P_s = 0$ still provides a useful base reference.

3.1. Simulation of Unprotected DM Inner Codes

For hard-decision decoding, one may describe the operation of the DM inner codes as a translation of channel insertions and deletions at the bit level into substitutions at the symbol level. We seek to understand this performance by simulating DM inner codes without any channel code. For block sizes of 19 200 bits, we simulate the inner codes used by Davey [1] for $k/n = 4/5$ to $3/7$, and lower rate codes down to $k/n = 4/15$.

Through most of its range, the empirical symbol error rate (SER) performance ε_i is proportional to the channel parameter p: $\varepsilon_i = \alpha p$. The effect of k/n on the empirically-determined α is tabulated in Table 1. Corresponding SER results are shown in Figure 1.

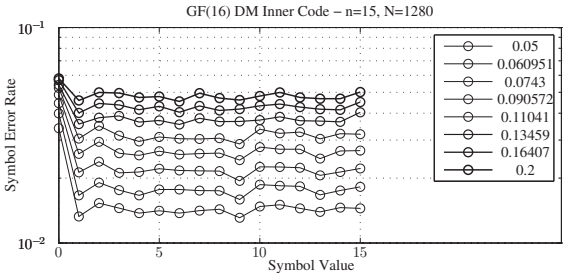

Figure 2: Error profile of $k/n = 4/15$ DM inner code

As expected, higher-rate codes perform worse, in that for the same p, ε_i is higher. Clearly, outer-layer protection is essential for obtaining the performance reported by Davey. Observe also that in terms of SER, the $k/n = 3/7$ DM inner code has the best performance within the simulated set. Surprisingly, it is as good as (or better than) other optimally-spread codes with lower rate and higher sparse-symbol length n, such as the $k/n = 4/15$ code. Within the set of codes with a given value of k, performance improves as expected by increasing n. However, the gain in performance quickly becomes negligible. There is little gain going from $k/n = 4/8$ through $4/15$, and an even smaller difference between that set and the $k/n = 3/6$ and $3/7$ codes used by Davey.

3.2. Error Profile of DM Inner Codes

The isolated performance of the outer turbo codes was considered in [4] for the q-ary symmetric channel (QSC). The QSC is the discrete memoryless channel with q-ary input and output alphabets; for a substitution rate P_s the probability that an input symbol will be unchanged is $(1 - P_s)$; otherwise each of the other $q - 1$ symbols is and equally likely substitution.

We have already shown [4] that the performance of the DM-Turbo codes is not accurately predicted by mapping the turbo code performance on a QSC channel to the DM inner code performance on a BSID channel. A first attempt at understanding this discrepancy is to determine the error profile for DM inner codes as dependent on the message symbol. For uniform random messages in $GF(16)$, we simulate the SER performance, keeping separate error counts according to the corresponding message symbol. Simulations are performed over a range of channel error rates p. Results for the $k/n = 4/15$ code is shown in Figure 2, for $0.05 \leq p \leq 0.2$.

The $k/n = 4/15$ DM inner code is an optimally-spread code, consisting of the zero codeword and all codewords of weight 1. This means that the zero code-

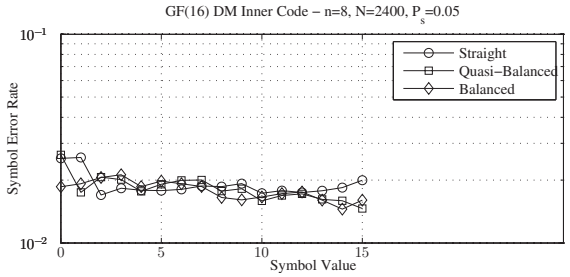

Figure 3: Error profile of DM inner codes with different codebooks

word is at unit Hamming distance from all 15 remaining codewords, while each of the non-zero codewords is at unit Hamming distance from the zero codeword and at a Hamming distance of two from other weight-one codewords. It is then to be expected that message symbols corresponding to the zero codeword will have a higher probability of error, while all other codewords will have approximately equal error rates. This is verified by the simulation results of Figure 2.

Also observe from the figure how the difference in performance is reduced as the channel error rate is increased. This is also to be expected, since increasing p reduces the difference between p and p^2, that is between the likelihoods of a single- or a double-bit error.

3.3. Effect of Codebook Design on Error Profile

When an optimally-spread codebook cannot be found, it is not clear which codewords were used in [1]. Consider, for instance $k/n = 4/8$. We compare three different codebooks.

The *straight* code uses the zero codeword, all eight weight-one codewords, and the first seven weight-two codewords (in lexicographical order). Let $c_i(\mathbf{c})$ be the number of codewords at distance i from $\mathbf{c}$. There are two groups of codewords: 0 and 1 have $c_1 = 8$, while the remaining codewords only have $c_1(\mathbf{c}) = 2$. In this case one expects the error probability to be approximately the same for the 0 and 1 codewords, and lower for the remaining codewords. Simulation results for this code, shown in Figure 3, confirm this prediction.

To some extent c_i depends on the column-weight distribution for the codebook. It is usually desirable in a code that the columns have equal weights. In the $k/n = 4/8$ DM inner code we replace codewords $9 \leq d \leq 15$ so that the column weights are as balanced as possible. Several codebooks achieve this, with six columns of weight 3, and two columns of weight 2. One such codebook is called the *quasi-balanced* codebook in the figure. All the codewords have $c_1 = 2$ or $c_1 = 3$, except zero with $c_1 = 8$. Figure 3 confirms the expected

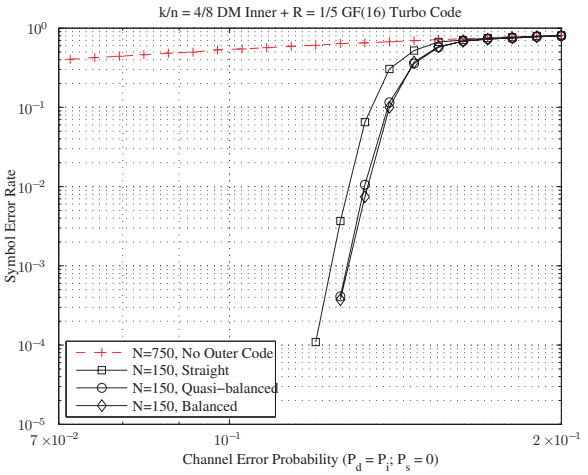

Figure 4: Performance of DM-Turbo code for different codebooks

behavour, where the error probability when word 1 is submitted drops significantly.

Given that turbo codes in general assume a symmetric channel error distribution, one would expect better performance for the concatenated code with a flat error profile. By replacing the zero word by a word of weight 2, we can get $c_1 = 3$ for all columns. Half the words have $c_2 = 2$ and the other half has $c_2 = 7$. The result is the *balanced* codebook with an almost flat error profile in Figure 3. A slight difference can be observed between the two groups of words, but a simulation with tighter tolerance limits is necessary to make a confident statement about this.

3.4. Codebook Design and Error Rates

More interesting than the error profile from the inner decoder, is the error rates of the different codebooks. We have simulated the three different $k/n = 4/8$ inner codebooks with no outer code, with an 80% confidence interval within $\pm 2\%$. Under the same channel conditions, the quasi-balanced codebook reduces SER by 4.2%, and the balanced codebook by 13.7%, both compared to the straight one. Although the difference is relatively small, it is consistent.

More importantly, we want to know how the codebook design affects the error rate of the concatenated code. Figure 4 shows this comparison, using a rate 1/5 outer code and the 4/8 inner codes. Again we can see that both the balanced and quasi-balanced codebooks outperform the straight one in the waterfall region. It is particularly interesting to note that the balanced codebook is the most efficient one in some cases, as it deviates from [1] by not using the lowest weight codeword.

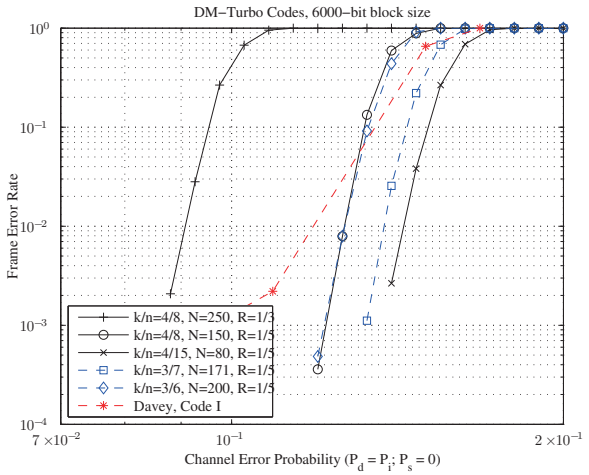

Figure 5: Performance of DM-Turbo codes for a 6 000-bit block size

3.5. Overall Design Choices

Finally we discuss the complete system, and compare it to the results claimed by Davey and MacKay. We have restricted ourselves to turbo codes without puncturing. Let R be the outer code rate, $R_c = \frac{k}{n}R$ the rate of the concatenated code, and $N_c = \frac{1}{R}Nn$. The simulation results are shown in Figure 5 for a frame size of approximately 6 000 channel bits.

Davey's code 'I' has $R_c = 1/20$. We can see that our codes have a steeper waterfall and apparently a lower error floor than code 'I'. The $R_c = 1/10$ has better error rates at twice the information rate of 'I'. The $R_c = 3/35 \approx 1/12$ and $R_c = 4/75 \approx 1/19$ codes give successively better performance at a cost of information rate.

There is little to distinguish the two $R_c = 1/10$ codes, with $q = 8, 16$ respectively. However, it is rather surprising that using $k/n = 3/7$ is a massive improvement over 3/6 at a slight loss of information rate. This partly confirms the results in Table 1, that the $k/n = 3/7$ code is remarkably good.

4. Conclusions

All other things being equal, it appears that the most important performance contributor is the outer code rate. A secondary but significant effect is due to the optimality of the inner code, its codebook design, the overall block size (determined by outer code interleaver size), and the outer code alphabet size. It is difficult to decouple these effects in order to analyse them separately, and further investigation is required to establish a recommended design practice.

The significant effect of the inner code performance suggests that a better explanation of its workings would be useful. There is still no explanation for the improved performance of the $k/n = 3/7$ over the $k/n = 4/15$ inner code (when simulated separately). Given that both codes are optimally-spread, and that the $k = 4$ code has the lower rate, one would rather expect the opposite.

We have shown that the DM inner codes provide unequal protection, and verified by simulation that the error rate is dependent on the data symbol values. Modifications to the code construction have also been proposed to alleviate this asymmetry without any increase in complexity. The improved performance for the modified construction has been verified by simulation.

Acknowledgment

This work was funded by the Engineering and Physical Sciences Research Council UK, grant EP/E056407/1. The first author also expresses his gratitude for funding under the EUMedGrid FP6 project, contract № 026024.

References

[1] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 687–698, 2001.

[2] G. Sharma and D. J. Coumou, "Watermark synchronization: Perspectives and a new paradigm," *Information Sciences and Systems, 2006 40th Annual Conference on*, pp. 1182–1187, 22-24 March 2006.

[3] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Attacks on copyright marking systems," in *Second Workshop on Information Hiding*, ser. Lecture Notes in Computer Science, D. Aucsmith, Ed., vol. 1525. Portland, Oregon, USA: Springer-Verlag, Apr. 14–17th, 1998, pp. 218–238.

[4] J. A. Briffa and H. G. Schaathun, "Non-binary turbo codes and applications," in *Proc. IEEE Intern. Symp. on Turbo Codes & related topics*, Lausanne, Switzerland, Sep. 1–5, 2008, pp. 294–298.