An Improved Decoding Algorithm for the Davey-MacKay Construction

Johann A. Briffa, Hans Georg Schaathun, and Stephan Wesemeyer
University of Surrey, Dept. of Computing
GU2 7XH Guildford, England
Email: J.Briffa@surrey.ac.uk, H.Schaathun@surrey.ac.uk, and S.Wesemeyer@surrey.ac.uk

***Abstract*—The Deletion-Insertion Correcting Code construction proposed by Davey and MacKay consists of an inner code that recovers synchronization and an outer code that provides substitution error protection. The inner code uses low-weight codewords which are added (modulo two) to a pilot sequence. The receiver is able to synchronise on the pilot sequence in spite of the changes introduced by the added codeword.**

The original bit-level formulation of the inner decoder assumes that all bits in the sparse codebook are identically and independently distributed. Not only is this assumption inaccurate, but it also prevents the use of soft a-priori input to the decoder. We propose an alternative symbol-level inner decoding algorithm that takes the actual codebook into account. Simulation results show that the proposed algorithm has an improved performance with only a small penalty in complexity, and it allows other improvements using inner codes with larger minimum distance.

***Index Terms*—deletion-insertion correcting codes, turbo codes, non-binary codes**

I. Introduction

The Davey-MacKay (DM) construction [1] is a Deletion-Insertion Correcting Code (DICC) scheme for the Binary Substitution, Insertion, and Deletion (BSID) channel. In addition to bit substitutions, this channel model allows for the possibility of deletion or unbounded insertions at every timestep. While such channel models have not led to practical solutions for most conventional communication channels, they are receiving increasing interest in digital watermarking (e.g. [2]).

Deletion/Insertion errors in watermarking stem from two causes. Firstly, they are caused by malicious attacks through local geometric distortions. In image watermarking both the jitter and the celebrated StirMark [3] attacks are examples of this. The other common cause is errors in the demodulator. The embedded information is often linked to feature points in the media signal. Noise may change the strength of the feature points, so that the demodulator loses feature points (deletion) or introduces false feature points (insertion). One recent example of this is [4].

The inner code in the DM construction recovers synchronization through a pilot sequence. Information is transmitted by modulating it on the pilot sequence in the form of sparse substitutions. The inner decoder is able to resynchronise in spite of these substitutions. A non-binary outer code is used to correct substitution errors. In practical terms the information is encoded first by the outer code. Each symbol of the resulting codeword is encoded using a non-linear binary code (the inner code) with low-weight codewords, which is added (modulo 2) to the pilot sequence.

The original work used non-binary LDPC codes as outer codes. On poor channels, requiring low rate codes, turbo codes have been shown to have better error performance [5]. We have also previously demonstrated that improvements can be made by fine-tuning the codebook for the inner code [6].

In the original work, the pilot sequence is decoded while treating the modulated information as random, independent substitutions. Clearly, the structure of the inner code means that the bits are dependent, making the original algorithm suboptimal. We propose an alternative, symbol-level decoder, where the pilot sequence is tracked one q-ary symbol at a time taking its probability distribution into account.

By using more information about the underlying code, the new algorithm is theoretically more accurate. The improvement is confirmed in simulations, and especially for high code rates, these improvements are significant. The asymptotic complexity is the same as for the original algorithm, and the actual run time in our simulations is only 0%–20% slower. Furthermore, allowing for soft input, the new algorithm can support iterative decoding, although this is left as an open problem where issues of speed and numerical precision have to be resolved. Preliminary simulations with non-optimised iteration show slight, but encouraging improvements of the frame error rate.

We will start with descriptions of the model and the original algorithm in Section II. The new algorithm is defined in Section III and evaluated in Section IV. The final section summarises and concludes.

II. Background

A. The channel

We define the channel in terms of the transition diagram in Figure 1. At each *time* i, one bit is input to the channel. One of four events may happen: insertion with probability P_i where a random bit is output; deletion with probability P_d where the input is discarded; or transmission with probability $1-P_d-P_i$ where the input bit is output with probability $1-P_s$ or its negation with probability P_s. In the case of deletion or transmission, we proceed to time $i+1$.

A strict terminology is necessary, so we stress that time i is the point where i bits have been input to the channel, and

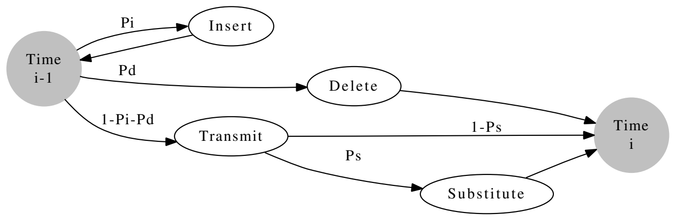

Figure 1. Event transition diagram for the BSID channel

the last event was not an insertion. It necessarily follows that time $i = 0$ is the origin, i.e. before transmission starts.

The output of the channel is defined in terms of the *time-step*, or the period from the input of one bit to the next. During each Time-step i, the channel takes one input bit, and produces η_i output bits; this corresponds to the transition from Time $i-1$ to Time i in Figure 1.

B. The inner code

The inner encoder operates on one q-ary *symbol* at a time. A *sparse codebook* maps each possible symbol $0 \leq d < q$ to a low-weight binary codeword $\lambda(d)$ of length n. Note that q need not be a power of two.

On the other hand, the *decoder* has to address the fact that a *block* of N q-ary symbols has been transmitted. We define a block as the sequence of N symbols d_ι, $0 \leq \iota < N$ transmitted during one use of the system. The block would normally be encoded using a q-ary error-correcting code, to protect against substitution errors from the inner decoder.

The model assumes that synchronisation is guaranteed before and after each block. In a watermarking application, for instance, a block would typically represent the symbols embedded in one file. In other applications, block-level synchronisation may be achieved using conventional pilot sequences.

With a length n inner code and a length N outer code, we require a binary pilot sequence $\mathbf{w}$ of length nN. Davey and MacKay recommended using a random sequence. The block is encoded symbol by symbol using the sparse codebook, to give another length nN sequence which is added modulo two to the pilot sequence. Note that we refer to a length N q-ary sequence as a block, and the corresponding binary sequence of length nN as a frame.

We have three levels of information units—bit, symbol, and block. To avoid confusion, we define the symbol-level timing as follows. *Slot* ι consists of the time-steps corresponding to the transmission of symbol d_ι; that is, to the period from time $n\iota$ to $(n+1)\iota$. *Slot boundary* ι is the point where ι slots have been transmitted; that is, the point at the end of slot $\iota - 1$ and the beginning of slot ι. This corresponds to time $n\iota$, and it necessarily follows that slot boundary $\iota = 0$ corresponds to the fixed temporal origin.

Each symbol d_ι is represented by the sum of the corresponding segment from the *pilot sequence* with the sparse symbol representing d_ι. For any word or vector $\mathbf{z} = (z_0, z_1, z_2, \ldots, z_m)$, we let $\mathbf{z}|_a^b$ denote the subword $(z_a, z_{a+1}, \ldots, z_{b-1})$. Note that, in the case of a transmitted frame $\mathbf{z}$, the bits transmitted between Time a and Time b form the word $\mathbf{z}|_a^b$.

At bit level, the *transmitted frame* is the sequence of τ bits (t_i), $0 \leq i < \tau$ representing the block of N symbols. Since each symbol is represented by n bits, the frame consists of $\tau = nN$ bits. Thus the transmission of one τ-bit frame lasts from time 0 to time τ, and spans τ time-steps numbered $0 \ldots \tau - 1$. Symbol d_ι is represented by the segment $\mathbf{t}_\iota = \mathbf{t}_{n\iota, n(\iota+1)}$, which is equal to $\mathbf{w}_\iota \oplus \lambda(d_\iota)$.

Similarly, the *received frame* is the sequence of ρ bits (r_i), $0 \leq i < \rho$ corresponding to the transmitted frame. It is assumed that the frame boundaries are known.

The amount of desynchronisation at a particular point in time is defined as the *drift*. Formally, the drift ς_i is the difference between the number of transmitted bits and the number of received bits up to time i. That is, $\varsigma_i = \sum_{k=0}^{i-1}(\eta_i - 1)$. The state of the decoder is used to represent the drift.

C. The Bit-Level Algorithm

Davey and MacKay used a forward/background algorithm for inner decoding. It is easiest to understand the algorithm if we start with the last step. The output is the likelihood of each possible transmitted symbol, for each slot, given the transmitted sequence, written as follows:

$$\begin{aligned} L(d_\iota) = \sum_{x_1,x_2} & F(n\iota, x_1) \\ & \cdot \Pr\left\{\mathbf{r}|_{n\iota+x_1}^{n(\iota+1)+x_2} \mid d_\iota\right\} \cdot B(n(\iota+1), x_2), \end{aligned} \tag{1}$$

where

$$F(i, x_1) = \Pr\left\{\mathbf{r}|_0^{n\iota+x_1}, \varsigma_i = x_1\right\} \tag{2}$$

$$B(i, x_2) = \Pr\left\{\mathbf{r}|_{n(\iota+1)+x_2}^{\rho} \mid \varsigma_i = x_2\right\} \tag{3}$$

The summation indices x_1 and x_2 refer to the state of the system respectively before and after the transmission of symbol d_ι. The state of the system is the (possibly negative) difference between the number of received and the number of transmitted bits.

The two terms $F(i,x)$ and $B(i,x)$ are called the forward and backward metrics, and are calculated by (forward and backward) recursion on the time index i. Again, x refers to the state of the system at time i. A fundamental component of this calculation is the conditional probability $\Pr\{\mathbf{r}|t\}$ of a received sequence $\mathbf{r}$, given a transmitted bit t, which can be calculated using the following lemma (formula derived in [1]).

Lemma 1 *The channel-receiver probability function can be calculated as follows:*

$$\begin{aligned} R(\mathbf{r}, t) &= \Pr\{\mathbf{r} \mid t\} \\ &= \begin{cases} P_d & \text{if } \mu = -1 \\ \left(\frac{P_i}{2}\right)^\mu \left(P_t P_s + \frac{1}{2} P_i P_d\right) & \text{if } \mu \geq 0, r_\mu \neq t \\ \left(\frac{P_i}{2}\right)^\mu \left(P_t (1 - P_s) + \frac{1}{2} P_i P_d\right) & \text{if } \mu \geq 0, r_\mu = t, \end{cases} \end{aligned}$$

where $P_t = (1 - P_i - P_d)$ is the probability of a correct transmission.

The recursive formulæ are given as follows:

$$F(i,x) = \sum_{\varsigma} F(i-1,\varsigma) R(\mathbf{r}|_{i-1+\varsigma}^{i+x}, t_{i-1}), \tag{4}$$

$$B(i,x) = \sum_{\varsigma} B(i+1,\varsigma) R(\mathbf{r}|_{i+x}^{i+1+\varsigma}, t_i), \tag{5}$$

for $1 \le i < \tau$, and

$$F(0,x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$B(\tau,x) = \begin{cases} 1 & \text{if } x = \rho - \tau \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The last factor $\Pr\left\{\mathbf{r}|_{n\iota+x_1}^{n(\iota+1)+x_2} \mid d_\iota\right\}$ in the calculation of $L(d_\iota)$ is the probability of receiving a particular sequence $\mathbf{r}$ as the result of transmitting a single syumbol d_ι. This is calculated by calculating the forward metrics over a single slot (as opposed to the entire frame):

$$\Pr\left\{\mathbf{r}|_0^\nu \mid \mathbf{t}|_0^n\right\} = \Pr\left\{\mathbf{r}|_0^\nu, \varsigma_n = \nu - n\right\},$$

where $\mathbf{t}$ is the transmitted bits resulting from the watermark sequence from index $n\iota + x_1$ to $n(\iota+1) + x_2$ and the sparse encoding of d_ι, and ν is the number of bits received, given by $\nu = n + x_2 - x_1$.

III. Symbol-level Algorithm

As an alternative to the decoding algorithm proposed by Davey, we propose an algorithm where the forward-backward algorithm is applied at q-ary symbol boundaries. The state machine is still defined at bit level, as this depends uniquely on the BSID channel model. The forward and backward metrics, on the other hand, recurse on *slot indices* instead of bit indices. Consider again the reception of a single frame of ρ bits (r_i), $0 \le i < \rho$ corresponding to a transmitted frame of τ bits (t_i), $0 \le i < \tau$.

A. Implementation Details

As in [1], we define two implementation parameters that determine the decoding complexity: the limit on successive insertions I and the limit on considered drift $x_{\max}$.

The path complexity is limited by the value I, which specifies the maximum number of (successive) insertion events in a single time-step. In other words, once I insertions occur in a particular time-step, the next event *must* be a transmission or a deletion. This also means that the difference in drift between time i and $i+1$ is limited by $-1 \le \varsigma_{i+1} - \varsigma_i \le I$, representing a deletion and no insertions at one end, and a transmission and I insertions at the other. The choice of I depends on P_i and on the sequence length being considered. Davey claims that a fixed value of $I = 2$ causes only minimal degradation in decoding performance; we stick to that value in this paper.

The memory complexity is limited by the value $x_{\max}$, which specifies the range of legal drift values at any time i as $-x_{\max} \le \varsigma_i \le x_{\max}$. The choice of $x_{\max}$ depends on P_i and P_d and on the sequence length being considered; as in [1] we consider channels where $P_i = P_d$. For the sake of clarity, we use the term $x_{\max}$ when dealing with the whole frame; we define $\delta_{x\max}$ as the corresponding term when dealing with the bits in a single slot.

B. Computation of Symbol-level Forward Metrics

Definition 1 (Symbol-level Forward Metric) *The symbol-level forward metric $\alpha(\iota, x)$ at slot boundary ι and state x is defined as the joint probability*

$$\alpha(\iota, x) = \Pr\left\{\mathbf{r}|_0^{n\iota+x}, \varsigma_{n\iota} = x\right\}. \tag{8}$$

In other words, the symbol-level forward metric is the probability that the state at slot boundary ι is x, and that the first $n\iota + x$ bits received correspond to the bits emitted by the channel up to that time.

The symbol-level forward metric is computed from the receiver likelihoods using the recursion:

$$\alpha(\iota, x_2) = \sum_{x_1, d_{\iota-1}} \alpha(\iota-1, x_1) \Pr\left\{\mathbf{r}|_{n(\iota-1)+x_1}^{n\iota+x_2}, \mathbf{t}_{\iota-1}\right\} \tag{9}$$

for $1 \le \iota < N$. This represents the summation over all possible prior states and all possible symbols transmitted in the previous slot:

$$\Pr\left\{\mathbf{r}|_0^{n\iota+x_2}, \varsigma_{n\iota} = x_2\right\} = \sum_{x_1, d_{\iota-1}} \begin{bmatrix} \Pr\left\{\mathbf{r}|_0^{n(\iota-1)+x_1}, \varsigma_{n(\iota-1)} = x_1\right\} \\ \times \Pr\left\{\mathbf{r}|_{n(\iota-1)+x_1}^{n\iota+x_2}, \mathbf{t}_{\iota-1}\right\} \end{bmatrix} \tag{10}$$

Initial conditions for the forward metrics are given by:

$$\alpha(0, x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Observe that unlike Davey's bit-level algorithm, there is no need to average over changes due to the sparse vector. The problem of determining the receiver likelihood $\Pr\left\{\mathbf{r}|_{n(\iota-1)+x_1}^{n\iota+x_2}, \mathbf{t}_{\iota-1}\right\}$ is calculated as for the bit-level algorithm.

C. Computation of Symbol-level Backward Metrics

Definition 2 (Symbol-level Backward Metric) *The symbol-level backward metric $\beta(\iota, x)$ at slot boundary ι and state x is defined as the conditional probability*

$$\beta(\iota, x) = \Pr\left\{\mathbf{r}|_{n\iota+x}^{\rho} \mid \varsigma_{n\iota} = x\right\}. \tag{12}$$

In other words, the symbol-level backward metric is the probability that the channel would emit the sequence $\mathbf{r}|_{n\iota+x}^{\rho}$ from slot ι up to the end of the frame, given that the system is in state x at slot boundary ι.

The symbol-level backward metric is computed in a similar way to the forward metric, using the recursion:

$$\beta(\iota, x_1) = \sum_{x_2, d_\iota} \beta(\iota+1, x_2) \Pr\left\{\mathbf{r}|_{n\iota+x_1}^{n(\iota+1)+x_2}, \mathbf{t}_\iota\right\} \tag{13}$$

for $N > \iota \geq 1$. This represents the summation over all possible posterior states and all possible symbols transmitted in the next slot:

$$\begin{aligned}\Pr&\left\{\mathbf{r}|^{\rho}_{n\iota+x_1} \mid \varsigma_{n\iota}=x_1\right\} \\ &= \sum_{x_2,d_\iota}\begin{bmatrix}\Pr\left\{\mathbf{r}|^{\rho}_{n(\iota+1)+x_2} \mid \varsigma_{n(\iota+1)}=x_2\right\} \\ \times \Pr\left\{\mathbf{r}|^{n(\iota+1)+x_2}_{n\iota+x_1}, \mathbf{t}_\iota\right\}\end{bmatrix}\end{aligned} \tag{14}$$

Initial conditions for the backward metrics are given by:

$$\beta(N,x) = \begin{cases}1 & \text{if } x=\rho-\tau \\ 0 & \text{otherwise}\end{cases} \tag{15}$$

As in the computation of the symbol-level forward metrics, there is no need to average over changes due to the sparse vector.

D. Decoder Output

The decoder output for the symbol-level algorithm is computed by replacing the forward and backward metrics in Eq. (1), using the new symbol-level quantities $\alpha(\cdot)$ and $\beta(\cdot)$:

$$L(d_\iota) = \sum_{x_1,x_2}\begin{bmatrix}\alpha(\iota,x_1)\beta(\iota+1,x_2) \\ \cdot\Pr\left\{\mathbf{r}|^{n(\iota+1)+x_2}_{n\iota+x_1} \mid d_\iota\right\}\end{bmatrix}.$$

IV. Results

To investigate the performance of the symbol-level decoder, we consider a number of constructions, at different information rates. Firstly, we consider the rate $1/10$ codes simulated in [6] using turbo outer codes. We have also reimplemented Davey's original Codes F (rate $\frac{1}{2}$) and D (rate 0.71) using LDPC codes.

The channels is a BSID channel with $P_d = P_i =: p$ and $P_s = 0$. Simulated results have an 80% confidence interval of $\pm 20\%$. Path truncation is not performed, so that results are independent of any effect it may have on decoding performance.

A. Rate $1/10$

We compare results for two systems with a frame size of approximately $6\,000$ channel bits, one using the $(7,8)$ inner code and the other using a $(8,16)$ inner code with a balanced codebook.

The outer codes are unpunctured turbo codes of rate $R = 1/5$. For $q = 8$ we use an outer code in $\mathsf{GF}(8)$ with an interleaver size of $N = 171$. Its constituent codes are of memory order $\nu = 2$ (64-state) with feedback polynomial $1+D+\alpha^4D^2$ and feed-forward polynomials $1+\alpha D+\alpha^4D^2$ and $1+\alpha^5D+\alpha^4D^2$, where α is a root of x^3+x+1. For $q = 16$ the outer code is in $\mathsf{GF}(16)$ with $N = 150$; constituent codes are of memory order $\nu = 2$ (256-state) with feedback polynomial $1+D+\alpha^4D^2$ and feed-forward polynomials $1+\alpha D+\alpha^4D^2$ and $1+\alpha^2D+\alpha^9D^2$, where α is a root of x^4+x+1. Both codes use unterminated trellises, and an S-random interleaver [7] of block size N and spread $S = 9, 10$ for $N = 150, 171$ respectively.

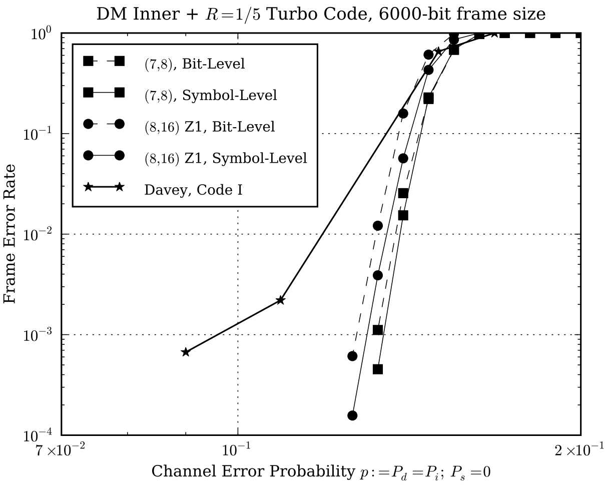

Figure 2. Performance of DM-turbo codes with symbol-level decoder

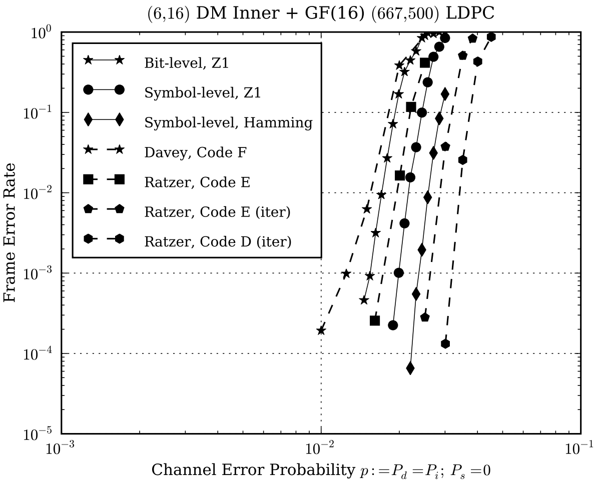

Figure 3. Error performance of rate-$1/2$ DM-LDPC codes with symbol-level inner decoder, and comparison with bit-level decoder and published results.

We can see from the frame-error rate (FER) results in Figure 2 that for the same code, the decoding performance improves when using symbol-level inner decoding. It is notable that the symbol-level decoder results in a greater improvement in the $4/8$ code than the optimally-spread $3/7$ code. This makes sense because the approximation made by the original algorithm assuming independent bits is more severe when the inner code has higher density (i.e. more one-bits per column in the codebook).

B. Rate $\frac{1}{2}$ and 0.71

Figure 3 shows simulations for Davey's Code F, with a $(6, 16)$ inner code and rate $500/667$ LDPC outer code. Similarly, Figure 4 shows Davey's Code D, with a $(5, 16)$ inner code and rate $8/9$ LDPC outer code. Additionally, we have shown Ratzer's results [8] for marker codes at similar rates.

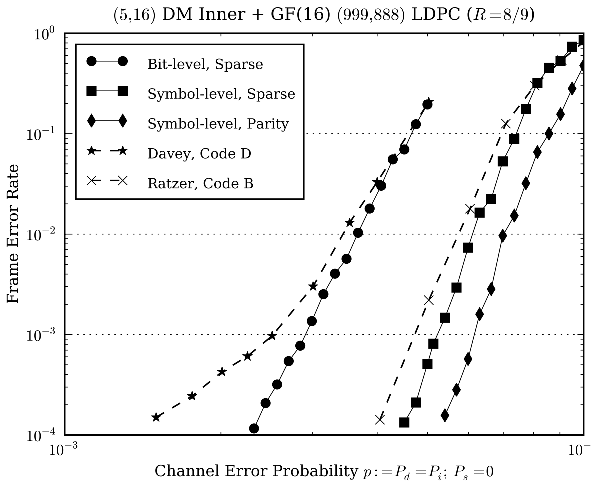

Figure 4. Error performance of high rate ($R = 0.71$) DM-LDPC codes with symbol-level inner decoder, and comparison with bit-level decoder and published results.

Firstly, we observe that the benefit of symbol-level decoding which was only a slight improvement at rate $1/10$ becomes very significant at higher rates.

We have also shown results with alternative, non-sparse inner codes. For Code F, we have a $[6, 4, 2]$ punctured Hamming code, and for Code D, a $[5, 4, 2]$ parity check code. These codes cause too many substitutions on the pilot sequence to work with the bit-level decoder. They become decodable because the symbol-level decoder takes the actual codewords into account, and then they perform better because of the increased minimum distance.

V. Conclusions

We have proposed a symbol-level decoding algorithm for the Davey-MacKay construction that results in improved performance at a small cost in complexity. The improvement is particularly significant at high code rates. With the proposed algorithm, the Davey-MacKay construction also outperforms Ratzer's marker codes with non-iterative decoding, although not marker codes with iterative decoding.

An important aspect of our result is that the disadvantage in decoding speed is negligible. It would have been reasonable to expect a significant increase in computational cost, which may have discouraged this research in the past. However, this is not the case. It can be shown that the asymptotic complexity is the same as for Davey's bit-level decoder, namely $O(Nnqx_{\max}\delta^2_{x\max}I)$. Empirical running times show a modest 0%–20% increase in decoding time per block, depending on code parameters and channel conditions.

Iterative decoding remains an open problem. The symbol-level decoder allows soft *a priori* input, and is thereby prepared for iteration. Preliminary experiments show a slight, but encouraging improvement in frame error rates. However, there are issues of numerical precision and computational optimisation which must be solved before a reliable iterative decoder can be created. Also for Ratzer's Marker Codes, iterative decoding is not fully understood. The optimal Marker Code with non-iterative decoding is not optimal with iteration, and the reason for this is not known.

This decoding algorithm does not assume that all bits in the sparse codebook have the same distribution, removing earlier restrictions on inner code construction, and we demonstrated that this can be exploited by using inner codes with larger minimum distance. The effect of path truncation (as suggested by Davey) on decoding performance and complexity still needs to be investigated. At low code rates, preliminary results indicate that only marginal speed gain is possible without a significant loss in decoding performance. However, path truncation still seems to have potential at high code rates. Optimising the truncation thresholds for particular channel parameters remains an open question.

Acknowledgment

This work was funded by the Engineering and Physical Sciences Research Council UK, grant EP/E056407/1.

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] D. J. Coumou and G. Sharma, "Insertion, deletion codes with feature-based embedding: A new paradigm for watermark synchronization with applications to speech watermarking," *Information Forensics and Security, IEEE Transactions on*, vol. 3, no. 2, pp. 153–165, June 2008.

[5] 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.

[6] ——, "Improvement of the Davey-MacKay construction," in *Proc. IEEE Intern. Symp. on Inform. Theory and its Applications*, Auckland, New Zealand, Dec. 7–10, 2008, pp. 235–238.

[7] D. Divsalar and F. Pollara, "Turbo codes for deep-space communications," Jet Propulsion Laboratory, California Institute of Technology, TDA Progress Report 42-120, Feb. 15th, 1995.

[8] E. A. Ratzer, "Marker codes for channels with insertions and deletions," *Annals of Telecommunications*, 2003.