Decoding of LDPC Block Codes over Convolutional Codes with Channels

Gaurav Vijay¹, Ravindra Prakash Gupta²
¹Research Scholar, Department of Electronics and Communication Engineering, Career Point University, Kota
²Professor, Department of Electronics and Communication Engineering, Manda Institute of Technology, Bikaner, India-334001

Abstract
Decoding of LDPC block codes over Convolutional codes with channels have been shown to be capable of achieving the same capacity-approaching performance as LDPC block codes with iterative message-passing decoding. However, for comparing block and convolutional codes tied to the implementation complexity of trellis based decoding are irrelevant for message-passing decoders. In this paper, we shows a comparison of LDPC block and convolutional codes based on several factors. In this paper the erasure channel are studied. Of special interest will be maximum distance profile (MDP) convolutional codes. These are codes which have a maximum possible column distance increase. This is shown how this strong minimum distance condition of MDP convolutional codes help us to solve error situations that maximum distance separable (MDS) block codes fail to solve. For this, two subclasses of MDP codes are defined: reverse-MDP convolutional codes and complete-MDP convolutional codes. Reverse-MDP codes have the capability to recover a maximum number of erasures using an algorithm which runs backward in time. Complete-MDP convolutional codes are both MDP and reverse-MDP codes. They are capable to recover the state of the decoder under the mildest condition. It is shown that complete-MDP convolutional codes perform in many cases better than comparable MDS block codes of the same rate over the erasure channel.

Keywords: Convolutional codes, maximum distance separable (MDS) block codes, decoding, erasure channel, maximum distance profile (MDP) convolutional codes, reverse-MDP convolutional codes, complete-MDP convolutional codes, maximum distance profile (MDP) convolutional codes, reverse-MDP convolutional codes.

I. INTRODUCTION
As an erasure channel is transmitting, one of the problems encountered is the delay experienced on the received information due to the possible retransmission of lost packets. One way to eliminate these delays is by using forward error correction. After the invention of turbo codes, researchers became aware that Gallager's low-density parity check (LDPC) block codes [1], were also capable of capacity-approaching performance on a variety of channels. Low-density parity-check (LDPC) codes, although introduced in the early 1960’s [20], were established as state-of-the-art codes only in the late 1990’s with the application of statistical inference techniques [21] to graphical models representing these codes [22], [23]. The promising results from LDPC block codes encouraged the development of convolutional codes defined by sparse parity-check matrices. Analysis and design of these codes quickly attracted considerable attention in the literature, beginning with the work of Wiberg [2], MacKay and Neal [3], and many others. The convolutional counterparts of LDPC block codes, namely LDPC convolutional codes, were subsequently proposed in [4]. Analogous to LDPC block codes, LDPC convolutional codes are defined by sparse parity-check matrices that allow them to be decoded using a sliding window-based iterative message passing decoder. The use of convolutional codes over the erasure channel has been studied much less. Recent studies have shown that LDPC convolutional codes are suitable for practical implementation in a number of different communication scenarios, including continuous transmission as well as block transmission in frames of arbitrary size [5], [6], [7]. In this paper, we define a class of convolutional codes with strong distance properties, which we call complete maximum distance profile (complete-MDP) convolutional codes, and we demonstrate how they provide an attractive alternative. They are also known for their encoding simplicity, since the original code construction method proposed in [4] yields a shift-register based systematic encoder for real time encoding of continuous data. This is an advantage when compared to randomly constructed LDPC block.
codes. Given their excellent bit error rate (BER) performance along with their simplicity of encoding, it is quite natural to compare LDPC convolutional codes with corresponding LDPC block codes. In this paper, we compare these codes under several different assumptions: equal decoding computational complexity, equal decoding processor (hardware) complexity, equal decoding memory requirements, and equal decoding delay.

The paper is organized as follows. In Section II, we provide a brief overview of LDPC convolutional codes. The main contribution of the paper is Section III, where comparisons of LDPC block and convolutional codes based on several criteria are presented. In the next section, we focus on finite block length comparisons between LDPC block codes and terminated LDPC convolutional codes. Finally, we provide some conclusions in Section V.

II. AN LDPC CONVOLUTIONAL CODES

Here defines the LDPC-CC as a rate R = b/c binary, time-varying LDPC-CC is defined as the set of semi-infinite binary row vectors v[∞] satisfying equation vH^T=0. An (m; J;K) regular LDPC convolutional code is the set of sequences v satisfying the equation vH^T = 0, where

\[
H^T = \begin{bmatrix}
H_0^T \left[ (0) \right] & \cdots & H_{m_2}^T \left[ (m_2) \right] \\
\vdots & \ddots & \vdots \\
H_{m_2}^T \left[ (t) \right] & \cdots & H_{m_2}^T \left[ (t + m_2) \right]
\end{bmatrix} \tag{1}
\]

As the given parameter m_2 is called the memory of the code and v_i = (m_2 + 1) c is referred to as the constraint length H^T is the (time-varying) semi-infinite syndrome former (transposed parity-check) matrix. For a rate R = b/c, b < c, LDPC convolutional code, the elements H^T(t), i =0,1,2,3,..m_2 are binary c X (c - b) sub matrices defined as

\[
H_i^T \left[ (t) \right] = \begin{bmatrix}
h_i \left[ (1,c-b) \right] (t) & \cdots & h_i \left[ (c,c-b) \right] (t) \\
\vdots & \ddots & \vdots \\
h_i \left[ (c-1,c-b) \right] (t) & \cdots & h_i \left[ (c,c-b) \right] (t)
\end{bmatrix} \tag{2}
\]

Starting from the m_2 (c - b)^th column, H^T has J ones in each row and K ones in each column. The value m_2, called the syndrome former memory, is determined by the maximal width of the nonzero area in the matrix H^T, and the associated constraint length is defined as. v_i = (m_2 + 1) c. In practical applications, periodic syndrome former matrices are of interest. Periodic syndrome formers are said to have a period T if they satisfy H_i^T (t) = H_i^T (t+T), i=0, 1..m_2.

The advantage that convolutional codes to block codes, which will be exploited in our algorithms, is the flexibility obtained through the “sliding window” characteristic of convolutional codes. The received information can be grouped in appropriate ways, depending on the erasure bursts, and then be decoded by decoding the “easy” blocks first. This flexibility in grouping information brings certain freedom in the handling of sequences; we can split the blocks in smaller windows, we can overlap windows and we can proceed to decode in a less strict order.

Although the corresponding Tanner graph has an infinite number of nodes, the distance between two variable nodes that are connected to the same check node is limited by the syndrome former memory of the code. This allows continuous decoding that operates on a finite window sliding along the received sequence, similar to a Viterbi sliding along the received sequence, [4]. The decoding of two variable nodes that are at least (m_2+1) time units apart can be performed independently, since the corresponding bits cannot participate in the same parity-check equation. This allows the parallelization of the I iterations by employing I independent identical processors working on different regions of the Tanner graph simultaneously. Alternatively, since the processors implemented in the decoder hardware are identical, single hopping. Processor that runs on different regions of the decoder memory successively can also be employed. A pipeline decoding architecture that is based on the ideas summarized in the previous paragraph was introduced by JimenezFelstrom and Zigangirov in [4]. The pipeline decoder outputs a continuous stream of decoded data once an initial decoding delay has elapsed. The operation of this decoder on the Tanner graph for a simple time-invariant rate R = 1/3 LDPC convolutional code with m_2 = 2 is shown in Figure 1. (Note that, to achieve capacity-approaching performance, an LDPC convolutional code must have a large value of m_2).
III. COMPARISONS OF LDPC BLOCK AND CONVOLUTIONAL CODES WITH AN IMPLEMENTATION

Here, we compare several aspects of decoding LDPC convolutional and block codes.

A. Complexity With Computational Methodology

Let $C_{\text{check}}$ ($C_{\text{var}}$) denote the number of computations required for a check (variable) node update for a check (variable) node of degree $N$ ($M$). Regardless of the code structure, $C_{\text{check}}$ and $C_{\text{var}}$ only depend on the values $M$ and $N$. For a rate $R = b/c$, $(m_s; M, N)$-LDPC convolutional code decoded using a pipeline decoder with $I$ iterations/processors, at every time instant each processor activates $c - b$ check nodes and $c$ variable nodes. The computational complexity per decoded bit is therefore given by

$$C_{\text{check}} = (c - b) \cdot C_{\text{check}} + c \cdot C_{\text{var}} \cdot I$$

$$C_{\text{var}} = (c - b) \cdot C_{\text{check}} + c \cdot C_{\text{var}} \cdot I$$

which is independent of the constraint length $v_s$.

Similarly, the decoding complexity for an $(L, M, N)$-LDPC block code is given by

$$C_{\text{block}} = (L \cdot M/N \cdot C_{\text{check}} + L \cdot C_{\text{var}}, 1/L) \cdot I$$

$$C_{\text{var}} = (1 - R) \cdot C_{\text{check}} + C_{\text{var}} \cdot I$$

which is again independent of the code length $L$.

Thus, there is no difference between block and convolutional LDPC codes with respect to computational complexity.

B. Processor (Hardware) Complexity

The sliding window decoder implementation of an LDPC convolutional code operates on $I \cdot v_s$ symbols. However, decoding can be carried out by using $I$ identical independent parallel processors, each capable of handling only $v_s$ symbols. Hence, it is sufficient to design the processor hardware for $v_s$ symbols. For an LDPC block code of length $L$, the processor must be capable of handling all $L$ symbols. Therefore, for the same processor complexity, the block length of an LDPC block code must be chosen to satisfy $L = v_s$.

C. Memory Requirements

For the pipeline decoder, we need a storage element for each edge in the corresponding Tanner graph. Each variable node also needs a storage element for the channel value. Thus, a total of $I \cdot (1 + (M + 1) \cdot v_s)$ storage elements are required for $I$ iterations of decoding. Similarly, we need $L \cdot (1 + (M + 1) \cdot v_s)$ storage elements for the decoding of an LDPC block code of length $L$. Thus, for the same memory requirements, an LDPC block code must satisfy $L = I \cdot v_s$.

D. Decoding Delay

Let $T_{ss}$ denote the time between the arrival of successive symbols, i.e., the symbol rate is $1/T_{ss}$. Then the maximum time from the arrival of a symbol until it is decoded is given by

$$\Delta_{\text{dec}} = ((c - 1) + (m_s + 1) \cdot I) \cdot T_{ss}$$

The first term $(c - 1)$ in (5) represents the time between the arrival of the first and last of the $c$
encoded symbols output by a rate \( R = \frac{b}{c} \) convolutional encoder in each encoding interval. The dominant second term \( (m_s + 1) \cdot 1 \) is the time each symbol spends in the decoding window. Since \( c \) symbols are loaded into the decoder simultaneously, the pipeline decoder also requires a buffer to hold the first \((c - 1)\) symbols.

With LDPC block codes, data is typically transmitted in a sequence of blocks. Depending on the data rate and the processor speed, several scenarios are possible. We consider the best case for block codes, i.e., each block is decoded by the time the first bit of the next block arrives. This results in a maximum input-output delay of \( L_{block} = K \cdot T_{cell} \). Thus, for equal decoding delays, the block length must satisfy \( L = (c - 1) + v_s \cdot 1 \), assuming the least possible delay for block codes.

**E. VLSI implementation requirements**

As previously noted, both LDPC block and convolutional codes can be decoded using message passing algorithms. Therefore decoder implementations in both cases consist of identical processing elements, namely variable nodes and check nodes. What differs between the two decoders is the total number of these elements and the way in which they are interconnected. It is well known that VLSI implementations of parallel LDPC block decoders suffer from an interconnection problem [9]. This is due to the fact that processing nodes must be placed on the silicon at specific locations and connected as defined by \( H \). Regardless of how the rows and columns of \( H \) are permuted, long interconnections are still required. The same observation was also noted for LDPC block codes constructed using algebraic techniques [10]. However, VLSI implementations of LDPC convolutional decoders are based on replicating identical units, termed processors. As illustrated in Fig. 2, the complete decoder can be constructed by concatenating a number of these processors together. For comparable BER performance, the size of an LDPC convolutional code processor needs to be about an order of magnitude less than the block length of an LDPC block code [11]. Therefore the routing complexity within a processor is also an order of magnitude less than for a block code.

There is a wealth of other considerations than impact upon VLSI implementations of LDPC codes. These include the following.

- The fully parallel LDPC block code decoder can be replaced with a smaller decoder that implements a fraction of the circuit per clock cycle over a number of cycles. This reduces power, area, and throughput in a linear fashion.

- The LDPC convolutional code architecture is more amenable to pipelining because it is inherently feed forward architecture. Therefore it may achieve higher clock speeds.

- Since LDPC convolutional code decoders require fewer check and variable processing elements, the marginal cost of optimizing their critical paths in return for increasing their area is less than for LDPC block codes.

- Memory-based architectures have been proposed for both LDPC block codes [12] and LDPC convolutional codes [13].

- Algebraic code construction techniques can simplify LDPC block code decoder implementations [14], [15]. However since LDPC convolutional codes can often be constructed using the same methods these benefits may be applied to both types of codes [11].

- The choice may be affected by system level issues such as variable frame size, variable code rates, latency budgets, and target frame error rate (FER).

![Fig. 2. LDPC convolutional code decoders can be implemented by concatenating sub-units termed processors.](image-url)
In Table I we present a brief summary of some existing LDPC block code and LDPC convolutional code VLSI implementations. In [16] a 54 MBPS decoder was implemented on a Xilinx Virtex-E FPGA. In [9] the decoder throughput was much higher (500 MBPS) because the design was implemented as an ASIC. The BER was about $2 \times 10^{-5}$ at 2 dB. The decoder presented in [10] targets the IEEE 802.3an standard [17], but it is a hard decision decoder. This permits a very high throughput but compromises performance. Also note that the codes in [12] and [10] are high-rate, which makes it easier to achieve higher throughput since less processing is required per information bit. The decoder presented in [7] is for a very powerful LDPC convolutional code, and as such it achieves very good performance at the cost of relatively high complexity and low throughput. In [18] the first ever LDPC convolutional code ASIC decoder is presented. It occupies three times less area (in a larger process) than the LDPC block code ASIC in [9], however it has about three times less throughput and the BER performance is worse. The discussion in this section illustrates the point that making a comparison between LDPC block codes and LDPC convolutional codes that includes BER/FER performance and VLSI implementation complexity is not simple. It depends on code choice, throughput, power, area, clock speeds, processing node sizes, and system considerations. It is very possible that there is no single best choice between LDPC block codes and LDPC convolutional codes. Instead, LDPC block codes and LDPC convolutional codes may provide complementary solutions and the appropriate choice may vary from one system to another. It is worth noting that LDPC block codes have been the focus of extensive research for several years whereas LDPC convolutional codes are relatively understudied. The first attempts to implement decoders for LDPC convolutional codes noted here are encouraging enough to suggest that further investigation is warranted.

IV. BER-FER PERFORMANCE COMPARISON OF LDPC BLOCK AND CONVOLUTIONAL CODES

In order to test the comparisons given in the previous section, in Figure 3 we plot the performance of a rate $R = 1/2$, $(2048; 3; 6)$-LDPC convolutional code with $I = 50$ iterations on an AWGN channel. Also shown is the performance of two $M = 3$, $N = 6$ LDPC block codes with a maximum of 50 iterations. The block lengths were chosen so that in one case the decoders have the same processor complexity, i.e., $L = v_c$, and in the other case the same memory requirements, i.e., $L = v_s \cdot I$. For the same processor complexity, the convolutional code outperforms the block code by about 0.6 dB at a bit error rate of $10^{-5}$. For the same memory requirements, the convolutional and block code performance is nearly identical. LDPC convolutional codes are very efficient for the transmission of streaming data since they allow continuous encoding decoding. However, in some applications, it is preferable to have the data encoded in frames of pre-determined size in order to maintain compatibility with some standard format. Therefore, we now consider the performance of terminated LDPC convolutional codes in this context.

![Fig. 3. BER performance comparison of LDPC block and convolutional codes.](image-url)

The information sequence must be terminated with a tail of symbols to force the encoder to the zero state at the end of the encoding process. For conventional polynomial convolutional encoders, the terminating tail consists of a sequence of zeros. For LDPC convolutional code encoders, the tail is, generally speaking, non-zero and depends on the encoded information bits. Therefore, a system of linear equations must be solved [19].

In Figure 4, we show FER performance comparisons of terminated LDPC convolutional codes versus LDPC block codes, assuming 100 decoding iterations. We terminate a rate $R = 1/2$ $(2048; 3; 6)$ LDPC convolutional code at various frame lengths, resulting in a variety of terminated block lengths and rates. We also provide simulation results for LDPC block codes of comparable block lengths. As shown in Figure 4, a single LDPC convolutional code can be employed to construct a family of codes of varying frame length and error performance via termination. This is an advantage in terms of flexibility compared to LDPC block codes, where a new code must be constructed each time a new transmission frame length is required.

Figure 4 also shows that, even though the LDPC convolutional code with syndrome former memory $m_s = 2048$ has a hardware complexity comparable to that of a length $L = 4096$ LDPC block code, its performance is similar to much longer LDPC block codes. In particular, for a terminated frame length of
L = 64512, the LDPC convolutional code outperforms the LDPC block code of length L = 10000 and performs almost as well as the LDPC block code of length L = 100000.

V. ERASURE CHANNELS WITH MEMORY

We now consider the performance of LDPC-CC ensembles and codes over erasure channels with memory. We consider the familiar two-state Gilbert-Elliott channel (GEC) [32], [33] as a model of an erasure channel with memory. In this model, the channel is either in a “good” state G, where we assume the erasure probability is 0, or in an “erasure” state E, in which the erasure probability is 1. The state process of the channel is a first-order Markov process with the transition probabilities P[G | E]=g and P[G | E]=b. With these parameters, we can easily deduce [34] that the average erasure rate ε and the average burst length Δ are given by

\[ \varepsilon = \frac{P[E]}{b+g}, \Delta = 1/g \]

We will consider the GEC to be parameterized by the pair (ε, Δ). Note that there is a one-to-one correspondence between the two pairs (b, g) and (ε, Δ).

Discussion: The channel capacity of a correlated binary erasure channel with an average erasure rate of \( \varepsilon \) is given as (1- ε), which is the same as that of the memoryless channel, provided the channel is ergodic. Therefore, one can obtain good performance on a correlated erasure channel through the use of a capacity-achieving code for the memoryless channel with an interleaver to randomize the erasures [24], [27]. This is equivalent to permuting the columns of the parity-check matrix of the original code. We are not interested in this approach since such permutations destroy the convolutional structure of the code and as a result, we are unable to use the WD for such a scheme.

Construction of LDPC block codes for bursty erasure channels has been well studied. The performance of a code over a bursty erasure channel is related to the maximum resolvable erasure burst length (MBL) denoted \( \Delta_{\text{max}} \) [27], which, as the name suggests, is the maximal length of a single solid erasure burst that can be decoded by a BP decoder. Methods of optimizing codes for such channels therefore focus on permuting columns of parity-check matrices to maximize \( \Delta_{\text{max}} \). Instead of permuting columns of the parity-check matrix, in order to maintain the convolutional structure of the code, we will consider designing \( C_{\text{rot}}(J;K) \) ensembles that \( \Delta_{\text{max}} \).

A. Asymptotic Analysis

1) BP: As noted earlier, the performance of LDPC-CC ensembles depends on stopping sets. The structure of protographs imposes constraints on the code that limit the stopping set sizes and locations, as will be shown shortly.

Let us define a protograph stopping set to be a subset S(B) of the VN set of the protograph B whose neighboring CNs are connected at least twice to S(B). These are also denoted as S(P), in terms of the set of polynomials defining the protograph. We define the size of the stopping set as the cardinality of S(B), denoted \( |S(B)| \). We call the least number of consecutive columns of B that contain the stopping set S(B) the span of the stopping set, denoted hS(B)i.

Let us denote the size of the smallest protograph stopping set of the protograph B by \( |S(B)|^* \), and the minimum number of consecutive columns of the protograph B that contain a protograph stopping set by \( <S(B)>^* \). When the protograph under consideration is clear from the context, we will drop it from the notation and use \( |S| \) and \( <S> \). The minimum span of a stopping set is of interest because we can give simple bounds for \( \Delta_{\text{max}} \) based on \( <S(B)>^* \). Note that the stopping set of minimal size and stopping set of minimal span are not necessarily the same set of VNs. However, we always have

\[ |S(B)|^* \leq <S(B)>^* \]

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Type</th>
<th>Code Params.</th>
<th>Code rate</th>
<th>Device</th>
<th>Perf.</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9]</td>
<td>Block</td>
<td>(1024,3,6)</td>
<td>0.5</td>
<td>52.5mm² in 0.16µm CMOS</td>
<td>BER=2e⁻⁶@2dB</td>
<td>500MBPS</td>
</tr>
<tr>
<td>[16]</td>
<td>Block</td>
<td>(9216,3,6)</td>
<td>0.5</td>
<td>FPGA</td>
<td>BER=1.0e⁻⁶@2dB</td>
<td>54MBPS</td>
</tr>
<tr>
<td>[12]</td>
<td>Block</td>
<td>(8176,7,154)</td>
<td>0.875</td>
<td>FPGA</td>
<td>N/A</td>
<td>109MBPS</td>
</tr>
<tr>
<td>[10]</td>
<td>Block</td>
<td>(2048,1723)</td>
<td>0.875</td>
<td>17.6mm² in 0.18µm CMOS</td>
<td>BER=5e⁻⁶@6dB</td>
<td>3200MBPS</td>
</tr>
<tr>
<td>[13]</td>
<td>Conv.</td>
<td>(128,3,6)</td>
<td>0.5</td>
<td>FPGA</td>
<td>BER=1e⁻⁷(1e⁻⁷)@2dB</td>
<td>75MBPS(25MBPS)</td>
</tr>
<tr>
<td>[7]</td>
<td>Conv.</td>
<td>(2048,3,6)</td>
<td>0.5</td>
<td>FPGA</td>
<td>BER=1e⁻¹⁰@2dB</td>
<td>2MBPS</td>
</tr>
<tr>
<td>[18]</td>
<td>Conv.</td>
<td>(128,3,6)</td>
<td>0.5</td>
<td>16mm² in 0.18µm CMOS</td>
<td>BER=3e⁻³@3dB</td>
<td>150MBPS</td>
</tr>
</tbody>
</table>
Fig. 4. FER performance comparison of terminated LDPC convolutional and block codes.

Fig. 5. Recovering capability ($\varphi$) of MDS block codes with different rates in terms of the erasure probability of the channel.

Fig. 6. Recovering capability of reverse-MDP convolutional codes with different rates in terms of the erasure probability of the channel.

Fig. 7. Recovering capability of complete-MDP convolutional codes with different rates in terms of the erasure probability of the channel.

VI. COMPARISON BETWEEN MDS BLOCK CODES AND MDP CONVOLUTIONAL CODES

As we have already pointed out through several examples MDP convolutional codes often are capable of decoding more erasures than comparable MDS block codes. In this section we would like to give some theoretical results on the decoding capabilities of (complete) MDP convolutional codes and compare these codes with MDS block codes of the same rate.
VII. CONCLUSIONS

In this paper, we have discuss a comparison of LDPC block and convolutional codes based on several channels which shows some complexity, including computational complexity, hardware complexity, memory requirements, decoding delay, and BER/FER performance. It has been shown via computer simulations that LDPC convolutional codes have an error performance comparable to that of their block code counterparts. In addition, several interesting tradeoffs have been identified between the two different types of codes with respect to VLSI implementation. For erasure channels, while close-to-optimal performance (in the sense of approaching capacity) was achievable for the BEC, we showed that the structure of LDPC-CC imposed constraints that bounded the performance over erasure channels with memory strictly away from the optimal performance (in the sense of approaching MDS performance). Nevertheless, the simple structure and good performance of these codes, as well as the latency flexibility and low complexity of the decoding algorithm, are attractive characteristics for practical systems.

REFERENCES


www.IJETI.com