Convolutional Codes for Iterative Decoding

Lentmaier, Michael; Fettweis, Gerhard; Zigangirov, Kamil; Costello Jr., Daniel J.

Published in:
[Host publication title missing]

DOI:
10.1109/ISSSTA.2008.153

2008

Convolutional Codes for Iterative Decoding

Michael Lentmaier†, Gerhard P. Fettweis†, Kamil Sh. Zigangirov‡ and Daniel J. Costello, Jr.‡
† Vodafone Chair Mobile Communications Systems, Dresden University of Technology, 01062 Dresden, Germany
Emails: {michael.lentmaier, fettweis}@ifn.et.tu-dresden.de
‡ Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
Emails: {Zigangirov.1, Costello.2}@nd.edu

***Abstract*— This paper gives an overview of three different classes of convolutional codes that are suitable for iterative decoding. They are characterized by the type of component codes that are used to construct the overall codes, which can be trivial parity-check constraints, block component codes, or convolutional component codes. Distance properties and iterative decoding performance are considered. All three classes of codes are asymptotically good, allow simple encoding, and can be decoded efficiently using iterative pipeline decoding architectures.**

I. Introduction

Error correcting codes defined by sparse parity-check matrices in combination with iterative decoding have proved to be very efficient in both theory and practice. The most prominent representatives of iteratively decodable codes are the *low-density parity-check* (LDPC) block codes invented by Gallager [1] and the *turbo codes* introduced by Berrou, Glavieux, and Thitimajshima [2]. Also, convolutional codes designed for iterative decoding combine excellent bit error rate (BER) performance with the additional advantage that their structure allows linear time encoding based on shift-registers, a parallel pipeline implementation of iterative decoding, and block transmission in frames of arbitrary size, which makes them attractive for practical implementation.

In this paper, we consider three different classes of convolutional codes. First, the convolutional code counterpart of Gallager's LDPC block codes is described in Section II. Then, in Section III, we introduce convolutional codes with block components, called braided block codes (BBCs). A similar construction based on convolutional components, braided convolutional codes (BCCs), are considered in Section IV. The distance properties and iterative decoding performance of the different code classes are presented in Sections V and VI, respectively. Section VII concludes the paper.

II. LDPC Convolutional Codes

We start with a brief definition of LDPC convolutional codes. Let

$$\boldsymbol{u} = [\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_t, \ldots] \ , \tag{1}$$

where $\boldsymbol{u}_t = u_t^{(1)}, \ldots, u_t^{(b)}$ and $u_t^{(\cdot)} \in \mathrm{GF}(2)$, be the information sequence, which is mapped by a convolutional encoder of rate $R = b/c$, $b < c$, into the code sequence

$$\boldsymbol{v} = [\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_t, \ldots] \ , \tag{2}$$

where $\boldsymbol{v}_t = v_t^{(1)}, \ldots, v_t^{(c)}$ and $v_t^{(\cdot)} \in \mathrm{GF}(2)$. A time-varying LDPC convolutional code can be defined as the set of sequences $\boldsymbol{v}$ satisfying the equation $\boldsymbol{v}\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{0}$, where

$$\boldsymbol{H}^{\mathrm{T}} = \begin{bmatrix} \ddots & & \ddots & & \\ \boldsymbol{H}_0^{\mathrm{T}}(0) & \ldots & \boldsymbol{H}_{m_{\mathrm{s}}}^{\mathrm{T}}(m_{\mathrm{s}}) & & \\ & \ddots & & \ddots & \\ & & \boldsymbol{H}_0^{\mathrm{T}}(t) & \ldots & \boldsymbol{H}_{m_{\mathrm{s}}}^{\mathrm{T}}(t+m_{\mathrm{s}}) \\ & & \ddots & & \ddots \end{bmatrix} \tag{3}$$

is an infinite transposed parity-check matrix, also called a *syndrome former*. The elements $\boldsymbol{H}_i^{\mathrm{T}}(t)$, $i = 0, 1, \ldots, m_{\mathrm{s}}$, in (3) are binary $c \times (c-b)$ submatrices

$$\boldsymbol{H}_i^{\mathrm{T}}(t) = \begin{bmatrix} h_i^{(1,1)}(t) & \cdots & h_i^{(1,c-b)}(t) \\ \vdots & & \vdots \\ h_i^{(c,1)}(t) & \cdots & h_i^{(c,c-b)}(t) \end{bmatrix} \ , \tag{4}$$

where $\boldsymbol{H}_{m_{\mathrm{s}}}^{\mathrm{T}}(t) \neq 0$ for at least one $t \in \mathbb{Z}$ and $\boldsymbol{H}_0^{\mathrm{T}}(t)$ has full rank for all t. We call m_{s} the syndrome former memory and $\nu_{\mathrm{s}} = (m_{\mathrm{s}}+1) \cdot c$ the associated constraint length. These parameters determine the span of the nonzero diagonal region of $\boldsymbol{H}^{\mathrm{T}}$. Sparsity of the syndrome former is ensured by demanding that the Hamming weights of its columns are much smaller than ν_{s}. The code is said to be regular if its syndrome former $\boldsymbol{H}^{\mathrm{T}}$ has exactly J ones in every row and K ones in every column, starting from the $[m_{\mathrm{s}} \cdot (c-b) + 1]$-th column. The other entries are zeros. We will refer to a code with these properties as an (m_{s}, J, K)-LDPC convolutional code.

An (m_{s}, J, K)-LDPC convolutional code is called periodic with period T if $\boldsymbol{H}_i^{\mathrm{T}}(t) = \boldsymbol{H}_i^{\mathrm{T}}(t+T)$ $\forall t, i$. A description of the construction of a periodic syndrome former $\boldsymbol{H}^{\mathrm{T}}$ from the transposed parity-check matrix of an LDPC block code is given in [3].

A. Efficient Encoding

The system of parity-check equations

$$\boldsymbol{v}_t\boldsymbol{H}_0^{\mathrm{T}}(t) + \boldsymbol{v}_{t-1}\boldsymbol{H}_1^{\mathrm{T}}(t) + \cdots + \boldsymbol{v}_{t-m_{\mathrm{s}}}\boldsymbol{H}_{m_{\mathrm{s}}}^{\mathrm{T}}(t) = \boldsymbol{0}, \ t \in \mathbb{Z} \tag{5}$$

can be used to define an encoder based on the syndrome former $\boldsymbol{H}^{\mathrm{T}}$. A systematic encoder can be obtained if the last $c-b$ rows of the submatrices $\boldsymbol{H}_0^{\mathrm{T}}(t)$ are linearly independent and the first b symbols of $\boldsymbol{v}_t$ coincide with the information symbols,

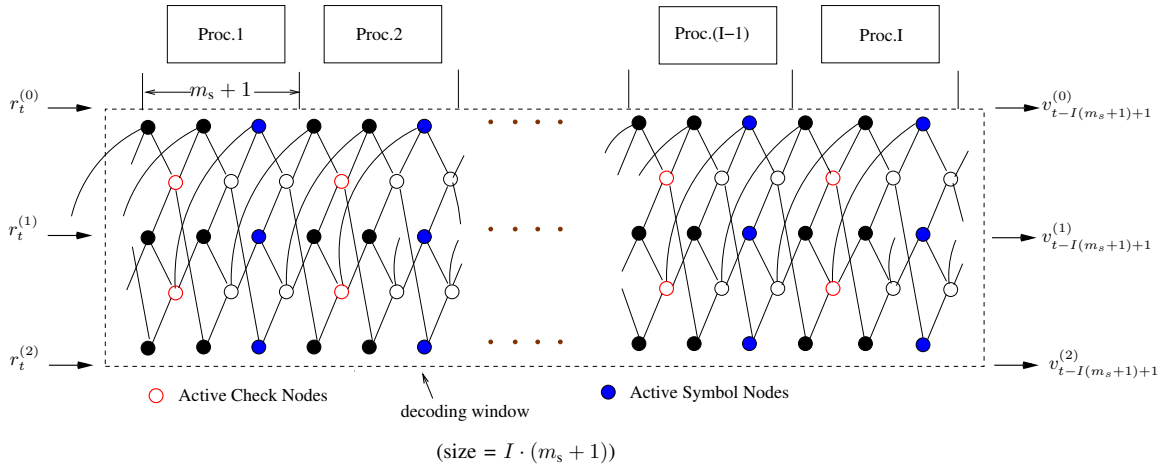

Fig. 1. Tanner graph of an $R = 1/3$ LDPC convolutional code and an illustration of pipeline decoding.

i.e., $v_t^{(j)} = u_t^{(j)}$, $j = 1, \ldots, b$. Consider, for example, the case where the last $c - b$ rows of $\boldsymbol{H}_0^{\mathrm{T}}(t)$ are equal to the $(c - b) \times (c - b)$ identity matrix. Then the code symbols at time t are determined by

$$v_t^{(j)} = u_t^{(j)}, \quad j = 1, \ldots, b \ , \tag{6}$$

$$v_t^{(j)} = \sum_{k=1}^{b} v_t^{(k)} h_0^{(k,j-b)}(t) + \sum_{i=1}^{m_s} \sum_{k=1}^{c} v_{t-i}^{(k)} h_i^{(k,j-b)}(t) \ , \quad j = b+1, \ldots, c \ . \tag{7}$$

This encoder can be implemented by a length $cm_s + b$ shift register with time-varying tap-weights corresponding to the matrix entries $h_i^{(k,j)}(t)$ [3]. The encoder realization requires $c \cdot m_s + b$ memory units and the encoding complexity per bit is proportional to the associated column weight of $\boldsymbol{H}^{\mathrm{T}}$ (i.e., K for regular codes), independent of the codeword length and the syndrome former memory m_s. A realization based on partial syndromes, requiring only $(c - b) \cdot m_s$ memory units, has been proposed in [4]. It also provides a convenient way for the computation of termination bits for LDPC convolutional codes.

B. Pipeline Decoding

Although the Tanner graph corresponding to $\boldsymbol{H}^{\mathrm{T}}$ 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. The decoding of two variable nodes that are at least $(m_s + 1)$ 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, a single "looping processor" that runs on different regions of the decoder memory successively can also be employed. Based on these ideas, a pipeline decoding architecture, which outputs a continuous stream of decoded data once an initial decoding delay has elapsed, was introduced in [3]. The operation of this decoder on the Tanner graph for a simple rate $R = 1/3$ LDPC convolutional code with $m_s = 2$ is shown in Fig. 1.

III. Braided Block Codes

A. Tightly Braided Block Codes

Tightly braided block codes (TBBCs) are convolutional codes that are composed of block component codes and can be described by means of an infinite array. Every symbol stored in the array is protected by a horizontal code C^{h} and a vertical code C^{v}. The symbols in each row of the array form a code word of C^{h}, the symbols in each column a code word of C^{v}. The array representation of a TBBC based on $(7, 4)$ Hamming component codes is illustrated in Fig. 2.

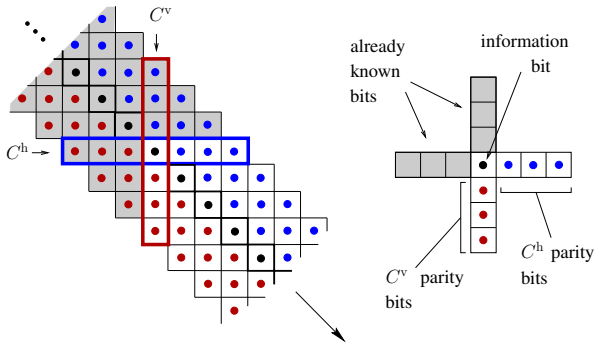

Fig. 2. Array representation of tightly braided block codes with (7,4) Hamming component codes.

At each time t an information symbol enters the main diagonal of the array. The shaded area to the left and above that symbol corresponds to symbols that have already been encoded. The component encoders use the shaded symbols of row t and column t together with the current information

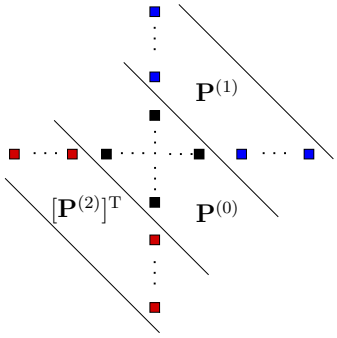

Fig. 3. Array representation of sparsely braided codes.

symbol to produce the parity check symbols of row t and column t, respectively.

B. Sparsely Braided Block Codes

In general, the cells within the array of TBBCs can be divided into three ribbons, corresponding to vertical parity symbols, information symbols, and horizontal parity symbols, respectively. In a sparsely braided block code (SBBC) the symbols within these ribbons are not adjacent but spread apart from each other. The number of symbols per row and column of each ribbon are preserved but the array is sparse. The positions of the symbols in the ribbons can be described by multiple convolutional permutors $[\mathbf{P}^{(2)}]^{\mathrm{T}}$, $\mathbf{P}^{(0)}$, and $\mathbf{P}^{(1)}$ [5], as illustrated in Fig. 3.

The idea of generalizing LDPC codes to non-trivial component block codes goes back to Tanner [6]. SBBCs can be interpreted as generalized LDPC convolutional codes, the convolutional version of generalized LDPC (GLDPC) codes. Their encoding can be implemented either directly from the array representation, similar to the description above for TBBCs, or by means of a partial syndrome realization [5]. Iterative decoding of SBBCs can be performed with a pipeline architecture, analogous to that of LDPC convolutional codes [5]. The messages from constraint nodes to variable nodes are defined by the outputs of two a posteriori probability (APP) component decoders, as in the decoding of GLDPC codes [7]. As shown in [8], the asymptotic bit error probability converges to zero at least double exponentially with the number of iterations if the condition $(d_{\min}-1)(J-1) > 1$ is satisfied. This illustrates that the number of component codes per symbol can be traded against their strength and shows that a generalization of the array to more than two dimensions is not required (see also Section V).

IV. Braided Convolutional Codes

The concept of braided codes can also be applied to convolutional component codes. The resulting BCCs can again be described by an infinite array consisting of three ribbons, as shown in Fig. 3. The implementation of a rate $R = 1/3$ BCC encoder, consisting of two rate $R_{\mathrm{cc}} = 2/3$ recursive systematic component encoders, is illustrated in Fig. 4. The information symbols u_t enter the first input of Encoder 1 directly, and the permuted information symbol $\tilde{u}_t$ at the output of convolutional permutor $\mathbf{P}^{(0)}$ enters the first input of Encoder 2. Encoders 1 and 2 generate the parity symbols $\hat{v}_t^{(1)}$ and $\hat{v}_t^{(2)}$, respectively. The permuted parity symbol $\tilde{v}_t^{(1)}$ at the output of convolutional permutor $\mathbf{P}^{(1)}$ is fed back to the second input of Encoder 2, and the permuted parity symbol $\tilde{v}_t^{(2)}$ at the output of convolutional permutor $\mathbf{P}^{(2)}$ is fed back to the second input of Encoder 1.

A special property of BCCs is that both information and parity bits are connected to each of the component codes in a symmetric manner. This feature makes them more similar to (G)LDPC codes than other turbo-like constructions.

The decoding of BCCs is comparable to continuous turbo decoding. In a pipeline implementation of the decoder [9], the component codes are decoded using a parallel bank of $2I$ APP processors based on a windowed BCJR algorithm [10]. APP values are calculated for all the code symbols, not only for the information symbols.

It is also possible to perform encoding and decoding in a block-wise manner. In this case, information and parity symbols are collected in vectors of size N, and $\mathbf{P}^{(0)}$, $\mathbf{P}^{(1)}$, and $\mathbf{P}^{(2)}$ now denote block permutors of size N rather than convolutional permutors. The component encoders are then rate $R = 2/3$ tail-biting convolutional encoders that start from and end in the same state. Windowed decoding is no longer required, and the processing of each block is similar to that of a conventional turbo code.

Another approach to constructing braided convolutional codes from block permutations has been proposed in [11]. Here the ones in the syndrome former of a tightly braided base code are replaced by permutation matrices. These codes can be interpreted as convolutional variants of protograph-based LDPC codes [12] and decoded either like Gallager's LDPC codes using belief propagation or with a turbo-like algorithm as described above.

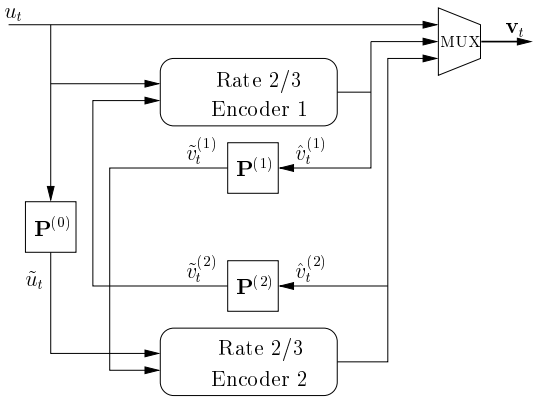

Fig. 4. Encoder for a rate $R = 1/3$ braided convolutional code.

V. Distance Properties

Gallager showed that random regular LDPC block codes with $J > 2$ are asymptotically good, i.e., the minimum distance $d_{\min}$ of randomly chosen codes grows linearly with block length [1]. For GLDPC block codes, this is also true

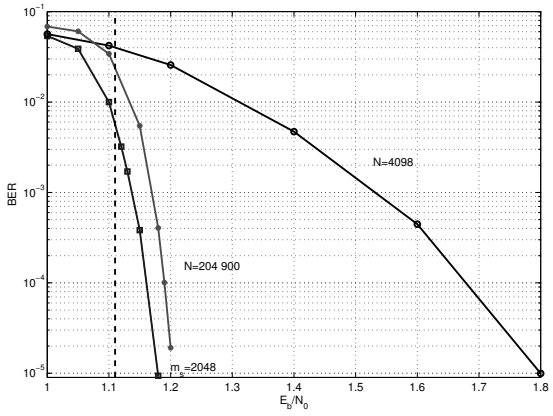

Fig. 5. Simulation results for a (2048,3,6) LDPC convolutional code in comparison with a (4096,3,6) LDPC block code and a (204900,3,6) LDPC block code on the AWGN channel [4].

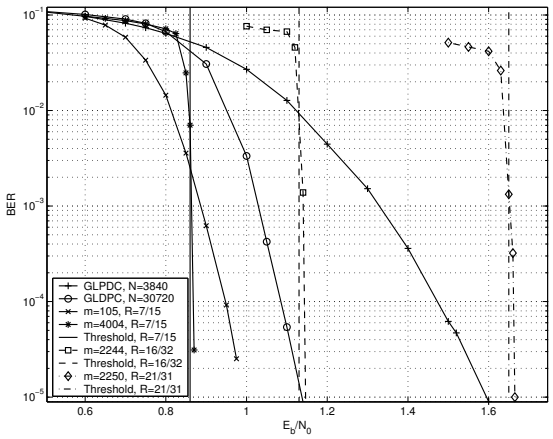

Fig. 6. Simulation results for continuous SBBCs on the AWGN channel [5].

for the case of $J = 2$ component codes per symbol [13]. For (multiple) turbo codes, on the other hand, $d_{\min}$ grows less than linearly with block length, even if $J > 2$ component codes are used [14]. Results on the distance properties of the convolutional code classes considered in this paper are summarized in the following.

A. Markov Permutor Ensembles

A technique to derive the average distance spectrum of LDPC convolutional codes has been introduced in [15]. It can be applied to all codes described by (multiple) convolutional permutors. A random ensemble is defined by replacing the convolutional permutors by a stochastic device called a Markov permutor. From the average distance spectrum it is possible to compute lower bounds on the free distance d_{free} of codes from an ensemble as a function of their constraint length ν_s.

The distance growth rates observed for regular LDPC convolutional codes [15] and BBCs [5] are comparable to their block code variants: d_{free} grows linearly with ν_s and the distance ratios d_{free}/ν_s are typically worse than the Costello bound for random convolutional codes [16] but better than the ratio $d_{\min}/N$ of the corresponding block codes of length N.

Remarkable is the fact that linear growth for d_{free} can also be observed for BCCs [9], which distinguishes them from other turbo-like constructions. For the self-concatenated convolutional codes considered in [15], which are closely related to multiple turbo codes, d_{free} grows less than linearly with constraint length. The superior asymptotic distance behavior of BCCs may be related to their similarity to GLDPC codes.

B. Permutation Matrix Based Ensembles

For the Markov permutor ensembles, a bound on d_{free} is evaluated for different constraint lengths to estimate the asymptotic distance ratios. An explicit lower bound on the asymptotic free distance of LDPC convolutional codes has been presented in [17]. The codes in the considered ensemble are defined by syndrome formers that are composed of permutation matrices, like in protograph-based LDPC codes [12]. The technique from [17] has also been applied in [18] to compute distance ratios for the protograph-based BCCs considered in [11]. It can also be used for tail-biting variants of LDPC convolutional codes [19] and BCCs [18].

VI. Iterative Decoding Performance

A. LDPC Convolutional Codes

Simulated BERs of a rate $R = 1/2$, $(2048, 3, 6)$-LDPC convolutional code with $I = 50$ iterations on an AWGN channel are shown in Fig. 5. Also shown is the performance of two $J = 3$, $K = 6$ 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., $N = \nu_s$, and in the other case the same memory requirements, i.e., $N = \nu_s \cdot I$ [4]. 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.

The vertical dashed line corresponds to the convergence threshold of the block codes. From the graph structure of LDPC convolutional codes it follows that they also can achieve this threshold. Actually, it can be shown [20] that terminated LDPC convolutional codes have better thresholds than the block codes they are derived from, since the symbols close to the start and end of the block have stronger protection.

B. Braided Block Codes

Simulation results of continuous SBBCs after $I = 50$ iterations are shown in Fig. 6. The BERs are shown for rate $R = 7/15$, $16/32$, and $21/31$ SBBCs based on Hamming component codes of length 15, 31, and 32. The curves indicate a waterfall effect very close to the convergence thresholds of the respective GLDPC block codes. For the codes with $(15, 11)$ component codes, the BERs of two rate $R = 1/2$ GLDPC block codes with block lengths $N = 3840$ and $N = 30720$ are shown for comparison [7].

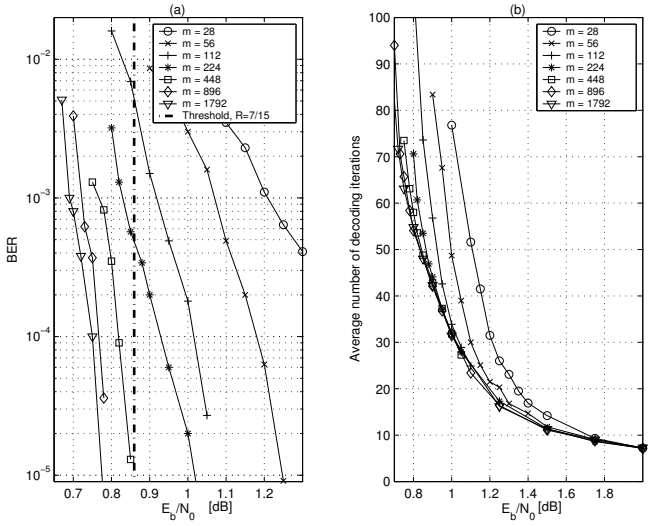

Fig. 7. Simulation results for ideally terminated BBBCs on the AWGN channel [5]. a) Bit error rate. b) Average number of iterations.

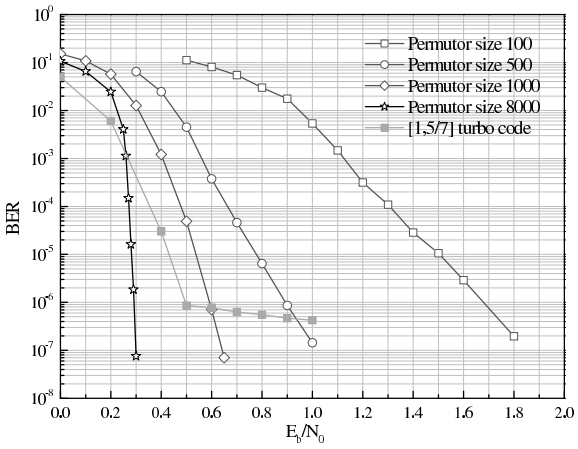

Fig. 8. Error performance of rate $R = 1/3$ blockwise BCCs and turbo codes on the AWGN channel [9].

Fig. 7-a) shows the results for ideally terminated blockwise BBCs using $(15, 11)$-Hamming component codes. The length of the block is $292m$, including a tail of $8m$ information symbols. The effective code rates in all cases are $R = 0.45$. Fig. 7-b) shows the average number of iterations performed before the decoder converged to a valid code word. From these curves, we see that the convergence threshold for GLDPC block codes is surpassed at a BER level of 10^{-5} for $m \geq 448$. This suggests that blockwise BBCs also have better thresholds than their GLDPC block code counterparts.

C. Braided Convolutional Codes

The BER performance of rate $R = 1/3$ blockwise BCCs is shown in Figure 8 for permutor sizes $N = 100$ to 8000. The tail-biting versions of rate $R_{\text{cc}} = 2/3$ component encoders with polynomial parity-check matrix $\mathbf{H}(D) = [1, 1 + D^2, 1 + D + D^2]$ were employed and the three block permutors used in the encoder were chosen randomly. The transmission of 50 information blocks was terminated with 2 all-zero blocks. The blockwise BCCs are compared to a rate $R = 1/3$ turbo code with 4-state $[1, 5/7]$ (octal format) component encoders and permutor size $N = 8192$. In contrast to the turbo code, no error floor can be observed for any of the blockwise BCCs, which indicates their superior distance properties.

VII. Conclusions

In this paper we have presented three different classes of convolutional codes for iterative decoding based on various component codes. LDPC convolutional codes and braided block codes are counterparts of Gallager's LDPC block codes and Tanner's generalization to arbitrary component codes. Braided convolutional codes, like turbo codes, are based on convolutional component codes, but their construction is more similar to (G)LDPC codes than other turbo-like codes. All the presented codes are asymptotically good, resulting in low error floors, and their convergence thresholds under iterative decoding are at least as good as those of the block codes they are derived from.

References

[1] R. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, MA, 1963.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Communications*, Geneva, Switzerland, May 1993, vol. 2, pp. 1064–1070.

[3] A. Jiménez Feltström and K.Sh. Zigangirov, "Periodic time-varying convolutional codes with low-density parity-check matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 2181–2190, Sept. 1999.

[4] A.E. Pusane, A. Jiménez Feltström, A. Sridharan, M. Lentmaier, K. Sh. Zigangirov, and D.J. Costello, Jr., "Implementation aspects of LDPC convolutional codes," to appear in *IEEE Trans. Commun.*, July 2008.

[5] A. Jiménez Feltström, M. Lentmaier, D.V. Truhachev, and K.Sh. Zigangirov, "Braided block codes," submitted to *IEEE Trans. Inf. Theory*, June 2006.

[6] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 9, pp. 533–547, Sept. 1981.

[7] M. Lentmaier and K. Sh. Zigangirov, "On generalized low-density parity-check codes based on Hamming component codes," *IEEE Communications Letters*, vol. 3, no. 8, pp. 248–250, Aug. 1999.

[8] M. Lentmaier, D.V. Truhachev, K.Sh. Zigangirov, and D.J. Costello, Jr., "An analysis of the block error probability performance of iterative decoding," *IEEE Trans. Inf. Theory*, vol. 51, no. 11, pp. 3834–3855, Nov. 2005.

[9] W. Zhang, , M. Lentmaier, K.Sh. Zigangirov, and D.J. Costello, Jr., "Braided convolutional codes: a new class of turbo-like codes," submitted to *IEEE Trans. Inf. Theory*, May 2006.

[10] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[11] M.B.S. Tavares, K.Sh. Zigangirov, and G.P. Fettweis, "LDPC convolutional codes based on braided convolutional codes," to appear in *Proc. IEEE Int. Symp. Information Theory (ISIT'08)*, July 2008.

[12] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," in *IPN Progress Report 42-154, JPL*, Aug. 2005.

[13] M. Lentmaier and K. Sh. Zigangirov, "Iterative decoding of generalized low-density parity-check codes," in *Proc. IEEE Int. Symp. Information Theory*, Boston, USA, August 1998, p. 149.

[14] N. Kahale and R. Urbanke, "On the minimum distance of parallel and serially concatenated codes," submitted to *IEEE Trans. Inf. Theory*.

[15] K. Engdahl, M. Lentmaier, and K. Sh. Zigangirov, "On the theory of low-density convolutional codes," *Lecture Notes in Computer Science (AAECC-13)*, vol. 1719, pp. 77–86, Springer-Verlag, New York 1999.

[16] D. J. Costello, Jr., "Free distance bounds for convolutional codes," *IEEE Trans. Inf. Theory*, vol. IT-20, pp. 356–365, July 1974.

[17] A. Sridharan, D.V. Truhachev, M. Lentmaier, D.J. Costello, Jr., and K. Sh. Zigangirov, "Distance bounds for an ensemble of LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, Dec. 2007.

[18] M.B.S. Tavares, M. Lentmaier, K.Sh. Zigangirov, and G.P. Fettweis, "Analysis of braided protograph convolutional codes," in preparation.

[19] D.V. Truhachev, K.Sh. Zigangirov, and D.J. Costello, Jr., "Distance bounds for periodically time-varying and tail-biting LDPC convolutional codes," submitted to *IEEE Trans. Inf. Theory*, Dec. 2007.

[20] M. Lentmaier, A. Sridharan, K. Sh. Zigangirov, and D. J. Costello, Jr., "Terminated LDPC convolutional codes with thresholds close to capacity," in *Proc. IEEE Int. Symp. Information Theory*, Adelaide, Australia, Sept. 2005, pp. 1372–1376.