New Perspectives on Braided Convolutional Codes

Costello, Jr., Daniel J.; Lentmaier, Michael; Mitchell, David G. M.

Published in:
Proc. 9th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)

DOI:
10.1109/ISTC.2016.7593145

2016

New Perspectives on Braided Convolutional Codes

Daniel J. Costello, Jr.*, Michael Lentmaier†, and David G. M. Mitchell‡

*Dept. of Electrical Engineering, University of Notre Dame, Notre Dame, USA, costello.2@nd.edu
†Dept. of Electrical and Information Technology, Lund University, Lund, Sweden, Michael.Lentmaier@eit.lth.se
‡Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, USA, dgmm@nmsu.edu

***Abstract*—Braided convolutional codes (BCCs) are a type of parallel-concatenated convolutional code in which the parity outputs of one component encoder are fed back and used as inputs to the other component encoder at the succeeding time unit. In this paper, we review the published results on BCCs that have appeared in the literature over the past ten years and present a unified view of BCCs in the context of other types of turbo-like codes. We also include some recent results on iterative decoding thresholds for BCCs, weight enumerators and distance growth rates, window decoding for low latency operation, and rate-compatible BCCs for high-rate applications.**

I. Introduction

Braided codes [1] were originally introduced as a convolutional counterpart of product codes [2]. Their characteristic feature is that the parity symbols of one component encoder are used as information symbols of the other and vice versa. As a result, both information and parity symbols are protected by both component codes in a symmetric fashion. In this paper we consider as an example *braided convolutional codes (BCCs)* [3], [4] of rate $R = 1/3$, which are defined by means of two systematic component convolutional encoders of rate $R_{\text{cc}} = 2/3$ and three permutors of length N. In particular, our focus will be on *blockwise BCCs*, for which an encoder diagram is shown in Fig. 1. The parity symbols created by one encoder at time t pass a delay of one block, D^N, and a permutor before entering the other encoder at time $t + 1$.

Two other variations of BCCs have been considered in [4]. *Tightly BCCs* are obtained if we reduce the blocksize to $N = 1$ and the permutations become obsolete. This construction is deterministic and simple but performs worse due to the absence of permutors, which improve both the sparseness and the strength of the codes at the cost of an increased delay. Alternatively, we can set the blocksize to $N = 1$ but replace the block permutors with convolutional permutors that span several time instants. The strength and sparseness of the resulting *bitwise BCCs* are then determined by the maximum delay, or the overall constraint length M, of the convolutional permutors. Tavares et al. have introduced a variation of BCCs called braided protograph convolutional codes [5], [6]. Here the parity-check matrix of tightly BCCs is used to define the protograph of an LDPC convolutional code. Since the structure of the component encoders is maintained in the protograph, a trellis-based iterative decoding is possible.

Blockwise BCCs have the advantage that both the encoding and decoding can be performed in a fashion similar to turbo codes [7], [8], for which efficient hardware implementations already exist and can be reused. It can be seen from the encoder diagram in Fig. 1 that BCCs are closely related to turbo codes. If the parity feedback is removed completely, the encoder is equivalent to parallel concatenated codes (PCCs). On the other hand, if the parity feedback is removed from one of the decoders only and the delay D^N from the other, the resulting encoder is equivalent to serial concatenated codes (SCCs) with systematic encoding. This shows that BCCs combine some features of both PCCs and SCCs.

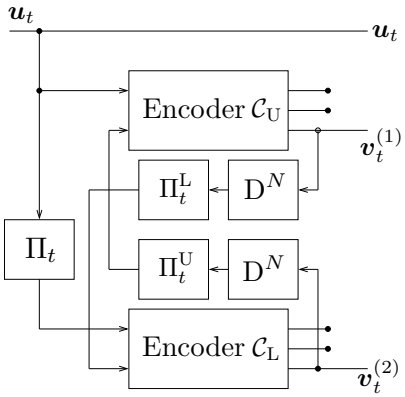


Fig. 1: Blockwise BCCs: turbo-like codes with parity feedback ($R = 1/3$).

It follows from the delay in the parity feedback of the BCC encoder that blocks which are encoded at consecutive time instants are interconnected. This is illustrated in Fig. 2 by depicting a chain of encoders that operate at different time instants. Because of the interconnection between blocks, we can see that BCCs inherently form a class of spatially coupled (SC) codes.

For comparing BCCs with PCCs and SCCs it sometimes can be useful to define an uncoupled equivalent of BCCs. This can be achieved by removing the delay in the encoder depicted in Fig. 1. Since the feedback now occurs without a delay, a straightforward encoding by means of the component encoders is no longer possible. But the code is still well-defined by the trellis constraints that the code symbols have to satisfy. The resulting *uncoupled BCCs* can be interpreted as a tailbiting version of the original BCCs. In the same way as we can generate a convolutional code from a block code by means of spatial coupling, we can obtain a block code from a convolutional code by means of tailbiting.

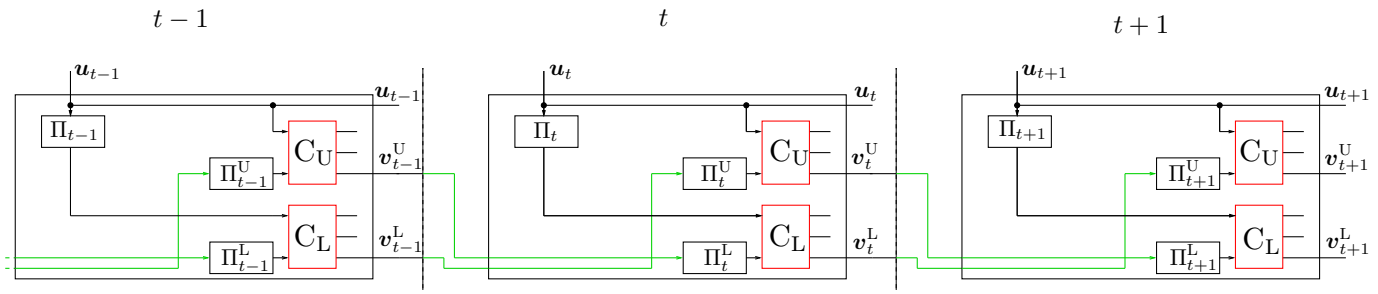


Fig. 2: Encoder chain of blockwise BCCs with inherent coupled structure.

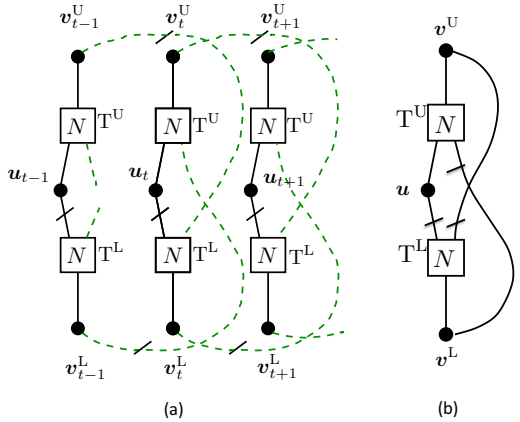


Fig. 3: Compact graph representation of (a) BCCs and (b) their uncoupled equivalent.

II. Iterative Decoding Thresholds

Turbo-like codes, like LDPC codes, can be described by means of factor graphs. Such a representation is useful for describing the exchange of messages in an iterative belief propagation (BP) decoder, as well as for the corresponding density evolution analysis. Instead of a conventional factor graph, we use a compact graph representation as introduced by Moloudi et al. in [9]. The compact graph of a BCC and its uncoupled equivalent is shown in Fig. 3. Each block of symbols is represented by a variable node and each trellis by a factor node. A permutor is indicated by a short line that crosses an edge in the graph.

From the graphs in Fig. 3 we can see the analogy between BCCs and SC-LDPC codes. A spatially coupled code can be obtained by repeating the graph of an uncoupled code and spreading some edges across $m+1$ neighboring blocks, where m denotes the coupling memory. In the same way it is possible to apply spatial coupling to PCCs and SCCs [10]. While the original BCCs shown in Fig. 3(a) have coupling memory $m = 1$, a generalization to $m > 1$ is possible [11].

Table I shows the thresholds of SC-PCCs, SC-SCCs, and BCCs, resulting from a density evolution analysis for the BEC [9]. All component encoders have memory $m_{\mathrm{cc}} = 2$, and the overall code rate $R = 1/2$ is achieved by puncturing. The BCC Type-I ensemble is the original ensemble as introduced in [4], but generalized to larger coupling memories. The BCC Type-II ensemble is a modified version of BCCs, introduced in [11], in which not only the parity symbols but also the information symbols are coupled.

TABLE I: Thresholds for SC-PCCs, SC-SCCs, and BCCs [9]

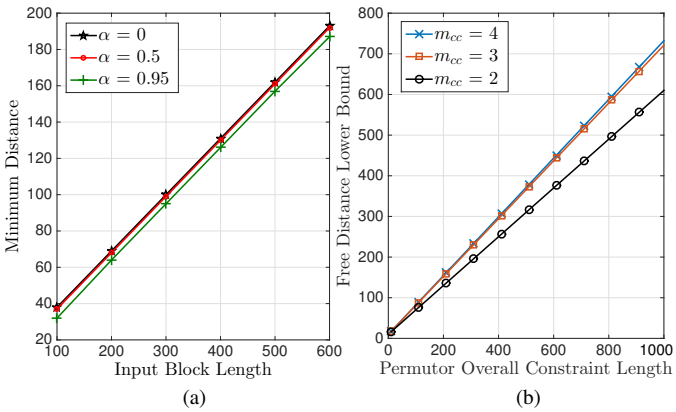
Ensemble	Rate	$\varepsilon_{\mathrm{BP}}$	$\varepsilon_{\mathrm{MAP}}$	ϵ_{SC}		
				$m=1$	$m=3$	$m=5$
SC-PCC	1/2	0.4606	0.4689	0.4689	0.4689	0.4689
SC-SCC	1/2	0.3594	0.4981	0.4708	0.4975	0.4981
BCC Type-I	1/2	0.3013	0.4993	0.4932	0.4980	0.4988
BCC Type-II	1/2	0.3013	0.4993	0.4988	0.4993	0.4993

These results show that, without spatial coupling, PCCs have the best BP threshold $\varepsilon_{\mathrm{BP}}$, BCCs have the best MAP threshold $\varepsilon_{\mathrm{MAP}}$, and the thresholds of SCCs lie in between. We can conclude from this observation that optimizing component codes for iterative decoding does not necessarily optimize the strength of the resulting overall code. With spatial coupling the BP thresholds ϵ_{SC} improve and, for large enough coupling memory, threshold saturation to the MAP threshold is observed, which also can be proved analytically [9]. For Type-II BCCs, with coupling of information symbols, the thresholds improve faster with m. Furthermore, we can see that for any given m the coupled BP thresholds ϵ_{SC} of BCCs are superior to PCCs or SCCs.

Without spatial coupling, it is well known that PCCs yield good BP thresholds but poor error floors, while SCCs show low error floors but poor BP thresholds. For this reason it is interesting to analyze the distance properties of BCCs.

III. Distance Properties

Consider the ensemble of uncoupled BCCs, defined by the compact graph in Fig. 3(b) and a given pair of component encoders. The finite block-length ensemble weight enumerator function (WEF) of this ensemble has been derived in [12], considering uniform random permutors. A lower bound on the minimum distance of codes in the ensemble, which can be derived from the average WEF, is shown in Fig. 4(a) as function of the information block length N. For a given parameter $\alpha < 1$, a fraction α of the codes in the ensemble must have a minimum distance that is larger or equal to this bound. The results show that the minimum distance of BCCs grows linearly with the block length, unlike PCCs or SCCs. For larger values of α the bound gets only slightly weaker,



(a) (b)

Fig. 4: (a) Lower bound on the minimum distance for the uncoupled BCC ensemble as a function of the input block length N ($m_{cc} = 2$) [12], and (b) lower bounds on the free distance of bitwise BCCs with different component encoders [4].

which indicates that a majority of codes in the ensemble have large minimum distance. It is also shown in [12] that every valid code sequence of a coupled code can be mapped into a codeword of equal or lower weight in the uncoupled equivalent code. It follows that the free distance of BCCs is lower bounded by the minimum distance of the uncoupled code and thus grows linearly with the permutor size according to the bound in Fig. 4(a).

An expurgated ensemble can be defined by excluding the fraction $1 - \alpha$ of codes with lowest minimum distance. The error floor of this expurgated ensemble can be estimated by the truncated union bound. It is demonstrated in [12] that very low error floors can be achieved this way for moderate block lengths.

A lower bound on the free distance d_{free} was also derived in the paper by Zhang et al. [4] for an ensemble of bitwise BCCs, with block length $N = 1$ and convolutional permutors of constraint length M. The analysis of this ensemble was simplified by using the concept of a *Markov permutor*, a statistical device first introduced in [13] to analyze the distance properties of LDPC convolutional codes. Since these bitwise BCCs are *convolutional* codes, the relevant measure of distance growth is the *free distance to constraint length ratio* $\delta_{\text{free}} = d_{\text{free}}/M$.

Using this model, the authors were able to numerically calculate a bound on d_{free} as a function of M for constraint lengths as large as $M = 1000$ for $R = 1/3$ BCCs with $R_{\text{cc}} = 2/3$ component encoders with memory $m_{\text{cc}} = 2, 3$, and 4, i.e., for 4-state, 8-state, and 16-state component encoders, respectively. The results are shown in Fig. 4(b), where we note that, like the linear growth of d_{min} with N for uncoupled BCCs, d_{free} is observed to grow linearly with M for coupled BCCs. The resulting free distance growth rates, along with the generator matrices of the component encoders (in octal notation), are given in Table II. It is interesting to note that these growth rates are approximately half of the asymptotic growth rate $\delta_{\text{free}} = 1.3028$ of the general ensemble of all rate $R = 1/3$ convolutional codes [14]. In other words, with the use of very simple component encoders and correspondingly low-complexity decoders, BCCs are roughly half as strong as convolutional codes with arbitrarily large encoding and decoding complexity. This result suggests again that BCCs will perform very well in the error floor, thus circumventing a weakness of conventional PCCs.

TABLE II: Free distance growth rates for rate $R = 1/3$ bitwise BCCs with different component encoders [4].

Component encoder memory	Generator matrix	Asymptotic ratio δ_{free}
$m_{\text{cc}} = 2$	$\begin{pmatrix} 1 & 0 & 4/7 \\ 0 & 1 & 5/7 \end{pmatrix}$	0.6069
$m_{\text{cc}} = 3$	$\begin{pmatrix} 1 & 0 & 17/15 \\ 0 & 1 & 13/15 \end{pmatrix}$	0.7230
$m_{\text{cc}} = 4$	$\begin{pmatrix} 1 & 0 & 25/35 \\ 0 & 1 & 23/35 \end{pmatrix}$	0.7341

IV. Windowed Decoding of BCCs

Due to the convolutional structure of BCCs, both encoding and decoding can be performed in a continuous fashion. By means of window decoding the performance, latency, storage and computational complexity can be made independent of the number of blocks in the coupled sequence.

In the paper by Zhang et al. [4], a *pipeline decoder* architecture was proposed, for which a predefined number I of decoding iterations can be performed in parallel by independent identical processors. This set of processors operates within a window of $w = I$ blocks, as illustrated in Fig. 5. When a new block with time index $t + w - 1$ is received, it enters the first processor at the right hand side of the decoding window. Then each processor performs a single round of *forward-backward BCJR decoding* [15] on the two component codes and passes the results to the next processor. The leftmost block with time index t has now completed all I iterations and thus leaves the decoding window, which is shifted one block to the right. The N information symbols of this decoded block are called the *target symbols*.

The results of this pipeline decoder can be shown to be equivalent to a conventional decoder that operates on the entire sequence of blocks using a flooding schedule with I decoding iterations. After an initial delay of I blocks, decoded blocks are produced continuously, making such an arrangement well suited for streaming applications. Very high decoding speeds are feasible, since each processor performs a single iteration only and the processors operate in parallel.

On the other hand, if a large number of iterations is required to obtain the desired level of performance, the initial delay becomes large, which can cause problems in delay-constrained applications. Also, since the pipeline architecture fixes the number of iterations, stopping rules cannot be applied to the BCJR decoders, which are often desirable for reducing computational complexity. Finally, it can be observed that a flooding schedule is not well-suited for decoding SC codes, for which other window decoding schedules turn out to be more efficient [17], [18].

To overcome these limitations of the pipeline decoder, Zhu et al. [16], [19] recently introduced a *sliding window decoder*

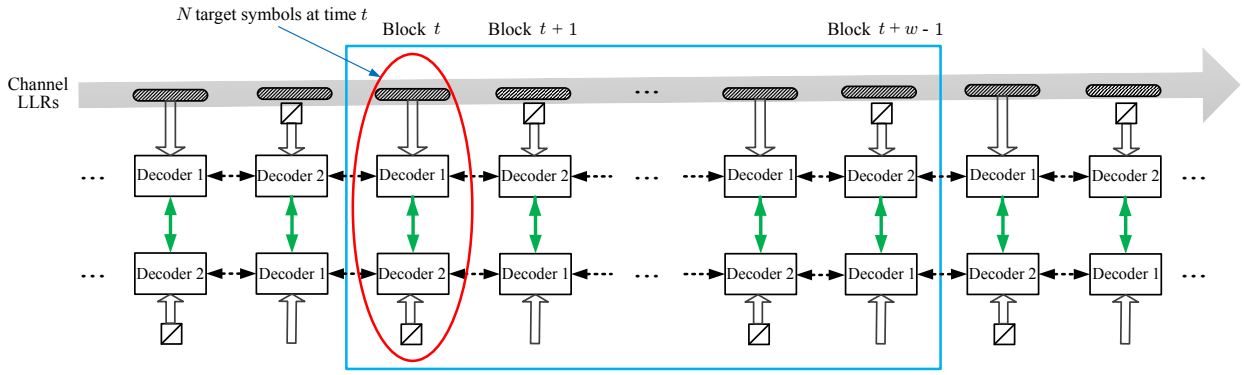


Fig. 5: Continuous decoding of BCCs via pipeline decoding ($w = I$) [4] or sliding window decoding [16].

architecture for BCCs, analogous to the sliding window decoding methods that have been employed to decode SC LDPC codes [20], [21], [22]. Briefly, the sliding window decoder concept allows low-latency operation with negligible performance loss, the use of stopping rules to limit computational complexity, and the ability to traverse the decoding window in both a forward and backward direction, resulting in improved performance compared to pipeline decoding. Like with the pipeline decoder, in decoding the block received at time t, the window contains the blocks received at times $t, t+1, \ldots, t+w-1$, as shown in Fig. 5. The essential difference is that now the window size w is chosen independently of I and the decoders within the window instead perform more than a single iteration according to some schedule before the window moves forward. Different uniform and non-uniform decoding schedules for SC-LDPC codes have been investigated in [18].

Broadly speaking, a decoding schedule for BCCs consists of doing I_1 turbo, or *vertical*, iterations on the block at time t, during which messages on the N information bits in that block are passed between the two component decoders at time t, then passing the resulting messages on the parity bits forward and doing I_1 vertical iterations on the block at time $t+1$, and continuing until I_1 vertical iterations are performed on the most recently received block at time $t+w-1$.[1] This process is then repeated in the backward direction, with soft LLRs on parity bits being passed back through the $2w$ BCJR decoders in the window, until the block at time unit t is reached again. This round trip of decoding is referred to as a *horizontal iteration*. After some number I_2 of horizontal iterations, the N target symbols are decoded, a new block is received, and the window shifts forward to the next position, where the set of target symbols for the block at time $t+1$ will be decoded.

Various decoding schedules have been proposed in [19]. It turns out that the most computationally efficient schedules perform a small number of vertical iterations, usually $I_1 = 1$, and a larger number of horizontal iterations, typically $5 \leq I_2 \leq 20$. Also, schedules that include some horizontal iterations that don't span the entire window, but instead are concentrated on the blocks closest to the target symbol block, are found to result in the best performance/complexity tradeoffs, where computational complexity is measured in terms of the total number of turbo (vertical) iterations required to decode a set of target symbols. Stopping rules, analogous to the kind of stopping rules used in standard turbo decoding, have been shown to result in further reductions in complexity with almost no loss in performance [19].

[1]For simplicity we assume the same number of vertical iterations are performed per block, but more complex schedules can also consider varying the number of vertical iterations performed on each block.

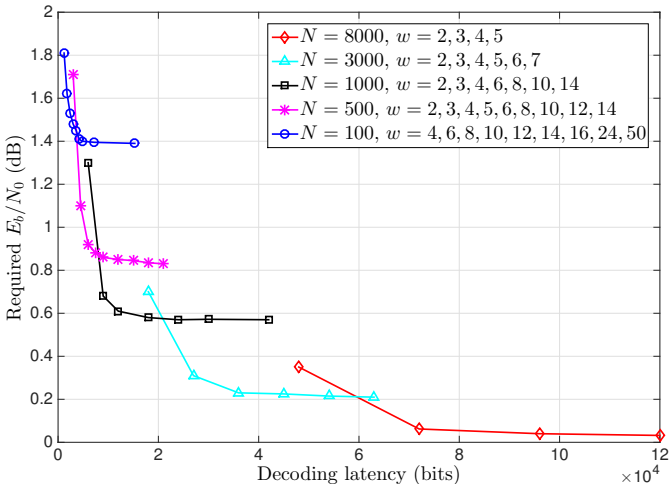


Fig. 6: Required E_b/N_0 to achieve a BER of 10^{-5} for rate $R = 1/3$ BCCs with window decoding as a function of decoding latency [19].

In Fig. 6, we plot the *signal-to-noise ratio* (SNR), measured in terms of E_b/N_0, needed to achieve a *bit-error rate* (BER) of 10^{-5} on the binary-input additive white Gaussian noise (BIAWGN) channel for rate $R = 1/3$ coupled blockwise BCCs with iterative sliding window BCJR decoding as a function of *decoding latency*.[2] The decoding latency is given by

$$\tau = 3Nw, \tag{1}$$

which represents the total number of symbols (information and parity) occupying the window at any given time. Results

[2]The results presented here were obtained by simulating frames of 50 blocks, each containing N information bits, with one added block for frame termination. Thus there is a slight rate loss associated with the BCC results. However, termination is not necessary, and essentially the same results can be obtained running the decoder in a continuous fashion.

are given for a wide range of block sizes, from very short ($N = 100$) to quite long ($N = 8000$). From the figure, we see that the performance improves with block size, as we would expect. We also see that, particularly for large block sizes, the window size w does not need to be large in order to achieve the best performance. For example, for $N = 8000$, $w = 3$ is sufficient, since the performance improves only slightly for larger w. For short blocks, however, larger window sizes are needed, since the base turbo code is weak. In this case, more blocks must be included in the decoding window to achieve the best performance. For example, for $N = 100$, the performance continues to improve with w up until about $w = 14$.

V. Puncturing of BCCs

In recent years, several varieties of braided codes have become popular choices for optical communication applications, which require high code rates and low error floors. The braided BCH codes proposed in [23] and the staircase codes introduced in [24], [25] are two examples that employ the concept of braided block codes. Because of their turbo-like structure, BCCs do not lend themselves naturally to high rates. Using puncturing of low rate BCCs up to higher rates to obtain a set of *rate-compatible* BCCs has recently been shown to yield surprisingly good results, however [16], [19]. Here we briefly review some of those results.

Considering the rate $R = 1/3$ coupled blockwise BCC with $N = 8000$ as the *mother code*, periodic puncturing patterns can be applied to the parity sequences to produce code rates of $R = 1/2$ and $R = 2/3$. Simulation results on the BIAWGN channel are shown in Fig. 7, where the BER performance of the rate-compatible BCCs with window decoding is compared both to the Shannon limit and the finite-length bound of Polyanskiy et al. [26]. The window size was $w = 3$, the decoding schedule used $I_1 = 1$ and $I_2 = 20$, and the decoding latencies for rates $1/3$, $1/2$, and $2/3$ were $72,000$, $48,000$, and $36,000$, respectively. At a BER of 10^{-5}, the rate $1/3$, $1/2$, and $2/3$ rate-compatible codes perform about 0.56dB, 0.58dB, and 0.62dB away from the Shannon limit, respectively, and they show no visible sign of an error floor down to a BER of 10^{-8}. We also see that, at a BER of 10^{-8}, the rate-compatible codes are less than 0.5dB away from the finite-length bound.

In Fig. 8, we compare the performance of a BCC, a turbo code, and a SC LDPC code, all with the same rate $R = 1/2$ and decoding latency $\tau = 48,000$.[3] The BCC code was obtained by puncturing the rate $R = 1/3$ BCC mother code with 4-state, $R_{\text{cc}} = 2/3$ component codes, the turbo code was obtained by puncturing the rate $R = 1/3$ turbo code with 8-state, $R_{\text{cc}} = 1/2$ component codes used in the CDMA 2000 standard and a randomly chosen permutor, and the SC LDPC code was randomly constructed based on a $(3, 6)$-regular protograph [27]. We observe that, for the same decoding latency $\tau = 48,000$ bits, the coupled blockwise BCC outperforms the SC LDPC code by about 0.3dB at a BER of 10^{-6}. Moreover, the BCC code outperforms the turbo code by about 0.1dB and, unlike the turbo code, displays no visible error floor. These results show that BCCs are very competitive with other types of capacity-approaching codes even when puncturing must be applied to achieve higher rates.

[3]Since each of the code simulations was performed by terminating long frames, the actual rate in each case was slightly less than $1/2$.

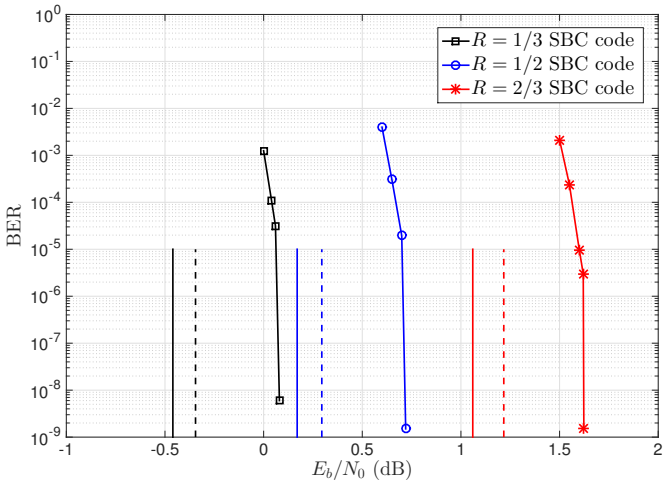


Fig. 7: Window decoding BER performance of rate-compatible BCCs with obtained by periodic puncturing [19]. Also shown for comparison are the corresponding finite length bounds (dashed lines) [26] and the Shannon limit (solid lines).

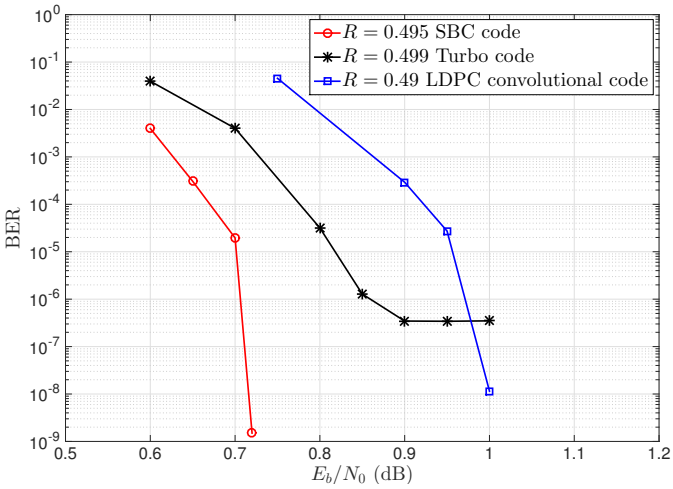


Fig. 8: BER performance of a BCC, a turbo code, and a SC LDPC code, all with the same rate R and decoding latency [19].

Even higher rates can be achieved by starting with a higher rate BCC mother code [19]. For example, a rate $R = 1/2$ coupled blockwise BCC with two identical rate $R_{\text{cc}} = 3/4$ component encoders can be constructed by dividing each information block of length $2N$ into two sub-blocks. The sub-blocks of length N comprise two of the three inputs to each component encoder, with the delayed and permuted parity output from the other component encoder being the third input. In this case, periodic puncturing of the parity sequences produces a rate-compatible set of BCCs with rates $R = 1/2$, $2/3$, and $3/4$.

Finally, we note that the absence of any visible error floor for the BCCs is consistent with the fact that the free distance of coupled BCCs grows linearly with constraint length and that coupled BCCs have relatively large free distance growth rates, as shown in Section III. The existence of an error floor for the turbo code, on the other hand, is due to the fact that the minimum distance of turbo codes only grows sub-linearly with block length.

VI. Concluding Remarks

In this paper, we presented a broad overview of the class of braided convolutional codes and summarized some recent advances in the field. Classical BCCs were viewed as a type of spatially coupled code, and an uncoupled (block code) version of BCCs, with no memory connecting successive blocks, was introduced. The iterative decoding thresholds of BCCs were then compared to those of conventional parallel and serially concatenated convolutional codes and their SC counterparts, offering insight into the relationship between BCCs and turbo codes. A finite-length distance analysis of both uncoupled and coupled BCCs demonstrated linear growth of minimum (free) distance with block (constraint) length. Further, the free distance growth rates were shown to compare favorably to the asymptotic bounds for the entire class of convolutional codes, thus promising good error floor performance.

Next, pipeline decoding of BCCs was reviewed, and a recently proposed window decoding method was described that allows low latency operation. Then puncturing was shown to result in robustly good performance for BCCs in both the waterfall and error floor regions of the BER curve for rates as high as $3/4$. Finally, comparisons were made between BCCs, turbo codes, and spatially coupled LDPC codes that demonstrate the excellent performance of BCCs for latency-constrained applications.

Acknowledgment

The authors would like to thank B. Bai, A. Graell i Amat, S. Moloudi, W. Zhang, M. Zhu, and K. Sh. Zigangirov for their contributions to the work presented in this paper.

References

[1] A. J. Felström, D. Truhachev, M. Lentmaier, and K. Sh. Zigangirov, "Braided block codes," *IEEE Transactions on Information Theory*, vol. 55, no. 6, pp. 2640–2658, June 2009.

[2] P. Elias, "Error free coding," *IRE Transactions on Information Theory*, vol. 4, no. 4, pp. 29–37, Sept. 1954.

[3] W. Zhang, M. Lentmaier, D. J. Costello, Jr., and K. Zigangirov, "Braided convolutional codes," in *Proc. IEEE International Symposium on Information Theory*, Adelaide, Australia, 2005, pp. 592–596.

[4] W. Zhang, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Braided convolutional codes: A new class of turbo-like codes," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 316–331, Jan. 2010.

[5] M. B. S. Tavares, M. Lentmaier, K. Sh. Zigangirov, and G. P. Fettweis, "LDPC convolutional codes based on braided convolutional codes," in *Proc. IEEE International Symposium on Information Theory*, Toronto, Canada, July 2008.

[6] M. B. S. Tavares, M. Lentmaier, G. P. Fettweis, and K. Sh. Zigangirov, "Asymptotic distance and convergence analysis of braided protograph convolutional codes," in *Proc. Forty-sixth Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, Sept. 2008.

[7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes," in *Proc. IEEE International Conference on Communications*, Geneva, Switzerland, May 1993.

[8] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.

[9] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-like codes," *submitted to the IEEE Transactions on Information Theory*, 2015. [Online]. Available: http://arxiv.org/abs/1604.07315

[10] ——, "Spatially coupled turbo codes," in *International Symposium on Turbo Codes and Iterative Information Processing*, Bremen, Germany, Aug. 2014, pp. 82–86.

[11] M. Lentmaier, S. Moloudi, and A. Graell i Amat, "Braided convolutional codes - a class of spatially coupled turbo-like codes," in *International Conference on Signal Processing and Communications*, Bangalore, India, July 2014, pp. 1–5.

[12] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Finite length weight enumerator analysis of braided convolutional codes," in *Proc. International Symposium on Information Theory and Its Applications*, Monterey, CA, Oct. 2016.

[13] K. Engdahl, M. Lentmaier, and K. S. Zigangirov, "On the theory of low-density convolutional codes," in *Proceedings of the Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, Honolulu, Hawaii, June 1999, pp. 77–86.

[14] D. J. Costello, Jr., "Free distance bounds for convolutional codes," *IEEE Transactions on Information Theory*, vol. 20, no. 3, pp. 356–365, May 1974.

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

[16] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, and B. Bai, "Window decoding of braided convolutional codes," in *Proc. IEEE Information Theory Workshop*, Jeju Island, Korea, Oct. 2015.

[17] M. Lentmaier, M. M. Prenda, and G. Fettweis, "Efficient message passing scheduling for terminated LDPC convolutional codes," in *Proc. IEEE International Symposium on Information Theory*, St. Petersburg, Russia, Aug. 2011, pp. 1826–1830.

[18] N. ul Hassan, A. E. Pusane, M. Lentmaier, G. P. Fettweis, and D. J. Costello, Jr., "Non-uniform windowed decoding schedules for spatially coupled codes," in *Proc. IEEE Global Communications Conference*, Atlanta, GA, Dec. 2013, pp. 1862–1867.

[19] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, and B. Bai, "Braided convolutional codes with low-latency sliding window decoding," *submitted to the IEEE Transactions on Communications*, 2016.

[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 International Symposium on Information Theory*, Adelaide, Australia, Sept. 2005.

[21] M. Lentmaier, A. Sridharan, D. J. Costello, Jr., and K. Sh. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.

[22] A. R. Iyengar, M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Coralli, and G. E. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2303–2320, Apr. 2012.

[23] Y.-Y. Jian, H. D. Pfister, K. R. Narayanan, R. Rao, and R. Mazahreh, "Iterative hard-decision decoding of braided BCH codes for high-speed optical communication," in *IEEE Global Communications Conference*, Atlanta, GA, Dec. 2013, pp. 2376–2381.

[24] B. Smith, A. Farhood, A. Hunt, F. Kschischang, and J. Lodge, "Staircase codes: FEC for 100 Gb/s otn," *IEEE/OSA Journal of Lightwave Technology*, vol. 30, no. 1, pp. 110–117, 2012.

[25] L. Zhang and F. Kschischang, "Staircase codes with 6% to 33% overhead," *IEEE/OSA Journal of Lightwave Technology*, vol. 32, no. 10, pp. 1999–2002, 2014.

[26] Y. Polyanskiy, H. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[27] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, Jr., "Spatially coupled LDPC codes constructed from protographs," *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 4866–4889, Sep. 2015.