Window Decoding of Braided Convolutional Codes

Zhu, Min; Mitchell, David G.M.; Lentmaier, Michael; Costello Jr., Daniel J.; Bai, Baoming

Published in:
2015 IEEE Information Theory Workshop - Fall (ITW)

DOI:
10.1109/ITWF.2015.7360751

2015

Window Decoding of Braided Convolutional Codes

Min Zhu[†*], David G. M. Mitchell[*], Michael Lentmaier[‡], Daniel J. Costello, Jr.[*], Baoming Bai[†]
[†] State Key Laboratory of ISN, Xidian University, Xi'an, P. R. China, {zhumin@stu, bmbai@mail}.xidian.edu.cn
[*] Dept. of Electrical Engineering, University of Notre Dame, Notre Dame, USA, {mzhu4, david.mitchell, dcostel1}@nd.edu
[‡] Dept. of Electrical and Information Technology, Lund University, Sweden, michael.lentmaier@eit.lth.se

***Abstract*—We consider a window decoding scheme for Braided Convolutional Codes (BCCs) based on the BCJR algorithm. We describe the principle of this decoding scheme and show that BCCs with window decoding exhibit excellent performance. The tradeoff between performance and decoding latency is examined and, to reduce decoding complexity, both uniform and nonuniform message passing schedules within the decoding window are considered. Finally, puncturing is employed to obtain rate-compatible code rates of 1/2 and 2/3 starting from a rate 1/3 example mother code. Simulation results show that, with nonuniform message passing and puncturing, near capacity performance can be maintained throughout the rate range considered with reasonable decoding complexity and no visible error floors.**

I. Introduction

Braided block codes (BBCs) [1] can be regarded as a diagonalized version of product codes [2] or expander codes [3], [4]. A BBC is constructed by interconnecting two block component codes: a horizontal component code and a vertical component code. Information symbols are checked by the two component encoders, and the parity symbols of one component encoder are used as inputs to the other component encoder. Recently, BBCs with Bose-chaudhuri-Hocqenghem (BCH) component codes [5] and the closely related staircase codes [6] have been investigated for high-speed optical communication and they have been found to achieve excellent performance with iterative hard decision decoding.

As a counterpart of BBCs, braided convolutional codes (BCCs) [7], a class of turbo-like codes, were first introduced in [8]; however, in contrast to BBCs, BCCs use short constraint length convolutional codes as component codes. BBCs and BCCs are similar in terms of the encoding process. The encoding of BCCs can be described by a two-dimensional sliding array, where each symbol is protected by two component convolutional codes. The connections between the two component encoders are defined by the positions where information symbols and parity symbols are stored in the two-dimensional array. Analogous to BBCs, a tightly braided convolutional (TBC) code results when a dense array is used to store the information and parity symbols. Alternatively, sparsely braided convolutional (SBC) codes have low density, resulting in improved iterative decoding performance [7]. It was also shown (numerically) in [7] that the minimum distance of SBC codes grows linearly with the overall constraint length, leading to the conjecture that SBC codes are asymptotically good. In [9] and [10], the BP threshold of SBC codes was analyzed on the binary erasure channel, and it was demonstrated that threshold saturation occurs, i.e., SBC codes behave in a manner similar to LDPC convolutional (spatially coupled) codes.

In this paper, we introduce a new decoding scheme for SBC codes. Instead of the pipeline decoder used in [7] and [8], a window decoder is proposed for SBC codes operating over the binary-input additive white Gaussian noise (AWGN) channel. Window decoding, which has been extensively studied for LDPC convolutional codes [11],[12],[13], provides a simple and efficient way to trade off decoding performance for reduced latency. Unlike window decoding of sparse LDPC convolutional codes, which typically use an iterative message passing algorithm based on belief propagation, window decoding of SBC codes is based on the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm. In addition, the tradeoff between performance and decoding latency is explored, and we discuss how to choose the permutor size and the window size to achieve the best performance when the decoding latency is fixed. Moreover, in order to reduce the decoding complexity, different massage passing schedules (both uniform and nonuniform) are investigated. Finally, we consider periodic puncturing of SBC codes to achieve rate-compatible SBC codes and their performance using window decoding is compared to that of LDPC convolutional codes.

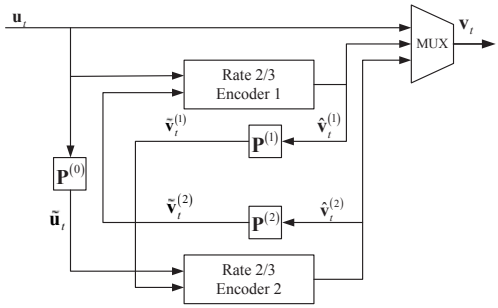

Fig. 1. Encoder for a rate R = 1/3 blockwise SBC code.

II. Braided Convolutional Codes

Braided convolutional codes are constructed using an infinite two-dimensional array consisting of one horizontal and one vertical encoder. These two encoders are linked through parity feedback. In this manner, the systematic and parity symbols are "braided" together. There are two classes of SBC codes: bitwise and blockwise. In this paper we limit ourselves to rate $R = 1/3$ blockwise SBC codes. In this case, the information sequence enters the encoder in a block-by-block manner with a relatively large block size. An example of a rate $R = 1/3$ blockwise SBC encoder is illustrated in Fig. 1. It consists of two systematic convolutional component encoders each of rate $R_{cc} = 2/3$. The information sequence is divided into blocks of length T symbols, i.e., $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_t, \ldots)$, where $\mathbf{u}_t = (u_{t,1}, u_{t,2}, \ldots, u_{t,T})$. $\mathbf{P}^{(0)}$, $\mathbf{P}^{(1)}$, and $\mathbf{P}^{(2)}$ are each block permutors of length T.

At time instant $t = 0$, information block $\mathbf{u}_0$ and its permuted version $\tilde{\mathbf{u}}_0 = \mathbf{u}_0\mathbf{P}^{(0)}$ enter the first inputs of Encoder 1 and Encoder 2, respectively. Meanwhile, blocks $\tilde{\mathbf{v}}_{-1}^{(2)}$ and $\tilde{\mathbf{v}}_{-1}^{(1)}$, consisting of T zeros each, enter the second inputs of Encoder 1 and Encoder 2, respectively. Encoders 1 and 2 then generate length T parity blocks $\hat{\mathbf{v}}_0^{(1)}$ and $\hat{\mathbf{v}}_0^{(2)}$, and blocks $\mathbf{v}_0^{(0)} = \mathbf{u}_0$, $\mathbf{v}_0^{(1)} = \hat{\mathbf{v}}_0^{(1)}$, and $\mathbf{v}_0^{(2)} = \hat{\mathbf{v}}_0^{(2)}$ are sent over the channel. At time instant t, parity block $\mathbf{v}_t^{(1)}$ is calculated by Encoder 1 as a function of $\mathbf{u_t}$

and $\tilde{\mathbf{v}}_t^{(1)} = \mathbf{v}_{t-1}^{(2)}\mathbf{P}^{(2)}$. Similarly, parity block $\mathbf{v}_t^{(2)}$ is calculated by Encoder 2 as a function of $\tilde{\mathbf{u}}_t = \mathbf{u}_t\mathbf{P}^{(0)}$ and $\tilde{\mathbf{v}}_t^{(2)} = \mathbf{v}_{t-1}^{(1)}\mathbf{P}^{(1)}$. The blocks $\mathbf{v}_t^{(0)} = \mathbf{u}_t$, $\mathbf{v}_t^{(1)} = \hat{\mathbf{v}}_t^{(1)}$, and $\mathbf{v}_t^{(2)} = \hat{\mathbf{v}}_t^{(2)}$ are multiplexed into the code sequence $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_t, \ldots)$, where

$$\mathbf{v}_t = \left(v_{t,1}^{(0)}, v_{t,1}^{(1)}, v_{t,1}^{(2)}, v_{t,2}^{(0)}, v_{t,2}^{(1)}, v_{t,2}^{(2)}, \ldots, v_{t,T}^{(0)}, v_{t,T}^{(1)}, v_{t,T}^{(2)}\right), \quad (1)$$

and sent over the channel.

In this paper, we consider rate $R = 2/3$ unterminated convolutional codes as component codes for blockwise SBC codes. At the end of the information sequence, termination is used to protect the final information blocks. In this case, after the information blocks $\mathbf{u}_{[0,L-1]} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{L-1})$ enter the blockwise SBC encoder, Λ additional all-zero blocks $\mathbf{u}_L, \ldots, \mathbf{u}_{L+\Lambda-1}$ enter the encoder. Note that these Λ blocks are not sent over the channel but the resulting parity blocks are transmitted. The actual rate of the SBC code, including the tail, is given by

$$\tilde{R} = \frac{1}{3}\frac{L}{L + (2\Lambda/3)}. \quad (2)$$

III. Window Decoding

A parallel pipeline decoder was used for the BCCs in [7] in order to achieve high throughput continuous decoding. However, the decoding latency required for good performance in this case is very large. Significantly reduced decoding latency can be obtained with little or no loss in performance by using window decoding. In this section, we introduce a window decoding scheme for SBC codes.

A. Window Decoding

Assume that the rate $R = 1/3$ blockwise BCC encoder described in Section II is used. The code sequence is $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_t, \ldots)$, where $\mathbf{v}_t$ is given by (1). After transmission over an AWGN channel, the received sequence is $\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_t, \ldots)$, where

$$\mathbf{r}_t = \left(r_{t,1}^{(0)}, r_{t,1}^{(1)}, r_{t,1}^{(2)}, r_{t,2}^{(0)}, r_{t,2}^{(1)}, r_{t,2}^{(2)}, \ldots, r_{t,T}^{(0)}, r_{t,T}^{(1)}, r_{t,T}^{(2)}\right).$$

Let $\mathbf{l}^c = (\mathbf{l}_0^c, \mathbf{l}_1^c, \ldots, \mathbf{l}_t^c, \ldots)$ denote the channel log-likelihood ratios (LLRs), where

$$\begin{aligned}\mathbf{l}_t^c &= \left(l_{t,1}^{c,(0)}, l_{t,1}^{c,(1)}, l_{t,1}^{c,(2)}, l_{t,2}^{c,(0)}, l_{t,2}^{c,(1)}, l_{t,2}^{c,(2)}, \ldots, l_{t,T}^{c,(0)}, l_{t,T}^{c,(1)}, l_{t,T}^{c,(2)}\right) \\ &= \left(\mathbf{l}_t^{c,(0)}, \mathbf{l}_t^{c,(1)}, \mathbf{l}_t^{c,(2)}\right),\end{aligned}$$

and $\mathbf{l}_t^{c,(0)}$, $\mathbf{l}_t^{c,(1)}$, and $\mathbf{l}_t^{c,(2)}$ represent the channel LLRs of the information symbols, the parity symbols from Encoder 1, and the parity symbols from Encoder 2, respectively, at time instant t.

These LLRs are demultiplexed into two streams. For component Encoder i, $i \in \{1, 2\}$, the channel LLRs corresponding to the outputs $\mathbf{v}^{(i)} = \left(\mathbf{v}_0^{(i)}, \mathbf{v}_1^{(i)}, \ldots, \mathbf{v}_t^{(i)}, \ldots\right)$, where $\mathbf{v}_t^{(i)} = \left(v_{t,1}^{(0)}, v_{t,1}^{(i)}, v_{t,2}^{(0)}, v_{t,2}^{(i)}, \ldots, v_{t,T}^{(0)}, v_{t,T}^{(i)}\right)$, are given by $\mathbf{l}^{c,(i)} = \left(\mathbf{l}_0^{c,(i)}, \mathbf{l}_1^{c,(i)}, \ldots, \mathbf{l}_t^{c,(i)}, \ldots\right)$, where

$$\mathbf{l}_t^{c,(i)} = \left(l_{t,1}^{c,(0)}, l_{t,1}^{c,(i)}, l_{t,2}^{c,(0)}, l_{t,2}^{c,(i)}, \ldots, l_{t,T}^{c,(0)}, l_{t,T}^{c,(i)}\right).$$

The window decoding diagram is shown in Fig. 2. Before describing the window decoding scheme, we give some notation. For time instant t, $t \in [0, L + \Lambda - 1]$, $\mathbf{l}_{\text{Inf}}^c(t)$, $\mathbf{l}_{\text{Inf}}^a(t)$, and $\mathbf{l}_{\text{Inf}}^e(t)$ represent the channel LLRs, the *a priori* LLRs, and the extrinsic LLRs of the information symbols of Decoder 1, respectively. $\mathbf{l}_{\text{Pin1}}^c(t)$, $\mathbf{l}_{\text{Pin1}}^a(t)$, and $\mathbf{l}_{\text{Pin1}}^e(t)$ denote the channel LLRs, the *a priori* LLRs, and the extrinsic LLRs of the input parity symbols of Decoder 1, respectively. $\mathbf{l}_{\text{Pout1}}^c(t)$, $\mathbf{l}_{\text{Pout1}}^a(t)$, and $\mathbf{l}_{\text{Pout1}}^e(t)$ denote the channel LLRs, the *a priori* LLRs, and the extrinsic LLRs of the output parity symbols of Decoder 1, respectively. Similarly, $\tilde{\mathbf{l}}_{\text{Inf}}^c(t)$, $\tilde{\mathbf{l}}_{\text{Inf}}^a(t)$ and $\tilde{\mathbf{l}}_{\text{Inf}}^e(t)$ are the channel LLRs, the *a priori* LLRs, and the extrinsic LLRs of the information symbols of Decoder 2, respectively, and the Decoder 2 LLRs $\mathbf{l}_{\text{Pin2}}^c(t)$, $\mathbf{l}_{\text{Pin2}}^a(t)$, $\mathbf{l}_{\text{Pin2}}^e(t)$, $\mathbf{l}_{\text{Pout2}}^c(t)$, $\mathbf{l}_{\text{Pout2}}^a(t)$, and $\mathbf{l}_{\text{Pout2}}^e(t)$ are defined analogously to the corresponding LLRs of Decoder 1. Finally, we assume that the window size is w, which means there are w blocks in the decoding window at any particular time.

When the transmitted symbols are received, the first w blocks of channel LLRs enter the window decoder. Among these blocks, the first one to enter contains the *target symbols*, i.e., the first block of symbols to be decoded. There are two types of information exchange in the decoder. One is the information exchange between two component decoders at the same time instant, which operates on the information symbols in a given block iteratively. Since the individual blocks are not terminated, the component codes are decoded using the windowed BCJR algorithm [14]. The other is the information exchange between two component decoders at different time instants, which operates on the parity symbols associated with two successive blocks iteratively. For simplicity, we call them the vertical exchange and the horizontal exchange, respectively.

At time t, the decoding window covers w blocks at time indexes $s \in [t, t + w - 1]$. We initialize the window decoder as follows. For Decoder 1, we have

$$\begin{aligned}&\mathbf{l}_{\text{Inf}}^c(s) = \mathbf{l}_s^{c,(0)} \\ &\mathbf{l}_{\text{Pout1}}^c(s) = \mathbf{l}_s^{c,(1)} \\ &\mathbf{l}_{\text{Pin1}}^c(s) = \begin{cases} k_1, & s = 0 \\ \mathbf{l}_{s-1}^{c,(2)}\mathbf{P}^{(2)}, & s \in (0, w-1], \end{cases}\end{aligned} \quad (3)$$

and for Decoder 2, we have

$$\begin{aligned}&\tilde{\mathbf{l}}_{\text{Inf}}^c(s) = \mathbf{l}_s^{c,(0)}\mathbf{P}^{(0)} \\ &\mathbf{l}_{\text{Pout2}}^c(s) = \mathbf{l}_s^{c,(2)} \\ &\mathbf{l}_{\text{Pin2}}^c(s) = \begin{cases} k_2, & s = 0 \\ \mathbf{l}_{s-1}^{c,(1)}\mathbf{P}^{(1)}, & s \in (0, w-1], \end{cases}\end{aligned} \quad (4)$$

where k_1 and k_2 are negative constants, because we assume that the second inputs of Encoders 1 and 2 are all zeros at time instant $s = 0$. Also, the *a priori* and the extrinsic LLRs are zeros at initialization.

The window decoder starts with vertical decoding of the first block in the window (time instant t in Fig. 2) with I_1 iterations (termed *intra-block iterations*). This procedure is similar to the decoding of turbo codes. The major difference is that the *a posteriori* probability (APP) values for all the code symbols are calculated, instead of only for the information symbols. After I_1 intra-block iterations, the extrinsic information is permuted appropriately and passed to the adjacent decoders at time instant $t + 1$. After receiving these *a priori* LLRs, the decoders at time instant $(t+1)$ perform I_1 iterations of vertical decoding and pass the extrinsic information to the decoders at time instant $t + 2$. This forward (horizontal) process continues until the last block at time instant $(t+w-1)$ finishes vertical decoding. For the forward

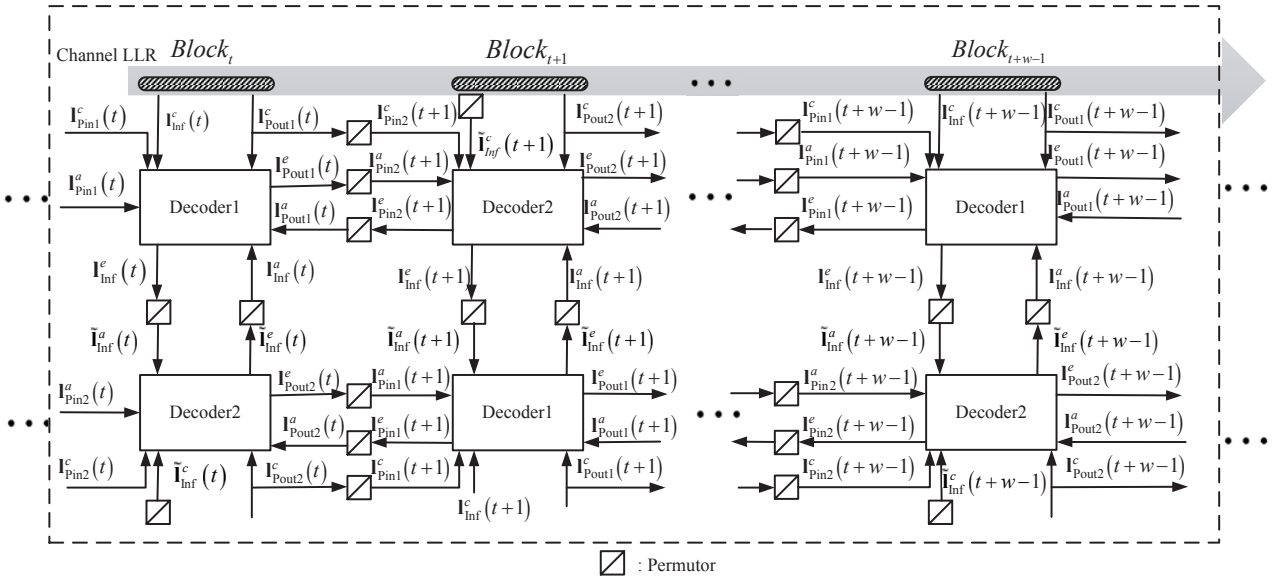

Fig. 2. Window decoder for a rate R = 1/3 SBC code.

exchange process, we have

$$\begin{aligned} \mathbf{l}^{a}_{\mathrm{Pin1}}(s) &= \mathbf{l}^{e}_{\mathrm{Pout2}}(s-1)\,\mathbf{P}^{(2)} \\ \mathbf{l}^{a}_{\mathrm{Pin2}}(s) &= \mathbf{l}^{e}_{\mathrm{Pout1}}(s-1)\,\mathbf{P}^{(1)}, \end{aligned} \tag{5}$$

where $\mathbf{l}^{e}_{\mathrm{Pout2}}(-1) = \mathbf{l}^{e}_{\mathrm{Pout1}}(-1) = \mathbf{0}$, for $s = 0$.

Following this, the decoders at time instant $(t+w-1)$ transmit the extrinsic information of their parity input symbols back to the decoders at time instant $(t+w-2)$. After receiving these *a priori* LLRs, the decoders at time instant $(t+w-2)$ perform I_1 iterations of vertical decoding and then pass the extrinsic information back to the decoders at time instant $(t+w-3)$. This backward (horizontal) process continues until extrinsic information is passed back to the first block in the window. For the backward exchange process, we have

$$\begin{aligned} \mathbf{l}^{a}_{\mathrm{Pout1}}(s) &= \mathbf{l}^{e}_{\mathrm{Pin2}}(s+1)\left[\mathbf{P}^{(1)}\right]^{T} \\ \mathbf{l}^{a}_{\mathrm{Pout2}}(s) &= \mathbf{l}^{e}_{\mathrm{Pin1}}(s+1)\left[\mathbf{P}^{(2)}\right]^{T}, \end{aligned} \tag{6}$$

where $\mathbf{l}^{e}_{\mathrm{Pin2}}(t+w) = \mathbf{l}^{e}_{\mathrm{Pin1}}(t+w) = \mathbf{0}$, for $s = t+w-1$. Then a new round of horizontal decoding begins.

The horizontal decoding process (forward and backward) is performed I_2 times (termed *inter-block iterations*). Then hard decisions are made on the target symbols (the first block in the window). After that, the window shifts by one time instant. At the same time, one new block enters the window. After the initialization of the new block, the decoding process then starts again in the new window.

B. Window Decoding Schedules

From above description, it is clear that the number of intra-block iterations I_1 and inter-block iterations I_2 plays an important role in the tradeoff between performance and computational complexity. In this section, we present several window decoding message passing schedules as a means to investigate this tradeoff. Assume that the window size is w.

1) Cyclic Schedule: In this schedule, described in Section III above, the component decoders in the window operate sequentially, so that the middle decoders are updated more often than those at the end. The pattern of the cyclic schedule is shown in Fig. 3(a) for every time instant $s \in [t, t+w-1]$, where a rectangle represents a pair of component decoders and I_1 is the number of intra-block iterations. The total number of iterations in this case is given by $2(w-1)I_1I_2$.

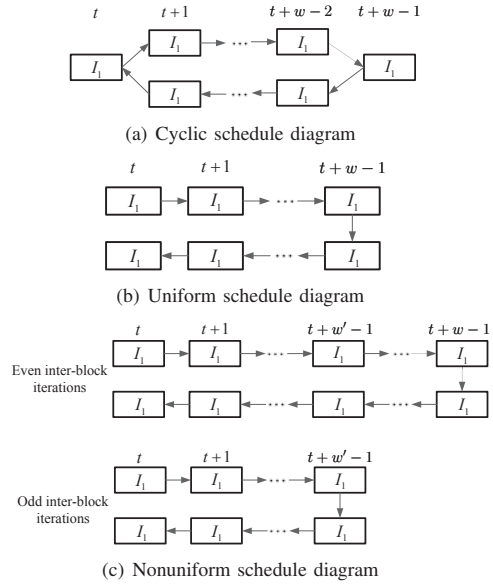

Fig. 3. Different window decoding schedules.

2) Uniform Schedule: In this schedule, for every inter-block iteration, each block in the window is updated with the same number of iterations. The pattern of the uniform schedule is shown in Fig. 3(b), and the total number of iterations in this case is given by $2wI_1I_2$.

3) Nonuniform Schedule: In this schedule, during the even inter-block iterations, the decoding schedule is the same as the uniform schedule. During the odd inter-block iterations, however, the forward process stops at block $(w'-1)$, $0 < w' < w$, and the backward process starts at block $(w'-1)$. The pattern of this nonuniform schedule is shown in Fig. 3(c). The average

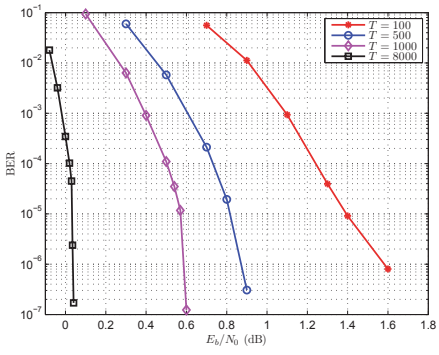

Fig. 4. BER performance of rate $R = 1/3$ SBC codes with window decoding and different permutor sizes T. The window size w is 5 in the case of $T = 8000$, and $w = 14$ in the other cases. The intra-block iteration and inter-block iteration numbers are $I_1 = 5$ and $I_2 = 30$, respectively.

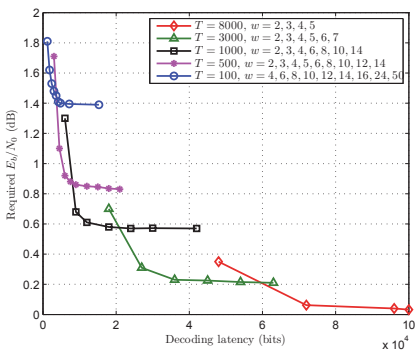

Fig. 5. Required E_b/N_0 to achieve a BER of 10^{-5} for the rate $R = 1/3$ SBC code as a function of decoding latency.

total number of iterations in this case is given by $I_1 I_2 (w + w')$. The idea behind the nonuniform schedule is to devote a greater fraction of the computational resources to updating the blocks closest to the target symbols.

IV. Numerical Results

In this section, some examples are given to illustrate the performance of BCCs with window decoding over the AWGN channel with binary phase-shift keying (BPSK) signaling.

We consider a rate $R = 1/3$ blockwise SBC code with two identical rate $R_{cc} = 2/3$, memory $m_{cc} = 2$, low complexity (4-state) RSC component encoders. The generator matrix of the component encoders is given by

$$G(D) = \begin{pmatrix} 1 & 0 & \frac{1}{1+D+D^2} \\ 0 & 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix}. \tag{7}$$

The unterminated version of the component encoder is employed. The three block permutors $\mathbf{P}^{(0)}$, $\mathbf{P}^{(1)}$, and $\mathbf{P}^{(2)}$ used in the encoder were chosen randomly with the same size T. We assume that a transmission consists of an information sequence of length $50T$ plus one zero termination block of length T.[1]

A. Performance of SBC Codes with Window Decoding

The bit error rate (BER) performance of the rate $R = 1/3$ blockwise SBC code with window decoding is shown in Fig. 4, where the permutor size changes from $T = 100$ to 8000. We see that the performance with window decoding improves as we increase the size of the block permutors, as expected.

In addition to BER performance, the decoding latency introduced by channel coding is a crucial factor in the design of a practical communication system. For the rate $R = 1/3$ SBC code, the decoding latency of the sliding window decoder is given by

$$\tau = 3Tw \tag{8}$$

symbols, where T and w are the permutor size and window size, respectively.

The bit signal-to-noise ratio E_b/N_0 required to achieve a BER of 10^{-5} as a function of decoding latency is shown in Fig. 5. We observe that the performance of the blockwise SBC code with fixed permutor size improves as the window size w increases; however, it does not improve much further beyond a certain window size. Moreover, beyond a certain latency, using a larger permutor size T with a smaller window size w gives better performance. Finally, we note that the performance of the blockwise SBC code with window decoding is at least as good as the curves published in [7], which used pipeline decoding with a much higher decoding latency.

[1]Note that the actual code rate is slightly less than the nominal rate R.

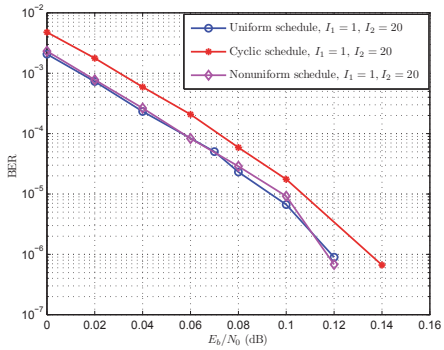

Fig. 6. BER performance of rate $R = 1/3$ SBC codes with window decoding using different decoding schedules for the same number of inter-block and intra-block iterations. The window size w is 3 and the block permutor size T is 8000. For the nonuniform schedule, $w' = 1$.

B. Window Decoding Schedules

For a fixed number of iterations, the performance of the SBC code with window decoding and different decoding schedules is compared in Fig. 6. The cyclic, uniform, and nonuniform schedules are examined with $I_1 = 1$ and $I_2 = 20$, where the total number of iterations for window size $w = 3$ is 80, 120, and 100, respectively.[2] As we see in Fig. 6, the performance of the window decoder with the nonuniform schedule is close to that of the uniform schedule, but this performance is achieved with fewer iterations using the nonuniform schedule. We also see that the performance of both the uniform and nonuniform schedules is better than the cyclic schedule, since, with the cyclic schedule, the *target symbols* are updated less often during a round of horizontal decoding than with the uniform and nonuniform schedules.

C. Puncturing

We now investigate the performance of rate-compatible blockwise SBC codes with window decoding. Using the rate $R = 1/3$ blockwise SBC code as the mother code, we obtain blockwise

[2]Experimentally, the window decoder displayed good performance with $w = 3$, $I_1 = 1$, and $I_2 = 20$. Further analysis of various schedules and decoder parameters, along with a complexity comparison, is included in [16].

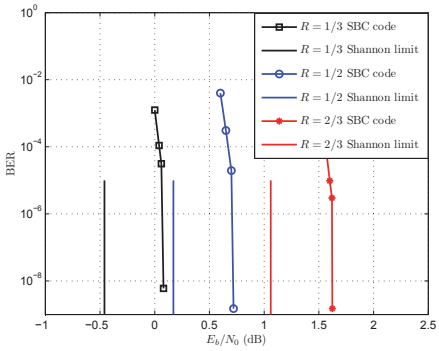

Fig. 7. BER performance of rate-compatible SBC codes with window decoding obtained by periodic puncturing. The window size w is 3 and the block permutor size T is 8000.

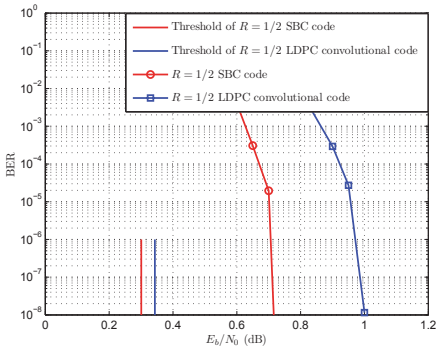

Fig. 8. BER performance of the blockwise SBC code and an LDPC convolutional code with the same decoding latency. For the SBC code, the window size w and the block permutor size T are 3 and 8000, respectively. For the LDPC convolutional code, the window size, the protograph lifting factor, and the coupling length are 6, 4000, and 50, respectively.

SBC codes with code rates of 1/2 and 2/3 by periodic puncturing. The BER performance of the blockwise SBC codes with code rates of 1/3, 1/2, and 2/3 with window decoding is shown in Fig. 7. At a BER of 10^{-5}, the blockwise SBC codes with code rates of 1/3, 1/2, and 2/3 perform about 0.56 dB, 0.58 dB, and 0.62 dB away from the Shannon limit and they show no visible sign of an error-floor down to a BER of 10^{-8}.

Next, the performance of the rate $R = 1/2$ blockwise SBC code is compared to a rate $R = 1/2$ LDPC convolutional (spatially coupled) code with the same decoding latency in Fig. 8.[3] The LDPC convolutional code is based on the $(3, 6)$-regular protograph with base matrix $\mathbf{B} = [3, 3]$, where the component base matrices are $\mathbf{B}_0 = [2, 1]$ and $\mathbf{B}_1 = [1, 2]$ (see [15] for details of the protograph construction). The thresholds of these two codes are also given [9]. We observe that, for the same decoding latency $\tau = 48,000$ symbols and the same code rate $R = 1/2$, the blockwise SBC code outperforms the LDPC convolutional code by about 0.3 dB at a BER of 10^{-6}.

V. Conclusion

In this paper, we introduced a low latency window decoding scheme for blockwise SBC codes. We also proposed both uniform and nonuniform decoding schedules for the window and compared their BER performance. Periodic puncturing was then used to achieve rate-compatible blockwise SBC codes, with the rate $R = 1/3$, $R = 1/2$ and $R = 2/3$ blockwise SBC codes performing about 0.56 dB, 0.58 dB, and 0.62 dB away from the Shannon limit at a BER of 10^{-5}. Moreover, these codes show no visible sign of an error-floor down to a BER of 10^{-8}. Finally, a comparison of a blockwise SBC code and an LDPC convolutional code, both with code rate $R = 1/2$, showed that blockwise SBC codes can outperform LDPC convolutional codes with window decoding under an equal decoding latency condition.

Based on their excellent performance in both the waterfall and error-floor regions, the robustness of this performance to puncturing, and the ability to employ a low-complexity soft decision decoding algorithm at high code rates, blockwise SBC codes appear to be worthy competitors to the product-like codes that have recently been proposed for high-speed optical communication.

[3] A more detailed performance comparison, including both turbo codes and LDPC convolutional codes, is presented in [16].

Acknowledgment

This work was supported in part by the U. S. NSF under Grant CCF-1161754, by the 973 Program of China under Grant 2012CB316100, and by the NSFC under Grant 61372074.

References

[1] A. J. Feltström, M. Lentmaier, D. V. Truhachev, and K. S. Zigangirov, "Braided block codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2640-2658, Jun. 2009.

[2] P. Elias, "Error free coding," *IRE Trans. Inf. Theory*, vol. 4, no. 4, pp. 29-37, Sep. 1954.

[3] M. Slipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1710-1722, Nov. 1996.

[4] G. Zémor, "On expander codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 835-837, Feb. 2001.

[5] Y. Y. Jian, H. D. Pfister, K. R. Narayanan, R. Rao and R. Mazahreh, "Iterative hard-decidion decoding of braided BCH codes for high-speed optical communication," in *Proc. IEEE Glob. Telecom. Conf.*, Dec. 2013.

[6] B. P. Smith, A. Farhood, A. Hunt, F. R. Kschischang and J. Lodge, "Staircase codes: FEC for 100 Gb/s OTN," *J. Lightwave Technol.*, vol. 30, no. 1, pp. 110-117, 2012.

[7] W. Zhang, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Braided convolutional codes: a new class of turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 316-331, Jan. 2010.

[8] W. Zhang, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Braided convolutional codes," in *Proc. IEEE Int. Symp. Information Theory*, Adelaide, Australia, Sep. 2005, pp. 592-596.

[9] S. Moloudi and M. Lentmaier, "Density evolution analysis of braided convolutional codes on the erasure channel," in *Proc. IEEE Int. Symp. Information Theory*, Honolulu, HI, USA, July 2014, pp. 2609-2613.

[10] M. Lentmaier, S. Moloudi, and A. Graell i Amat, "Braided convolutioanl codes - a class of spatially coupled turbo-like codes," in *Proc. Int. Conf. Signal Processing and Communications*, Bangalore, India, July 2014.

[11] M. Lentmaier, A. Sridharan, K. S. Zigangirov, and D. J. Costello, Jr., "Terminated LDPC convolutional codes with thresholds close to capcacity," in *Proc. IEEE Int. Symp. Information Theory*, Adelaide, Australia, Sep. 2005, pp. 1372-1376.

[12] M. Lentmaier, A. Sridharan, D. J. Costello, Jr., and K. S. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274-5289, Oct. 2010.

[13] 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 Trans. Inf. Theory*, vol. 58, no. 4, pp. 2303-2320, April 2012.

[14] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "A soft-input soft-output maximum a posteriori (map) module to decode parallel and serial concatenated codes," *JPL TDA Progr. Rep.*, vol. 42, no.127, pp. 1-20, Nov. 1996.

[15] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, Jr., "Spatially coupled LDPC codes constructed from protographs," accepted for publication in *IEEE Trans. Inf. Theory*. [Online]: http://arxiv.org/abs/1407.5366.

[16] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, Jr., and B. Bai, "Blockwise braided convolutional codes with low-latency decoding," in preparation.