Braided Convolutional Self-orthogonal Codes with Double Sliding Window Decoding

Min Zhu*, Andrew D. Cummins†, David G. M. Mitchell†, Michael Lentmaier‡, and Daniel J. Costello, Jr.§
*State Key Laboratory of ISN, Xidian University, Xi'an, P. R. China, zhunanzhumin@gmail.com
†Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM, USA, {andrewdc, dgmm}@nmsu.edu
‡Department of Electrical and Information Technology, Lund University, Lund, Sweden, michael.lentmaier@eit.lth.se
§Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, USA, dcostel1@nd.edu

2023 12th International Symposium on Topics in Coding (ISTC) | 979-8-3503-2611-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISTC57237.2023.10273524

***Abstract*—In this paper, we investigate a class of braided convolutional codes (BCCs), where the component codes are convolutional self-orthogonal codes (CSOCs), called braided convolutional self-orthogonal codes. Compared to conventional BCCs, the advantages of braided CSOCs include the availability of several low-complexity decoding methods and the relative ease of extending these methods to high rates. More specifically, to construct high-rate braided codes, it is necessary to use higher-rate component codes, which results in exponentially higher decoding complexity with conventional BCJR decoding, whereas the complexity of the CSOC decoding methods proposed here grows only linearly with rate. In particular, we introduce a double sliding window decoding method based on belief propagation (BP) for braided CSOCs, which exhibits good performance while maintaining low-complexity decoding for the higher rates required in many applications.**

***Index Terms*—Braided convolutional codes, convolutional self-orthogonal codes, sliding window decoding, threshold decoding**

I. Introduction

Braided convolutional codes (BCCs), first introduced in [1], are a type of parallel-concatenated (turbo) 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.[1] It was shown numerically in [1], [2] that the (minimum) free distance of BCCs grows linearly with the overall constraint length, leading to the conjecture that BCCs, unlike parallel or serially concatenated codes, are asymptotically good, and it has been shown that they can achieve excellent waterfall and error floor performance [3]. Due to their turbo-like structure, BCCs can be decoded based on the BCJR algorithm, and a low-latency *sliding window decoding* (SWD) algorithm was introduced in [4]. The code rate $R = (k_c - 1)/n_c$ of BCCs depends on the component code rate $R_c = k_c/n_c$, where k_c and n_c are the number of inputs and the number of outputs of the component code, respectively. In order to obtain high-rate BCCs, it is therefore necessary to use even higher rate component codes, for which the decoding complexity of the BCJR algorithm grows exponentially. Here, we investigate the use of convolutional self-orthogonal codes (CSOCs) as a means of reducing the decoding complexity of high-rate BCCs.

[1]BCCs are the convolutional counterparts of *braided block codes*, several varieties of which have been proposed for optical communication applications, which require high code rates and low error floors.

CSOCs and corresponding low-complexity hard-decision and soft-decsion (APP) *threshold decoding* algorithms were originally introduced by Massey [5], who proposed a construction procedure for finding codes with J orthogonal parity checks on each information symbol for arbitrary values of J and code rate R_c. The advantages of CSOCs include simple encoder and decoder implementation, a guaranteed error-correcting capability of $\lfloor J/2 \rfloor$, and high speed decoding. More efficient code constructions based on the notion of difference sets were introduced in [6], [7] (see also [8]). With threshold decoding, high rate CSOCs can achieve good performance with very high throughputs and moderate decoding complexity. In order to improve the performance, iterative threshold decoding was proposed in [9], [10]. Furthermore, due to the self-orthogonal structure of CSOCs, there are no 4-cycles in the Tanner graph of the parity check matrix, and hence belief-propagation (BP) decoding can also be employed [11], [12].

In this paper, we investigate a class of BCCs with rate $R_c = k_c/(k_c + 1)$, $k_c \geq 1$, using CSOCs as component codes, referred to as braided CSOCs. Then, with a goal of achieving both low-latency and low-complexity decoding, we propose a double sliding window decoding (DSWD) algorithm based on BP. Simulation results demonstrate that high-rate braided CSOCs with DSWD can maintain good performance in both the waterfall and error floor with modest encoding and decoding complexity.

II. Braided Convolutional Codes and Convolutional Self-orthogonal Codes

A. Braided Convolutional Codes

BCCs are constructed using a turbo-like parallel concatenation of two component encoders. However, unlike turbo codes, the two encoders share parity feedback. In this manner, the information and parity symbols are "braided" together. The information sequence $\mathbf{u}$ enters the encoder in a block-by-block manner, typically with a relatively large block size. Fig. 1 depicts the encoding process as a chain of encoders operating at different time instants for a rate $R = (k_c - 1)/(k_c + 1)$ BCC utilizing two *recursive systematic convolutional* (RSC) component encoders each of rate $R_c = k_c/(k_c + 1)$, where $\mathbf{P}^{(0)}$, $\mathbf{P}^{(1)}$, and $\mathbf{P}^{(2)}$ are each block permutors of size T. The information sequence is divided into blocks of length $(k_c - 1)T$ symbols, i.e., $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_t, \ldots)$, where $\mathbf{u}_t = \left(\mathbf{u}_t^{(0)}, \mathbf{u}_t^{(1)}, \ldots, \mathbf{u}_t^{(k_c-2)}\right)$ represents the block of $(k_c - 1)T$ in-

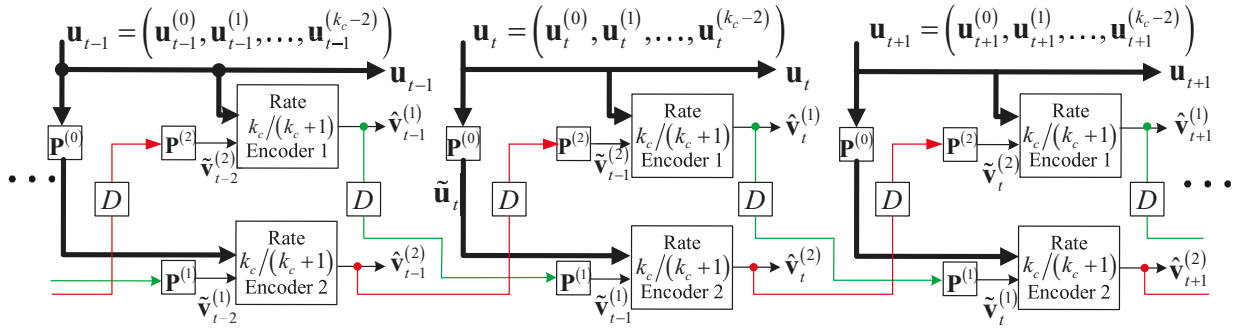

Fig. 1. Chain of encoders for a rate $R = (k_c - 1)/(k_c + 1)$ BCC using a rate $R_c = k_c/(k_c + 1)$ component code.

formation symbols at time i, with $\mathbf{u}_t^{(i)} = \left(u_{t,1}^{(i)}, u_{t,2}^{(i)}, \ldots, u_{t,T}^{(i)}\right)$ denoting the T information symbols that enter the ith encoder input, $i = 0, 1, \ldots, k_c - 2$. $\mathbf{u}_t$ is then interleaved using $\mathbf{P}^{(0)}$ to form $\tilde{\mathbf{u}}_t$, where $\mathbf{P}^{(0)}$ operates independently on each set of T symbols $\mathbf{u}_t^{(i)}$, $i = 0, 1, \ldots, k_c - 2$, and $\mathbf{u}_t$ and $\tilde{\mathbf{u}}_t$ enter the component encoders. The parity outputs $\hat{\mathbf{v}}_t^{(i)}$ from encoder i, $i \in \{1, 2\}$, at time t are delayed by one time unit, interleaved using $\mathbf{P}^{(1)}$ and $\mathbf{P}^{(2)}$, respectively, and then enter the component encoders as the input sequences $\tilde{\mathbf{v}}_{t+1}^{(i)}$, $i \in \{1, 2\}$, at time $t + 1$, where the parity input sequences $\tilde{\mathbf{v}}_{t+1}^{(i)}$ are associated with encoder input position $k_c - 1$. The information block $\mathbf{u}_t$, the parity output block $\hat{\mathbf{v}}_t^{(1)}$ of encoder 1, and the parity output block $\hat{\mathbf{v}}_t^{(2)}$ of encoder 2 are sent over the channel as the encoded block $\mathbf{v}_t = \left(\mathbf{u}_t, \hat{\mathbf{v}}_t^{(1)}, \hat{\mathbf{v}}_t^{(2)}\right)$ at time t. Both pipeline [1] and SWD [4] based on the BCJR algorithm have been proposed for BCCs, with SWD allowing lower-latency operation with negligible performance loss.

B. Convolutional Self-orthogonal Codes

A set of J nonnegative integers $A = \{\alpha_1, \alpha_2, \ldots, \alpha_J\}$, $\alpha_1 < \alpha_2 < \ldots < \alpha_J$, $J \geq 2$, is said to be *self-orthogonal* if the differences $(\alpha_j - \alpha_k)$ are distinct for all (j, k), $j \neq k$ [8]. Now assume a *nonrecursive systematic convolutional* (NSC) encoder with rate $R_c = k_c/(k_c + 1)$, memory order m, and constraint length $v = (k_c + 1)\cdot(m + 1)$. If $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_t, \ldots)$ represents the information sequence, where $\mathbf{u}_t = \left(u_t^{(0)}, u_t^{(1)}, \ldots, u_t^{(k_c-1)}\right)$, and $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_t, \ldots)$ represents the encoded sequence, where $\mathbf{v}_t = \left(u_t^{(0)}, u_t^{(1)}, \ldots, u_t^{(k_c-1)}, v_t^{(k_c)}\right)$, then, assuming a binary symmetric channel (BSC), the received sequence is $\mathbf{y} = \mathbf{v} \oplus \mathbf{e}$, where the *error sequence* $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \ldots \mathbf{e}_t, \ldots)$, $\mathbf{e}_t = \left(e_t^{(0)}, e_t^{(1)}, \ldots, e_t^{(k_c)}\right)$, and $e_t^{(0)}, e_t^{(1)}, \ldots, e_t^{(k_c-1)}$ represent the *information error bits* at time t.

Such a code is said to be self-orthogonal if, for each information error bit in $\mathbf{e}_0$, the set of all *syndrome bits* that check it forms an orthogonal set, i.e., no other error bits are checked more than once (see [8] for details). If J orthogonal checks can be formed on each information error bit in $\mathbf{e}_0$, then the code is capable of correcting any pattern of $\lfloor J/2 \rfloor$ errors within one constraint length with simple majority-logic (threshold) decoding.[2] $R_c = k_c/(k_c + 1)$ NSC encoders with memory m can be described by a set of k_c *generator polynomials* $\mathbf{g}^{(1)}(D), \mathbf{g}^{(2)}(D), \ldots, \mathbf{g}^{(k_c)}(D)$, where m is the largest degree of any of the generators. Since, for CSOCs, the generators are typically sparse, they can also be described by the sets $\mathbf{g}^{(1)}, \mathbf{g}^{(2)}, \ldots, \mathbf{g}^{(k_c)}$ of their non-zero coefficients.

[2]Since increasing the size J of the orthogonal set increases the error-correcting capability of the code, larger values of J correspond to stronger codes.

C. Braided Convolutional Self-orthogonal Codes

We now consider nonrecursive systematic CSOCs of rate $R_c = k_c/(k_c + 1)$ as component codes for the braided codes shown in Fig. 1. At time unit $t \in [1, L + \Lambda]$, the braided CSOC code symbols are denoted $\mathbf{v}_t = \left(\mathbf{u}_t^{(0)}, \mathbf{u}_t^{(1)}, \ldots, \mathbf{u}_t^{(k_c-2)}, \hat{\mathbf{v}}_t^{(1)}, \hat{\mathbf{v}}_t^{(2)}\right)$, where each component of $\mathbf{v}_t$ is a block of T symbols and Λ is the number of blocks used to terminate encoding. The rate of the braided CSOCs is $R = (k_c - 1)/(k_c + 1)$.

III. Double Sliding Window Decoding based on BP

In this section, we introduce DSWD based on BP for braided CSOCs. Specifically, there are two windows in the decoding process. One is an outer window, which covers w blocks from time t to time $t + w - 1$. The other is an inner window which operates only on one block at a time.

A. Outer Sliding Window Decoding

A diagram of the proposed outer SWD process for braided CSOCs is shown in Fig. 2, where w represents the window size. At time unit t, a decoding window covers w blocks from time t to $t + w - 1$. There are three types of iterations performed in the window: component iterations, vertical iterations, and horizontal iterations. Component iterations represent iterative BP decoding operating on a component CSOC. Vertical iterations represent the exchange of extrinsic LLRs on the information input symbols between the two component CSOCs at the same time unit. Horizontal iterations represent the exchange of the extrinsic LLRs on the parity output symbols between blocks at neighboring time units in a decoding window. One horizontal iteration includes both a forward horizontal exchange of LLRs from the first block to the last block in the decoding window and a backward horizontal exchange of LLRs from the last block to the first block in the decoding window. Let I_1, I_2, and I_3 denote the maximum number of allowed component iterations, vertical iterations, and horizontal iterations, respectively.

After w blocks enter the window, the decoder is initialized. In decoder 1, for each block at time $s \in [0, w - 1]$, the channel

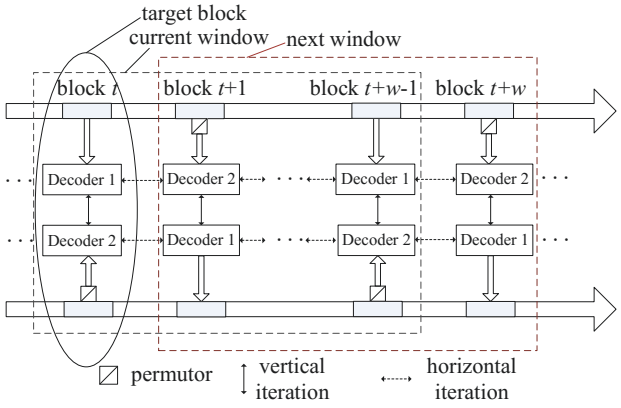

Fig. 2. The outer sliding window decoding process.

log likelihood ratios (LLRs) of the information input symbols are given by

$$\boldsymbol{\lambda}_{\mathrm{Inf}}^{c,(i)}(s)=\mathbf{l}_s^{(i)}, i=0,1,\ldots,k_{\mathrm{c}}-2, \tag{1}$$

the channel LLRs of the parity input symbols are given by

$$\boldsymbol{\lambda}_{\mathrm{Pin}}^{c}(s)=\begin{cases}\Psi, & s=0\\ \mathbf{l}_{s-1}^{(k_c)}\mathbf{P}^{(2)}, & s\in(0,w-1],\end{cases} \tag{2}$$

and the channel LLRs of parity output symbols are given by

$$\boldsymbol{\lambda}_{\mathrm{Pout}}^{c}(s)=\mathbf{l}_s^{(k_{\mathrm{c}}-1)}, \tag{3}$$

where $\mathbf{l}_s^{(i)}$ represents the LLRs of the T received symbols corresponding to the transmitted block $\mathbf{c}_s^{(i)}$ of T code symbols, $i=0,1,\ldots,k_c$, and Ψ is a constant vector of size T in which each component is a large negative value, reflecting the fact that, at time unit $t=0$, the blocks of parity inputs into encoder 1 and encoder 2 are all-zero.

After initialization, a *horizontal iteration* begins that involves the blocks from time t to $t+w-1$ in the window. First, for the block at time t, component decoder 1 performs iterative decoding (see the next subsection for details). After I_1 component iterations, decoder 1 transmits the extrinsic LLRs $\boldsymbol{\lambda}_{\mathrm{Inf}}^{e,(i)}(t)$ of the information input symbols, $i=0,1,\ldots,k_c-2$, to decoder 2. After receiving these extrinsic LLRs, component decoder 2 performs I_1 component iterations and passes the extrinsic LLRs $\tilde{\boldsymbol{\lambda}}_{\mathrm{Inf}}^{e,(i)}(t)$ of the information input symbols, $i=0,1,\ldots,k_c-2$, back to decoder 1, completing a *vertical iteration* as shown in Fig. 3. Thus, for $s\in[t,t+w-1]$, the *a priori* LLRs $\tilde{\boldsymbol{\lambda}}_{\mathrm{Inf}}^{a,(i)}(s)$ of the information input symbols in decoder 2 are given by

$$\tilde{\boldsymbol{\lambda}}_{\mathrm{Inf}}^{a,(i)}(s)=\boldsymbol{\lambda}_{\mathrm{Inf}}^{e,(i)}(s)\,\mathbf{P}_i^{(0)}, i=0,1,\ldots,k_c-2, \tag{4}$$

and the *a priori* LLRs $\boldsymbol{\lambda}_{\mathrm{Inf}}^{a,(i)}(s)$ of the input information symbols in decoder 1 are given by

$$\boldsymbol{\lambda}_{\mathrm{Inf}}^{a,(i)}(s)=\tilde{\boldsymbol{\lambda}}_{\mathrm{Inf}}^{e,(i)}(s)\left(\mathbf{P}_i^{(0)}\right)^{-1}, i=0,1,\ldots,k_c-2, \tag{5}$$

where $\mathbf{P}_i^{(0)}\left(\text{resp. }\left(\mathbf{P}_i^{(0)}\right)^{-1}\right)$ represents the permutation (inverse permutation) applied to $\boldsymbol{\lambda}_{\mathrm{Inf}}^{e,(i)}(s)\left(\tilde{\boldsymbol{\lambda}}_{\mathrm{Inf}}^{e,(i)}(s)\right)$, $i=0,1,\ldots,k_c-2$.

Once the number of vertical iterations for decoder 2 reaches I_2, the extrinsic LLRs $\boldsymbol{\lambda}_{\mathrm{Pout}}^{e}(s)$ of the parity output symbols of decoder 1 at time s are permuted by $\mathbf{P}^{(1)}$ and sent to decoder 2 as the *a priori* LLRs $\tilde{\boldsymbol{\lambda}}_{\mathrm{Pin}}^{a}(s+1)$ of the parity input symbols at time $s+1$. In the same way, the extrinsic LLRs $\tilde{\boldsymbol{\lambda}}_{\mathrm{Pout}}^{e}(s)$ of the parity output symbols of decoder 2 are permuted by $\mathbf{P}^{(2)}$ and sent to decoder 1 as the *a priori* LLRs $\boldsymbol{\lambda}_{\mathrm{Pin}}^{a}(s+1)$ of the parity input symbols at time $s+1$. Block $s+1$ now performs vertical decoding.

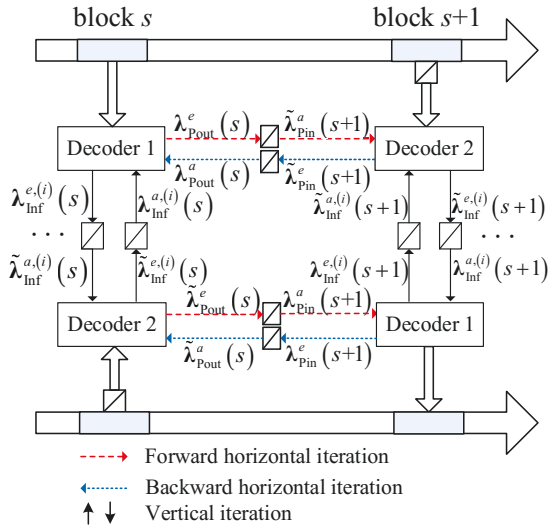

Fig. 3. Extrinsic information exchange in a vertical/horizontal iteration.

When the last block at time $s=t+w-1$ in the current window finishes I_2 vertical iterations, the *backward horizontal iteration* begins. Using the permutor inverses $\left(\mathbf{P}^{(2)}\right)^{-1}$ and $\left(\mathbf{P}^{(1)}\right)^{-1}$, respectively, the decoders at time $s=t+w-1$ send the extrinsic LLRs $\boldsymbol{\lambda}_{\mathrm{Pin}}^{e}(t+w-1)$ and $\tilde{\boldsymbol{\lambda}}_{\mathrm{Pin}}^{e}(t+w-1)$ of the parity input symbols back to the component decoders as the *a priori* LLRs $\tilde{\boldsymbol{\lambda}}_{\mathrm{Pout}}^{a}(t+w-2)$ and $\boldsymbol{\lambda}_{\mathrm{Pout}}^{a}(t+w-2)$ of the parity output symbols at time $s=t+w-2$. The decoders at time $s=t+w-2$ then perform I_2 vertical iterations. This process continues until the first block at time $s=t$ in the window finishes I_2 vertical iterations, thus completing a full horizontal iteration. This horizontal iteration process is illustrated in Fig. 3. Finally, after the allowed number of vertical and horizontal iterations have been performed, decisions are made on the target block of $(k_c-1)T$ information symbols and the window shifts to the next position (see Fig. 2).

B. Inner Sliding Window Decoding

The self-orthogonal structure of CSOCs guarantees that there are no 4-cycles in the Tanner graph of the code. Based on this observation, we propose to use sliding window BP decoding on the component CSOCs. At any time $t\in[0,L+\Lambda]$, a component decoder in a braided CSOC contains $(k_{\mathrm{c}}+1)\cdot T$ received symbols, i.e., there are T groups of received symbols, where each group includes $k_{\mathrm{c}}-1$ information input symbols, 1 parity input symbol, and 1 parity output symbol. A diagram of the parity check matrix $\mathbf{H}_t$ and the SWD process for a component decoder is shown in Fig. 4, where the extra $(k_{\mathrm{c}}+1)\cdot m$ columns in $\mathbf{H}_t$ that overlap the next time unit are needed to decode the last group at time t. Therefore, the size of the parity-check matrix $\mathbf{H}_t$ of a component decoder at any time unit is $(T+2m)\times[(k_{\mathrm{c}}+1)\cdot(T+m)]$. BP decoding can be performed on this entire matrix using a conventional flooding schedule. However, in order to reduce the computational complexity and memory requirements of the

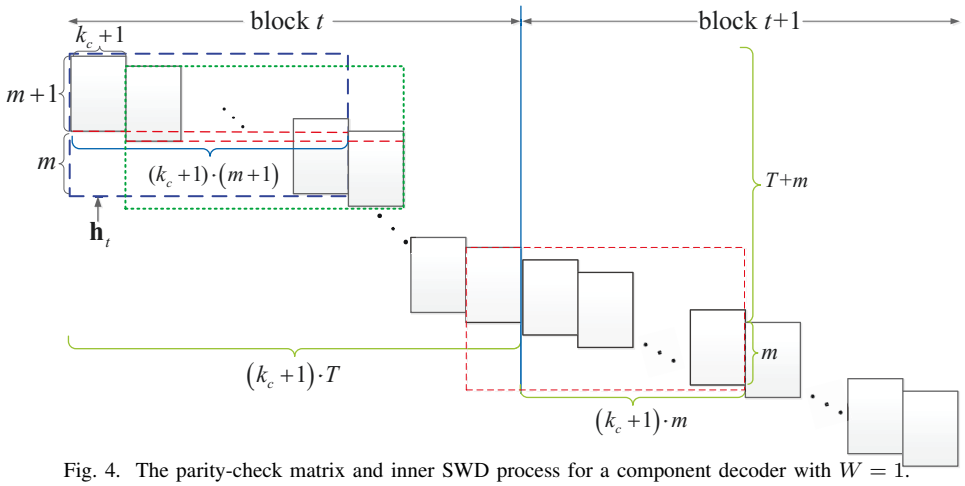

Fig. 4. The parity-check matrix and inner SWD process for a component decoder with $W = 1$.

decoder, we propose instead to process a small parity-check matrix $\mathbf{h}_t$ of size $[W(m+1)+m] \times [W(k_c+1)\cdot(m+1)] = [W(m+1)+m] \times W\nu$ in a sliding window fashion, where W, which represents the number of constraint lengths of received symbols in the window, is typically a small positive integer. This small window covers $W(m+1)$ groups of symbols, with the first group being the target group. When the target group is decoded, the small window shifts by one group as shown in Fig. 4 (drawn for $W = 1$). After the last group in block t is decoded, component decoder 1 (or 2) at time t transmits the extrinsic LLRs of the $(k_c - 1)T$ information symbols to component decoder 2 (or 1) at time t, as well as the extrinsic LLRs of the T parity output symbols to component decoder 2 (or 1) at time $t+1$.

IV. Numerical Results

We now investigate the performance of braided CSOCs with DSWD based on BP over the binary-input AWGN channel with BPSK modulation.

Example 1: Consider a rate $R = \frac{1}{3}$ braided CSOC using a component systematic CSOC with rate $R_c = 2/3$, $m = 13$, $J = 4$, and generator polynomials $\mathbf{g}^{(1)} = \{0, 8, 9, 12\}$ and $\mathbf{g}^{(2)} = \{0, 6, 11, 13\}$ [8]. We first investigate the effect of the permutor size T on the bit error rate (BER) performance, as shown in Fig. 5. We observe that, with increasing T, the BER performance improves in the waterfall region as well as in the error floor region.

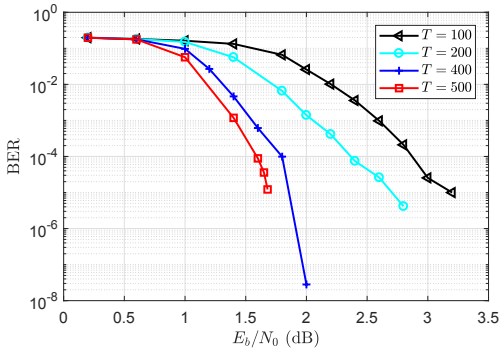

Fig. 5. The BER of $R = 1/3$ braided CSOCs with different permutor sizes T and $J = 4$, $w = 4$, $W = 1$, $I_1 = 3$, $I_2 = 1$, and $I_3 = 5$.

Example 2: We next investigate the effect of the size J of the orthogonal set on the BER performance. Rate $R = 1/3$ braided CSOCs are again considered with three $R_c = 2/3$ component codes: (1) $m = 2$, $J = 2$, $\mathbf{g}^{(1)} = \{0, 1\}$, $\mathbf{g}^{(2)} = \{0, 2\}$; (2) $m = 13$, $J = 4$, $\mathbf{g}^{(1)} = \{0, 8, 9, 12\}$, $\mathbf{g}^{(2)} = \{0, 6, 11, 13\}$; and (3) $m = 40$, $J = 6$, $\mathbf{g}^{(1)} = \{0, 2, 6, 24, 29, 40\}$, $\mathbf{g}^{(2)} = \{0, 3, 15, 28, 35, 36\}$ [8]. The results are shown in Fig. 6 for $T = 400$. We see that, using the proposed decoding algorithm, the braided CSOC with $J = 2$ performs best in the waterfall, and we note that increasing J increases the strength of the code[3], and thus has the effect of weakening the waterfall performance. Larger values of J, however, have better error floor performance.

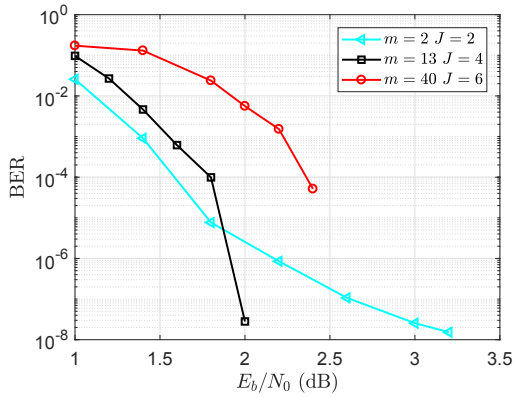

Fig. 6. The BER of $R = 1/3$ braided CSOCs with different orthogonal set sizes J with $w = 4$, $W = 1$, $I_1 = 3$, $I_2 = 1$, and $I_3 = 5$.

Example 3: Using the $R_c = 2/3$ component CSOC with $m = 2$, $J = 2$ and generator polynomials $\mathbf{g}^{(1)} = \{0, 1\}$, and $\mathbf{g}^{(2)} = \{0, 2\}$ [8], the performance of the $R = 1/3$ braided CSOC based on BP decoding is compared to an $R = 1/3$ BCC based on BCJR decoding with $R_c = 2/3$ RSC component codes [4] and to an $R = 1/3$ spatially coupled LDPC (SC-LDPC) code with sliding window (SW) decoding and the same decoding latency. The results are shown in Fig. 7, where we see that the performance of the braided

[3]Increasing J corresponds to increasing both the error-correcting capability of the component code and the degree of each information variable node (VN) in the Tanner graph of the parity-check matrix (the parity VN has degree 1).

CSOCs with suboptimal, low-complexity BP decoding of the component codes is competitive (within 0.2 dB) with the BCCs with optimal, high-complexity BCJR decoding and compares favorably with the SC-LDPC code with SW decoding. Also shown in Fig. 7 is the BER of braided CSOCs with inner decoding based on Massey's original APP threshold decoding algorithm [5], which further lowers decoding complexity (the component decoders are non-iterative) at a cost of performance (see [13] for details of applying APP decoding in a turbo-like configuration).

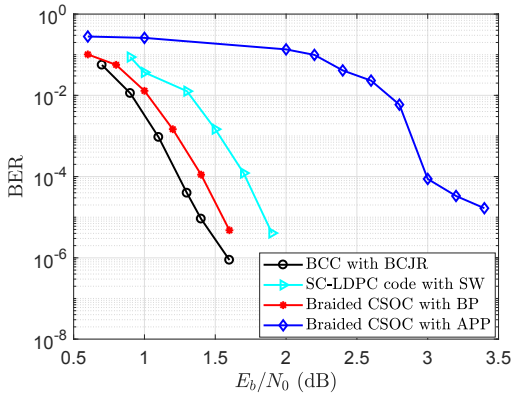

Fig. 7. The BER performance of $R = 1/3$ BCCs with $T = 100$ and $w = 16$ for (1) BP decoded braided CSOCs with $W = 1$, $I_1 = 2$, $I_2 = 5$, and $I_3 = 30$, (2) BCJR decoded BCCs with $I_1 = 2$, $I_2 = 5$, and $I_3 = 30$, and (3) APP decoded braided CSOCs with $I_2 = 5$, and $I_3 = 30$. Also shown is the BER performance of an SW decoded (4,6)-regular SC-LDPC code with $R = 1/3$, $m_s = 1$, $w = 16$, $M = 100$, and $I_{\max} = 200$.

Example 4: To illustrate the ease with which high-rate BCCs can be constructed using CSOC component codes, Fig. 8 shows the BER performance of braided CSOCs with (i) a rate $R = 7/9$ braided CSOC using an $R_c = 8/9$ component CSOC with $m = 77$ and $J = 3$, (ii) a rate $R = 9/11$ braided CSOC using an $R_c = 10/11$ component CSOC with $m = 77$ and $J = 3$, and (iii) a rate $R = 9/10$ braided CSOC using an $R_c = 19/20$ component CSOC with $m = 168$ and $J = 3$. (The generator polynomials of the component CSOCs were taken from [6].) Since CSOCs with rates as high as $R_c = 49/50$ have been published in the literature [7], this example illustrates the advantage of using braided CSOCs to achieve good performance in both the waterfall and the error floor with modest decoding complexity, a task that would be much more complex at very high rates with conventional BCCs decoded using the BCJR algorithm.[4]

V. Conclusion

In this paper, we presented a new approach to the design of BCCs with an emphasis on high rate constructions and low complexity decoding. In particular, we proposed the use of CSOCs as component codes along with a novel low-complexity DSWD method. Results indicate that braided CSOCs with DSWD offer competitive performance with less complexity compared to conventional BCCs with BCJR decoding, with the complexity advantage being particularly significant at high rates. Ongoing work includes examining

[4]Using the dual representation of convolutional codes and decoding on the dual trellis would be required in this case.

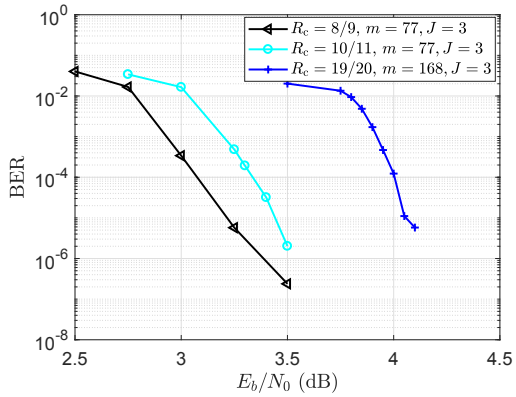

Fig. 8. The BER of three high-rate braided CSOCs with $T = 200$, $w = 4$, $W = 1$, $I_1 = 2$, $I_2 = 1$, and $I_3 = 5$.

the performance/complexity tradeoffs involved in increasing the iteration limit I_1 and the window size W of component SWD. Also, the use of CSOCs with nonsystematic $\mathbf{H}$ matrices and RSC encoders is being investigated as a way of improving the convergence of the component BP decoder by increasing the degree of the parity VN, resulting in a fully regular Tanner graph, thus improving the reliabilities (LLRs) of the parity symbols that are passed between blocks in a horizontal iteration.

Acknowledgment

This material is supported in part by the NSFC under Grant 62271380 and the National Science Foundation under Grant Nos. CNS-2148358 and OIA-1757207.

References

[1] 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.

[2] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Finite length weight enumerator analysis of braided convolutional codes," in *Proc. Int. Symp. Inf. Theory and Its Applications*, Monterey, CA, USA, Oct. 30-Nov. 2, 2016, pp. 488-492.

[3] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-like codes: A new trade-off between waterfall and error floor," *IEEE Trans. on Commun.*, vol. 67, no. 5, pp. 3114-3123, May 2019.

[4] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, Jr., and B. Bai, "Braided convolutional codes with sliding window decoding," *IEEE Trans. on Communications*, vol. 65, no. 9, pp. 3645-3658, Sept. 2017.

[5] J. L. Massey, "Threshold decoding," Cambridge, M.I.T. Press, 1963.

[6] W. W. Wu, "New convolutional codes-part I", *IEEE Trans. on Communications*, vol. com-23, No. 9, pp. 942-955, Sep. 1975.

[7] W. W. Wu, "New convolutional codes-part II", *IEEE Trans. on Communications*, vol. com-23, No. 9, pp. 19-33, Jan. 1976.

[8] S. Lin and D. J. Costello, Jr., "Error Control Coding: Fundamentals and Applications," 2nd Ed., Upper Saddle River, NJ: Prentice-Hall, 2004.

[9] M. Belkasmi, M. Lahmer, and M. Benchrifa, "Iterative threshold decoding of parallel concatenated block codes," in *Proc. Int. Symp. on Turbo Codes & Related Topics*, Munich, Germany, April 3-7, 2006, pp. 4-7.

[10] C. Cardinal, D. Haccoun, and F. Gagnon, "Iterative threshold decoding without interleaving for convolutional self-doubly orthogonal codes," *IEEE Trans. Commun*, vol. 51, no. 8, pp.1274-1282, Aug. 2003.

[11] R. Lucas, M.P.C. Fossorier, Yu Kou, and Shu Lin, "Iterative decoding of one-step majority logic deductible codes based on belief propagation," *IEEE Trans. on Communications*, vol. 48, no. 6, pp.931-937, June 2000.

[12] Y. He and D. Haccoun, "Analysis of the orthogonality structures of convolutional codes for iterative decoding," *IEEE Trans. Inf. Theory*, vol. 51, no. 9, pp. 3247-3261, Sep. 2005.

[13] A. D. Cummins, D. G. M. Mitchell, and D. J. Costello, Jr., "Iterative threshold decoding of spatially coupled parallel-concatenated codes," in *Proc. Int. Symp. on Topics in Coding*, Montreal, QC, Canada, Aug. 30 - Sept. 3, 2021, pp. 1-5.