Turbo Code Design for Short Blocks

Thomas Jerkovits, Balázs Matuz

***Abstract*—This work considers the design of short parallel turbo codes (PTCs) with block lengths in the order of (a few) hundred code bits. In particular we aim at designing codes with large minimum distance. To this end a structured approach is presented to find suitable component code configurations, as well as interleavers. As a result the proposed turbo codes possess low error floors and outperform competing binary low-density parity-check (LDPC) codes by approximately $0.9\,\mathrm{dB}$ and state-of-the-art binary turbo codes by $0.4\,\mathrm{dB}$ at a code word error rate (CER) of about $\approx 10^{-7}$. The loss w.r.t. the random coding bound (RCB) is only about $0.8\,\mathrm{dB}$.**

I. Introduction

The need of powerful short and moderate length codes is currently raising due to the developments in machine-type communications, where short data packets are often used, especially over random access channels [1]. In the context of satellite communication systems, short codes are typically used over telecommand links [2]. The design of long channel codes for iterative decoding is well-established and based on a set of performance metrics, such as the iterative decoding threshold [3], or the ensemble typical minimum distance [4]. Those performance metrics which are derived for code ensembles under the assumption of infinitely long codewords allow to restrict the search of good codes at an early stage of the code design process.

For moderate and short block lengths, though, asymptotic performance indicators can become inaccurate in predicting which ensembles contain good codes. Furthermore, the search of good codes within the selected ensembles plays a more important role with respect to the large block length setting. This is done, for instance, for low-density parity-check (LDPC) codes via semi-heuristic algorithms, such as the progressive edge growth [5] and the approximate cycle extrinsic message degree algorithms [6] used to construct bipartite graphs with large girths. Regarding PTCs, once the ensemble, i.e., the component codes are fixed, a good code construction requires a careful design of the interleaver and, if needed, of the puncturing patterns applied to the code bits.

Among the most powerful solutions for short blocks, non-binary LDPC [7] codes and PTCs [8] constructed over large order finite fields gained a prominent role thanks to their excellent performance down to very low error rates. The price to pay is a relatively high decoding complexity. It is well established that for large blocks modern channel codes, such as binary low-density parity-check (LDPC) codes [4], PTCs [9], or polar codes [10] provide excellent performance on various communication channels. For the particular case of polar codes it has been recently proven that they are capacity achieving on any binary-input discrete memoryless channel (DMC) [10]. When considering short blocks, modern codes tend to exhibit either high error floors or non-negligible coding gain losses [11] w.r.t. existing benchmarks such as the RCB [12].

T. Jerkovits, and B. Matuz are with Institute of Communication and Navigation of the Deutsches Zentrum für Luft- und Raumfahrt (DLR), 82234 Wessling, Germany (e-mail: {thomas.jerkovits, balazs.matuz}@dlr.de).

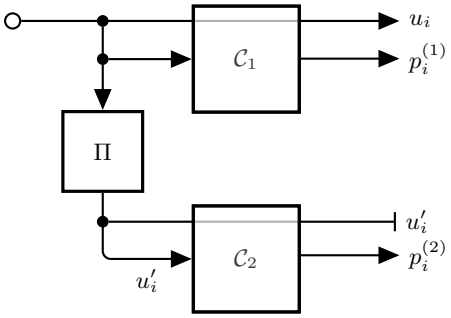

Fig. 1. Turbo encoder.

Interleaver design for short PTCs has been investigated in [13] and an improvement of S-random interleaver has been presented. In [14], it was shown that binary 16-states turbo codes, though not achieving the performance of non-binary codes, attain a small performance loss ($\approx 0.3\,\mathrm{dB}$) at moderate CERs ($\approx 10^{-3}$) while at lower error rates error floors arise. Likewise, to improve the code performance one may elaborate on the decoding algorithms for the aforementioned codes. For instance, approximate maximum-likelihood (ML) soft-decision decoders [15], [16] are shown to yield a further performance boost [14]. Yet, those decoding algorithms still turn out to be too complex for many applications. The same holds for list decoders recently re-introduced in the setting of successive cancellation decoding of polar codes [17].

In this paper we focus on binary PTCs. They have the advantage of low complexity decoding. Further we illustrate that a structured code design yields very good performances for short/moderate block lengths together with remarkable low error floors.

This paper is structured as follows. Section II gives an introduction to the considered class of turbo codes and provides the necessary notation. In Section III we illustrate component code, puncturing and interleaver choice. Numerical results are then provided in Section IV, followed by a wrap-up in Section V.

II. Preliminaries

A. Parallel Turbo Codes

In this work we focus on PTCs, i.e. in our case a parallel concatenation of two tail-biting recursive systematic convolutional (RSC) codes. Denote the binary information sequence of length K at the first encoder input by $\mathbf{u} = (u_0, \ldots, u_{K-1})$

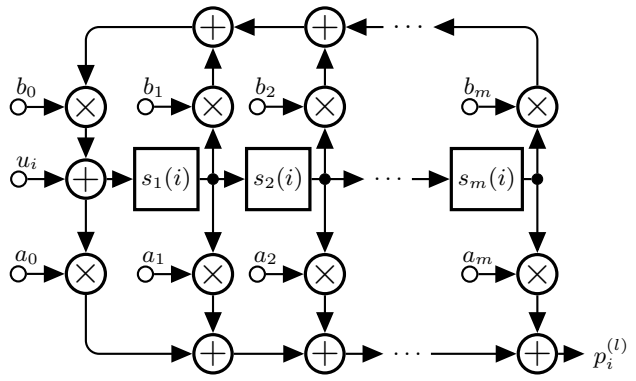

Fig. 2. Recursive convolutional encoder structure of the l-th component code.

and its permuted version at the second encoder input by $\mathbf{u}' = (u'_0, \ldots, u'_{K-1})$. Further, $\mathbf{c}^{(1)} = (\mathbf{u}, \mathbf{p}^{(1)})$ is a codeword of the first (N_1, K) component code $\mathcal{C}_1$ and by $\mathbf{c}^{(2)} = (\mathbf{u}', \mathbf{p}^{(2)})$ a codeword of the second (N_2, K) component code $\mathcal{C}_2$. As depicted in Figure 1 the (N, K) PTC codeword $\mathbf{c}$ is finally formed by the concatenation

$$\mathbf{c} = (\mathbf{u}, \mathbf{p}^{(1)}, \mathbf{p}^{(2)})$$

where we assume that the permuted information symbols $\mathbf{u}'$ are not transmitted. The resulting code rate R is given by

$$R = \frac{K}{N_1 + N_2 - K}\,.$$

Assuming rate $1/2$ component codes, this construction allows code rates of $R = 1/3$. To obtain PTCs of higher rate, puncturing of the PTC codewords is required. In this paper we restrict to puncturing the non-systematic part of the RSC component codes equally, such that $N_1 = N_2$. Due to the puncturing it follows that $N_1 < 2K$ and $N_2 < 2K$ and our component codes $\mathcal{C}_1$ and $\mathcal{C}_2$ in Figure 1 become *time-variant*.

Throughout this work we use the following notations for the Hamming weight of the systematic part and the parity parts of a PTC codeword

$$\begin{aligned} w &\triangleq \mathrm{w_H}(\mathbf{u}) = \mathrm{w_H}(\mathbf{u}') \\ r_1 &\triangleq \mathrm{w_H}(\mathbf{p}^{(1)}) \\ r_2 &\triangleq \mathrm{w_H}(\mathbf{p}^{(2)}). \end{aligned}$$

Furthermore, we define the interleaving rule of the interleaver $\mathbf{\Pi}$ as follows. Let the interleaver $\mathbf{\Pi} = (\Pi(0), \ldots, \Pi(K-1))$ be a vector of length K containing indices $0, \ldots, K-1$ in a predefined order. The relation between u_i and u'_i is then given by

$$u'_{\Pi(i)} = u_i,\ \forall i = 0, 1, \ldots, K-1\,.$$

B. Component Code Description

Figure 2 depicts the recursive convolutional encoder structure for the parity output $\mathbf{p}^{(l)} = (p_0^{(l)}, p_1^{(l)}, \ldots, p_{K-1}^{(l)})$ of the l-th component code. The content of the vector $\mathbf{s}(i) = (s_1(i), s_2(i), \ldots, s_m(i))$ defines the state of the encoder at time step i. The binary vector $\mathbf{b} = (b_0, b_1, \ldots, b_m)$ contains the feedback coefficients and the vector $\mathbf{a} = (a_0, a_1, \ldots, a_m)$ the feed-forward coefficients.

We use the right-justified octal notations to represent the feedback and feed-forward coefficients by an integer. For example consider an RSC code with memory $m = 4$ and $\mathbf{b} = (1, 0, 1, 1, 1)$. The right-justified binary notation of the feedback coefficients is thus 010111 and the octal integer representation is 27. In the following, we consider the case of periodically punctured component codes. Periodic puncturing can be obtained by employing a time-variant encoder. To describe the feedback and feed-forward coefficients at time step i when no parity bit is punctured we write

$$\mathbf{g}(i) = [\mathbf{b}, \mathbf{a}]$$

while if the parity bit is punctured at time step i we write

$$\mathbf{g}(i) = [\mathbf{b}].$$

A time-variant RSC code is said to be periodic with period T if $\mathbf{g}(i) = \mathbf{g}(i+T)$ for all $i = 0, 1, \ldots, K-T-1$ [18]. To denote a time-variant RSC code with period T we write

$$\mathbf{g}_T = (\mathbf{g}(0), \mathbf{g}(1), \ldots, \mathbf{g}(T-1)).$$

We call $\mathbf{g}_T$ the descriptor of a time-variant RSC code.

Moreover, we use a tail-biting technique for the RSC codes as described in [19] where we impose the constraint that initial state $\mathbf{s}(0)$ equals the last state $\mathbf{s}(K-1)$ after the message $\mathbf{u}$ has been encoded.

In the following, we focus on a specific design which targets a $(128, 64)$ PTC. We consider the period $T = 2$ puncturing pattern of the Consultative Committee for Space Data Systems (CCSDS) standard in [20]. Puncturing patterns with larger periodicity are also appealing [21]. However, by increasing the periodicity the search space also grows. We denote by $\mathbf{g}_2^{(1)} = (\mathbf{g}^{(1)}(0), \mathbf{g}^{(1)}(1))$ and $\mathbf{g}_2^{(2)} = (\mathbf{g}^{(2)}(0), \mathbf{g}^{(2)}(1))$ the descriptors of the first and second component code, respectively. We have for the first component code

$$\mathbf{g}_2^{(1)} = ([\mathbf{b}, \mathbf{a}], [\mathbf{b}]) \tag{1}$$

and for the second component code

$$\mathbf{g}_2^{(2)} = ([\mathbf{b}], [\mathbf{b}, \mathbf{a}]). \tag{2}$$

C. Turbo Code Ensemble

Definition 1 (Uniform Turbo Code Ensemble): Let two component codes $\mathcal{C}_1$ and $\mathcal{C}_2$ with information block size K be given. Define the set

$$\mathscr{C} = \{\mathcal{C}_1^{\mathsf{TC}}, \mathcal{C}_2^{\mathsf{TC}}, \mathcal{C}_3^{\mathsf{TC}}, \ldots, \mathcal{C}_{K!}^{\mathsf{TC}}\}$$

where $\mathcal{C}_i^{\mathsf{TC}}$ for $i = 1, \ldots, K!$ are PTCs obtained by all $K!$ possible interleavers. The uniform turbo code ensemble (TCE) is given by $\mathscr{C}$, where each element of $\mathscr{C}$ is picked with probability $\frac{1}{K!}$.

The average weight enumerator for the uniform TCE is known to be

$$\bar{A}_d = \sum_{w=0}^{K} \sum_{r_1+r_2=d-w} \frac{A_{w,r_1}^{(1)} \cdot A_{w,r_2}^{(2)}}{\binom{K}{w}}$$

where $\bar{A}_d$ is the expected number of codewords with weight d in the TCE and $A^{(1)}_{w,r_1}$ and $A^{(2)}_{w,r_2}$ are the input-redundancy-weight enumerators for the component codes $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

D. Union Bound

The block error probability $P_B(\mathcal{C})$ of a (N, K) block code $\mathcal{C}$ with weight enumerator A_d under ML decoding can be upper bounded by an union bound (UB) as

$$P_B(\mathcal{C}) \leq \frac{1}{2} \cdot \sum_{d=1}^{N} A_d \cdot \operatorname{erfc}\left(\sqrt{d \cdot R \cdot \frac{E_b}{N_0}}\right).$$

The UB above can also be applied on a TCE with average weight enumerators $\bar{A}_d$. It is of interest, since it guarantees that there exists at least one code in the ensemble that achieves for a certain ${}^{E_b}\!/_{N_0}$ the codeword error probability obtained by the UB [22]. We use this UB on a TCE in the next section for the choice of our component codes.

III. Turbo Code Design

A. Ensemble Search

We want to identify suitable code ensembles for given (N, K). In particular we aim at low codeword error probabilities at high values of ${}^{E_b}\!/_{N_0}$. In a first step we proceed as follows.

- Randomly generate a set of (time variant) component codes.
- For each component code in the set generate a second component code. In our case the descriptors have the form as in (1) and (2).
- Each pair of component codes in the set forms a (uniform) TCE. For each TCE compute the corresponding average weight enumerators $\bar{A}_d$.
- Evaluate the UB on the uniform TCEs at a sufficiently low target codeword error probability P_B°. In our case $P_B^{\circ} = 10^{-8}$.
- Obtain $({}^{E_b}\!/_{N_0})^{\circ}$ for which the UB achieves P_B°.
- Order the set of uniform TCEs according to their $({}^{E_b}\!/_{N_0})^{\circ}$ values in an increasing order.

In a second step we evaluate the iterative decoding thresholds $({}^{E_b}\!/_{N_0})^{*}$ of the TCEs in the set [23]. We start with the first TCE in the ordered set, i.e., the one with lowest $({}^{E_b}\!/_{N_0})^{\circ}$. If $({}^{E_b}\!/_{N_0})^{*}$ is above a predefined threshold, we discard the ensemble and take the next TCE in the ordered set. We then check again for the iterative decoding threshold and proceed until we obtain a component configuration that yields a $({}^{E_b}\!/_{N_0})^{*}$ below the predefined threshold (and has a low $({}^{E_b}\!/_{N_0})^{\circ}$). If the search does not yield a single TCE we repeat the two design steps above and increase $({}^{E_b}\!/_{N_0})^{*}$. We performed a search for $(N = 128, K = 64)$, where puncturing is done by making use of time variant component codes as described in Section II. We restricted the memory m of the component encoders to $m \in \{3, 4, 5\}$. The result of that search is summarized in Table I showing the memory m of the component codes, the descriptors as $\mathbf{g}^{(1)}_{T=2}$ for the first component code and $\mathbf{g}^{(2)}_{T=2}$ for the second component code, the minimum distance $d^{(1)}_{\mathsf{min}}$ and $d^{(2)}_{\mathsf{min}}$ of the component codes, the iterative decoding threshold $({}^{E_b}\!/_{N_0})^{*}$ and in the last column the optimization parameter $({}^{E_b}\!/_{N_0})^{\circ}$ from the UB. Observe that increasing the memory size only slightly worsens the threshold, but considerably improves the $({}^{E_b}\!/_{N_0})^{\circ}$, thus the average distance spectrum of the ensemble. In Section IV this will be underpinned by CER performances.

TABLE I
Resulting component codes from the ensemble search.

m	$\mathbf{g}^{(1)}_{T=2}, \mathbf{g}^{(2)}_{T=2}$	$d^{(1)}_{\mathsf{min}}/d^{(2)}_{\mathsf{min}}$	$({}^{E_b}\!/_{N_0})^{*}$	$({}^{E_b}\!/_{N_0})^{\circ}$
3	$([013, 015], [013])$, $([013], [013, 015])$	4 4	0.62 dB	7.5 dB
4	$([023, 033], [023])$, $([023], [023, 033])$	5 5	0.63 dB	6.5 dB
5	$([067, 045], [067])$, $([067], [067, 045])$	5 5	0.77 dB	5.5 dB

Algorithm 1 Algorithm to test an interleaver for the d°_{min} constraint.

Require:
Interleaver: $\mathbf{\Pi}$
Target minimum distance: d°_{min}
Set of harmful patterns of $\mathcal{C}_1$: $\mathcal{U}^{(1)}_{w,r_1}$ s.t. $w + r_1 < d^{\circ}_{\mathsf{min}}$
Second component code: $\mathcal{C}_2$

for $w \in 1, \ldots, d^{\circ}_{\mathsf{min}} - 1$ **do**
  for $r_1 \in \{0, \ldots, d^{\circ}_{\mathsf{min}} - w - 1\}$ **do**
    for $\mathbf{u} \in \mathcal{U}^{(1)}_{w,r_1}$ **do**
      $u'_{\Pi(i)} = u_i, \quad \forall i = 0, \ldots, K-1$
      $\mathbf{c}^{(2)} = (\mathbf{u}', \mathbf{p}^{(2)})$ ▷ encode $\mathbf{u}'$ by $\mathcal{C}_2$
      $r_2 = \mathrm{w_H}(\mathbf{p}^{(2)})$
      if $w + r_1 + r_2 < d^{\circ}_{\mathsf{min}}$ **then**
        return "discard $\mathbf{\Pi}$."
      end if
    end for
  end for
end for
return "$\mathbf{\Pi}$ fulfills d°_{min} constraint."

B. Interleaver Design

The interleaver plays an important role for the turbo codes minimum distance and thus also for its error floor performance. We consider two classes of interleavers from literature and select the interleaver parameters, in a way that the code's minimum distance is maximized (within our optimization). We call those interleavers code matched interleavers. The first class of interleavers are S-random interleavers [24] which spread neighboring input symbols apart after interleaving. Since our component code configurations use tail-biting RSC codes the properties of the interleaver are defined in a circular sense.

Definition 2 (Circular Minimum Spread Constraint): Let an interleaver $\mathbf{\Pi}$ of size K be given. The interleaver has circular

spread S if and only if

$$\mathrm{d}_L\left(i_1, i_2\right) < S \Rightarrow \mathrm{d}_L\left(\Pi\left(i_1\right), \Pi\left(i_2\right)\right) \geq S$$

$\forall i_1, i_2 \in \{0, 1, \ldots, K-1\}$ and $\mathrm{d}_L(i_1, i_2)$ being the distance of two indices i_1, i_2 in Lee metric defined as

$$\mathrm{d}_L\left(i_1, i_2\right) = \min\left(\left|i_1 - i_2\right|, K - \left|i_1 - i_2\right|\right).$$

An interleaver that guarantees to have at least a circular minimum spread S_{min}, thus

$$S \geq S_{\mathsf{min}}$$

satisfies the circular minimum spread constraint.

Due to the absence of structure the search space for S-random interleavers can become large and we may not always find a suitable interleaver. We therefore consider also dithered relative prime (DRP) interleavers [25]. The amount of randomness can be controlled by the size of the read and write *dither* namely $\tilde{\mathbf{r}} = (\tilde{r}_0, \ldots, \tilde{r}_{R-1})$ and $\tilde{\mathbf{w}} = (\tilde{w}_0, \ldots, \tilde{w}_{W-1})$ to be chosen freely. The interleaver is completely defined by

$$\Pi_{\mathsf{DRP}}(i) = \Pi_a(\Pi_b(\Pi_c(i))),\ \forall i \in \{0, \ldots, K-1\}.$$

We have that

$$\begin{aligned}
\Pi_a(i) &= R\left\lfloor \frac{i}{R} \right\rfloor + \tilde{r}_{(i \bmod R)} \\
\Pi_b(i) &= (\tilde{s} + \tilde{p} \cdot i) \bmod K \\
\Pi_c(i) &= W\left\lfloor \frac{i}{W} \right\rfloor + \tilde{w}_{(i \bmod W)}
\end{aligned}$$

with $\tilde{s} = 0, \ldots, K-1$ and $\gcd(\tilde{p}, K) = 1$, where $\gcd(a, b)$ denotes the greatest common divisor of two integers a and b. In the sequel we discuss how to find good interleavers under the S-random and DRP constraints.

We introduce basic concepts for the interleaver design. Let $\mathcal{U}_{w,r}^{(l)}$ be the set of input sequences of Hamming weight w for which the l-th component encoder generates codewords of parity Hamming weight r_l, i.e.,

$$\mathcal{U}_{w,r_l}^{(l)} = \{\mathbf{u} |\, \mathrm{w_H}\left(\mathbf{u}\right) = w, \mathrm{w_H}\left(\mathbf{p}^{(l)}\right) = r_l\}.$$

We can now formalize the interleaver constraint used for our code design.

Definition 3 (Minimum Distance Constraint): An interleaver guarantees to yield a lower bound d_{min}° on the minimum distance d_{min} of the resulting PTC, i.e. $d_{\mathsf{min}} \geq d_{\mathsf{min}}^{\circ}$, if

$$\forall \mathbf{u} \in \mathcal{U}_{w,r_1}^{(1)} \Rightarrow \mathbf{u}' \notin \mathcal{U}_{w,r_2}^{(2)}$$

s.t. $w + r_1 + r_2 < d_{\mathsf{min}}^{\circ}$.

Hence, a straightforward approach to the interleaver design is to discard all interleavers for which the condition provided by Definition 3 is not fulfilled. This can be achieved by checking, for all $\mathbf{u} \in \mathcal{U}_{w,r_1}^{(1)}$ with $w + r_1 < d_{\mathsf{min}}^{\circ}$, if the interleaved vector $\mathbf{u}'$ belongs to $\mathcal{U}_{w,r_2}^{(2)}$, with $r_2 < d_{\mathsf{min}}^{\circ} - (w + r_1)$. This approach is summarized in Algorithm 1. Instead of testing if $\mathbf{u}'$ belongs to $\mathcal{U}_{w,r_2}^{(2)}$ we may encode the interleaved input vector $\mathbf{u}'$ to obtain r_2. If $w + r_1 + r_2 < d_{\mathsf{min}}^{\circ}$ the testing procedure is stopped and we know that the resulting PTC produces at least one codeword with output weight smaller than d_{min}° and thus the interleaver does not fulfill the d_{min}° constraint.

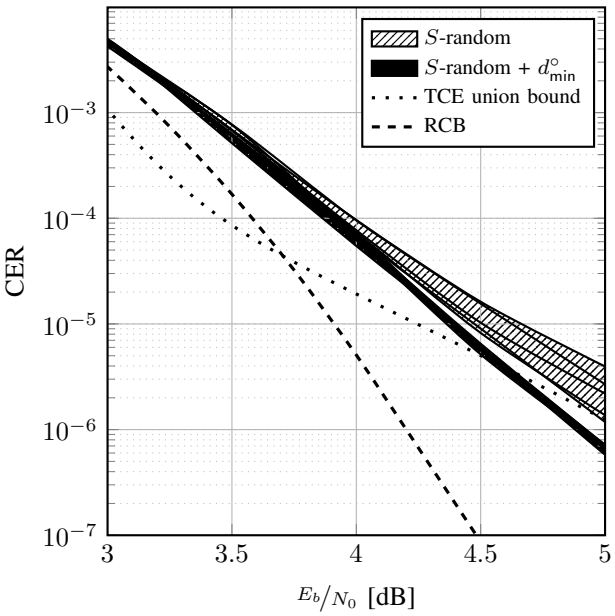

Fig. 3. PTCs peformances for S-random and code matched S-random interleavers.

IV. Numerical Results

A. S-Random Interleaver with d_{min}° Constraint

We exemplify how to obtain good S-random interleavers for a $(128, 64)$ memory $m = 4$ PTC. However, our technique is also applicable to other code rates, block lengths and different types of interleavers. For the design we choose $d_{\mathsf{min}}^{\circ} = 12$ and identify all harmful input sequences of the first component code, i.e., all vectors in the sets $\mathcal{U}_{w,r_1}^{(1)}(t_1)$, s.t. $w + r_1 < 12$. The component codes are specified in Table I as a result of our ensemble search. We generate S-random interleavers with $S_{\mathsf{min}} = 5$ and apply Algorithm 1 to discard those not fulfilling d_{min}°. We call the resulting interleavers code matched S-random interleavers. In Figure 3 we present CERs versus E_b/N_0 for five PTCs obtained by this approach. As reference the RCB for rate-$1/2$ block codes with dimension 64 is depicted as well [12]. Likewise, we designed five PTCs with $S_{\mathsf{min}} = 5$, but without a d_{min}° constraint. Observe that for PTCs with standard S-random interleavers the error floor already starts to show up at a $E_b/N_0 \approx 4.5\,\mathrm{dB}$, whereas the slope of the simulation curves of PTCs with code matched S-random interleavers shows lower error floors. These simulations suggest that by selecting an interleaver only w.r.t. its minimum spread is not sufficient to obtain a PTC with excellent distance properties for that short block lengths.

B. DRP Interleavers with d_{min}° Constraint

Likewise, we designed a set of code matched DRP interleavers for the component code configurations given in Table I yielding $(128, 64)$ PTCs. We first generated a set of DRP interleavers that satisfy a minimum spread of $S_{\mathsf{min}} = 5$ allowing $W \leq 8$ and $R \leq 8$ and then used Algorithm 1 to select those interleavers that satisfy the d_{min}° constraint. The parameters of the best interleavers resulting from our search are summarized

TABLE II
PARAMETERS FOR DRP INTERLEAVERS WITH d°_{min} CONSTRAINT.

m	S_{min}	d_{min}	$\tilde{\mathbf{r}}$	$\tilde{\mathbf{w}}$	$\tilde{p}$	$\tilde{s}$
3	5	12	$(1, 0, 6, 7, 2, 5, 4, 3)$	$(7, 2, 5, 0, 3, 1, 4, 6)$	19	61
4	6	14	$(2, 3, 0, 1)$	$(3, 2, 1, 0)$	7	31
5	6	16	$(0, 3, 2, 1)$	$(0, 1, 2, 3)$	9	10

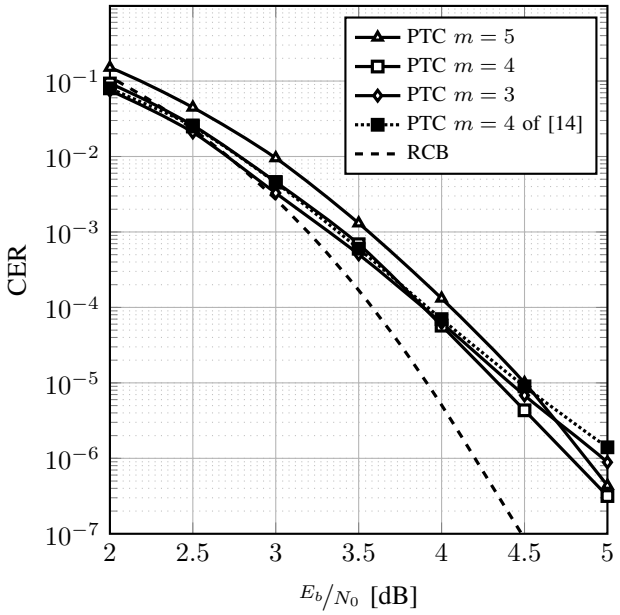

Fig. 4. CER versus E_b/N_0 of $(128, 64)$ PTCs with different memories m.

in Table II. In Figure 4 we compare performances of PTCs with the proposed code matched DRP interleavers. Again the RCB for rate-$1/2$ block codes with dimension 64 is depicted as a reference. Note that both memory $m = 3$ and memory $m = 4$ codes perform virtually the same in the waterfall region, whereas the latter one outperforms in the error floor region. Interestingly, our $m = 3$ code outperforms the $m = 4$ code from [14] that is added here as a reference. In [14], zero-tail terminated convolutional codes as component codes are used and a rate-$1/2$ PTC is obtained by puncturing as a last step of the code design. For memory $m = 5$ a loss in performance of about ≈ 0.15 dB in the waterfall region is visible which is in alignment with iterative decoding threshold predictions. It is expected that due to the higher d_{min} the memory $m = 5$ code will have the best performance for CERs lower than 10^{-7}.

Figure 5 compares the performance of the proposed memory 4, $(128, 64)$ PTC with codes of same length and rate: a binary $m = 4$ PTC with $d_{\mathsf{min}} = 10$ [14], a non-binary LDPC code over the finite field of order 256 with $d_{\mathsf{min}} = 14$ [26] and the CCSDS telecommand which is a binary LDPC code with $d_{\mathsf{min}} = 14$ [2]. The proposed PTC gains 0.4 dB at a CER of about $\approx 3 \cdot 10^{-7}$ w.r.t. to the PTC from [14], while a gain of about 0.85 dB w.r.t. the binary LDPC code is shown. Compared to the non-binary code only a loss of about 0.5 dB is visible making the proposed code an interesting candidate for low-complexity applications. Note that we estimated the

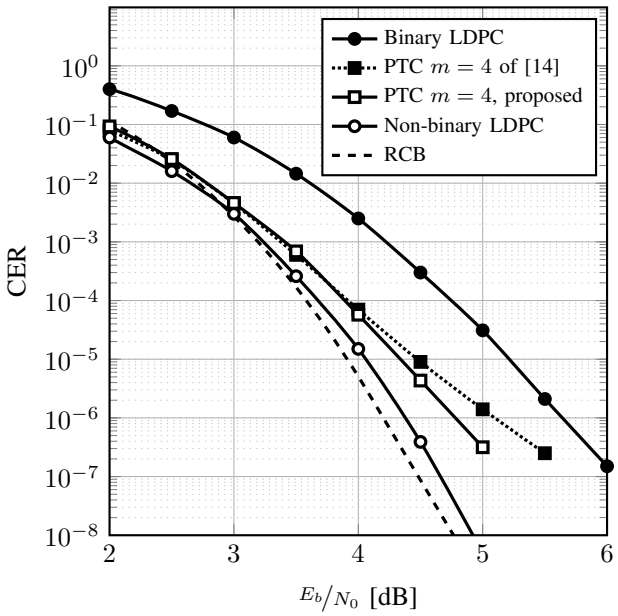

Fig. 5. Comparison of the proposed $(128, 64)$ PTC with $m = 4$, with binary and non-binary $(128, 64)$ LDPC and the $(128, 64)$ PTC with $m = 4$ from [14].

algorithmic decoding complexity of the non-binary LDPC code to be approximately 60 times higher than that of its binary counterpart. The results illustrate that a structured turbo code design as proposed in this work allows low error floors paired with excellent waterfall performance even for short blocks.

V. CONCLUSIONS

This paper provides a methodical way to construct PTCs with high minimum distances for short block lengths and arbitrary rates. The obtained codes outperform state-of-the-art binary turbo and LDPC codes at comparable decoding complexity. We also illustrate that for short blocks PTCs using larger memory RSC codes as component codes bring only a minor performance loss in the waterfall region while the error floor can be lowered considerably.

ACKNOWLEDGMENT

The authors wish to thank Gianluigi Liva for the fruitful discussions and helpful comments around the topic of the paper.

REFERENCES

[1] G. Durisi, T. Koch, and P. Popovski, "Towards massive, ultra-reliable, and low-latency wireless: The art of sending short packets," *CoRR*, vol. abs/1504.06526, 2015. [Online]. Available: http://arxiv.org/abs/1504.06526

[2] *Next Generation Uplink*, Green Book, Issue 1, Consultative Committee for Space Data Systems (CCSDS) Report Concerning Space Data System Standards 230.2-G-1, Jul. 2014.

[3] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.

[4] R. Gallager, *Low-density parity-check codes*. Cambridge, MA, USA: MIT Press, 1963.

[5] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[6] T. Tian, C. Jones, J. D. Villasenor, and R. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.

[7] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *Proc. IEEE Inf. Theory Workshop*, Paris, France, Mar. 2003, pp. 70–73.

[8] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani, "Short turbo codes over high order fields," *IEEE Trans. Commun.*, vol. 61, no. 6, pp. 2201–2211, Jun. 2013.

[9] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun.*, Geneva, Switzerland, May 1993.

[10] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[11] D. Divsalar, S. Dolinar, and C. Jones, "Short protograph-based LDPC codes," in *Proc. Int. Conf. Mil. Commun.*, Orlando, FL, USA, Oct. 2007, pp. 1–6.

[12] R. Gallager, "The random coding bound is tight for the average code (corresp.)," *IEEE Transactions on Information Theory*, vol. 19, no. 2, pp. 244–246, Mar. 1973.

[13] H. R. Sadjadpour, M. Salehi, N. J. A. Sloane, and G. Nebe, "Interleaver design for short block length turbo codes," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, New Orleans, LA, USA, Jun. 2000, pp. 628–632.

[14] M. Baldi, M. Bianchi, F. Chiaraluce, R. Garello, I. Sanchez, and S. Cioni, "Advanced channel coding for space mission telecommand links," in *Proc. IEEE Vehicular Technology Conference Fall*, Las Vegas, NV, USA, Sep. 2013, pp. 1–5.

[15] A. Valembois and M. Fossorier, "Box and match techniques applied to soft-decision decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 796–810, May 2004.

[16] Y. Wu and C. N. Hadjicostis, "Soft-decision decoding using ordered recodings on the most reliable basis," *IEEE Trans. Inf. Theory*, vol. 53, no. 2, pp. 829–836, Feb. 2007.

[17] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[18] M. Mooser, "Some periodic convolutional codes better than any fixed code," *IEEE Trans. Inf. Theory*, vol. 29, no. 5, pp. 750–751, Sep. 1983.

[19] C. Weiss, C. Bettstetter, and S. Riedel, "Code construction and decoding of parallel concatenated tail-biting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 1, pp. 366–386, Jan. 2001.

[20] *TM Synchronization and Channel Coding*, Blue Book, Issue 2, Consultative Committee for Space Data Systems (CCSDS) Recommendation for Space Data System Standard 131.0.B.2, Aug. 2011.

[21] A. R. Calderbank, G. D. Forney, and A. Vardy, "Minimal tail-biting trellises: the golay code and more," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1435–1455, Jul. 1999.

[22] S. Benedetto and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 409–428, Mar. 1996.

[23] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[24] D. Divsalar and F. Pollara, "Multiple turbo codes for deep-space communications," NASA JPL, Pasadena, CA, USA, TDA Progress Report 42-121, May 1995.

[25] S. Crozier and P. Guinand, "High-performance low-memory interleaver banks for turbo-codes," in *Proc. IEEE Vehicular Technology Conference Fall*, Atlantic City, NJ, USA, Oct. 2001, pp. 2394–2398.

[26] G. Liva, E. Paolini, T. de Cola, and M. Chiani, "Codes on high-order fields for the ccsds next generation uplink," in *Advanced Satellite Multimedia Systems Conference (ASMS) and 12th Signal Processing for Space Communications Workshop (SPSC), 2012 6th*, Baiona, Spain, Sep. 2012, pp. 44–48.