Rate-Compatible, Bandwidth-Efficient LDPC Codes for Aeronautical Telemetry

Andrew D. Cummins *, David G. M. Mitchell, Erik Perrins

Posted Date: 29 October 2024

doi: 10.20944/preprints202410.2273.v1

Keywords: FEC; LDPC codes; telemetry; avionics; IRIG; spectral efficiency; puncturing; shortening; iterative decoding

 doi:10.20944/preprints202410.2273.v1

Article

Rate-Compatible, Bandwidth-Efficient LDPC Codes for Aeronautical Telemetry

Andrew D. Cummins [1,*,†], David G. M. Mitchell [1,†] and Erik Perrins [2,†]

[1] Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM
[2] Department of Electrical Engineering & Computer Science, University of Kansas, Lawrence, KS
* Correspondence: andrewdc@nmsu.edu
† These authors contributed equally to this work.

Abstract: Low-density parity-check (LDPC) codes form part of the IRIG-106 standard and have been successfully deployed for the Telemetry Group version of shaped-offset quadrature phase shift keying (SOQPSK-TG) modulation. Recently, LDPC code solutions have been proposed and optimized for continuous phase modulations (CPMs), including pulse code modulation/frequency modulation (PCM/FM) and the multi-h CPM developed by the Advanced Range TeleMetry program (ARTM CPM). These codes were shown to perform around one dB from the respective channel capacities of these modulations. In this paper, we consider the effect of random puncturing and shortening of these LDPC codes to further improve spectrum efficiency. We perform asymptotic analyses of the ARTM0 code ensembles and present numerical simulation results that affirm the robust decoding performance promised by LDPC codes designed for ARTM CPM.

Keywords: FEC; LDPC codes; telemetry; avionics; IRIG; spectral efficiency; puncturing; shortening; iterative decoding

1. Introduction and Motivation

Low density parity check (LDPC) codes [1] are a family of forward error correction (FEC) block codes that achieve capacity-approaching performance under low-complexity iterative decoding schemes [2,3]. Efficient LDPC encoder designs have given rise to implementations [4,5] that meet the stringent requirements of low-power transmitters and LDPC decoding algorithms are well suited to highly-parallelized decoder structures that facilitate high-throughput FEC [6,7]. LDPC codes have been successfully deployed for aeronautical telemetry systems that employ continuous phase modulations (CPM), specifically SOQPSK-TG [8]. Recent work outlining techniques for the construction of LDPC codes for the remaining two CPM modulation schemes, PCM/FM and ARTM CPM, have demonstrated performance within one dB of channel capacity in numerical simulations [9,10]. While keeping encoding complexity low, these codes represent the current best FEC performance published for ARTM CPM and further optimizing these codes for deployment in aeronautical telemetry systems under bandwidth constraints is an open problem.

One remaining challenge is to develop spectrally efficient LDPC coding schemes for CPM that can adapt to the changing conditions of time-varying channels. Coding schemes that can adaptively respond to changing channel conditions without need for alterations to either their encoder or decoder hardware configurations are known as *rate-compatible* codes [11]. An efficient means for accomplishing this is to omit sending a portion of the encoded information stream, a process known as *puncturing* [11,12]. This technique relies on the strength of the code and effect of the puncturing pattern on the decoding algorithm to provide sufficient, though diminished, error correction performance without requiring changes to the decoder [13–15]. No *a priori* information on punctured bits is shared with the decoder prior to transmission, and thus the code rate is increased at the expense of a handicapped decoder. A complementary method, *shortening* [16], follows much the same logic, but certain code symbols are fixed at the encoder and omitted from transmission where the decoder knows the positions of the shortened bits. This modification can be achieved at the decoder by providing known, perfect *a priori* information on shortened bits, improving the overall iterative decoding performance at the expense of a decrease in both the code rate and the dimension. Shortening has been shown to improve the performance of LDPC codes [17,18] and an analysis of random shortening was performed in [19].

 doi:10.20944/preprints202410.2273.v1

1.1. Methods and Contributions

In this paper we use random puncturing and shortening to explore the trade-off between coding rate and error control performance for some of the LDPC coding schemes proposed in [9] and [10], enabling the development of spectrally-efficient, rate-compatible coding strategies which require minimal alterations to proven hardware implementations. We follow the notation established in [9] and [10] by using ARTM0, ARTM1, and ARTM2 to refer to PCM/FM, SOQPSK-TG, and ARTM CPM, respectively. We begin by performing asymptotic analyses of the ARTM0 code ensembles. First, we determine the iterative belief propagation (BP) decoding thresholds of randomly punctured ARTM0 LDPC code ensembles on the binary erasure channel (BEC). We then perform an asymptotic minimum distance analysis and show that both the mother code ensembles and the punctured code ensembles are asymptotically good. Moreover, we find that, even though the minimum distance growth rates decrease as the puncturing fraction increases, the gap to the Gilbert-Varshamov bound decreases with puncturing. Taken together, the asymptotic analyses indicate that the ARTM0 LDPC code ensembles are good candidates for random puncturing.

We then present numerical simulation results for various puncturing ratios to create effective coding rates in-between those previously demonstrated. To isolate the effect of modulation on the LDPC encoder-decoder pair, we present results both with and without CPM, highlighting the significant coding gain achieved through optimized design of the LDPC codes with respect to specific modulation schemes. We observe that modulation feedback is responsible for greater than 2 dB of coding gain over the same code without modulation, illustrating the impact of the optimization for that regime. Our results demonstrate that significant gains in spectral efficiency can be obtained for reasonable degradation in FEC performance. For example, with 5% of the symbols omitted from transmission by puncturing, we are able to increase a code with rate $R = 2/3 = 0.667$ to $R = 0.701$ with less than a 0.2 dB loss at a bit error rate of 10^{-6} and with no alterations to the component code or decoder required.

We also provide numerical simulation results for shortened codes showing marked improvements in waterfall performance for codes of sufficient rate and whose blocksizes are of sufficient length. For example, for the ARTM0 mother code with $R = 4/5$ and information length $k = 4096$, a coding gain of more than 0.2dB can be obtained with random shortening under the CPM modulated scheme. Furthermore, interchanging hardware to allow for shortening and puncturing is not necessary, as both techniques utilize overlapping function blocks with few alterations and may be combined for further flexibility when designing rate-compatible codes.

1.2. Paper Structure

The paper is structured as follows. Section 2 provides the necessary background material, including protograph-based LDPC code construction as well as the code modification techniques we will use, namely puncturing and shortening. In Section 3, we perform asymptotic analyses of the randomly punctured protograph-based ARTM0 ensembles to determine their suitability for puncturing from the perspective of BP and minimum distance. Sections 4 and 5 provide the system models and numerical simulation results obtained for random puncturing and shortening, respectively. Section 6 then provides a brief description of the hardware considerations before some concluding remarks are provided in Section 7.

2. Background

2.1. Protograph-Based LDPC Codes

An (n, k) LDPC code, with length n and dimension k, is described as the null space of a parity-check matrix $\mathbf{H}$ that has rank $n - k$. We say that an LDPC code is (J, K)-*regular* if it has precisely J ones in every column and K ones in every row, otherwise it is called *irregular*. A *protograph* [20], with design rate $R = 1 - n_c/n_v$, is a small bipartite graph that connects a set of n_v variable nodes to a set of n_c check nodes by a set of edges. The protograph can be represented by a *base* biadjacency matrix $\mathbf{B}$, where $B_{x,y}$ is taken to be the number of edges connecting variable node v_y to check node c_x. The parity-check matrix $\mathbf{H}$ of a protograph-based LDPC code can be created by replacing each non-zero entry in $\mathbf{B}$ by a sum of $B_{x,y}$ non-overlapping permutation matrices of size $M \times M$ and each zero entry by the $M \times M$ all-zero matrix.

 doi:10.20944/preprints202410.2273.v1

Example: The (irregular) ARTM0 protographs [9] are given as

$$\mathbf{B}_{0,1/2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 2 \\ 3 & 3 & 2 & 2 & 2 & 2 & 1 & 1 \end{bmatrix}, \tag{1}$$

$$\mathbf{B}_{0a,2/3} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 2 \\ 3 & 3 & 3 & 3 & 2 & 2 & 2 & 3 & 3 & 2 & 0 & 1 \end{bmatrix}, \tag{2}$$

$$\mathbf{B}_{0b,2/3} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 0 & 1 & 0 \end{bmatrix}, \text{ and} \tag{3}$$

$$\mathbf{B}_{0,4/5} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

following the naming convention of [9] that $\mathbf{B}_{0,1/2}$ refers to the ARTM0 LDPC code ensemble of design rate $R = 1/2$, and so on.

2.2. Puncturing Linear Codes

A (n, k) linear code is *punctured* by removing a set of p columns from its generator matrix, which has the effect of reducing the codeword length from n to $n - p$. After puncturing a linear code with *puncturing fraction* $\alpha_p = p/n$, the resulting transmission rate is

$$R(\alpha_p) = \frac{R}{1 - \alpha_p}, \quad \alpha_p \in [0, 1), \tag{5}$$

where $R(0) = R$ is the rate of the mother (unpunctured) code. The dimension of the code is preserved, and therefore the target rate $R(\alpha_p)$ is achieved, provided no two distinct codewords differ within the p punctured symbols only. This can be achieved, for example, by restricting punctured symbols to the $n - k$ parity-check symbols of a code in systematic form. A code can be punctured randomly or according to a particular pattern. It is assumed that the receiver knows the positions of the punctured symbols, and the decoder estimates both the punctured and transmitted symbols during decoding.

2.3. Shortening Linear Codes

A (n, k) linear code is *shortened* by keeping only the codewords from the mother code that have zeros at the postions of the s shortened bits. This is equivalent to removing the corresponding set of s columns from its parity-check matrix. In this paper, we restrict shortening to the k information symbols. Shortening will reduce both the dimension and length of the code. After shortening a linear code with *shortening fraction* $\alpha_s = s/n$, the resulting transmission rate is

$$R_s(\alpha_s) = \frac{R - \alpha_s}{1 - \alpha_s} = \frac{k - s}{n - s}, \quad \alpha_s \in [0, 1), \tag{6}$$

where $R_s(0) = R$ is the rate of the mother (unshortened) code.

The shortening positions must satisfy the constraint that the shortened parity-check matrix has rank $n - k$. This can be achieved, for example, by restricting shortened symbols to the k information symbols of a code in systematic form. A code can be shortened randomly or according to a particular

 doi:10.20944/preprints202410.2273.v1

pattern. In order to be rate-compatible, we assume that the receiver knows the positions of the shortened symbols and maintains perfect information about those bits in order to estimate the transmitted symbols during decoding.

3. Asymptotic Random Puncturing Analyses of ARTM0 Protographs

In this section, we examine the suitability of the ARTM0 protographs for random puncturing. We remark that specific, finite-length graph liftings were optimized in [9] in order to achieve excellent performance with CPM and a two-stage iterative decoder; however, in this section, we assume asymptotic random liftings for the purpose of analysis of the code ensembles.

3.1. Thresholds of Randomly Punctured ARTM0 Code Ensembles on the BEC

Consider puncturing a length n codeword $\mathbf{v}$ for transmission over a BEC with erasure probability ϵ. We assume that a fixed fraction $\alpha = p/n$ of the code symbols are punctured, such that the transmitted codeword $\mathbf{v}^{\text{punc}}$ has length $n^{\text{punc}} = (1-\alpha)\cdot n$. After transmission, the received vector $\mathbf{r}$ will contain, on average, $\epsilon \cdot n^{\text{punc}}$ erased symbols and $(1-\epsilon)\cdot n^{\text{punc}}$ correct symbols. The receiver knows the positions of the punctured and erased symbols and proceeds to decode the overall code of length n. From [14], we recall that the BP threshold $\epsilon_{\text{BP}}(\alpha_p)$ of a randomly punctured LDPC code ensemble on the BEC with puncturing fraction α_p is given by

$$\epsilon_{\text{BP}}(\alpha_p) = 1 - \frac{1-\epsilon_{\text{BP}}(0)}{R}\cdot R(\alpha_p), \tag{7}$$

where $\epsilon_{\text{BP}}(0) = \epsilon_{\text{BP}}$ and R are the BP threshold and design rate of the (unpunctured) mother code ensemble, respectively, and $R(\alpha_p)$ is the target rate after puncturing.

Figure 1 shows numerically calculated BP thresholds of the randomly punctured ARTM0 LDPC code ensembles for a variety of puncturing fractions α. The mother LDPC code ensemble thresholds $\epsilon_{\text{BP}}(0)$ are shown by the markers (circle, cross, square, and diamond), with the corresponding punctured thresholds $\epsilon_{\text{BP}}(\alpha_p)$, for various values of α_p and corresponding rate $R(\alpha_p)$, shown by the lines from a given marker. We observe relatively robust performance under puncturing, particularly for the $R = 2/3(b)$ and $R = 4/5$ ensembles, for which $\epsilon_{\text{BP}}(0)$ begins closer to capacity. For each ensemble, the threshold values decrease with increasing α_p as $R(\alpha_p)$ increases. Since none of the curves cross, this indicates that random puncturing beyond the rates of the next mother code ensemble are unlikely to provide any advantage in terms of performance. We also note that the rate for which the lines reach a 0 value of threshold is indicative of the maximum rate achievable for random puncturing of the given ARTM0 LDPC code ensemble.

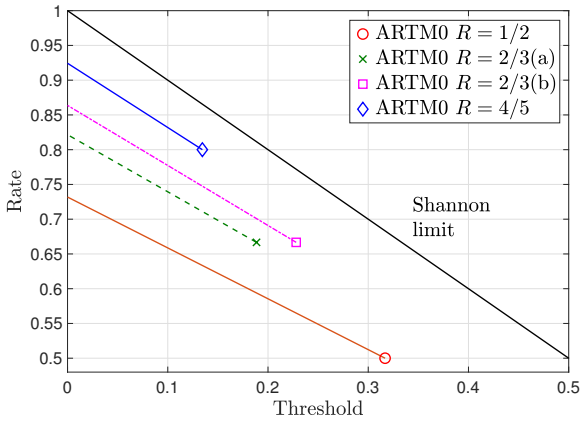

Figure 1. BEC BP thresholds of randomly punctured ARTM0 LDPC code ensembles for a variety of puncturing fractions α_p.

3.2. Minimum Distance Growth Rates of Randomly Punctured ARTM0 Code Ensembles

For $J > 2$, (J, K)-regular LDPC codes are known to be *asymptotically good* [21], in the sense that the minimum distance typical of most members of the ensemble is at least as large as $\delta_{\min} \cdot n$, where

 doi:10.20944/preprints202410.2273.v1

$\delta_{\min} > 0$ is called the *minimum distance growth rate* of the ensemble. The *asymptotic spectral shape* of a code ensemble, defined as

$$r(\delta) = \limsup_{n\to\infty} \frac{1}{n} \ln(A_{\lfloor \delta n \rfloor}), \tag{8}$$

can be used to test if an ensemble is asymptotically good, where $\delta = d/n$ is the normalized Hamming distance, $n \in \mathbb{N}$ is the block length, and A_d is the ensemble weight enumerator. A technique to calculate $r(\delta)$ for protograph-based LDPC code ensembles was presented in [22]. As a result of the variable node degree optimization of the ARTM0 protographs, we find that the ensembles are asymptotically good, with computed minimum distance growth rates shown in Table 1.

Table 1. Minimum distance growth rates for the ARTM0 LDPC code ensembles.

ARTM0	$R = 1/2$	$R = 2/3(a)$	$R = 2/3(b)$	$R = 4/5$
$\delta_{\min}$	0.00344	0.00066	0.00161	0.00032

Given the asymptotic spectral shape $r(\delta)$ of an asymptotically good code ensemble, the expected asymptotic spectral shape of the randomly punctured code ensemble can be obtained as [23]

$$r^{\text{punc}}(\delta) = \frac{1}{1-\alpha_p} \left(\max_{0\le\lambda\le 1} \left\{ \lambda \cdot h\left(\frac{(1-\alpha_p)\delta}{\lambda} \right) + (1-\lambda)\cdot h\left(\frac{\alpha_p + (1-\alpha_p)\delta - \lambda}{1-\lambda} \right) + r(\lambda) \right\} - h(\alpha_p) \right), \tag{9}$$

where $\alpha_p = p/n$ is the fraction of punctured bits, $0 \le \alpha_p < \delta_{\min}$, and $h(\delta) = -(1-\delta)\ln(1-\delta) - \delta\ln(\delta)$ is the binary entropy function. The average weight enumerators used in the formulation of (9) are obtained over all possible p-bit puncturing patterns; therefore, we require $\alpha_p < \delta_{\min}$ to guarantee no rate loss.[1]

In Figure 2, we present numerical results for the ARTM0 $R = 1/2$ and $R = 2/3$(b) ensembles obtained using (9) for a variety of puncturing fractions α_p. We note that, in most cases, $\alpha_p > \delta_{\min}$ and, as a consequence, there can be some rate loss, where the true rate is bounded above by $R(\alpha_p)$. We note, however, that in practice the bits to be punctured would be selected to avoid rate loss by preserving the dimension of the code (semi-randomly or otherwise), and therefore the numerical results obtained in this section are useful indicators of actual punctured code performance.

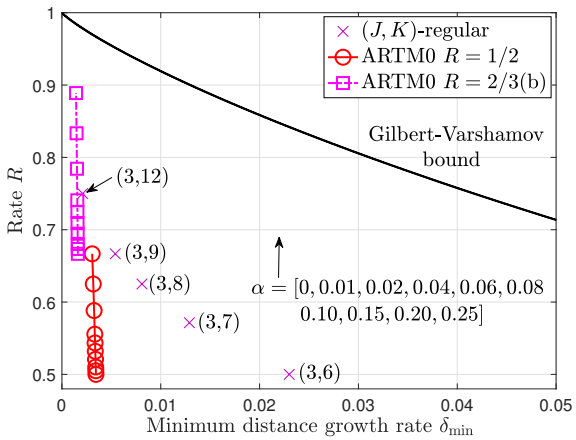

Figure 2. Minimum distance growth rates for the punctured ARTM0 LDPC code ensembles.

[1] For $\alpha_p \ge \delta_{\min}$ the rate of the ensemble can be written as $(R - \Delta_R)/(1-\alpha_p)$, where $\Delta_R \ge 0$. Bounds on Δ_R and conditions such that $\Delta_R = 0$ were given for (J, K)-regular LDPC code ensembles in [24].

 doi:10.20944/preprints202410.2273.v1

We observe that, like the mother code ensembles, the punctured ARTM0 LDPC code ensembles are also asymptotically good. Although the minimum distance growth rates must decrease with α_p, the ARTM0 growth rates show robust behavior as the code rate increases, where we observe that the gap to the Gilbert-Varshamov bound decreases with α_p. In particular, the punctured $R = 2/3$ ensemble with $\alpha_p = 0.1$ has a similar growth rate as the $(3, 12)$-regular LDPC code ensemble. Taken along with the iterative threshold analysis results from Section 3.1, it appears that the ARTM0 LDPC code ensembles are indeed good candidates for random puncturing.

4. Random Puncturing of ARTM0 LDPC Codes

4.1. System Model

The transmitter model utilized in our system in shown in Figure 3, with function blocks representing the LDPC encoder, interleaver (Π), puncturing (ϕ), CPM modulator, and a module that allows for the insertion of a known symbol sequence referred to as an attached sync marker (ASM) which helps to identify the beginning of each codeword $\mathbf{c}$ at the receiver. LDPC encoding is performed by multiplying source information sequence $\mathbf{x}$ of length k with the (systematic, in the case of the ARTM0 codes) code generator matrix $\mathbf{G}$ of size $k \times n$ to form codeword $\mathbf{y} = \mathbf{xG}$. The resulting codeword $\mathbf{y}$ is interleaved to form the sequence $\mathbf{i}$, on which puncturing is performed. The resulting punctured codeword, $\mathbf{i}_\phi$, is concatenated with the ASM to form a frame $\mathbf{u}$ which is modulated by the CPM and sent over the communication channel as the sequence of q-ary frames $s(t; \mathbf{u})$ where q is the modulation order. Puncturing (ϕ) must be performed at the encoder prior to the concatenation of the interleaved codeword and ASM in order to avoid desynchronization during decoding. For random puncturing, if the puncturing pattern and interleaver are sufficiently random, puncturing may be performed before or after interleaving as the distribution of punctured symbols will remain uniformly distributed in either case, while nonrandom puncturing would be constrained to placement after interleaving.

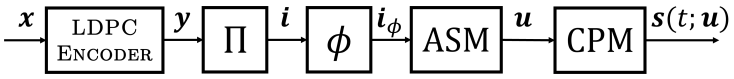

Figure 3. Model transmitter for punctured LDPC code with CPM modulation.

Efficient LDPC encoding algorithms are suitable for many low-power, memory-constrained applications [7]. This is true for the ARTM LDPC codes we have used, which have a quasi-cyclic (QC) structure in which the code's parity check matrix is an array of sparse repeating units called *circulants* [4]. The most commonly implemented decoder for LDPC codes is a special case of a more general BP algorithm used for inference on graphical models, known as the sum-product algorithm (SPA) [25]. In this paper, we have used SPA decoding because it provides good performance with soft-input soft-output (SISO) functionality. Efficient hardware implementations of such decoders are numerous and widely adopted [6], including reduced complexity variants such as the min-sum algorithm (MSA) [7].

The receiver model is shown in Figure 4. In this diagram, the values passed between blocks are all real-valued vectors, i.e., log-likelihood ratios (LLRs) [16], that corresponding to the discrete-valued sequence indicated at the input/output of the function blocks. The received sequence $r(t)$ is input to the CPM demodulator corresponding to a CPM word $\mathbf{c}$ along with with no initial *a priori* information (zero-valued LLRs) on $\mathbf{i}_\phi$, the binary interleaved and punctured codeword from the LDPC code. After demodulation of a codeword, it is decoded iteratively in a global loop by a pair of CPM and LDPC decoders, which exchange *a priori* information on $\mathbf{i}_\phi$ through an interleaver Π and puncturing block ϕ, identical to those used for encoding, with corresponding blocks for the inverse deinterleaver Π^{-1} and the depuncturer ϕ^{-1} (explained below). For the first global iteration, the results of the CPM demodulator are permuted and used as soft-valued inputs (channel LLRs) for the local LDPC decoder loop by placing the switch in position A. For subsequent global iterations, the signal flow is altered by placing the switch position in B, with the soft output LLRs of the CPM SISO decoder's update now serving as *a priori* inputs to the LDPC decoder.

 doi:10.20944/preprints202410.2273.v1

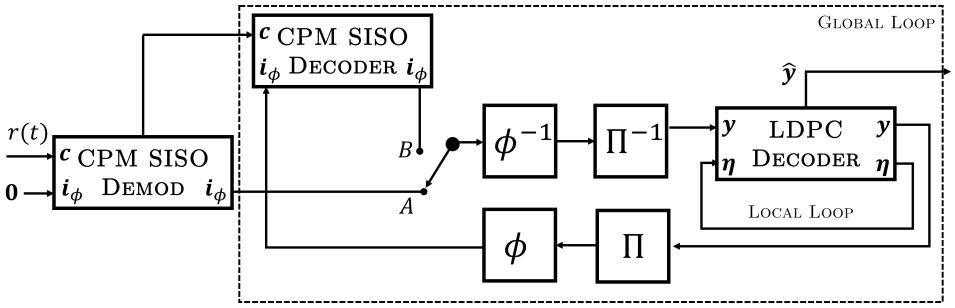

Figure 4. Model receiver for punctured LDPC code with CPM modulation and iterative decoding.

LDPC SPA decoding is performed using the code's parity-check matrix $\mathbf{H}$, a sparse matrix which satisfies the condition $\mathbf{GH}^T = 0$. Sparsity keeps the memory and computational requirements tractable for coding schemes with long block lengths that are needed to obtain capacity-approaching performance. The rows of $\mathbf{H}$ represent the parity check conditions of the code while each column represents a codeword symbol. If a one is present in the k-th column of row j, code symbol k is found in parity check equation A_j. The LDPC decoder performs iterative SPA decoding in the local loop, before passing the output LLRs on $\mathbf{y}$ as *a priori* information on $\mathbf{i}_\phi$ to the CPM SISO decoder to start the next global iteration, and so on. We note that in our implementation, the LDPC decoder retains soft values on the punctured symbols within the local loop, this is indicated in Figure 4 by the signal path $\boldsymbol{\eta}$. The process continues until some stopping criteria is reached (such a ceiling on the allowed number of iterations) and the resulting estimated codeword $\hat{\mathbf{y}}$ is passed out of the decoder. Upon completion, hard decisions on $\hat{\mathbf{y}}$ are made to complete decoding, where $\hat{\mathbf{x}}$ is directly accessible since the encoder is systematic.

Puncturing is performed in the global loop between the CPM SISO decoder and the LDPC decoder. The module ϕ removes codeword soft information symbols in accordance with the known puncturing pattern set by the transmitter, resulting in a truncated sequence $\mathbf{i}_\phi$, as shown in Figure 5. The module ϕ^{-1} intercalates zeros into these same puncturing positions to expand the sequence to its original length before interleaving, in a process referred to as *depuncturing*. Since the messages passed in the SPA correspond to LLRs, the zero valued LLRs that are placed into the puncturing positions correspond to a bit being equally likely a one or a zero. Finally, we note that at the receiver (Figure 4) the puncturing, depuncturing, interleaving, and deinterleaving all operate on real-vauled vectors.

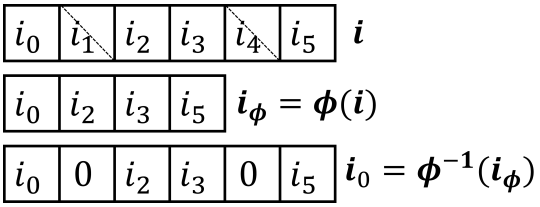

Figure 5. Puncturing by truncation and depuncturing by zero-padding within the global decoding loop of the receiver.

4.2. Random Puncturing Overhead

The code rate, $R = \frac{k}{n}$ is the ratio of the number of symbols in the source information sequence, k, to the number of symbols in the transmitted information sequence, n, including parity information. As described in Section 2.2, for a block code of length n, wherein p symbols are punctured, we express the amount of puncturing *overhead* as the fraction $\alpha_p = p/n$, or as the percentage, $\Delta = \alpha_p \cdot 100\%$. Rate adjustments are made by altering the degree of puncturing overhead. Because puncturing operates on the factor in the denominator of the rate expression (5), the resulting reduction of this quantity increases the code rate. Increasing the code rate results in fewer parity symbols transmitted per unit time, thus increasing throughput or reducing spectrum utilization for any desired target throughput.

For the block code we are implementing, the code rate after puncturing, $R(\alpha_p)$, is determined solely by the mother (unpunctured) code rate and puncturing overhead, as given in (5). Therefore, the amount of overhead required for any desired code rate is $\Delta = \left(1 - R/R(\alpha_p)\right) \cdot 100\%$. The codes outlined in [9] are *systematic* LDPC codes, wherein the source information sequence appears unaltered

 doi:10.20944/preprints202410.2273.v1

in the encoded sequence, but in general this is not a requirement because no distinction is made between information and parity symbols in our scheme and puncturing is performed after interleaving. These codes exhibit sufficient regularity such that information symbols and parity symbols are equally protected and punctured symbols are therefore not limited to parity bits in our study.

4.3. Numerical Results

In this section we present the results of our numerical simulations which were performed on the additive white Gaussian noise (AWGN) channel with and without CPM modulation. For the CPM (full system) simulations, we followed [9] and set a maximum of 512 global iterations and a maximum of 1 LDPC decoder iteration with a syndrome-based stopping rule. For the LDPC code only simulations, we use binary phase-shift keying (BPSK) modulation and set a maximum of 100 LDPC decoder iterations and a syndrome-based stopping rule. To obtain an estimate of the average random puncturing performance, we use a different random puncturing pattern per transmitted codeword in our simulations.

4.3.1. Isolated Code, No CPM

In order to isolate the effect puncturing has on decoding performance of the LDPC code alone, we have removed CPM and associated support modules, resulting in the greatly simplified transceiver model shown in Figure 6. With this model, puncturing is performed immediately before encoded information is sent over the communications channel, and depunctured after being received only once. Due to the elimination of the global iterative loop used to exchange information between the CPM and LDPC decoders (c.f., Figure 4), the depunctured information (soft-valued channel LLRs) directly enters the SPA LDPC decoder, where it is iteratively decoded to produce codeword estimate $\hat{\mathbf{y}}$. Simulated error correction performance on the AWGN channel for the $R = 2/3$ ARTM0 code with blocklength $n = 1024$ and puncturing overhead values of $\Delta = (0\%, 1\%, 5\%, 10\%, 16.7\%)$ is shown in Figure 7.[2] The value $\Delta = 16.7\%$ is the percentage of puncturing necessary create a $R(0.167) = 0.8$ code from a $R = 2/3$ mother code and thus represents the intersection between these two approaches.[3]

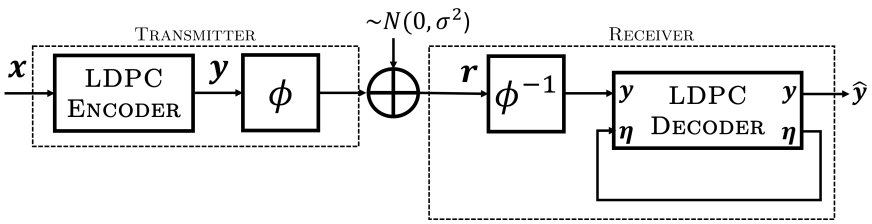

Figure 6. Simplified transceiver model without CPM.

Without CPM, SPA decoding is strongly affected by random puncturing, with 0.5 dB of coding gain lost at a bit error rate (BER) of 10^{-4} when moving from $\Delta = 0$ to $\Delta = 5$, which represents an effective rate increase from $R = 2/3$ to $R(0.05) = 0.702$. Although small values of puncturing are well tolerated, such as shown for $\Delta = 1\%$, these do not provide a significant increase in code rate (*e.g.*,, $\Delta = 1\%$ increases $R = 2/3$ to $R(0.01) = 0.673$). It is also observed that puncturing to the next available code rate of $R = 4/5$ by setting $\Delta = 16.7\%$ results in poor performance, since the decoder cannot compensate for this large Δ. Although it may happen that a code optimized for use with CPM is suboptimal for SPA decoding alone, it is worth nothing that a coding gain of over 7 dB was reported [10] with the modulation scheme in place, outperforming all existing codes which were not optimized for the same modulation schemes.

---

[2] The $k = 1024$, $R = 2/3$, ARTM0 code is constructed from the lifted base matrix $\mathbf{B}_{0b,2/3}$ [9].
[3] In this paper, rates are rounded to three decimal places.

 doi:10.20944/preprints202410.2273.v1

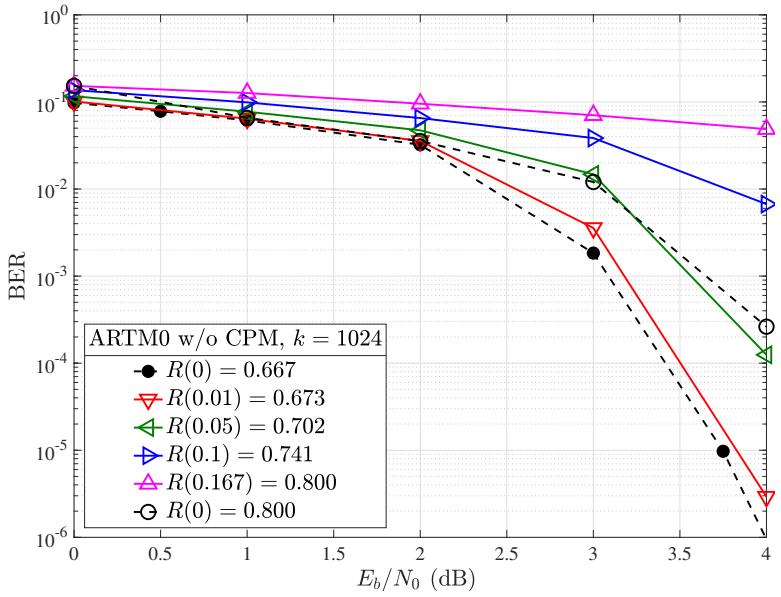

Figure 7. BER performance without CPM for the $R = 2/3$, $k = 1024$ ARTM LDPC code with (dashed) and without (solid) random puncturing.

4.3.2. CPM Included

With CPM included, as shown in the systems presented in Figures 3 and 4, a significant coding gain can be attributed to the global decoder-modulation loop alone. As shown in Figure 8, a coding gain in excess of 2 dB is observed between the systems with and without modulation even without considering puncturing ($\Delta = 0\%$). This illustrates the chosen LDPC code's optimization for ARTM modulation and the ability of the SPA decoder to take advantage of information provided by the CPM decoder, working in concert by exchanging successively more accurate *a priori* information in a doubly-iterative fashion.

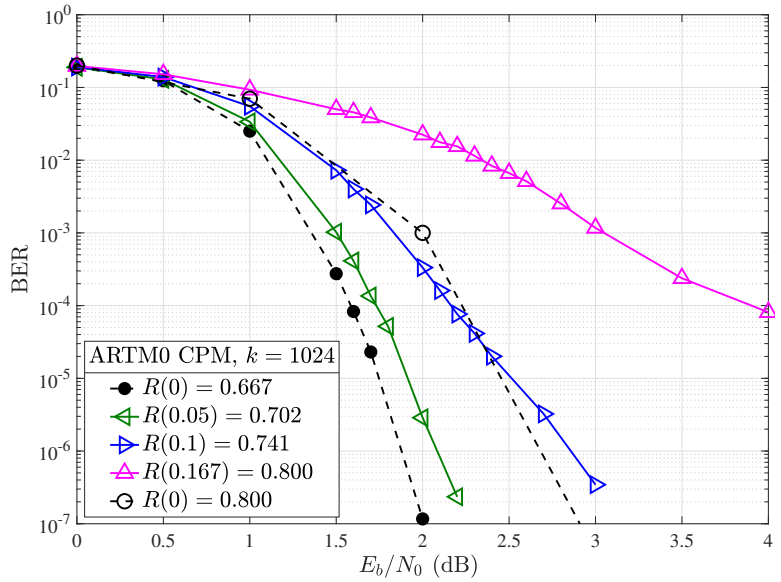

Figure 8. BER performance with CPM for the $R = 2/3$, $k = 1024$ ARTM LDPC code with (dashed) and without (solid) random puncturing.

For the full system, puncturing has a much more gradual influence on the knee or *waterfall* region of the resulting BER curve, with an *error floor* only developing for overheads in excess of 10%. For example, with $\Delta = 5\%$, corresponding to an increase of the code rate from $R = 2/3 = 0.667$ to $R(0.05) = 0.702$, we observed a loss of approximately 0.5 dB at a BER of 10^{-4} in the LDPC only case

 doi:10.20944/preprints202410.2273.v1

(Figure 7), whereas in the CPM case we see that this is reduced to approximately 0.1 dB. At a bit error rate of 10^{-6}, the loss remains less than 0.2 dB. At $\Delta = 10\%$, with $R(0.1) = 0.741$, we see the punctured code still has reasonable performance (unlike no CPM). At this overhead, it performs similarly to the next highest available rate code of $R = 4/5$ indicating that, at this point, it may be preferable to use the next code in the standard for higher rates rather than puncture the lower rate codes. $R = 4/5$ is not achieved until $\Delta = 16.7\%$, at which point the performance dramatically degrades (like the case with no CPM). This occurs as a result of the LDPC decoder failing to converge satisfactorily. However, we conjecture that optimized puncturing patterns [13] may be able to reduce this gap. This is the subject of ongoing research.

Our simulation results provide evidence for augmentation of existing codes optimized for specific modulations with rate-compatibility by puncturing. In particular, our study demonstrates that small to moderate Δ can yield attractive trade-offs in rate vs. performance; however, it does not yet appear practical to achieve rates approaching or exceeding the next available ARTM LDPC code with random puncturing alone. From comparison with the non CPM results, the modulation feedback certainly appears to help. Furthermore, the less prominent error-floor features of the modulated version may indicate that more robust puncturing performance is possible for longer block lengths. In this regime, puncturing does not have as strong an impact, since most of the loss in performance due to puncturing comes from finite length effects of the SPA decoder and LDPC code (which itself improves with code length).

In order to better examine the region of interest around the $R = 4/5$ target code, Figure 9 displays the relevant curves across the punctured $R = 2/3$ codes along with the unpunctured $R = 4/5$ modulated ARTM code in terms of normalized symbol energy E_s/N_0. The gain between the punctured $\Delta = 10\%$, $R(0.1) = 0.741$ code and unpunctured mother $R = 4/5$ code is approximately 0.4 dB at a BER of 10^{-4}, approximately the same as the loss in gain when going from the mother $R = 2/3$ code to punctured $\Delta = 5\%$, $R(0.05) = 0.702$ code, highlighting that a window for further improvements is viable. We see, however, that the gap between the punctured $R(0.1) = 0.741$ code and the mother $R(0) = 0.8$ code has reduced to approximately 0.15 dB at a BER of 10^{-6}, and looks likely to cross at higher SNR since the punctured waterfall is not as steep as the higher rate mother code. Indeed, this error floor behavior continues for $\Delta \geq 10\%$. See, for example, the $R(0.167) = 0.800$ code, which has a significant error floor and loss of over 1dB (and increasing) for BERs lower than 10^{-3} when compared to the mother $R(0) = 0.8$ code.

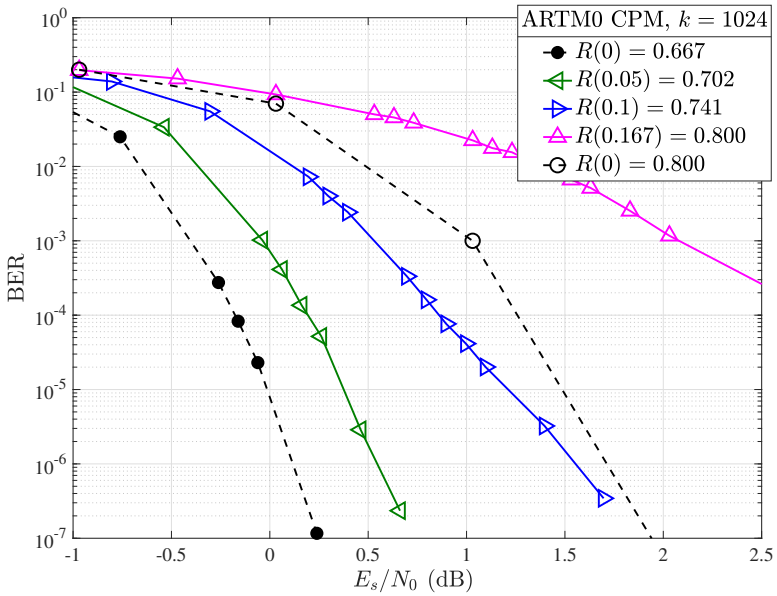

Figure 9. BER performance in terms of E_s/N_0 with CPM for the $R = 2/3$, $k = 1024$ ARTM LDPC code with (dashed) and without (solid) random puncturing.

Simulation results presented previously [10] confirm formation of an error floor below a BER of 10^{-6} for the $R = 2/3$ mother code, perhaps indicating an underlying weakness in that particular code that is exacerbated by puncturing with or without CPM decoding. At low values of overhead

 doi:10.20944/preprints202410.2273.v1

$\Delta \leq 5\%$, no error floor is yet apparent at 10^{-6} and thus error-floor effects seem to dominate only in high overhead regimes. This gives possible allowance for low overhead random puncturing to be realizable without any further considerations necessary. We also note that low-overhead puncturing where $\Delta \leq 10\%$ is well tolerated and performance is robust with only a gradual decrease in the slope of the BER curve within the waterfall region. Operation within the waterfall region is nevertheless practical and could serve as an adaptive mechanism when spectral bandwidth is limited and channel conditions are favorable.

5. Random Shortening of ARTM0 LDPC Codes

As described in Section 2.3, shortening is a complementary technique to puncturing wherein codes of shorter length and lower rate can be constructed from a higher-rate mother code. To accomplish this in our proposed rate-compatible architecture, infinite reliabilities are assigned for the selected (shortened) information bits whose positions are known to both the transmitter and receiver, removing those bits from the resulting codeword. This corresponds to the removal of the matching information columns from the LDPC code's parity-check matrix, reducing the code rate as given in (6).

5.1. Model and Method

The system model for shortening is nearly identical to that provided in Section 4.1 for puncturing (i.e., Figures 3 and 4), with the only alteration being to the function blocks ϕ and ϕ^{-1} and the inclusion of a function block S that is used to shorten the information sequence. In the shortening transmitter model (Figure 10), module S forces s bits in the input sequence, $\mathbf{x}$, to zero, producing shortened sequence $\mathbf{x}_S$ which serves as input to the LDPC encoder. Since the encoder is systematic, the corresponding code symbols will also be zero, as prescribed by our shortening model. Note also that this excludes any parity bits from being shortened (fixed to zero) and effectively removes the corresponding rows from the generator matrix utilized by the LDPC encoder.

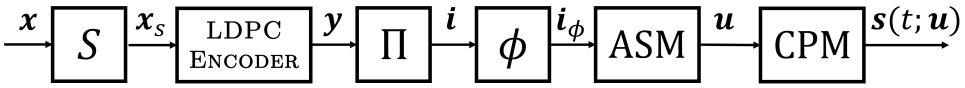

Figure 10. Modification to transmitter circuit includes module S to shorten the information bit sequence.

In the shortening transmitter and receiver models, ϕ operates in a similar way to puncturing, but now only removes the known (shortened) *information* bits of the code (after interleaving). With puncturing, ϕ^{-1} operates to insert zero-valued LLRs, corresponding to an equal probability of a bit being a zero or one since this information is not shared between the transmitter and receiver prior to transmission (but must still be estimated). However, in the shortening receiver model (see Figure 11), module ϕ_s^{-1} operates by inserting *infinite* valued LLRs (practically, a very large positive value) in the exact positions corresponding to the shortening pattern selected by the transmitter. This ensures the shortened symbols are fixed to zero, removing them from the parity check equations performed by the LDPC decoder. Similar to puncturing, we will consider various values of shortening overhead $\alpha_s = s/n$, as defined in Section 2.3, or as a percentage $\Delta_s = \alpha_s \cdot 100\%$.

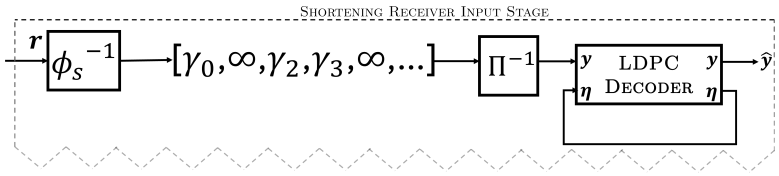

Figure 11. Simplified input stage of the shortening decoder showing an example of the ϕ^{-1} module producing infinite LLR outputs.

5.2. Numerical Results

Simulation results were obtained with the same experimental setup used for random puncturing, as described in Section 4.3, with the exception that a different random shortening pattern is selected per transmitted codeword. Both the rate and block length of the mother code play an important role in the performance of ARTM0 codes under random shortening. If we first consider the performance of

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2024 doi:10.20944/preprints202410.2273.v1

the shortened $k = 1024$, $R = 2/3(b)$ and $R = 4/5$ codes in terms of symbol energy E_s/N_0 (Figure 12), we observe an improvement in performance with increasing Δ_s (and lowering rate $R_s(\alpha_s)$) as expected. However, when we adjust for rate and consider the same examples in terms of information bit energy E_b/N_0 (Figure 13), no gain is observed by shortening the mother $R = 2/3$ code with $\Delta_s = 10\%$, and an observed loss as we further increase to $\Delta_s = 20\%$. However, with the higher rate $R = 4/5$ mother code, the larger gains observed in E_s/N_0 from Figure 12 do translate to improvements in terms of E_b/N_0. We observe that each 10% step increases in shortening from $\Delta_s = 10\%$ to $\Delta_s = 20\%$ produces an approximate gain of about 0.05dB per step at a BER of 10^{-5}. We remark that the with $\Delta_s = 40\%$ (not shown), the shortened code rate is $R_s(0.4) = 2/3$, but the performance falls short of the mother $R = 2/3$ mother code. Similar results to those of the $R = 2/3$ code are obtained for the shortened $R = 1/2$ codes, where gains are observed in terms of E_s/N_0, but not when correcting for rate with E_b/N_0. These results indicate the short ($k = 1024$) ARTM0 block codes of low rate are less amenable to shortening, making rate-matching difficult for lower-rate mother codes, but possible with the higher rate mother codes. As with the random puncturing, it is possible that specific shortening patterns could yield gains over random shortening.

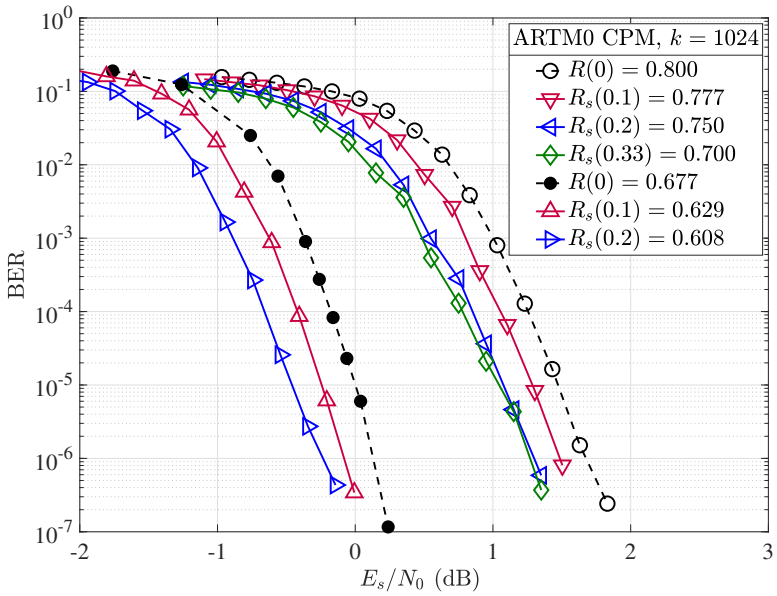

Figure 12. Random shortening applied to the ARTM0 $R = 2/3$ and $R = 4/5$ codes in terms of symbol energy E_s/N_0.

If larger block sizes are considered, shortening produces better results in terms of overall BER performance, especially for higher-rate codes. As shown in Figure 14, a coding gain of 0.2dB is observed in the waterfall region of the $R = 4/5$ code with information block size $k = 4096$ and $\Delta_s = 20\%$ whereas a gain of approximately 0.1dB was observed in the code with $k = 1024$ of the same rate and the same amount ofshortening. This implies that longer length block codes may be more amenable to random shortening, making rate-matching easier to accomplish with acceptable losses in performance.

For both block sizes considered, the code with mother rate $R = 4/5$ shortened by $\Delta_s = 20\%$ produces a code of rate $R = 3/4 = k/(k+1)$, providing a convenient rate-compatible option for framing and hardware implementation. For the $R = 2/3$ mother code, moving to a larger block size of $k = 4096$ does produce a slight gain at $R_s(0.2) = 0.583$ within a narrow portion of the waterfall region, which was not observed for random shortening of the corresponding $k = 1024$ code. Even with the longer blocklengths, random shortening may have a negative impact on performance when adjusting for rate.

 doi:10.20944/preprints202410.2273.v1

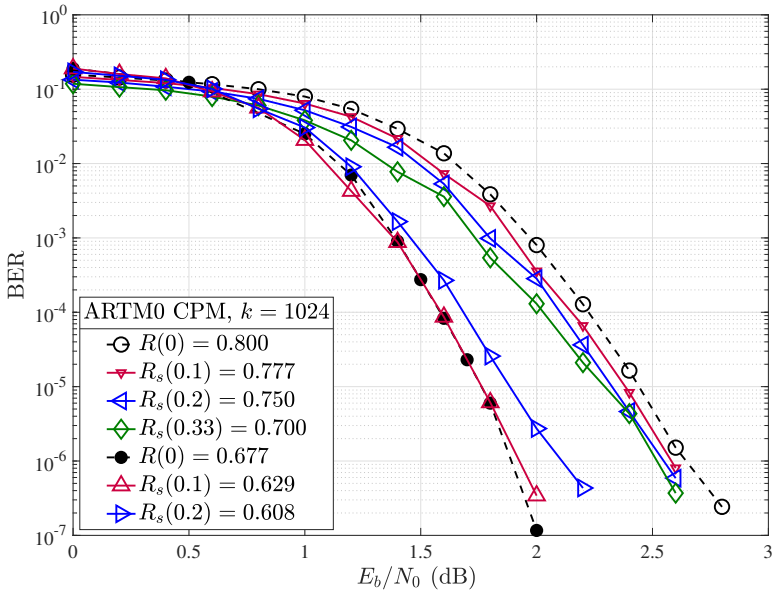

Figure 13. Random shortening applied to the ARTM0 $R = 2/3$ and $R = 4/5$ codes in terms of information symbol energy E_b/N_0.

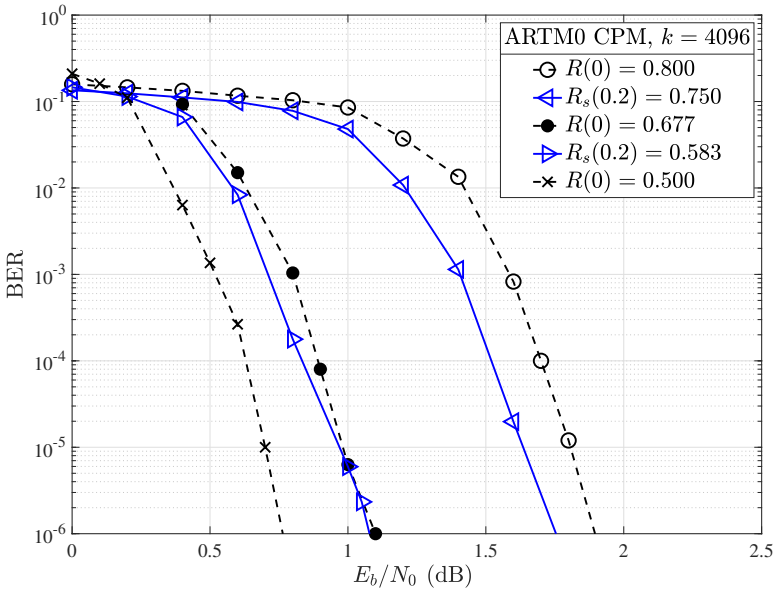

Figure 14. Random shortening applied to the ARTM0 $R = 4/5$ and $R = 2/3$ codes with $k = 4096$.

5.3. Puncturing vs. Shortening

Finally, we briefly discuss the trade-offs observed by puncturing a lower rate ARTM0 code verus shortening a higher rate ARTM0 code. Figure 15 shows the results obtained for random puncturing of the $R = 2/3$, $k = 1024$, code with $\Delta = 5$ and 10% (solid lines) alongside those obtained for random shortening of the $R = 4/5$, $k = 1024$, code with $\Delta_s = 20$ and 33% (dot-dash lines) with a goal of hitting target rates of approximately 0.7 (blue) and 0.75 (red). We observe that for rates close to the lower rate ARTM0 code, puncturing offers the best performance whereas for the higher rates between the mother codes, it may be preferable to shorten the higher rate code. For example, to obtain $R = 0.7$ we find that it is a better strategy to puncture the $R = 2/3$ mother code; whereas for a target rate of $R = 3/4$ we find that it is better to shorten the mother $R = 4/5$ code.

 doi:10.20944/preprints202410.2273.v1

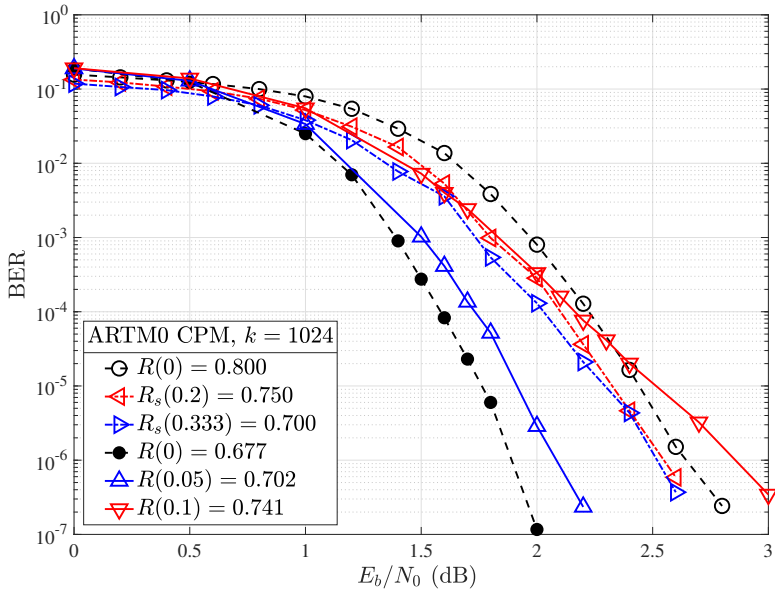

Figure 15. BER performance for the $R = 2/3$, $k = 1024$ ARTM LDPC code with random puncturing (solid) alongside the $R = 4/5$, $k = 1024$ ARTM LDPC code with random shortening (dot-dash).

6. Hardware Considerations

Random puncturing and shortening may be implemented by the insertion of a switch in the registers that serve as buffers to the inputs of each module in the signal chain. At random intervals chosen by the overhead parameters, block length, and system bus speed, the switch is placed in the high impedance position for one clock cycle. Depuncturing/deshortening is accomplished by insertion of a random delay, at which point a zero-value (puncturing) or infinite value (shortening) is inserted into the buffer. Although this introduces additional latency, because all modules operate blockwise, it is possible to perform both puncturing and depuncturing in parallel.

Thus when operating in a low-overhead regime as our results suggest, random puncturing and shortening offer robust rate-compatible performance with no changes to the submodules of Figures 3 and 4. Since no substantive hardware alterations are required, this allows for new operational modes at little expense in terms of latency and memory. However, it is worth noting that because the LDPC SPA decoder still operates on the full depunctured/deshortened sequence, SPA decoding computational complexity will remain largely unchanged.

7. Conclusions

In this paper we have laid the groundwork for the construction of rate-compatible LDPC codes for IRIG-106 waveforms by the use of random puncturing and shortening. Our initial results show robust performance with codes designed for deployment with ARTM CPM modulation and open promising avenues for further improvement. Such systems would allow for the adaptive allocation of bandwidth in response to changing demand or channel conditions without requiring significant changes to the underlying encoding and decoding hardware and thus represent a low-complexity, low-cost implementation option. Furthermore, because puncturing is a type of erasure channel, it is easily modeled and those results are readily extended to other channel models, reducing design iteration schedules.

By isolating the LDPC encoder and decoder, we were able to show the considerable coding gain afforded by the CPM-LDPC decoder loop, reinforcing that codes designed specifically for these modulations are better optimized than codes that do not consider modulation during design. Though in neither case did we observe that random puncturing alone achieves the same performance as a code designed for that specific target rate, the gap is much more tractable with modulation, having a span of 0.5 dB, whereas without modulation the required overhead elicits an impractical error floor. Shortening $k = 4096$ codes with modulation is a viable method for producing rates in-between the mother code rates, but the technique works best for shortening the highest rate $R = 4/5$ code with diminishing returns for lower-rate mother codes.

 doi:10.20944/preprints202410.2273.v1

In future work, other codes, including those of different rates and lengths, will be considered in order to better understand the interplay between the strength of the code and its resilience to puncturing and shortening under CPM. Modulation appears to be no impediment to the implementation of puncturing, with perturbations in the codeword's structure most strongly felt within the inner LDPC decoding loop of Figure 4. Because the code we are using in our model is designed specifically to take advantage of information from the CPM decoder, further work should examine the selection of puncturing patterns which least disturb this feedback mechanism in order to preserve its portion of the total coding gain, which was observed to be in excess of 2 dB. Furthermore, analytical results from density evolution on the BEC indicate not only that preferential puncturing patterns exist, but that catastrophic patterns exist which yield extremely low decoding thresholds and should be avoided [14]. This knowledge will help inform the selection of nonrandom or quasi-random puncturing patterns which may bridge the gap between the punctured and mother codes seen in our present analysis. It will also better describe the limits of the code design procedure and may offer new avenues for optimization.

Author Contributions: Investigation of techniques and system simulation, original draft preparation, A.D.C.; Asymptotic analyses, writing and editing, D.G.M.M.; Construction of ARTM codes and development of the iterative CPM decoding framework, E.P. All authors have read and agreed to the published version of this manuscript.

Funding: This material is based upon work supported by the National Science Foundation under Grant Nos. CNS-2148358 and HRD-1914635.

Acknowledgments: This article is a revised and expanded version of a paper entitled "Spectrally Efficient LDPC Codes for IRIG-106 Waveforms via Random Puncturing," which was presented at the 2024 International Conference on Telemetry, Glendale, AZ, Oct. 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ARTM	Advanced Range TeleMetry
ASM	Attached Sync Marker
AWGN	Additive White Gaussian Noise
BEC	Binary Erasure Channel
BER	Bit Error Rate
BP	Belief Propagation
BPSK	Binary Phase-shift Keying
CPM	Continuous Phase Modulation
FEC	Forward Error Correction
FM	Frequency Modulation
LDPC	Low-Density Parity-Check
LLR	Log-likelihood Ratio
MSA	Min-sum Algorithm
PCM	Pulse Code Modulation
QC	Quasi-cyclic
SISO	Soft-in, Soft-out
SOQPSK-TG	Telemetry Group version of Shaped-Offset Quadrature Phase Shift Keying
SPA	Sum-product Algorithm

References

1. R. Gallager, "Low-density parity-check codes," *IRE Trans. on Info. Theory*, vol. 8 no. 1, 1962.
2. D. J. C. MacKay, "Near Shannon Limit Performance of Low Density Parity Check Codes," *Elect. Letters*, vol. 33 no. 6, 1997.
3. T. J. Richardson and R. L. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. on Info. Theory*, vol. 47 no. 2, 2001.
4. Z. Li, et al., "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. on Comms.*, vol. 54 no. 1, 2006.

 doi:10.20944/preprints202410.2273.v1

5. T. Richardson and H. Jin, "LDPC encoding methods and apparatus," *U.S. Patent No. 7,346,832 (Assigned to Qualcomm Inc.)*, 2008.
6. X-Y Hu, et al., "Efficient implementations of the sum-product algorithm for decoding LDPC codes," *Proc. IEEE Global Telecomms. Conf.*, vol. 2, cat. no. 01CH37270, 2001.
7. C-C. Cheng, et al., "A fully parallel LDPC decoder architecture using probabilistic min-sum algorithm for high-throughput applications," *IEEE Trans. on Circ. and Sys. I: Reg. Papers*, vol. 61 no. 9, 2014.
8. Range Commanders Council Telemetry Group, Range Commanders Council, "IRIG Standard 106-23," *Telemetry Standards*, White Sands Missile Range, New Mexico, 2024 (irig106.org).
9. E. Perrins, "LDPC codes for IRIG-106 waveforms: Part I–Code Design," *Proc. Int. Telemetering Conf.*, (Las Vegas, NV), Oct. 2023.
10. E. Perrins, "LDPC codes for IRIG-106 waveforms: Part II–Receiver Design," *Proc. Int. Telemetering Conf.*, (Las Vegas, NV), Oct. 2023.
11. J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. on Comms.*, vol. 36 no. 4, 1988.
12. J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. on Info. Theory*, vol. 50 no. 11, 2004.
13. H. Zhou, D. G. M. Mitchell, N. Goertz, and D. J. Costello Jr., "Robust Rate-Compatible Punctured LDPC Convolutional Codes," *IEEE Trans. on Comms.*, vol. 61 no. 1, Nov. 2013.
14. D. G. M. Mitchell, M. Lentmaier, A. Pusane, and D. J. Costello Jr., "Randomly Punctured LDPC Codes," *IEEE Journ. on Selected Areas in Comms.*, vol. 34 no. 2, Feb. 2016.
15. A. D. Cummins, D. G. M. Mitchell, and E. Perrins, "Spectrally Efficient LDPC Codes for IRIG-106 Waveforms via Random Puncturing," *Proc. Int. Telemetering Conf.*, (Glendale, AZ), Oct. 2024.
16. S. Lin and D. J. Costello, Jr., "Error Control Coding: Fundamentals and Applications," Prentice Hall, 2004.
17. T. Tian and C. R. Jones, "Construction of rate-compatible LDPC codes utilizing information shortening and parity puncturing," *EURASIP J. Wirel. Commun. Netw.*, vol. 2005, no. 5, pp. 789-795, Oct. 2005.
18. S. -R. Kim and D. -J. Shin, "Lowering Error Floors of Systematic LDPC Codes Using Data Shortening," *IEEE Communications Letters*, vol. 17, no. 12, pp. 2348-2351, Dec. 2013.
19. A. Suls, Y. Lefevre, J. Van Hecke, M. Guenach and M. Moeneclaey, "Error Performance Prediction of Randomly Shortened and Punctured LDPC Codes," *IEEE Communications Letters*, vol. 23, no. 4, pp. 560-563, April 2019.
20. J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," Jet Propulsion Laboratory, Pasadena, CA, INP Progress Report 42-154, Aug. 2003.
21. R. G. Gallager, "Low-density parity-check codes," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1963.
22. D. Divsalar, S. Dolinar, C. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.
23. E. C. Boyle and R. J. McEliece, "Asymptotic weight enumerators of randomly punctured, expurgated and shortened code ensembles," in *Proc. Forty-Sixth Annual Allerton Conference*, Monticello, IL, Sept. 2008.
24. C.-H. Hsu and A. Anastasopoulos, "Capacity achieving LDPC codes through puncturing," *IEEE Trans. Inf. Theory*, vol. 54, no. 10, pp. 4698–4706, Oct. 2008.
25. F. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Info. Theory*, vol. 47 no. 2, 2001.