International Zurich Seminar on Information and Communication (IZS), February 26 – 28, 2020

Joint Decoding of Distribution Matching and Error Control Codes

Patrick Schulte[1], Wafa Labidi[2], Gerhard Kramer[1]
[1]Institute for Communications Engineering, Technical University of Munich, 80333, Munich, Germany
[2]Institute of Theoretical Information Technology, Technical University of Munich, 80333, Munich, Germany

***Abstract*—An improved decoder for low-density parity-check (LDPC) codes and for probabilistic amplitude shaping with constant composition distribution matching (CCDM) is presented. The decoder combines standard LDPC belief propagation with a soft-input soft-output processor that exploits the constraints imposed by CCDM and it gains up to 0.5 dB at a frame error rate of 10^{-4} for a block-length $n = 192$ 5G code.**

I. Introduction

Probabilistic amplitude shaping (PAS) [1] is a block based probabilistic shaping (PS) technique that induces a non-uniform distribution on a signal constellation. A distribution matcher (DM) encodes a message into a non-linear set that satisfies a constraint on the average symbol distribution. A systematic forward error correction (FEC) encoder preserves the distribution in the systematic part.

A constant composition distribution matcher (CCDM) [2] is a DM that imposes a common empirical distribution on the constellation points' amplitudes within a block. The CCDM thus introduces dependencies over all symbols in a block. For very long blocks, the PAS rate is not affected by these dependencies, but systems with short length DMs suffer in transmission rate [3]. In [4]–[8], DMs with smaller rate-loss are proposed. In [9] the dependencies introduced by an extremely short 4-D shell mapping (SMDM) [4]–[6] are resolved by a 4-D demodulator. The authors of [10] use polar codes with list decoding and check if the codeword candidates fulfill the constant composition (CC) constraint.

PAS uses a systematic FEC encoder in a manner similar to the Bliss scheme [11] for constrained sequence coding. To improve the Bliss scheme's performance, [12] and [13] use a supplementary soft input soft output (SISO) decoder and iterate with the usual FEC decoder. We adopt this approach for PAS and let a low-density parity-check (LDPC) decoder iterate with a SISO CC code decoder based on the forward backward (BCJR) algorithm to improve performance. For this purpose, we introduce the trellis of a CC code. The resulting decoder is a generalized LDPC (GLDPC) decoder [14] with a non-linear constraint.

This paper is structured as follows. In Sec. II we introduce notation and the basic components of PAS. In Sec. III we introduce the interface of the BCJR algorithm and construct a trellis for CC codes. In Sec. IV we show combinations of BCJR and LDPC-belief propagation (BP) decoders. Simulation results are presented in Sec. V. We draw conclusions in Sec. VI.

II. Preliminaries and Notation

A. Notation

We write matrices in capital bold letters $\mathbf{L}$, random variables with uppercase sans-serif letters X, and their realizations with lowercase letters x. Let A be a discrete random variable with probability mass function (pmf) P_{A} defined on the set $\mathcal{A}$. The *entropy* of a random variable A is

$$\mathbb{H}(\mathsf{A}) = \sum_{a \in \mathrm{supp}(P_{\mathsf{A}})} -P_{\mathsf{A}}(a) \log_2 \left(P_{\mathsf{A}}(a)\right) \tag{1}$$

where $\mathrm{supp}(P_{\mathsf{A}}) \subseteq \mathcal{A}$ is the support of P_{A}, i.e., the subset of a in $\mathcal{A}$ with positive probability. We denote a length n vector of random variables as $\mathsf{A}^n = \mathsf{A}_1\mathsf{A}_2 \cdots \mathsf{A}_n$ with realization $a^n = a_1 a_2 \cdots a_n$, and the number of occurrences of letter $\alpha \in \mathcal{A}$ in a^n as $n_\alpha(a^n)$. Next, we describe the channel model and the components of the PAS transceiver.

B. Channel Model

For transmission we consider M-amplitude shift keying (ASK), i.e., transmission symbols X take on values in $\mathcal{X} = \{-M+1, -M+3, \cdots, M-3, M-1\}$. Each symbol can be factored into a sign and amplitude

$$\mathsf{X} = \mathsf{A} \cdot \mathsf{S}. \tag{2}$$

The corresponding amplitude set is

$$\mathcal{A} = \{\alpha_1, \alpha_2, \cdots, \alpha_{M/2}\} = \{1, 3, \cdots, M-1\}. \tag{3}$$

We consider additive white Gaussian noise (AWGN), i.e., the output symbols of the channel are obtained via

$$\mathsf{Y} = \mathsf{X} + \mathsf{Z} \tag{4}$$

where Z is a Gaussian random variable with zero mean and variance σ^2. The signal-to-noise ratio (SNR) is

$$\mathrm{SNR} = \frac{\mathbb{E}\left[\mathsf{X}^2\right]}{\sigma^2}. \tag{5}$$

C. Probabilistic Amplitude Shaping

PAS [1] is a coded modulation scheme that can approach the Shannon capacity for the AWGN channel [15], [16] and is rate adaptive. An important building block is the DM which encodes messages into sequences of amplitudes with a desired average distribution. One can use any DM, and common choices are CCDM and SMDM [6]. A systematic LDPC encoder generates parity bits from a binary representation of the amplitudes. The parities serve as signs for the amplitudes.

For high rate codes, additional source bits are encoded without distribution matching. We refer to [1] for a detailed review of PAS.

D. Labeling Function

An invertible labeling function β converts m bits to an $M = 2^m$-ary symbol $x \in \mathcal{X}$:

$$\beta(b_1, \cdots, b_m) = x. \tag{6}$$

The inverse function is

$$\beta^{-1}(x) = [b_1, \cdots, b_m]. \tag{7}$$

We refer to the j-th bit of the label by $\beta_j^{-1}(x)$. We use a binary reflected Gray code (BRGC) [17] where b_1 decides the symbol's sign, i.e., we have

$$\beta_A(b_2, \cdots, b_m) = |\beta(0, b_2, \cdots, b_m)| = |\beta(1, b_2, \cdots, b_m)|. \tag{8}$$

The notation $b_{i,j}$ refers to the j-th bit of the i-th symbol x_i. We write $\boldsymbol{B}$ to refer to all bits $b_{i,j}$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$.

E. Demodulation

We consider a symbol-wise demodulator that is aware of the signal statistics P_{A}, P_{B_j}. The log-likelihoods (LLs) $\tilde{L}_i(x)$ of the i-th transmitted symbol are

$$\tilde{L}_i(x) = \log(p_{\mathsf{Y}|\mathsf{X}}(\mathsf{Y}_i|\mathsf{X}_i = x) \cdot P_{\mathsf{X}}(x)), \forall x \in \mathcal{X}. \tag{9}$$

The demodulator calculates the bit-wise LLs

$$\tilde{L}_{i,j}(b) = \log(p_{\mathsf{Y}|\mathsf{B}_j}(\mathsf{Y}_i|\mathsf{B}_{i,j} = b) \cdot P_{\mathsf{B}_j}(b)), \forall b \in \{0, 1\}. \tag{10}$$

Thus, one symbol-channel splits into m parallel bit-channels [1]. The log-likelihood ratio (LLR) of the j-th bit in the i-th transmitted symbol is

$$L_{i,j} = \tilde{L}_{i,j}(0) - \tilde{L}_{i,j}(1). \tag{11}$$

For convenience, we collect LLs and LLRs in the matrices $\tilde{\mathbf{L}}$ and $\mathbf{L}$, respectively. The (i, j)-th entry of the LLR matrix $\mathbf{L}$ corresponds to $L_{i,j}$. The (i, j)-th entry of the LL matrix $\tilde{\mathbf{L}}$ corresponds to $\tilde{L}_i(\xi_j)$, $\xi_j \in \mathcal{X}$.

F. LDPC Codes and BP Decoding

A (n, k) LDPC code [18] is a binary linear block code described by an $r \times n$ parity-check matrix $\mathbf{H}$ with entries $h_{i,j}$, $i = 1, 2, \ldots, r$, $j = 1, 2, \ldots, n$, where $r \geq n - k$. LDPC codes can be visualized through a bipartite graph also known as the Tanner graph $\mathcal{G}$. This graph consists of a set $\mathcal{V} = \{V_1, V_2, \cdots, V_n\}$ of n variable nodes, a set $\mathcal{C} = \{C_1, C_2, \cdots, C_r\}$ of r check nodes and a set $\mathcal{E} = \{e_{j,i}\}$ of edges. The check node C_j is connected to the variable node V_i through the edge $e_{j,i}$ if the entry $h_{i,j}$ of the parity-check matrix is one.

An LDPC BP decoder operates on LLRs [1]. Based on the channel observations $\mathbf{L}^{\mathsf{CH}}$, the LDPC decoder outputs the APP LLRs:

$$\mathbf{L}^{\mathsf{APP}} = \mathbf{L}^{\mathsf{CH}} + \mathbf{L}^{\mathsf{E,LDPC}} \tag{12}$$

where $\mathbf{L}^{\mathsf{E,LDPC}}$ denotes the extrinsic information.

G. Constant Composition Distribution Matching

The type $\boldsymbol{t}$ of a sequences a^n expresses how many times each letter $\alpha \in \mathcal{A}$ appears in a^n, i.e., we have

$$\boldsymbol{t} = (n_{\alpha_1}(a^n), n_{\alpha_2}(a^n), \cdots, n_{\alpha_{|\mathcal{A}|}}(a^n)). \tag{13}$$

The set of sequences of type $\boldsymbol{t}$ is

$$\mathcal{T}_{\boldsymbol{t}} = \{a^n \in \mathcal{A}^n \, | n_{\alpha_i}(a^n) = t_i \, , \quad i = 1, \cdots, |\mathcal{A}|\} \tag{14}$$

where t_i is the i-th entry of $\boldsymbol{t}$. The cardinality is

$$|\mathcal{T}_{\boldsymbol{t}}| = \frac{n!}{\prod_{i=1}^{|\mathcal{A}|} t_i!}. \tag{15}$$

The CCDM is a function

$$f_{\text{ccdm},\boldsymbol{t}} : \{0, 1\}^k \to \mathcal{C}_{\text{ccdm}} \tag{16}$$

where $\mathcal{C}_{\text{ccdm}}$ is a subset of $\mathcal{T}_{\boldsymbol{t}}$. Thus, all codewords of the CCDM have the same type and therefore the same empirical distribution. The dematcher $f_{\text{ccdm},\boldsymbol{t}}^{-1}$ implements the inverse operation. For large n, the CCDM rate

$$R_{\text{ccdm}} = k/n \tag{17}$$

tends to $\mathbb{H}(P_{\mathsf{A}})$ with $P_{\mathsf{A}}(i) = \frac{t_i}{n}$ [2], where $\mathbb{H}(P_{\mathsf{A}})$ is the entropy of a discrete memoryless source (DMS) with symbol probabilities P_{A}. The difference

$$R_{\text{loss}} = \mathbb{H}(P_{\mathsf{A}}) - R_{\text{ccdm}} \tag{18}$$

is called the rate-loss R_{loss} [3]. In [19, Sec. IV] the CCDM rate-loss is upper and lower bounded by $\mathcal{O}(\log(n)/n)$ where n is the block length. The rate-loss of a CCDM is negligible for large blocks, but for short blocks the CC constraint adds substantial redundancy. Consider a sequence a^n with a type constraint. If we know all symbols except for one, we can recover its value by counting how often each letter appears. This holds for any constraint length n. We want to exploit the redundancy of a CC code at the decoder.

III. Forward-Backward Algorithm for Constant Composition Codes

The BCJR algorithm [20], also known as the forward-backward algorithm, is a SISO algorithm that calculates the a posteriori symbol probabilities

$$P^{\mathsf{APP}}(a_i) = P(a_i|\tilde{\mathbf{L}}) \tag{19}$$

where $\tilde{\mathbf{L}}$ are LLs and a_i is the i-th transmitted symbol. From these probabilities, we can compute the extrinsic LLs $\tilde{\mathbf{L}}^{\mathsf{E}}$ [20]. For binary codes, the input interface may be LLRs, because we can convert easily from LLRs to LLs and vice versa. The constant composition BCJR (CCBCJR) decoder builds the code trellis from the type vector $\boldsymbol{t}$, i.e. it is a function

$$\text{CCBCJR} : \tilde{\boldsymbol{L}} \times \boldsymbol{t} \mapsto \tilde{\boldsymbol{L}}^{\mathsf{E}}. \tag{20}$$

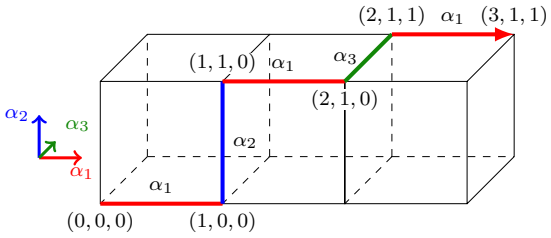

Fig. 1. Constant composition code trellis for type $\boldsymbol{t} = (3, 1, 1)$. This trellis consists of 16 states and 28 branches and represents 20 different CC codewords, thus paths.

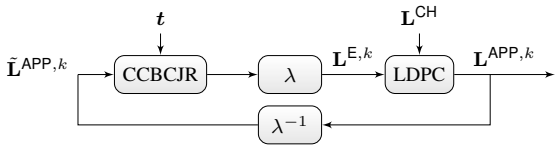

Fig. 2. Symbol-based decoder.

A. CC Code Trellis

The construction of the CC trellis borrows ideas from [21]. The trellis states are tuples

$$\mathcal{S} = \{0, 1, \cdots, n_{\alpha_1}\} \times \{0, 1, \cdots, n_{\alpha_2}\} \times \cdots \{0, 1, \cdots, n_{\alpha_{|\mathcal{A}|}}\}. \tag{21}$$

The number of states in the trellis is

$$|\mathcal{S}| = \prod_{\alpha \in \mathcal{A}} (n_\alpha + 1) \tag{22}$$

and the number of edges is

$$E = \sum_{\alpha \in \mathcal{A}} n_\alpha \prod_{\alpha' \neq \alpha} (n_{\alpha'} + 1). \tag{23}$$

The initial and final states are $(0, \cdots, 0)$ and $(n_{\alpha_1}, \cdots, n_{\alpha_{|\mathcal{A}|}})$, respectively. State $s \in \mathcal{S}$ is connected to an earlier state $s' \in \mathcal{S}$ via symbol α_q if all entries are identical except for the q-th entry of s that is augmented by one.

Example 1. Consider a CC code on the alphabet $\mathcal{A} = \{\alpha_1, \alpha_2, \alpha_3\}$ and with type $\boldsymbol{t} = (3, 1, 1)$. The trellis is depicted in Fig. 1. It consists of $|\{0, 1, 2, 3\}| \cdot |\{0, 1\}| \cdot |\{0, 1\}| = 16$ states. The colored path corresponds to the sequence $(\alpha_1\alpha_2\alpha_1\alpha_3\alpha_1)$. It includes three increment-steps of α_1, one increment-step of α_2, and one increment-step of α_3, and therefore matches the sequence type.

Note that an CCBCJR decoder assumes that that we may use the complete set $\mathcal{T}_{\boldsymbol{t}}$ of sequences of type $\boldsymbol{t}$, however $\mathcal{C}_{\text{ccdm}}$ is usually only a subset [2].

IV. Joint Decoding

We study how the decoder can exploit CC code properties to decrease the error probability.

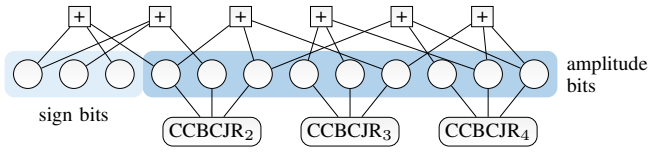

Fig. 3. Tanner graph of a naive bit-based decoder for $m = 4$ and $n = 3$. The amplitude bit variable nodes are connected to the respective BCJR node. The sign bit variable nodes are not connected to a BCJR node.

A. Symbol-Based Decoder

The symbol-based decoder consists of a CCBCJR decoder and a LDPC decoder that exchange messages iteratively, see Fig. 2. The BCJR decoder has a symbol based interface, while the LDPC decoder has a bit based interface. The demodulator provides the LLs of the symbols and bit levels. The symbol-wise LLs are passed to the CCBCJR decoder. The LDPC decoder and the CCBCJR decoder iterate extrinsic information $\mathbf{L}^{\mathsf{E}}$. We use functions λ and λ^{-1} to convert from symbol based to bit based and vice versa. The function λ converts LL into LLRs via

$$L_{i,j} = \ln \left(\frac{\sum\limits_{x:\beta_j^{-1}(x)=0} \exp\left(\tilde{L}_i(x)\right)}{\sum\limits_{x:\beta_j^{-1}(x)=1} \exp\left(\tilde{L}_i(x)\right)} \right). \tag{24}$$

The function λ^{-1} converts from bit-level to symbol-level. For simplicity, we assume for a fixed i that the $\mathsf{B}_{i,j}$, $j = 1, 2, \ldots, m$, are pairwise independent given Y_i. The conversion is then

$$\tilde{L}_i(x) = \log \left(\prod_{j=2}^{m} \frac{\exp(L_{i,j} \cdot (1 - 2\beta_{A,j}^{-1}(x)))}{1 + \exp(L_{i,j} \cdot (1 - 2\beta_{A,j}^{-1}(x)))} \right). \tag{25}$$

Note that $1 - 2\beta_{A,j}^{-1}(x)$ is 1 for the bit 0 and -1 for the bit 1.

B. Bit-Based Decoder

The number of states and edges of the CCBCJR decoder increases exponentially with the alphabet size and polynomially in n. One idea to decrease complexity is to replace one $|\mathcal{A}|$-ary CCBCJR decoder by $\log_2 |\mathcal{A}|$ binary CCBCJR decoders. Additionally, the conversion functions λ, λ^{-1} become obsolete.

Consider a transmission sequence x^n with type constraint $\boldsymbol{t}$ on the amplitudes and its binary representation $\boldsymbol{B} \in \{0, 1\}^{n \times m}$ according to the labeling function β, where the entry $b_{i,j}$ corresponds to the j-th bit of the i-th symbol. Let $\boldsymbol{b}_j^{\mathsf{I}} = b_{1,j}, b_{2,j}, \cdots b_{n,j}$ be the j-th column of $\boldsymbol{B}$, i.e., the j-th bit-level of the binary representation of the symbol sequence. Since x^n has a type $\boldsymbol{t}$ constraint on the amplitudes only, the sign bits are unconstrained. All other bit levels j, $2 \leq j \leq m$ are constrained. We derive the type constraint for each bit-level depending on the type $\boldsymbol{t}$ of the sequence x^n and the labeling function β. The number of zeros in bit-level j is equal to the number of amplitudes in the sequence x^n whose binary

representation is zero in the j-th position, i.e., we have

$$n_b(\boldsymbol{b}_j^{|}) = \sum_{\alpha\in\mathcal{A},\ \beta_j^{-1}(\alpha)=b} n_\alpha(\mathrm{amp}(x^n)) \tag{26}$$

where $\mathrm{amp}(x^n)$ is the element-wise absolute value of x^n, $b \in \{0,1\}$, and $2 \le j \le m$. Thus for one amplitude type constraint $\boldsymbol{t}$, we obtain $m-1$ bit constraints $\boldsymbol{t}_2, \cdots, \boldsymbol{t}_m$, where the index denotes the respective bit-level with

$$\boldsymbol{t}_j = \left[n_0(\boldsymbol{b}_j^{|}), n_1(\boldsymbol{b}_j^{|})\right]. \tag{27}$$

Example 2. Consider a sequence x^n with amplitude constraint $\boldsymbol{t} = [37, 20, 6, 1]$, i.e., 37 ones, 20 threes, 6 fives and 1 sevens, and the BRGC labeling β shown below.

x	-7	-5	-3	-1	1	3	5	7
$\beta(x)$	111	110	100	101	001	000	010	011

We find $n_1(\boldsymbol{b}_2^{|}) = 7$ because the second bit of the labeling β is '1' for amplitudes 5 and 7 and they appear 6 times and once, respectively. The corresponding bit types $\boldsymbol{t}_2$ and $\boldsymbol{t}_3$ are

$$\boldsymbol{t}_2 = [57, 7] \tag{28}$$

$$\boldsymbol{t}_3 = [26, 38]. \tag{29}$$

For decoding, we add $m-1$ BCJR nodes into the Tanner graph, as shown in Fig. 3. Note that the bit-based CCBCJR decoders run independently. Their combined trellises allow sequences that do not fulfill the type constraint $\boldsymbol{t}$.

C. Improved Bit-Based Decoder

Each of the $m-1$ BCJR nodes is connected to n/m nodes. This suggests that the girth, i.e., the shortest cycle in the graph, is small. Loopy BP for small-girth was investigated in [22] and leads to oscillations. There are two basic approaches to deal with this issue. Firstly, we may filter the beliefs and thus attenuate oscillations. Second, we could introduce multiple short length CC constraints on a bit-level, i.e., introduce lower degree CCBCJR nodes which increases both the girth and the rate-loss. We consider only the first approach in this paper.

The LDPC decoder outputs the a posteriori LLRs $\boldsymbol{L}_j^{\mathsf{APP}}$. Based on the channel observation, the type vector $\boldsymbol{t}_j$ and a posteriori information, the j-th BCJR decoder CCBCJR_j generates the extrinsic information $\mathbf{L}_j^{\mathsf{E}}$. The outputs of the $m-1$ CCBCJRs are collected in the matrix $\mathbf{L}^{\mathsf{E}}$. $\mathbf{L}$ and $\mathbf{L}^{\mathsf{E}}$ are then processed by the function

$$\mathrm{g}\left(\mathbf{L}^{\mathsf{CH}}, \mathbf{L}^{\mathsf{E}}, \mathbf{L}^{\mathsf{APP}}, k\right) \approx \mathbf{L}^{\mathsf{CH}} + \underbrace{\left(\mu \cdot \mathbf{L}^{\mathsf{E},k-1} + (1-\mu)\cdot \mathbf{L}^{\mathsf{E},k}\right)}_{\text{prior information}} \tag{30}$$

with $k \ge 1$ and $\mu \in [0,1]$. After a number of iterations, the LDPC decoder outputs new a posteriori information, which is sent back to the CCBCJR decoders. The optimal parameter μ is found by grid search.

D. Computational Complexity Comparison

For the computational complexity analysis, we focus on the number of edges E in the code trellises, since the BCJR complexity is $\Theta(E)$ [23]. This analysis depends on the trellis representation of the CC code.

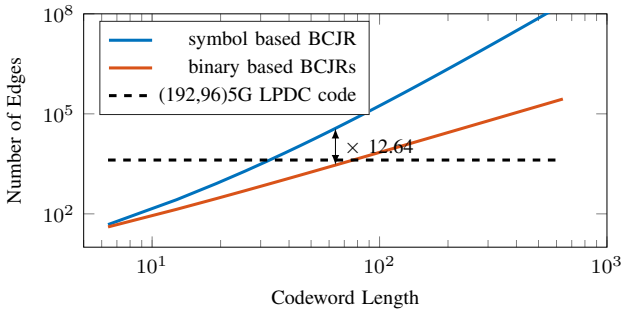

Fig. 4. Number of branches to compute for the bit-based and symbol-based BCJR algorithms. The empirical distribution is $[37, 20, 6, 1]/64$. We interpret (31) and (32) as continuous functions. At output length 64 symbols, the symbol-based BCJR algorithm needs about 12.5 times more states than the binary-based BCJR algorithm. We compare with the number of branches of an iterative LDPC decoder using the BCJR algorithm.

1) Symbol-Based Decoder: For a type $\boldsymbol{t} = [n_1, \cdots, n_{M/2}]$ constraint, we have

$$E_{\text{symb}} = \sum_{i=1}^{M/2} n_i \prod_{j\neq i}(n_j + 1) \tag{31}$$

branches. An increasing alphabet size even for the same block-length may result in a large increase in the number of states and therefore the computational complexity. For a given empirical distribution, the number of states scales with the power of the support of the empirical distribution.

2) Bit-Based Decoder: For the bit-based decoder, we split one amplitude type constraint $\boldsymbol{t}$ into $m-1$ bit constraints $\boldsymbol{t}_2, \cdots, \boldsymbol{t}_m$. The number of edges is then

$$E_{\text{bit}} = \sum_{j=2}^{m} 2n_0(\boldsymbol{b}_j^{|})n_1(\boldsymbol{b}_j^{|}) + n_0(\boldsymbol{b}_j^{|}) + n_1(\boldsymbol{b}_j^{|}). \tag{32}$$

In Fig. 4 we show the number of branches vs. the codeword length for the empirical distribution $[37, 20, 6, 1]/64$. We also add the number of branches that are evaluated during one iteration of LDPC decoding of an (192,96) 5G LDPC code, i.e., we compute the number of branches of all single parity check and repetition nodes. Single parity check and repetition nodes have 4 times and 2 times their degree edges, respectively.

V. Simulation Results

We compare the performance of PAS with the bit-level decoder proposed in [1] with the symbol-based and the heuristically improved bit-based decoder with supplementary CC constrained nodes. We target a spectral efficiency of 1.5 bits per channel use with 8-ASK constellation.

For encoding, we use a DM with type $\boldsymbol{t} = [37, 20, 6, 1]$ from Example 2 and a rate $3/4$ code from the 5G eMBB standard [25] with block length 192. The reference LDPC decoder [1] is biased with the empirical distribution of the FEC input. The symbol-based decoder uses $\boldsymbol{t}$ and the bit-based decoder has two CCBCJRs with $\boldsymbol{t}_2 = [7, 57]$ and $\boldsymbol{t}_3 = [38, 26]$.

Simulation results in Fig. 5 show that the LDPC decoder with a linear combination of $\mathbf{L}^{\mathsf{E,LDPC},k-1}$ and $\mathbf{L}^{\mathsf{E},k}$ outperforms the LDPC decoder with $\mathbf{L}^{\mathsf{E},k}$ as prior information only.

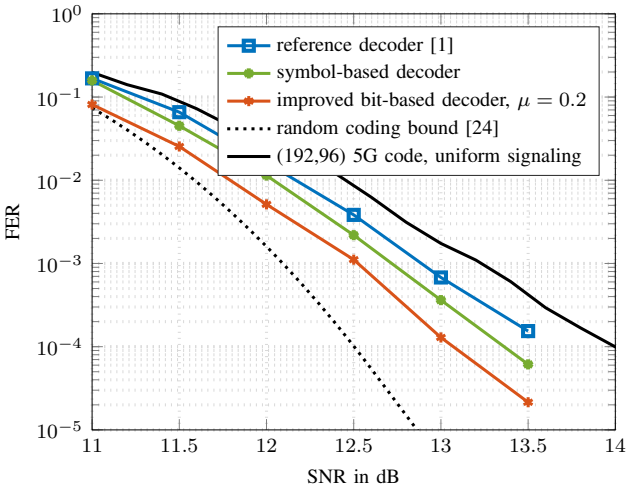

Fig. 5. FER of the different strategies for 24 outer-iterations and 100 inner-iterations. We collected 100 erroneous frames per simulation point. The scheme is implemented by using $8-$ASK with code rate $3/4$ and block-length $n = 192$. The rate-loss R_{loss} is about 0.145 bit/symbol.

We include the performance of a (192,96) 5G LDPC code with an optimized interleaver as a non-shaped baseline with the same spectral efficiency. The bit-based decoding strategy gains 0.5 dB in the simulation setup as compared to the LDPC decoder in [1].

VI. Conclusions and Outlook

A trellis structure for CC codes is introduced. Different decoding strategies based on the combination of BCJR and LDPC decoders are proposed that gain 0.5 dB in the considered short length scenario at a frame error rate of 10^{-4}. In future work, we plan to investigate the design of LDPC codes with CCBCJR nodes. This way long LDPC codes could be combined with short block length DMs that run in parallel during encoding and decoding.

VII. Acknowledgements

We would like to thank Georg Böcherer and Fabian Steiner for continuous support and Gianluigi Liva for the initial idea. Wafa Labidi was supported by the Bundesministerium für Bildung und Forschung (BMBF) through Grant 16KIS1003.

References

[1] G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth efficient and rate-matched low-density parity-check coded modulation," *IEEE Trans. Commun.*, vol. 63, no. 12, pp. 4651–4665, Dec 2015.

[2] P. Schulte and G. Böcherer, "Constant composition distribution matching," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 430–434, Jan 2016.

[3] G. Böcherer, P. Schulte, and F. Steiner, "High throughput probabilistic shaping with product distribution matching," *arXiv preprint arXiv:1702.07510*, 2017.

[4] Y. C. Gültekin, F. M. J. Willems, W. J. van Houtum, and S. Şerbetli, "Approximate enumerative sphere shaping," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, June 2018, pp. 676–680.

[5] Y. C. Gültekin, W. J. van Houtum, S. Şerbetli, and F. M. Willems, "Constellation shaping for IEEE 802.11," in *IEEE Ann. Int. Symp. on Personal, Indoor, and Mobile Radio Commun. (PIMRC)*. IEEE, 2017, pp. 1–7.

[6] P. Schulte and F. Steiner, "Divergence-optimal fixed-to-fixed length distribution matching with shell mapping," *IEEE Wireless Commun. Lett.*, pp. 1–1, 2019.

[7] T. Fehenberger, D. S. Millar, T. Koike-Akino, K. Kojima, and K. Parsons, "Multiset-partition distribution matching," *IEEE Trans. Commun.*, pp. 1–1, 2018.

[8] M. Pikus and W. Xu, "Bit-level probabilistically shaped coded modulation," *IEEE Commun. Lett.*, vol. 21, no. 9, pp. 1929–1932, Sep. 2017.

[9] F. Steiner, F. Da Ros, M. P. Yankov, G. Böcherer, P. Schulte, G. Kramer *et al.*, "Experimental verification of rate flexibility and probabilistic shaping by 4D signaling," in *Proc. Optical Fiber Commun. Conf.* IEEE, 2018, pp. 1–3.

[10] P. Yuan, G. Böcherer, P. Schulte, G. Kramer, R. Böhnke, and W. Xu, "Error detection using symbol distribution in a system with distribution matching and probabilistic amplitude shaping," German WO Application, 10 31, 2016.

[11] W. Bliss, "Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation," *IBM Tech. Discl. Bul.*, vol. 23, pp. 4633–4634, 1981.

[12] J. L. Fan and J. M. Cioffi, "Constrained coding techniques for soft iterative decoders," in *IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 1. IEEE, 1999, pp. 723–727.

[13] A. P. Hekstra, "Use of a d-constraint during LDPC decoding in a Bliss scheme," *arXiv preprint arXiv:0707.3925*, 2007.

[14] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[15] R. A. Amjad, "Information rates and error exponents for probabilistic amplitude shaping," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2018.

[16] G. Böcherer, "Achievable rates for probabilistic shaping," *arXiv preprint arXiv:1707.01134*, 2017.

[17] F. Gray, "Pulse code communication," *US Patent 2632058*, 1953.

[18] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[19] P. Schulte and B. C. Geiger, "Divergence scaling of fixed-length, binary-output, one-to-one distribution matching," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*. IEEE, 2017, pp. 3075–3079.

[20] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[21] J. Schalkwijk, "An algorithm for source coding," *IEEE Trans. Inf. Theory*, vol. 18, no. 3, pp. 395–399, 1972.

[22] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proc. Conf. on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 467–475.

[23] R. J. McEliece, "On the bcjr trellis for linear block codes," *IEEE Transactions on Information Theory*, vol. 42, no. 4, pp. 1072–1092, 1996.

[24] G. Liva and F. Steiner, "pretty-good-codes.org: Online library of good channel codes," http://pretty-good-codes.org, Oct. 2017.

[25] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5G new radio," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 28–34, Mar. 2018.