IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 7, 2019, accepted June 6, 2019, date of publication July 16, 2019, date of current version August 15, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929221

Efficient Implementation of Density Evolution for Punctured Polar Codes

CHRISTOPHER SCHNELLING[1,2], MARKUS ROTHE[2], NIKLAS KOEP[2], RUDOLF MATHAR[2], AND ANKE SCHMEINK[1,2]

[1]Research Group (Information Theory and Systematic Design of Communication Systems), RWTH Aachen University, 52062 Aachen, Germany
[2]Institute for Theoretical Information Technology, RWTH Aachen University, 52062 Aachen, Germany

Corresponding author: Christopher Schnelling (schnelling@ti.rwth-aachen.de)

ABSTRACT Polar codes asymptotically achieve the symmetric capacity of arbitrary binary-input discrete memoryless channels under low-complexity sequential decoding algorithms such as successive cancellation decoding. However, in their original formulation, the block length of polar codes is limited to integer powers of the dimension of the underlying polarization kernel used, thus imposing strict constraints on possible application scenarios. While leeway in the choice of kernel or concatenation with other codes mitigates this drawback to a certain extent, puncturing presents a promising approach to specify the target length of a polar code with much greater flexibility. In this paper, we present an efficient implementation of the construction of punctured polar codes based on density evolution, a crucial tool in the construction of both regular, i.e., unpunctured, as well as punctured polar codes. Our implementation of density evolution covers the construction of both regular and punctured polar codes and allows for treating the construction of both code classes in a unified framework. Using our implementation, we achieve substantial reductions in the number of density convolutions necessary for the construction of punctured polar codes and obtain tight upper bounds on the block error rates.

INDEX TERMS Polar codes, puncturing, density evolution.

I. INTRODUCTION

Polar codes present the first channel coding scheme provably achieving the symmetric capacity of arbitrary binary-input discrete memoryless channels (BDMCs). As they build on explicit low-complexity encoding and decoding methods providing this favourable asymptotic behaviour, polar codes (PCs) give a constructive solution to a long-standing open problem in information theory. Thus, PCs have received a lot of attention since their introduction [1]. These properties come at the cost of restricting possible block lengths to integer powers of two due to the original PC construction based on the two-dimensional polarization kernel [1]

$$\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{1}$$

Several options have been explored to increase the length flexibility of PCs as required in modern communication systems. Concatenated coding schemes including PCs as inner or outer codes present a natural option to achieve this goal. Several such schemes have been presented in the literature, concatenating PCs with, *e.g.*, low-density parity-check (LDPC) codes [2], Reed-Solomon (RS) codes [3], [4], or Bose-Chaudhuri-Hocquenghem (BCH) codes [5]. Alternatively, the two-dimensional kernel $\mathbf{F}$ as in (1) may be replaced by other kernels $\mathbf{K} \in \mathbb{F}_2^{m \times m}$ [6], which yields PCs of lengths m^n when constructing a polarizing matrix from $\mathbf{K}^{\otimes n}$, the n-th Kronecker power of $\mathbf{K}$, where $\mathbb{F}_2 := \{0, 1\}$ denotes the binary finite field.

These approaches facilitate greater length flexibility, but come with substantially complex encoder and decoder structures, and do not promote length flexibility of PC schemes to the desired extent. To overcome this, both puncturing and shortening of PCs based on the standard kernel $\mathbf{F}$ have been considered, targeting length adaptions of PCs as granular as possible, *e.g.*, [2], [7], [8], and [9].

In this work, we focus on efficiently implementing the construction of punctured PCs. Such a code is defined by a PC $\mathcal{C}$ of length $N = 2^n$, $n \in \mathbb{N}$, commonly referred to as the mother code, and a puncturing pattern $\mathcal{P}$ defining the positions of the codewords of $\mathcal{C}$ which are punctured, that is, omitted upon

The associate editor coordinating the review of this manuscript and approving it for publication was Zilong Liu.

transmission over the channel. This results in a punctured PC $\mathcal{C}'$ of length $M = N - |\mathcal{P}|$. Albeit allowing for arbitrary target lengths $M < N$, puncturing codeword positions changes the properties of the coordinate channels, i.e. the virtual channels based on which individual information bits are decoded when using a successive cancellation (SC) decoder.

The changes to the channel polarization induced by puncturing codeword positions necessitate evaluation to construct an information index set for a punctured PC. A tool of paramount importance to assess the error performance of potential information positions under SC decoding is given by a modified version of density evolution (DE). Based on these individual error probability estimates, such index set may be selected in order to minimize the union bound on the block error rate (BLER) under SC decoding. As no information is received for punctured positions, corresponding channel output log-likelihood ratios (LLRs) are set to zero at the SC or successive cancellation list (SCL) decoder [10], [11], hence treating such positions as random bits. Consequently, punctured positions are treated as erasures, which is accounted for by replacing corresponding LLR densities accordingly, thus resulting in a modified version of the DE construction proposed for regular, that is, unpunctured PCs, in [12]. Selecting the index set for a punctured PC containing the indices of the information positions for a given puncturing pattern via a modified DE was first proposed in [7] and [13].

In this work, we present an efficient implementation for the construction of both regular and punctured PCs based on a DE, which allows treating the practical construction problem for both classes of PCs in a unified framework. In general, DE builds on convolutions of probability density functions, *viz.*, densities, of inbound message values, in order to quantify densities of message values produced at both variable and check nodes of a factor graph (FG) representation of a given code. Consequently, the cost of performing a DE is mainly driven by the complexity of these convolution operations. As any practical implementation of DE performs these operations on quantized densities, a large number of samples is necessary to produce accurate results [14]. For any given implementation of the convolution operations, we present an algorithm which efficiently reuses convolution expressions, resulting in a drastic reduction of the number of convolutions necessary for code construction. Depending on the puncturing patterns used, we report significant reductions of the number of convolutions, hence verifying the efficiency of our approach. For puncturing patterns chosen uniformly at random, we observe savings in terms of the number of necessary convolutions between 35 % and almost 70 %, while for quasi-uniform puncturing (QUP), the savings amount to up to 90 %, depending on the length of the mother code. Furthermore, we present simulation results for both regular and punctured PCs with corresponding upper bounds on the respective BLERs obtained by our implementation of DE. We note that these bounds are extremely tight and hence provide empirical evidence that our implementation is suitable to predict the BLERs of both punctured and regular PCs.

This paper is structured as follows. In Section II, we briefly discuss the preliminaries of both regular as well as punctured PCs to provide the background and notation used over the course of this work. Section III presents our implementation of a DE for the construction of both punctured and regular PCs. To do so, we present an in-depth discussion of the implicit assumptions such a DE rests on, and formalize the effects of the modifications implied by puncturing certain codeword positions on the convolution operations necessary for a DE. These steps facilitate drastic reductions of the total number of convolutions necessary for constructing such a code. Section IV discusses the complexity of our approach and bounds the number of convolutions. Furthermore, we show how the framework may be extended using Gaussian approximations (GAs) of the message densities. Numerical examples for two popular puncturing approaches, namely, quasi-uniform puncturing as well as uniform random puncturing, are provided in Section V. Furthermore, we present simulation results for both punctured and regular PCs alongside corresponding upper bounds on the respective BLERs, obtained by our DE implementation. The tightness of these bounds confirms the predictive power of our implementation, that is, resulting empirical BLERs are observed to closely match the performance bounds obtained by the proposed DE implementation. We conclude the paper in Section VI.

II. PRELIMINARIES

A. POLAR CODES

In his seminal work [1], Arıkan presents a phenomenon referred to as channel polarization (CP) and devises a channel coding scheme based on it. To do so, he shows how to combine $N = 2^n$, $n \in \mathbb{N}$, independent copies of a given BDMC $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$, with input alphabet $\mathcal{X} = \{0, 1\}$, and output alphabet $\mathcal{Y}$, into a set of N (virtual) channels $\{\mathcal{W}_{N,i} : i \in [N]\}$, where we write $[N] := \{1, \ldots, N\} \subset \mathbb{N}$. These channels

$$\mathcal{W}_{N,i} : \mathcal{X} \to \mathcal{Y}^N \times \mathcal{X}^{i-1}, \tag{2}$$

also referred to as coordinate channels, polarize in the sense that as N approaches infinity, the fraction of indices i for which $I(\mathcal{W}_{N,i})$ neither approaches 0 (useless channels) nor 1 (perfect channels) vanishes, where $I(\mathcal{W}_{N,i})$ denotes the symmetric capacity of channel $\mathcal{W}_{N,i}$, i.e. its mutual information assuming uniformly distributed channel inputs. In the limit, the fraction of indices i such that $I(\mathcal{W}_{N,i}) \to 1$ is arbitrarily close to $I(\mathcal{W})$ [1]. If $\mathcal{W}$ is output-symmetric, i.e. a binary-input memoryless output-symmetric channel (BMSC), the symmetric capacity $I(\mathcal{W})$ is equal to the Shannon capacity. As the proposed SC decoding algorithm exploits the polarization of the coordinate channels, PCs present a capacity-achieving coding scheme for the family of BMSCs with discrete output alphabets.

To induce CP given N independent copies of a basic channel $\mathcal{W}$ used for transmission, these copies are combined recursively. This recursion implies a linear code $\mathcal{C}$, which

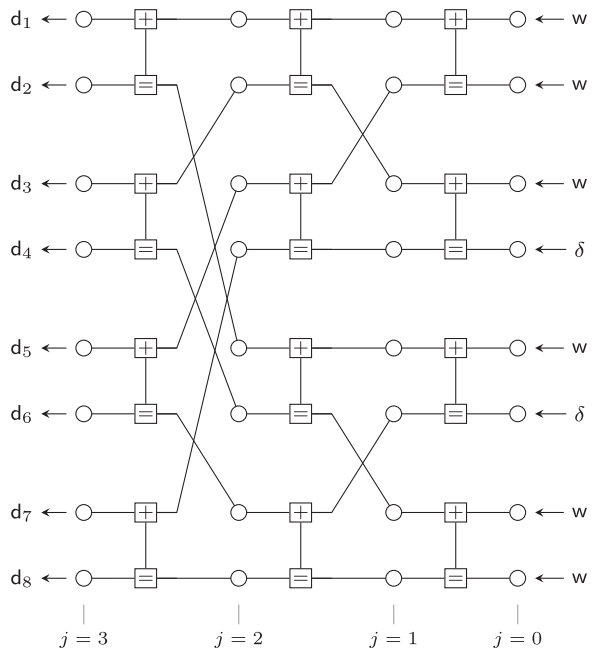

FIGURE 1. Factor graph of $\mathrm{B}_8\mathrm{F}^{\otimes 3}$, used here to exemplify construction of a punctured PC of length $M = 6$, by puncturing positions x_4 and x_6 of every codeword x of a mother PC of length $N = 8$. Consequently, these positions are initalized with an L-density δ, as punctured positions correspond to erasures for a DE.

forms codewords containing N linear combinations

$$\mathbf{x} = (x_1, \ldots, x_N) = \mathbf{u}_{\mathcal{A}}\mathbf{G}_{N,\mathcal{A}} \tag{3}$$

of $\mathbf{u} \in \mathbb{F}_2^N$. We follow the subvector notation used in [1] and let $\mathbf{u}_{\mathcal{A}} = (u_i)_{i\in\mathcal{A}}$ denote a subvector of $\mathbf{u}$ containing the information bits, where $\mathcal{A} \subset [N]$ provides the indices of these information positions. This implies a generator $\mathbf{G}_{N,\mathcal{A}}$ of the code formed by selecting rows $\mathbf{g}_i$, $i \in \mathcal{A}$ from the matrix $\mathbf{G}_N = \mathbf{B}_N\mathbf{F}^{\otimes n}$, where $\mathbf{B}_N$ is a bit-reversal permutation, and $\mathbf{F}$ as in (1) is the binary polarization kernel [1]. This sets the remaining elements $\mathbf{u}_{\mathcal{A}^{\mathrm{c}}}$ which are referred to as *frozen bits* to zero, where $\mathcal{A}^{\mathrm{c}} = [N] \setminus \mathcal{A}$ denotes the complement of $\mathcal{A}$ with respect to $[N]$. Selecting an information index set $\mathcal{A}$ of cardinality $|\mathcal{A}| = K$, we obtain a code of rate $R = \frac{K}{N}$.

By SC decoding, Arıkan suggests a sequential algorithm to decode PCs. Such a decoder exploits the structure of the FG representation of $\mathbf{G}_N$ given in Figure 1 to compute LLRs

$$L_{N,i}\left(\mathbf{y}_1^N, \hat{\mathbf{u}}_1^{i-1}\right) := \log \frac{W_{N,i}\left(\mathbf{y}_1^N, \hat{\mathbf{u}}_1^{i-1} \mid 0\right)}{W_{N,i}\left(\mathbf{y}_1^N, \hat{\mathbf{u}}_1^{i-1} \mid 1\right)} \tag{4}$$

for all information bits u_i, $i \in \mathcal{A}$, recursively based on the received channel output $\mathbf{y}_1^N$ and $i-1$ previous estimates $\hat{\mathbf{u}}_1^{i-1}$. These estimates are obtained by hard-decisions on the corresponding LLRs. Using random vectors $\left(U_1^N, Y_1^N\right)$ taking values $\left(\mathbf{u}_1^N, \mathbf{y}_1^N\right) \in \mathcal{X}^N \times \mathcal{Y}^N$ according to a distribution implied by the channel transition probabilities W and the prior of U_1^N, [1] gives an upper bound for the block error probability under SC decoding. To do so, [1] defines events

$$\begin{aligned}\mathsf{B}_i \;:=\; \Big\{\Big(\mathbf{u}_1^N, \mathbf{y}_1^N\Big) \in \mathcal{X}^N \times \mathcal{Y}^N :\\ \mathbf{u}_1^{i-1} = \hat{\mathbf{u}}_1^{i-1},\; u_i \neq h\Big(L_{N,i}\Big(\mathbf{y}_1^N, \hat{\mathbf{u}}_1^{i-1}\Big)\Big)\Big\},\end{aligned} \tag{5}$$

where $\left(\mathbf{u}_1^N, \mathbf{y}_1^N\right)$ are realizations of the random vectors $\left(U_1^N, Y_1^N\right)$, and h denotes the probabilistic hard-decision function acting on the LLR $L_{N,i}$, that is, h returns an equally likely random bit if $L_{N,i} = 0$. We have

$$\begin{aligned}\mathsf{B}_i &= \Big\{\Big(\mathbf{u}_1^N, \mathbf{y}_1^N\Big) \in \mathcal{X}^N \times \mathcal{Y}^N :\\ &\qquad \mathbf{u}_1^{i-1} = \hat{\mathbf{u}}_1^{i-1},\; u_i \neq h\Big(L_{N,i}\Big(\mathbf{y}_1^N, \mathbf{u}_1^{i-1}\Big)\Big)\Big\}\\ &\subseteq \Big\{\Big(\mathbf{u}_1^N, \mathbf{y}_1^N\Big) \in \mathcal{X}^N \times \mathcal{Y}^N :\\ &\qquad u_i \neq h\Big(L_{N,i}\Big(\mathbf{y}_1^N, \mathbf{u}_1^{i-1}\Big)\Big)\Big\} =: \mathsf{A}_i\,,\end{aligned} \tag{6}$$

which we give here relying on notation used in [12]. An upper bound on the block error probability under SC decoding is then given by

$$\mathbb{P}\big[\hat{\mathbf{u}} \neq \mathbf{u}\big] = \sum_{i\in\mathcal{A}} \mathbb{P}\big[\mathsf{B}_i\big] \;\leq\; \sum_{i\in\mathcal{A}} \mathbb{P}\big[\mathsf{A}_i\big]\,, \tag{7}$$

where the probabilities $\mathbb{P}\big[\mathsf{A}_i\big]$, $i \in [N]$, may be upper-bounded by the Bhattacharyya parameter of the corresponding coordinate channel $\mathcal{W}_{N,i}$, *cf.* [1].

B. PUNCTURED POLAR CODES

A punctured PC $\mathcal{C}'$ of length $M < N$ is defined by a puncturing pattern $\mathcal{P} \subset [N]$ such that $|\mathcal{P}| = N - M$, and then obtained from a mother PC $\mathcal{C}$ of length N by puncturing codeword positions indexed by $\mathcal{P}$ after encoding. It is an open problem to find the optimal puncturing pattern of a desired cardinality $|\mathcal{P}| = N - M$ [8], [15]. Consequently, several heuristics have been proposed.

Puncturing of PCs was first considered in [2]. In this work, the authors suggest puncturing positions connected to the fewest number of stopping trees in order to optimize performance of the punctured code under belief propagation (BP) decoding. Another straightforward heuristic for constructing $\mathcal{P}$ is given by choosing $N - M$ positions from $[N]$ uniformly at random, which we refer to as uniform random puncturing (URP) over the course of this work. This approach has been employed in [7], [8], and [9].

As an alternative, a deterministic method referred to as QUP is presented in [13]. To construct a QUP pattern $\mathcal{P}$ of cardinality $|\mathcal{P}| = N - M$, an incidence vector $\mathbf{p}$ of $\mathcal{P}$ is constructed as [13]

$$\mathbf{p} = (p_i)_{i\in[N]} = (\underbrace{1, \ldots, 1}_{\text{first } N-M \text{ positions}}, 0, \ldots, 0) \cdot \mathbf{B}_N\,. \tag{8}$$

Hence, we have

$$\mathcal{P} = \{i : p_i = 1\} \subset [N]\,. \tag{9}$$

III. EFFICIENT DENSITY EVOLUTION FOR PUNCTURED POLAR CODES

In this section, we will present an efficient implementation of DE which unifies the construction of both regular, i.e. unpunctured, as well as punctured PCs in a single algorithm. To do so, we describe DE to introduce concepts relevant to our work, and establish relevant notation. Other construction approaches such as Arıkan's explicit recursion in the Bhattacharyya parameters valid for binary erasure channels (BECs) or Monte Carlo (MC) constructions [1], as well as constructions by faithful estimations of the coordinate channels [16], [17], go beyond the scope of this work. We then inspect the adaptions required to allow for a DE-based construction of punctured PCs. Based on this, we present an algorithm reducing the number of convolution operations necessary to perform such construction. As these operations are the predominant source of cost in such implementations, the construction effort can be reduced drastically. Exact reductions depend on the given puncturing patterns, and are characterized in Sections IV and V. Numerical examples reported in Section V give examples of the savings obtained when puncturing with URP and QUP.

A. CONSTRUCTING POLAR CODES BY DENSITY EVOLUTION

Constructing a PC of a desired length N translates to selecting an index set $\mathcal{A} \subseteq [N]$, indicating the positions used for information bits, such that the upper bound on the BLER under SC decoding given in (7) is minimized. To facilitate construction of PCs for arbitrary BMSCs, in [12] Mori and Tanaka propose to use DE to evaluate the probabilities $\mathbb{P}[\mathsf{A}_i]$, i.e. the probability of decision error in the i-th position of $\hat{\mathbf{u}}$ under SC decoding, assuming previous positions given by $\hat{\mathbf{u}}_1^{i-1} = (u_1, \dots, u_{i-1})$ correctly supplied by a genie. The all-zero codeword $\mathbf{x} = \mathbf{0}$ is assumed to be transmitted, which does not impose a loss of generality, as the channel assumed is output-symmetric [14]. Let $\mathsf{w} \in \mathbb{L}$ denote the density of the channel output LLR assuming the input $X = 0$, where X is the random channel input. We write $\mathbb{L}$ to denote the space of L-densities, and refer to [14] for more details.

Inspecting the message schedule used in SC decoding, Mori and Tanaka note that SC decoding on the FG given in Figure 1 may be considered as an instance of BP decoding on a tree, whose leaves correspond to channel output LLRs. This structure hence allows for evaluating the message densities via DE. To see this, we inspect the message updates at a single processing element (PE) of the FG as given in Figure 2.

Initially, we have $\beta_1 = \beta_2 = 0$, which represent LLRs of either linear combinations of information bits to be estimated (b_1), or of future information bits (b_2), considered random for decoding b_1. Given LLRs α_1 and α_2, the decoder forwards the message $\alpha_1 \boxplus \alpha_2$. The box-plus operator $\boxplus : \overline{\mathbb{R}} \times \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ is given by

$$a, b \mapsto 2\operatorname{artanh}\left(\tanh\left(\frac{a}{2}\right)\tanh\left(\frac{b}{2}\right)\right), \tag{10}$$

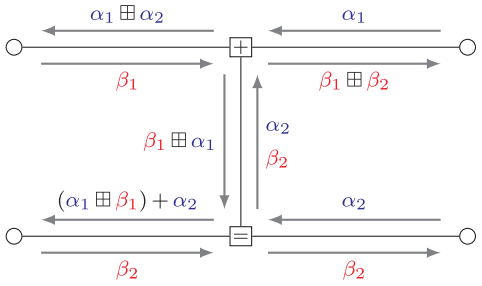

FIGURE 2. Message schedule of SC decoding at a processing element of the factor graph.

where $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ denotes the extended reals, while $\oplus$ denotes mod-2 addition over $\mathbb{F}_2$. Upon receiving $\beta_1 \in \{\pm\infty\}$, or equivalently, a bit $b_1 \in \mathbb{F}_2$, the SC decoder computes

$$\alpha_1 \boxplus \beta_1 + \alpha_2 = \alpha_1(1 - 2b_1) + \alpha_2 \tag{11}$$

based on which information bits linearly combined to form b_2 are decoded. Finally, $\beta_1 \boxplus \beta_2$ and β_2, or equivalently, corresponding hard-decisions $b_1 \oplus b_2$ and b_2 are propagated.

Thus, in the event A_i, i.e. decoding the i-th information position u_i assuming correct bits $\mathbf{u}_1^{i-1} = \mathbf{0}$ supplied by a genie, the corresponding β_1-messages take values fixed to $+\infty$. Furthermore, the β_2-messages corresponding to random future positions u_j, $j \in [N] \setminus [i]$, are initialized to zero. As a result, when assessing A_i, corresponding edges incident to the check and repetition nodes involved do not affect decoding of u_i, which is hence decoded on a tree.

The central idea of DE is to consider all messages exchanged on the FG as random variables (RVs). Assuming incoming messages to be independent, tracking the evolution of the densities under the operations is facilitated by convolutions $*$ and $\circledast$ of these densities, which replicate the effects of the operations $+$ and $\boxplus$ performed at repetition nodes and check nodes, respectively.

We inspect the PE again and write $\mathsf{a}_1, \mathsf{a}_2 \in \mathbb{L}$ to denote the densities of the LLRs α_1 and α_2, respectively. As all β messages are either equal to $+\infty$ or zero, message densities are thus evaluated by performing the convolutions

$$\mathsf{a}_1 \circledast \mathsf{a}_2 \quad \text{and} \quad \mathsf{a}_1 * \mathsf{a}_2, \tag{12}$$

to obtain the densities resulting from check node and repetition node operations, respectively. Hence, on the FG representation of a PC, given in Figure 1 for $N = 8$, a DE may be computed column-wise from right to left, starting at the channel level

$$\mathbf{w}_0 = (\mathsf{w}, \dots, \mathsf{w}) \in \mathbb{L}^N. \tag{13}$$

Let $\mathbf{w}_j = (\mathbf{w}_{i,j})_{i \in [2^{n-j}]}$, $\mathbf{w}_{i,j} \in \mathbb{L}^{2^j}$ denote the densities available after performing the convolutions at the PEs in the j-th column. Given $\mathbf{w}_{j-1}$, by the structure of the FG for each $i \in [2^{n-j}]$ we compute $\mathbf{w}_{i,j} \in \mathbb{L}^{2^j}$ by

$$\mathbf{w}_{i,j}[2k-1] = \mathbf{w}_{2i-1,j-1}[k] \circledast \mathbf{w}_{2i,j-1}[k], \tag{14}$$

$$\mathbf{w}_{i,j}[2k] = \mathbf{w}_{2i-1,j-1}[k] * \mathbf{w}_{2i,j-1}[k], \tag{15}$$

$k \in \left[2^{j-1}\right]$, to form $\mathbf{w}_j$ for each $j \in [n]$. We write $\mathbf{W} = [\mathbf{w}_j] \in \mathbb{L}^{N\times(n+1)}$ to denote the matrix containing all evaluated densities.

However, to construct an (unpunctured) PC using DE, the number of convolutions to perform may be reduced significantly. As all densities in $\mathbf{w}_0$ are equal to w, there are only 2^j different densities in each $\mathbf{w}_j$, $j \in [n]$. By the structure of the FG, these may be given by evaluating the recursion

$$\mathbf{v}_j[2k-1] = \mathbf{v}_{j-1}[k] \circledast \mathbf{v}_{j-1}[k], \tag{16}$$

$$\mathbf{v}_j[2k] = \mathbf{v}_{j-1}[k] * \mathbf{v}_{j-1}[k], \tag{17}$$

$k \in \left[2^{j-1}\right]$, $j \in [n]$, starting from $\mathbf{v}_0 = (\mathsf{w})$ [12]. Note that in this simplified formulation, we have $\mathbf{v}_j \in \mathbb{L}^{2^j}$, whereas in the general approach presented above, $\mathbf{w}_j \in \mathbb{L}^{2^n}$ for all $j \in \{0, \ldots, n\}$. Starting from $\mathbf{w}_0$ as in (13), we have $\mathbf{w}_n = \mathbf{v}_n$, that is, both approaches result in the same set of densities for the n-th column of the FG corresponding to the information bit level.

Given the densities $\mathbf{w}_n = (\mathsf{d}_i)_{i\in[N]}$, we consider the LLRs $L_{N,i}$ as in (4) as RVs given $\mathbf{u} = \mathbf{0}$ and $\hat{\mathbf{u}}_1^{i-1}$ correctly supplied by a genie. By construction, we have $L_{N,i} \sim \mathsf{d}_i$. Hence, the probabilities $\mathbb{P}\left[\mathsf{A}_i\right]$ are given by [12]

$$\begin{aligned}\mathbb{P}\left[\mathsf{A}_i\right] &= \mathbb{P}\left[L_{N,i} \leq 0\right] \\ &= \lim_{\epsilon\to 0}\left(\int_{-\infty}^{-\epsilon} \mathsf{d}_i(x)\,dx + \frac{1}{2}\int_{-\epsilon}^{+\epsilon} \mathsf{d}_i(x)\,dx\right)\end{aligned} \tag{18}$$

for all $i \in [N]$. As $\mathbb{P}\left[\mathsf{A}_i\right]$ only depends on d_i, we define the corresponding error probability of a hard-decision based on the LLR $L_{N,i} \sim \mathsf{d}_i$ as

$$P_e(\mathsf{d}_i) := \mathbb{P}\left[L_{N,i} \leq 0\right]. \tag{19}$$

B. DENSITY EVOLUTION FOR PUNCTURED POLAR CODES

To decode a punctured PC of length $M < N$ derived from a mother PC of length N using an SC decoder, a regular SC decoder for a PC of length N is used, treating punctured positions as erasures. To do so, channel LLRs corresponding to punctured positions y_i, $i \in \mathcal{P}$, are initialized with zero, *cf.*, *e.g.*, [7] or [13].

Thus, to evaluate the message densities for a punctured PC decoded by an SC decoder, two LLR densities have to be considered. In addition to the channel L-density w, we consider the density δ, modeling the erasures corresponding to punctured positions. As these LLRs take zero values with probability 1, we have

$$\delta(x) = \begin{cases} 1, & \text{if } x = 0, \\ 0, & \text{otherwise,} \end{cases} \tag{20}$$

i.e. the Dirac delta function with a single point mass located at $x = 0$.

In Figure 1, we give an example for the construction of a punctured PC of length $M = 6$, by puncturing codeword positions x_i for $i \in \{4, 6\}$ of a mother PC of length $N = 8$. Consequently, for the DE, we initialize $\mathbf{w}_0[4] = \mathbf{w}_0[6] = \delta$, while all other positions in $\mathbf{w}_0$ take the channel L-density w. After performing the DE, we obtain densities d_i, $i \in [8]$, based on which $\mathcal{A}$ may be selected as described in Section III-A.

To evaluate the densities in a column-wise implementation as given in (14) and (15), the number of convolutions is given by $N \log_2 N = n \cdot 2^n$, attained by evaluating every convolution. As convolutions are expensive for fine-grained quantizations necessary for the desired accuracy of the results [14], we strive for minimizing the number of convolutions at the expense of a minimum overhead.

Inspecting the column update equations, we note that the number of distinct convolution expressions to evaluate may be significantly smaller than the number of updates implied by a verbatim implementation of (14) and (15) for $j \in [n]$. To see this, we observe that the number of distinct expressions to evaluate when constructing $\mathbf{w}_j$ from $\mathbf{w}_{j-1}$ depends on both the number of distinct expressions in $\mathbf{w}_{j-1}$, as well as their arrangement within $\mathbf{w}_{j-1}$. As for $\mathbf{w}_0$ corresponding to the channel level, we only have two distinct input densities, namely, w and δ, the number of distinct expressions grows slowly in j.

To operationalize this approach, we analyze the effects of δ on the convolution operators $*$ and $\circledast$, implementing convolutions at repetition and check nodes, respectively. In a subsequent step, we inspect how the commutativity of the operators affects the number of convolutions. Both steps result in a drastic reduction of the number of convolutions necessary when performing a column update from $\mathbf{w}_{j-1}$ to $\mathbf{w}_j$. We conclude this section by presenting an algorithmic implementation which exploits both phenomena, implementing a DE for construction of punctured PCs at a massively reduced number of convolutions. Furthermore, the algorithm developed includes DE as in [12] as a special case, thus presenting a unified implementation to construct both regular as well as punctured PCs efficiently.

C. CONVOLVING DENSITIES WITH δ

Densities are invariant under repetition node convolution $*$ with δ, that is, for a given density $\mathsf{d} \in \mathbb{L}$ we have

$$\mathsf{d} * \delta = \mathsf{d}, \tag{21}$$

as δ represents the identity element of the binary operation $*$ on the space of L-densities [14].

Furthermore, the density δ is the zero element of the operation $\circledast$, i.e. for a given density d, we have

$$\mathsf{d} \circledast \delta = \delta. \tag{22}$$

Usually, in implementations of DE, as a first step of implementing the operation $\mathsf{a} \circledast \mathsf{b}$, the operand L-densities a and b are transformed into G-densities, defined on $\mathbb{F}_2 \times [0, +\infty]$ [14, p. 180 ff.]. However, to show (22), technicalities may be avoided by interpreting $\mathsf{a} \circledast \mathsf{b}$ as an L-density resulting from transforming both a and b into G-densities, performing the convolution $\circledast$ on the group $\mathbb{F}_2 \times [0, +\infty]$, and transforming the resulting G-density into an L-density.

As a result, the effect of convolving an L-density d with δ at a check node using the operation $\circledast$ may be interpreted probabilistically by inspecting the operation $\boxplus$ performed at a check node. For this, we assume random LLRs $P \sim \mathsf{d}$ and $D \sim \delta$, and inspect the random output of the check node

$$C = P \boxplus D. \tag{23}$$

As $\mathbb{P}[D = 0] = 1$, by the definition of $\boxplus$ given in (10), we obtain $\mathbb{P}[C = 0] = 1$, as we have $\operatorname{artanh}(0) = \tanh(0) = 0$. Hence, we conclude $C \sim \delta$.

As a result, both properties (21) and (22) of the convolution operators allow for immediate reductions of the number of convolutions to perform. Check node convolutions $\circledast$ with δ reproduce δ, while convolutions $\mathsf{p} * \delta$ performed at repetition nodes reproduce the operand density p. Hence, for either check or repetition nodes, no additional convolution needs to be performed in the presence of at least one operand equal to δ.

D. EXPLOITING COMMUTATIVITY OF THE CONVOLUTIONS

In addition to the reduction of convolutions implied by the effects of convolving δ by $*$ or $\circledast$ as given in Section III-C, exploiting the fact that both $*$ and $\circledast$ are commutative operations results in additional savings. Formally, for L-densities $\mathsf{a}, \mathsf{b} \in \mathbb{L}$, we have [14]

$$\mathsf{a} * \mathsf{b} = \mathsf{b} * \mathsf{a}, \text{ and}$$
$$\mathsf{a} \circledast \mathsf{b} = \mathsf{b} \circledast \mathsf{a}.$$

By induction, this extends to slices of length greater than one, computed according to the update equations (14) and (15). When performing the convolutions column-wise as in (14) and (15), we note that for each pair of two consecutive slices $\mathbf{w}_{2i-1,j}, \mathbf{w}_{2i,j} \in \mathbb{L}^{2^j}$ of $\mathbf{w}_j = (\mathbf{w}_{h,j})_{h \in [2^{n-j}]}$, where $i \in [2^{n-j-1}]$, the order in which these slices appear in such a pair does not change the resulting slice $\mathbf{w}_{i,j+1}$ of $\mathbf{w}_{j+1}$ [18]. This will allow for additional savings in the algorithm presented in the following.

E. ALGORITHMIC IMPLEMENTATION

In order to construct a punctured PC for a given puncturing pattern $\mathcal{P}$, we compute the probabilities of decision error under genie-aided SC decoding, given by $\mathbb{P}[\mathsf{A}_i]$, for $i \in [N]$. Algorithm 1 presents our approach, exploiting the simplifications given above.

We determine the necessary convolutions for a given length N and a puncturing pattern $\mathcal{P}$ by performing the column-wise DE symbolically, representing expressions

$$\mathsf{d}_i = \mathsf{d}_{i_1} \diamond \mathsf{d}_{i_2} \tag{24}$$

for $\diamond \in \{*, \circledast\}$ by tuples

$$\mathbf{e}_i = (i_1, \diamond, i_2, i) \in \mathbb{N} \times \{*, \circledast\} \times \mathbb{N} \times \{i\}, \tag{25}$$

identifying expressions with a unique index $i \in \mathbb{N}$.

To keep track of the symbolic convolution operations, we construct a matrix $\mathbf{D} \in \mathbb{N}^{N \times (n+1)}$, initializing its first column $\mathbf{d} = (d_i)_{i \in [N]}$ corresponding to channel level with the incidence vector of $\mathcal{P}^{\mathrm{c}} = [N] \setminus \mathcal{P}$, which is obtained by $\mathbb{I}_N : \mathscr{P}([N]) \to \{0, 1\}^N$ for arbitrary sets $\mathcal{S} \subseteq [N]$, where $\mathscr{P}([N])$ denotes the power set of $[N]$. Hence, we have $d_i = 1$ if $i \notin \mathcal{P}$, and $d_i = 0$, otherwise. Doing so, we define

$$\mathsf{d}_0 := \delta \text{ and } \mathsf{d}_1 := \mathsf{w}. \tag{26}$$

Note that we maintain our convention of one-based indexing for vectors and matrices here, thus $\mathbf{D}[:, 1]$ denotes the first column of $\mathbf{D}$ containing the indices of densities corresponding to the channel level. To accommodate for the density indexing scheme as in (26), the container $\mathfrak{D}$ holding densities obtained via convolutions is maintained using zero-based indexing, and we have $\mathfrak{D}[0] = \delta$ and $\mathfrak{D}[1] = \mathsf{w}$.

We then construct the remaining columns of $\mathbf{D}$ using the update rules given in (14) and (15). Whenever possible, we exploit the commutativity of slices as described above. As both operations are commutative, we require $i_1 \leq i_2$ and store tuples representing expressions in an associative container $\mathfrak{L}$ of lists indexed by i_1, such that

$$\mathfrak{L}[i_1] = \{(i_1, \diamond, i_2, i) : \diamond \in \{*, \circledast\},\ i_2, i \in \mathbb{N}\} \tag{27}$$

represents the list of tuples $\mathbf{e}_i$ having i_1 as its first operand, in order to facilitate cheap look-ups for existing expressions. These look-ups and, if necessary, new convolutions and corresponding insertions of expression tuples are performed by the method EXPR given in Algorithm 2. Finally, by applying (19)

Algorithm 1 Density Evolution for a Punctured PC

input: code length $N = 2^n$**, puncturing pattern** $\mathcal{P} \subset [N]$**, target channel LLR density** w.
output: probabilities $\left(\mathbb{P}[\mathsf{A}_i]\right)_{i \in [N]} \in [0, 1]^N$

function DENSITYEVOLUTION($n, \mathcal{P}, \mathsf{w}$)
  global $\mathfrak{L} \leftarrow \emptyset$ ▷ **associative container for expressions**
  global $\mathfrak{D} \leftarrow \{\delta, \mathsf{w}\}$ ▷ **densities,** $\mathfrak{D}[0] = \delta, \mathfrak{D}[1] = \mathsf{w}$
  $N \leftarrow 2^n \in \mathbb{N}$
  $\mathbf{d} \leftarrow \mathbb{I}_N(\mathcal{P}^{\mathbf{c}}) \in \mathbb{N}^N$
  $\mathbf{D} \leftarrow [\mathbf{d}\ \mathbf{0}] \in \mathbb{N}^{N \times (n+1)}$ ▷ **expression indices**
  for all $j \in [n]$ **do**
    $(\mathbf{a}_i)_{i \in [2^{n-j}]}^{\mathsf{T}} \leftarrow \mathbf{0}$ ▷ $\mathbf{a}_i \in \mathbb{N}^{2^j}$
    $(\mathbf{d}_i)_{i \in [2^{n-j+1}]}^{\mathsf{T}} \leftarrow \mathbf{D}[:, j]$ ▷ $\mathbf{d}_i \in \mathbb{N}^{2^{j-1}}$
    for all $i \in [2^{n-j}]$ **do**
      if for some $l \in [i-1]$
        $(\mathbf{d}_{2i-1}, \mathbf{d}_{2i}) = (\mathbf{d}_{2l-1}, \mathbf{d}_{2l})$ **or**
        $(\mathbf{d}_{2i}, \mathbf{d}_{2i-1}) = (\mathbf{d}_{2l-1}, \mathbf{d}_{2l})$ **then**
          $\mathbf{a}_i \leftarrow \mathbf{a}_l$
      else
        for all $k \in [2^{j-1}]$ **do**
          $\mathbf{a}_i[2k-1] \leftarrow$ **EXPR**$(\mathbf{d}_{2i-1}[k], \circledast, \mathbf{d}_{2i}[k])$
          $\mathbf{a}_i[2k] \leftarrow$ **EXPR**$(\mathbf{d}_{2i-1}[k], *, \mathbf{d}_{2i}[k])$
    $\mathbf{D}[:, j+1] \leftarrow (\mathbf{a}_i)_{i \in [2^{n-j}]}^{\mathsf{T}}$
  return $\left(P_e(\mathfrak{D}[d_{i,n+1}])\right)_{i \in [N]}$ ▷ $\mathbf{D} = [d_{i,j}] \in \mathbb{N}^{N \times n+1}$

Algorithm 2 Evaluate an Expression

input: operand indices $i_1, i_2 \in \mathbb{N}$, **operator** $\diamond \in \{*, \circledast\}$
output: density index $d \in \mathbb{N}$
function EXPR$(i_1, \diamond, i_2)$
  if $i_1 > i_2$ **then**
    $j \leftarrow i_2, i_2 \leftarrow i_1, i_1 \leftarrow j$
  if $i_1 = 0$ **then**
    if $\diamond = *$ **then**
      return i_2
    else
      return 0
  if $(i_1, \diamond, i_2, j) \notin \mathfrak{L}[i_1]$ **for some** $j \in \mathbb{N}$ **then**
    $i \leftarrow |\mathfrak{D}|$ ▷ $\mathfrak{D}$ **uses zero-based indexing**
    $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{\mathfrak{D}[i_1] \diamond \mathfrak{D}[i_2]\}$
    $\mathfrak{L}[i_1] \leftarrow \mathfrak{L}[i_1] \cup \{(i_1, \diamond, i_2, i)\}$
    return i ▷ **index of new expression**
  else
    return j ▷ **index of existing expression**

to the densities indexed by column $\mathbf{D}[:, n+1]$, we obtain the desired probabilities.

IV. CONSTRUCTION COMPLEXITY

The number of convolutions necessary to construct a regular, i.e. unpunctured, PC of length $N = 2^n$ is given by

$$C_{\text{PC}} = \sum_{i=1}^{n} 2^i - 1 = 2^{n+1} - 2 = 2N - 2, \tag{28}$$

such that an equal number of repetition node convolutions $*$ and check node convolutions $\circledast$ is performed. To see this, we note that the update rules given in (16) and (17) construct a binary tree of convolutions. On the other hand, for a punctured PC, a trivial upper bound is given by

$$C_{\text{PPC}} \leq N \log_2 N = n \cdot 2^n, \tag{29}$$

attained by a verbatim implementation of the column-wise update equations (14) and (15).

While the exact number of convolutions necessary to construct a punctured PC using the approach presented above depends on the choice of $\mathcal{P}$, we give an upper bound for C_{PPC} depending on the cardinality of $\mathcal{P}$, that is, the number of punctured positions. Therefore, we bound the number of potential density expressions, outline a strategy for tightening this bound, and exploit structural properties of the FG to provide an upper bound on the number of necessary convolutions for constructing a punctured PC.

A. BOUNDING THE NUMBER OF EXPRESSIONS

The following theorem provides an upper bound on C_{PPC} for a mother code length of $N = 2^n$, obtained by a characterization of the set of possible density expressions after performing the column-wise density updates of n columns reflecting the corresponding FG.

Theorem 1: The number of convolutions $C_{PPC} \in \mathbb{N}$ necessary to construct a punctured PC of length $N = 2^n$ via DE is upper-bounded by

$$C_{PPC} \leq 2 + \sum_{i=2}^{n} |\mathcal{D}_i| \left(2 \sum_{k=0}^{i-2} |\mathcal{D}_k| + |\mathcal{D}_{i-1}| - 1 \right),$$

where $\mathcal{D}_j$ is the set of densities newly obtained by the convolutions corresponding to the PEs in the j-th column of the FG.

Proof: To prove the bound, we define $\mathcal{D}_j$ as the set of densities that become *newly* available as operands for future convolutions after performing the convolutions in the j-th column of PEs in the FG when using the approach presented in Section III-B. By initialization, we have $\mathcal{D}_0 = \{\delta, \mathsf{w}\}$, where w denotes the channel L-density.

The set of *all* densities available after processing the convolutions of the first j columns is then given by

$$\bar{\mathcal{D}}_j = \bigcup_{i=0}^{j} \mathcal{D}_i \,. \tag{30}$$

Hence, the number of convolutions necessary to construct a punctured PC for a given length $N = 2^n$ and a puncturing pattern $\mathcal{P}$ is upper-bounded by $|\bar{\mathcal{D}}_n| - |\mathcal{D}_0| = |\bar{\mathcal{D}}_n| - 2$, as $\mathcal{D}_0$ is given. To assess the cardinality of $\bar{\mathcal{D}}_n$, we note that

$$\mathcal{D}_{j+1} \subseteq \left\{ \mathsf{u} \diamond \mathsf{v} \in \bar{\mathcal{D}}_j \times \{*, \circledast\} \times \bar{\mathcal{D}}_j \right\}, \tag{31}$$

and

$$\mathcal{D}_1 = \{\mathsf{w} \circledast \mathsf{w}, \mathsf{w} * \mathsf{w}\}. \tag{32}$$

Given $\bar{\mathcal{D}}_j$, we construct $\mathcal{D}_{j+1}$ for $j \geq 1$ accounting for the commutativity of the convolutions, and obtain

$$\begin{aligned} \mathcal{D}_{j+1} = \bigcup_{\diamond \in \{*, \circledast\}} \big(& \left\{ \mathsf{u} \diamond \mathsf{u} : \mathsf{u} \in \mathcal{D}_j \right\} \cup \\ & \left\{ \mathsf{u} \diamond \mathsf{v} : (\mathsf{u}, \mathsf{v}) \in \mathcal{D}_j^2, \mathsf{u} \neq \mathsf{v} \right\} \cup \\ & \left\{ \mathsf{u} \diamond \mathsf{v} : \mathsf{u} \in \mathcal{D}_j, \mathsf{v} \in \mathcal{D}_i, i \in [j-1] \right\} \cup \\ & \left\{ \mathsf{u} \diamond \mathsf{v} : \mathsf{u} \in \mathcal{D}_j, \mathsf{v} = \mathsf{w} \right\} \big), \end{aligned} \tag{33}$$

and note that $[k] = \emptyset$ if $k = 0$. Thus, we have

$$\begin{aligned} |\mathcal{D}_{j+1}| &= 2 \left(|\mathcal{D}_j| + \binom{|\mathcal{D}_j|}{2} + |\mathcal{D}_j| \cdot |\bar{\mathcal{D}}_{j-1} \setminus \mathcal{D}_0| + |\mathcal{D}_j| \right) \\ &= 2 \left(2|\mathcal{D}_j| + \frac{|\mathcal{D}_j|(|\mathcal{D}_j| - 1)}{2} + |\mathcal{D}_j| \left(|\bar{\mathcal{D}}_{j-1}| - 2 \right) \right) \\ &= |\mathcal{D}_j| \left(2|\bar{\mathcal{D}}_{j-1}| + |\mathcal{D}_j| - 1 \right), \end{aligned} \tag{34}$$

where

$$|\bar{\mathcal{D}}_j| = \sum_{i=0}^{j} |\mathcal{D}_i| \tag{35}$$

by (30). Noting that $|\mathcal{D}_1| = 2$, and by combining (34) and (35), the claim follows. □

B. ACCOUNTING FOR THE STRUCTURE OF THE FACTOR GRAPH

The bound given by Theorem 1 may be evaluated explicitly for any $n > 1$ by determining the cardinalities of the sets $\mathcal{D}_j$ and $\bar{\mathcal{D}}_j$ successively for $j \in \{2, \dots, n\}$. However, we note that this bound may be tightened substantially. To see this, we build on the applicability of the commutativity of the operations $*$ and $\circledast$ as described in Section III-D. Similar to the proof of Theorem 1, we inspect the number of possible convolutions, but consider the structure of the FG and the commutativity of slices of densities.

To that end, we define sets $\mathcal{U}_j$ and $\mathcal{V}_j$ for $j \in [n]$ which contain all possible slices $\mathbf{u}, \mathbf{v} \in \mathbb{L}^{2^j}$ available before and after performing the convolutions at the j-th column of PEs. That is, the j-th column of densities $\mathbf{w}_j$ as in Section III-A may be composed of combinations of $\mathbf{v} \in \mathcal{V}_j$. These result from convolutions of $\mathbf{u} = (\mathbf{v}_1, \mathbf{v}_2) \in \mathcal{U}_j \subseteq \mathcal{V}_{j-1} \times \mathcal{V}_{j-1}$ according to (14) and (15). As both $\mathbf{u}$ as well as $\mathbf{u}' = (\mathbf{v}_2, \mathbf{v}_1)$ result in the same $\mathbf{v}$, only $\mathbf{u}$ is taken into account for construction of $\mathcal{U}_j$. In addition to that, we write $\mathcal{W}_j \subseteq \mathbb{L}$ to denote the set of densities that become newly available after performing the convolutions at the j-th column of PEs. For a punctured PC, we have

$$\mathcal{V}_0 = \mathcal{W}_0 = \{\mathsf{d}_0 = \delta, \mathsf{d}_1 = \mathsf{w}\}, \tag{36}$$

where w denotes the L-density of the channel considered.

Similar to the proof of Theorem 1, the total number of convolutions necessary to construct a punctured PC of block length $N = 2^n$ is upper-bounded by

$$|\bar{\mathcal{W}}_n| - 2 = \sum_{j=0}^{n} |\mathcal{W}_j| - |\mathcal{W}_0|, \tag{37}$$

where $\bar{\mathcal{W}}_n = \bigcup_{j=0}^{n} \mathcal{W}_j$ denotes the set of all densities potentially available after performing the convolutions for constructing a punctured PC of length $N = 2^n$.

As an example, we give the sets for $j \in \{1, 2\}$ in Table 1, given the initial sets $\mathcal{V}_0 = \mathcal{W}_0$ as in (36). We note that

$$|\bar{\mathcal{W}}_2| - 2 = 8 < 10 = |\bar{\mathcal{D}}_2| - 2, \tag{38}$$

where we use $\bar{\mathcal{D}}_2$ as in (30). Hence, by constructing the sets $\mathcal{W}_j, j \in [n]$ as described, the bound provided by Theorem 1 may be tightened.

However, as both approaches characterize upper bounds on C_{PPC} via bounding the cardinality of the set of density expressions available after a certain number of column updates, it is non-trivial to establish a dependency on the given puncturing pattern. To provide an explicit bound on C_{PPC} depending on $\mathcal{P}$, we establish the following properties of our construction with respect to the FG structure exemplified for a length of $N = 8$ in Figure 1. We point out that [7] presents the fact that exactly P information positions become incapable, and we hence have exactly P corresponding LLRs in the leftmost column of the FG representation of $\mathbf{G}_N$. Furthermore, the following lemma in this form has been obtained in [18]. We state it for completeness and transfer its statement

TABLE 1. Possible Combinations of Densities at the j-th Column of Processing Elements.

j	$\mathcal{U}_j \subseteq \mathcal{V}_{j-1} \times \mathcal{V}_{j-1}$	$\mathcal{V}_j$	$\mathcal{W}_j$
1	$\mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_1\ \mathsf{d}_0\ \mathsf{d}_1\ \mathsf{d}_1$	$\mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_2\ \mathsf{d}_0\ \mathsf{d}_1\ \mathsf{d}_3$	$\mathsf{d}_2 = \mathsf{d}_1 \circledast \mathsf{d}_1$ $\mathsf{d}_3 = \mathsf{d}_1 * \mathsf{d}_1$
2	$\mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_2$ $\mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_1\ \mathsf{d}_0\ \mathsf{d}_1\ \mathsf{d}_3$ $\mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_2\ \mathsf{d}_2\ \mathsf{d}_2$ $\mathsf{d}_0\ \mathsf{d}_1\ \mathsf{d}_1\ \mathsf{d}_3\ \mathsf{d}_3\ \mathsf{d}_3$	$\mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_6$ $\mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_2\ \mathsf{d}_2\ \mathsf{d}_7$ $\mathsf{d}_0\ \mathsf{d}_0\ \mathsf{d}_2\ \mathsf{d}_0\ \mathsf{d}_4\ \mathsf{d}_8$ $\mathsf{d}_0\ \mathsf{d}_1\ \mathsf{d}_3\ \mathsf{d}_3\ \mathsf{d}_5\ \mathsf{d}_9$	$\mathsf{d}_4 = \mathsf{d}_1 \circledast \mathsf{d}_3$ $\mathsf{d}_5 = \mathsf{d}_1 * \mathsf{d}_3$ $\mathsf{d}_6 = \mathsf{d}_2 \circledast \mathsf{d}_2$ $\mathsf{d}_7 = \mathsf{d}_2 * \mathsf{d}_2$ $\mathsf{d}_8 = \mathsf{d}_3 \circledast \mathsf{d}_3$ $\mathsf{d}_9 = \mathsf{d}_3 * \mathsf{d}_3$

in order to highlight the aspects relevant to our work, and omit the proof.

Lemma 2: [18] For any puncturing pattern $\mathcal{P} \subset [N]$ given to construct a punctured PC of length $M = N - |\mathcal{P}|$ from a mother PC $\mathcal{C}$ of length $N = 2^n$, $n \in \mathbb{N}$, the number of densities equal to δ encountered in every column $\mathbf{w}_j \in \mathbb{L}^N$, $j \in \{0, \dots, n\}$, of a corresponding DE performed as in Algorithm 1, is constant and equal to $|\mathcal{P}|$.

Using this lemma, we obtain the following result characterizing the number of both variable and check node convolutions performed at a DE-based construction of a punctured PC.

Lemma 3: For any puncturing pattern $\mathcal{P} \subset [N]$ given to construct a punctured PC $\mathcal{C}'$ of length $M = N - |\mathcal{P}|$ from a mother PC $\mathcal{C}$ of length $N = 2^n$, $n \in \mathbb{N}$, we have

$$C_{*,PPC} = C_{\circledast,PPC},$$

where $C_{*,PPC}$ and $C_{\circledast,PPC}$ denote the number of non-trivial repetition and check node convolutions, respectively, when constructing $\mathcal{C}'$ using the implementation of DE presented above.

Proof: We inspect the process of obtaining $\mathbf{w}_j$ from $\mathbf{w}_{j-1}$ by the update rule (14) and (15), given $\mathbf{w}_0$, holding the L-densities corresponding to channel level. Let an update using densities a and b in $\mathbf{w}_{j-1}$ result in densities

$$\mathsf{c} = \mathsf{a} \circledast \mathsf{b}, \tag{39}$$

$$\mathsf{d} = \mathsf{a} * \mathsf{b}, \tag{40}$$

for inclusion in $\mathbf{w}_j$. By these rules and the principles exploited in the algorithm presented above, for $\mathsf{a}, \mathsf{b} \neq \delta$, i.e., non-trivial convolutions, either both c and d have been computed already, or neither of them. As this holds for any column update, the total number of non-trivial convolutions is even, and we have $C_{*,\text{PPC}} = C_{\circledast,\text{PPC}}$. □

Using these facts, we may establish the following theorem, which provides an explicit upper bound on C_{PPC} depending on the cardinality of the puncturing pattern $\mathcal{P}$.

Theorem 4: For any puncturing pattern $\mathcal{P} \subset [N]$ given to construct a punctured PC $\mathcal{C}'$ of length $M = N - |\mathcal{P}|$ from a mother PC $\mathcal{C}$ of length $N = 2^n$, $n \in \mathbb{N}$, the total number C_{PPC} of convolutions necessary to construct $\mathcal{C}'$ via

DE is upper-bounded by

$$C_{PPC} \leq n \cdot \left(N - 2 \left\lceil \frac{|\mathcal{P}|}{2} \right\rceil \right).$$

Proof: As in the proof of Lemma 3, we inspect a single pair of updates on a and b in $\mathbf{w}_{j-1}$, resulting in densities c and d for inclusion in $\mathbf{w}_j$, *cf.* (39) and (40). If either one of the densities a and b is equal to δ, or both of them are equal to δ, the number of convolutions $C_{*,\mathrm{PPC}}$ and $C_{\circledast,\mathrm{PPC}}$ each will decrease by one.

By Lemma 3, the number of densities equal to δ in each column $\mathbf{w}_j$ is equal to $|\mathcal{P}|$. As the worst case, in a given column all of them align in the sense that they result in updates as in (39) and (40) such that $\mathsf{a} = \mathsf{b} = \delta$. Hence, at maximum, for construction of the j-th column $\mathbf{w}_j$, we have to perform $N - 2\left\lceil \frac{|\mathcal{P}|}{2} \right\rceil$ convolutions in total. This applies to every $j \in [n]$, yielding the theorem. □

We note that Theorem 4 is generic in the sense that it does not take into account the structure of the underlying puncturing pattern. We expect that doing so would allow for refined bounds. However, as our focus is on implementing a DE-based construction efficiently, we consider the bound provided by Theorem 4 as a baseline to compare the savings in terms of the number of convolutions obtained by our implementation. In fact, the bound given in Theorem 4 presents the worst-case number of convolutions, that is, the necessary number of convolutions without applying any of the modifications proposed in this work, but only exploiting the properties of the convolutions with δ as an operand, given in (21) and (22).

C. IMPLEMENTATIONAL ASPECTS

As a meta algorithm, the approach presented above relies on implementations of the convolution operations. While the repetition or variable node convolution $*$ may be implemented by a regular convolution, easily accelerated by fast Fourier transforms (FFTs) performing calculations in the frequency domain, implementing the check node convolution $\circledast$ is more involved.

A discretized version of $\circledast$ is presented in [19], which may be approximated by precomputing quantization mappings to form a lookup table [14], which we employ to obtain the numerical results reported in the following. Another approach, inspired by an FFT, is given in [20].

In the following, we will sketch the extension of the presented framework to Gaussian approximations of the densities.

D. GAUSSIAN APPROXIMATION

To construct a regular PC, Trifonov [5], [21] suggests approximating the recursive DE as given in (16) and (17) using GAs of the message densities as in [22]. Consequently, these update equations turn into a recursion tracking the means of the message densities, and are given as

$$\mathbb{E}\big[V_j[2k-1]\big] = \phi^{-1}\Big(1 - \big(1 - \phi\big(\mathbb{E}\big[V_{j-1}[k]\big]\big)\big)^2\Big), \quad (41)$$

Algorithm 3 Evaluate Expressions for GA

input: expectations $\mu_1, \mu_2 \in \mathbb{R}$**, operator** $\diamond \in \{*, \circledast\}$
output: expectation $\mu \in \mathbb{N}$
function EXPRGA(μ_1, $\diamond$, μ_2)
  if $\diamond = \circledast$ **then**
    if $\mu_1 = 0$ **or** $\mu_2 = 0$ **then**
      return 0
    else
      return $\phi^{-1}(1 - (1 - \phi(\mu_1))\,(1 - \phi(\mu_2)))$
  else
    return $\mu_1 + \mu_2$

$$\mathbb{E}\big[V_j[2k]\big] = 2\mathbb{E}\big[V_{j-1}[k]\big], \quad (42)$$

for $j \in [n]$, given in notation analogous to (16) and (17), using the functions ϕ and ϕ^{-1} as defined in [22].

Assuming an additive white Gaussian noise (AWGN) channel with one-sided power spectral density N_0 used with binary phase-shift keying (BPSK) modulation, the Gaussian assumption is satisfied, as the corresponding channel L-density is Gaussian and given by [14]

$$L \sim N\left(\frac{2}{\sigma^2}, \frac{4}{\sigma^2}\right), \quad (43)$$

where $\sigma^2 = \frac{N_0}{2}$ denotes the corresponding noise power. We may thus start the recursion using

$$\mathbb{E}[V_0] = \frac{2}{\sigma^2}. \quad (44)$$

The error probabilities of the virtual channels as given in (19) are then approximated by computing

$$P_e(\mathsf{v}_i) \approx Q\Big(\sqrt{\mathbb{E}[V_n[i]]}\Big), \quad (45)$$

assuming

$$\mathsf{v}_i \approx N\left(\mathbb{E}[V_n[i]], 2\mathbb{E}[V_n[i]]\right), \quad (46)$$

that is, we assume that the L-density v_i is well approximated by a Gaussian density, and hence sufficiently characterized by $\mathbb{E}[V_n[i]]$.

In a similar fashion, DE-based constructions of punctured PCs may be translated to a GA. Using a GA to construct punctured PCs has been considered in [18], interpreting channel output LLRs at punctured positions as Gaussian RVs with zero mean and zero variance. Clearly, random LLRs $D \sim \delta$ corresponding to punctured positions have expectation $\mathbb{E}[D] = 0$, and so have random LLRs produced at check nodes, if at least one operand message has expectation equal to 0. The approach presented above in Algorithm 1 may be consequently adapted by replacing the matrix $\mathbf{D} \in \mathbb{N}^{N\times(n+1)}$ with a matrix $\mathbf{E} \in \mathbb{R}^{N\times(n+1)}$ that keeps track of the expectations corresponding to the respective densities. Initializing the first column of $\mathbf{E} = \mathbf{0}$ with means $\frac{2}{\sigma^2} \cdot \mathbb{I}_N(\mathcal{P}^c)$, corresponding to positions punctured according to a pattern $\mathcal{P}$, density expressions do not have to be evaluated symbolically, but corresponding means may be tracked immediately, which is

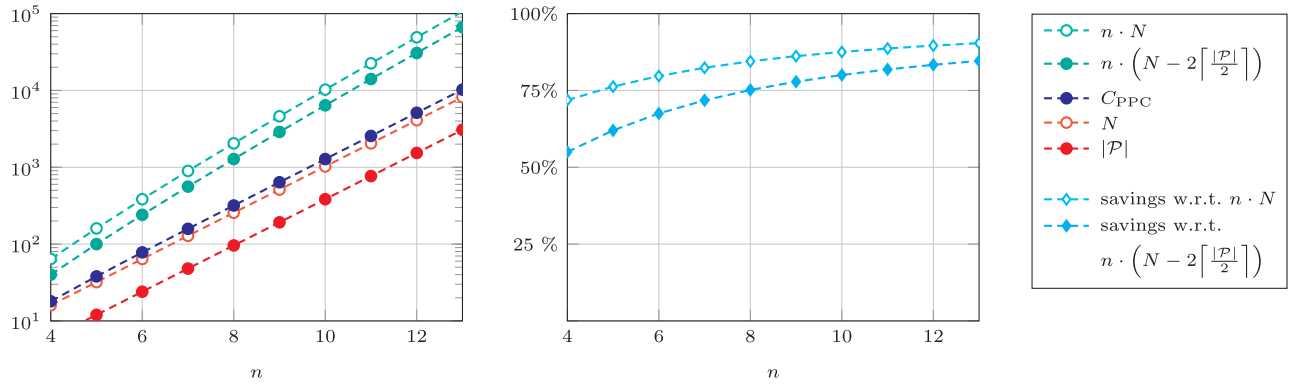

FIGURE 3. *Left-hand side:* **Bounds and actual number of convolutions necessary to construct quasi-uniformly punctured PCs of lengths $M = \frac{5}{8}N$, where $N = 2^n$, $n \in \{4, \ldots, 13\}$, is the length of the mother code.** *Right-hand side:* **Savings in terms of necessary convolutions achieved by the presented approach with respect to the bounds given.**

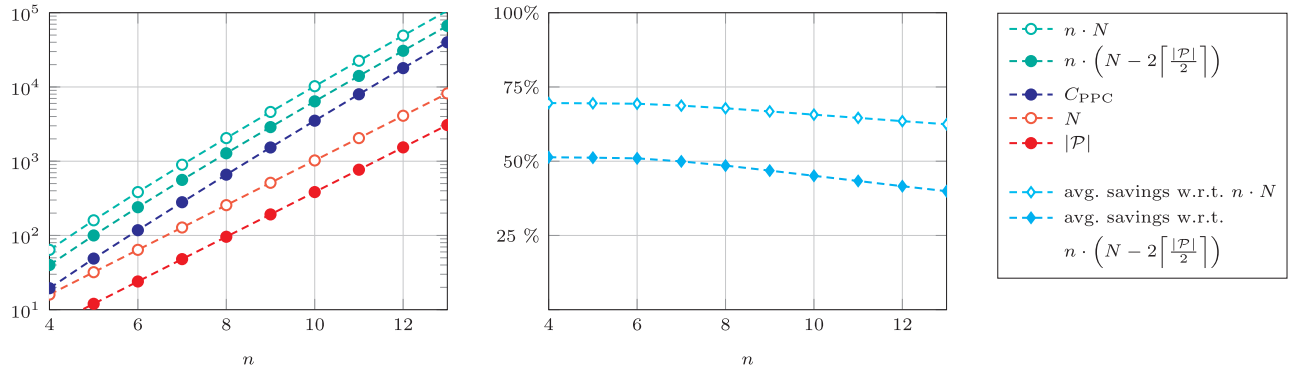

FIGURE 4. *Left-hand side:* **Bounds and actual number of convolutions necessary to construct uniformly random punctured PCs of lengths $M = \frac{5}{8}N$, where $N = 2^n$, $n \in \{4, \ldots, 13\}$, is the length of the mother code.** *Right-hand side:* **Savings in terms of necessary convolutions achieved by the presented approach with respect to the bounds given.**

implemented by replacing the method EXPR with EXPRGA as given in Algorithm 3. Finally, approximations of the desired error probabilities are evaluated as in (45) based on the expectations in the last column of $\mathbf{E} = [E_{i,j}]$, and obtained for each $i \in [N]$ as

$$\mathbb{P}[\mathsf{A}_i] \approx Q\left(\sqrt{\mathbb{E}[E_{i,n+1}]}\right). \tag{47}$$

V. NUMERICAL EXAMPLES

In this section, we provide numerical examples for construction of punctured PCs to exemplify the computational savings of the presented implementation of DE. We do so for two popular puncturing approaches, QUP and URP, described in Section II-B.

For both puncturing schemes, we use the DE approach presented above to construct punctured PCs from a mother PC of dimension $K = \frac{1}{2}N$, i.e. rate $R = \frac{1}{2}$. Using patterns of length

$$|\mathcal{P}| \in \left\{\frac{1}{4}N, \frac{3}{8}N\right\}, \tag{48}$$

we obtain punctured codes of lengths

$$M = N - |\mathcal{P}| \in \left\{\frac{3}{4}N, \frac{5}{8}N\right\} \tag{49}$$

and effective rates

$$\tilde{R} = \frac{K}{M} \in \left\{\frac{2}{3}, \frac{4}{5}\right\}. \tag{50}$$

Results are given in Figures 3 and 4, while in Tables 2 and 3, we provide additional details for certain values of N. In Figure 3, we provide the results for punctured PCs of length $M = \frac{5}{8}N$, obtained by QUP of a mother PC of length $N = 2^n$ for $n \in \{4, \ldots, 13\}$. In particular, we give bounds as in (29), given as (-○-), and Theorem 4 (-●-), as well as the length N (-○-), the cardinality $|\mathcal{P}|$ (-●-), and C_{PPC} (-●-), the number of convolutions necessary to construct the codes using the approach presented in this work. Due to the high regularity of the puncturing pattern obtained by QUP, C_{PPC} is much lower than $n \cdot N$. The right-hand plot in Figure 3 confirms this observation by reporting savings in terms of necessary convolutions with respect to $n \cdot N$ well above 75 % for most values of n, given as (-◆-). In fact, when constructing a code obtained as described from a mother code of length $N = 2^{13}$, by the approach presented above

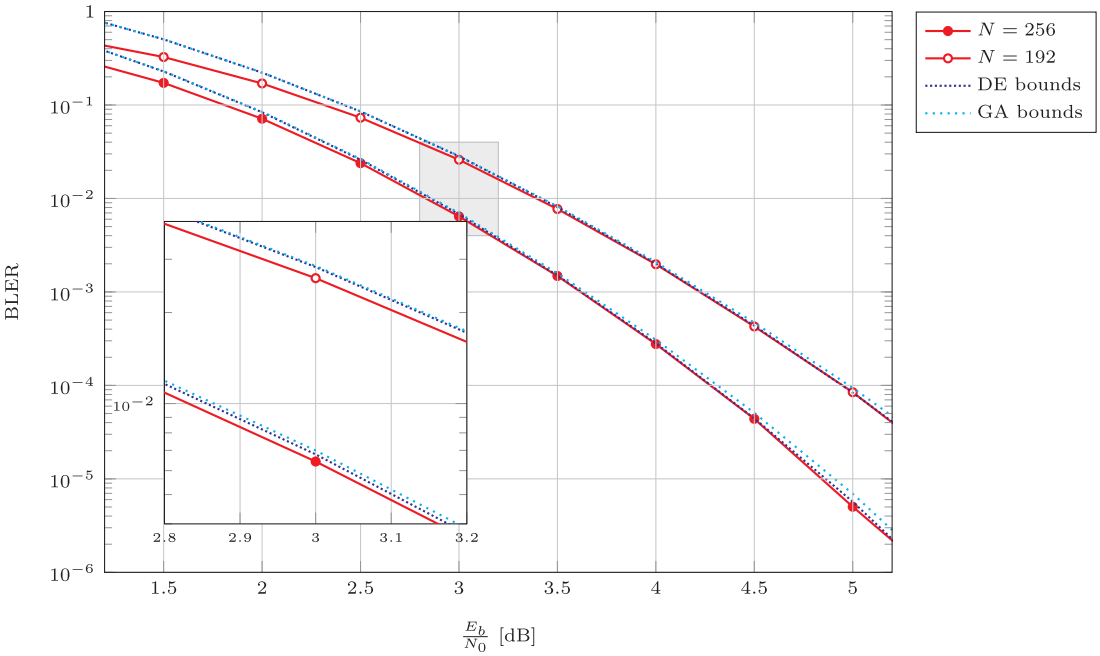

FIGURE 5. Simulation results for a regular PC of length 256, rate $\frac{3}{8}$, and a quasi-uniformly punctured PC of length 192, rate $\frac{1}{2}$, both under SC decoding. Dashed lines give corresponding upper bounds on the BLERs, obtained via DE and GA by the implementation presented in this work.

TABLE 2. Numerical Examples for QUP.

Parameter	$M = \frac{3}{4}N$	$M = \frac{5}{8}N$
length of mother code N	1024	8192
length of punctured code M	768	5120
# of convolutions C_{PPC}	1534	10 238
savings w.r.t. $n \cdot N$	85.02 %	90.39 %
savings w.r.t. $n \cdot \left(N - 2\left\lceil \frac{\lvert\mathcal{P}\rvert}{2} \right\rceil\right)$	80.03 %	84.62 %

TABLE 3. Numerical Examples for URP.

Parameter	$M = \frac{3}{4}N$	$M = \frac{5}{8}N$
length of mother code N	512	512
length of punctured code M	384	320
average # of convolutions C_{PPC}	1886.05	1531.17
avg. savings w.r.t. $n \cdot N$	59.07 %	66.77 %
avg. savings w.r.t. $n \cdot \left(N - 2\left\lceil \frac{\lvert\mathcal{P}\rvert}{2} \right\rceil\right)$	45.43 %	46.83 %

we may save 90.39 % of the convolutions compared to an implementation performing $n \cdot N$ convolutions. Additional numerical values are provided in Table 2.

In a similar fashion, Figure 4 and Table 3 give results for constructing punctured PCs via URP, averaging over 10^3 draws of $\mathcal{P}$ for each N. While the general trends agree with those identified for QUP, the necessary number of convolutions is considerably higher, which may be attributed to the puncturing patterns used here, which, on average, are less structured than those obtained by QUP. Nevertheless, savings of more than 35 % are obtained with respect to either bound.

As a remark, we point out that regardless of the target rate, the savings in terms of necessary convolutions depend only on the length of the mother code N, the target length of the punctured code M, as well as on the structure of the puncturing pattern $\mathcal{P}$. Given parameters N and M, the rate of the punctured code only depends on the dimension of the mother code $K \in \mathbb{N}$, which may take any value such that $1 \leq K < M$ to obtain reasonable rates.

Finally, in Figure 5, we provide simulation results and corresponding bounds obtained by the DE implementation presented in this work. We present BLERs under SC decoding of an unpunctured mother PC of length 256, as well as a quasi-uniformly punctured PC of length 192. Both codes are of dimension 96, and employ index sets selected based on the respective DEs. Hence, the first code is of rate $\frac{3}{8}$, while the latter is of rate $\frac{1}{2}$. For transmission, we assume an AWGN channel with one-sided power spectral density N_0 used with BPSK modulation, and hence a channel L-density given by (43). For construction, both codes target such a channel at $\frac{E_b}{N_0} = 1.5\,\text{dB}$. For construction as well as evaluation of the bounds, all densities are sampled on the range $[-20, 20]$ with equidistant sampling points to obtain a resolution of 0.01.

As evident from the figure, the bounds obtained are extremely tight for both the unpunctured as well as the punctured PC. We hence provide empirical evidence that for both classes of PCs, a DE with the parameters given above is a precise yet efficient tool for predicting code performance in terms of BLER under SC decoding. In addition to the

bounds obtained via DE, we present the bounds resulting from corresponding GAs of the message densities, and note that the bounds are very close to those obtained via DE.

VI. CONCLUSION

In this work, we present an efficient implementation for constructing punctured PCs using DE. To do so, we establish DE assuming an SC decoder for both regular, i.e. unpunctured, as well as punctured PCs in a unified framework.

To minimize the number of convolutions necessary for the construction of a punctured PC, we analyze how punctured positions, treated as erasures, affect the convolutions and exploit these observations in the algorithm presented. In addition, the presented approach allows for the construction of regular PCs via DE as well, resulting in a unified way to construct both regular and punctured PCs using DE. In order to analyze the reduction in construction complexity achieved by our method, we provide upper bounds on the number of unique density expressions to compute, and consequently on the number of convolutions necessary for constructing a punctured PC.

Empirical investigations validate our approach. Using puncturing patterns obtained via both QUP as well as URP, we are able to save a substantial number of the convolutions with respect to an implementation that performs all convolutions, thereby evaluating identical density expressions multiple times. In particular, when using random puncturing patterns obtained via URP, we save more than 40 % of the convolutions on average for mother codes shorter than 2^{13}, while for QUP patterns and mother code lengths greater than 2^9, these savings amount to well above 80 % of the convolutions.

As a result, the method presented allows for greatly accelerating the construction of punctured PCs using DE, and, as a meta algorithm, facilitates these speed-ups for arbitrary implementations of the convolution operations. Finally, we present simulation results for both regular as well as punctured PCs and corresponding upper bounds on the respective BLERs obtained by the implementation presented in this work. For both classes of codes, we observe extremely tight bounds, and hence provide empirical evidence for the predictive power of the bounds obtained by our implementation.

REFERENCES

[1] E. Arıkan, "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.
[2] A. Eslami and H. Pishro-Nik, "A practical approach to polar codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul./Aug. 2011, pp. 16–20.
[3] M. Bakshi, S. Jaggi, and M. Effros, "Concatenated polar codes," Jan. 2010, *arXiv:1001.2545*. [Online]. Available: http://arxiv.org/abs/1001.2545
[4] H. Mahdavifar, M. El-Khamy, J. Lee, and I. Kang, "On the construction and decoding of concatenated polar codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 952–956.
[5] P. Trifonov and P. Semenov, "Generalized concatenated codes based on polar codes," in *Proc. IEEE 8th Int. Symp. Wireless Commun. Syst.*, Nov. 2011, pp. 442–446.
[6] S. Korada, E. a o lu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6253–6264, Dec. 2010.
[7] D. Shin, S.-C. Lim, and K. Yang, "Design of length-compatible polar codes based on the reduction of polarizing matrices," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2593–2599, Jul. 2013.
[8] V. Miloslavskaya, "Shortened polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4852–4865, Sep. 2015.
[9] J. Kim, J.-H. Kim, and S.-H. Kim, "An Efficient search on puncturing patterns for short polar codes," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2015, pp. 182–184.
[10] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aug. 2011, pp. 1–5.
[11] I. Tal and A. Vardy, "List decoding of polar codes," May 2012, *arXiv:1206.0050*. [Online]. Available: http://arxiv.org/abs/1206.0050
[12] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2009, pp. 1496–1500.
[13] K. Niu, K. Chen, and J.-R. Lin, "Beyond turbo codes: Rate-compatible punctured polar codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2013, pp. 3423–3427.
[14] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge Univ. Press, 2008.
[15] L. Chandesris, V. Savin, and D. Declercq, "On puncturing strategies for polar codes," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2017, pp. 766–771.
[16] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.
[17] C. Schnelling and A. Schmeink, "Construction of polar codes exploiting channel transformation structure," *IEEE Commun. Lett.*, vol. 19, no. 12, pp. 2058–2061, Dec. 2015.
[18] L. Zhang, Z. Zhang, X. Wang, Q. Yu, and Y. Chen, "On the puncturing patterns for punctured polar codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun./Jul. 2014, pp. 121–125.
[19] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
[20] H. Jin and T. Richardson, "A new fast density evolution," in *Proc. IEEE Inf. Theory Workshop-ITW Punta Del Este*, Mar. 2006, pp. 183–187.
[21] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3221–3227, Nov. 2012.
[22] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

CHRISTOPHER SCHNELLING received the Diploma degree in computer engineering and the master's degree in management, business, and economics from RWTH Aachen University, Germany, in 2014 and 2016, respectively. He is currently pursuing the Ph.D. degree in electrical engineering. His research interests include channel coding and polar codes.

MARKUS ROTHE received the Diploma and Ph.D. degrees in electrical engineering from RWTH Aachen University, Germany, in 2011 and 2018, respectively. His research focuses on optimization.

NIKLAS KOEP received the B.Sc. and M.Sc. degrees in electrical engineering from RWTH Aachen University, Germany, in 2010 and 2013, respectively, where he is currently pursuing the Ph.D. degree in electrical engineering with the Institute for Theoretical Information Technology. His research interests include the broad areas of statistical signal processing, compressed sensing, as well as nonlinear optimization.

RUDOLF MATHAR received the Ph.D. degree from RWTH Aachen University, in 1981. His previous positions include Lecturer positions with Augsburg University and with the European Business School. In 1989, he joined the Faculty of Natural Sciences, RWTH Aachen University. He has held the International IBM Chair in computer science with Brussels Free University, in 1999. In 2004, he was appointed the Head of the Institute for Theoretical Information Technology, Faculty of Electrical Engineering and Information Technology, RWTH Aachen University. Since 1994, he has been holding numerous visiting professor positions with the University of Melbourne, Canterbury University, Christchurch, Johns Hopkins University, Baltimore, and others. In 2002, he was a recipient of the prestigious Vodafone Innovation Award. In 2010, he was an Elected Member of the NRW Academy of Sciences and Arts. He is the co-founder of two spin-off enterprises. From October 2011 to July 2014, he has served as the Dean of the Faculty of Electrical Engineering and Information Technology. In April 2012, he was an Elected Speaker of the Board of Deans, RWTH Aachen University. Since August 2014, he is serving as Prorector for the research and structure with RWTH Aachen University. His research interests include information theory, mobile communication systems, particularly optimization, resource allocation, and access control.

ANKE SCHMEINK received the Diploma degree in mathematics, with a minor in medicine, and the Ph.D. degree in electrical engineering and information technology from RWTH Aachen University, Germany, in 2002 and 2006, respectively. She was a Research Scientist for Philips Research before joining RWTH Aachen University, in 2008, where she has been an Associate Professor, since 2012. She has spent several research visits with the University of Melbourne, and with the University of York. She is a member of the Young Academy at the North Rhine-Westphalia Academy of Science. Her research interests include information theory, systematic design of communication systems, and bioinspired signal processing.