Highly Flexible Design of Multi-Rate Multi-Length Quasi-Cyclic LDPC Codes

Moritz Beermann, Florian Wickert, Peter Vary
Institute of Communication Systems and Data Processing (ind)
RWTH Aachen University, Germany
{beermann|wickert|vary}@ind.rwth-aachen.de

***Abstract*—The design of Low-Density Parity-Check (LDPC) codes with fixed code rate and block length for a fixed channel condition has been well investigated and very close-to-capacity performance can be achieved by careful optimization of a code's *degree distributions*. The growing variety of services supported by mobile communication systems, however, constitutes the need for highly flexible Forward Error Correction (FEC) schemes. Pursuing this need, we present a design method for the construction of quasi-cyclic (QC) LDPC codes supporting arbitrarily many block lengths and code rates using only a single common mother code. Different block lengths are achieved by an optimized expansion of the mother code's *lifting matrix*. For the support of multiple code rates, a joint optimization of shortening and puncturing distributions as well as an optimized check matrix construction based on the *progressive edge-growth* algorithm is employed.**

I. Introduction

Low-Density Parity-Check (LDPC) codes doubtlessly belong to today's most widely used and most extensively optimized Forward Error Correction (FEC) schemes [1]. For many fixed transmission channels, the performance of specifically tailored LDPC codes has been shown to closely approach the capacity limit, given that the channel condition is known and the block length, and thus the latency, tends to infinity. FEC implementations of modern communication systems, however, are required to operate over a wide range of transmission qualities and at the same time support an ever-growing variety of applications. While theoretically possible, it is far from being practical to keep one specifically optimized code in readiness for each of the overabundance of possible scenarios.

A number of contributions have dealt with constructing *rate-compatible* LDPC codes which are capable of realizing multiple code rates using only a single so-called *mother code*. Tian and Jones [2] were the first to propose a combined approach using *information shortening* and *parity puncturing* to cover code rates 0.1 to 0.9 with a single mother code. More recently, also *length-compatible* LDPC codes have been investigated [3], which allow to use multiple code lengths at a fixed code rate. Most practically employed LDPC codes are *quasi-cyclic* (QC) LDPC codes, which are especially suitable for a parallelized implementation due to their structure of cyclically shifted identity matrices. Furthermore, QC LDPC codes inherently support multiple code lengths by using different *lifting factors* together with a common *lifting matrix* to construct codes of different lengths without changing the underlying code structure and, thus, allowing to use a single hardware implementation.

While some approaches exist to combine multi-length QC [4] or other structured [5], [6] LDPC codes with rate-compatibility, they are either limited to special ranges of code rate and length combinations, or only the rate adaptation is optimized without explicitly considering how to achieve good performance at multiple code lengths.

In this paper, we propose a novel highly flexible algorithm to construct a single QC lifting matrix that offers good error correction performance at arbitrarily many code rates and lengths. Both the asymptotic as well as the finite length performance are considered by using jointly optimized puncturing and shortening schemes as proposed in [7] together with a novel *progressive edge-growth* (PEG) [8] based construction method successively taking multiple lifting factors into account to guarantee good performance at all resulting code lengths.

II. Quasi-Cyclic LDPC Codes

A. Notation and General Description

A systematic, binary (N, K) LDPC code with N code bits, $K = N - M$ information bits, and thus code rate $r = \frac{K}{N}$ is described by a binary sparse parity check matrix $\boldsymbol{H}$ of full rank, with elements $H_{m,n}$, and of dimension $M \times N$. We denote the submatrix consisting of columns n_1 to n_2 by $\boldsymbol{H}_{[n_1 : n_2]}$. Equivalently to the matrix description, a bipartite graph with N variable nodes, M check nodes, and edges according to the non-zero entries of $\boldsymbol{H}$ is often used for illustrating *message-passing* decoders like the well-known *Belief Propagation* (BP) algorithm. Since a completely random structure of the parity check matrix is not well suited for implementation, most practical systems employ so-called *quasi-cyclic* (QC) LDPC codes. The parity check matrix of QC LDPC codes can be efficiently stored and its structure enables a straightforward parallelized decoder implementation. The QC check matrix is given by

$$\boldsymbol{H}^{[Z]} = \begin{pmatrix} \boldsymbol{I}_s^{A_{1,1}} & \boldsymbol{I}_s^{A_{1,2}} & \cdots & \boldsymbol{I}_s^{A_{1,N'}} \\ \boldsymbol{I}_s^{A_{2,1}} & \boldsymbol{I}_s^{A_{2,2}} & \cdots & \boldsymbol{I}_s^{A_{2,N'}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{I}_s^{A_{M',1}} & \boldsymbol{I}_s^{A_{M',2}} & \cdots & \boldsymbol{I}_s^{A_{M',N'}} \end{pmatrix}, \quad (1)$$

with the *lifting factor* Z and $N' = N/Z$ and $M' = M/Z$. All submatrices in (1) are either an all-zero matrix of size $Z \times Z$ or a cyclically shifted identity matrix of size $Z \times Z$. The $Z \times Z$ identity matrix, cyclically right shifted by c positions

is denoted $\boldsymbol{I}_s^c$. Furthermore, for notational convenience, we define $\boldsymbol{I}_s^{-1}$ as the $Z \times Z$ all-zero matrix. To efficiently store a QC check matrix, the so-called *lifting matrix* $\boldsymbol{A}$ with dimension $M' \times N'$ and entries $A_{m',n'} \in \{-1, 0, 1, \ldots, Z-1\}$ is used. Entry $A_{m',n'}$ represents the number of right shifts of the submatrix at rows $(m'-1)Z+1, \ldots, m'Z$ and columns $(n'-1)Z+1, \ldots, n'Z$ of $\boldsymbol{H}^{[Z]}$ (or an all-zero matrix if $A_{m',n'} = -1$). In the following, primed index variables (e.g., m') are used whenever referring to a position within the $M' \times N'$ lifting matrix $\boldsymbol{A}$ and unprimed index variables when referring to a position within the $M \times N$ check matrix $\boldsymbol{H}^{[Z]}$.

B. Degree Distributions and Thresholds

The distribution of the different column and row weights of $\boldsymbol{H}$ (and equally $\boldsymbol{H}^{[Z]}$) can be described by a so-called *degree distribution* (DD) *pair* [9] in *node perspective* with the following polynomial notation:

$$\Lambda(x) = \sum_{i=2}^{c_{\max}} \Lambda_i x^i, \quad R(x) = \sum_{j=2}^{r_{\max}} R_j x^j,$$

where $c_{\max}$ is the maximum column weight and $r_{\max}$ the maximum row weight. The coefficients Λ_i and R_j correspond to the proportion of columns of weight i and rows of weight j, respectively. Due to the unity column and row weights of a (shifted) identity matrix, both column and row DDs of the expanded parity check matrix $\boldsymbol{H}^{[Z]}$ are equivalent to the column and row DDs of the lifting matrix $\boldsymbol{A}$ if the weight of a column (row) in $\boldsymbol{A}$ is defined as the number of entries that differ from -1. The asymptotically (for $N \to \infty$) achievable error correcting performance of an LDPC code is mainly determined by its DD pair. Via *density evolution* (DE) analysis, a threshold of the lowest channel quality for which error free decoding is theoretically possible under BP decoding can be computed [9]. In this paper, we consider DDs optimized for the *Binary Input Additive White Gaussian Noise* (BIAWGN) channel from [7].

III. Multi-Rate Optimization

For achieving rate-compatibility using only a single mother code and supporting an arbitrary number of L code rates with arbitrary range between 0 and 1, we consider a combination of *information shortening* and *parity puncturing* as already presented in [7]. While all details of the DD based optimization process can be found in [7], here we only outline the general features of the approach and explain how to tailor the approach towards QC LDPC codes. Additionally, we present an improved method for the construction of a rate-compatible parity check matrix, based on the PEG algorithm for QC LDPC codes [10] using the PEG improvement proposed in [11] and ensuring fast recoverability [12] of punctured bits.

A. Information Shortening and Parity Puncturing

Starting from a systematic (N, K) mother code of fixed code rate r_μ, information shortening allows to realize lower code rates $r_l < r_\mu$ by only using K_{eff} of the information positions and w.l.o.g. setting the remaining $K_{\text{s}} = K - K_{\text{eff}}$ positions to zero. The resulting *effective* code rate is given by

$$r_l = \frac{K_{\text{eff}}}{K_{\text{eff}} + M} = \frac{\left(1 - \frac{K_{\text{eff}}}{K}\right) r_\mu}{1 - \frac{K_{\text{eff}}}{K} r_\mu} < r_\mu. \tag{2}$$

On the other hand, puncturing $M_{\text{p}} = M - M_{\text{eff}}$ of the generated parity bits before transmission results in the effective code rate

$$r_h = \frac{K}{K + M_{\text{eff}}} = \frac{r_\mu}{1 - (1 - r_\mu)\frac{M_{\text{p}}}{M}} > r_\mu. \tag{3}$$

In the following, we will assume that always the first K_{s} positions within the information bits are shortened and the last M_{p} positions within the parity bits are punctured. This can be achieved by intentionally constructing the parity check matrix this way or by reordering the matrix columns accordingly.

In case of shortening, the decoding process can equivalently be described by straightforward decoding of the code corresponding to the truncated matrix $\boldsymbol{H}_{[K_s+1:N]}$ [2], [13]. For the asymptotic analysis of a shortened code of rate r_l we refer to the column DD of the truncated matrix as the *effective* DD (in node perspective)

$$\Lambda^{[r_l]}(x) = \sum_{i=2}^{c_{\max}} \Lambda_i^{[r_l]} x^i.$$

The theoretical effect of punctured (and thus *erased*) bits during BP decoding was incorporated into the DE framework by Ha et al. in [14]. In accordance with [14], [7], we define the *puncturing polynomial* (PP) of the effective rate r_h as

$$\Pi^{[r_h]}(x) = \sum_{i=2}^{c_{\max}} \Pi_i^{[r_h]} x^i,$$

where $\Pi_i^{[r_h]}$ denotes the fraction of degree i nodes to be punctured. For the asymptotic evaluation of the shortened codes, straightforward density evolution using $\Lambda^{[r_l]}(x)$ can be used. For the punctured codes, an adaptation of the modified density evolution for punctured LDPC codes as proposed for a Gaussian Approximation in [14] to the case of discretized density evolution is employed.

B. Joint Rate-Compatible Optimization [7]

Given a set of L target code rates $r_1 < r_2 < \cdots < r_L$ including a mother code rate $r_\mu \in \{r_1, \ldots, r_L\}$ in an arbitrary range between 0 and 1, the joint rate-compatible optimization process presented in [7] yields L polynomials:

- $\Lambda^{[r_\mu]}(x) = \Lambda(x)$: a variable node degree distribution for the mother code of rate $r_\mu \in \{r_1, \ldots, r_L\}$,
- $\Lambda^{[r_l]}(x)$: an effective variable node degree distribution for each target code rate $r_l \in \{r_1, \ldots, r_{\mu-1}\}$ with $r_l < r_\mu$,
- $\Pi^{[r_h]}(x)$: a puncturing polynomial for each target code rate $r_h \in \{r_{\mu+1}, \ldots, r_L\}$ with $r_h > r_\mu$.

During the optimization, check node DDs are assumed to be *concentrated* to at most two consecutive degrees, which has been proven not to degrade the performance [15] and can be ensured by the subsequent matrix construction.

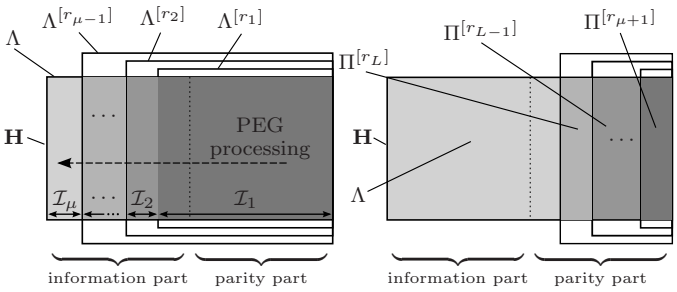

Fig. 1. Visualization of rate-compatible matrix structure

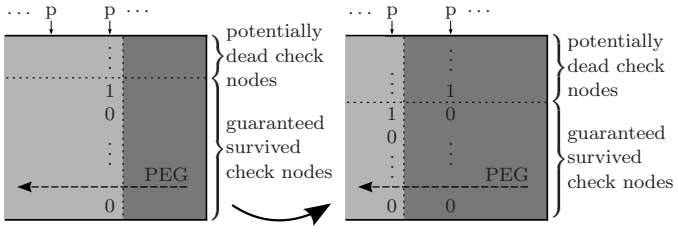

Fig. 2. Ensuring recoverability of all punctured bits marked by "p". Dark shaded area has already been filled by PEG, light shaded area is still empty. Definition of *dead check nodes* and *survived check nodes* according to [12].

C. Application to QC LDPC Codes

Even though, the described bit-wise shortening and puncturing methods can in principle be applied to QC LDPC codes, we restrict our approach to operate on submatrix level to comply with the structured quasi-cyclic concept. By always shortening (puncturing) all Z bits corresponding to one submatrix, operations can be performed based on the lifting matrix $\boldsymbol{A}$ instead of the fully expanded check matrix $\boldsymbol{H}^{[Z]}$, maintaining the block structure of the resulting shortened (punctured) code. Consequently, either $K'_{\mathrm{s}} = \frac{K_{\mathrm{s}}}{Z}$ blocks are shortened or $M'_{\mathrm{p}} = \frac{M_{\mathrm{p}}}{Z}$ blocks are punctured. As described in Sec. II-B, by designing the lifting matrix $\boldsymbol{A}$ according to the jointly optimized DDs, the expanded check matrix $\boldsymbol{H}^{[Z]}$ will by definition exhibit the same DDs. This strategy will also prove useful for supporting multiple block lengths since the same shortening (puncturing) rules can be used for different lifting factors Z.

D. Optimized Rate-Compatible (RC) Matrix Construction

To construct a rate-compatible (RC) lifting matrix $\boldsymbol{A}$ that follows the shortening and puncturing distributions specified by the joint optimization, we define the variable node degree sequence $\boldsymbol{d} = (d_1, \ldots, d_{N'})$, such that each submatrix as shown in Fig. 1 has the correct column degree distribution.

1) Check Node Concentration: Using the RC degree sequence $\boldsymbol{d}$, we employ the QC PEG algorithm [10], processing columns from right to left to ensure a (nearly) concentrated check node DD of the effective shortened matrices.

2) ACE Properties: We have observed that the performance of PEG codes (especially in the error floor domain) is strongly influenced by the order in which degrees are generated, which is confirmed by analyzing the codes' *approximate cycle extrinsic message degree* (ACE) properties [16]. By generating degrees in strictly ascending order, as initially proposed by Hu [8], cycles with only low degree variable nodes, and thus low ACE values, are effectively avoided. If a more randomized degree order, as in case of our specially constructed degree sequence, is used, this property is not conserved and low ACE cycles are created. To counteract this effect, we employ two improvements. Firstly, using the RC degree sequence from above, within each submatrix relevant for shortening, columns are generated in ascending degree order. The index sets of these matrices are given by $\mathcal{I}_1 = \{N' - \frac{M'}{1-r_1} + 1, \ldots, N'\}$ and $\mathcal{I}_i = \{N' - \frac{M'}{1-r_i} + 1, \ldots, N' - \frac{M'}{1-r_{i-1}}\}$ for $i \in \{2, \ldots, \mu\}$, where $\mathcal{I}_1$ is visualized by the darkest and $\mathcal{I}_\mu$ by the lightest shade of gray in the left-hand side of Fig. 1. This way, the check concentration of submatrices is maintained and low ACE cycles within each submatrix are reduced. Secondly, to further reduce the number of low ACE cycles between submatrices, we use the improved PEG rule as proposed in [11], which selects the check node that maximizes the local ACE value whenever multiple check node candidates are present throughout the PEG algorithm.

3) Recoverability of Punctured Bits: To ensure fast recoverability of punctured bits, whenever processing a punctured variable node during PEG, we connect it to exactly one *guaranteed survived check node* [12] that has not been connected to any other punctured node so far (cf. Fig. 2). Hence, the current punctured node will be *1-step-recoverable*. Before proceeding with the next node, the connected guaranteed survived check node is marked as *potentially dead check node* [12]. Since this check node might later be connected to other punctured nodes, not all punctured nodes will remain 1-step-recoverable. However, due to the inherent randomness of the PEG algorithm, creating long chains of dependent punctured nodes with the proposed method is very unlikely. Empirical observations show that even if as many as $90\,\%$ of the parity bits are punctured, all of them are recoverable within a few iterations. A full algorithmic description is included in Alg. 1 together with the multi-length optimization of Sec. IV.

IV. Multi-Length Optimization

The proposed algorithm yields a multi-rate multi-length code by considering the RC degree sequence $\boldsymbol{d}$ from above (cf. Sec. III-D) and a set $\mathcal{P}$ containing the punctured positions (which in our case are always the rightmost positions). It is based on the QC PEG [10] algorithm and, thus, connects variable node blocks to check node blocks one at a time. To obtain a lifting matrix $\boldsymbol{A}$ that can be expanded by factors $Z_1 < \cdots < Z_e$, whenever placing a value in the lifting matrix, we first select candidates from all combinations of block indices and shift values from $\{0, \ldots, Z_1 - 1\}$ for the expansion by Z_1, and then for Z_2 add all combinations with shift values from $\{Z_1, \ldots, Z_2 - 1\}$ that do not change the candidates for the smaller expansion if the shift value is taken modulo Z_1. For Z_3 to Z_e we repeat this step and finally pick an index and shift value combination from the resulting candidates that hence proved to be good at all expansions.

We define $\mathcal{V}^{[i]}$ ($\mathcal{C}^{[i]}$) as the set of all variable (check) node indices and $\mathcal{M}_n^{[i]}$ ($\mathcal{N}_m^{[i]}$) as the set of all check (variable)

node indices connected to variable (check) node n (m), each with respect to the current check matrix $\boldsymbol{H}^{[Z_i]}$ expanded from $\boldsymbol{A}$ with lifting factor Z_i. The order in which blocks are processed was already discussed in Sec. III-D2. The set $\mathcal{S}$ is defined as all *guaranteed survived* check node block positions in the lifting matrix and initially contains all of them. Whenever the first check node block is connected to variable node block n', a minimum degree check node m from $\mathcal{C}^{[e]}$ is picked. If n' is punctured, this chosen node additionally has to be a guaranteed survived check node from $\mathcal{S}$ to ensure *1-step-recoverability* for now. Because of the circular structure of the submatrices in $\boldsymbol{H}^{[Z_i]}$, the remaining variable nodes in group n' are implicitly connected through the cyclically shifted identity matrix. The corresponding shift value $A_{\lceil \frac{m}{Z_i} \rceil, n'} = s^{[e]}(m)$ in the lifting matrix is determined by the function $s^{[i]}(m) := (M' Z_i + 1 - m) \bmod Z_i$, which ensures connecting row m and column $n = (n'-1)Z_i + 1$ (i.e., the first column index within block n') in the expanded matrix $\boldsymbol{H}^{[Z_i]}$. If variable node block n' is punctured, the connected check node block $\lceil m/Z_e \rceil$ is declared *potentially dead* and therefore removed from $\mathcal{S}$.

The remaining $d_{n'} - 1$ block connections are established one after another by randomly picking a check node m from candidate set $\mathcal{K}^{[e]}$, which is constructed as follows. $\mathcal{K}^{[i]}$ denotes the set of check node index candidates for expansion factor Z_i. Candidate set $\mathcal{K}^{[1]}$ at the smallest expansion factor Z_1 is initialized with the indices of all check nodes in $\boldsymbol{H}^{[Z_1]}$ that are not yet connected to variable node block n'. If n' is punctured, all guaranteed survived check nodes are removed from $\mathcal{K}^{[1]}$ since exactly one of them was already connected by the first edge (cf. Sec. III-D3). Next, a tree in $\boldsymbol{H}^{[Z_i]}$ is expanded to find the set $\mathcal{T}_n^l$, containing all check nodes in the tree originating from the first variable node n in group n' up to depth l, such that $\mathcal{T}_n^l$ stops increasing or $\bar{\mathcal{T}}_n^l \neq \emptyset$ but $\bar{\mathcal{T}}_n^{l+1} = \emptyset$, where $\bar{\mathcal{T}}_n^l$ is the complement set of $\mathcal{T}_n^l$. Then, l is decremented until there is at least one check node in $\mathcal{T}_n^l$ that is also a candidate in $\mathcal{K}^{[i]}$. The temporary set $\mathcal{H}$ comprises all check nodes having minimum current degree within $\mathcal{T}_n^l \cap \mathcal{K}^{[i]}$. As proposed in [11], the candidate set $\mathcal{K}^{[i]}$ is then compiled from all check nodes in $\mathcal{H}$ that maximize $\mathrm{ACE}_m^{[i]}(n)$, the ACE value of the shortest cycle passing through variable node n and check node m if an edge was placed between n and m in matrix $\boldsymbol{H}^{[Z_i]}$. For the next expansion factor Z_{i+1}, candidate set $\mathcal{K}^{[i+1]}$ is constructed by including all check nodes, for which a corresponding check node in each $\mathcal{K}^{[j]}$ with $j \leq i$ exists, such that these check nodes are in the same group and the remainders of their shift values are equal if divided by the respective lifting factor Z_j. This process is repeated until $\mathcal{K}^{[e]}$ is found. Finally, a check node m is randomly chosen from $\mathcal{K}^{[e]}$ and connected to variable node n. Again, the remaining variable nodes in group n' are implicitly connected by the shifted identity matrix with shift value $s^{[e]}(m)$.

Algorithm 1 Multi-Rate Multi-Length QC ACE-Opt. PEG

Input: M', N', $\boldsymbol{Z} = (Z_1, \ldots, Z_e)$, $\boldsymbol{d} = (d_1, \ldots, d_{N'}), \mathcal{P}$
Output: $\boldsymbol{A} \in \{-1, 0, 1, \ldots, Z_e - 1\}^{M' \times N'}$

```
A_{m',n'} ← −1, ∀ m', n'
S ← {1, ..., M'}
for ñ' = N' ... 1 do
  pick n' ∈ I_i as highest unconnected index with lowest degree and i ∈ {1, ..., μ} such that ñ' ∈ I_i
  for d = 1 ... d_{n'} do
    if d = 1 and n' ∈ P then
      randomly pick m ∈ argmin_{m̃ ∈ C^[e] : ⌈m̃/Z_e⌉ ∈ S} |N_m̃^[e]|
      S ← S \ {⌈m/Z_e⌉}
    else if d = 1 then
      randomly pick m ∈ argmin_{m̃ ∈ C^[e]} |N_m̃^[e]|
    else
      K^[1] ← C^[1] \ ⋃_{ñ=(n'−1)Z_1+1}^{n'Z_1} M_ñ^[1]
      if n' ∈ P then
        K^[1] ← K^[1] \ {m ∈ C^[1] : ⌈m/Z_1⌉ ∈ S}
      end if
      for i = 1 ... e do
        n ← (n' − 1)Z_i + 1
        expand tree from variable node n in H^[Z_i] until T_n^l stops increasing or T̄_n^l ≠ ∅ but T̄_n^{l+1} = ∅
        while T̄_n^l ∩ K^[i] = ∅ do
          l ← l − 1
        end while
        H ← argmin_{m ∈ T_n^l ∩ K^[i]} |N_m^[i]|;  K^[i] ← argmax_{m ∈ H} ACE_m^[i](n)
        if i < e then
          K^[i+1] ← {m ∈ C^[i+1] : ∀ j ≤ i, ∃ m̃ ∈ K^[j],
            ⌈m/Z_{i+1}⌉ = ⌈m̃/Z_j⌉ ∧ s^[i+1](m) mod Z_j = s^[j](m̃)}
        end if
      end for
      randomly pick m ∈ K^[e]
    end if
    A_{⌈m/Z_e⌉, n'} ← s^[e](m)
  end for
end for
```

V. Simulation Results

To show the effectiveness of the proposed algorithm, a single rate $1/2$ lifting matrix with $M' = 63$ and $N' = 126$ was constructed, which is capable of operating at effective code rates $r_i \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and using lifting factors $Z_i \in \{4, 8, 16, 32, 64, 128, 256\}$ at block lengths $N' Z_i = N_i \in \{504, 1008, 2016, 4032, 8064, 16128, 32256\}$. The optimized degree distributions with maximum variable node degree 10 and puncturing polynomials were taken from setup *M5* in [7, Table I]. Performance is evaluated by measuring the gap in terms of E_b/N_0 of bit error rate (BER) curves after 100 BP iterations to the BIAWGN capacity at BER 10^{-4}, which is chosen since possible error floor effects are not yet specifically handled during code construction.

Figure 3 shows the gap to capacity over the code rate for different code lengths from 504 to 32256. As a reference, the performance of dedicated PEG codes is shown, which have been constructed with optimized DDs of the same maximum degree of 10 for each rate and with the same effective code length as the multi-rate multi-length code at each operating

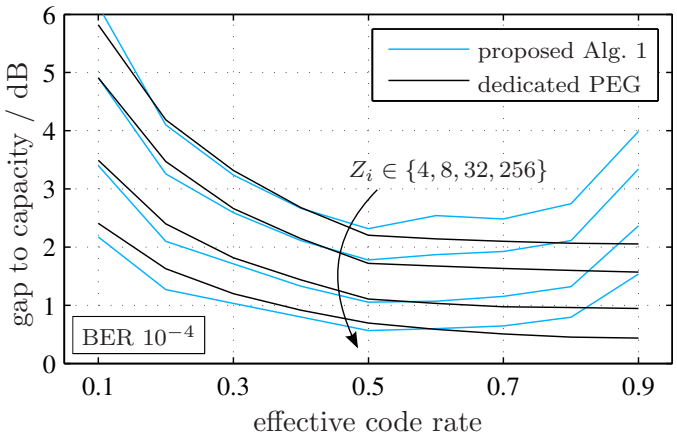

Fig. 3. E_b/N_0-gap to BIAWGN capacity for BER 10^{-4} with 100 BP iterations over code rate for different lifting factors of proposed Alg. 1. Black lines show performance of dedicated PEG codes of same code rate and length.

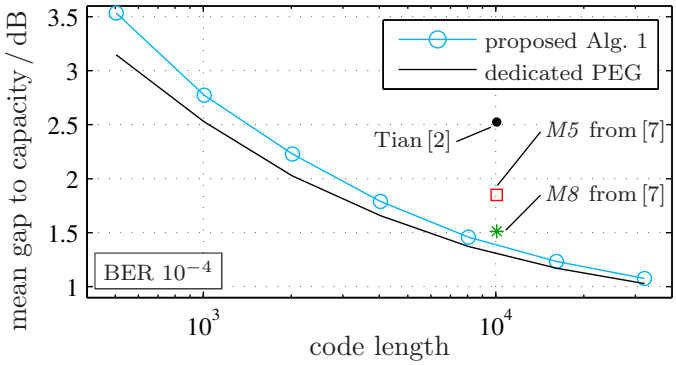

Fig. 4. Mean gap to capacity at BER 10^{-4} with 100 BP iterations averaged over all code rates for each code length. Black lines shows performance of dedicated PEG codes of same code lengths averaged over all code rates.

point. It can be seen that the proposed approach and the dedicated PEG codes perform very similar at the mother code rate 0.5 and at code rates where shortening is used (i.e., below 0.5)[1], while a performance loss of the multi-rate multi-length code is visible which grows with increasing amount of punctured bits (i.e., at code rates above 0.5).

In Fig. 4, the mean gap to capacity achieved at each code length (averaged over all code rates 0.1 to 0.9) of the proposed approach and again the dedicated PEG codes is depicted. It can be seen that the performance penalty of having only a single code for all rates and lengths as compared to having multiple dedicated codes decreases with increasing block length. Since we are not aware of any scheme based on a single mother code offering a comparable extent of flexibility in terms of both code rate and length, as further reference the three schemes Tian [2], [7, *M5*], and [7, *M8*] with code lengths around 10000 are shown, which only support rate-compatibility but no length-compatibility.

VI. Conclusion

A novel PEG-based algorithm was proposed to construct a single multi-rate and multi-length lifting matrix of a QC LDPC code by optimized matrix expansion and jointly optimized information shortening and parity puncturing. The effectiveness of the proposed scheme was demonstrated by a single mother code covering the code rate range from 0.1 to 0.9 and code lengths from 504 to 32256 with only a total average performance loss of 0.19 dB compared to a set of 63 specifically optimized PEG codes for each rate and length combination. While 63 encoder/decoder pairs are needed for this set, the proposed multi-rate multi-length code can be handled by a single encoder/decoder implementation. To the best of our knowledge, no multi-rate multi-length approach based on a single mother LDPC code with the same flexibility and comparable performance has been presented so far. Even better performance can be expected for less widespread code rate ranges which still cover the requirements of many applications.

[1]While the DE thresholds of the dedicated codes are better than those of the proposed scheme, they are in some cases outperformed at BER 10^{-4} due to the different degree ordering in the PEG part of the proposed algorithm.

References

[1] D. J. C. MacKay and R. M. Neal, "Good Codes Based on Very Sparse Matrices," *Cryptography and Coding, 5th IMA Conference*, Springer, Berlin, Germany, 1995, pp. 100–111.

[2] T. Tian and C. R. Jones, "Construction of Rate-Compatible LDPC Codes Utilizing Information Shortening and Parity Puncturing," *EURASIP J. Wirel. Commun. Netw.*, vol. 5, pp. 789–795, Oct. 2005.

[3] K.-J. Kim, J.-H. Chung, and K. Yang, "Design of Length-Compatible Low-Density Parity-Check Codes," *IEEE Comm. Lett.*, vol. 16, no. 5, pp. 734–737, May 2012.

[4] L. Fan, K. Peng, C. Pan, and J. Song, "Multiple-Rate Multiple-Length QC-LDPC Codes Design with Near Shannon Limit Performance," *Proc. IEEE Int. Symp. on Broadband Multimedia Systems and Broadcasting*, June 2013, pp. 1–6.

[5] M. El-Khamy, J. Hou, and N. Bhushan, "Design of Rate-Compatible Structured LDPC Codes for Hybrid ARQ Applications," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 965–973, Aug. 2009.

[6] T. Nguyen, A. Nosratinia, and D. Divsalar, "The Design of Rate-Compatible Protograph LDPC Codes," *IEEE Trans. Comm.*, vol. 60, no. 10, pp. 2841–2850, Oct. 2012.

[7] M. Beermann and P. Vary, "Joint Optimization of Multi-Rate LDPC Code Ensembles for the AWGN Channel Based on Shortening and Puncturing," *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Istanbul, Turkey, Apr. 2014.

[8] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[9] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[10] Z. Li and B. V. K. V. Kumar, "A Class of Good Quasi-Cyclic Low-Density Parity Check Codes Based on Progressive Edge Growth Graph," *Asilomar Conference on Signals, Systems and Computers*, Nov 2004.

[11] H. Xiao and A. H. Banihashemi, "Improved Progressive-Edge-Growth (PEG) Construction of Irregular LDPC Codes," *IEEE Comm. Lett.*, vol. 8, no. 12, pp. 715–717, Dec. 2004.

[12] J. Ha, J. Kim, D. Klinc, and S. McLaughlin, "Rate-Compatible Punctured Low-Density Parity-Check Codes with Short Block Lengths," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 728–738, Feb. 2006.

[13] M. Beermann and P. Vary, "Breaking Cycles with Dummy Bits: Improved Rate-Compatible LDPC Codes with Short Block Lengths," *Proceedings of International ITG Conference on Systems, Communications and Coding*, München, Germany, Jan. 2013.

[14] J. Ha, J. Kim, and S. McLaughlin, "Rate-Compatible Puncturing of Low-Density Parity-Check Codes," *IEEE Trans. Inform. Theory*, vol. 50, no. 11, pp. 2824–2836, Nov. 2004.

[15] S. Chung, T. J. Richardson, and R. Urbanke, "Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[16] T. Tian, C. Jones, J. Villasenor, and R. Wesel, "Selective Avoidance of Cycles in Irregular LDPC Code Construction," *IEEE Trans. Inform. Theory*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.