Breaking Cycles with Dummy Bits: Improved Rate-Compatible LDPC Codes with Short Block Lengths

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

Abstract—**It is well-known that the presence of cycles in a factor graph degrades the performance of message-passing algorithms due to the violation of the assumption of statistical independence of messages. While finding and counting all cycles in a graph is a very hard problem, efficient algorithms have been proposed recently to find all short cycles up to some maximum length. For the message-passing decoding of Low-Density Parity-Check (LDPC) codes, these short cycles are the main cause for the existing performance gap to optimal Maximum Likelihood (ML) decoding. In this paper, we exploit this observation in the proposed novel method of realizing rate-compatible (RC) LDPC codes with short block lengths by inserting known dummy bits into the information bit sequence before encoding. This technique can be seen as a special case of traditional code shortening and allows to achieve lower code rates using one fixed rate mother code. The novelty of the proposed method lies in the selection of the specific dummy bit positions in a way that "breaks" a significant number of short cycles in the mother code's graph which, thus, no longer degrade the decoding performance.**

I. Introduction

Forward Error Correction (FEC) is a key component of virtually every modern digital communication system and enables the reliable transmission of any kind of digital content. In many of these systems, most notably in the field of wireless communications, the transmission channel is not fixed and its quality might change quickly over time. At the same time, the demanded data rate and strength of error protection requirements strongly depend on the supported services, like, e.g., speech, video, or data transmission. The FEC functionality of such systems commonly allows for an adaptation to the varying user requirements and the current state of the transmission channel, in order to make efficient use of the available resources, as, e.g, the limited bandwidth of a wireless communication system. FEC codes that can achieve different protection levels and information data rates (i.e., different code rates) based on one fixed so called *mother code* are frequently called *rate-compatible* (RC).

One of the most powerful FEC schemes are Low-Density Parity-Check (LDPC) codes, originally invented by Gallager in [1] and rediscovered by MacKay in [2]. LDPC codes are included in many of today's communication systems like IEEE 802.16e (WiMAX) [3] or IEEE 802.11n (WLAN) [4]. However, all standards that provide different code rates have in common that separate LDPC codes are specified for each code rate. As a consequence, transmitter and receiver have to support all specified codes, which hinders an efficient implementation and leads to an increased hardware complexity.

This work was supported by the research cluster UMIC located at RWTH Aachen University, Germany, funded by the German research council DFG.

Rather than using multiple codes, there exist many alternative techniques for the construction of LDPC codes of higher or lower code rates from a fixed-rate mother code (in the following this process will be referred to as *rate-matching*). In [5], Li et al. studied parity bit puncturing to construct LDPC codes of higher code rates and proposed a special code extension to achieve lower code rates. Another approach of constructing codes of lower code rates from a high rate mother code was presented in [6] by Tian and Jones using information shortening (which has also been termed *dummy bit insertion* in [7]). Their approach is especially useful for medium to large block lengths since it uses a density evolution [8] based optimization of degree distributions to construct a parity check matrix especially suitable for information shortening. While these kinds of optimizations rely on asymptotic assumptions, like infinite block size and a cycle free graph representation, in the finite length regime, the influence of a code's degree distributions on its performance is less distinctive. Instead, as the block size gets smaller, the density of edges in the code's Tanner graph [9] gets higher and the increasing number of short cycles significantly degrades the decoding performance. To account for short block lengths, different variations of dummy bit insertion [10], [11], puncturing [12], and code extension [13] have been presented.

Motivated by the aforementioned observation that the main cause of performance degradation for short to medium block lengths is the cycle structure of the Tanner graph, we propose a novel method of rate-matching by dummy bit insertion that utilizes the dummy bit positions to significantly reduce the number of short cycles and, thus, is especially effective for short to medium block sizes. The effect of increasing a code's *girth* by deleting columns has also been utilized by Milenkovic et al. in [14]. However, their approach can only be used for specially structured so called *array codes* and was not designed to be used for rate-matching, while our approach does not require any special code structure.

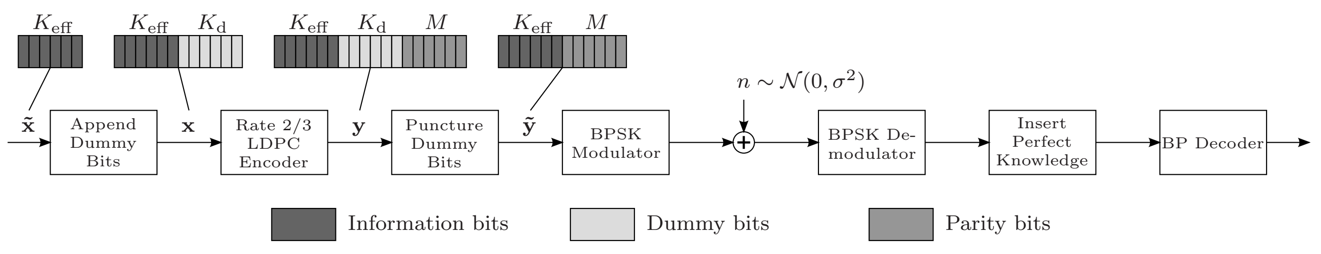

Fig. 1. Exemplary transmission system with dummy bit insertion using a rate $2/3$ LDPC Encoder, BPSK modulation, an AWGN channel and Belief Propagation (BP) Decoding. Encoding with insertion of K_d dummy bits results in the effective code rate $R_{\text{eff}} = \frac{K_{\text{eff}}}{K_{\text{eff}}+M} = \frac{1}{2}$.

Note that in the literature many terms have been used to describe the concept of inserting known bits before encoding, including pruning (e.g., [15]) and code or information shortening (e.g., [6]). Throughout this work, however, we use the term dummy bit insertion as introduced in [7], since we think it best describes the proposed mechanism.

II. Low-Density Parity-Check Codes

A binary (N, K) LDPC code is a linear block code defined by a sparse parity check matrix $\mathbf{H}$ of dimension $M \times N$, with K denoting the number of information bits, N the total number of coded bits and $M = N - K$ the number of parity bits. In the following, we assume $\mathbf{H}$ to have full rank $\text{rank}(\mathbf{H}) = M$. The code rate is then given by $R = \frac{K}{N} = \frac{N-M}{N}$. H_{mn} denotes the entry of $\mathbf{H}$ at row m and column n. The set $\mathfrak{N}(m) = \{n : H_{mn} \neq 0\}$ contains all bits that participate in parity check equation m. Similarly, $\mathfrak{M}(n) = \{m : H_{mn} \neq 0\}$ denotes the set of all check equations in which bit n participates.

An LDPC code is commonly described by a bipartite graph $\mathcal{G} = (\mathcal{V} \cup \mathcal{C}, \mathcal{E})$ (known as factor or Tanner graph [9]) consisting of the set $\mathcal{V}$ of N variable nodes, the set $\mathcal{C}$ of M check nodes and the set $\mathcal{E}$ of E edges connecting variable nodes to check nodes. Each variable node corresponds to a code bit and each check node corresponds to a parity check equation as defined by a row of the parity check matrix. The set of variable nodes $\mathcal{V}$ can be further partitioned into the set $\mathcal{I}$ of K information nodes (variable nodes corresponding to information bits) and the set $\mathcal{P}$ of M parity nodes (variable nodes corresponding to parity bits). Variable node n is connected to all check nodes in the set $\mathfrak{M}(n)$ and check node m is connected to all variable nodes in the set $\mathfrak{N}(m)$.

For the decoding of LDPC codes, we apply the iterative Belief Propagation algorithm as described in [16]. After initialization of each variable-to-check message v_{mn} (i.e., the message sent from variable node n to check node m) with the received channel-related L-value Z_n [17] for the nth bit, the so called horizontal and vertical step are computed alternately. The horizontal step (or check node update) computes the check-to-variable messages c_{mn} (i.e., the message sent from check node m to variable node n) as

$$c_{mn} = \sideset{}{^{\boxplus}}\sum_{n' \in \mathfrak{N}(m) \setminus n} v_{mn'}, \qquad (1)$$

where "$\boxplus$" denotes the *boxplus* operator [17], and "$\setminus$" denotes the exclusion of an element from a set. Likewise, the vertical step updates all variable-to-check messages according to

$$v_{mn} = Z_n + \sum_{m' \in \mathfrak{M}(n) \setminus m} c_{m'n} \,. \qquad (2)$$

Additionally, after each vertical step, the hard decision

$$\hat{y}_n = \text{sign}\{Z_n + \sum_{m' \in \mathfrak{M}(n)} c_{m'n}\} \qquad (3)$$

is computed for each variable node n to evaluate the parity check equations. Decoding is stopped if either all equations are fulfilled or a maximum number of iterations is reached.

III. Rate-Matching by Dummy Bit Insertion

The general concept of achieving lower code rates using a mother code of fixed rate and dummy bit insertion is depicted in Fig. 1 for an exemplary system using a rate $R = \frac{2}{3}$ LDPC code, Binary Phase Shift Keying (BPSK) modulation, an Additive White Gaussian Noise (AWGN) channel, and Belief Propagation (BP) decoding.

A. Encoding

The rate R mother code takes $K = K_{\text{eff}} + K_d$ bits, composed of K_{eff} information bits and K_d known dummy bits, as input and generates $M = \frac{1-R}{R}(K_{\text{eff}} + K_d)$ additional parity bits at the output. In the following, we will w.l.o.g. assume that all dummy bits are set to 0. Before passing the resulting bit stream to the modulator, the dummy bits are removed (punctured) since they are known at the receiver and therefore do not need to be transmitted. The effective block length is denoted $N_{\text{eff}} = K_{\text{eff}} + M$ resulting in the effective code rate

$$\begin{aligned} R_{\text{eff}} &= \frac{K_{\text{eff}}}{K_{\text{eff}} + M} = \frac{K_{\text{eff}}}{K_{\text{eff}} + \frac{1-R}{R}(K_{\text{eff}} + K_d)} \\ &= \frac{1 - k_d}{R^{-1} - k_d} \end{aligned} \qquad (4)$$

where $k_d = \frac{K_d}{K_d + K_{\text{eff}}}$ denotes the fraction of information bit positions that are filled with dummy bits (e.g., $50\,\%$ in Fig. 1). It can easily be seen that any code rate $0 < R_{\text{eff}} \leq R$ can be achieved by an accordant choice of k_d (of course only in discrete steps depending on the absolute values of K_d and K_{eff}). If no dummy bits are used (i.e., $k_d = 0$) $R_{\text{eff}} = R$ holds and if the number of information bits approaches zero (i.e., $k_d \to 1$) the effective code rate also approaches zero.

B. Decoding

At the receiver the dummy bits are known and, thus, provide perfect *a priori* information that is inserted (after demodulation) into the received sequence of L-values in form of values $+\infty$ (according to the known bit value of 0). These positions with perfect knowledge influence the message-passing decoding as follows.

If a parity check equation of the form $x_1 + x_2 + \dots x_\eta = 0$ is considered, in which one or more dummy bits participate that are known to be 0, the equation can be equivalently described by simply discarding those bits from the equation, due to $x_i + 0 = x_i$ (note that here "+" denotes the bitwise XOR operator). An analogous equivalence also holds for the computation of messages in the L-value domain during a check node update according to (1) since $L_i \boxplus +\infty = L_i$. Consequently, the outgoing messages $v_{mn} = +\infty$ of variable nodes n that are associated with dummy bits (these nodes are denoted *dummy nodes* in the following) do not have to be considered in any check node update. Since there is no need for improving the reliability of a dummy bit's (known) value, also the incoming messages c_{mn} of dummy nodes n are of no particular use. According to this observation, all edges connected to dummy nodes can be removed from the Tanner graph without any influence on the decoding process. Hence, the effective graph as "seen" by the decoder is the graph of the mother code with all dummy nodes (and all connected edges) simply removed. Accordingly, the effective parity check matrix is achieved by removing the columns associated with dummy bit positions from the mother code's parity check matrix.

As a consequence, even though the block length of the employed mother code is N, the complexity of decoding is equivalent to that of decoding an LDPC code with smaller block length $N' = N_{\text{eff}} = \frac{1-R}{1-R_{\text{eff}}} N$ and code rate $R' = R_{\text{eff}}$. However, note that it is not mandatory to remove any nodes or edges at all but just serves to reduce complexity. Existing decoder implementations do not have to be changed in any way and are fully compatible with this rate-matching technique.

IV. Proposed Method: Breaking Cycles

As described in the previous section, from a message-passing decoder's point of view, inserting dummy bits means removing edges from the Tanner graph. We propose a novel method of choosing the positions for the dummy bits so that the removal of edges "breaks" as many short cycles as possible. More precisely, since removing a node also removes all cycles running through it, the nodes chosen for dummy bits are the ones with most cycles running through them.

Naturally, this approach will tend to select many high degree nodes, since these are more likely to have a high number of cycles. If the resulting effective graph is desired to have a specific predetermined variable node degree distribution [8], this can easily be achieved by a minor modification of the proposed approach (i.e., pre-compute the required number of dummy bits for each degree and stop choosing these degrees as soon as the target quantity is reached). However, the optimization of degree distributions is not the focus of this paper and will be disregarded for the moment (the influence of a modified degree distribution will be shown in Sec. V-B).

Algorithm 1 Breaking Cycles algorithm for finding optimized order $\mathbf{d}$ of dummy bit positions

```
1: Given a Tanner graph $\mathcal{G} = (\mathcal{I} \cup \mathcal{P} \cup \mathcal{C}, \mathcal{E})$
2: Initialize $\mathcal{I}' = \mathcal{I}$, $\mathcal{E}' = \mathcal{E}$, $\mathcal{G}' = \mathcal{G}$
3: Initialize vector of dummy nodes $\mathbf{d} = [\,]$   // empty vec.
4: while size($\mathbf{d}$) $< K$ do
5:     Find the length $g$ of the shortest cycle in $\mathcal{G}'$
6:     Find the length $g_{\mathcal{I}'}$ of the shortest cycle running through
       any information node $i \in \mathcal{I}'$
7:     if ($g_{\mathcal{I}'} > 2g - 2$) then
8:         break                                   // cycles cannot be found
9:     end if
10:    Compute $C_i^{g_{\mathcal{I}'}}$ for all $i \in \mathcal{I}'$
11:    Find $i^* = \arg\max_{\forall i \in \mathcal{I}'} C_i^{g_{\mathcal{I}'}}$
12:    $\mathbf{d} = [\mathbf{d}, i^*]$                         // append $i^*$ to $\mathbf{d}$
13:    $\mathcal{I}' = \mathcal{I}' \setminus \{i^*\}$                     // remove $i^*$ from $\mathcal{I}'$
14:    $\mathcal{E}' = \mathcal{E}' \setminus \mathcal{E}_{i^*}$   // $\mathcal{E}_{i^*}$ denotes all edges connected to $i^*$
15:    $\mathcal{G}' = (\mathcal{I}' \cup \mathcal{P} \cup \mathcal{C}, \mathcal{E}')$
16: end while
17: if size($\mathbf{d}$) $< K$ then
18:    for all $i \in \mathcal{I}'$ do
19:        $\mathbf{d} = [\mathbf{d}, i]$   // Select remaining positions arbitrarily
20:    end for
21: end if
```

In [18], an efficient message-passing algorithm for counting the number of cycles of length up to $2g - 2$ was presented, with g being the girth, i.e., the length of the shortest cycle in the graph. The complexity of the algorithm grows as $\mathcal{O}\left(gE^2\right)$ which is only feasible for moderate block lengths. We use a slight modification of this algorithm that does not count the global number of cycles but the number C_i^l of cycles of length l running through each information node $i \in \mathcal{I}$ (i.e., each potential candidate for a dummy bit position).

Given a Tanner graph $\mathcal{G} = (\mathcal{I} \cup \mathcal{P} \cup \mathcal{C}, \mathcal{E})$, the proposed algorithm selects information nodes from $\mathcal{I}$ to be used for dummy bits in a node by node manner until all K nodes have been selected. In each step, the selected node i^* is the one with the largest number of cycles $C_{i^*}^{g_\mathcal{I}}$ of the currently shortest length $g_\mathcal{I}$ in the set $\mathcal{I}$. After node i^* has been removed from the graph, the numbers of cycles C_i^l of other nodes are potentially reduced and the shortest cycle length $g_\mathcal{I}$ might increase. Thus, $g_\mathcal{I}$ and $C_i^{g_\mathcal{I}}$ have to be recomputed for all remaining nodes i after each step. It is theoretically possible that at some stage the value $g_\mathcal{I}$ becomes larger than $2g - 2$ (with g still denoting the current global girth). In this case, the values $C_i^{g_\mathcal{I}}$ can no longer be computed by the algorithm in [18]. However, in practice this only happens after almost all K nodes from the set $\mathcal{I}$ have already been selected, so that the few remaining nodes can simply be chosen in some arbitrary fashion. Finally, the algorithm yields an ordered vector of bit positions $\mathbf{d}$ of size K. For each mother code, this algorithm has

Algorithm 2 Simplified Breaking Cycles algorithm for finding optimized order $\mathbf{d}$ of dummy bit positions

```
1: Given a Tanner graph $\mathcal{G} = (\mathcal{I} \cup \mathcal{P} \cup \mathcal{C}, \mathcal{E})$
2: Initialize $\mathcal{I}' = \mathcal{I}$
3: Initialize vector of dummy nodes $\mathbf{d} = [\,]$   // empty vec.
4: Find the length $g_i$ of the shortest cycle running through
   node $i$, for all $i \in \mathcal{I}$
5: Compute $C_i^{g_i}$ for all $i \in \mathcal{I}$
6: Set $C_i^{2t} = 0$ for all $i \in \mathcal{I}$ and all $t$ with $2 \le t < g_i/2$
7: while size($\mathbf{d}$) $< K$ do
8:     Find $l = \min_{\forall i \in \mathcal{I}'} g_i$
9:     Find $i^* = \arg\max_{\forall i \in \mathcal{I}'} C_i^l$
10:    $\mathbf{d} = [\mathbf{d}, i^*]$   // append $i^*$ to $\mathbf{d}$
11:    $\mathcal{I}' = \mathcal{I}' \setminus \{i^*\}$   // remove $i^*$ from $\mathcal{I}'$
12: end while
```

to be executed only once offline to find the vector $\mathbf{d}$. To realize any effective code rate $R_{\text{eff}} < R$, the first $K_d = \frac{1-R_{\text{eff}}R^{-1}}{1-R_{\text{eff}}}K$ positions in $\mathbf{d}$ are then simply chosen as dummy bits. An algorithmic description is given in Alg. 1.

Of course, the columns of the parity check matrix can simply be reordered according to $\mathbf{d}$. This allows to achieve the performance gain of the proposed method by appending the desired number of dummy bits to the end (or likewise to the beginning) of the information bits.

One drawback of the proposed approach is its high computational complexity, since the algorithm of [18] has to be executed K times (line 10 of Alg. 1), which leads to an overall complexity of $\mathcal{O}(N^3)$. Although, the method is mainly intended to be used for moderate block lengths N and for a given code only has to be executed once in advance, this complexity becomes quickly infeasible for block lengths of about a few thousand bits. To deal with this complexity problem, we also propose a simplified version of the above algorithm, that only computes the numbers of cycles C_i^l once and selects dummy bit positions solely based on these values. The simplified algorithm is described in Alg. 2.

Empirical investigations show, that for this simplified version the amount, by which the number of short cycles in the graph is reduced, is almost the same as for the more complex first proposed algorithm. In Tab. I the numbers of cycles are given for an optimized rate 1/2 Progressive Edge-Growth (PEG) LDPC code [19] with block length $N = 2304$, girth $g = 6$, and degree distributions (in node perspective)

$$\lambda(x) = 0.45x^2 + 0.37x^3 + 0.03x^4 + 0.15x^{11}$$
$$\text{and} \quad \rho(x) = 0.47x^7 + 0.53x^8.$$

In the upper part, the global numbers of cycles C^l of lengths $l \in \{4, 6, 8, 10\}$ are shown, while the lower part shows only those cycles that are running through information nodes $\mathcal{I}$ denoted by $C_{\mathcal{I}}^l$, which are of special interest since only these cycles can be eliminated by dummy bits. To achieve these numbers $C_{\mathcal{I}}^l$, we simply count the cycles in a node by node manner leading to cycles potentially being counted multiple times, which is why the values $C_{\mathcal{I}}^l$ are larger than the corresponding values C^l. In addition to the cycles of the mother code, the according numbers are also given if dummy bit insertion with $K_d = 768$ (resp. $k_d = 66.7\,\%$) is used to achieve $R_{\text{eff}} = \frac{1}{4}$. Four different approaches of finding optimized dummy bit positions are considered: the one presented by Liu et al. in [10], the one we presented in [11], the proposed Alg. 1, and the proposed simplified Alg. 2.

TABLE I
Cycles of rate 1/2 PEG LDPC mother code with $N = 2304$

global cycles	C^4	C^6	C^8	C^{10}
mother code, $K_d = 0$	0	2285	141934	3379631
Approach of [10], $K_d = 768$	0	1768	101415	2202533
Approach of [11], $K_d = 768$	0	1822	104556	2294903
Proposed, $K_d = 768$	0	269	9573	119791
Proposed simplified, $K_d = 768$	0	269	10438	128547
node wise cycles of all $i \in \mathcal{I}$	$C_{\mathcal{I}}^4$	$C_{\mathcal{I}}^6$	$C_{\mathcal{I}}^8$	$C_{\mathcal{I}}^{10}$
mother code, $K_d = 0$	0	3506	281903	8414973
Approach of [10], $K_d = 768$	0	2436	183644	4989422
Approach of [11], $K_d = 768$	0	2559	190889	5249099
Proposed, $K_d = 768$	0	0	1086	21745
Proposed simplified, $K_d = 768$	0	0	1990	31124

While the approaches of [10] and [11] only reduce the number of cycles of length 6 by about 20–25 %, both proposed approaches achieve a reduction by about 90 %. A similar behavior is observed for the cycles of length 8 and 10. In the lower part of the table, it is worth noting that within the information nodes $\mathcal{I}$, in fact, the proposed approaches eliminate all cycles of length 6 and more than 99 % of the cycles of lengths 8 and 10.

V. Simulation Results

To numerically evaluate the performance of the proposed rate-matching approach, we employ the rate 1/2 PEG LDPC code with block length $N = 2304$ as described in Sec. IV as mother code and apply dummy bit insertion with $K_d \in \{768, 1024\}$ (resp. $k_d \in \{66.7\,\%, 88.9\,\%\}$) dummy bits to match effective code rates $R_{\text{eff}} \in \{\frac{1}{4}, \frac{1}{10}\}$. The following five setups are considered in Fig. 2:

- Approach of [10] (square markers)
- Approach of [11] (triangle markers)
- Proposed method (see Alg. 1) (plus markers)
- Proposed simplified method (see Alg. 2) (star markers)
- Reference PEG codes (circle markers)

The reference PEG codes (including the used mother code) have been optimized for their specific code rates and serve for comparison only. They have code rates $R \in \{\frac{1}{2}, \frac{1}{4}, \frac{1}{10}\}$ and block lengths $N \in \{2304, 1536, 1280\}$, which correspond to the effective code rates R_{eff} and block lengths N_{eff} achieved by the dummy bit insertion.

After LDPC encoding of the equiprobable information bits, BPSK modulation is applied and the resulting symbols are

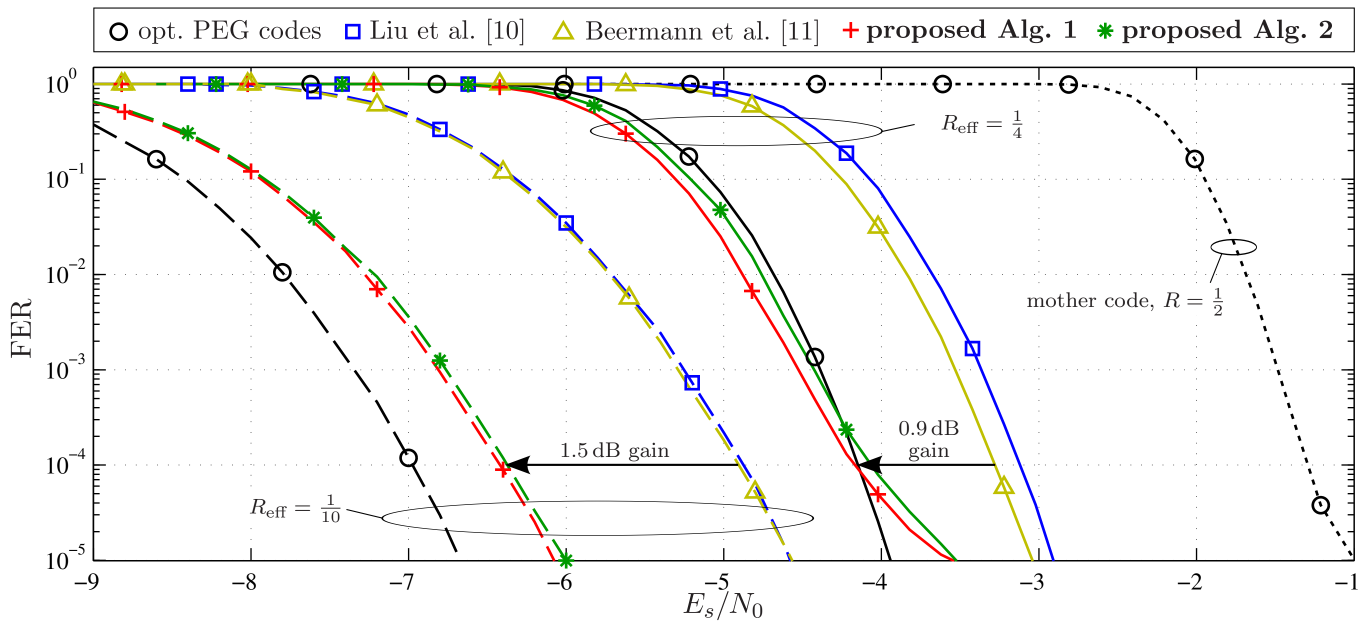

Fig. 2. Frame error rate (FER) results for rate $\frac{1}{2}$ PEG mother code with block length $N = 2304$ and four different approaches of dummy bit insertion: algorithm of [10], algorithm of [11], proposed Alg. 1, and proposed simplified Alg. 2. As a reference, optimized PEG codes of rates $\frac{1}{4}$ and $\frac{1}{10}$ are also shown.

transmitted with energy $E_\mathrm{s} = 1$. The channel is modeled as additive white Gaussian noise (AWGN) channel with (known) power spectral density $\sigma_n^2 = N_0/2$. The receiver uses a soft demapper followed by Belief Propagation decoding as described in Sec. II with a maximum of 100 iterations.

A. Frame error rate performance

Fig. 2 shows the frame error rate performance over the channel quality E_s/N_0 in dB. For $R_{\text{eff}} = \frac{1}{4}$ (solid lines) it can be seen that for the approaches that use dummy bit insertion according to [10] and [11], there is a performance gap of about 1 dB compared to the optimized PEG code. Both of the proposed approaches, however, perform even a little better than the optimized PEG code in the waterfall region, but show a slight error floor for error rates below 10^{-4}. At an error rate of 10^{-4}, a gain of about 0.9 dB is achieved compared to the other two methods. For $R_{\text{eff}} = \frac{1}{10}$, the proposed approaches show a performance gap of about 0.6 dB compared to the reference PEG code, but still clearly outperform the approaches of [10] and [11] by about 1.5 dB.

Furthermore, the simulation results show that in terms of error correction performance, the proposed simplified Alg. 2 performs very close to Alg. 1, which complies with the observation that it also reduces the number of short cycles to almost the same extent.

As a result, the proposed novel method increases the range of code rates for which rate-matching by dummy bit insertion can compete with specifically optimized codes, as compared to previously known methods.

B. Influence of degree distributions

As already indicated at the beginning of Sec. IV, different algorithms for the selection of dummy bit positions can result in different effective degree distributions if the allowed degrees of the dummy nodes are not predefined. The variable node degree distributions of the mother code, the effective rate $\frac{1}{10}$ code after dummy bit insertion according to [10] and according to Alg. 1, as well as the optimized rate $\frac{1}{10}$ PEG code are shown in Tab. II. The decoding thresholds (theoretical waterfall cliffs) have been computed with density evolution [8] under the assumption of concentrated check node degree distributions.

Since these varying degree distributions obviously have an influence on the frame error rate results from the previous section, we have conducted the following additional experiments for the effective rate $R_{\text{eff}} = \frac{1}{10}$: we instantiated 20 random selections of dummy bit positions (i.e., random shortening) with the same degree distribution as resulting from Alg. 1. Furthermore, by predefining the allowed number of dummy nodes for each degree, we have constrained the proposed Alg. 1 to exactly match the degree distribution as achieved by [10]. The results are shown in Fig. 3.

It can be seen that the proposed Alg. 1 achieves better performance than random dummy bit insertion with the same degree distribution (red versus turquois). Moreover, also for the degree distribution achieved by the algorithm from [10] with a much higher threshold, the proposed method still outperforms the reference algorithm (purple versus blue).

Altogether, the novel approach of breaking cycles with dummy bits is an effective method to find an optimized set of dummy bit positions that shows superior error correction performance in a rate-matching scenario with a single mother code. It can easily be used in conjunction with a predetermined optimized degree distribution to achieve near capacity performance over a wide range of code rates (e.g., 0.5 to 0.1 as presented here).

TABLE II

EFFECTIVE VARIABLE NODE DEGREE DISTRIBUTIONS IN NODE PERSPECTIVE AND E_s/N_0 THRESHOLDS

$\lambda(x) = \sum_{i\geq 2} \lambda_i x^i$	λ_2	λ_3	λ_4	λ_{11}	$\left(\frac{E_s}{N_0}\right)^*_{\text{dB}}$
mother code, $R = \frac{1}{2}$	0.450	0.370	0.030	0.150	-2.63
[10], $R_{\text{eff}} = \frac{1}{10}$	0.403	0.334	0.028	0.235	-6.42
Proposed, $R_{\text{eff}} = \frac{1}{10}$	0.492	0.345	0.028	0.135	-8.34
opt. PEG, $R = \frac{1}{10}$	0.718	0.136	0.098	0.048	-11.03

It should further be noted that the proposed method does not rely on any specific code contruction algorithm. In this section the PEG construction was only used as an example since it is one of the most prominent code construction methods. Even though, for this specific combination of code construction and dummy bit insertion, it might be possible to incorporate the cycle breaking apporoach directly into the process of constructing the PEG code (which is also based on the idea of avoiding short cycles), for the sake of generality we consider both processes as seperate entities.

VI. CONCLUSION

In this paper, we have presented a novel approach of dummy bit insertion for the implementation of rate-compatible LDPC codes, allowing to achieve close-to-capacity error correction performance over a wide range of code rates using only a single fixed rate mother code. Known dummy bits are inserted into the information bit sequence before encoding, resulting in a reduced effective code rate. The novelty of the proposed approach lies in the strategy of selecting the positions to be used for dummy bits. Based on a recently published algorithm for counting all short cycles in a bipartite graph, the proposed algorithm selects dummy bit positions in a node by node manner to break as many of those short cycles as possible. As a result, the effective graph, as seen by the decoder, contains only a small fraction of the cycles of the initial mother code. Furthermore, it has been demonstrated that a simplified version with reduced complexity is capable of reducing the number of cycles almost to the same extent. Simulation results have been presented for the two proposed approaches and two reference approaches from the literature which are clearly outperformed by the novel methods.

It has further been demonstrated how the insertion of dummy bits modifies the degree distribution and to what extent this influences the error correction performance. Even though the decoding threshold of the effective code's degree distribution after rate-matching already has a significant effect, the error correction performance is even further improved by the proposed method of breaking cycles with dummy bits.

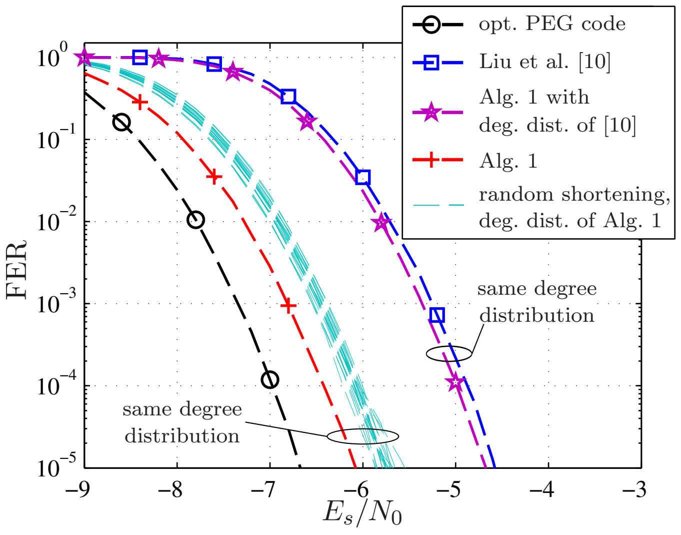

Fig. 3. Evaluation of the influence of different degree distributions of the proposed Alg. 1 and [10] for effective code rate $R_{\text{eff}} = \frac{1}{10}$.

REFERENCES

[1] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Trans. Inform. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[2] 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.

[3] "IEEE 802.16e: Air Interface for Fixed and Mobile Broadband Wireless Access Systems," IEEE Standard 802.16e, 2004.

[4] "IEEE 802.11n, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 5: Enhancements for Higher Throughput," IEEE Standard 802.11n-2009, 2009.

[5] J. Li and K. Narayanan, "Rate-Compatible Low Density Parity Check Codes for Capacity-Approaching ARQ Schemes in Packet Data Communications," *Proceedings Int. Conf. on Communications, Internet, and Information Technology*, Virgin Islands, USA, Nov. 2002, pp. 201–206.

[6] 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.

[7] W. Xu and J. Romme, "A Class of Multirate Convolutional Codes by Dummy Bit Insertion," *IEEE Global Telecommunications Conference (GLOBECOM)*, San Francisco, CA, USA, November 2000.

[8] T. Richardson and R. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[9] R. M. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, Sept. 1981.

[10] X. Liu, X. Wu, and C. Zhao, "Shortening for Irregular QC-LDPC Codes," *IEEE Comm. Lett.*, vol. 13, no. 8, pp. 612–614, Aug. 2009.

[11] M. Beermann, T. Breddermann, and P. Vary, "Rate-Compatible LDPC Codes Using Optimized Dummy Bit Insertion," *8th International Symposium on Wireless Communication Systems*, Nov. 2011, pp. 447–451.

[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] G. Yue, X. Wang, and M. Madihian, "Design of Rate-Compatible Irregular Repeat Accumulate Codes," *IEEE Trans. Comm.*, vol. 55, no. 6, pp. 1153–1163, June 2007.

[14] O. Milenkovic, N. Kashyap, and D. Leyba, "Shortened Array Codes of Large Girth," *IEEE Trans. Inform. Theory*, vol. 52, no. 8, pp. 3707–3722, Aug. 2006.

[15] O. Collins and M. Hizlan, "Determinate State Convolutional Codes," *IEEE Trans. Comm.*, vol. 41, no. 12, pp. 1785–1794, Dec. 1993.

[16] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA: Morgan Kaufmann, 1988.

[17] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. Comm.*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[18] M. Karimi and A. Banihashemi, "A Message-Passing Algorithm for Counting Short Cycles in a Graph," *IEEE Information Theory Workshop (ITW)*, Jan. 2010.

[19] 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.