Rate-Compatible LDPC Codes with Short Block Lengths Based on Puncturing and Extension Techniques

Jingjing Liu and Rodrigo C. de Lamare

Abstract

In this paper, we present novel strategies for generating rate-compatible (RC) irregular low-density parity-check (LDPC) codes with short/moderate block lengths. We propose three puncturing and two extension schemes, which are designed to determine the puncturing positions that minimize the performance degradation and the extension that maximizes the performance. The first puncturing scheme employs a counting cycle algorithm and a grouping strategy for variable nodes having short cycles of equal length in the Tanner Graph (TG). The second scheme relies on a metric called Extrinsic Message Degree (EMD) and the third scheme is a simulation-based exhaustive search to find the best puncturing pattern among several random ones. In addition, we devise two layer-structured extension schemes based on a counting cycle algorithm and an EMD metric which are applied to design RC-LDPC codes. Simulation results show that the proposed extension and puncturing techniques achieve greater rate flexibility and good performance over the additive white Gaussian noise channel, outperforming existing techniques.

Index Terms

Rate-compatible (RC) codes, irregular low-density parity-check (LDPC) codes, cycle counting algorithm, Extrinsic Message Degree (EMD), puncturing technique, extension scheme.

I. Introduction

When the channel state information (CSI) is known at the transmit end and the data transmission takes place over time-varying channels, an error control coding scheme with a fixed code rate is not the best solution. In such situations, an error-correction scheme with flexibility in code rates is desirable since such scheme can encode data at different rates depending on the reliability of the channel. High-rate codes are applied to achieve higher data throughput if the channel condition is good, otherwise low-rate codes are used to ensure reliable transmission. Thus, both capacity and reliability can be realized in such scenarios. However, deploying many pairs of encoders and

Jingjing Liu and Rodrigo C. de Lamare {jl622;rcdl500}@york.ac.uk are with Department of Electronics, The University of York, Heslington, York, YO10 5DD, UK.

decoders is not feasible in practical applications due to their high memory and storage costs. Rate-compatible (RC) codes refer to a family of codes where higher rate codes are embedded in lower rate codes; in other words, the factor graphs of higher rate codes are subgraphs of lower rate codes [1]. For example, Lin and Yu [2] designed an RC coding scheme for a hybrid automatic repeat-request with forward error correction (ARQ/FEC) system, where the transmitter sends additional redundant bits upon request until the decoder claims a successful decoding. Having been applied to convolutional codes [1] and turbo codes [3], RC techniques are proven not only to enhance system performance but also to only require low hardware complexity thanks to the structure of a single pair of encoder and decoder.

Low-Density Parity-Check (LDPC) codes were invented by Gallager [4] and were re-discovered by MacKay et al. [5] as an advanced coding technique for Shannon capacity-approaching performance over a variety of channels [6], [7]. In [8], [9], finite-length LDPC codes were studied for both "waterfall" performance in the low signal-to-noise ratio (SNR) region and error-floor behavior in the high SNR region. Since RC-LDPC codes were first considered in [10], there has been a fair amount of work in this area. Ha et al. [11] derived puncturing distributions via asymptotic analysis while assuming infinite block length and absence of short cycles. Later, in [12] the authors focused on minimizing the number of iterations required to recover punctured bits. Unlike [12], the work reported in [13] tries to maximize the minimum reliability provided via check nodes. An efficiently-encodable irregular LDPC codes along with a puncturing method were derived in [14], where good performance can only be achieved via puncturing degree-2 non-systematic bits. Also, in [15] protograph-based RC ensembles were implemented for hybrid ARQ applications and systematic construction of punctured LDPC codes was achieved via successive maximization, respectively. Other puncturing methods have been reported in [16], [17], where the authors proved the existence of a puncturing threshold with an improved decoding algorithm [16] and enhanced the performance at high SNRs by grouping nodes [16], [17]. More recently, rate-compatible LDPC code designs have been reported based on the approximate cycle extrinsic message degree (ACE) metric [19], protograph structures [20], robust techniques that employ a set of criteria for puncturing [21] and application [22] and hardware [23] constraints. Moreover, extension methods [10] and [18] have been applied by adding extra parity-check bits to increase the size of the parity-check matrix of the mother code. As a result, lower rate codes can be generated based on a high-rate mother code.

In this work, we develop design techniques for RC-LDPC codes with reduced performance degradation as compared to unpunctured codes and existing RC-LDPC codes with the same rates. In our preliminary work [24], three puncturing schemes have been proposed that are able to generate finite-length RC-LDPC codes with high decoding performance at high rates (ranging from 0.5 to 0.9). The first puncturing scheme is a cycle-counting

based (CC-based) technique that exploits the algorithm reported in [25] to determine the puncturing pattern. Given a mother code and a target rate, variable nodes having the largest number of girth-length cycles will be punctured first, such that the decoding performance is expected to improve while breaking the shortest cycles. Using a metric for evaluating the extrinsic message degree (EMD) [26] and [27], a second scheme called approximate cycle EMD based (ACE-based) puncturing is developed, which selects the puncturing pattern by considering the cycle length and graph connectivity simultaneously. Additionally, a third scheme relies on a simulation-based greedy search for the best puncturing pattern among many randomly generated patterns. Based on the structure of short cycles and the ACE spectrum, two extension schemes have been reported in [28]. In this paper, we present puncturing and extension techniques that are further enhanced and detailed as compared to the methods first reported in [24], [28] along with a technical analysis and a comprehensive set of numerical results in terms of bit error rate (BER), frame error rate (FER) and system throughput. The main contributions can be summarized as:

- We present three puncturing schemes which depend upon the counting cycle algorithm, ACE metric and an exhaustive search, respectively. Further performance gain is observed regarding the puncturing techniques reported in [24].
- We propose two extension schemes based on the structure of short cycles and the ACE spectrum. The extension schemes proposed show an improvement over the techniques described in [28].
- A comprehensive study of RC-LDPC codes is carried out from the short cycles point of view. A set of puncturing/extension strategies is made available to create RC-LDPC codes that are highly flexible in block length, regularity (available for both regular and irregular codes) and rate (across a wide range from 0.1 to 0.9).
- A simulation study including extensive comparisons to existing algorithms is presented. The performance of the RC-LDPC codes proposed is shown with respect to BER, FER and throughput for a type-II hybrid automatic repeat-request (ARQ) system, which performs only error detection for the first transmitted message block and only employs LDPC codes for retransmitted blocks.

The organization of this paper is as follows: Section II explains the system model and the basic notation. The proposed puncturing schemes and extension schemes are detailed in Section III and Section IV, respectively. Section V presents simulation results with explanations. Finally, Section VI concludes this paper.

II. System Model and Basic Notation

This section presents a system model for the proposed puncturing and extension schemes, as well as the design strategy behind them. The proposed techniques are based on cycle conditioning for each subgraph (puncturing) or an extended graph (extension). Note that traditional cycle conditioning has only focused on eliminating short cycles in a Tanner Graph (TG) with very long block sizes. However, it has been shown that avoiding short cycles alone is not sufficient to achieve good performance, particularly in the error-floor region, [26] and [27], and cycle conditioning is a challenging task for finite-length LDPC codes. Departing from prior studies, in this work we derive design strategies for RC-LDPC codes over the additive white Gaussian noise (AWGN) channel.

A. Construction of RC-LDPC Codes Using Puncturing

Given a mother or parent LDPC code containing K information bits, the code rate is given by $R = K/N$ where N is the block length. Suppose that $\boldsymbol{m}$ represents the message with K bits from the source, $\boldsymbol{c}$ denotes the encoded data with N bits, $\boldsymbol{c'}$ is the punctured data, and $\hat{\boldsymbol{m}}$ denotes the estimate of the original message using a belief propagation (BP) decoding algorithm given the unpunctured data $\boldsymbol{r}$. The log-likelihood ratio (LLR) of a punctured bit is set to 0 at the beginning of the decoding process. Suppose that P bits are punctured before transmission, so the resulting code rate is given by $R' = K/(N-P)$ and the puncturing rate by $\rho = P/N$. We assume that the decoder has perfect knowledge regarding the puncturing pattern, i.e., the positions of punctured bits in a codeword. Otherwise, some side information is needed to send the puncturing pattern to the receiver end. Puncturing is a common and simple method to construct RC codes, for which a higher rate is achievable by means of removing a subset of encoded bits $\boldsymbol{c}$ [16]. A randomly chosen puncturing pattern [10] can be used to realize the rate compatibility at the expense of significant performance degradation. Intentional puncturing methods have been investigated for short block-length LDPC codes in [12] - [14] and [17], ranging from asymptotic analysis to grouping and sorting variable nodes. In contrast to those methods, the proposed puncturing schemes aim to reduce the performance loss caused by puncturing from a cycle distribution perspective, i. e., the puncturing pattern is selected in the sense that the removed bits will break a certain number of short cycles, which significantly improves the connectivity of the TG.

B. Construction of RC-LDPC Codes Using Extension

The authors in [10] and [18] have shown that extension is another effective approach to constructing good RC-LDPC codes. We employ the idea of cycle conditioning to devise the proposed extension schemes. The proposed

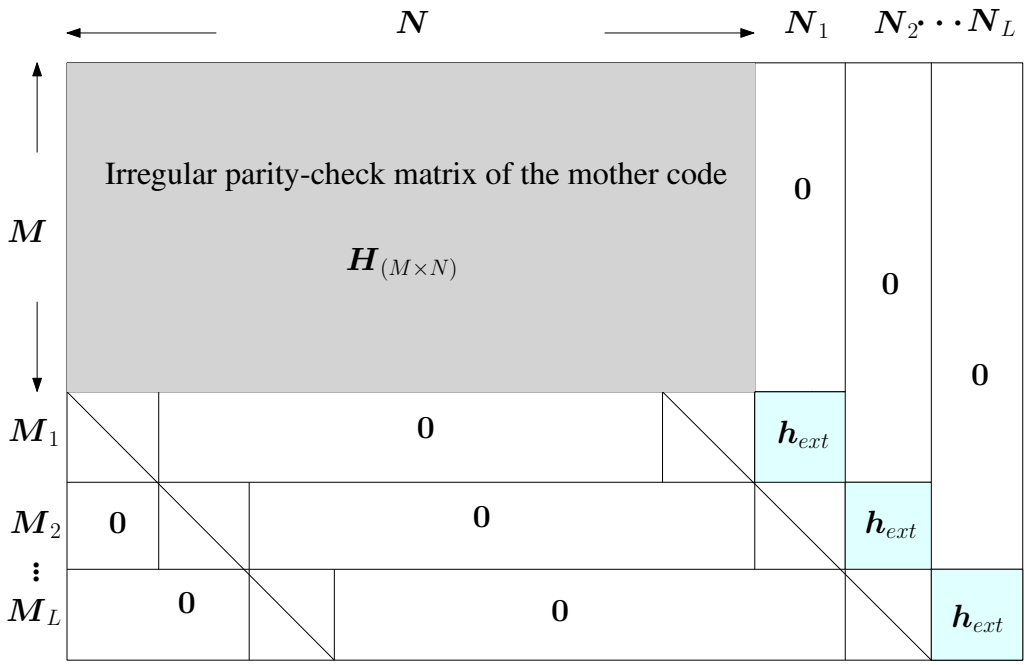

Fig. 1. To extend the parity-check matrix $\boldsymbol{H}_{(M\times N)}$ to $\boldsymbol{H}_{\mathrm{ext}(M_L\times N_L)}$ via a multi-level strategy.

extension framework is built as shown in Fig. 1, in which, starting from level 1 and running to level L, the current parity-check matrix is extended in such a way that the same number of rows and columns are added in each level. Consequently, the corresponding code rate gradually reduces. Since $M_l = N_l = B(l = 1, \ldots, L)$, the matrix $\boldsymbol{h}_{\mathrm{ext}}$, along with two accompanying identity matrices, is a $B \times B$ square matrix. Note that $\boldsymbol{h}_{\mathrm{ext}}$ is fixed from one level to the next. In Fig. 1, the areas filled by "**0**" ensure the sparsity of the extended parity-check matrix $\boldsymbol{H}_{\mathrm{ext}}$, and the existence of identity matrices guarantees a relatively uniform degree of check node distribution as well as creating sufficient dependency between the original matrix $\boldsymbol{H}$ and the extended matrix $\boldsymbol{H}_{\mathrm{ext}_l}$. Our framework is similar to that in [18], which enables fast linear-time encoding as the matrix $\boldsymbol{H}_{\mathrm{ext}}$ is always obtained in systematic form by using Gauss-Jordan elimination. Furthermore, the proposed extension schemes have two extra features: 1) possible cycles of length 4 are avoided by not inserting two identity matrices together; 2) more importantly, the submatrices $\boldsymbol{h}_{\mathrm{ext}}$ are carefully chosen with cycle conditioning for each subgraph. If $\boldsymbol{I}$ denotes the identity matrix and $\boldsymbol{G}$ denotes the generator matrix of the mother code, $\boldsymbol{G}_1, \ldots, \boldsymbol{G}_L$ are systematically transformed from the extended counterparts of $\boldsymbol{H}_{\mathrm{ext}1}, \ldots, \boldsymbol{H}_{\mathrm{ext}L}$. Similar to the puncturing model, some side information is needed at the receiver end to indicate the desired rate and corresponding parity-check matrix.

III. Proposed Puncturing Techniques

Inspired by the cycle-conditioning and the ACE metric, in this section we introduce the proposed puncturing schemes, i. e., CC-based puncturing, ACE-based puncturing, and simulation-based puncturing. The first two methods are developed using the counting cycle algorithm and the ACE metric, while the last one is based on an exhaustive search. As mentioned in the last section, all the proposed puncturing techniques can be applied offline (independently of the data transmitted), and without any side information the puncturing patterns need to be stored at both the transmitter and the receiver ends. Unlike the preliminary results reported in [24], we have replaced the cycle counting algorithm [25] with a more efficient algorithm [29], modified the puncturing order of the proposed ACE-based scheme, and employed an improved PEG code (ACE PEG) [30] as the mother code.

A. CC-Based Puncturing Scheme

The proposed counting cycle (CC)-based puncturing technique is developed based on the counting cycle algorithms [25] and [29]. The former algorithm [25] employs matrix multiplications while the latter [29] takes advantage of the message-passing nature of BP decoding. Given the same TG, we have verified that both algorithms produce similar results for counting cycles of length g and $g+2$, where g is the girth. However, the algorithm in [29] has much lower complexity ($\mathcal{O}(g|E|^2)$) than its counterpart [25] ($\mathcal{O}(gN^3)$), especially for graphs with large sizes. Given the cycle distribution, the objective is to select an ideal puncturing pattern that can break as many girth-length cycles as possible, which may reduce the performance degradation introduced by puncturing. The idea behind the proposed algorithm is inspired by the fact that the existence of short cycles creates a statistical dependency between the extrinsic messages being exchanged in the current decoding iteration, such that the extrinsic messages for the next iteration will, inaccurately, have high reliability.

According to the PEG algorithm [31], high degree nodes are placed in the leftmost positions of $\boldsymbol{H}_{(M\times N)}$ that correspond to information bits, as they provide more protection for the original data. Following this design rule we only puncture the set of variable nodes $s_j \in V_s$, where $K+1 \leq j \leq N$. Define the vector $\boldsymbol{c}_{s_j} = \{N_g, N_{g_2}, N_{g_4}\}^T$ whose element refers to the number of $g-$cycles, $(g+2)-$cycles and $(g+4)-$cycles passing through a variable node s_j. Any $s_j(K+1 \leq j \leq N)$ will be included as a punctured candidate if $N_g \neq 0$. For each candidate node, another vector $\boldsymbol{v}_{g\_s_j}$ is formed as:

$$\boldsymbol{v}_{g\_s_j} = \{v_{g\_s_0}, v_{g\_s_1}, \ldots, v_{g\_s_{N-1}}\}^T, \quad s_j \in V_c, \tag{1}$$

where the entries represent the number of cycles of length g that s_j has, and are arranged in decreasing order. Similarly, we can also define $\boldsymbol{v}_{g_2\_s_j}$ or $\boldsymbol{v}_{g_4\_s_j}$ if necessary. There are two criteria to determine the set of punctured

nodes: 1) to find variable nodes having the shortest cycles passing through; 2) to find variable nodes having more such cycles than others. In addition, we have also tried to arrange the entries of (1) in a reverse manner, i.e., we start by puncturing the variable nodes having the least number of cycles of length g. But with such a formation, the performance deteriorates dramatically. If the candidates on the $g-\text{cycles}$ are less than P, we puncture P nodes at first then arrange the remaining candidates with respect to the $(g+2)-$ and $(g+4)-\text{cycles}$. This situation rarely occurs in practice unless an unreasonable puncturing rate ρ is given. Compared to random puncturing schemes, CC-based puncturing requires more computational complexity due to the cycle counting algorithm. On the other hand, CC-based puncturing has been verified to significantly outperform random puncturing techniques [24]. Obviously, the complexity of CC-based puncturing is mainly increased by counting short cycles. It is worth noting that the practical complexity is lower than $\mathcal{O}(g|E|^2)$ since most of the time only a counting cycle of length g is required. The proposed counting cycle (CC)-based puncturing technique can be summarized as:

Step 1: given the block size N, the rate R and the degree distribution, generate the parity-check matrix of the mother code $\boldsymbol{H}_{(M\times N)}$ by using the improved PEG algorithm [30];

Step 2: for $\boldsymbol{H}_{(M\times N)}$ compute $g-\text{cycle}$, $(g+2)-\text{cycle}$ and $(g+4)-\text{cycle}$ with respect to variable nodes $s_j \in V_s$ where $(K+1 \leq j \leq N)$;

Step 3: based on the knowledge from Step 2, define the vector $\boldsymbol{c}_{s_j} = \{N_g, N_{g_2}, N_{g_4}\}^T$ for every variable node s_j. If $N_g \neq 0$, s_j is chosen as one of the punctured candidates;

Step 4: for all the candidates chosen in Step 3, define the vector $\boldsymbol{v}_{g\_s_j}$ $(s_j \in V_s)$. Puncture the first P candidates in $\boldsymbol{v}_{g\_s_j}$.

Now we illustrate how CC-based puncturing affects the cycle distribution as well as the overall performance. As for a cycle of length c, the cycle distribution is defined as (N_c, μ_c, σ_c), where N_c denotes the number of cycles of length c, while μ_c and σ_c denote the mean and standard deviation of c-length cycles with respect to the variable nodes, respectively. As an example, we use an irregular TG of code A in Section V. Table I shows the cycle distributions of the mother code and the punctured code. Applying CC-based puncturing, short cycles of length $g = 8$ are reduced by 42% while short cycles of length 10 and 12 are reduced by 30% and 3%, respectively.

TABLE I

CYCLE DISTRIBUTIONS OF CODE A BEFORE AND AFTER CC-BASED PUNCTURING

	Mother code $N = 1000$				Punctured code $N = 800$		
	N_c	μ_c	σ_c		N_c	μ_c	σ_c
$c = 8$	513	3.5	5.45	$c = 8$	295	28.1	32.17
$c = 10$	18553	148.52	143.31	$c = 10$	12853	385.53	372.31
$c = 12$	198607	2482.6	2298.3	$c = 12$	191287	4684	4558.3

Even if the number of girth-length cycles diminishes, it is worth noticing that the cycle distribution becomes less uniform after puncturing. In [25], the authors suggest that with the same girth codes with a rather uniform cycle distribution perform better than codes with a non-uniform cycle distribution. As a consequence, the proposed CC-based puncturing removes a fair number of cycles of girth length but also damages the connectivity of the TG. This fact motivated us to devise more advanced puncturing schemes, as detailed in what follows.

B. ACE-Based Puncturing Scheme

The second puncturing algorithm proposed is an improved version of the CC-based puncturing scheme, which is called ACE-based puncturing algorithm, thanks to the use of the ACE metric. ACE-based puncturing strives to remove a certain of short cycles and simultaneously maintain good graph connectivity. Since not all short cycles of the same length are equally detrimental to iterative decoding, the ACE metric [26] and ACE spectrum [27] were introduced to evaluate the effect of short cycles with a certain length in a TG. For a cycle $\mathfrak{C}$ and a corresponding set of variable nodes $V_{\mathfrak{C}}$, all the edges connected to $\mathfrak{C}$ can be categorized into three groups [27], in which $E_{\text{ext}}(V_{\mathfrak{C}})$, including extrinsic edges incident to those check nodes with a single connection to $V_{\mathfrak{C}}$, is expected to be large so that short cycles possess more singly connected extrinsic edges, which decreases the probability of cycles forming a small stopping set [8] or trapping set [32]. For short cycles of the same length, a larger ACE value indicates better connections to the rest of the graph. Here we define the average ACE value regarding a variable node s_j contained in N_g cycles of length g as:

$$\alpha_g = 1/N_g \sum_{1}^{N_g} \epsilon_{ACE}, \tag{2}$$

where α_{g2} and α_{g4} are defined with respect to the cycles of length $g+2$ and $g+4$. Moreover, for each $s_j \in V_s$ where $(K+1 \leq j \leq N)$, α_{s_j} is defined as:

$$\alpha_{s_j} = \min\{\alpha_g, \alpha_{g2}, \alpha_{g4}\}. \tag{3}$$

TABLE II

CYCLE DISTRIBUTIONS OF CODE A BEFORE AND AFTER ACE-BASED PUNCTURING

	Mother code $N = 1000$				Punctured code $N = 800$		
	N_c	μ_c	σ_c		N_c	μ_c	σ_c
$c = 8$	513	3.5	5.45	$c = 8$	402	10.88	15.7
$c = 10$	18553	148.52	143.31	$c = 10$	15236	175.4	200.37
$c = 12$	198607	2482.6	2298.3	$c = 12$	172547	2787.25	2523.44

Compared to the work reported in [24], the ACE puncturing proposed has the following three improvements: 1) the puncturing ordering is adjusted to consider the connectivity of cycles prior to their length; 2) a new code design [30] for generating the mother code makes indexing ACE values more convenient since these designs are based on the ACE metric, which lends itself naturally to the indexing of ACE values that are obtained from the design procedure; 3) the combination of a new design method and ordering leads to improved performance for both mother code and punctured code. The proposed ACE puncturing can be summarized as follows:

Step 1: given the block size N, the rate R, and the degree distributions, generate the parity-check matrix for the mother code $\boldsymbol{H}_{(M\times N)}$ by using the improved PEG [30];

Step 2: for $\boldsymbol{H}_{(M\times N)}$ compute $g-\text{cycle}$, $(g+2)-\text{cycle}$ and $(g+4)-\text{cycle}$ for the variable nodes $s_j \in V_s$ where $(K+1 \leq j \leq N)$;

Step 3: with the knowledge from Step 2, define the vector $\boldsymbol{c}_{s_j} = \{N_g, N_{g_2}, N_{g_4}\}^T$ for every variable node s_j ($s_j \in V_s$). Calculate α_{s_j} using (2) and (3);

Step 4: find the set of puncturing candidates $\boldsymbol{w} = \{\alpha_{s_0}, \alpha_{s_1}, \ldots, \alpha_{s_{N-1}}\}^T$ by sorting α_{s_j} in increasing order;

Step 5: puncture the first P candidates in $\boldsymbol{w}$.

We use Table II to illustrate the change in the cycle distribution after running the ACE puncturing scheme. Compared to the results from Table I, ACE puncturing is able to maintain a relatively uniform cycle distribution by first removing the variable nodes which get involved with longer cycles but have low ACE values. From the

decoding point of view, in a subgraph with good connectivity, the LLR of punctured bits is expected to be recovered within a few iterations, even though there might be other punctured bits in the same neighborhood. On the other hand, unlike CC puncturing, ACE puncturing does not work for regular codes since all the ACE values of variable nodes are identical. In that case, it is impossible to consider puncturing priority with the ACE metric.

C. Simulation-Based Puncturing Scheme

The third puncturing scheme proposed, denoted SIM-based puncturing, has been developed on the basis of an exhaustive search among a large number of random puncturing patterns. Then, the best puncturing pattern is determined simply by choosing the one having the best average BER performance. At the receiver end, in order to find the best pattern, we need to send a training sequence then compute the average BER values at T SNR points for each puncturing pattern. For Q possible patterns the best pattern $\boldsymbol{p}_{\text{opt}}$ is selected as:

$$\boldsymbol{p}_{\text{opt}} = \arg\min_{q} \frac{1}{rT} \sum_{i=1}^{r} \sum_{t=1}^{T} \text{BER}(\boldsymbol{p}_q), \quad q = 1, \ldots, Q. \tag{4}$$

The proposed SIM-based algorithm can be described as follows:

Step 1: given the block size N, the rate R and the degree distributions, generate the parity-check matrix of the mother code $\boldsymbol{H}_{(M\times N)}$ by using the improved PEG [30];

Step 2: for the desired rate R', randomly generate Q puncturing patterns represented by a row vector $\boldsymbol{p}_q$ where $q = 1, \ldots, Q$, in each of which P bits are randomly punctured from the encoded data;

Step 3: for each pattern in $\boldsymbol{p}_q$, send a training sequence of length $1,000$ at T SNR points then calculate BER values;

Step 4: after running r repetitions, for all Q patterns calculate an average based on accumulated BER values;

Step 5: select the best puncturing pattern $\boldsymbol{p}_{\text{opt}}$ among $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_Q$ by choosing the pattern with the minimum average BER.

From (4), it is apparent that given a desired rate R' it is possible to obtain the optimal pattern p_{opt} when all $\frac{N!}{M!(N-M)!}$ possible puncturing patterns are considered, which seems infeasible in practice. Since the quality of the

best pattern $\boldsymbol{p}_{\text{opt}}$ depends on Q, the proposed SIM-based puncturing scheme offers flexible trade-offs between the performance and the number of candidate patterns. In [24], SIM-based puncturing always outperforms CC-based puncturing and ACE-based puncturing. Nevertheless, with the additional improvement, ACE-based puncturing is able to provide at least comparable performance to SIM-based puncturing, even when we increase Q to 500.

IV. Proposed Extension Techniques

In this section, we investigate another approach to generating RC-LDPC codes which employs extension techniques. In particular, two novel schemes are presented along with a detailed explanation about the construction of RC-LDPC codes. An extension framework introduced in [18] is exploited which enables fast encoding and off-line operation for the proposed extension schemes. To refine the techniques described in [28], we replace the cycle counting algorithm [25] by a more efficient algorithm [29], further develop the design procedure, and utilize the improved PEG code (ACE PEG) [30] as the mother code.

A. Counting-cycle based extension

The first extension scheme proposed is the counting cycle (CC)-based extension which employs an algorithm for counting short cycles in order to select an extension submatrix $\boldsymbol{h_{ext}}$ among S candidates. We set the parameter S to the number of desired extension rates, e.g. $S = 3$ if $R = 5/10$, $R_1 = 5/12$ and $R_2 = 5/13$, where R is the rate of the mother code. In this case, three distinct submatrix candidates are constructed using the ACE PEG algorithm [30] with different degree distributions, which are derived via density evolution (DE) [6] and provided with the maximum variable nodes' degree $d_{v_{\max}}$ and check nodes' degree $d_{c_{\max}}$. As per the extended part in Fig. 1, the $B \times B$ submatrix $\boldsymbol{h}_{\text{ext}}$ is very likely to have many short cycles while the rest of the extended part can be proved cycle free. Define $g_{h(s)}(s = 1, 2, \ldots, S)$ as the local girths for each candidate submatrix, and $N_{g(s)}$ as the number of cycles of length $g_{h(s)}$ corresponding to each $\boldsymbol{h}_s(s = 1, 2, \ldots, S)$. After running the counting cycle algorithm [29], we select the candidate submatrix with the largest g_h and the smallest N_g as $\boldsymbol{h}_{ext}$. As a result, the CC-based extension scheme maximizes the local girth g_h of $\boldsymbol{h}_{ext}$, and the selected $\boldsymbol{h}_{ext}$ has the smallest number of length-g_h cycles. The algorithm flow of the proposed CC-based extension is summarized as follows:

Step 1: given the parity-check matrix $\boldsymbol{H}_{(M\times N)}$ and the desired code rates $R_1, R_2, ..., R_L$, determine the number of extension levels L which ensures $M_l = N_l = B(l = 1, \ldots, L)$;

Step 2: set $S = L + 1$, given $d_{v_{\max}}$ and $d_{c_{\max}}$ derive S degree distributions according to DE;

Step 3: based on Step 2, construct S candidates for $B \times B$ submatrices by using the improved PEG algorithm [30];

Step 4: for each submatrix candidate compute the $g_{h(s)}$ and $N_{g(s)}$ of each subgraph;

Step 5: choose the candidate with the largest $g_{h(s)}$ and the smallest $N_{g(s)}$ as $\boldsymbol{h}_{\text{ext}}$;

Step 6: for $1 \leq l \leq l$ gradually extend $\boldsymbol{H}_{(M\times N)}$ to $\boldsymbol{H}_{\text{ext}(L)}$ by adding zero entries, identity matrices, and $\boldsymbol{h}_{\text{ext}}$ as in Fig. 1.

B. ACE-based extension

The second scheme proposed is an ACE-based extension scheme. Unlike CC-based extension, the candidate submatrix with the largest $\alpha(g_h)$ will be selected as the $\boldsymbol{h}_{\text{ext}}$, where $\alpha(g_{h(s)})$ is the average ACE spectrum with respect to $N_{g(s)}(s = 1, 2, \ldots, S)$, as defined in (2). Similar to ACE-based puncturing, it is straightforward to compute $\alpha(g_h)$ if the submatrix candidates are created using the ACE PEG algorithm [30]. The proposed ACE-based extension is described in the following:

Step 1: given the parity-check matrix $\boldsymbol{H}_{(M\times N)}$ and the desired code rates $R_1, R_2, ..., R_L$, determine the number of extension levels L which ensures $M_l = N_l = B(l = 1, \ldots, L)$;

Step 2: set $S = L + 1$, given $d_{v_{\max}}$ and $d_{c_{\max}}$ derive S degree distributions according to DE;

Step 3: based on Step 2, construct S candidates for $B \times B$ submatrices by using the improved PEG algorithm [30];

Step 4: for each submatrix candidate compute $g_{h(s)}$ and $N_{g(s)}$ for each subgraph;

Step 5: provided with $g_{h(s)}$ and $N_{g(s)}$ in Step 3, calculate $\alpha(g_{h(s)})$;

Step 5: choose the candidate with the largest $\alpha(g_{h(s)})$ as $\boldsymbol{h}_{\text{ext}}$;

Step 6: for $1 \leq l \leq l$ gradually extend $\boldsymbol{H}_{(M\times N)}$ to $\boldsymbol{H}_{\text{ext}(L)}$ by adding zero entries, identity matrices, and $\boldsymbol{h}_{\text{ext}}$ as in Fig. 1.

V. Simulation Results

In this section, we present numerical results corresponding to the evaluation of the three proposed puncturing schemes and the two proposed extension algorithms. We then carry out comparisons of the puncturing and extension schemes at different rates, which shows that the former performs better at higher rates while the latter is superior at lower rates. In all the simulations, mother codes are finite-length irregular LDPC codes generated by the improved PEG algorithm [30]. Code A has block length of $N = 1000$, code rate $R = 0.5$ and degree distributions $\mu(x) = 0.21 \times x^5 + 0.25 \times x^3 + 0.25 \times x^2 + 0.29 \times x$, $\nu(x) = x^5$. Code B has a block length equal to $N = 2,000$, code rate $R = 0.4$ and degree distributions $\mu(x) = 0.45 \times x^9 + 0.26 \times x^2 + 0.29 \times x$, $\nu(x) = x^5$. It should be noted that other codes could be used in the comparison and will result in similar performance hierarchy among the proposed and existing RC-LDPC codes. The decoder applies the standard BP algorithm in the logarithm domain.

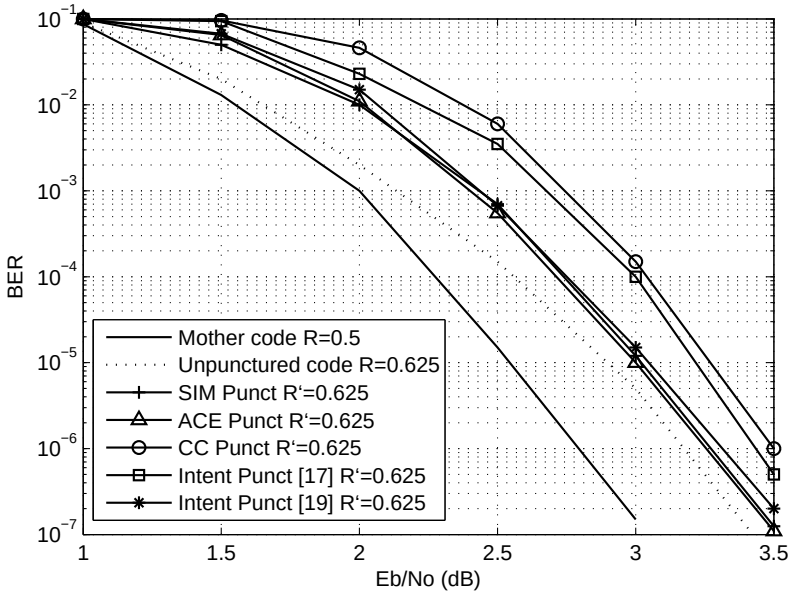

Fig. 2. Comparisons of the proposed puncturing schemes with existing puncturing schemes [17] and [19] with respect to BER performance, where code A is the mother code.

In order to assess the proposed puncturing schemes, we first choose code A as the mother code, then compare the performance to that of the puncturing techniques reported in [17] and [19]. In this scenario, the decoder runs a

maximum of 60 decoding iterations. Fig. 2 shows a BER performance comparison of the three proposed schemes with the existing methods [17] and [19], in which the resulting rate R' is 0.625 and the puncturing rate $\rho = 0.2$, such that 200 bits are punctured prior to transmission. It is clear that the proposed ACE-based puncturing outperforms the existing methods as well as CC-based puncturing and the SIM-based puncturing technique. For comparison purposes, we also include unpunctured irregular LDPC code, with $N = 800$, $R = 0.625$, which has the same degree distributions as the mother code A. Notice that the performance gap between ACE-based puncturing and the unpunctured code is less than 0.2 dB at BER of 10^{-6}. Further results over a range of puncturing rates are shown in Fig. 3, in which CC-based puncturing outperforms [17] by 0.25 dB at BER 10^{-5} at rate equal to 0.714, whereas the proposed ACE puncturing scheme outperforms the puncturing schemes reported in [17] and [19] for rates 0.526 and 0.714.

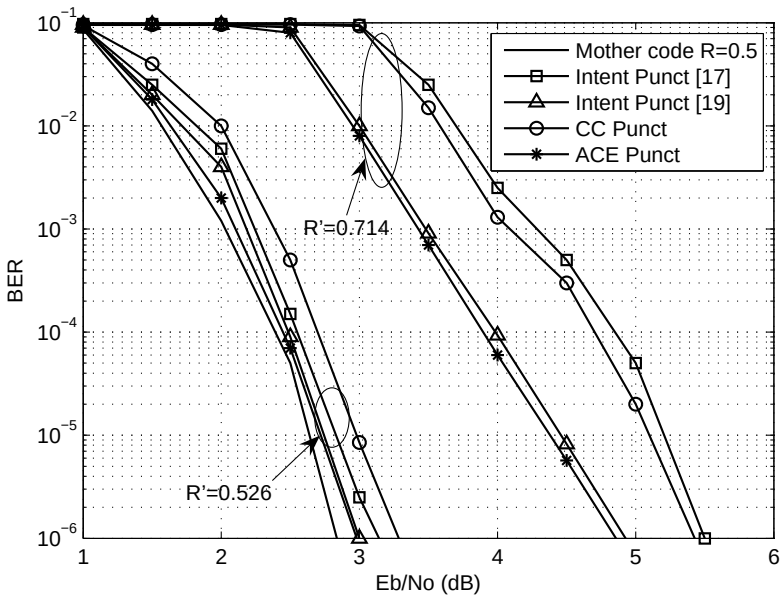

Fig. 3. Comparison of the proposed CC-based and ACE puncturing with existing puncturing schemes [17] and [19] at different resulting rates. R is the rate of the mother code, A, and R' is the resulting rate.

To evaluate the performance of the proposed puncturing techniques in the type-II ARQ system, in Fig. 4 we compare the proposed ACE-based puncturing with the puncturing scheme in [14] in terms of FER performance. Specifically, we assess the throughput for a type-II hybrid ARQ system, which performs only error detection for the first transmitted message block and only employs LDPC codes for retransmitted blocks. In this case, code B is used as the mother code and the maximum number of decoding iterations is increased to 200. From Fig. 4, we see that the puncturing scheme of [14] works better in the low SNR region but is outperformed by ACE-based

puncturing in the high SNR region. The advantages of ACE-based puncturing over the method reported in [14] are as follows:

- the method [14] is easy to implement in hardware, thanks to a specific code structure. But it usually has to compromise on the optimal degree distribution so as to fulfil the design requirement that may affect the performance. ACE-based puncturing is a more general technique, and can be applied to any irregular mother code;

- the ACE-based method aims to obtain an ACE spectrum via puncturing such that graph connectivity is optimized for each rate computed. Due to the design nature of [14], one has to maximize the number of degree 2 variable nodes whose ACE value is 0. Once a cycle is formed, that will severely reduce performance, especially in the high SNR region;

- the best puncturing performance for [14] results from $N_v(2) = M - 1$ where $N_v(2)$ is the number of degree 2 nodes. However, this requirement is difficult to realize for a mother code with a low rate;

- the ACE scheme is expected to achieve any puncturing rate without limitations, while method [14] always has a puncturing threshold of $R_H = K/(N - N_v(2))$, above which one can only use random puncturing to achieve a higher rate.

As for the extension schemes, we compare the proposed extension techniques with the technique reported in [18]. In the following simulations, we use the mother code C constructed by the improved PEG algorithm [30] with block length $N = 1000$, rate $R = 5/10$ and degree distributions $\mu(x) = 0.438x^6 + 0.416x^2 + 0.315x$ and $\nu(x) = 0.561x^6 + 0.438x^5$. The decoder terminates after a maximum of 100 iterations.

In Fig. 5, we compare the proposed CC-based and ACE-based extension algorithms with the existing extension method [18]. From rates $5/10$ to $5/14$, the extension operates at three levels and 100 bits are added per level. Notice that all the degree distributions are constrained by $d_{v_{\max}} \leq 7$. In Fig. 5, both proposed schemes outperform the existing method at different rates, and the performance gap increases as more parity bits are inserted.

Fig. 6 shows that the proposed extension schemes outperform the proposed puncturing schemes at low rates. Both CC-based and ACE-based extensions originates from the mother code, C, while all punctured schemes are from the mother code $(M = 500, N = 1400, R = 5/14)$. On the other hand, Fig. 7 shows that at a high rate, $R' = 5/8$ of the punctured codes (mother code C) offer better performance as compared to the proposed extension

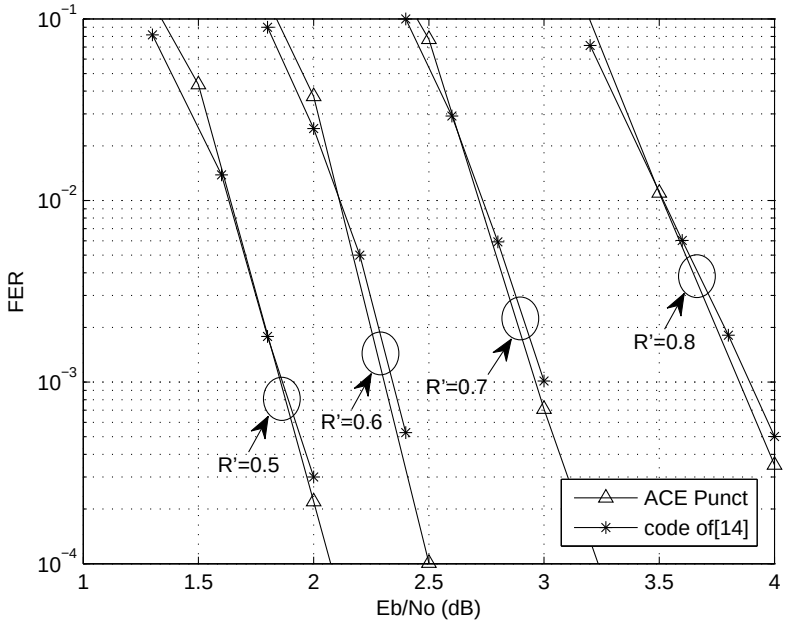

Fig. 4. The comparison of the puncturing FER performance between ACE-based puncturing and [14]. The puncturing rates R' are 0.5, 0.6, 0.7 and 0.8. We use the mother code B with block length $N = 2,000$ and rate $R = 0.4$.

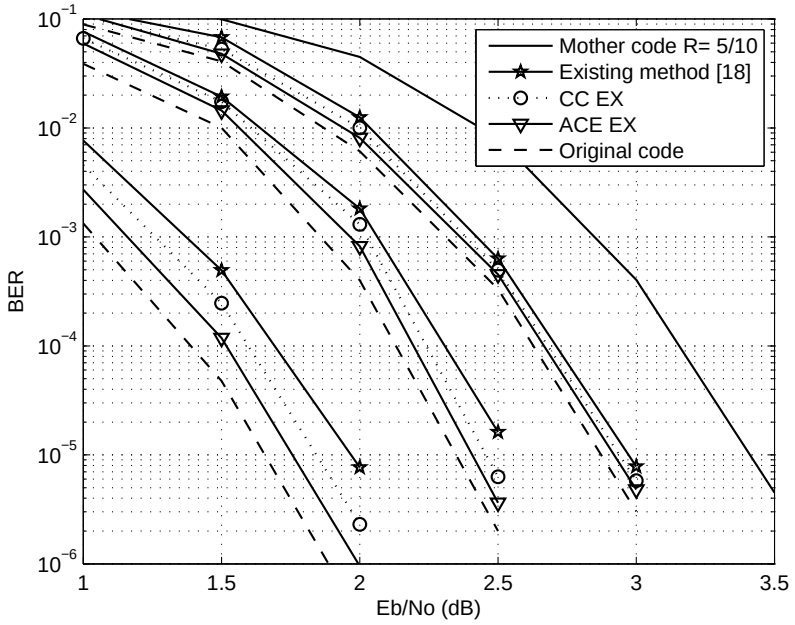

Fig. 5. Comparison of the proposed extension schemes with another existing scheme [18] at different rates for irregular PEG LDPC codes. The mother code corresponds to the rightmost curve with $N_0 = 1,000$ and $R = 5/10$. For other codes the rates from left to right are $5/14$, $5/13$, $5/12$.

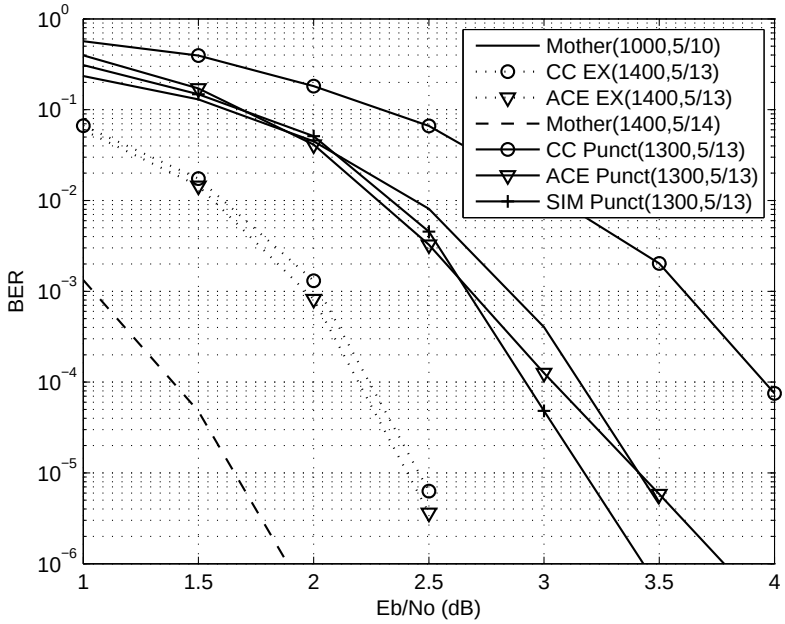

Fig. 6. Comparison of the proposed extension schemes with the proposed puncturing schemes at a low rate, $5/13$, for irregular LDPC code.

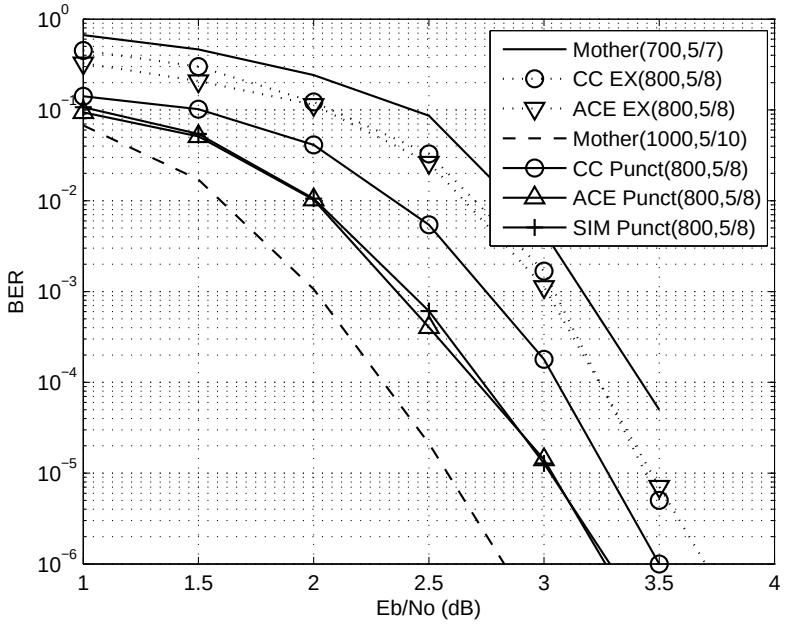

Fig. 7. Comparison of the proposed extension schemes with the proposed puncturing schemes at a high rate, $5/8$, for irregular LDPC code.

codes whose mother code has parameters $M = 500, N = 700, R = 5/7$. To illustrate the overall performance of the proposed RC-LDPC codes, we compare the proposed RC-LDPC codes with the existing RC-LDPC family in the system throughput [33] as shown in Fig. 8, in which E_b in previous figures is replaced by E_s, the average

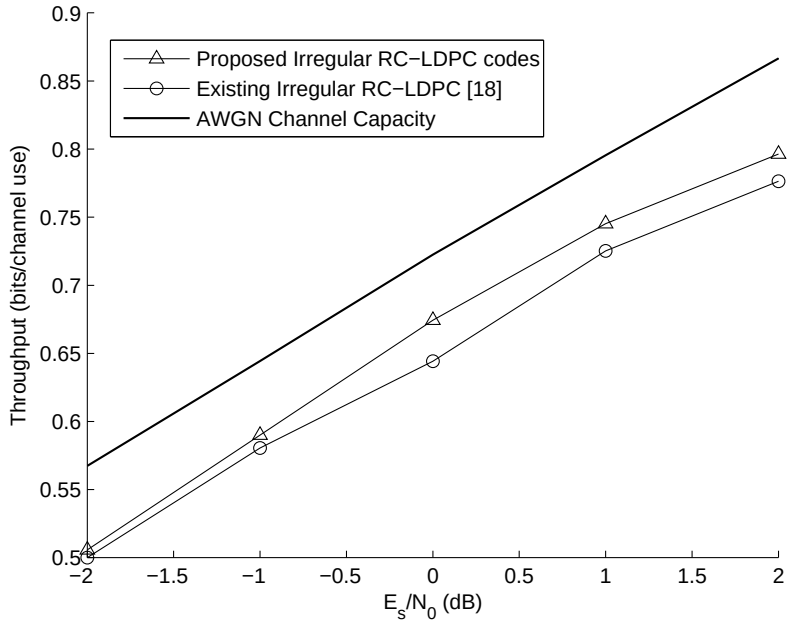

Fig. 8. Comparison of the proposed irregular RC-LDPC codes with the irregular RC-LDPC code [18] in system throughput. The capacity of AWGN channel is also included.

energy per transmitted symbol. Fig. 8 shows that the proposed RC-LDPC codes are superior to the RC codes of [18] and can approach channel capacity.

VI. Conclusion

In this paper, we have investigated irregular RC-LDPC codes from both puncturing and extension perspectives. By applying counting cycle algorithms, the ACE spectrum and exhaustive searches, three puncturing schemes as well as two extension schemes have been devised. All proposed schemes manage to achieve various resulting rates, and at the same time provide better performance than existing methods. Simulation results have shown that the proposed extension designs are suitable for creating RC-LDPC with low rates $(R < 0.5)$ and ACE-based extension performs better than the CC-based extension. On the other hand, the puncturing designs are preferred for codes with high rates. With the additional improvement, the ACE puncturing has been proven to generate the optimal puncturing pattern and slightly outperform simulation-based puncturing. As a consequence, taking advantage of a combined puncturing/extension strategy, we have devised algorithms to generate RC-LDPC codes with a wide range of rates $(0.1 < R < 0.9)$.

REFERENCES

[1] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Transactions on Communications*, vol. 36, no. 4, pp. 389-400, Apr. 1988.

[2] S. Lin and P. S. Yu, "A hybrid ARQ scheme with parity retransmission for error control of satellite channels," *IEEE Transactions on Communications*, vol. 30, no. 7, pp. 1701-1719, Jul. 1982.

[3] D. N. Rowitch and L. B. Milstein, "On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes", *IEEE Transactions on Communications*, vol.48, no.6, pp.948-959, Jun 2000.

[4] R. G. Gallager, "Low-density parity check codes," *IRE Transactions on Information Theory*, vol. 39, no. 1, pp. 37-45, Jan. 1962.

[5] D. J. C. Mackay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457-458, Mar. 1997.

[6] T. J. Richardson, M. A. Shokrollahi and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619 - 637, Feb. 2001.

[7] S.-Y. Chung, G. D. Forney, T. J. Richardson and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters*, vol.5, no.2, pp.58-60, Feb 2001.

[8] C. Di, D. Proietti, I. E. Telatar, T. Richardson and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 48, pp. 1570-1579, Jun. 2002.

[9] M. Yang, W. E. Ryan and Y. Li, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Transactions on Communications*, vol.52, no.4, pp. 564- 571, April 2004.

[10] J. Li and K. R. Narayanan, "Rate-Compatible Low Density Parity Check (RC-LDPC) Codes for Capacity-Approaching ARQ Schemes in Packet Data Communications," *Proc. of International Conference on on Communications, Internet and Information Technology (CIIT)*, US Virgin Islands, pp. 201-206, Nov. 2002.

[11] J. Ha, J. Kim and S.W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol.50, no.11, pp. 2824- 2836, Nov. 2004.

[12] J. Ha, J. Kim, D. Klinc and S. W. Mclaughlin, "Rate-compatible punctured low-density parity-check codes with short block length", *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 728-738, Feb. 2006.

[13] H. Y. Park, J. W. Kang, K. S. Kim and K. C. Whang, "Efficient puncturing method for rate-compatible low-density parity-check codes", *IEEE Transactions on Wireless Communications*, pp. 3914-3919, Nov. 2007.

[14] J. Kim, A. Ramamoorthy and S. Mclaughlin, "The design of efficiently-encodable rate-compatible LDPC codes," *IEEE Transactions on Communications*, vol.57, no.2, pp.365-375, February 2009.

[15] M. El-khamy, J. Hou and N. Bhushan, "Design of rate-compatible structured LDPC codes for hybrid ARQ applications," *IEEE Journal on Selected Areas in Communications*, pp. 965-973, Aug. 2009.

[16] H. Pishro-Nik and F. Fekri, "Result on punctured low-density parity-check codes and improved iterative decoding techniques," in *Proc. 2004 IEEE Information Theory Workshop*, pp. 24-29, San Antonio, TX, Oct. 2004.

[17] B. N. Vellambi and F. Fekri, "Finite-length rate-compatible LDPC codes: a novel puncturing scheme," *IEEE Transactions on Communications*, vol. 57, no. 2, pp. 297-301, Feb. 2009.

[18] M. R. Yazdani and A. H. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *IEEE Communications Letters*, vol. 8, no. 3, pp. 159-161, Mar. 2004.

[19] R. Asvadi and A. H. Banihashemi, "A Rate-Compatible Puncturing Scheme for Finite-Length LDPC Codes," *IEEE Communications Letters*, vol.17, no.1, pp.147-150, January 2013.

[20] T. V. Nguyen and A. Nosratinia, "Rate-Compatible Short-Length Protograph LDPC Codes," *IEEE Communications Letters*, vol. 17, no. 5, pp. 948-951, May 2013.

[21] H. Zhou, D. G. M. Mitchell, N. Goertz and D.J. Costello, Jr., "Robust Rate-Compatible Punctured LDPC Convolutional Codes," *IEEE Transactions on Communications*, vol.61, no.11, pp. 4428-4439, November 2013.

[22] S. I. Park, Y. Wu, H. M. Kim, N. Hur and J. Kim, "Raptor-Like Rate Compatible LDPC Codes and Their Puncturing Performance for the Cloud Transmission System," *IEEE Transactions on Broadcasting*, vol.60, no.2, pp.239-245, June 2014.

[23] H. Zeineddine, L. M. A. Jalloul and M. M. Mansour, "Hardware-Oriented Construction of a Family of Rate-Compatible Raptor Codes," *IEEE Communications Letters*, vol.18, no.7, pp.1131-1134, July 2014.

[24] J. Liu and R.C. de Lamare, "Novel intentional puncturing schemes for finite-length irregular LDPC codes," *17th International Conference on Digital Signal Processing (DSP)*, vol., no., pp.1-6, 6-8 July 2011.

[25] T. R. Halford and K. M. Chugg, "An algorithm for counting short cycles in bipartite graph," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 287-292, Jan. 2006.

[26] T. Tao, C.R. Jones, J.D. Villasenor and R.D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Transactions on Communications*,, vol.52, no.8, pp. 1242- 1247, Aug. 2004.

[27] D. Vukobravic and V. Senk, "Evaluation and design of irregular LDPC codes using ACE spectrum," *IEEE Transactions on Communications*, vol. 57, no. 8, pp. 2272 - 2279, Aug. 2009.

[28] J. Liu and R.C. de Lamare, "Finite-length rate-compatible LDPC codes based on extension techniques," *8th International Symposium on Wireless Communication Systems (ISWCS)*, vol., no., pp.41-45, 6-9 Nov. 2011.

[29] M. Karimi and A.H. Banihashemi, "A message-passing algorithm for counting short cycles in a graph," *IEEE Information Theory Workshop (ITW)*, vol., no., pp.1-5, 6-8 Jan. 2010.

[30] H. Xiao and A.H. Banihashemi, "Improved progressive-edge-growth (PEG) construction of irregular LDPC codes," *IEEE Communications Letters*, vol.8, no.12, pp. 715- 717, Dec. 2004.

[31] Y. Hu, E. Eleftheriou and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graph," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386-398, Jan. 2005.

[32] T. J. Richardson, "Error Floor of LDPC Codes," in *Proc. 41st Annu. Allerton Conf. on Communication, Control, and Computing*, Urbana-Champaign, Oct. 2003, pp. 1426-1435.

[33] R. Mantha and F.R. Kschischang, "A capacity-approaching hybrid ARQ scheme using turbo codes", *Proc. of IEEE Global Telecommunications Conference*, vol.5, pp. 2341- 2345, 1999.