Low-Density Parity-Check Codes with Rate Adaptability

Jeongseok Ha · Demjan Klinc

Over time varying channels, channel codes must be flexible in terms of their code rate to efficiently adapt to the channel state and provide the required QoS (Quality of Service). Moreover, they must provide strong error correction capability. Low-Density Parity-Check (LDPC) codes are considered as candidates for error-correcting codes in the future generation communication systems due to their powerful error-correcting capability and tractable mathematical structure. Thus, rate-adaptable LDPC codes have been subject to extensive studies in the recent years. In this paper, we survey the existing literature dealing with rate-adaptable LDPC codes in which puncturing and parity-check equation combining techniques are used for increasing the code rate, while extending and shortening are utilized for decreasing it. It is desirable that the range of code rates be as broad as possible. Unfortunately, as the range of code rates gets broader, the performance of LDPC codes tends to deteriorate. To our best knowledge, it is not possible to efficiently cover a broad range of code rates with a single technique. Thus, combinations of the aforementioned techniques are needed. We show an example of a practical communication system that uses such a combination to provide a very broad range of available code rates.

Keywords: Error-correction codes, Low-density parity-check codes, Rate-adaptability, Puncturing, Shortening, Extending, parity-check equation combining

I. INTRODUCTION

To ensure the maximum transmission data rate, the code rate of a channel code should be as high as possible, given the channel state, given the fidelity of the signal after decoding satisfies some predefined quality of service requirements. In some applications it is possible to fix the code rate since its determining factors remain static during communication. In other applications, these factors are time-varying. For example, on wireless channels, the channel state varies with time and the code rate must be adjusted to cope with the worst case scenario. That is, the code rate must be low enough to protect the message even when the channel is in the worst possible state. However, as the channel state improves the low code rate offers more protection than actually needed, causing inefficient use of the available bandwidth. To overcome this problem the channel code used by the system needs to be adaptable in terms of the code rate.

Our main focus in this paper is to introduce the most popular techniques for changing the code rate of low-density parity-check (LDPC) codes on the fly. We divide these techniques into two groups: the ones for 1) increasing and 2) decreasing the code rate. The former includes puncturing [Hagenauer88, Li02, Ha04, Ha06, Yue06] [Yazdani04, Klinc05] and parity-check equation combining [Casado04, Casado05]. Puncturing in essence means that some parity bits of a codeword are deleted and hence not transmitted. As a result the number of transmitted parity bits is reduced and the code rate becomes higher as compared to the original code -- in this paper we refer to the original code as mother code. The
parity-check equation combining, on the other hand, reduces the number of parity-check equations by combining the parity-check equations of the mother code which, again, results in a smaller number of parity bits and, in effect, a higher code rate. The main difference between puncturing and parity-check equation combining is in the block length. The resulting block length of a codeword after puncturing varies, since for a given message length the number of transmitted parity bits depends on the number of punctured parity bits. On the other hand, with parity-check equation combining, it is the message length that varies in accordance with the reduced number of parities, while the block length, which is the sum of all message and parity bits, remains unchanged.

The second part of this paper covers techniques for decreasing code rate which include extending [Li02] [Yue06] [Yazdani04] and [Tian04] [Tian05] shortening. In the extending, additional parity bits are generated for a given mother code. Thus, the resulting channel code has a lower code rate and better error correcting capability. In contrast, shortening will set some of message bits to known values such that they need not be transmitted at all. The message length of the shortened code thus becomes smaller, while the number of parity bits remains the same-the code rate becomes lower.

Finally, we demonstrate on a particular example how a communication system providing a wide range of possible code rates can be designed by combining some of the introduced techniques.

II. INCREASING THE CODE RATE

An effective way to realize rate adaptability is to use a single code and puncture it in a rate-compatible fashion. Such codes are commonly referred to as rate-compatible punctured codes (RCPC) [Hagenauer88] [Mandelbaum74] [Cain79]. Note that the locations of the punctured symbols must be known to the receiver/receivers. Since the decoder for the lowest rate (the mother code) is compatible with ones for the higher rates, RCPC provide rate adaptability without additional cost in complexity. Moreover, RCPC make it possible to progressively transmit redundancies in conjunction with the automatic repeat request (ARQ) protocol [Mandelbaum74], [Lugand89]. Let us start by introducing the capacity-approaching punctured LDPC codes presented in [Ha04].

1. Puncturing

1.1. A Capacity-Approaching Punctured LDPC Codes

We present a way to puncture LDPC codes which does not disturb the optimality of the mother code, and where the resulting punctured codes maintain threshold optimality across a range of rates. In [Ha04], the authors focus mainly on asymptotic thresholds of the punctured LDPC codes instead of practical issues such as code performance at short block lengths, number of iterations to achieve a saturated performance, and finite precision effects due to quantization.

The puncturing problem can be set up as follows. Variable nodes of a bipartite graph can be grouped in accordance with their edge degrees. Thus, all coded symbols with an edge degree \( j \) belong to a group denoted by \( G_j \), where \( 2 \leq j \leq d_{f} \). A proportion \( \pi_j^{(0)} \) of the symbols in \( G_j \) is randomly punctured, where \( \pi_j^{(0)} \) is determined by optimization. The total puncturing fraction \( \rho_j^{(0)} \) is defined as \( \rho_j^{(0)} = \left( \text{the number of punctured variable nodes} \right) / \left( \text{the number of variable nodes} \right) \). Also, the distribution pair \( (\lambda(x), \rho(x)) \) is extended to include a puncturing distribution, so we have \( (\lambda(x), \rho(x), \pi_j^{(0)}(x)) \) where

\[
\pi_j^{(0)}(x) = \sum_{j=2}^{d_{f}} \pi_j^{(0)} x^{j-2}
\]

and \( 0 \leq \pi_j^{(0)} \leq 1 \). Now the puncturing fraction \( \rho_j^{(0)} \) can be expressed as

\[
\rho_j^{(0)} = \frac{\sum_{j=2}^{d_{f}} \pi_j^{(0)} \lambda_j}{\sum_{j=2}^{d_{f}} \lambda_j} = \frac{d_{f}}{\sum_{j=2}^{d_{f}} \lambda_j} \pi_j^{(0)}
\]

The rate of punctured LDPC codes specified by a three-tuple distribution \( (\lambda(x), \rho(x), \pi_j^{(0)}(x)) \) is given by \( r(\lambda, \rho, \pi_j^{(0)}) = r(\lambda, \rho) (1 - \rho_j^{(0)}) \), where \( r(\lambda, \rho) \) is the code rate of the mother code.

Figure 1 depicts the proposed puncturing scheme. The design rule for punctured LDPC codes is to optimize (minimize) the SNR threshold for a given puncturing fraction \( \rho_j^{(0)} \) by changing the puncturing proportions \( \pi_j^{(0)} \) s for all \( j \).
To evaluate the performance, punctured LDPC codes of block length 131072 were implemented at puncturing fractions ranging from 0.00 to 0.45 in the steps of 0.05, which corresponds to code rates from 0.5 (the mother code) to 0.95. The $E_v/N_o$ thresholds of the punctured LPDC codes were evaluated and compared with their asymptotic $E_v/N_o$ thresholds in Figure 2. At the maximum rate of 0.95 the punctured LDPC code requires an additional 0.3 dB of $E_v/N_o$ over that of the mother code.

The work in [Ha04] proves the existence of good puncturing distributions for LDPC codes and proposes a simple but efficient design rule. However, the main focus is set on the theoretical aspects of the problem. Namely, the design rule is based on the density evolution technique which assumes infinite block lengths. In the following we show how the puncturing problem can be efficiently solved for LDPC codes of short block lengths.

1.2. Punctured LDPC Codes at Short Block Lengths

A very efficient design rule for punctured LDPC codes
at short block lengths was proposed by Ha et al. in [Ha06]. It is based on the level of recoverability of the punctured nodes.

In general, a punctured node will be recovered with a reliable message when it has 1) more neighboring (check) nodes, and 2) each of the check nodes has reliable neighbors (variable nodes), not counting the punctured node in question. To clarify this idea further, imagine a punctured variable node. In the first iteration that node will receive nonzero messages from all neighboring check nodes whose other neighboring variable nodes are all unpunctured (see an example in Figure 3). This event, when a punctured node receives at least one non-zero message from its neighboring check nodes, is called recovery by analogy with the one over binary-erasure channels. The punctured node in the preceding example will be called one-step-recoverable (1-SR) since it is recovered in the first iteration. The recovered 1-SR nodes and unpunctured nodes will help recover some of the remaining punctured nodes in the second iteration, and so on. In general, the punctured nodes recovered in the $k$-th iteration are called $k$-SR nodes. An example of a 3-SR node is depicted in Figure 4. According to the results in [Ha06] it can be assumed that the more iterations a punctured node needs for its recovery, the less statistically reliable the recovery message is. Thus, it is preferable to
Figure 5. Comparison between the intentional (filled) and random (unfilled) puncturing of an LDPC code at block length 1024: code rates are 0.5, 0.6, 0.7, and 0.8 from the left to the right, the puncturing distributions are from Intentional and Random 2 in Table 1 and the BERs of the half rate mother code are represented with the diamonds.

Table 1. Group distribution of the intentional puncturing and the random puncturing of a regular LDPC code with $\lambda(x) = x^2$ and $\rho(x) = x^6$ at a block length of 1024: the largest code rate is 0.8

<table>
<thead>
<tr>
<th>Group</th>
<th>$G_0$</th>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_3$</th>
<th>$G_4$</th>
<th>$G_5$</th>
<th>$G_6$</th>
<th>$G_7$</th>
<th>$G_8$</th>
<th>$G_9$</th>
<th>$G_{10}$</th>
<th>$G_{11}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intentional</td>
<td>640</td>
<td>294</td>
<td>78</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Random 2</td>
<td>640</td>
<td>100</td>
<td>50</td>
<td>38</td>
<td>36</td>
<td>26</td>
<td>26</td>
<td>28</td>
<td>27</td>
<td>30</td>
<td>28</td>
<td>19</td>
</tr>
</tbody>
</table>

puncture nodes such that they will be recovered as fast as possible, which results not only in less required iterations for decoding but also in a better performance at a given code rate.

Based on this, the design rule can be defined as follows. First, the group containing 1-SR nodes, denoted by $G_1$, should be maximized. Subsequently, the group with 2-SR nodes $G_2$ should be maximized, and so on. The groups $G_1, G_2, ..., G_k$ are used to determine the puncturing patterns for various rates in a rate-compatible fashion. Namely, the variable nodes from $G_1$ are punctured first, followed by the nodes from $G_2$, etc. In this manner a set of RCPC can easily be derived from a given mother code.

The authors propose two greedy algorithms (the details are given in [Ha06]): grouping and sorting. The grouping algorithm distributes all variable nodes into groups $G_1, G_2, ..., G_k$, while the sorting algorithm specifies an order in which the nodes should be punctured within each group such that the best performance is achieved.

The performance of the proposed algorithm is evaluated and compared with the conventional puncturing where punctured nodes are chosen randomly. According to [Ha06] the conventional and the proposed puncturing schemes are called random and intentional puncturing, respectively.

Although the algorithms are applicable to both irregular and regular LDPC codes, we present performance comparison between intentionally and randomly punctured LDPC codes with a regular mother LDPC code. The half rate mother LDPC code is at block length 1024 and its degree distribution pair is $\lambda(x) = x^2$ and $\rho(x) = x^6$. The punctured LDPC codes are evaluated at code rates 0.5, 0.6, 0.7 and 0.8. The intentional puncturing clearly outperforms the random puncturing at all rates, where the performance improvement grows as the rate increases. At the rate 0.8 and the BER of $10^{-5}$, the intentionally punctured LDPC code outperforms the randomly punctured one by 3dB of $E_b/N_0$.

Very recently, a new family of LDPC codes, called efficiently encodable rate-compatible (c2RC) codes, was introduced in [Kim06]. These codes provide linear encoding and very good puncturing patterns, based on the above described ideas.

2. Parity-Check Equation Combining Technique

Puncturing has turned out to be an efficient way to
increase the code rate, but the block length of the punctured codes is smaller due to the punctured bits. As a result, the performance of punctured LDPC codes deteriorates as the number of punctured bits grows higher. We will see later that something similar happens to shortened codes.

In an effort to combat this problem Casado et al. [Casado04] proposed a parity-check equation combining technique in which rows (parity-check equations) of a parity-check matrix are combined and the resulting parity-check matrix has a smaller number of rows. At the same time the number of columns, which is the sum of parity and message lengths, and therewith the code block length remain the same.

Since the sum of parity and message bits is fixed, the reduced number of parity bits results in a higher number of message bits, which makes the code rate lower.

As an example, a parity-check matrix of a half rate LDPC code, \( \mathbf{H} \), is given by (1) and the corresponding Tanner graph is depicted in Figure 6. By combining the rows of \( \mathbf{H} \), a rate 3/4 LDPC code \( \mathbf{H}^C \) can be constructed as shown in (2). Specifically, \( \mathbf{H}_1^C = \mathbf{H}_1 + \mathbf{H}_3, \mathbf{H}_2^C = \mathbf{H}_2 + \mathbf{H}_4 \), and \( \mathbf{H}_3^C = \mathbf{H}_3 + \mathbf{H}_6 \).

\[
\mathbf{H} = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\mathbf{H}^C = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

where \( \mathbf{H}_i^C \) and \( \mathbf{H}_j \) are the \( i \)th and the \( j \)th rows of \( \mathbf{H}^C \) and \( \mathbf{H} \), respectively.

Note that \( \mathbf{H}^C \) in (2) is just one of many possible ways of constructing rate 3/4 LDPC codes from \( \mathbf{H} \).

Obviously, other code rates can be obtained by combining a different number of rows of \( \mathbf{H} \).

\[
\mathbf{H}^C = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
(\mathbf{H}_1^C \mathbf{H}_2^C \mathbf{H}_3^C)^T
\]

where \( \mathbf{H}_j^C \) is on the \( j \)th row of \( \mathbf{H}^C \).

2.1. Strict Row Combining (SRC)

In the preceding example, the combined rows do not have common nonzero terms in the columns, which prevents the nullifying of nonzero terms due to the combining. That is, the combining neither deletes nor introduces edges to the combined parity-check matrix. Such a combining scheme is called Strict Row Combining (SRC). If the selections of rows to be combined are obtained in accordance with SRC, the variable node degree distribution and the code length remain unchanged, whereas the check node degree distribution becomes different from that of the mother code.

LDPC codes obtained by SRC have a few problems. First, the variable node degree distribution remains unchanged over a range of code rates, while the optimal
variable node degree distribution varies for different rates. Generally, the proportion of degree two variable nodes controls the performance of the water-fall region and the error-floor region. It is well known that the proportion of degree two nodes must be less than the number of rows to get reasonable error-floor performance. In the SRC scheme, at higher code rates, the parity-check matrix has a small number of rows. Thus, the maximum number of degree two nodes in the mother code is limited to the number of rows at the highest code rate where the parity-check matrix has the smallest number of rows. Such limitation on the proportion of degree two variable nodes causes the performance losses to be more distinctive at lower rates. The performance of LDPC codes with SRC and standalone LDPC codes optimized at the specific code rates are compared in Figure 7. The curves confirm our reasoning.

Second, the check node degree distribution is not in the concentrated form. Most good check node degree distributions are in the concentrated form, meaning they have only one or two consecutive check node degrees. However, after combining, the check node degree distribution becomes disperse which degrades the performance.

To address these problems, the authors proposed LDPC codes with Row Combining with Edge Variation (RCEV).

2.2. Row Combining with Edge Variation (RCEV)

In the SRC scheme, the problems stem from the fact that no edges are neither added nor removed in the graph. Let’s see how this restriction can be relaxed to achieve better performance.

In the RCEV scheme the goal is to obtain an optimal degree distribution of the mother code, which was not possible with SRC. After combining two rows, the number of rows is decreased by one, thus the number of degree two nodes must be reduced as well. Towards this end, each time a row is combined, an edge is added to one of the degree two variable nodes such that the number of degree two variable node is always smaller than the number of rows after combining.

The performance of LDPC codes obtained by SRC and RCEV schemes are compared in Figure 8. The codes obtained by the RCEV scheme clearly outperform those with the SRC at lower rates at a small sacrifice in the performance at the highest code rate.

III. DECREASING THE CODE RATE

Now we turn our attention to some popular techniques
for decreasing the code rate. If a lower rate code is to be derived from a mother code, two approaches seem to be the most intuitive (not counting the simple repetition). One can preserve all parity bits from the mother code and reduce the number of information bits. This is called shortening or information nulling. In contrast, one can preserve all information bits and introduce additional parity bits. A common name for this technique is extending. In both cases, the result is a set of new codes whose rates are lower as compared to the mother code. We will present some efficient techniques that can be applied for shortening and extending.

1. Shortening

Take an optimized LDPC code of a medium or high rate and call it a mother code. It is our goal now to derive from it a new set of codes with decreasing code rates. Let us also impose a constraint that the set of codes should be decodable on the same graph as the mother code, similarly as it was the case with puncturing. One of the first things that come to mind is to reduce the number of information bits per block and keep the number of parities unchanged. Of course, the number of transmitted information bits should not be reduced by means of puncturing, since that would only impair the code’s performance and not improve it. Namely, the motivation behind reducing a code rate is to make the code more resilient to channel errors. One could, however, assume a predefined value for certain information bits during the encoding process and afterward skip the transmission of those bits. The decoder, on the other hand, would know the predefined values of those bits and would hence know their values with absolute certainty. Sometimes this technique is also referred to as information nulling - the value of predefined information bits is set to be zero (although in general it could be set to 1 as well).

This leads to an interesting question. If a certain subset of the information bits is to be "nullled", how should that subset be chosen such that it will result in the best possible performance? Here, we briefly present an approach introduced by Tian et al. [Tian05].

Figure 9 presents a structure of a mother code parity check matrix that can be shortened to achieve lower code rates. The columns starting from left represent the information bits that can be shortened, the transmitted information bits and the transmitted parities, respectively. If some of the information bits are shortened, the degree distribution of the transmitted bits changes with respect to the mother code. The idea in [Tian05] is to pick a good degree distribution for a mother code of rate $R_0$ and use a "constrained" density evolution to derive from it a set of optimized degree distributions for each desired rate $R < R_0$. Afterwards they employ a simple algorithm to assign a
degree to each variable node on the graph of the mother code such that the degree distributions of the shortened codes are as close as possible to the optimized degree distributions - the bits should be shortened starting from the first column.

To evaluate the performance, the mother code of block length 10000 and rate 0.5 was designed and shortened to achieve rates 0.4, 0.3, 0.2, 0.1. Additionally, the threshold for the corresponding degree distributions was calculated to serve as a reference for the performance evaluation. The results are presented in Figure 10 in terms of the gap to the threshold vs. the code rate, where the comparison was drawn at the BER of 10^{-5}.

Similarly as with puncturing, the gap to the threshold grows with the number of shortened bits. These results are quite expected, since the design of shortened codes was constrained by the original mother code degree distribution. We note, however, that the effective block length stays the same even though a smaller number of bits is transmitted - all the codes are decoded on the same graph. This property is very advantageous from the implementation point of view, since only one encoder and one decoder are needed to implement the entire set of proposed rates. Moreover, the encoding/decoding complexity for each code equals that of the mother code.

Unfortunately, there is also a disadvantage to it. Namely, a set of codes obtained by shortening cannot be employed in a type-II hybrid ARQ protocol, where
additional redundancy bits are transmitted stepwise to the receiver when the decoding is unsuccessful.

In the next section, we present another technique for lowering the rate, called extending, that will provide this functionality.

2. Extending

From the type-II hybrid ARQ point of view it would be interesting to see how a set of lower rate codes can be derived from a mother code, given by $H_0$, by adding more parity bits. This process can viewed as appending additional columns and rows to $H_0$, like shown in Figure 11. Each additional bit is a parity bit, therefore the number of new columns must be equal to the number of new rows.

Usually, it is desired that the set of codes be rate-compatible, in which case the section $A$ of the matrix in Figure 11 must contain only zero entries. This ensures the codes of higher rate do not involve bits from the codes of lower rates. The main challenge presented by extending is the design of the remaining sections, $B$ and $C_i$'s, which we
address in the following.

Say we add $M_1$ new parity bits to $H_0$. The column weights of $C_1$ will determine the degree of the new variable nodes. On the other hand, the weights of the $M_1$ new rows (below the last row of $H_0$) determine the degree of the new check nodes. They should be connected with the mother code bits, otherwise they would not provide any additional information. Hence, the $M_1$ new rows of section B must contain non-zero entries, although they should remain sparse.

Same observations hold when we add another set of parity bits, say $M_2$. Note, however, that each non-zero entry in $B$ changes the degree of the variable node represented by the corresponding column.

In [Yazdani04], the authors propose the following approach to the design of extended codes. All $C_1$ matrices should be square and identical to each other. They have a lower triangular form, which enables linear encoding, and they are designed with the PEG algorithm [Hu05] using a degree distribution identical to the mother code (see Figure 12).

The section B is composed of systematically placed identity matrices I which introduce the dependencies between newly added parity bits and the bits of higher rate codes. Notice that exactly two identity matrices are added in each expansion step.

In the first expansion step, the new parity bits build dependencies only with the mother code bits, while in every subsequent expansion, dependencies with both the mother code bits and the bits from the previous expansion step are introduced. This approach has many good properties. The extended rows and columns have a deterministic structure which makes the implementation modular and simple. Furthermore, the almost uniform weight distribution of new rows results in a better performance of the extended codes. The performance results, shown in Figure 13, adhere to the already seen tendency. The further we go from the code rate of the mother code, the less efficient the derived codes.

Notice that the extended codes cannot be decoded on the graph of the mother code. After each extension, the overall block length of the lower rate code becomes larger and hence requires a more complex decoder. One option is to decode all codes on the graph of the lowest rate code and set the LLR values of the untransmitted bits to 0. On the other hand, some extra circuitry could be added to dynamically enable/disable the extended parts of the graph and adapt it to the appropriate rate. Similar observations are true for encoding. The parity bits can be generated on demand, unlike with puncturing, where all parity bits are generated for all rates.

Finally, we note that other approaches have been studied in the literature. We would like to mention the efforts in [Li02] and [Yue05]. In the latter, the authors derive optimal extending distributions with density evolution but their reasoning applies only to a special family of LDPC codes called irregular repeat accumulate (IRA) codes.

**IV. COMMUNICATION SYSTEM**

In this final section, we show how a rate-compatible
communication system with a wide range of code rates can be designed by employing a combination of the techniques presented in the previous chapters.

As we progressed through different techniques for increasing and decreasing the code rate, one observation in particular has always been present—the further we go from the rate of a mother code, in any direction, the weaker the performance of the derived codes. Consequently, it is quite intuitive to set the rate of the mother code in the middle of the desired code rate range of a system. In example, if a system should cover rate from 0.3 to 0.9, the mother code should have a rate around 0.6, while the codes of other rates are derived using techniques we have covered in the previous sections. For increasing the rate, puncturing seems to prevail in virtually any scenario. On the other hand, for decreasing the rate the choice of a technique depends on system requirements. If HARQ support is required, extending should be used, while in the opposite case, shortening may be preferred since it requires less complex hardware.

As an example, we present a wide-range rate-compatible communication system from [Yue06], which covers rates from 0.1 to 0.95. The mother code is an IRA code of rate 0.5. Puncturing is used for increasing the rate from 0.5 to 0.95 and extending for decreasing it to 0.1. The complexity of the decoder (and encoder) is hence higher for lower rate codes, but the system can support HARQ and provide a higher error-resilience of lower rate codes due to longer block lengths. Figure 14 shows the throughput analysis of the system over the AWGN channel, when HARQ is employed to adapt to the time-varying channel state. The system throughput is constantly within less than 1 dB of the Shannon limit.

Several other rate-compatible systems based on LDPC codes have been studied in the literature as well as within some recent standardization activities. For further information refer to [Li02], [Yazdani04], [Tian05], and [3GPP].

V. CONCLUSION

We surveyed up-to-date techniques for making LDPC codes rate-adaptive. The techniques are categorized into two groups: the ones for 1) increasing and 2) decreasing the code rate. As for techniques for increasing the code rate, we introduced puncturing and parity-check equation combining. Shortening and extending techniques, on the other hand, are introduced for decreasing code rates.

Finally, we showed an example of a rate-compatible communication system based on LDPC codes that covers a very broad range of rates, from 0.1 to 0.95. It also fully supports operation under the HARQ protocol and yields throughput performance within less than 1 dB of the Shannon limit. Multiple recent standardization proposals have proven that rate-compatible LDPC codes are not solely limited to academia, but are making their way into practical communication systems as well. We hope this paper can be helpful to the researchers who work on designing rate-adaptive channel codes for the future communication systems.
Acknowledgement
This work was partly funded by Samsung Advanced Institute of Technology (SAIT) and Electronics and Telecommunications Research Institute (ETRI).

References
[16] 3GPP R1-061019 Structure LDPC coding with rate matching.
Jeongseok Ha

received the B.E. degree in Electronics from Kyungpook National University, Taegu, Korea in 1992, M.S. degree in Electronic and Electrical Engineering from the Pohang University of Science and Technology, Pohang, Korea, in 1994, and Ph.D. degree in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, in 2003. From 1994 to 1998, he was a Research Engineer with Electronics and Telecommunications Research Institute, Daejun, Korea, where he was involved in developing base stations of IS-95 cellular system and wireless local loop. Currently, he is with Information and Communications University (ICU), Daejun, Korea as an assistant professor. His research interests include digital communications and error-control systems.

E-mail: jsha@icu.ac.kr
Tel:+82-42-866-6167

Demijan Klinč

received his M.S. degree in electrical and information engineering from the Technical University Munich in 2004. He is now with the Georgia Institute of Technology working towards the Ph.D. degree. His principal interests include channel coding and information theory.

E-mail: demijan.klinč@gmail.com
Tel:+1-404-385-6692