Rate-Compatible Low Density Parity Check Codes for Capacity-Approaching ARQ Schemes in Packet Data Communications

Jing Li (Tiffany)*
Electrical Engineering Dept, Texas A&M University
College Station, TX 77843-3128
email: jingli@ee.tamu.edu

Krishna R. Narayanan
Electrical Engineering Dept, Texas A&M University
College Station, TX 77843-3128
email: krishna@ee.tamu.edu

ABSTRACT
Strong rate-compatible codes is important to achieve high throughput in hybrid automatic repeat request with forward error correction (ARQ/FEC) systems in packet data transmission. This paper focuses on the construction of efficient rate-compatible low density parity check (RC-LDPC) codes over a wide rate range. The conventional approach of puncturing is first studied. Analysis on the code ensemble and the asymptotic performance shows that it works well only at high rates and only when the amount of puncturing is small. To extend the dynamic rate range, a special approach of extending is proposed and is shown to produce good RC-LDPC codes at low rates. Combining both approaches, efficient RC-LDPC codes are constructed and a hybrid ARQ/FEC system using RC-LDPC codes is evaluated. The proposed LDPC-ARQ system can achieve information rate about 1 dB away from the theoretical limit, which is comparable to turbo ARQ systems [1][2], yet with lower decoding complexity.

KEY WORDS
rate compatible, low density parity check (LDPC) codes, density evolution, automatic repeat request (ARQ)

1 Introduction

Flexible rate is always desired in the design of practical error control systems. Rate-compatible (RC) codes are a family of nested codes where the codeword bits from the higher-rate codes are embedded in the lower-rate codes and, hence, can be encoded and decoded using a single encoder/decoder pair. They are of particular interest in packet data systems that allow for retransmission request such as automatic repeat request with forward error correction (ARQ/FEC) systems to achieve desired throughput efficiency with a high degree of flexibility.

Key elements concerning the throughput efficiency of an ARQ/FEC system include a wise ARQ strategy and, perhaps more importantly, a powerful rate-compatible code. Successful attempts in creating RC codes have used BCH codes [3], convolutional codes [4][5] and turbo codes [1][2][6]-[8]. BCH codes and convolutional codes are easily implementable but cannot provide near capacity performance. Turbo codes have demonstrated impressive performance, but the decoding complex is pretty high.

This paper focuses on low density parity check (LDPC) codes [9][10] which have been shown to provide performance comparable to turbo codes, yet with lower decoding complexity. Two approaches are investigated in this work to construct simple and efficient rate-compatible LDPC (RC-LDPC) codes: the conventional technique of puncturing and a special approach of extending. Analysis on code ensemble and asymptotic performance using density evolution (DE) reveals that the efficiency of puncturing is limited to high rate range where the amount of puncturing is small. To extend the dynamic code range, we propose and discuss a special code structure which uses extending to construct good RC-LDPC codes at low rates. Combining both techniques, a systematic model is presented to construct efficient RC-LDPC codes which offer strong error correction capability at a wide range of code rates.

The encouraging performance of RC-LDPC codes has opened the possibility for capacity-approaching throughput. A type II hybrid ARQ/FEC system using RC-LDPC codes is investigated and system throughput is analyzed. The LDPC-ARQ system can achieve information rate at about 1 dB from the capacity, which is comparable to turbo-ARQ systems in [1][2], yet with less (decoding) complexity.

The paper is organized as follows. Section II analyzes a hybrid ARQ/FEC system and pinpoints the importance of a strong FEC code. Section III discusses the construction of efficient RC-LDPC codes using both puncturing and extending. Simulations are provided along with the discussion. Section IV provides concluding remarks.

2 Hybrid ARQ/FEC Using RC-LDPC Codes

2.1 ARQ System Using RC-LDPC Codes

A typical ARQ/FEC system uses both error correction codes (ECC) and error detection codes (EDC, such as a cyclic redundancy check (CRC)). After the transmitted codeword is decoded by ECC, it is examined by EDC. If the decoding is deemed in error, a negative acknowledgment (NACK) is

*Tiffany Li is currently with the Electrical and Computer Engineering Dept. in Lehigh Unversity, Bethlehem, PA 18015.

sent as a retransmission request. When LDPC codes are used in an ARQ protocol, their strong error detection ability enables them to act as both ECC and EDC. This obviates the need for another EDC, and thereby reduces the overhead. Whether the error detection capability of LDPC codes can match that of a CRC is not fully discussed in this paper. Nevertheless we assume that the error detection capability provided by the LDPC code is sufficient for the application at hand.

The type-II hybrid ARQ/FEC system under investigation uses rate compatible codes along with code-combining and packet-combining to ensure the transmission reliability and to maximize the throughput. A packet is first transmitted using the highest rate code. If it is not deemed correctly decoded, an NACK is fed-back and a new set of parity bits is provided by the transmitter (incremental retransmission). This new set of parity bits, combined with all previous transmissions, is treated as a codeword of a lower rate code in the family (code combining) which provides stronger error correction capability.This procedure continues until all supplemental parity bits are used up, and then the procedure restarts with another "initial transmission". When a new copy of the same coded bits (either data or parities) are received, old copies are not discarded, but are combined with the new ones to facilitate decoding (packet combining). In general, packet combining is done by averaging the soft decision values from the multiple copies, and on AWGN channels, this is equivalent to maximum-likelihood (ML) diversity combining.Specifically for LDPC codes with soft message-passing decoder, the input message to the decoder (in log-likelihood ratio (LLR) form) of a bit s_i is obtained by $\sum_{j=1}^{k} \frac{2r_i^{(j)}}{\sigma^2}$, where $r_i^{(1)}, r_i^{(2)}, \cdots, r_i^{(k)}$ are the multiple copies received for the same bit s_i. The above strategy is optimal for achieving high throughput either in stop-and-wait ARQ or selective-repeat ARQ systems, under the assumption that the feedback channel is noiseless, that the buffer size is infinite, and that the transmission latency, the feedback channel traffic and the decoding complexity are not a concern.

2.2 Throughput Analysis

A standard measure for the efficiency of an ARQ scheme is throughput, which is defined as the average number of coded and modulated symbols that need to be transmitted for a single data bit to reach the destination error-free. Define p_i, $i=0, 1, \cdots$, as the probability that the decoder succeeds after the i_{th} retransmission but fails at all previous attempts (the initial transmission is considered 0_{th} retransmission). We have $p_i = (1-F_i)\sum_{j=0}^{i-1} F_j$, where F_i is the word/frame error rate (WER/FER) after the i_{th} retransmission. The system throughput, η, is given by:

$$\eta = K_0/(N_0 + \sum_{i=1}^{\infty} p_i M_i), \quad (1)$$

$$= R_0 \Big/ \Big(1 + F_0 \sum_{i=1}^{\infty} \frac{M_i}{N_0}(1-F_i)\sum_{j=1}^{i-1} F_j\Big), \quad (2)$$

Where K_0, N_0, and M_i denote the number of data bits in a frame/codeword, the packet size of the initial transmission, and the packet size of the i_{th} retransmissions, respectively. It is apparent from (2) that the error rate of the initial transmission, F_0, plays an important role in throughput efficiency, since subsequent transmissions occur only when the initial transmission fails. If F_0 is small, in particular, if $F_0 \to 0$, then the throughput $\eta \to R_0$, the highest possible rate in the system. Further, the granularity of the retransmission, $\frac{M_i}{N_0}$, also has an impact on η. Smaller retransmission size M_i means finer adjustment, which will improve throughput efficiency, but may incur more delay and decoding complexity in practical systems.

Since the word error rate, F_i, plays the key role in achieving high ARQ/FEC throughput, it is crucial to choose a strong code. Below we focus on the construction of good RC-LDPC codes.

3 Constructing RC-LDPC Codes

Due to the space limitation, we skip background introduction of LDPC codes. Interested readers please refer to [9][10] and the references therein. A regular LDPC code has parameters (N, K, t, s), which denote the codeword length, data block size, column weight and row weight of the parity check matrix, respectively. We use a sequential design, the bit filling method, to obtain t=3 regular LDPC codes as the mother code for the RC-LDPC code family. To ensure decent performance of the mother code, we have enforced the constraint that the girth (the length of the shortest cycle in the code graph) be at least 6 in the construction.

We would like to mention that irregular LDPC codes (with nonuniform column weights) have been shown to outperform regular LDPC codes in bit error rate (BER) [10], but the difference is very marginal for short code lengths (a few hundred to a few thousand bits). Further, whether they are also better in *word error rate* (which is the determining factor of ARQ throughput as shown in (2)) is less known and needs to be bench marked. We note that word error rate and error detection capability is much related to the minimum distance, d_m, of the code. For regular LDPC codes with $t \geq 3$, the ensemble average minimum distance increases linearly with code length [9]. However, this may not be true for irregular LDPC codes. Since we would like LDPC codes to assume the dual role of error correction and error detection, it is thus desirable to start with a regular code that typically has a large minimum distance.

3.1 Puncturing

Puncturing has been widely used in BCH, convolutional and turbo codes to achieve rate compatibility [1]-[8]. It is also applicable to LDPC codes. Through proper puncturing, a series of higher rate codes are obtained from the low rate mother code. The encoder generates the full set of parity bits, but some are not transmitted (punctured). The decoder inserts erasures to where parities are punctured and performs

the decoding algorithm as in a non-punctured case.

An LDPC code can be viewed as a parallel concatenation where each row in the parity check matrix H acts as a simple component code (a parity check). Consider an (N, K) LDPC code where L (parity) bits/columns are punctured. Those rows/checks that happen to have "1"s in the positions of the punctured bits are treated as being erased. To see how puncturing impairs the code performance, we examine the effect of puncturing on the ensemble of the LDPC codes and study the asymptotic performance of the punctured codes.

(1) Code Ensemble:

Consider the ensemble of (N, K, t, s) LDPC codes. Randomly pick a parity check matrix from the ensemble and puncture $L = \rho N$ columns, where $\rho = L/N$ is defined as the puncturing rate. Assuming all rows are independent, the average portion of rows being affected by at least one erasure, $\lambda_1(\rho)$, and by multiple (two or more) erasures, $\lambda_2(\rho)$, are given by:

$$\lambda_1(\rho) = 1 - \frac{\binom{N-\rho N}{s}}{\binom{N}{s}}, \quad (3)$$

$$\lambda_2(\rho) = 1 - \frac{\binom{N-\rho N}{s} + \binom{\rho N}{1}\binom{N-\rho N}{s-1}}{\binom{N}{s}}. \quad (4)$$

We observe that large value of $\lambda_2(\rho)$ has a destructive affect on the decoder performance. To see this, lets get back to the message-passing decoding algorithm of LDPC codes. Suppose bits $j_1, j_2, \cdots, j_s$ participate in check j, the extrinsic LLR information of bit j_k, denoted as $L_{e,j_k}(x)$, to be obtained from check j is given by:

$$L_{e,j_k}(x) = \boxplus_{i=1, i\neq k}^{s} L_{j_i}(x), \quad k = 1, 2, \cdots, s, \quad (5)$$

where $L_{j_i}(x)$ denotes the LLR message content of bit j_i, and operation $\boxplus$ is defined as $\gamma = \alpha \boxplus \beta = log((1 + e^{\alpha}e^{\beta})/(e^{\alpha} + e^{\beta}))$. An erasure in position j_i means its initial message content is 0, i.e. $L_{j_i}(x) = 0$. When multiple erasures present in one check, at least one term on the right hand side of (5) is 0. Since $\alpha \boxplus 0 = 0$, this leads to $L_{e,j_k}(x) = 0, \forall k \in \{1, 2, \cdots, w\}$. In other words, no information is exchanged/obtained from this row/check. When the percentage of such rows is large, message exchange becomes quite inefficient and ineffective. Simulations show that, in such cases, the decoding algorithm may get stuck in a "zero-trapping" state, leading to poor performance.

Solid lines and dashed lines in Fig. 1 plot $\lambda_1(\rho)$ and $\lambda_2(\rho)$ vs ρ for code rates of $R = \frac{3}{4}$, $\frac{2}{3}$, $\frac{1}{2}$, $\frac{1}{3}$ and $\frac{1}{5}$, respectively. It can be clearly seen that puncturing has a larger adverse impact when the mother code is of low rate than when the mother code is of high rate (the punctured code having the same code rate), which matches our simulations (see Fig. 3 and Fig. 5).

(2) Asymptotic Performance:

To further insights into the effect of puncturing, we use density evolution to examine the asymptotic performance of punctured LDPC codes and to quantify the performance loss caused by puncturing. Here asymptotic refers to an infinite code length and an infinite number of iterations in decoding. The idea of density evolution is to track the distribution of the messages passed along the code graph in each step, and to examine the portion of the incorrect messages (i.e. messages leading to the wrong decision). Details of density evolution on (non-punctured) LDPC codes can be found in [10]. Here we focus on the difference in the computation between the punctured and non-punctured cases.

Assuming AWGN channels with noise variance σ^2 and antipodal signaling with unit energy (± 1), the density of the initial messages (from the channel) of a non-punctured LDPC code is given by a Gaussian distribution:

$$f_{o,nonpunc}(x) = \frac{\sigma}{2\sqrt{2\pi}}\, e^{-\frac{(x-2/\sigma^2)^2}{8/\sigma^2}} = \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2}). \quad (6)$$

In the punctured case, the decoder inserts erasures (0 message content) to where the bits are punctured. With puncturing rate ρ, the density of the messages observed by the decoder is a mixture of Gaussian and Kronecker delta function:

$$f_{o,punc}(x) = (1 - \rho) \cdot \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2}) + \rho \cdot \delta(x). \quad (7)$$

The rest of the procedure is the same for both cases and can be found in [10]. Fig. 2 compares the thresholds computed using DE of non-punctured and punctured LDPC codes with puncturing rate $\rho = 10\%, 20\%, 30\%$ (regular t=3). Apparently, punctured codes suffer performance loss, and the effect is more evident as ρ increases. Hence, for a fixed desired rate (after puncturing), it is desirable to choose the mother code such that the percentage of puncturing is as small as possible. However, this will result in a limited range of achievable code rates. It can be seen from the plot that if the desired code rate (after puncturing) is high (which is typically the case), the performance loss suffered by picking a lower rate mother code is very large and increases as the desired code rate increases. Since for an ARQ system, the reliability of the first transmission is very important, creating high-rate codes using low-rate mother codes is not a good choice.

From the analysis of code ensemble and computation of the asymptotic performance, we conclude that puncturing provides a viable solution to produce RC-LDPC codes but the efficiency is limited at high rate range where the amount of puncturing is not large. This is also confirmed by computer simulations (see Fig. 3 and Fig. 5).

3.2 Extending

A strong motivation for using extending comes from the observation that the quality of the initial transmission is most important to achieve high ARQ throughput. Opposite to puncturing, extending builds RC codes from high rates to low rates by adding more parities. For RC-LDPC codes built from extending, the initial transmission corresponds to an *non-punctured* LDPC code, which has a good WER

in the first transmission (F_0 in (2)). Then, additional parity bits are added to reduce the rate in such a way that the extended code provides sufficiently good performance at the lower rate. Another motivation for using extending concerns the decoding complexity. Unlike puncturing where all parity bits are generated at the encoder regardless whether they will be used, extending allows bits to be generated only as needed, thus avoiding unnecessary computations at the encoder and the decoder.

Whereas this observation is not particularly new, it is not apparent how extending can be used to realize rate compatibility for most of the block codes and trellis codes. Previous researches have used repetition, which may be deemed as the simplest type of extending, to construct rate-compatible convolutional codes [5]. In this work, we exploit the intrinsic randomness in an LDPC code and propose a novel structure for building RC-LDPC codes using extending. Fig. 4(a) presents the proposed code structure. The parity check matrix of the highest rate code has $M_0 = N_0 - K$ rows and N_0 columns with column weight $t \geq 3$ (upper-left part in Fig. 4(a)). The parity check matrix of each lower rate code is constructed by padding M_i rows and M_i columns, until finally reaching a $(N_0 + \sum_{i=1}^{L} M_i, K)$ code after L levels of padding. A family of RC-LDPC codes of rates $R_0 > R_1 > \cdots > R_L$ thus results, where $R_i = K/(N_0 + \sum_{j=1}^{i} M_j), 1 \leq i \leq L$. To embed higher rate codewords in lower rate codewords, the upper-right part of each padding must be "0"s as shown. The squares in the bottom-right part (see Fig. 4(a)) have column weight 3 to ensure the resulting parity check matrix also has column weight of at least 3. The bottom-left area is made reasonably sparse to ease the construction and to save the decoding complexity, but at least one "1" is needed for each row in order to build sufficient dependencies between the code bits of the mother code and the newly added parity bits.

The encoder structure is shown in Fig. 4(b)(c). Like a conventional LDPC code, Gaussian elimination is used to derive generator matrix from the parity check matrix. It is easy to see that the generator matrices corresponding to the initial and subsequent transmissions take the form:

$$\Big[\mathbf{I} \,|\, \mathbf{G_0}\Big], \Big[\mathbf{I} \,|\, \mathbf{G_0} \,|\, \mathbf{G_1}\Big], \cdots, \Big[\mathbf{I} \,|\, \mathbf{G_0} \,|\, \mathbf{G_1} \,|\, \cdots \,|\, \mathbf{G_L}\Big].$$

where $\mathbf{I}$ is the identity matrix of size K, and $\mathbf{G}_i$ has dimensionality $K \times M_i$. This is important since this means that groups of parity bits can be generated independently and parity bits from the ith group is not required to decode the any $j_{th} < i_{th}$ transmission.

As mentioned above, one advantage of extending is its low complexity. Tab. 1 compares the encoding and decoding complexity of RC-LDPC codes constructed using puncturing and extending. As can be seen, puncturing requires a fixed complexity regardless of channel conditions, whereas the complexity of extending reduces as channel conditions improve. Most of the time extending involves less decoding complexity than puncturing.

Fig. 5 presents the performance of a set of RC-LDPC codes of rates $\frac{4}{8}\backslash\frac{4}{9}\backslash\frac{4}{10}\backslash\frac{4}{11}\backslash\frac{4}{12}$ constructed by extending. As can be seen, extending produces efficient RC-LDPC codes at low rates, but does not work nearly as well at high rates. In Fig. 3, a family of $\frac{16}{20}\backslash\frac{16}{21}\backslash\frac{16}{22}\backslash\frac{16}{23}\backslash\frac{16}{24}$ RC-LDPC codes constructed using puncturing demonstrates good performance, but those using extending fail to offer incremental improvement in performance. We observe that the extended H matrix therein has fairly large weights in the first M_0 rows (corresponding to the mother code), whereas very low weights in the padded rows and, hence, the padded rows have little influence on the overall code. It is shown that the row weight profile of an LDPC code should be "concentrated", i.e. all rows have similar weights, in order to achieve good performance [10]. This huge discrepancy among row weights may be the explanation why the performance of the extended H matrix is pretty bad in Fig. 3.

3.3 Overall System Model

With the above discussion and simulations, it follows that efficient RC-LDPC codes could take advantage of both approaches. Fig. 6 presents the overall system structure. As an example, we construct a family of $K = 1024$ RC-LDPC codes with rates ranging from $\frac{8}{11}$ to $\frac{8}{20}$ (Fig. 7). The mother code is a rate-$\frac{8}{14}$ regular LDPC code. Puncturing is used to to get rates $\frac{8}{13}\backslash\frac{8}{12}\backslash\frac{8}{11}$ and extending to get rates $\frac{8}{15}\backslash\frac{8}{16}\backslash\frac{8}{17}\backslash\frac{8}{18}\backslash\frac{8}{19}\backslash\frac{8}{20}$. As can be seen, each individual code provides good error correction capability, and they collectively offer a steady improvement in performance with code compatibility.

Throughput of the proposed ARQ/FEC system using RC-LDPC codes as defined in (2) is evaluated in Fig. 8 for the example considered above. Also shown is the throughput of ARQ systems using rate-compatible turbo codes in [1][2]. We see that the proposed LDPC-ARQ system can achieve error-free information rate around 1 dB away from the capacity limit, which is on par with turbo-ARQ systems. Yet, LDPC codes have less decoding complexity than turbo codes. In our example, the retransmission packet size is pretty small (128 bits), hence the reduction in rate required for adaptation is more gradual than most of the incremental redundancy ARQ systems in literature, and the capacity curve is smooth rather than "staircase-like". This is desired for maximizing throughput. However, in real applications, the retransmission sizes need to be balanced against the overhead of decoding complexity, the volume of traffic on the feedback channel and the delay caused by subsequent retransmissions. In case of need, several small retransmission packets may be combined as one to speed the process.

4 Conclusion

Efficient rate compatibility and adaptivity can be achieved from LDPC codes if the family of codes is carefully designed. The main result in this paper is that in order to obtain a good RC-LDPC code with a wide range of rates $R_1 < R_2 < \ldots R_M$, it is not a wise strategy to use a mother code with rate R_1 and puncture to obtain the other rates (the conventional practice). A special construction for

LDPC codes has been proposed which uses a mother code of rate R_j (closer to R_M); higher rates are obtained via puncturing and lower rates through a novel extending technique that has been discussed. The proposed LDPC-ARQ system can achieve near capacity throughput, and is appealing to a wide variety of packet data applications, where powerful codes are required, feedback channels are available and latency due to retransmission overhead is acceptable.

References

[1] D. N. Rowitch, and L. B. Milstein, "Rate compatible punctured turbo (RCPT) codes in a hybrid FER/ARQ system," *Proc. GLOBECOM*, Phoenix, AZ, USA, 1997, 55-59.

[2] R. Mantha, and F. R. Kschischang, "A capacity-approaching hybrid ARQ scheme using Turbo codes," *Proc. GLOBECOM*, 1999, 2341-2345.

[3] S. Lin, and P. S. Yu, "Hybrid ARQ Scheme with parity retransmission for error control of satellite channels," *IEEE Trans. Commun.*, July, 1982, 1701-1719.

[4] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. Com-36, 1988, 389-400.

[5] Z. Lin, and A. Svensson, "New rate-compatible repetition convolutional codes," *IEEE Trans. Info. Theory*, vol. 46, No. 7, Nov. 2000, 2651-2657.

[6] A. S. Barbulescu, and S. S.Pietrobon, "Rate compatible turbo codes," *Electron. Let.*, Mar., 1995, 535-536.

[7] D. N. Rowitch, and L. B. Milstein, "On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes," *IEEE Trans. Commun.*, vol.48, No.6, June, 2000, 55-67.

[8] T. Ji, and W. E. Stark, "Turbo-coded ARQ schemes for DS-CDMA Data Networks over fading and shadowing channels: throughput, delay and energy efficiency," *IEEE J. Sele. Area. Commun.*, Aug.2000, 1355-1364.

[9] R. G. Gallager, *Low density parity check codes*, (MIT press, Cambridge, MA, 1963).

[10] T. J. Richardson, A. Shokrollahi and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, Feb. 2001, 619-637.

Table 1. Complexity comparison: puncturing vs extending

Encoding		
	puncturing	extending
XOR	$(K^2-K)(\frac{1}{R_0}-1)$	$(K^2-K)(\frac{1}{R_i}-1)$
OR	$K^2(\frac{1}{R_0}-1)$	$K^2(\frac{1}{R_i}-1)$
Decoding (per iteration)		
	puncturing	extending
addition	$3tK/R_0$	$3t'K/R_i$
table-lookup	$2tK/R_0$	$2t'K/R_i$

$t=3,\ \ 3\le t'\le 3.3,\ \ R_i=K/(N_0+\sum_{j=1}^{i}M_j)$.

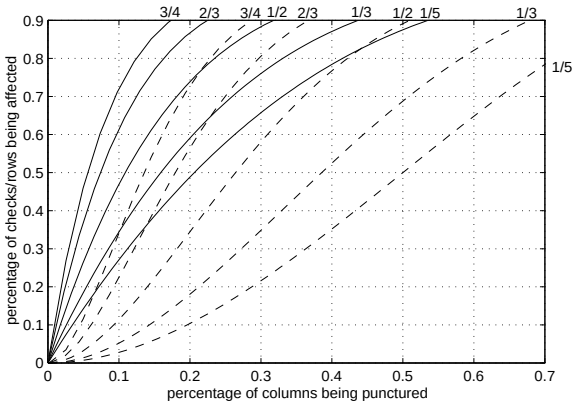

Figure 1. Effect of puncturing on t=3 LDPC code ensemble. Solid lines and dashed lines denote the percentage of rows affected by at at least one erasure and by multiple erasures.

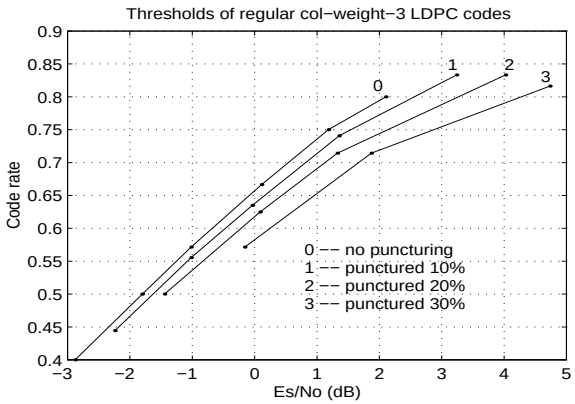

Figure 2. Thresholds of punctured LDPC codes (t=3).

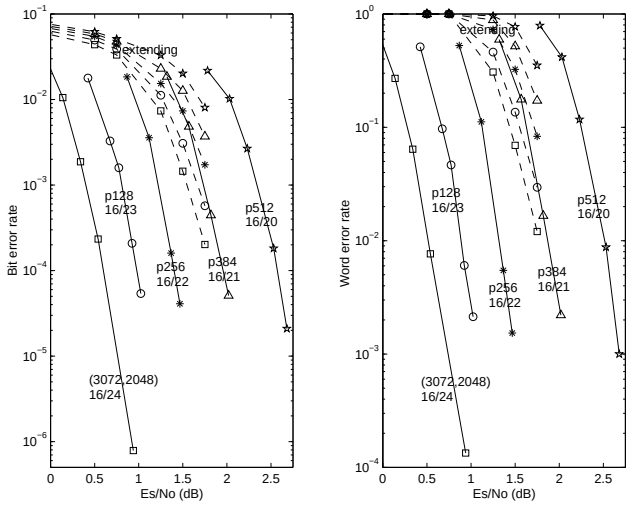

Figure 3. Puncturing builds efficient RC-LDPC codes at high rates. Solid lines: RC-LDPC by puncturing; left to right: (3072, 2048, $\frac{16}{24}$) mother code (non-punctured, t=3), 128, 256, 384, and 512 parities punctured. Dashed lines: RC-LDPC by extending (for comparison); right to left: (2560, 2048, $\frac{16}{20}$) mother code (unextended, t=3), 128, 256, 384, and 512 parities extended.

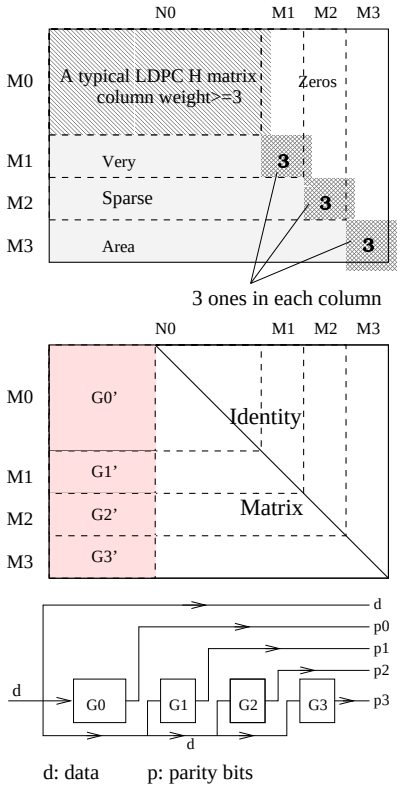

Figure 4. Illustration of RC-LDPC codes by extending. (a). Structure of parity check matrix. (b). Parity check matrix in its systematic form. (c). Encoder structure.

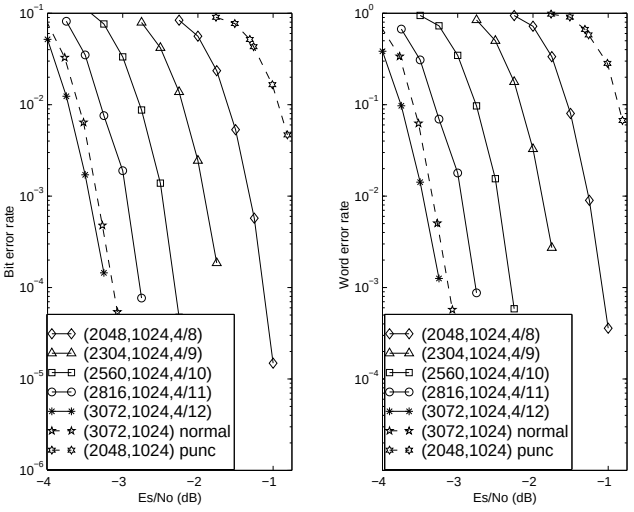

Figure 5. Extending builds efficient RC-LDPC codes at low rates. Solid lines: RC-LDPC by extending; from right to left: (2048, 1024, 4/8) mother code (unextended, t=3), 256, 512, 768, and 1024 parities extended, respectively (t=3.3 on average). Dashed lines: RC-LDPC codes by puncturing (for comparison); from left to right: (3072, 1024, 4/12) mother code (non-punctured, t=3), (2048, 1024, 4/8) code after 1024 parities punctured.

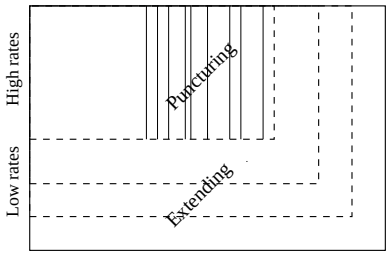

Figure 6. Overall system model for RC-LDPC codes using both puncturing and extending

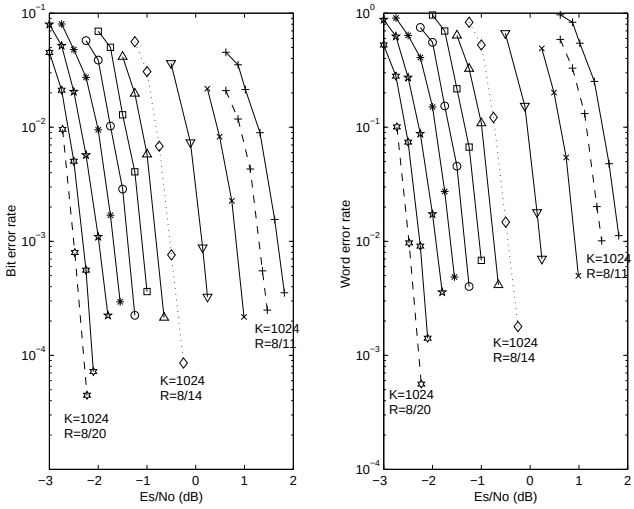

Figure 7. Performance of RC-LDPC codes using both puncturing and extending (K=1024). Dotted line: a regular rate $8/14$ LDPC code (mother code). Solid lines to the left of the dotted line are codes constructed by extending; from right to left: rate $8/15$, $8/16$, $8/17$, $8/18$, $8/19$, $8/20$. Solid lines to the right of the dotted line are codes constructed by puncturing; from left to right: rate $8/13$, $8/12$, $8/11$. Dashed lines: rate $8/20$ and $8/11$ regular (non-punctured) LDPC codes for comparison purposes.

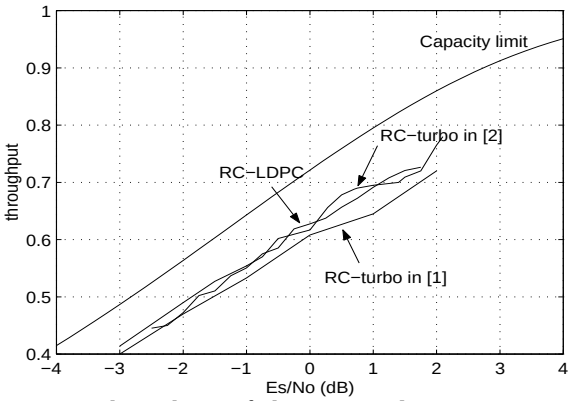

Figure 8. Throughput of the proposed ARQ system using RC-LDPC codes with K=1024 and results of RC-turbo ARQ from [1] [2].