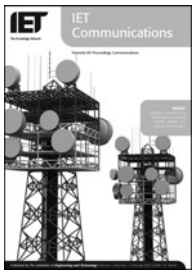
Published in IET Communications
Received on 4th September 2006
Revised on 14th June 2007
doi: 10.1049/iet-com:20060513



ISSN 1751-8628

# Improved regular and semi-random rate-compatible low-density parity-check codes with short block lengths

*S.F. Zaheer*[1] *S.A. Zummo*[2] *M.A. Landolsi*[2] *M.A. Kousa*[2]

[1]*Telecard Ltd., Karachi, Pakistan*
[2]*Electrical Engineering Department, KFUPM, Dhahran 31261, Saudi Arabia*
*E-mail: zummo@kfupm.edu.sa*

**Abstract:** Powerful rate-compatible codes are essential for achieving high throughput in hybrid automatic repeat request (ARQ) systems for networks utilising packet data transmission. The paper focuses on the construction of efficient rate-compatible low-density parity-check (RC-LDPC) codes over a wide range of rates. Two LDPC code families are considered; namely, regular LDPC codes which are known for good performance and low error floor, and semi-random LDPC codes which offer performance similar to regular LDPC codes with the additional property of linear-time encoding. An algorithm for the design of punctured regular RC-LDPC codes that have low error floor is presented. Furthermore, systematic algorithms for the construction of semi-random RC-LDPC codes are proposed based on puncturing and extending. The performance of a type-II hybrid ARQ system employing the proposed RC-LDPC codes is investigated. Compared with existing hybrid ARQ systems based on regular LDPC codes, the proposed ARQ system based on semi-random LDPC codes offers the advantages of linear-time encoding and higher throughput.

## 1 Introduction

A flexible code rate is always desired in the design of practical error control systems. Rate-compatible (RC) codes are a nested family of codes where the parity bits of the higher rate codes are embedded in the parity bits of the lower rate codes [1]. RC codes have many applications in packet data communications where adaptive coding and/or unequal error protection is required [1]. One of the main advantages of using RC schemes is that all the codes in the family can be encoded and decoded using a single encoder/decoder pair. In addition, these schemes provide an efficient framework for the transmission of information using hybrid automatic repeat request/forward error correction (ARQ/FEC) protocols.

In this paper, we focus on LDPC codes [2] to design hybrid ARQ schemes based on RC codes. RC-LDPC codes were introduced in [3], where it was shown that puncturing alone cannot provide a family of well-performing RC-LDPC codes with a wide range of rates. This is because at higher rates the large number of punctured bits (erasures) paralyses the iterative soft-decision decoder. To overcome this problem, both puncturing and extending were used in [3] to create a family of RC codes from a regular LDPC code. In [4], RC-LDPC codes were designed based on an optimised irregular LDPC code, constructed based on progressive edge growth [5]. In [6], optimal puncturing distributions for irregular LDPC codes were obtained from the perspective of minimising the code threshold, which is defined as the signal-to-noise ratio (SNR) value above which the probability of decoding error approaches zero, while the probability of decoding error is non-zero for values of SNR lower than threshold [7]. The theoretical performance of punctured LDPC codes was analysed in [6] using the Gaussian approximation approach [7]. Based on the analysis, a design rule for good puncturing distributions was proposed, which apply to LDPC codes of large block lengths in the order of $10^5$ bits.

Recently, in [8], a systematic method was proposed for finding good puncturing distributions for finite-length LDPC codes, in which the codeword length is small, which

makes asymptotic analysis techniques (such as density evolution) do not applicable. The idea is based on the observation that a punctured node will be recovered with reliable messages when two conditions are satisfied; namely, the punctured node has more neighbouring checknodes, and each of the checknodes has more reliable neighbours (variable nodes) except for the punctured node. For example, a punctured variable node that has check nodes whose remaining neighbouring variable nodes are unpunctured will have non-zero messages from the checknodes in the first iteration, and thus it is called a one-step-recoverable (1-SR) node since it is recovered in the first iteration. The 1-SR nodes and unpunctured nodes will help recover some of the remaining punctured nodes in the next iterations. In general, the punctured nodes recovered in the $i$th iteration are called $i$-SR nodes. Therefore the puncturing rule was to puncture nodes that require a smaller number of iterations for recovery, which results not only in less iterations to decode codewords but also in a better performance at a given code rate. This method enables the design of punctured finite-length LDPC codes that outperform randomly punctured LDPC codes.

In [9], the authors propose another approach for designing RC-LDPC codes based on the use of desirable node degree profiles satisfying certain row and column constraints (for the parity matrices of the RC codes). The authors present simulation results that demonstrate that their method produces codes that perform uniformly close to the Shannon theoretical limits, and also have some improvement in the frame error rate compared with the RC-LDPC code families presented in [4]. In this paper, we design improved RC-LDPC codes for short block length applications based on the regular and the semi-random LDPC code structures. The work presented in this paper differs from the previous work in the following aspects.

- The previous work on RC-LDPC codes with finite block lengths – with the exception of [8] – focuses on the design of punctured RC-LDPC codes using random puncturing. In this work, we design RC-LDPC codes using systematic algorithms for puncturing that outperform random puncturing. The criteria used in this paper for the design of punctured LDPC codes is different from those used in [8]. In particular, for punctured regular LDPC codes, we propose the criterion of maximisation of the girth average of the punctured code, whereas for punctured semi-random LDPC codes, we propose a simple puncturing pattern that results in high reliability of the punctured variable nodes.

- The RC-LDPC codes designed in [4] are based on irregular mother codes. The combination of an irregular mother code and 500 maximum decoder iterations leads to higher throughput as compared with the regular RC-LDPC codes of [3]. On the other hand, the RC-LDPC codes designed in this paper are based on a semi-random LDPC mother code which has a performance similar to that of a regular-(3,6) LDPC code (whose $\boldsymbol{H}$ matrix contains 3 ones in each column and 6 ones in each row), with the extra advantage of the low-complexity encoding. Results show that the RC-LDPC codes thus designed outperfrom the RC-LDPC codes of [3].

- We propose an algorithm to determine the girth of the variable nodes of an LDPC code. The algorithm is based on the adjacency matrices [10, 11] of the LDPC code.

- Results for RC-LDPC and hybrid ARQ are also presented for uncorrelated Rayleigh fading channels.

The paper is organised as follows. In Section 2, we present a brief overview of LDPC codes and type-II hybrid ARQ. An algorithm for the design of punctured regular RC-LDPC codes that have large girth average and low error floors is presented in Section 3. In Section 4, we present design guidelines for the construction of RC-LDPC codes based on semi-random LDPC codes. The designed RC-LDPC codes are applied to a type-II hybrid ARQ scheme, and the results are presented in Section 5. The main outcomes of the paper are summarised in Section 6.

# 2 System model

## 2.1 Regular LDPC codes

LDPC codes are block codes defined by a sparse parity-check matrix [2, 12]. A regular $(j, l)$ LDPC code with a rate of $R_c = k/n$ is defined by an $(n-k) \times n$ parity-check matrix having exactly $j$ ones in each column and exactly $l$ ones in each row, where $j < l$ and both are small compared with the block length $n$ [12]. The main advantage of regular LDPC codes is their large minimum distance, and hence low error floors as compared to turbo and irregular LDPC codes [13, 14]. The regular-(3,6) LDPC ensemble is the best regular ensemble [13] in terms of its minimum distance, and therefore it is used in this paper (as in [3]) for designing regular RC-LDPC codes.

An LDPC code can be represented by a Tanner graph, which is essentially a visual representation of the parity-check matrix of the code [12]. Recall that an $(n-k) \times n$ parity-check matrix $\boldsymbol{H}$ defines a code in which the $n$ bits of each codeword satisfy a set of $(n-k)$ parity-check equations. The Tanner graph contains $n$ 'variable' nodes, one for each codeword bit, and $(n-k)$ 'check' nodes, one for each of the parity-check equations. For the special case of a $(j, l)$ regular LDPC code, each bit is involved in $j$ parity-check equations. Hence, the number of edges emanating from a variable node is always $j$ and the variable node is said to be of degree $j$ [3]. Similarly, because each parity check equation involves $l$ bits, the number of edges emanating from each check node is always $l$ and the check node is said to be of degree $l$.

Puncturing constructs high-rate codes from low-rate codes by deleting parity bits [15]. Therefore the transmitter does not transmit the punctured parity bits. For the decoding of

a punctured LDPC code, the decoder inserts erasures where the parity bits are punctured and performs the decoding algorithm as in a non-punctured case [3]. On the other hand, extending constructs low-rate codes from high-rate codes by adding parity bits. At the decoder, the lowest rate code is used for decoding [4]. LDPC codes are decoded using an efficient iterative decoding algorithm, where soft likelihood information about the codeword bits is updated iteratively. The decoding algorithm, known as belief-propagation (or the sum-product algorithm) is shown to closely approximate the optimal maximum likelihood decoding [16], especially at high SNR.

### 2.2 Semi-random LDPC codes

For the purpose of encoding, the parity-check matrix $\boldsymbol{H}$ has to be transformed into the systematic form using Gaussian elimination [12]. The generator matrix $\boldsymbol{G}$ is then obtained from the systematic form of the $\boldsymbol{H}$ matrix. This transformation usually destroys the sparseness of the $\boldsymbol{H}$ matrix, resulting in a complex encoding process. Recent contributions have shown that LDPC codes are also amenable to simple encoding structures [17–19]. Particularly, in [19], a method has been proposed for the construction of a parity-check matrix which enables linear-time complexity encoding, while at the same time provides a performance similar to regular random LDPC codes. These LDPC codes are called semi-random LDPC codes. The parity-check matrix is obtained by concatenating a deterministic sub-matrix $\boldsymbol{H}_p$ with a randomly constructed sub-matrix $\boldsymbol{H}_d$. An example for the parity-check matrix of a (8,4) semi-random LDPC code is given as

$$\boldsymbol{H} = \left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right) \quad \begin{array}{c} \underbrace{\phantom{0000}}_{\boldsymbol{H}_p} \quad \underbrace{\phantom{0000}}_{\boldsymbol{H}_d} \end{array} \tag{1}$$

The systematic codeword is expressed as $\boldsymbol{c} = [\boldsymbol{p}\ \boldsymbol{d}]$, where $\boldsymbol{p}$ is a vector containing the parity bits and $\boldsymbol{d}$ is a vector containing the data bits. The parity-check matrix is decomposed as $\boldsymbol{H} = [\boldsymbol{H}_p \quad \boldsymbol{H}_d]$. Since every codeword $\boldsymbol{c}$ should satisfy the parity-check equations, we can write

$$\left[\boldsymbol{H}_p \quad \boldsymbol{H}_d\right]\left[\boldsymbol{p} \quad \boldsymbol{d}\right]^{\mathrm{T}} = \boldsymbol{0} \tag{2}$$

From (3), the parity vector $\boldsymbol{p} = \{p_i\}$ can be calculated [19] from the information sequence $\boldsymbol{d} = \{d_i\}$ as follows

$$\begin{aligned} p_1 &= \sum_{j=1}^{k} h_{d,1j} d_j \\ p_i &= p_{i-1} + \sum_{j=1}^{k} h_{d,ij} d_j \pmod 2, \quad i = 2, \ldots, n-k \end{aligned} \tag{3}$$

From (3), the codeword of a semi-random LDPC code can be generated recursively, with a complexity that grows linearly with the block length.

### 2.3 Type-II hybrid ARQ

Type-II hybrid ARQ adapts to changing channel conditions through the use of incremental redundancy [1]. In type-II hybrid ARQ, a packet is first transmitted using the highest rate code. If it is not deemed correctly decoded, a NACK (Negative ACKnowlegement) is fed-back to the transmitter and a new set of parity bits is provided by the transmitter (incremental retransmission) [12]. The range of code rates for a family of RC codes is defined as

$$R = \frac{P}{P + l}, \quad l = 1, \ldots, l', \ldots, L \tag{4}$$

For a family of codes obtained from a mother code of a rate $P/(P + l')$, codes with rates ranging from $P/(P + 1)$ to $P/(P + l' - 1)$ can be obtained through puncturing. On the other hand, codes with rates ranging from $P/(P + l' + 1)$ to $P/(P + L)$ can be obtained through extending, which is the transmission of additional parity bits.

## 3 Regular RC-LDPC codes

RC-LDPC codes based on regular-(3,6) LDPC codes were designed in [3] by using random puncturing in order to obtain codes with rates higher than that of the mother code. However, the puncturing pattern(s) chosen were not optimised with respect to any criterion. For LDPC codes, it is desirable to puncture variable nodes having the small degrees, in order to minimise the number of check nodes that are affected by erasures. However, in regular LDPC codes, all variable nodes have the same degree and therefore the selection of the variable nodes to be punctured cannot be done on the basis of the variable node degree. One method of comparing the performance of LDPC codes with different puncturing patterns would be through exhaustive search using Monte Carlo simulation, which is very complex and time consuming. Therefore methods of comparing different puncturing patterns without resorting to exhaustive search are required.

For a given block length and degree distribution of the underlying Tanner graph, the ensemble of LDPC code with short block length can have considerable performance variation, specially at high SNR [20]. An efficient heuristic algorithm for finding good LDPC codes based on the girth distribution of the Tanner graph was presented in [20]. In this paper, we propose a new algorithm to compute the girth distribution of an LDPC code. This algorithm is then used to obtain puncturing patterns that result in punctured codes with good performance. We claim that a search employing the proposed algorithm over an ensemble of puncturing patterns will result in a puncturing pattern




doi: 10.1049/iet-com:20060513

that outperforms a pattern chosen at random with high probability.

### 3.1 Computation of the node girth

The girth of an LDPC code refers to the length of the shortest loop (or cycle) present in the code's equivalent Tanner graph [12]. In [20], this term was used in a wider sense, where the girth at a variable node $u$ is defined as the length of the shortest cycle that passes through $u$. The girth distribution, $g(w)$, $w = 4, 6, \ldots, w_{\max}$ of a Tanner graph refers to the fraction of the variable nodes with girth $w$, where $w_{\max}$ is the maximum girth in the graph. The girth average for a graph is defined as

$$\sum_{t=2}^{w_{\max}/2} 2t \cdot g(2t) \tag{5}$$

Intuitively, the girth distribution is related to the sub-optimality of the iterative decoder. It is well known that for a cycle-free Tanner graph, belief propagation results in maxium likelihood decoding [21]. The girth of a variable node indicates the length of the shortest path, or equivalently the smallest number of iterations, for a message sent by that node to propagate back to the node itself. Before this number of iterations is reached, the 'belief' associated with the node is 'optimally' propagated to the rest of the graph. To have a performance close to the optimal, it is therefore favourable to make the girth of variable nodes' as large as possible, or in other words, to have more variable nodes with larger girths [20].

In the following, we propose to use the powers of the adjacency matrix [10, 11] of the code's parity-check matrix for the computation of the girth of a given variable node. Assume a matrix $\boldsymbol{H}$ of size $(m \times n)$. Denoting all the nodes (variable and check nodes) of the code's graph as $v_1, v_2, \ldots, v_p, ,$ where $p = m + n$, and define the adjacency matrix $\boldsymbol{A} = [a_{ij}]$ to be the $p \times p$ symmetric binary matrix, with the $(i, j)$th element defined as

$$a_{ij} = \begin{cases} 1 & \text{if an edge connects } v_i \text{ with } v_j \\ 0 & \text{otherwise} \end{cases}$$

The natural ordering of the nodes for an LDPC graph results [10] in the following relationship

$$\boldsymbol{A} = \begin{pmatrix} \mathbf{0} & \boldsymbol{H} \\ \boldsymbol{H}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} \tag{6}$$

It was shown in [10] that the $(i, j)$-entry of $\boldsymbol{A}^w$ equals the number of paths of length $w$ from $v_i$ to $v_j$. In the following, we provide necessary conditions for the node girth to be of a certain value.

*Proposition 1:* For a matrix $\boldsymbol{H}$ of size $(m \times n)$, the girth of a variable node $u$ is $w$ if

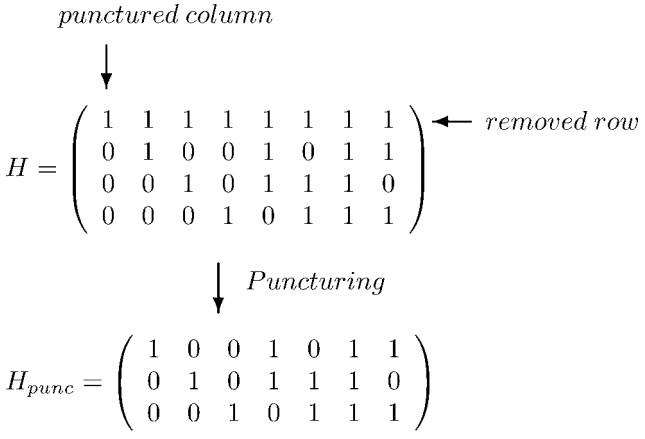
$$A_{ij}^{(w/2)} \geq 2 \quad \text{and} \quad A_{ij}^{(w/2)-2} = 0, \quad \forall i \text{ and } j = u + m \tag{7}$$

*Proof:* In the adjacency matrix corresponding to an LDPC code with a parity-check matrix $\boldsymbol{H}$, the rows and columns numbered 1 to $m$ correspond to the $m$ rows (check nodes) of $\boldsymbol{H}$, and the rows and columns numbered $(m+1)$ to $(m+n)$ correspond to the $n$ columns (variable nodes) of $\boldsymbol{H}$ [22]. For a variable node $u$ to have a girth $w$, there should be at least two paths of length $w/2$ between the node $u$ and any other node, that is, $A_{ij}^{(w/2)} \geq 2$. In addition, no paths of length $w/2 - 2$ should exist between the node $u$ and any other node, that is, $A_{ij}^{(w/2)-2} = 0$. The latter condition arises since the presence of paths of length $(w/2 - 2)$ would lead to the variable node having a girth of $(w - 2)$ instead of $w$. □

### 3.2 Heuristic search for good puncturing patterns

An algorithm to compute the girth average of an LDPC code (different from that in 3.1), was used in [20] to compare randomly constructed LDPC codes. In the following, the node girth computation discussed in 3.1 will be used to compute the girth average of LDPC codes, and then to compare randomly punctured codes of short block lengths according to their girth averages.

Consider a $(n, k)$ LDPC code, where the corresponding $\boldsymbol{H}$ matrix has a size of $(n - k) \times n$. Puncturing involves the deletion of parity bits. For example, deleting $p$ parity bits leads to a $(n - p, k)$ code, and the corresponding parity-check matrix will be of size $(n - p - k) \times (n - p)$. Hence, $p$ rows and and $p$ columns have to be removed from the original $\boldsymbol{H}$ matrix, as shown in Fig. 1. The rows which are removed correspond to the non-zero values of the selected columns [23]. For example, in a regular-(3,6) LDPC code,




$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$H_{punc} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

**Figure 1** *Effect of puncturing on the parity-check matrix of a linear block code (the first bit from the left is punctured resulting in removing the first column and first row of* $\boldsymbol{H}$*).*

each column has three rows which could possibly be removed since the column weight is 3. Consider the case of puncturing one parity bit. In this case, three different codes may be obtained since the removal of three different rows (and the same column for each case) results in three different parity-check matrices. As the number of the removed columns increases, the number of codes to be compared increases exponentially, which renders the comparison of punctured codes intractable. In addition, obtaining the 'actual' punctured matrix is viable only for systematic $\boldsymbol{H}$ matrices. For non-systematic random matrices, the possibilities for the punctured matrices are very large, and would increase with column weight and number of punctured bits.

In order to reduce the complexity of the search algorithm, the columns corresponding to the punctured bits are removed from the $\boldsymbol{H}$ matrix, while none of the rows are removed. The punctured codes thus obtained are then compared using the girth average criterion. The important question is: how does this approach affect the comparison between different puncturing patterns? This question can be answered in the following two points.

1. Removing only the punctured columns (and none of the rows) leaves a larger number of ones in the matrix for which the average girth is being computed, as compared with the actual punctured matrix (which has both columns and rows removed), which is expected to yield a larger fraction of ones, and subsequently more cycles (including short cycles). Therefore the value of average girth (for the variable nodes) will be smaller than that of the actual punctured matrix, and this was indeed observed through simulations.

2. The number of ones removed from the $\boldsymbol{H}$ matrix for each puncturing pattern is the same since for regular codes the number of ones removed equals the column weight $\times$number of bits punctured. Therefore probabilistically speaking, the number of variable nodes whose girth value is affected by this method is the same for all puncturing patterns. This would be supported by the random construction of the code. Therefore, if the actual punctured code would give an average girth of $x$ and this method gives an average girth of $x - y$, then the difference caused by this method (which is $y$) would be the same for all patterns with high probability.

Based on the above discussion, the proposed heuristic search algorithm can be summarised as follows.

1. Generate a random puncturing pattern, to select the bits to be punctured.

2. Remove the columns from the parity-check matrix $\boldsymbol{H}$ corresponding to the punctured bits (but not the rows), resulting in a matrix $\boldsymbol{H}_{\text{punc}}$.

3. Calculate the girth of the variable nodes in $\boldsymbol{H}_{\text{punc}}$ using (7).

4. Calculate the girth average of the punctured code using (5).

5. Repeat steps 1–4 a finite number of times, and select the random puncturing pattern which results in the punctured code with the maximum girth average.

Another advantage of the heuristic search algorithm is that it selects the non-punctured variable nodes of the mother code – represented by the columns of the parity-check matrix of the mother code that are not removed – to be those that have large girth. This is desirable since this will lead to high reliability for the non-punctured variable nodes, which will subsequently lead to a reliable estimate of the punctured variable nodes. Note that punctured LDPC codes are decoded using the parity-check matrix of the mother code.

Another valid criterion for selecting puncturing patterns is to select the puncturing pattern resulting in a code with the minimum number of variable nodes of the minimum girth [20]. Simulation results showed that this criterion leads to the same set of punctured codes as those selected via the maximum girth average criterion. The reason can be understood by looking at Tables 1 and 2, which show, respectively, the distribution of the girth of variable nodes averaged over 500 punctured codes, and the fraction of variable nodes with different girths for two different punctured codes. It can be seen that two terms – the fraction of variable nodes of girth 6 and girth 8 – completely dominate the distribution. Therefore the reduction in one term leads to a corresponding increase in the other term, and this effect is captured by the girth average.

The proposed heuristic search algorithm was used to select a puncturing pattern that maximises the average girth of the punctured code. It is shown in [10] that for a rate-1/2, girth-6, regular-(3,6) LDPC code with block length of 1000 bits, the error floor occurs at BER of $<10^{-8}$. Since the difference in the performance of codes with different girths is manifested in the error floor region at high SNR, the block length of the code has to be chosen such that the high SNR region is in the reach of simulation. Hence, the block length of the mother code is chosen to be $n = 256$, with 64 parity bits being punctured. The maximum number of decoder iterations used was set to 25.

**Table 1** Distribution of the girth of variable nodes averaged over 500 punctured codes

| Girth | Fraction of variable nodes, % |
|---|---|
| 6 | 69.6 |
| 8 | 30.2 |
| 10 | 0.2 |




doi: 10.1049/iet-com:20060513

**Table 2** Fraction of variable nodes with girths of 6 and 8 for two different punctured codes

| Girth | Code 1 | Code 2 |
|---|---|---|
| 6 | 59.3% | 58.8% |
| 8 | 40.6% | 41.1% |
| Average | 6.8125 | 6.687 |

To verify the proposed puncturing algorithm based on the girth average criterion, a search was performed over 500 randomly punctured codes. Fig. 2 shows the performance of two punctured codes over an additive white gaussian noise (AWGN) channel, one code having the maximum girth average whereas the other having the minimum girth average from among the 500 random punctured codes. It can be seen from this figure that there is a significant difference in performance of the two punctured codes at high SNR. The punctured code with the minimum girth has a higher error floor as compared with the punctured code with the maximum girth average (there is a 1 dB difference between the two curves at BER of $10^{-6}$). It is therefore clear that the maximum girth average criterion and the method for comparing puncturing patterns is a viable method for selecting good puncturing patterns from an ensemble of random puncturing patterns.

### 3.3 RC regular LDPC codes

In this section, we present an algorithm for obtaining RC-LDPC codes using the heuristic search algorithm. Consider a range of desired code rates $R_1 > R_2 > \cdots > R_J > R$, where $R$ is the code rate of the mother code. Note

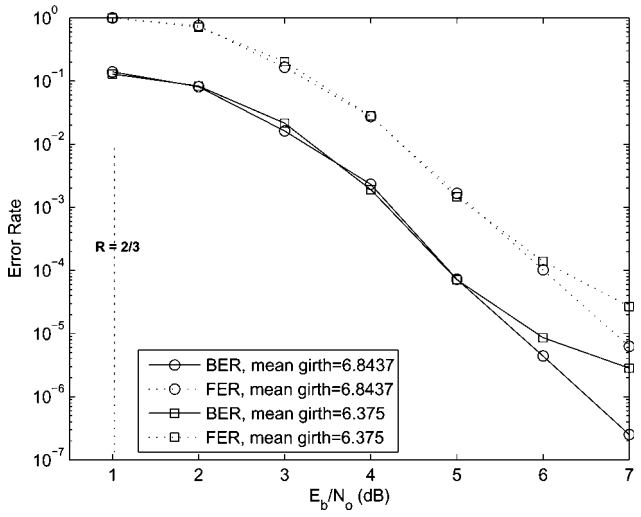



**Figure 2** *Performance of regular LDPC codes with two puncturing patterns that result in punctured codes with the maximum and the minimum girth average, $n = 256$ and 64 parity bits being punctured*

that a code with rate $R_j$ is obtained by puncturing $p_j$ bits from the mother code with rate $R$. Also, a $R_{j+1}$-rate code can be obtained from the $R_j$-rate code by puncturing $p_{j+1}$ bits, which is a subset of the $p_j$ bits punctured to obtain the code of rate $R_j$. Regular RC-LDPC codes that utilise puncturing patterns selected using the heuristic search algorithm can be obtained by the following algorithm

1. Select the number of punctured parity bits $p_1$ which yield a rate-$R_1$ code. Generate random puncturing patterns of size $p_1$. Perform heuristic search to obtain the best puncturing pattern corresponding to $p_1$ punctured bits.

2. Puncturing patterns for obtaining the rate-$R_2$ can be obtained by selecting random subsets of size $p_2$ from the pattern selected in the previous step (of size $p_1$), and performing heuristic search over them.

3. The puncturing patterns required to obtain the codes with rates $R_3$, .., $R_J$ may be obtained in a similar manner as in step 2.

## 4 Semi-Random RC-LDPC Codes

In this section, we consider the design of RC-LDPC codes based on the semi-random structure through puncturing and/or extending.

### 4.1 Puncturing

The structure of a semi-random parity-check matrix is shown in (1). The submatrix on the left corresponds to the parity bits, whereas the submatrix on the right corresponds to the data bits. The puncturing of an LDPC code involves the deletion of coded bits from the codeword and replacing them by erasures at the decoder. Since the parity bits have degree 2, while the degree of the data bits is >2, we chose to puncture some of the parity bits to minimize the degradation in the performance because of puncturing. In the following, we examine three puncturing schemes, namely, 'alternate' which refers to puncturing alternate parity bits, 'successive' which refers to puncturing successive parity bits, and 'random' which refers to puncturing a random pattern of parity bits.

Fig. 3 shows the message passing during the decoding of punctured semi-random LDPC codes. It can be seen that for the case of successive or random puncturing, a successive group of parity bits may be erased. As the decoding proceeds iteratively, after the first (and successive) iteration(s) the log-likelihood ratio (LLR) values of the punctured parity bits for the case of alternate puncturing will be larger than the LLR values of punctured parity bits using successive or random puncturing. It is well known that a large LLR magnitude for a variable node implies higher reliability as compared with the reliability of a variable node with a smaller LLR magnitude [24]. Hence, the LLRs of the punctured parity bits for the case of

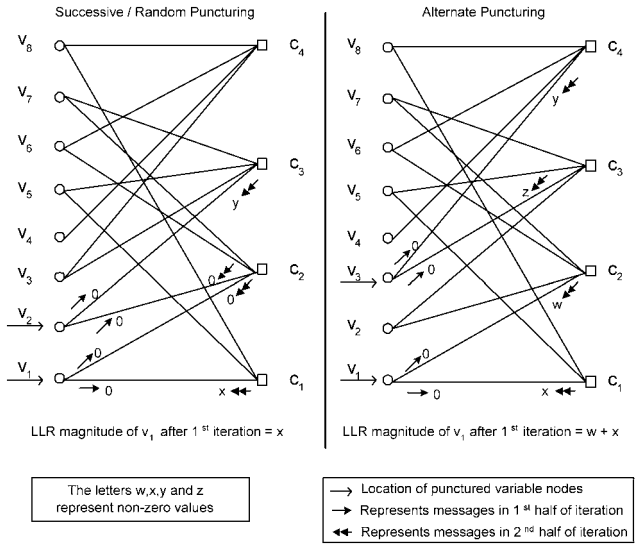



**Figure 3** *Message-passing during decoding of punctured semi-random LDPC code corresponding to the* ***H*** *matrix of (1)*

alternate puncturing converge in a smaller number of iterations as compared with the other two puncturing schemes, causing the LLRs of data bits to converge to their correct values (in the probabilistic sense) in a smaller number of iterations. This observation is confirmed in Fig. 4 which shows the evolution of LLR magnitudes of the punctured bits for the different puncturing schemes. In this figure, we can see that the average LLR values of the parity bits punctured 'alternately' is larger than that for the other puncturing schemes. Therefore it is expected that the performance of the 'alternately' punctured semi-random LDPC codes is better than punctured semi-random LDPC codes that are obtained by using other puncturing schemes, which is confirmed by the simulation results shown in Fig. 5.

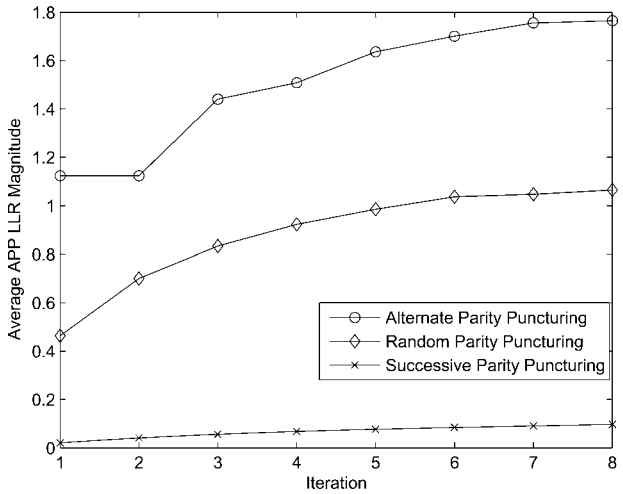



**Figure 4** *Evolution of LLR magnitudes of punctured variable nodes for semi-random LDPC Code, (mother code: n = 256, R = 1/2, punctured code: R = 2/3) at $E_b/N_0 = 2.5$ dB*

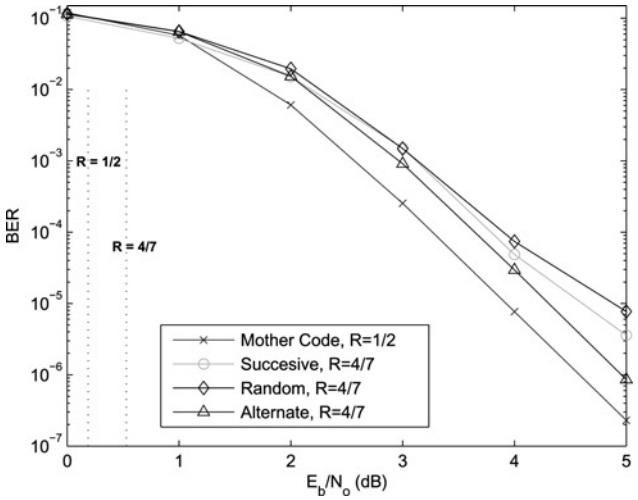



**Figure 5** *Performance of punctured semi-random LDPC codes over the AWGN channel, n = 256, max-iterations = 50*

For illustration purposes, we compare punctured semi-random LDPC codes with randomly punctured regular-(3,6) codes. Note that the block length of the mother code and code rates obtained through puncturing are the same as in [3]. The semi-random LDPC codes are constructed based on the guidelines presented in [19], and the four loops are removed using the algorithm presented in [25]. For the semi-random codes, results for the alternate and random puncturing patterns are shown, whereas for regular codes the results are shown for the random puncturing pattern. Since the simulation is restricted to the low SNR region, the heuristic search algorithm proposed in Section 3 is not used for the design of the randomly punctured regular codes.

The comparison results for AWGN and the uncorrelated Rayleigh fading channels are shown in Figs. 6 and 7, respectively. In these figures, Alternate denotes alternate puncturing pattern and Random denotes a random pattern of parity bits. From these figures, it can be seen that while the mother codes for the regular-(3,6) and the semi-random codes give similar performance, the performance of the punctured regular-(3,6) code is worse that that of the punctured semi-random code, and the difference in performance increases with increasing the number of punctured parity bits. This is because the punctured bits in the semi-random code correspond to degree-2 nodes, whereas they correspond to degree-3 nodes in the regular code. While one punctured bit of the semi-random code affects two check nodes, the puncturing of one bit of the regular-(3,6) code affects three check nodes. This makes the RC semi-random codes obtained via puncturing outperform those obtained by puncturing regular codes, when the mother codes of both have similar performance. Furthermore, the performance gain of alternately punctured semi-random codes over the randomly punctured semi-random codes increases with increasing the number of




doi: 10.1049/iet-com:20060513

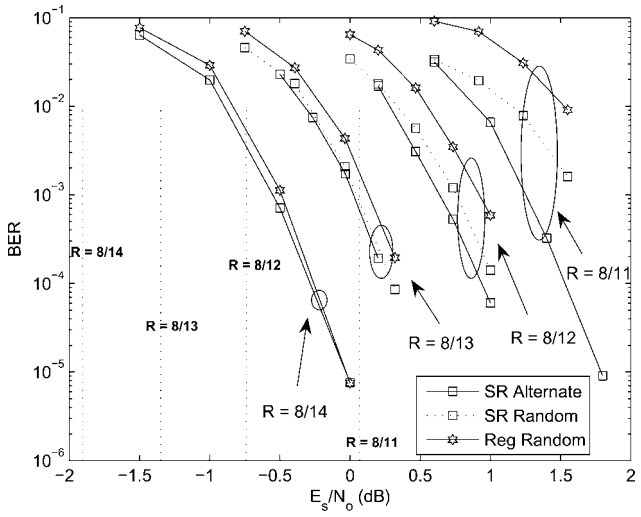



**Figure 6** *Performance of punctured semi-random and regular codes over the AWGN channel with code rates – from left to right – 8/14, 8/13, 8/12 and 8/11, mother code is of rate 8/14 with n = 1792, SR denotes semi-random LDPC, 'Reg' denotes regular LDPC, 'p' denotes the number of punctured parity bits, 'p = 0' denotes the mother code, max-iterations = 50*

punctured parity bits. This is due to the reason illustrated in Fig. 3, where the puncturing effect becomes more severe as the number of punctured bits increase.

It should be noted that the heuristic search algorithm that maximises the average girth in Section 3 can be used to compare randomly punctured semi-random LDPC codes.

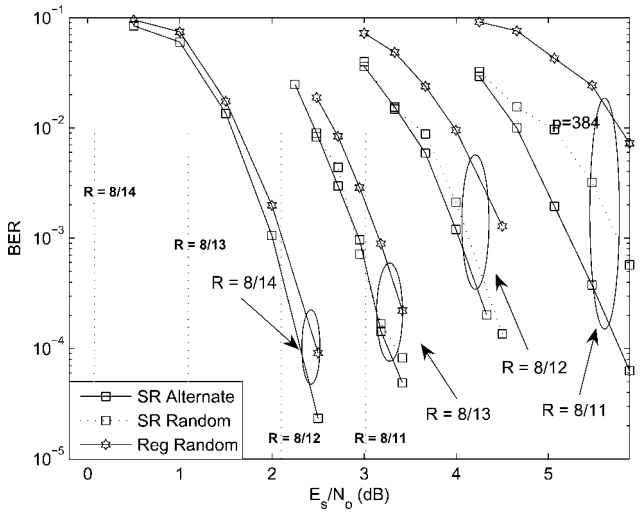



**Figure 7** *Performance of punctured semi-random and regular codes over the Rayleigh fading channel with code rates – from left to right – 8/14, 8/13, 8/12 and 8/11, mother code is of rate 8/14 with n = 1792, SR denotes semi-random LDPC, 'Reg' denotes regular LDPC,, 'p' denotes the number of punctured parity bits, 'p = 0' denotes the mother code, max-iterations = 50*

However, the code(s) that would be selected by this algorithm would outperform other randomly punctured codes only in the high SNR region. Moreover, the results presented show that the alternate puncturing outperforms random puncturing in the region of low SNR, and therefore the application of the heuristic search algorithm to punctured semi-random LDPC codes is obviated.

From our observation through numerical experiments, it was concluded that the following guidelines are suitable for the design of punctured LDPC codes:

- It is desirable to puncture variable nodes of the smallest degree in order to minimise the performance degradation. This can be applied for LDPC codes with non-uniform variable node degree distribution such as semi-random LDPC codes.

- If the degrees of the variable node are equal, puncturing can be performed such that the average girth of the punctured code is maximised. This can be achieved via the heuristic search algorithm presented in Section 3.

## 4.2 Extending

For a matrix $\boldsymbol{H}$ of size $(m \times n)$ representing the original code, each level of extension of size $u$ (adding $u$ parity bits) will add $u$ additional rows and $u$ addtional columns to $\boldsymbol{H}$. The extended matrix $\boldsymbol{H}_{\text{ext}}$ will be of size $(m+u) \times (n+u)$. As an example, Fig. 8 shows the $(n = 11, k = 7)$ parity-check matrix obtained after extending the $(n = 8, k = 4)$ semi-random LDPC matrix of (1) by adding three parity bits. For one level of extension of a semi-random LDPC code, the following steps are proposed.

- Generate $\boldsymbol{H}_{p_{\text{new}}}$ of size $(m+u) \times (m+u)$, with the particular 'staircase' structure that is required for the deterministic part of a semi-random parity-check matrix as in (1).

- Generate $\boldsymbol{H}_{d_{\text{new}}}$ which is a vertical concatenation of two sub-matrices: $\boldsymbol{H}_d$ from the unextended matrix, and a matrix $\boldsymbol{H}_{\text{sparse}}$ of size $(u) \times (n-m)$.

$$H_{ext} = \left(\begin{array}{ccccccc|cccc} 1&0&0&0&0&0&0&1&0&0&1\\ 1&1&0&0&0&0&0&0&1&1&0\\ 0&1&1&0&0&0&0&1&0&1&0\\ 0&0&1&1&0&0&0&0&1&0&1\\ 0&0&0&1&1&0&0&1&0&1&0\\ 0&0&0&0&1&1&0&0&1&1&0\\ 0&0&0&0&0&1&1&1&0&1&0 \end{array}\right) \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \rightarrow H_d \\ \\ \\ \\ \rightarrow H_{sparse} \end{array}$$

$H_{p_{new}}$ $H_{d_{new}}$

**Figure 8** *Illustrative example for extending of semi-random LDPC codes*

• The concatenation of $\boldsymbol{H}_{p_{\text{new}}}$ and $\boldsymbol{H}_{d_{\text{new}}}$ yields a semi-random parity-check matrix with one level of extension (Fig. 8).

The submatrix $\boldsymbol{H}_{\text{sparse}}$ is the only part of the extended matrix that can be designed using different methods, since the rest of the matrix is either identical to parts of the unextended matrix, or follows a deterministic construction approach to comply with the standard format of a semi-random parity-check matrix. In the following, we propose two schemes for the design of the $\boldsymbol{H}_{\text{sparse}}$ matrix. One scheme is reffered to as the 'Extended-identity' approach since it involves the use of identity matrices, and the other scheme is called the 'Extended-Permuted' approach since it involves the use of a matrix which is a random permutation of a particular matrix. The two methods are discussed as follows.

(1) *Extended-identity approach:* The motivation of using identity matrices for the design of $\boldsymbol{H}_{\text{sparse}}$ is that it leads to a simple and deterministic method of extending a semi-random parity-check matrix. It maintains the sparseness of the extended matrix, and does not lead to the creation of small loops. In this approach, $\boldsymbol{H}_{\text{sparse}}$ for each level of extension is formed by the concatenation of an identity matrix of size $u \times u$ and a matrix of zeros of size $u \times (n - m - u)$, where the size of the non-extended matrix is $m \times n$. Consider extending a semi-random LDPC code with a parity-check matrix of the form

$$\boldsymbol{H} = [\boldsymbol{H}_p | \boldsymbol{H}_d] \tag{8}$$

Using the extended-identity approach, the parity-check matrix of the extended code after the first-level extension follows the form

$$\boldsymbol{H}_{\text{ext},1} = \left( \begin{array}{ccc|c} 1 & & 0 & \boldsymbol{H}_d \\ 1 & 1 & & \\ & \ddots & \ddots & \boldsymbol{I} \quad \boldsymbol{0} \quad \boldsymbol{0} \\ 0 & & 1 \quad 1 & \end{array} \right) \tag{9}$$

where $\boldsymbol{I}$ is an identity matrix of size $u \times u$. Continuing on the same manner, the parity-check matrix of the extended code after the second-level extension is given by

$$\boldsymbol{H}_{\text{ext},2} = \left( \begin{array}{ccc|c} 1 & & 0 & \boldsymbol{H}_d \\ 1 & 1 & & \\ & \ddots & \ddots & \boldsymbol{I} \quad \boldsymbol{0} \quad \boldsymbol{0} \\ & \ddots & \ddots & \boldsymbol{0} \quad \boldsymbol{I} \quad \boldsymbol{0} \\ 0 & & 1 \quad 1 & \end{array} \right) \tag{10}$$

The extended-identity approach can be used as explained above to yield extended semi-random LDPC codes of any size.

(2) *Extended-permuted approach:* The extended-identity approach constructs extended semi-random LDPC codes which have only one connection between the additional parity-check equations and the variable nodes representing the data bits. Intuitively, increasing the connections between the additional parity-check equations and the variable nodes – while maintaining the sparseness of the matrix – is expected to lead to a better performance. This notion leads to the extended-permuted approach, in which $\boldsymbol{H}_{\text{sparse}}$ for each level of extension is constructed by the column permutations of a particular matrix $\boldsymbol{H}_{\text{perm}}$. This approach follows the method of constructing regular codes introduced by Gallager [2], where a parity-check matrix of an LDPC code is formed by the vertical concatenation of a number of submatrices. An example of the matrix $\boldsymbol{H}_{\text{perm}}$ which gives good empirical is a matrix in which each row contains two ones as follows

$$\boldsymbol{H}_{\text{perm}} = \begin{pmatrix} 1 & 1 & 0 & 0 & . & . \\ 0 & 1 & 1 & 0 & . & . \\ 0 & 0 & 1 & 1 & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \end{pmatrix} \tag{11}$$

The matrix $\boldsymbol{H}_{\text{sparse}}$ for each level of extension corresponds to a particular column permutation of $\boldsymbol{H}_{\text{perm}}$ in (11). We have also tested extending with a $\boldsymbol{H}_{\text{perm}}$ matrices that contain three and four ones in each row, but the performance is very similar to the $\boldsymbol{H}_{\text{perm}}$ with two ones in each row. Therefore the latter was preferred since it enables lower encoding and decoding complexities.

Consider a semi-random LDPC code whose parity-check matrix of the form in (8), the parity-check matrix of the extended code with the first-level extension employing the extended-permuted approach follows the form

$$\boldsymbol{H}_{\text{ext},1} = \left( \begin{array}{ccc|c} 1 & & 0 & \boldsymbol{H}_d \\ 1 & 1 & & \\ & \ddots & \ddots & \boldsymbol{H}_{\text{perm},1} \\ 0 & & 1 \quad 1 & \end{array} \right) \tag{12}$$

Continuing on the same manner, the parity-check matrix of the extended code with the second-level extension is given by

$$\boldsymbol{H}_{\text{ext},2} = \left( \begin{array}{ccc|c} 1 & & 0 & \boldsymbol{H}_d \\ 1 & 1 & & \\ & \ddots & \ddots & \boldsymbol{H}_{\text{perm},1} \\ & \ddots & \ddots & \\ 0 & & 1 \quad 1 & \boldsymbol{H}_{\text{perm},2} \end{array} \right) \tag{13}$$




doi: 10.1049/iet-com:20060513

Note that the former approach creates four loops during the extending process, which have to be removed using the loop removal algorithm [25]. This leads to higher complexity of the extended-permuted approach, whereas the extended-identity approach has lower complexity because the format for extension is deterministic and the extension does not result in the creation of four loops.

(3) *Numerical results:* Fig. 9 shows the performance of the two proposed extending approaches over the AWGN channel. From this figure, we can see that the extended-permuted approach offers a 0.2–0.3 dB performance advantage over the extended-identity approach. In addition, the difference in performance between the extended-permuted approach and the extended-identity approach increases with increasing the levels of extension. This can be explained as follows. An extended matrix resulting from extension using the extended-permuted approach is relatively denser (the parity-check matrix has more ones) as compared with an extended matrix resulting from extension using the extended-identity approach. This is due to the fact that extended-identity approach adds a single one per row in $\boldsymbol{H}_{\text{sparse}}$, whereas the extended-permuted adds two ones per row $\boldsymbol{H}_{\text{sparse}}$. This leads to a slightly larger average variable node degree for the extended-permuted approach as compared with the extended-identity approach. From simulations, it was found that the average variable node degree of the lowest rate matrices used is 3.1 for the extended-identity approach and 3.4 for the extended-permuted approach. As explained above, the difference in the average variable node degree leads to a slightly better performance for the extended-permuted approach, since variable nodes of comparatively larger degrees are connected to more check nodes, which improves the performance of the iterative decoder.

## 5 Type-II hybrid ARQ

In this section, we present the results for type-II hybrid ARQ systems employing the RC-LDPC codes designed in Sections 3 and 4. For each SNR point, simulation is stopped when at least 200 codeword errors are obtained. Figs. 10 and 11 show the throughput for ARQ schemes employing punctured semi-random and regular LDPC codes, over AWGN and uncorrelated Rayleigh fading channels, respectively. The mother code is a rate-8/14 code with a block length of 1792 bits from which codes of rates 8/13, 8/12 and 8/11 are obtained via puncturing. We observe that codes based on the semi-random family of LDPC codes outperform codes based on the regular-(3,6) family. This is because punctured semi-random LDPC codes outperfom punctured regular-(3,6) codes, as was shown in Figs. 6 and 7.

Fig. 12 shows the throughput of ARQ schemes employing the best-performing semi-random RC-LDPC codes designed in this paper (using alternate puncturing and extended-permuted extending), and regular codes from [3] over the AWGN channel. The mother code is a rate-8/14 code with a block length of 1792 bits. There are 1024 data bits in each frame. Codes with rates 8/13 to 8/11 are designed via puncturing, whereas codes with rates 8/15 to 8/20 are designed via extending. It can be seen that the ARQ schemes employing semi-random codes outperform those employing regular codes at high

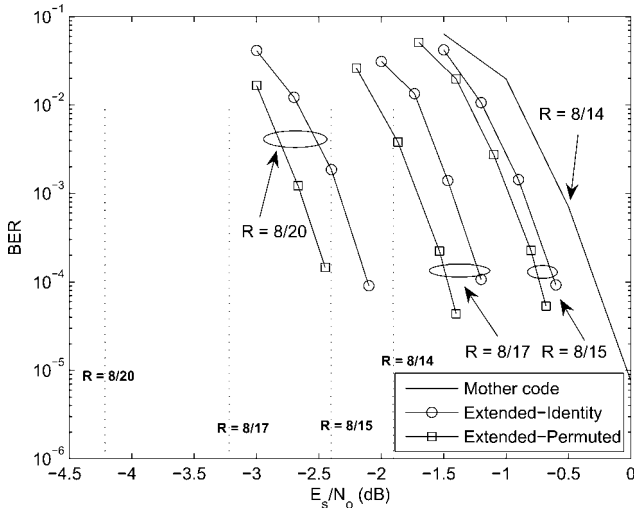



**Figure 9** *Performance of extended semi-random LDPC codes using the extended-identity and extended-permuted approaches over the AWGN channel, code rates – from right to left – 8/14, 8/15, 8/17 and 8/20, mother code is of rate 8/14 with n = 1792*

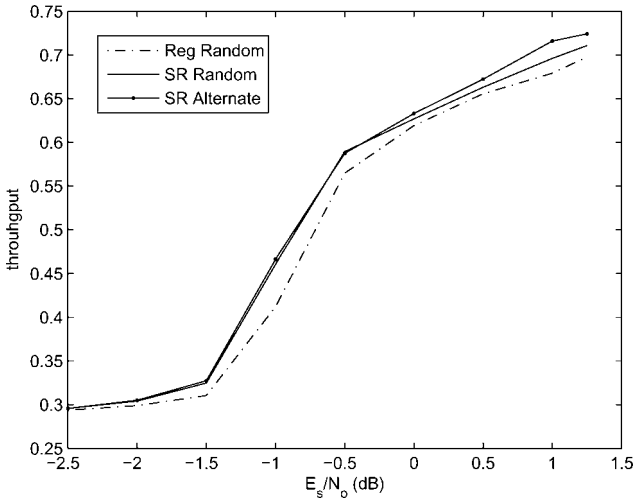



**Figure 10** *Throughput comparison of ARQ schemes based on punctured semi-random LDPC and regular LDPC codes over the AWGN channel, the mother code is a rate-8/14 code with n = 1792 (SR: semi-random codes, Reg: regular-(3,6) codes, Alternate: alternate puncturing, Random: random punctured, max-iterations = 50)*

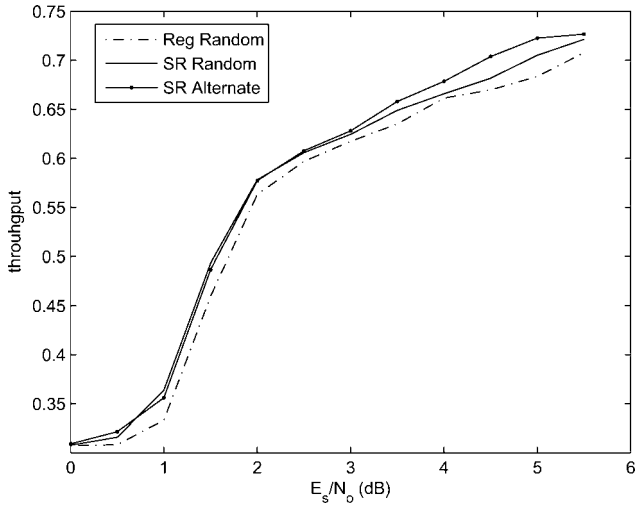



**Figure 11** *Throughput comparison of ARQ schemes based on punctured semi-random LDPC and regular LDPC codes over the Rayleigh fading channel, the mother code is a rate-8/14 code with n = 1792, (SR: semi-random codes, Reg: regular-(3,6) codes, Alternate: alternate puncturing, Random: random punctured, max-iterations = 50)*

SNR by up to 0.3–0.4 dB. At high SNR, the codes obtained via puncturing dominate the performance, and punctured semi-random codes significantly outperform punctured regular-(3,6) codes as shown in Fig. 6. Also, it can be seen that both curves reach the maximum throughput value of ~0.727 at the highest SNR. This is the maximum attainable throughput since it is the maximum code rate ($8/11 \simeq 0.727$) among the family of RC codes being employed.

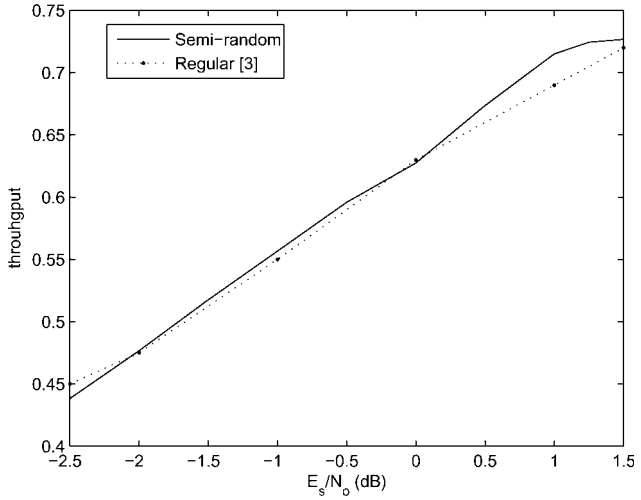



**Figure 12** *Throughput comparison of ARQ schemes based on the proposed semi-random RC-LDPC codes designed in this paper and regular LDPC codes from [3], the mother code is a rate-8/14 code with n = 1792, code rates 8/13 to 8/11 are obtained through alternate puncturing and code rates 8/15 to 8/20 are obtained through extending using the extended-permuted approach*

## 6 Conclusions

In this paper, an algorithm was proposed for the design of punctured regular LDPC codes that have low error floor. The algorithm is based on the concept of maximising the girth average of the underlying Tanner graph of the code. Since increasing the girth of an LDPC code leads to an improved performance, the algorithm selects the puncturing patterns that result in punctured codes with a large girth average. Simulation results verify that the puncturing patterns selected using this algorithm outperform randomly punctured codes.

Furthermore, a puncturing pattern was proposed for the design of punctured semi-random LDPC codes that leads to punctured codes that outperform randomly punctured semi-random LDPC codes. Two approaches for designing extended semi-random RC-LDPC codes were also proposed. A type-II hybird ARQ scheme based on the semi-random RC-LDPC codes designed in this paper was shown to outperform an existing scheme based on regular RC-LDPC codes. Additionally, the proposed hybrid ARQ scheme based on semi-random RC-LDPC codes offers the major advantage of linear-time encoding, in addition to performance gains.

## 7 Acknowledgments

The authors would like to acknowledge the support provided by King Fahd University of Petroleum and Minerals (KFUPM) under Grant FT040005.

## 8 References

[1] HAGENAUER J.: 'Rate-compatible punctured convolutional codes (RCPC codes) and their applications,', *IEEE Trans. Inf. Theory*, 1988, **36**, pp. 389–400

[2] GALLAGER R.G.: 'Low-density parity-check codes,', *IRE Trans. Inf. Theory*, 1962, **IT-8**, pp. 21–28

[3] LI J., NARAYANAN K.: 'Rate-compatible low-density parity-check codes for capacity-approaching ARQ scheme in packet data communications'. Int. Conf. Comm., Internet, and Info. Tech. (CIIT), November 2002

[4] YAZDANI M., BANIHASHEMI A.: 'On construction of rate-compatible low-density parity-check codes', *IEEE Commun. Lett.*, 2004, **8**, (3), pp. 159–161

[5] HU X.-Y., ELEFTHERIOU E., ARNOLD D.-M.: 'Regular and irregular progressive edge-growth tanner graphs', *IEEE Trans. Inf. Theory*, 2005, **51**, pp. 386–398



doi: 10.1049/iet-com:20060513

[6] HA J., KIM J., McLaughlin S.: 'Rate-compatible puncturing of low-density parity-check codes', *IEEE Trans. Inf. Theory*, 2004, **50**, (11), pp. 2824–2836

[7] CHUNG S., FORNEY G.D. JR., RICHARDSON T.J., URBANKE R.: 'On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit', *IEEE Commun. Lett.*, 2001, **5**, pp. 58–60

[8] HA J., KIM J., KLINC D., McLaughlin S.: 'Rate-compatible punctured low-density parity-check codes with short block lengths', *IEEE Trans. Inf. Theory*, 2006, **52**, (2), pp. 728–738

[9] BI D., PEREZ L.: 'Rate-compatible low-density parity-check codes with rate-compatible degree profiles', *IEE Electron. Lett.*, 2006, **42**, (1), pp. 41–43

[10] McGowan J., WILLIAMSON R.: 'Loop removal from LDPC codes'. Information Theory Workshop, Paris, France, 31 March–4 April 2003

[11] ALSUWAIYEL M.H.: 'Algorithms: Design Techniques and Analysis' (World Scientific, 1999)

[12] LIN S., COSTELLO D.: 'Error Control Coding' (Prentice Hall, 2004, 2nd edn.)

[13] RICHARDSON T.J., URBANKE R.: 'Design of capacity approaching irregular low-density parity-check codes', *IEEE Trans. Inf. Theory*, 2001, **47**, (2), pp. 619–637

[14] LUBY M.G., MITZENMACHER M., SHOKROLLAHI M.A., SPIELMAN D.A.: 'Improved low-density parity-check codes using irregular graphs', *IEEE Trans. Inf. Theory*, 2001, **47**, (2), pp. 585–598

[15] CLARK G., CAIN J.: 'Error-correcting coding for digital communications' (Plenum Press, 1981)

[16] MACKAY D.J.C., NEAL R.M.: 'Good error-correcting codes based on very sparse matrices'. Cryptography and Coding. 5th IMA Conf. in BOYD C. (ED.) LNCS Berlin, 1995, vol. 1025, (Springer), pp. 100–111

[17] RICHARDSON T.J., URBANKE R.: 'Efficient encoding of low-density parity-check codes', *IEEE Trans. Inf. Theory*, 2001, **47**, pp. 638–656

[18] JOHNSON S., WELLER S.: 'A Family of irregular LDPC codes with low encoding complexity', *IEEE Commun. Lett.*, 2003, **7**, pp. 79–81

[19] PING L., LEUNG W.K., PHAMDO N.: 'Low-density parity-check codes with semi-random parity check matrix', *IEE Electron. Lett.*, 1999, **35**, (1), pp. 38–39

[20] MAO Y., BANIHASHEMI A.H.: 'A heuristic search for good low-density parity-check codes at short block lengths'. IEEE Int. Conf. Communications, June 2001

[21] ETZION T., TRACHTENBERG A., VARDY A.: 'Which codes have cycle- free tanner graphs?', *IEEE Trans. Inf. Theory*, 1999, **45**, pp. 2173– 2181

[22] BALAKRISHNAN V.K.: 'Schaum's outline of graph theory' (McGraw-Hill, 1997)

[23] MORELOS-ZARAGOZA R.H.: 'The art of error correcting coding' (John Wiley, 2002)

[24] BARRY J.: 'Digital communication' (Springer, 2003, 3rd edn.)

[25] ZAHEER S.F.: 'Improved rate-compatbile low-density parity-check codes with applications to wireless channels', MS Thesis, King Fahd University of Petroleum and Minerals, Dhahran, KSA, 2006