Practical Interleavers for Systematic Repeat-Accumulate Codes

Sarah J. Johnson & Steven R. Weller
School of Electrical Engineering and Computer Science
The University of Newcastle
Callaghan 2308, NSW Australia
email: {sarah.johnson,steven.weller}@newcastle.edu.au

***Abstract*— In this paper we design practical interleavers for systematic repeat-accumulate (RA) codes. Despite their simple description the new interleavers are shown to outperform traditional interleavers, over a wide range of lengths and rates.**

I. Introduction

Repeat-accumulate (RA) codes are simultaneously a class of simple "turbo-like" codes [1] and a class of low-density parity-check (LDPC) codes. This dual representation of RA codes allows the flexibility to use a turbo code representation for the encoding and an LDPC code representation for the decoding, gaining the benefits of both schemes.

When viewed as a serially concatenated turbo code, the two constituent codes of an RA code are a rate-$\frac{1}{q}$ repetition code and a rate-1 $\frac{1}{1+D}$ convolutional code, called an accumulator, with an interleaver between them. While the interleaver of a turbo code can be chosen randomly, and indeed for long codes this produces very good performance results, a randomly chosen interleaver is more challenging to implement in practice. For this reason many deterministic interleavers have been considered for turbo codes, from the simple, and general, row-column interleaver to interleavers aimed at particular constituent convolutional codes, such as the interleaver specified for 3G applications [2].

For RA codes randomly chosen interleavers can also produce effective results, but again can pose implementation challenges. Further, traditional turbo interleavers, which have been designed for different constituent codes, typically do not work well with RA codes. Finally for repeat-accumulate codes decoded as LDPC codes the interleaver must be designed to control the properties of the resulting parity-check matrix. The aim of this paper is to design a simple, and effective, method for specifying RA code interleavers.

II. Repeat-accumulate codes

The encoding circuit of an RA code is shown in Fig 1. Each of the K message bits $\mathbf{m} = [m_1 \dots m_K]$, are repeated q times in the form

$$\mathbf{b} = [m_1, m_1, \dots, m_1 \quad \dots \quad m_K, m_K, \dots, m_K].$$

The length $n = Kq$ interleaver,

$$\Pi = [\pi_1, \pi_1, \dots, \pi_n],$$

then permutes the input bits, $\mathbf{b} = [b_1, b_2, \dots, b_n]$, to give the output bits

$$\mathbf{d} = [d_1, d_2, \dots, d_n] = [b_{\pi_1}, b_{\pi_2}, \dots, b_{\pi_n}].$$

The bits at the output of the interleaver are then combined, modulo 2, in sets of a bits by the combiner before being passed to the accumulator. The Kq/a bits, $\mathbf{r}$, at the output of the combiner, are given by

$$r_i = d_{(i-1)a+1} \oplus d_{(i-1)a+2} \oplus \cdots \oplus d_{ia} \quad i = 1, 2, ..., Kq/a \tag{1}$$

where $\oplus$ represents modulo 2 addition.

Finally, the kq/a parity bits, $\mathbf{p}$ at the output of the accumulator are described by

$$p_i = p_{i-1} \oplus r_i, \quad i = 1, 2, ..., Kq/a \tag{2}$$

and so the final codeword is

$$\mathbf{c} = [m_1, m_2, \dots m_K, p_1, p_2 \dots, p_{Kq/a}].$$

Thus we have a code with length, $N = K(1 + q/a)$, and rate, $R = a/(a + q)$. An RA code is called (q, a)-regular if every message bit is repeated a fixed number, q, times and the combiner always sums a fixed number of a bits.

The parity-check matrix H of an RA code has two parts;

$$H = [H_1, H_2]. \tag{3}$$

Here H_1 is a $Kq/a \times K$, column weight q, row weight a, matrix specified by the interleaver. That is, the rows of H_1 describe the equations in (1). The matrix H_2 in (3) is a $Kq/a \times Kq/a$ matrix which describes (2), and so has ones on the diagonal and subdiagonal and zeros elsewhere.

Like any LDPC code an RA code can be described by a Tanner graph. The Tanner graph of an RA code is directly described by H and consists of Kq/a parity-check nodes and $K + Kq/a$ bit nodes. Fig. 1 shows the Tanner graph for a systematic regular RA code with the message bits appearing at the top of the graph and the parity bits at the bottom.

RA codes are decoded by sum-product decoding on the code's Tanner graph in exactly the same way as for LDPC codes (see e.g. [3] for a description of sum-product decoding). As with LDPC codes, convergence to a valid codeword is easily detected and so it is possible to both halt decoding once a valid codeword has been found and to distinguish between detected and undetected errors.

III. RA interleavers

Designing a regular RA code requires that we design an interleaver,

$$\Pi = [\pi_1, \pi_2, \dots, \pi_n],$$

$$\text{such that} \quad \pi_i \in \{1, 2, \dots, n\}, \ \pi_i \neq \pi_j \ \forall \ i \neq j.$$

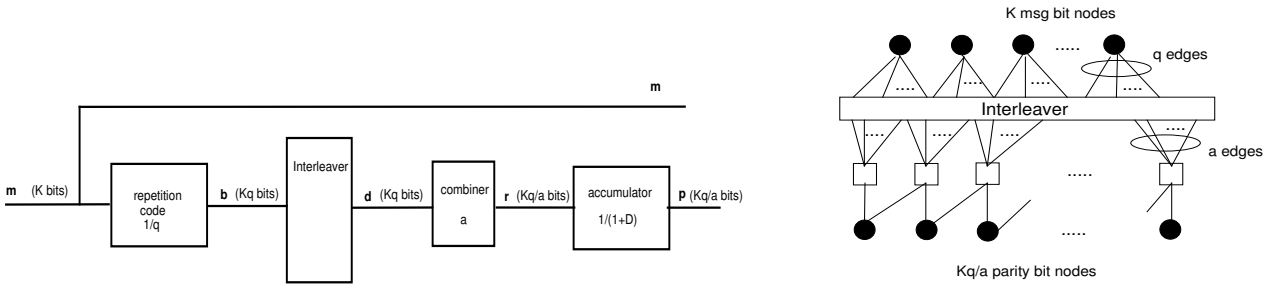

Fig. 1. A systematic RA code viewed as a turbo code, on the left, and as an LDPC code, on the right. Filled circles in the Tanner graph represent bit nodes, while open squares are check nodes.

The i-th entry, π_i, specifies that the $\lceil i/a \rceil$-th row of H has a one in the $\lceil \pi_i/q \rceil$-th column. The notation $\lceil x \rceil$ defines the smallest integer greater than or equal to x.

A. Desired RA interleaver properties

Since H is binary, repeated entries in H are not defined, which is equivalent to requiring that

$$\lceil \pi_i/q \rceil \neq \lceil \pi_j/q \rceil \quad \forall \ i \neq j \ \text{ such that } \ \lceil i/a \rceil = \lceil j/a \rceil.$$

Since we are decoding the code using sum-product decoding on the code's Tanner graph rather than by turbo decoding, the performance of the code is determined by the properties of the Tanner graph, as for an LDPC code. In particular an RA code will perform with fewer errors if small cycles in the code's Tanner graph are avoided. A cycle in the Tanner graph is a sequence of connected nodes which start and end at the same node and contain no other node more than once. The girth of a particular code parity-check matrix is the size of the smallest cycle in the corresponding Tanner graph. A cycle of size 4, called a 4-cycle, will occur if two columns of the parity-check matrix both have a '1' entry in the same two rows.

For an RA code we define two classes of 4-cycles. A type-1 4-cycle will occur if a column in H_1 contains two consecutive ones. A type-2 4-cycle will form if two columns of H_1 contain two entries in common.

Example 1: A type-1 4-cycle, in bold, in the parity-check matrix of a length-10 RA code. Dots represent zero entries.

$$H = \begin{bmatrix} 1 & . & 1 & . & 1 & . & . & . & . & . \\ . & \mathbf{1} & . & 1 & 1 & \mathbf{1} & . & . & . & . \\ 1 & \mathbf{1} & . & . & . & \mathbf{1} & 1 & . & . & . \\ . & . & 1 & 1 & . & . & 1 & 1 & . & . \\ 1 & . & 1 & . & . & . & . & 1 & 1 & . \\ . & 1 & . & 1 & . & . & . & . & 1 & 1 \end{bmatrix}$$

■

Example 2: A type-2 4-cycle, in bold, in the parity-check matrix of a length-10 RA code.

$$H = \begin{bmatrix} \mathbf{1} & . & \mathbf{1} & . & 1 & . & . & . & . & . \\ . & 1 & . & 1 & 1 & 1 & . & . & . & . \\ 1 & 1 & . & . & . & 1 & 1 & . & . & . \\ . & . & 1 & 1 & . & . & 1 & 1 & . & . \\ \mathbf{1} & . & \mathbf{1} & . & . & . & . & 1 & 1 & . \\ . & 1 & . & 1 & . & . & . & . & 1 & 1 \end{bmatrix}$$

■

To avoid type-1 4-cycles requires that

$$\lceil \pi_i/q \rceil \neq \lceil \pi_j/q \rceil \ \ \forall \ \ i \neq j \ \text{ such that } \ \lceil i/a \rceil = \lceil j/a \rceil \pm 1,$$

while to avoid type-2 4-cycles requires that

$$\lceil \pi_j/q \rceil \neq \lceil \pi_i/q \rceil \ \ \forall \ \ i \neq j \ \text{ such that } \ \exists \ \ k, l \ \text{ where}$$

$$\lceil l/a \rceil = \lceil j/a \rceil, \quad \lceil k/a \rceil = \lceil i/a \rceil, \ \text{ and } \ \lceil \pi_l/q \rceil = \lceil \pi_k/q \rceil.$$

Cycles cannot be formed solely within the columns of H_2 and so type-1 and type-2 4-cycles cover all possible 4-cycles in an RA code.

Type-1 4-cycles can be avoided by using a popular turbo interleaver called an S-type interleaver. An S-type interleaver requires that no two entries of Π within S of each other have a value within S of each other [4]. Pseudo-randomly constructed S-type interleavers are called S-random interleavers. Type-1 4-cycles can be avoided by specifying that $S \geq \max(q-1, 2a-1)$ however this can be a more stringent requirement than what is actually required to avoid type-1 4-cycles. More importantly though, an S-type interleaver cannot be used to avoid type-2 4-cycles when $a > 1$.

Another popular turbo interleaver is the row-column, or block, interleaver which, unlike S-random interleavers, has a very simply algorithmic implementation. A length n row-column interleaver is achieved by writing the bits 1 through n into a matrix row-wise and reading them out column-wise. However, the row-column interleaver is particularly bad for RA codes since it can add a large number of 4-cycles when $a > 1$. Example 3 shows the parity-check matrix of a $K = 8$ RA code with a row column interleaver when $q = a = 2$.

Example 3: The parity-check matrix of a length-16, rate-1/2, (2,2)-regular RA code with a row-column interleaver is:

$$H = \begin{bmatrix} 1 & . & 1 & . & . & . & . & . & 1 & . & . & . & . & . & . & . \\ . & . & . & . & 1 & . & 1 & . & 1 & 1 & . & . & . & . & . & . \\ 1 & . & 1 & . & . & . & . & . & 1 & 1 & . & . & . & . & . & . \\ . & . & . & . & 1 & . & 1 & . & . & . & 1 & 1 & . & . & . & . \\ . & 1 & . & 1 & . & . & . & . & . & . & . & . & 1 & 1 & . & . \\ . & . & . & . & . & 1 & . & 1 & . & . & . & . & 1 & 1 & . & . \\ . & 1 & . & 1 & . & . & . & . & . & . & . & . & . & . & 1 & 1 \\ . & . & . & . & . & 1 & . & 1 & . & . & . & . & . & . & 1 & 1 \end{bmatrix}$$

■

We require for RA codes an interleaver which, like a row-column interleaver, is easy to describe, and thus implement, but which avoids 4-cycles in the code. The interleaver we

propose for RA codes, and call an L-type interleaver, is presented in the following.

B. L-type interleavers

Given K, q and the interleaver parameter l, the L-type interleaver first selects the K message bits in order and then selects from the message bits again, skipping l bits ahead after each selection. This process is repeated q times, as given in (4). In practice the bits $iK+1$ to $(i+1)K$, specified in Π_{i+1}, are a type of row-column permutation of bits $(i-1)K+1$ to iK, specified in Π_i, where the bits are written row-wise into a matrix with l columns and read out column wise. Thus the L-type interleaver is the concatenation of $q-1$ of these matrix row-write, column-read, operations and so is very straightforward to implement. The parity-check matrix of a $K=8$, $q=a=l=2$ RA code with an L-type interleaver is shown in Example 4.

Example 4: The parity-check matrix of a length-16, rate-1/2, (2,2)-regular RA code with an L-type(l=2) interleaver is:

$$H = \begin{bmatrix} 1 & 1 & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . \\ . & . & 1 & 1 & . & . & . & . & 1 & 1 & . & . & . & . & . & . \\ . & . & . & . & 1 & 1 & . & . & . & 1 & 1 & . & . & . & . & . \\ . & . & . & . & . & . & 1 & 1 & . & . & 1 & 1 & . & . & . & . \\ 1 & . & 1 & . & . & . & . & . & . & . & . & 1 & 1 & . & . & . \\ . & . & . & . & 1 & . & 1 & . & . & . & . & . & 1 & 1 & . & . \\ . & 1 & . & 1 & . & . & . & . & . & . & . & . & . & 1 & 1 & . \\ . & . & . & . & . & 1 & . & 1 & . & . & . & . & . & . & 1 & 1 \end{bmatrix}$$

■

The implementation complexity of the L-type interleaver is comparable to that of the row-column interleaver, requiring $q-1$ size n/q matrix write/read operations instead of one length n matrix write/read operation. As well as being simple to describe, a benefit of the L-type interleaver is that we can now guarantee some girth properties for the codes. In the following we define by R_i the set of K rows of H_1 corresponding to Π_i.

Lemma 1: A $(3,a)$-regular RA code can be constructed without 4-cycles whenever $K > a^3$ by using an L-type interleaver and setting $l=a$.

Proof: Firstly we note that the rows in each set, R_i, are disjoint and so a 4-cycle will only occur across rows from two different sets. Thus a type-1 4-cycle can only be formed between the last row of one set and the first row of the next. This will only occur if the last a bits of Π_i, which correspond to the last row of R_i, have entries in common with the first a bits of Π_{i+1}, which will be selected from to form the first row of R_{i+1}. The first a bits of Π_{i+1} are selected from the first la bits of Π_i. Since $K > la+a = a^2+a$ the first la and last a bits of Π_i do not overlap and thus type-1 4-cycles are always avoided. Next setting $l=a$ ensures that each of the bits in a row in R_2 are chosen from different rows in R_1, and similarly that each of the bits in a row in R_3 are chosen from different rows in R_2. Further, since $K/l > a$ the interleaver cannot wrap back around the K bits within a single row in R_2. Setting $(K/l)/l > a$ i.e. $K > a^3$ ensures that the interleaver cannot wrap back around the K bits within a single row in R_2 and so type-2 cycles are also avoided. ■

Furthermore, the same construction can be used to guarantee RA codes without 6-cycles. In this case we define a type-1 6-cycle as one containing two accumulator columns, a type-2 6-cycle as one containing one accumulator column, and finally a type-3 6-cycle as one formed solely within H_1.

Lemma 2: A $(3,a)$-regular RA code can be constructed without 6-cycles whenever $K \geq 8a^3$ by using an L-type interleaver and setting $l=2a$.

Proof: Since each set of K rows, R_i corresponding to each Π_i, are disjoint, a 6-cycle must occur across three row sets. Thus a type-1 6-cycle can only be formed between the 2nd last (or last) rows of one set and the first (or respectively second) row of the next. This will only occur if the last $2a$ entries of Π_i, which correspond to the last two rows of R_i, are within the first $2la$ entries of Π_i which will be selected from to form the first two rows of R_{i+1}. Since $K > 2la + 2a = 4a^2 + 2a$ this cannot occur and type-1 6-cycles are always avoided. Next setting $l=2a$ ensures that each of the bits in a row in R_2 are chosen from rows in R_1 separated by at least one row, and similarly that each of the bits in a row in R_3 are chosen from rows in R_2 separated by at least one row. Further, since $K/l > 2a$ the interleaver cannot wrap back around the K bits within two consecutive rows in R_2. Setting $(K/l)/l > 2a$ ensures that the interleaver cannot wrap back around the K bits within within two consecutive rows in R_3 and so type-2 6-cycles are also avoided. Finally a type-3 6-cycle must involve one row from each of the three sets. Given any two overlapping rows in R_1 and R_2 the column entries in the row from R_1 are consecutive while the column entries in the row in R_2 are spaced l apart. Thus any two column entries from the row in R_1 and the row in R_2 are at most $(a-1)+l(a-1) = 2a^2-a-1$ columns apart. However the column entries in the rows in R_3 are at least $l^2 = 4a^2$ columns apart and so we cannot find a row in R_3 which overlaps both of the two rows from R_1 and R_2. Thus type-3 6-cycles are always avoided. ■

For the $a=1$ case we can do even better, and prove girth ≥ 10 codes for $l=2$ and K odd, $K \geq 7$, and girth ≥ 12 codes for $l=3$ and $K = 1,2 \bmod 3$, $K \geq 21$. Unfortunately, however, when $a>1$ the L-type interleaver always adds cycles of size 8. To see this consider that the first row of R_2 contains ones in columns 1 and $l+1$, while a later row in R_2 will contain ones in columns 2 and $2+l$. The columns 1 and 2 share a row in R_1, as do the columns l and $l+1$. These four rows always make up an 8-cycle, and unfortunately there are many 8-cycles formed similarly. Thus while the L-type interleaver will produce good codes for shorter lengths, where a girth of eight is beneficial, it will not be a good choice for long codes where codes with large girth can easily be constructed randomly.

For longer codes we propose a modified L-type interleaver in the following which breaks many of these 8-cycles.

C. Modified L-type interleavers

In this interleaver Π_1 is unchanged and Π_i is formed starting with the same process as previously, that is the bits of Π_{i-1} are written row-wise into a matrix, M_i with l columns and read out column-wise, but additionally the bits from each column are written row-wise into another matrix, A_j, and read out

$$\Pi = [\Pi_1(1), \ldots, \Pi_1(K)],\ \ldots,\ [\Pi_{q-1}(1) + (q-2), \ldots, \Pi_{q-1}(K) + (q-2)], [\Pi_q(1) + (q-1), \ldots, \Pi_q(K) + (q-1)], \quad (4)$$

where

$$\begin{aligned}
\Pi_1 &= [1, 1+q, 1+2q, 1+3q, \ldots, 1+(K-1)q], \\
\Pi_2 &= [\Pi_1(1), \Pi_1(1+l), \Pi_1(1+2l), \ldots, \Pi_1(1+K-l)],\ [\Pi_1(2), \Pi_1(2+l), \Pi_1(2+2l), \ldots, \Pi_1(2+K-l)], \\
&\quad \ldots, [\Pi_1(l), \Pi_1(l+l), \Pi_1(l+2l), \ldots, \Pi_1(l+K-l)], \\
&\vdots \\
\Pi_q &= [\Pi_{q-1}(1), \Pi_{q-1}(1+l), \Pi_{q-1}(1+2l), \ldots, \Pi_{q-1}(1+K-l)],\ [\Pi_{q-1}(2), \Pi_{q-1}(2+l), \Pi_{q-1}(2+2l), \ldots, \Pi_{q-1}(2+K-l)], \\
&\quad \ldots, [\Pi_{q-1}(l), \Pi_{q-1}(l+l), \Pi_{q-1}(l+2l), \ldots, \Pi_{q-1}(l+K-l)].
\end{aligned}$$

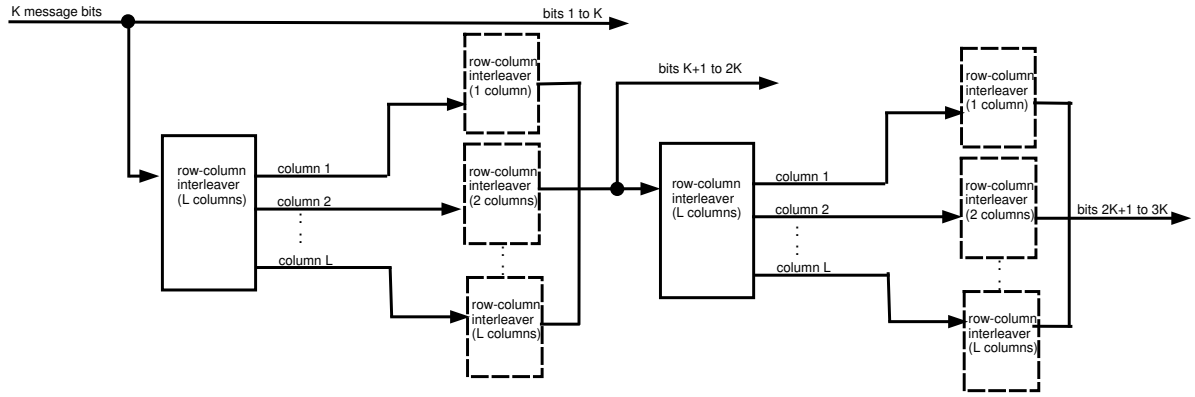

Fig. 2. Block diagram of the encoding circuit for a combined $q = 3$ repetition code and modified L-type interleaver. Removing the dashed row column interleaver blocks gives an L-type interleaver.

column-wise. The j-th column of M_i is written into a matrix A_j with j columns.

The implementation of this second RA interleaver will now require $(q-1)l$ matrix write-read operations instead of $q-1$. However the modified interleaver is still extremely simple to specify requiring just three parameters, l, q and K. Fig. 2 shows graphically the encoding system for a combined repetition code and modified L-type interleaver when $q = 3$.

While this new construction method does break most of the 8-cycles, and, as we will see from performance results, works very well, it can add 4-cycles in the process. While this happens so rarely that the decoding performance of the codes is not effected, it does mean that we cannot present analogous results to Lemmas 1 and 2 for the modified L-type interleavers.

IV. Decoding performance

In this section we consider the decoding performances of RA codes with the proposed interleavers on the binary input additive white Gaussian noise channel. Figs. 3-6 show the performance of the proposed L-type and modified L-type interleavers for a range of code rates and lengths. The performance of the proposed interleavers is compared to the performance of row-column interleavers, randomly constructed interleavers, and pseudo-randomly constructed S-random interleavers. For each presented simulation point at least 200 word error events were observed.

As expected the codes with row-column interleavers perform very poorly at all lengths and rates while the codes with random and S-random interleavers generally perform well. Despite their simple description the L-type interleavers perform extremely well for short RA codes, out-performing codes constructed with the alternative interleaver types, however, as expected they do not perform as well as a randomly constructed interleaver at long code lengths. The modified L-type interleavers however perform extremely well at all the lengths and rates considered. Even for long codes, with length of around 10,000 bits, the modified L-type interleavers perform as well as randomly constructed interleavers, which is not usually the case for structured codes when decoded by the sum-product algorithm.

V. Conclusion

In this paper we have presented a straightforward deterministic construction method for practical RA interleavers. These interleavers are shown to give excellent decoding performances for RA codes over a wide range of lengths and rates. Thus the new interleavers represent an excellent choice over both existing simple interleavers, which they outperform, and randomly constructed interleavers, which they perform at least as well as, with the added advantage of a much simpler description.

References

[1] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for "turbo-like" codes," in *Proc. 36th Allerton Conf. on Communications, Control, and Computing*, Allerton, Illinois, September 1998, pp. 201–210.

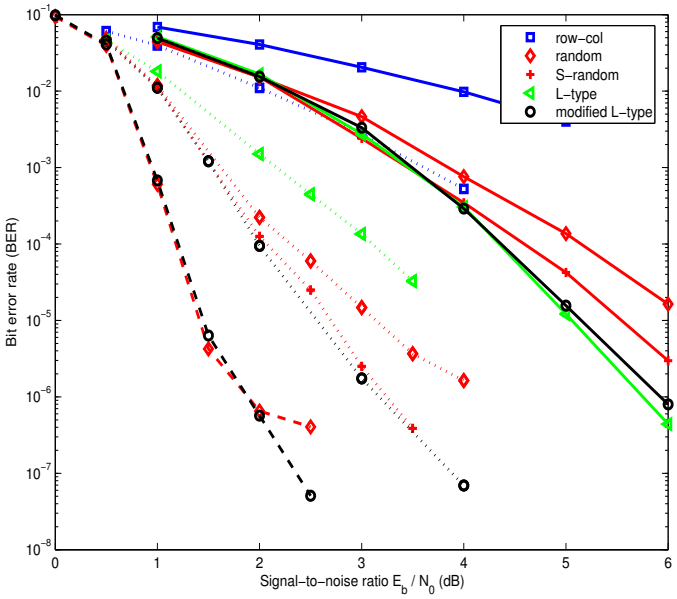

Fig. 3. The error correction performance on an AWGN channel of rate-$1/4$, $q = 3$, $a = 1$ RA codes. The solid lines show the performance of length-196 codes (max 10 iterations of sum-product decoding), the dotted lines show the performance of length-2004 codes (max 100 iterations of sum-product decoding) and the dashed lines show the performance of length-10,000 codes (max 1,000 iterations of sum-product decoding). The L-type codes are constructed with $l = 8$ and 21, while the modified L-type interleavers use $l = 4$, 10 and 30.

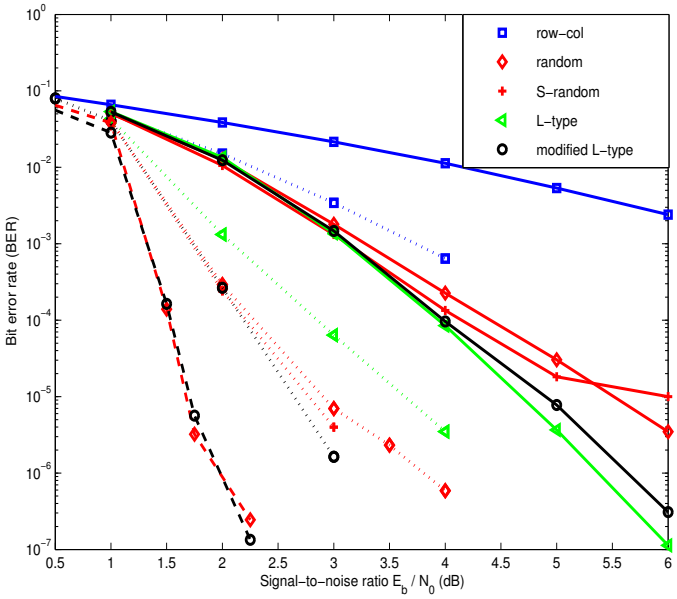

Fig. 4. The error correction performance on an AWGN channel of rate-$1/2$, $q = 3$, $a = 3$ RA codes. The solid lines show the performance of length-222 codes (max 10 iterations of sum-product decoding), the dotted lines show the performance of length-2022 codes (max 100 iterations of sum-product decoding) and the dashed lines show the performance of length-10,000 codes (max 1,000 iterations of sum-product decoding). The L-type codes are constructed with $l = 9$ and 30, while the modified L-type interleavers use $l = 6$, 20 and 30.

[2] "3rd Generation Partnership Project (3GPP); Technical specification group radio access network; Multiplexing and channel coding (FDD), 3GPP TS," [Online]. Available: http://www.3gpp.org.

[3] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, March 1999.

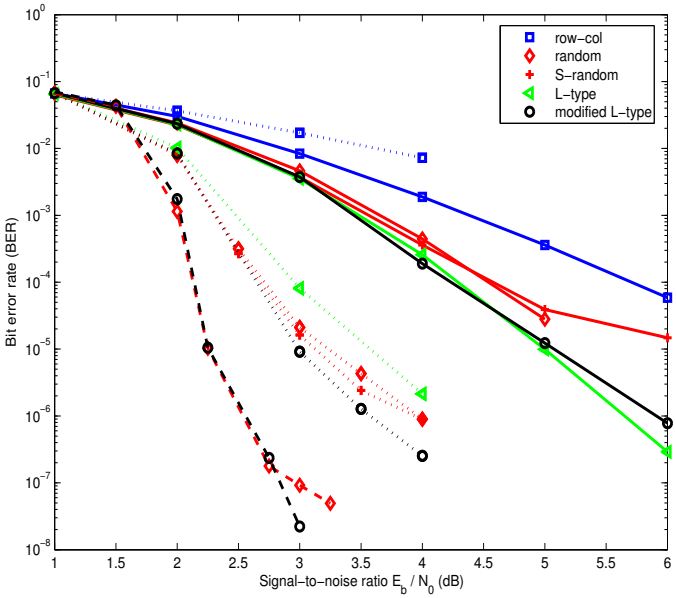

Fig. 5. The error correction performance on an AWGN channel of rate-$2/3$, $q = 3$, $a = 6$ RA codes. The solid lines show the performance of length-189 codes (max 10 iterations of sum-product decoding), the dotted lines show the performance of length-2007 codes (max 100 iterations of sum-product decoding) and the dashed lines show the performance of length-10,002 codes (max 1,000 iterations of sum-product decoding). The L-type codes are constructed with $l = 10$ and 25, while the modified L-type interleavers use $l = 6$, 10 and 30.

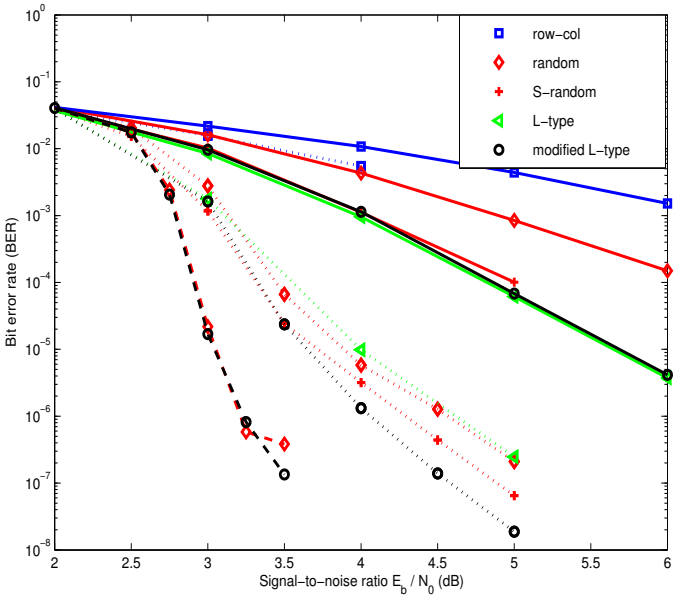

Fig. 6. The error correction performance on an AWGN channel of rate-$4/5$, $q = 3$, $a = 12$ RA codes. The solid lines show the performance of length-195 codes (max 10 iterations of sum-product decoding), the dotted lines show the performance of length-2085 codes (max 100 iterations of sum-product decoding) and the dashed lines show the performance of length-10,000 codes (max 1,000 iterations of sum-product decoding). The L-type codes are constructed with $l = 11$ and 19, while the modified L-type interleavers use $l = 11$, 19 and 30.

[4] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and non-random permutations," Aug. 1995, JPL TDA Progress Report 42-144.