LDPC Block and Convolutional Codes Based on Circulant Matrices

R. Michael Tanner, *Fellow, IEEE*, Deepak Sridhara, Arvind Sridharan, Thomas E. Fuja, *Fellow, IEEE*, and Daniel J. Costello, Jr., *Fellow, IEEE*

***Abstract*—A class of algebraically structured quasi-cyclic (QC) low-density parity-check (LDPC) codes and their convolutional counterparts is presented. The QC codes are described by sparse parity-check matrices comprised of blocks of circulant matrices. The sparse parity-check representation allows for practical graph-based iterative message-passing decoding. Based on the algebraic structure, bounds on the girth and minimum distance of the codes are found, and several possible encoding techniques are described. The performance of the QC LDPC block codes compares favorably with that of randomly constructed LDPC codes for short to moderate block lengths. The performance of the LDPC convolutional codes is superior to that of the QC codes on which they are based; this performance is the limiting performance obtained by increasing the circulant size of the base QC code. Finally, a continuous decoding procedure for the LDPC convolutional codes is described.**

***Index Terms*—Circulant matrices, iterative decoding, low-density parity-check (LDPC) block codes, LDPC convolutional codes, LDPC codes, message-passing, quasi-cyclic (QC) codes.**

I. Introduction

LOW-density parity-check (LDPC) codes have attracted considerable attention in the coding community because they can achieve near-capacity performance with iterative message-passing decoding and sufficiently long block sizes. For example, in [1], Chung *et al.* presented a block length 10^7 (ten million bits) rate-$1/2$ LDPC code that achieves reliable performance—a 10^{-6} bit error rate (BER)—on an additive white Gaussian noise (AWGN) channel with a signal-to-noise ratio (SNR) E_b/N_0 within 0.04 dB of the Shannon limit.

For many practical applications, however, the design of good codes with shorter block lengths is desired. Moreover, most methods for designing LDPC codes are based on random construction techniques; the lack of structure implied by this randomness presents serious disadvantages in terms of storing and accessing a large parity-check matrix, encoding data, and analyzing code performance (e.g., determining a code's distance properties). If the codes are designed with some (algebraic) structure, then some of these problems can be overcome.

In the recent literature, several algebraic methods for constructing LDPC codes have surfaced [2]–[7]. Among these, the one most relevant to this paper is the $[155, 64, 20]$ quasi-cyclic (QC) LDPC code designed by Tanner [2]. This code's minimum distance of 20 (determined by computer search using MAGMA) compares well with that of the best code known with the same block length and rate—a $[155, 64, 28]$ code. Furthermore, the code is $(3, 5)$-regular and has a sparse and elegant constraint graph (or *Tanner graph*) representation that makes it well suited to message-passing algorithms (for example, belief propagation (BP)). The parity-check matrix of the code is composed of blocks of circulant matrices, giving the code a QC property, which can potentially facilitate efficient encoder implementation. Further, the algebraic structure of the code allows for an efficient high-speed very large scale integration (VLSI) implementation.

The main contribution of this paper is to generalize the $[155, 64, 20]$ construction and obtain a large class of QC regular LDPC codes with similar properties. Further, by exploiting the close relationship between QC codes and convolutional codes [8], [9], a convolutional representation is derived from the QC block code. The convolutional representation has a LDPC matrix that retains the graph structure of the original QC code. Thus, the convolutional code is *also* well suited to decoding with message passing. Moreover, with the convolutional code, continuous encoding and decoding is possible—a desirable feature in many applications.

The algebraically constructed QC LDPC codes perform quite well compared to random regular LDPC codes at short to moderate block lengths, while for long block lengths, a randomly constructed regular LDPC code typically performs somewhat better. Moreover, random regular LDPC codes of column weight $j \geq 3$ are *asymptotically* good. In [10], Gallager showed that the cumulative distribution function of the minimum distance of a random ensemble of (N, j, k)[1] regular LDPC codes approaches a step function at some fraction of the block length $\delta(j, k) > 0$ as $N \to \infty$—meaning, "almost all" codes from this ensemble

Manuscript received December 15, 2003; revised August 20, 2004. This work was supported in part by the National Science Foundation under Grants CCF-0205310 and CCF-0231099 and by National Aeronautics and Space Administration under Grant NAG5-12792. The material in this paper was presented in part at the International Symposium on Communications Theory and Applications, Ambleside, U.K., July 2001; the Conference on Information Sciences and Systems Baltimore, MD, March 2001; and the IEEE International Symposium on Information Theory, Lausanne, Switzerland, June/July 2002.

R. M. Tanner is with the University of Illinois at Chicago, Chicago, IL 60607-7128 USA (e-mail: rmtanner@uic.edu).

D. Sridhara was with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN. He is now with the Department of Mathematics, Indian Institute of Science, Bangalore 560012, India (e-mail: dsridhar@math.iisc.ernet.in).

A. Sridharan, T. E. Fuja, and D. J. Costello, Jr. are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: Arvind.Sridharan.1@nd.edu; Thomas.E.Fuja.1@nd.edu; Costello.2@nd.edu).

Communicated by M. P. Fossorier, Associate Editor for Coding Techniques.
Digital Object Identifier 10.1109/TIT.2004.838370

[1]An (N, j, k) regular LDPC code is a code of block length N that is described by a parity-check matrix containing j ones in every column and k ones in every row.

have minimum distances at least as large as $N\delta(j,k)$ when N is large. The algebraic LDPC codes designed in this paper are not asymptotically good in this sense.

To emphasize the close relationship between the QC LDPC block codes and the corresponding convolutional codes, this paper describes the two in parallel. Section II introduces the circulant-based code construction; it also shows how the construction can be modified to generate irregular LDPC codes [11]. *Thresholds* (i.e., the channel SNRs above which the BP decoder is guaranteed to converge on a cycle-free constraint graph [11]) are better for the irregular LDPC codes than for the regular ones. Section III describes some of the properties of the codes and their constraint graphs, including girth, minimum distance, and encoding procedures. The performance of the codes on an AWGN channel with iterative message-passing decoding is examined in Section IV. Section V summarizes the results and concludes the paper.

II. Code Construction

This section describes the means by which the underlying structure of multiplicative groups in the set of integers modulo m may be used to construct LDPCs—both block codes and convolutional codes.

A. Construction of QC LDPC Block Codes

We use the structure of multiplicative groups in the set of integers modulo m to "place" circulant matrices within a parity-check matrix so as to form regular QC LDPC block codes with a variety of block lengths and rates. For prime m, the integers $\{0,1,\ldots,m-1\}$ form a field under addition and multiplication modulo m—i.e., the Galois field GF(m). The nonzero elements of GF(m) form a cyclic multiplicative group. Let a and b be two nonzero elements with multiplicative orders $o(a)=k$ and $o(b)=j$, respectively.[2] Then we form the $j\times k$ matrix P of elements from GF(m) that has as its (s,t)th element $P_{s,t}=b^s a^t$ as follows:

$$P=\begin{bmatrix} 1 & a & a^2 & \ldots & a^{k-1} \\ b & ab & a^2b & \ldots & a^{k-1}b \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ b^{j-1} & ab^{j-1} & a^2b^{j-1} & \ldots & a^{k-1}b^{j-1} \end{bmatrix}. \quad (1)$$

(Here, $0\le s\le j-1$ and $0\le t\le k-1$.)

The LDPC code is constructed by specifying its parity-check matrix H. Specifically, H is made up of a $j\times k$ array of circulant submatrices as shown in the following:

$$H=\begin{bmatrix} I_1 & I_a & I_{a^2} & \ldots & I_{a^{k-1}} \\ I_b & I_{ab} & I_{a^2b} & \ldots & I_{a^{k-1}b} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ I_{b^{j-1}} & I_{ab^{j-1}} & I_{a^2b^{j-1}} & \ldots & I_{a^{k-1}b^{j-1}} \end{bmatrix} \quad (2)$$

where I_x is an $m\times m$ identity matrix with rows cyclically shifted to the left by x positions. The circulant submatrix in position (s,t) within H is obtained by cyclically shifting the rows of the identity matrix to the left by $P_{s,t}$ places. The resulting binary parity-check matrix is of size $jm\times km$, which means the associated code has a rate $R\ge 1-(j/k)$. (The rate may be greater than $1-(j/k)$ due to linear dependence among the rows of H; it is easy to see that there are, in fact, at least $j-1$ dependent rows in H.) By construction, every column of H contains j ones and every row contains k ones, and so H represents a (j,k) regular LDPC code. (We observe here that for the case $j=2$, our construction yields the graph-theoretic error-correcting codes proposed by Hakimi *et al.* in [12].)

The codes constructed using this technique are QC with period k—i.e., cyclically shifting a codeword by one position within each of the k blocks of circulant submatrices (where each block consists of m code bits) results in another codeword.[3]

This construction can be extended to nonprime m. For any integer m, the set of nonnegative integers less than m and relatively prime to m, $\mathbb{Z}_m^*$, forms a multiplicative group. In general, $\mathbb{Z}_m^*$ has order

$$\phi(m)=m\prod_{p|m,p\ prime}(1-1/p)$$

i.e., the Euler "phi" function. Let a and b be two elements of $\mathbb{Z}_m^*$ with orders k and j, respectively.[4] Then the $j\times k$ matrix P and the corresponding parity-check matrix H are obtained as above. As earlier, the binary parity-check matrix H is of size $jm\times km$ and the associated code has rate $R\ge 1-(j/k)$. Since every column of H contains j ones and every row contains k ones, H represents a (j,k) regular LDPC code. Examples of codes constructed in this manner from a prime m are shown in Table I [13], and examples constructed from a nonprime m are shown in Table II.

- Example 1: A $[155,64,20]$ QC code $(m=31)$ [2]
  Elements $a=2$, $b=5$ are chosen from GF(31); then $o(a)=5$, $o(b)=3$, and the parity-check matrix is given by

$$H=\begin{bmatrix} I_1 & I_2 & I_4 & I_8 & I_{16} \\ I_5 & I_{10} & I_{20} & I_9 & I_{18} \\ I_{25} & I_{19} & I_7 & I_{14} & I_{28} \end{bmatrix}_{(93\times 155)}$$

  where I_x is a 31×31 identity matrix with rows shifted cyclically to the left by x positions. The parity-check matrix H has rank 91 (determined using Gaussian elimination), so that H describes a rate $R=64/155\approx 0.4129$ code. The Tanner graph resulting from H is shown in Fig. 1, where the code bit nodes correspond to the columns of H and the constraint (parity-check) nodes correspond to the rows of H. (See [14] for an explanation of Tanner graphs.) The length of the shortest cycle (i.e., the *girth*) of the Tanner graph is 8. The sparse Tanner graph along with the large girth for a code of this size makes the code

[2]There exists such elements if k and j divide $\phi(m)=m-1$, the order of the multiplicative group.

[3]Strictly speaking, the word "quasi-cyclic" means the code has the property that when a codeword is cyclically shifted by k positions another codeword is obtained; to observe this property in the codes constructed above, the bit positions in each codeword must be permuted to a different order than the one indicated by the construction.

[4]As before, if j and k are prime factors of $\phi(m)$, then such elements always exist.

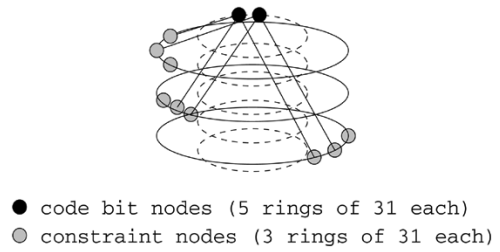

Fig. 1. Tanner graph for a $[155, 64, 20]$ QC code.

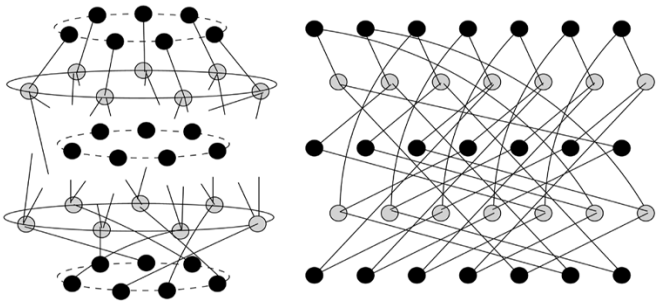

Fig. 2. Tanner graph for a $[21, 8, 6]$ QC code.

ideal for graph-based message-passing decoding. The associated code has minimum distance $d_{\min} = 20$ (determined using MAGMA) and is a regular $(3, 5)$ LDPC code. As noted before, this compares well with the minimum distance $d_{\min} = 28$ of the best linear code known with the same rate and block length. It can be shown that the Tanner graph of this code has diameter 6, which is the best possible for a $(3, 5)$ regular bipartite graph of this size. Also, the Tanner graph has girth 8, while an upper bound on girth is 10 (from the tree bound; see Section III-D).

- Example 2: A $[21, 8, 6]$ QC code $(m = 7)$.
  Elements $a = 2$, $b = 6$ are chosen from GF(7); then $o(a) = 3$, $o(b) = 2$, and the parity-check matrix is given by

$$H = \begin{bmatrix} I_1 & I_2 & I_4 \\ I_6 & I_5 & I_3 \end{bmatrix}_{(14\times 21)}$$

  where I_x is a 7×7 identity matrix with rows cyclically shifted to the left by x positions. The resulting Tanner graph has girth 12 and is shown in Fig. 2. (Fig. 2 shows both the "ring-like" structure characteristic of these constructions and a "flattened" representation that will be useful in what follows.) The associated code has minimum distance $d_{\min} = 6$ and is a regular $(2, 3)$ LDPC code. The best $[21, 8]$ linear code has $d_{\min} = 8$.
- Example 3: A $[104, 30]$ QC code with nonprime $m(m = 26)$.
  Elements $a = 5, b = 9$ are chosen from $\mathbb{Z}^*_{26}$; then $\phi(m) = 12$, $o(a) = 4$, $o(b) = 3$, and the parity-check matrix is given by

$$H = \begin{bmatrix} I_1 & I_5 & I_{25} & I_{21} \\ I_9 & I_{19} & I_{17} & I_7 \\ I_3 & I_{15} & I_{23} & I_{11} \end{bmatrix}_{(78\times 104)}$$

  where I_x is a 26×26 identity matrix with rows shifted cyclically to the left by x positions. The code is a regular $(3, 4)$ LDPC code with minimum distance $d_{\min} = 14$ (determined using MAGMA).
- Example 4: A $[5219, 4300]$ QC code $(m = 307)$.
  Elements $a = 9, b = 17$ are chosen from GF(307) (note that 307 is a prime); then $o(a) = 17$, $o(b) = 3$, and the parity-check matrix is given by the first matrix at the bottom of the page, where I_x is a 307×307 identity matrix with rows cyclically shifted to the left by x positions. H is a 921×5219 matrix and describes a regular $(3, 17)$ LDPC code with minimum distance upper-bounded by 24 (see Section III-E).

These examples show that the construction technique described above yields codes with a wide range of rates and block lengths.

One possible modification to the above construction is now proposed. We prune certain edges of the constraint graph obtained from the above construction in a selective manner. We begin by constructing a regular (j, k) parity-check matrix H as described above. For $0 \le i \le j - 3$, we then replace the last $(j-1-i)$ circulant submatrices in the ith row of circulant submatrices with all-zero matrices.[5] The modified parity-check matrix $\widetilde{H}$ is as shown in (3) at the bottom of the page, where 0 is the $m \times m$ all-zero matrix and the LDPC code is now irregular. The irregular codes are still QC, and hence their parity-check matrices can be described efficiently and they can be used to generate LDPC convolutional codes (see Section II-B). We show (in Section III-F) that these codes can be encoded efficiently with complexity linear in the block length of the code. A similar construction of LDPC codes that can be encoded efficiently

[5]The circulant rows are numbered beginning $0, 1, 2, \ldots$.

$$H = \begin{bmatrix} I_1 & I_9 & I_{81} & I_{115} & I_{114} & I_{105} & I_{24} & I_{216} & I_{102} & I_{304} & I_{280} & I_{64} & I_{269} & I_{272} & I_{299} & I_{235} & I_{273} \\ I_{17} & I_{153} & I_{149} & I_{113} & I_{96} & I_{250} & I_{101} & I_{295} & I_{199} & I_{256} & I_{155} & I_{167} & I_{275} & I_{19} & I_{171} & I_4 & I_{36} \\ I_{289} & I_{145} & I_{77} & I_{79} & I_{97} & I_{259} & I_{182} & I_{103} & I_6 & I_{54} & I_{179} & I_{76} & I_{70} & I_{16} & I_{144} & I_{68} & I_{305} \end{bmatrix}$$

$$\widetilde{H} = \begin{bmatrix} I_1 & \cdots & I_{a^{k-j-1}} & I_{a^{k-j}} & 0 & \cdots & 0 & 0 & 0 \\ I_b & \cdots & I_{a^{k-j-1}b} & I_{a^{k-j}b} & I_{a^{k-j+1}b} & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ I_{b^{j-3}} & \cdots & I_{a^{k-j-1}b^{j-3}} & I_{a^{k-j}b^{j-3}} & I_{a^{k-j+1}b^{j-3}} & \cdots & I_{a^{k-3}b^{j-3}} & 0 & 0 \\ I_{b^{j-2}} & \cdots & I_{a^{k-j-1}b^{j-2}} & I_{a^{k-j}b^{j-2}} & I_{a^{k-j+1}b^{j-2}} & \cdots & I_{a^{k-3}b^{j-2}} & I_{a^{k-2}b^{j-2}} & I_{a^{k-1}b^{j-2}} \\ I_{b^{j-1}} & \cdots & I_{a^{k-j-1}b^{j-1}} & I_{a^{k-j}b^{j-1}} & I_{a^{k-j+1}b^{j-1}} & \cdots & I_{a^{k-3}b^{j-1}} & I_{a^{k-2}b^{j-1}} & I_{a^{k-1}b^{j-1}} \end{bmatrix} \quad (3)$$

TABLE I
EXAMPLES OF CODES CONSTRUCTED FROM (PRIME) CIRCULANT SIZES

Block length	Design Parameters		Design Rate	Actual Rate	Circulant Size
N	j	k	r_d	r	m
21	2	3	1/3 = 0.3333	0.3809	7
93	2	3	1/3 = 0.3333	0.3441	31
129	2	3	1/3 = 0.3333	0.3411	43
155	3	5	2/5 = 0.4000	0.4129	31
186	5	6	1/6 = 0.1667	0.1882	31
305	3	5	2/5 = 0.4000	0.4065	61
755	3	5	2/5 = 0.4000	0.4423	151
905	3	5	2/5 = 0.4000	0.4022	181
1055	3	5	2/5 = 0.4000	0.4018	211
1205	3	5	2/5 = 0.4000	0.4016	241
1477	3	7	4/7 = 0.5714	0.5727	211
1477	5	7	2/7 = 0.2857	0.2884	211
1703	5	13	8/13 = 0.6153	0.6177	131
1928	3	8	5/8=0.6250	0.6260	241
1928	5	8	3/8=0.3750	0.3771	241
1967	5	7	2/7 = 0.2857	0.2877	281
2041	3	13	10/13 = 0.7692	0.7702	157
2248	5	8	3/8=0.3750	0.3768	281
2947	3	7	4/7 = 0.5714	0.5721	421
2947	4	7	3/7 = 0.4286	0.4296	421
3641	3	11	8/11 = 0.7272	0.7278	331
3641	5	11	6/11 = 0.5454	0.5465	331
5219	3	17	14/17 = 0.8235	0.8239	307
11555	3	5	2/5 = 0.4000	0.4001	2311

TABLE II
EXAMPLES OF CODES CONSTRUCTED FROM (NONPRIME) CIRCULANT SIZES

Block length	Design Parameters		Design Rate	Actual Rate	Circulant Size
N	j	k	r_d	r	m
104	3	4	1/4 = 0.2500	0.2885	26
610	3	5	2/5 = 0.4000	0.4066	122
994	5	7	2/7 = 0.2857	0.2938	142
1032	3	6	3/6 = 0.5000	0.5078	172
1240	3	5	2/5 = 0.4000	0.4129	248
5430	3	5	2/5 = 0.4000	0.4022	1086
5894	5	7	2/7 = 0.2857	0.2871	842

has been proposed in [15]. We also note that the thresholds of the irregular codes, described here, calculated using the technique described in [16], are superior to those of the regular codes of the original construction.

The rank of the parity-check matrix $\widetilde{H}$ is $(j-2)m$ plus the number of linearly independent rows in the last $2m$ rows of $\widetilde{H}$. (The first $(j-2)m$ rows are linearly independent, and none of the last $2m$ rows can be expressed as a linear combination of the first $(j-2)m$ rows.)

- Example 5: A $[155, 63]$ irregular QC code.

Consider the $[155, 64, 20]$ QC code of Example 1. The parity-check matrix of this $(3, 5)$ regular LDPC code is obtained from the original construction. Hence, we can obtain an irregular LDPC code with parity-check matrix given by

$$\widetilde{H} = \begin{bmatrix} I_1 & I_2 & I_4 & 0 & 0 \\ I_5 & I_{10} & I_{20} & I_9 & I_{18} \\ I_{25} & I_{19} & I_7 & I_{14} & I_{28} \end{bmatrix}_{(93\times 155)}$$

where I_x is a 31×31 identity matrix with rows shifted cyclically to the left by x positions and 0 is the 31×31

all-zero matrix. $\widetilde{H}$ is a 93×155 matrix and describes a rate $R = 63/155 \approx 0.4065$ irregular LDPC code with minimum distance $d_{\min} = 16$ (MAGMA).

B. Construction of LDPC Convolutional Codes

An LDPC convolutional code can be constructed by replicating the constraint structure of the QC LDPC block code to infinity [8]. Naturally, the parity-check matrices (or, equivalently, the associated constraint graphs) of the convolutional and QC codes form the key link in this construction.

Each circulant in the parity-check matrix of a QC block code can be specified by a unique polynomial; the polynomial represents the entries in the first column of the circulant matrix. For example, a circulant matrix whose first column is $[1\,1\,1\,0\,1\,0]^T$ is represented by the polynomial $1 + D + D^2 + D^4$. Thus, the $jm \times km$ binary parity-check matrix of a regular LDPC code obtained from the construction described above can be expressed in polynomial form (with indeterminate D) to obtain the following $j \times k$ matrix

$$H(D) = \begin{bmatrix} D & D^a & D^{a^2} & \cdots & D^{a^{k-1}} \\ D^b & D^{ab} & D^{a^2 b} & \cdots & D^{a^{k-1}b} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ D^{b^{j-1}} & D^{ab^{j-1}} & D^{a^2 b^{j-1}} & \cdots & D^{a^{k-1}b^{j-1}} \end{bmatrix}_{(j \times k)}.$$

Note that, since the circulant submatrices in the LDPC code construction are all shifted identity matrices, the polynomials in $H(D)$ are all monomials; the power of D indicates how many places the identity matrix was shifted to form the corresponding circulant submatrix. $H(D)$ is the parity-check matrix of a corresponding LDPC convolutional code in polynomial form. (The indeterminate D is now interpreted as the delay operator in the convolutional code.) We note here that the parity-check matrix of the QC code, when written in circulant form, is over the ring $\mathbb{F}_2[D]/\langle D^m + 1\rangle$, i.e., the polynomial ring $\mathbb{F}_2[D]$ modulo the ideal $\langle D^m + 1\rangle$, whereas the parity-check matrix of the convolutional code is over the rational field $\mathbb{F}_2(D)$. In all cases that were examined, the rate of the LDPC convolutional codes obtained from the QC codes was equal to the design rate of the original QC code, i.e., $R = 1 - (j/k)$. This rate is slightly less than the rate of the original QC code.

Since the irregular LDPC codes obtained from the modified construction are also QC, the above procedure can also be applied on them to obtain corresponding irregular LDPC convolutional codes.

- Example 6: A rate 2/5 LDPC convolutional code.
  From the $[155, 64, 20]$ QC code in Example 1, we can obtain a rate–$2/5$ convolutional code with parity-check and generator matrices given by

$$H(D) = \begin{bmatrix} D & D^2 & D^4 & D^8 & D^{16} \\ D^5 & D^{10} & D^{20} & D^9 & D^{18} \\ D^{25} & D^{19} & D^7 & D^{14} & D^{28} \end{bmatrix}_{(3\times 5)}$$

$$G(D) = \begin{bmatrix} \frac{a_1(D)}{\Delta(D)} & \frac{a_2(D)}{\Delta(D)} & \frac{a_3(D)}{\Delta(D)} & 1 & 0 \\ \frac{b_1(D)}{\Delta(D)} & \frac{b_2(D)}{\Delta(D)} & \frac{b_3(D)}{\Delta(D)} & 0 & 1 \end{bmatrix}_{(2\times 5)}$$

where

$$\begin{aligned} a_1(D) &= D^4(1 + D^7 + D^{10} + D^{14} + D^{18} + D^{29}) \\ a_2(D) &= D^3(1 + D^3 + D^6 + D^{18} + D^{21} + D^{36}) \\ a_3(D) &= D^7(1 + D^4 + D^8 + D^{11} + D^{15} + D^{22}) \\ b_1(D) &= D^{13}(1 + D^6 + D^{14} + D^{15} + D^{23} + D^{28}) \\ b_2(D) &= D^{12}(1 + D^2 + D^{11} + D^{21} + D^{23} + D^{35}) \\ b_3(D) &= D^{21}(1 + D^3 + D^4 + D^5 + D^{10} + D^{16}) \\ \Delta(D) &= 1 + D^4 + D^{14} + D^{25} + D^{26} + D^{33}. \end{aligned}$$

The generator matrix $G(D)$ was obtained from the parity-check matrix $H(D)$ using Gaussian elimination. We conjecture that the above convolutional code has a free distance d_{free} of 24. By choosing one of the information sequences equal to the denominator polynomial, i.e., $1 + D^4 + D^{14} + D^{25} + D^{26} + D^{33}$, and the other information sequence as the all-zero sequence, we obtain a code sequence of weight 24. This only provides an upper bound on the d_{free} of this convolutional code, but we have been unable to find any lower weight code sequences. Interestingly, as we shall see, this is the same as the upper bound obtained by MacKay and Davey [17] for LDPC matrices constructed by using nonoverlapping permutation matrices that commute with each other, as is the case here. Further, the results of [8] guarantee that the minimum distance of the QC block code, 20 in this case, provides a lower bound on the free distance of the associated convolutional code.

Note, however, that $G(D)$ above is not in minimal form. The minimal-basic generator matrix [18] has the minimum overall constraint length among all equivalent rational and polynomial generator matrices and is thus of interest. The minimal-basic form of $G(D)$, with overall constraint length $\nu = 23 + 22 = 45$, is given by

$$G_{\min}(D) = \begin{bmatrix} a'_1(D) & a'_2(D) & a'_3(D) & a'_4(D) & a'_5(D) \\ b'_1(D) & b'_2(D) & b'_3(D) & b'_4(D) & b'_5(D) \end{bmatrix}_{(2\times 5)}$$

where

$$\begin{aligned} a'_1(D) &= D^6 + D^{10} + D^{14} + D^{17} + D^{18} \\ a'_2(D) &= D^5 + D^8 + D^9 + D^{11} + D^{13} + D^{14} \\ &\quad + D^{15} + D^{19} + D^{21} + D^{22} + D^{23} \\ a'_3(D) &= D^9 \\ a'_4(D) &= D^2 + D^{10} + D^{13} + D^{15} + D^{16} + D^{18} \\ &\quad + D^{19} + D^{20} + D^{21} + D^{23} \\ a'_5(D) &= 1 + D + D^3 + D^9 + D^{10} + D^{11} + D^{12} \\ &\quad + D^{13} + D^{15} \\ b'_1(D) &= D^4 + D^5 + D^6 + D^8 + D^{10} + D^{11} \\ &\quad + D^{12} + D^{13} + D^{14} + D^{16} + D^{17} \\ b'_2(D) &= D^3 + D^4 + D^5 + D^6 + D^8 + D^{11} \\ &\quad + D^{12} + D^{13} + D^{14} + D^{16} \\ &\quad + D^{17} + D^{18} + D^{19} + D^{20} \\ b'_3(D) &= D^7 + D^8 + D^9 + D^{12} \end{aligned}$$

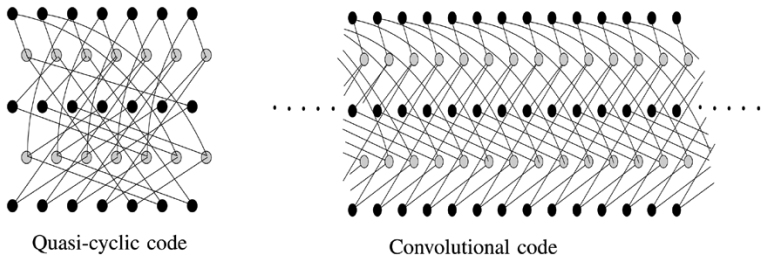

Fig. 3. Tanner graph for the rate–$1/3$ convolutional code of Example 7.

$$b_4'(D) = 1 + D + D^2 + D^5 + D^8 + D^9 + D^{10} + D^{11} + D^{13} + D^{14} + D^{15} + D^{16} + D^{21} + D^{22}$$
$$b_5'(D) = 1 + D^2 + D^4 + D^7 + D^8 + D^{13} + D^{14}$$

Several different LDPC convolutional codes can be obtained from the same QC LDPC block code. For example, reordering the rows of the parity-check matrix of the QC code within each block of circulant submatrices will leave it unchanged but can lead to a completely different LDPC convolutional code.[6]

Consider once again the $[155, 64, 20]$ QC code. Let us cyclically shift the first block of $m = 31$ rows above by one position, the middle block of 31 rows by five positions, and the last block of 31 rows by 25 positions, so that now the first row in each block has a 1 in the first column. The resulting LDPC matrix is given by

$$H = \begin{bmatrix} I_0 & I_1 & I_3 & I_7 & I_{15} \\ I_0 & I_5 & I_{15} & I_4 & I_{13} \\ I_0 & I_{25} & I_{13} & I_{20} & I_3 \end{bmatrix}_{(93\times 155)}$$

where again I_x is a 31×31 identity matrix with rows cyclically shifted to the left by x positions. Clearly, the QC block code and its associated constraint graph are unaffected by these row shifts. However, the convolutional code obtained by following the above procedure has the parity-check matrix

$$H_1(D) = \begin{bmatrix} 1 & D & D^3 & D^7 & D^{15} \\ 1 & D^5 & D^{15} & D^4 & D^{13} \\ 1 & D^{25} & D^{13} & D^{20} & D^3 \end{bmatrix}_{(3\times 5)}.$$

Looking at the third constraint equation of $H(D)$ and $H_1(D)$, we see that that the two codes are in fact different. (Note that the first and second constraint equations of $H(D)$ and $H_1(D)$ are equivalent, since the first and second rows of $H(D)$ are just delayed versions of the first and second row of $H_1(D)$.)

- Example 7: A rate–$1/3$ LDPC convolutional code.
  From the QC code of Example 2, a rate–$1/3$ convolutional code with the following parity-check and generator matrices is obtained:

$$H(D) = \begin{bmatrix} D & D^2 & D^4 \\ D^6 & D^5 & D^3 \end{bmatrix}_{(2\times 3)}$$
$$G(D) = \begin{bmatrix} \frac{D+D^3}{1+D^2+D^4} & 1 & \frac{D^2}{1+D^2+D^4} \end{bmatrix}_{(1\times 3)}.$$

  In this case, the minimal basic generator matrix has overall constraint length $\nu = 4$. The convolutional code has a free distance $d_{\text{free}} = 6$, equal to the minimum distance of the original QC code.

In this manner, it is possible to construct numerous convolutional codes with a sparse constraint graph representation. For example, for every entry in Table I, it is possible to construct a convolutional code with an actual rate equal to the design rate indicated in the table.

III. Properties of Constructed Codes

This section describes the properties of the codes constructed in Section II. Specifically, the structure of the constraint graphs, the minimum distance of the codes, and encoding techniques are described.

A. Relation Between the Block and Convolutional Constraint Graphs

Similar to the block codes, a constraint graph based on a parity-check matrix can be obtained for the convolutional codes. However, in the case of the convolutional codes, the constraint graph is infinite. The constraint graphs for the rate-$1/3$ convolutional code in Example 7 and the corresponding QC code (Example 2) are shown in Fig. 3. We observe here that the constraint graph of the convolutional code is strikingly similar to that of the QC code. This reflects the similarity in the QC and convolutional parity-check constraints. (The constraint graph of the convolutional code can be viewed as being unwrapped from that of the QC code.)

B. QC Block Codes Viewed as Tail-Biting Convolutional Codes

Tail biting is a technique by which a convolutional code can be used to construct a block code without any loss of rate [19], [20]. An encoder for the tail-biting convolutional code is obtained by reducing each polynomial entry in the generator matrix of the convolutional code modulo $D^m + 1$ for some positive integer m and replacing each of the entries so obtained with circulant matrices of size $m \times m$. The block length of the block code[7] so derived depends on m. Since the generator matrix consists of circulant submatrices, the tail-biting convolutional code obtained is a QC block code. However, this tail-biting code has a rate equal to that of the convolutional code—which is less than or equal to the rate of the original QC code. As expected, the two

[6]Here we are only interested in those convolutional codes that have the same graph connectivity (i.e., nodes of the same degrees) as the base QC block code. For example, equivalent representations for the QC block code can be obtained by suitable linear combinations of rows of H. However, in such a case, the new representation(s) of the block code and that of the corresponding convolutional code will in general not have the same node degrees as the original representation of the QC code.

[7]With a feedforward encoder for the convolutional code, a tail-biting code of any block length may be obtained [21].

QC codes, i.e., the original QC code and the tail-biting convolutional code are closely related, as the following theorem shows.

Theorem 3.1: Let C be a length km QC code with the $jm \times km$ parity-check matrix H, where H is composed of $m \times m$ circulants (i.e., its period is k). Let $\mathcal{C}$ be a convolutional code obtained by unwrapping H. Then the QC block code (tail-biting convolutional code) $\widetilde{C}$ of length km constructed from $\mathcal{C}$ is a subcode of C.

Proof: Since the tail-biting code $\widetilde{C}$ is QC with period k, any codeword in $\widetilde{C}$ can be described by a set of polynomials in the ring $\mathbb{F}_2[D]/\langle D^m+1\rangle$. Therefore, any codeword in $\widetilde{C}$ is of the form $a(D)G(D) \bmod \langle D^m+1\rangle$, where $G(D)$ is the generator matrix of the convolutional code $\mathcal{C}$ and $a(D)$ is an information polynomial. Now, we know that the polynomial generator matrix of the convolutional code $\mathcal{C}$ satisfies the parity constraints imposed by its parity-check matrix $H(D)$, i.e., $G(D)H^T(D) = 0$. Therefore, $G(D)H^T(D) \equiv 0 \bmod \langle D^m+1\rangle$. Since, by construction, $H(D)$ is also the parity-check matrix of the original QC block code C, with the entries in $H(D)$ now interpreted as being in the ring $\mathbb{F}_2[D]/\langle D^m+1\rangle$, any codeword in $\widetilde{C}$ satisfies the constraints of the original QC code C, i.e., the tail-biting code $\widetilde{C}$ is a subcode of C. □

If there were no rank reduction in the parity-check matrix of the original QC block code, it would have a rate equal to that of the convolutional code. In such a case, it is easy to see that the tail-biting convolutional code would be exactly the same as the original QC block code.

We have derived a rate-$2/5$ convolutional code (Example 5) from the $[155, 64, 20]$ QC block code (Example 1); from the rate-$2/5$ convolutional code we can, in turn, derive a tail-biting convolutional code, i.e., another QC block code of block length 155 and 62 information bits (rate $2/5$), that is a subcode of the original $[155, 64, 20]$ QC block code and has $d_{\min} \geq 20$.

C. Graph Automorphisms

The structure of the multiplicative groups used in constructing the regular LDPC codes leads to certain graph automorphisms in the constraint graphs of the constructed codes. These symmetries are the topic of this subsection. A node x belonging to the vertex set V of the graph $\mathcal{G}$ is denoted by $x \in V$. If $\mathcal{G}$ contains an edge between x and y, then the pair (x, y) is said to belong to the edge set E of $\mathcal{G}$, i.e., $(x, y) \in E$. A graph automorphism is a bijective map f from V to V that preserves edges, i.e., $(x, y) \in E$ iff $(f(x), f(y)) \in E$.

The bit nodes of the constraint graph correspond to the columns of the parity-check matrix H (in (2)) and the constraint nodes correspond to the rows. The bit and constraint nodes can be divided into blocks of m nodes, called bit blocks and constraint blocks, respectively, where the first m columns of H correspond to the first bit block of m nodes, the next m columns correspond to the second bit block of m nodes, and so on. Similarly, dividing the rows of H, the first m rows correspond to the first constraint block of m nodes, the next m rows to the second constraint block, and so on. To illustrate the graph automorphisms, the bit nodes are labeled as $[x_0, x_1, \ldots, x_{j-1}]$; this notation denotes that a bit node participates in x_ith parity-check equation in the ith constraint block, $0 \leq x_i \leq m-1$, $0 \leq i \leq j-1$. For instance, for the parity-check matrix in Example 1, the first bit node is denoted by $[1, 5, 25]$. A constraint node is denoted by a k tuple $[\cdot, \ldots, \cdot, x_p, \cdot, \ldots, \cdot]$, where only the ith position is defined. This represents the x_pth parity-check equation in the ith constraint block, $0 \leq x_p \leq m-1$, $0 \leq i \leq k-1$. For the parity-check matrix in Example 1, the tth parity-check equation in the third constraint block is denoted by[8] $[\cdot, \cdot, t]$, for $0 \leq t \leq 30$.

The constraint graph for the parity-check matrix in (2) has the following graph automorphisms:

1) Bit nodes:

$$\sigma([x_0, x_1, \ldots, x_{j-1}]) = [x_0+1, x_1+1, \ldots, x_{j-1}+1] \bmod m$$

Constraint nodes:

$$\sigma([\cdot, \ldots, \cdot, x_p, \cdot, \ldots, \cdot]) = [\cdot, \ldots, \cdot, x_p+1, \cdot, \ldots, \cdot] \bmod m$$

2) Bit nodes:

$$\pi([x_0, x_1, \ldots, x_{j-1}]) = [ax_0, ax_1, \ldots, ax_{j-1}] \bmod m.$$

Constraint nodes:

$$\pi([\cdot, \ldots, \cdot, x_p, \cdot, \ldots, \cdot]) = [\cdot, \ldots, \cdot, ax_p, \cdot, \ldots, \cdot] \bmod m$$

3) Bit nodes:

$$\rho([x_0, x_1, \ldots, x_{j-1}]) = [bx_{j-1}, bx_0, \ldots, bx_{j-2}] \bmod m$$

Constraint nodes:

$$\rho([\cdot, \ldots, \cdot, x_p, \cdot, \ldots, \cdot]) = [\cdot, \ldots, \cdot, bx_p, \cdot, \ldots, \cdot] \bmod m$$

($\uparrow$ ith posn. $\quad$ $\uparrow$ $(i+1)$th posn.)

where a and b are the elements of $\mathrm{GF}(m)$ (orders k and j, respectively) used to construct the parity-check matrix as described earlier.

The automorphism σ maps a bit node to another bit node in the same bit block as the original node, whereas π maps a bit node in one bit block to a bit node in another bit block. Likewise, ρ maps a constraint node in one constraint block to a constraint node in another constraint block. All the bit nodes can be generated as an orbit of a single bit node x under the group of automorphisms generated by σ and π, i.e.,

$$V|_{\text{bit nodes}} = \{\theta(x) \mid \theta \in \langle \sigma, \pi \rangle\}.$$

In Example 1, the bit nodes are the orbit of $[1, 5, 25]$. Similarly, all the constraint nodes can be generated as an orbit of a single constraint node y under the group of automorphisms generated by σ and ρ, i.e.,

$$V|_{\text{constraint nodes}} = \{\theta(y) \mid \theta \in \langle \sigma, \rho \rangle\}.$$

In Example 1, the constraint nodes are the orbit of $[1, \cdot, \cdot]$. We also note that, using σ, π, and ρ, any edge of the constraint graph can be mapped to any other edge.

[8] In the notation $[\cdot, \cdot, t]$, "$\cdot$" denotes that the quantity is undefined, i.e., to denote the tth check equation in the third constraint block, since H contains three blocks of constraints, the first two components are undefined.

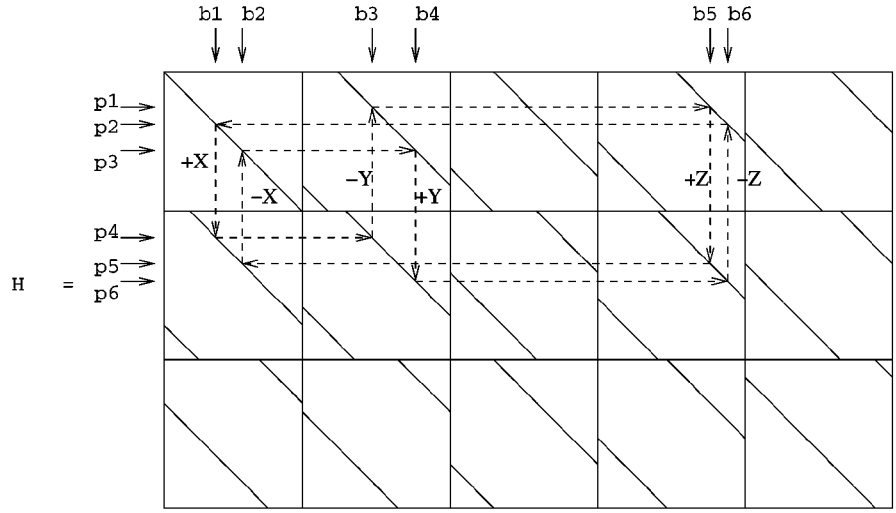

Fig. 4. A 12-cycle.

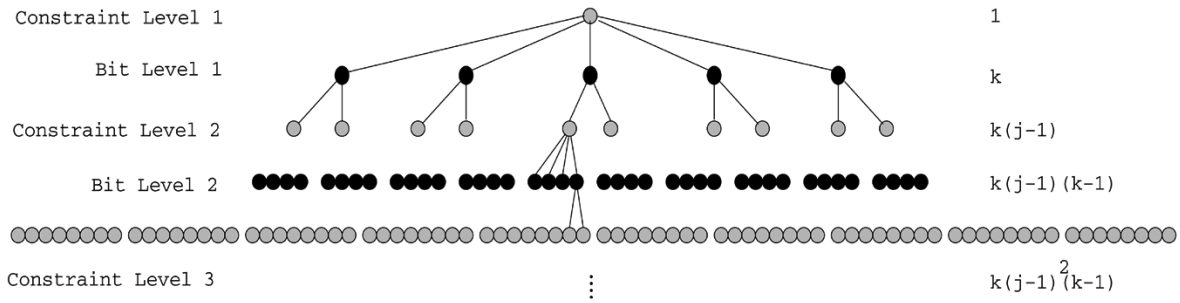

Fig. 5. A tree beginning at a constraint node for a (j, k) regular LDPC graph.

The σ map is just a restatement of the QC nature of the code and is therefore valid for any QC code represented in circulant form. Hence, it is also valid for constraint graphs of the irregular LDPC codes of the modified construction. In addition, an automorphism analogous to σ is valid for the constraint graph of the convolutional code. This automorphism indicates the time invariance property of the convolutional codes.

D. Girth

The BP decoding algorithm converges to the maximum *a posteriori* (MAP) solution for graphs that are trees—i.e., have no cycles; for graphs with cycles, there is no such optimality. The *girth* of a graph is the length of the shortest cycle in the graph. If the girth of the constraint graph is g, then the iterative BP algorithm is known to be *exact* for $\lceil \frac{g-4}{4} \rceil$ iterations. (By *exact*, we mean that the messages exchanged between the nodes of the graph during BP decoding are independent.) For prime circulant sizes, m, the constraint graph representing the parity-check matrix H in (2), cannot have girth less than 6. This is seen by observing that the relative difference between the shifts across any two columns is different along different rows. For nonprime m, however, the girth can be as low as 4. Irrespective of prime or nonprime m, the constraint graph of H in (2) cannot have girth larger than 12. Fig. 4 is a graphical illustration of why this is so (see also [6]). The dark diagonal lines in the figure represent the nonzero entries of the parity-check matrix and the empty regions represent the zeros. The dotted lines indicate a cycle of length 12 in the constraint graph involving bit nodes $b1$ through $b6$ and constraint nodes $p1$ through $p6$. The structure of the circulant submatrices in H gives rise to numerous 12-cycles. Likewise, the upper bound on the girth of the irregular LDPC constraint graphs of $\widetilde{H}$ in (3) is also 12.

An upper bound on the girth of a graph is obtained from the well-known tree bound [10]. For a constraint graph representing a (j, k) regular LDPC code, a tree is generated beginning at a constraint node (see Fig. 5). The first (constraint) level of the tree is a single constraint node. This is connected to k bit nodes at the first bit level of the tree. Each of the k bit nodes are then connected to $j-1$ other constraint nodes at the second constraint level of the tree, and so on.

Enumerating the nodes in this manner, the number of bit nodes counted up to the ith bit level is

$$P_i = k + k(j-1)(k-1) + k(j-1)^2(k-1)^2 + \cdots + k(j-1)^{i-1}(k-1)^{i-1}.$$

If P_i exceeds km, the total number of bit nodes in the graph, then at least one of the bit nodes in the tree enumerated up to bit level i must have occurred more than once, meaning there is a closed path (or a cycle). Hence, if

$$C = \max\{i : P_i \leq km\}$$

an upper bound on the girth of the graph is

$$g_{\text{bit}} = \begin{cases} 4C+2, & \text{if } P_C < km \\ 4C, & \text{if } P_C = km. \end{cases}$$

Similarly, counting the number of constraint nodes enumerated up to the ith constraint level, we have

$$S_i = 1 + k(j-1) + k(j-1)^2(k-1) + \cdots + k(j-1)^{i-1}(k-1)^{i-2}.$$

If S_i exceeds jm, the total number of constraint nodes in the graph, then this implies the existence of a cycle in the tree enumerated up to constraint level i. Therefore, if

$$C' = \max\{i : S_i \leq jm\}$$

an upper bound on the girth of the graph is

$$g_{\text{chk}} = \begin{cases} 4C', & \text{if } S_{C'} < jm \\ 4C' - 2, & \text{if } S_{C'} = jm. \end{cases}$$

Therefore, the girth g of the regular (j,k) LDPC codes is upper-bounded by the minimum of g_{bit} and g_{chk}.

For the parity-check matrix H of the $[155, 64, 20]$ code of Example 1, the tree bound is 10, whereas the actual girth of the graph is 8. For the H matrix of the $[21, 8, 6]$ code of Example 2, the tree bound is 12 and so is the actual girth. The tree bound is tight typically for short block lengths. As the circulant size m increases, the girth of the constraint graphs of (2) tends to 12. The inherent algebraic structure in the parity-check matrices of the constructed codes allows one to check for the presence of short cycles very efficiently.

The constraint graphs of the LDPC convolutional codes constructed herein have their girth lower-bounded by the girth of the corresponding QC LDPC constraint graph. For any cycle in the convolutional code constraint graph we can find an equivalent cycle in the QC constraint graph. Say, a particular set of bit and constraint nodes form a cycle in the convolutional code constraint graph. Then the relative shifts between the bit nodes sum to zero. The corresponding sequence of bit and constraint nodes in the QC code (obtained by reducing indices modulo m, where m is the circulant size in the QC code) have exactly the same relative shifts (now read modulo m) between the bit nodes, and hence sum to zero—i.e., in the QC code constraint graph, we find a corresponding cycle of the same length. However, a cycle in the QC constraint graph does not always lead to a cycle in the convolutional constraint graph. While the relative shifts between bit nodes may sum to zero modulo m, the shift may be a nonzero multiple of m, meaning the corresponding bit nodes in the convolutional code do not form a cycle. Hence, it is possible that the convolutional code constraint graph may have a larger girth than the QC code constraint graph. Observe, however, that in Fig. 4 the relative shifts between bit nodes sum to zero, so that the corresponding bit nodes in the convolutional code also form a 12-cycle. Hence, the girth of the convolutional codes is also upper-bounded by 12.

E. Minimum Distance

At high SNRs, the maximum-likelihood decoding performance of an error-correcting code is dominated by the code's minimum distance. MacKay and Davey obtained an upper bound on the minimum distance of certain LDPC codes whose parity-check matrices are composed of nonoverlapping blocks of permutation submatrices that all commute with each other [17]. The regular LDPC codes constructed here satisfy this condition, so the upper bound in [17] on the minimum distance is applicable. The result in [17] says that if a parity-check matrix $H_{M\times N}$ is composed of an $M \times (j+1)M/j$ array of $j \times (j+1)$ nonoverlapping permutation submatrices that all commute with each other, then the minimum distance of the corresponding code is at most $(j+1)!$. For small column weight j, the bound imposes a stringent limitation on the code's minimum distance, especially for codes with large block lengths. Using the same arguments as in [17], it can be shown that for the codes obtained using the modified irregular construction the minimum distance is upper-bounded by $3 \times 2^{(j-1)}$ ($\leq (j+1)!$ for $j > 1$).

The definition of an "asymptotically good" code construction is one that yields a sequence of codes whose minimum distances grow linearly with the block length N as $N \to \infty$. In [10], Gallager considered an ensemble of regular (N, j, k) LDPC codes and showed that, with high probability, a code chosen at random from this ensemble has a minimum distance $d_{\min} \geq N\delta(j,k)$, where $\delta(j,k) > 0$ is a parameter that is dependent on j and k and independent of N. In other words, this result shows the existence of (N, j, k) regular LDPC codes in Gallager's ensemble that have minimum distance at least $N\delta(j,k)$. The regular LDPC codes described here cannot have a minimum distance larger than $(j+1)!$, and hence they are not asymptotically good. However, they may compare well with random (N, j, k) LDPC codes for block lengths up to N', where $N'\delta(j,k) \approx (j+1)!$. For example, in the case of $(3, 5)$ LDPC codes, $\delta(j,k) \approx 0.04$, which means the $(3, 5)$ algebraic LDPC codes of Section II are comparable in minimum distance to $(3, 5)$ random LDPC codes for block lengths up to around $N' = \frac{(j+1)!}{\delta(j,k)} = 600$. (This inference assumes that, as the block length increases, the minimum distance of the algebraically constructed LDPC codes increases until the bound $(j+1)!$ is met—an assumption that has not been shown to be true in general.)

Using simple graph-based analysis, lower bounds on the minimum distance of LDPC codes can be obtained [22]. For instance, when the column weight $j = 2$ and the girth of the constraint graph is g, then any $g/2 - 1$ columns of the parity-check matrix are linearly independent. This implies that $d_{\min} \geq g/2$, and in fact, it is *exactly* $g/2$. For the $[21, 8, 6]$ code of Example 2, the girth of the constraint graph is 12 and the minimum distance is 6. Similar bounds can be obtained for column weight $j > 2$, as in [23].

The results in [8] show that the LDPC convolutional codes obtained by unwrapping the constraint graph of the QC codes have their free distance d_{free} lower-bounded by the minimum distance of the corresponding QC code. Essentially, this is because wrapping back any codeword of the convolutional code produces a valid codeword in the QC code. More succinctly, a codeword in the convolutional code reduced modulo $\langle D^m + 1\rangle$ is a codeword in the QC code, as shown for the tail-biting codes in Theorem 3.1.

F. Encoding

With the use of iterative message-passing decoders with near-optimal performance, decoding has become a simpler task requiring only $O(N)$ complexity [11], whereas traditional encoding techniques require $O(N^2)$ complexity. The general pro-

cedure for encoding any linear block code, specified by a parity-check matrix, is to find a suitable generator matrix for the code. This is done by reducing the parity-check matrix H to systematic form using elementary row and column operations. The computational cost of reducing the H matrix to systematic form is in general $O(N^3)$. Further, the cost of the actual encoding is itself $O(N^2)$.

In [24], Richardson *et al.* show how random LDPC codes can be encoded with *almost* linear cost. They exploit the sparseness of the LDPC matrices to show that the complexity of encoding LDPC codes can be reduced from $O(N^2)$ to $O(N+h^2)$, where $h \ll N$ is a number that depends on the particular LDPC matrix. In the following subsections, we look at alternative ways to encode the LDPC codes introduced in Section II—ways that exploit the inherent algebraic structure of the codes.

1) Encoding Based on the Structure of the Parity-Check Matrix: The parity-check matrix of the QC regular LDPC codes written in polynomial form is

$$H(D) = \begin{bmatrix} D & D^a & D^{a^2} & \cdots & D^{a^{k-1}} \\ D^b & D^{ab} & D^{a^2 b} & \cdots & D^{a^{k-1}b} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ D^{b^{j-1}} & D^{ab^{j-1}} & D^{a^2 b^{j-1}} & \cdots & D^{a^{k-1}b^{j-1}} \end{bmatrix}_{(j \times k)}.$$

In the above matrix, we observe that an entry along a row is the ath power of the previous entry in that row. (The entries are modulo $\langle D^m + 1 \rangle$.) Similarly, an entry along a column is the bth power of the previous entry in that column. Using Tanner's transform theory [25], it can be shown that this code can be alternately described by a generator matrix composed of a block of circulant submatrices, since the code is QC. Hence, the code can be encoded using shift registers, as is done with cyclic codes. A generator matrix composed of circulants can be obtained as follows: a convolutional code can be derived from the original QC code as described earlier. Then a systematic parity-check matrix for the convolutional code, expressed over the field of rationals $\mathbb{F}_2(D)$, where $\mathbb{F}_2$ is the binary field, can be obtained using elementary row and column operations. (This step is simple because the parity-check matrix is of size $j \times k$.) A systematic (in general, feedback) generator matrix of the convolutional code is easily determined from the corresponding systematic parity-check matrix. Note that the convolutional code can now be encoded as usual using shift registers. By rewriting the generator matrix of the convolutional code in feedforward form and reducing the entries modulo $\langle D^m + 1 \rangle$, a polynomial matrix with entries in the ring $\mathbb{F}_2[D]/\langle D^m + 1 \rangle$ is obtained. As has already been shown (Theorem 3.1), this is the generator matrix of a tail-biting convolutional code that is a subcode of the original QC code.[9] Rewriting the polynomial entries as $m \times m$ circulant matrices, a matrix of size $(k-j)m \times km$ spanning the tail-biting subcode is obtained. This matrix forms a partial generator matrix for the original QC code. The original QC code is spanned by this partial generator matrix and a few additional rows. The additional rows can be found using Tanner's transform theory approach (see [25] for details).

Consider the $[155, 64, 20]$ code of Example 1. A generator matrix of the corresponding convolutional code is the feedforward minimal basic encoder shown in Example 6. The generator matrix of a tail-biting subcode is obtained by reducing the entries in $G_{\min}(D)$ modulo $\langle D^{31} + 1 \rangle$, resulting in the matrix $G'(D)$. A binary matrix G' of size 62×155 is the binary equivalent of $G'(D)$ when written in circulant form, and it has full rank. G' forms a partial generator matrix for the $[155, 64, 20]$ QC code. Since the $[155, 64, 20]$ code has 64 information bits, we are missing two linearly independent rows. The missing (linearly independent) rows are found from the transform theory approach.[10] In this case, referring to Fig. 1, an all-one vector in any two of the five bit rings solves the constraint equations and those solutions complete the space. Note that the linear space spanned by the vectors that are all-one in any two of the five bit rings has dimension 4. However, since two of these dimensions are already spanned by $G_{\min}$, as can be seen by inspection, we need to add only a subset of the vectors that are all-one in any two of the five bit rings. By contrast, these solutions do not exist in the convolutional code.

Alternatively, by reducing the (nonminimal) generator matrix $G(D)$ in Example 6 (written in nonsystematic feedforward form) modulo $\langle D^{31} + 1 \rangle$, a generator matrix of a different QC code is obtained. Rewriting this in circulant form, a 62×155 matrix G'' is obtained. However, we find that G'' is not of full rank; there are two linearly dependent rows (and hence, it describes a $[155, 60]$ QC code).

The transform theory approach can also be used to find the complete set of generators without having to obtain a partial set of generators from the tail-biting code; however, we prefer the above procedure since it is computationally simpler.

For the regular QC LDPC codes it has been observed that, in most cases, the rank of H is $(j-1)$ less than full rank. Thus, $\operatorname{rank}(H) = mj - (j-1)$. However, it is possible that the rank of H is less than $mj - (j-1)$. An example, of a $(3, 5)$ code where the drop in the rank of H is $30 + j - 1 = 30 + 3 - 1 = 32$ for $m = 151$ is given in the Appendix. In either case, a partial generator matrix for the QC code can be obtained from the rows of the tail-biting generator matrix and the remaining (linearly independent) rows can be obtained from the transform theory approach. An efficient encoder based on shift registers can then be implemented using this generator matrix.

2) Encoding by Erasure Decoding: Luby *et al.*, in their pioneering work on the design of codes for the binary erasure channel [26] also showed how the graph-based message-passing decoder can be exploited to encode their codes. In this subsection, we investigate their approach for encoding the algebraic LDPC codes introduced in Section II.

To perform graph-based encoding, the bit nodes corresponding to information bits are assumed to be known and the bit nodes corresponding to parity bits are assumed to be unknown, or, *erased*. The parity bits can potentially be recovered by simulating the graph-based decoder for an erasure

[9]The generator matrix of the convolutional code can either be in minimal or nonminimal form. In either case, a (possibly different) QC code, which is a subcode of the original QC code (Theorem 3.1), is obtained by reducing the entries of the convolutional generator matrix modulo $\langle D^m + 1 \rangle$.

[10]In fact, in this case, the missing rows are obtained from just the 0th eigenvalue matrix of the transform [25].

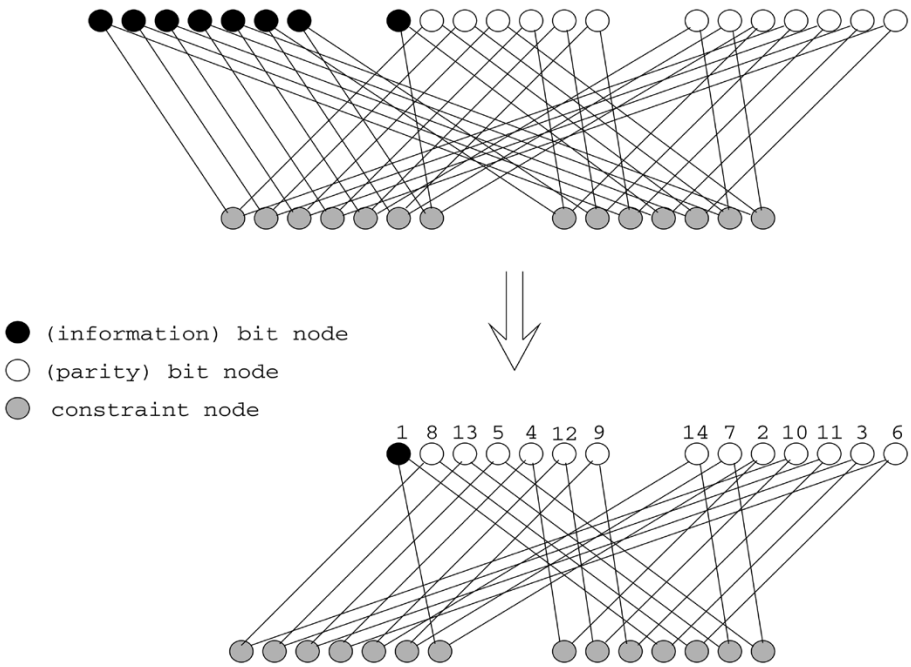

Fig. 6. Encoding the $[21, 8, 6]$ QC LDPC code by erasure decoding.

channel. Recently, Di *et al.* in [27] introduced the notion of a *stopping set* in a constraint graph. The graph-based decoder cannot successfully recover all erasures if the erased bit nodes contain a stopping set of the constraint graph. In the context of encoding, if none of the stopping sets of the LDPC constraint graph contain only parity-bit nodes, then the result of erasure decoding on such a constraint graph is a successful recovery of all the parity bits, implying a successful encoding!

The constraint graphs of the algebraically designed LDPC codes introduced in Section II have structure that can be exploited in encoding. The code of Example 2 provides an illustration of how the constraint graph itself can be used to perform encoding without the need to find the generator matrix of the code. The constraint graph of the $[21, 8, 6]$ QC code in Fig. 2 can be redrawn as in Fig. 6. Of the 21 bit nodes, eight correspond to information bits and 13 correspond to parity bits. Suppose all the bits in the first circulant block (the first seven nodes) and the first bit in the second circulant block are chosen as information bit nodes (as shown in the figure). Then, to encode, the values of the remaining bit nodes must be determined. The values of the information bit nodes are obtained from the bits to be encoded and the parity-bit nodes are assumed to be erased. The numbers on the nodes indicate the order in which the parity bits are recovered by the erasure decoding procedure. The node labeled "1" is an information bit node; the decoder will first recover nodes labeled "2" and "3," and then the nodes labeled "4" and "5," and so on.

The erasure decoding procedure can be described as follows. Each constraint node is assigned a value that equals the value of the known adjoining bit node in the first circulant block; these known bit nodes in the first circulant block are then deleted from the graph. Examining the reduced constraint graph shown at the bottom of Fig. 6, it can be seen that there is now a single path of length 28 that traverses through all 14 nodes once, i.e., the reduced graph is a single cycle of length 28. Since every constraint node now has degree 2, assigning a value to the remaining information bit node determines the values of all the parity-bit nodes uniquely.

A natural question that arises is whether all the LDPC codes having the structure in (2) can be encoded in the above manner, and if so, how do we choose the nodes that represent information bits? The answer to this is not known in general; however, for a specific case, the question may be answered in the affirmative by the following lemma.

Lemma 3.1: For all regular $(2, k)$ LDPC codes constructed as described, if $(a-1)(b-1) \neq 0 \bmod m$, then the code can be encoded via erasure decoding with linear complexity.

Proof: If the code is designed from the multiplicative group of the integer m, then the block length of the code is km. Since the sum of all $2m$ rows of H is the zero vector, there is at least one dependent row in the H matrix. Moreover, because $(a-1)(b-1) \neq 0 \bmod m$ it can be shown that the $2m \times 2m$ submatrix comprised of the last $2m$ columns of H has a rank of exactly $2m-1$. Hence, the rank of H is reduced by one and consequently the dimension of the code is $(k-2)m+1$. We can let the last $(2m-1)$ columns of H correspond to parity-bit nodes and hence, the first $(k-2)m+1$ columns of H (2) represent information-bit nodes. By specifying the values of the first $(k-2)m$ information bits, the values of the adjoining constraint nodes can be updated and the $(k-2)m$ information-bit nodes can be deleted. The reduced graph now contains $2m$ bit nodes and $2m$ constraint nodes, where each node is of degree 2. The claim is that all of these nodes are connected by a single cycle of length $4m$. Given the claim, by specifying the value of one of the remaining bit nodes (i.e., the last information bit node), the $2m-1$ parity-bit nodes are determined uniquely, thereby completing the encoding. The claim is easily seen as follows: The matrix representing the

reduced constraint graph is a submatrix of the original H matrix with all the rows present and some of the columns removed. Therefore, the single dependent row in this submatrix is the sum of the remaining rows. Since the column and row weights are two in the submatrix, the reduced graph itself is a single cycle of length $4m$ connecting the $4m$ nodes. Finally, we note that the complexity of encoding, which is the complexity of erasure decoding on the constraint graph, is linear in the block length. □

For prime m, the condition $(a-1)(b-1) \neq 0 \bmod m$ is always true; in particular, all of the $(2,k)$ LDPC codes in Table I satisfy Lemma 3.1. However, this is not necessarily true for nonprime m.

We now show that the irregular LDPC codes obtained from the modified construction technique proposed in Section II can also be encoded, using their constraint graphs, by the above procedure.

Theorem 3.2: For an irregular LDPC code, from the modified construction, if $(a-1)(b-1) \neq 0 \bmod m$, then the code can be encoded via erasure decoding with linear complexity.

Proof: Assume the code is originally designed as a (j,k) regular LDPC code with block length km. Then the irregular code has a block length of km and a parity-check matrix $\widetilde{H}$ of size $jm \times km$. The last $2m$ rows of $\widetilde{H}$ contain at least one dependent row. Since $(a-1)(b-1) \neq 0 \bmod m$ it follows, as in the proof of Lemma 3.1, that the rank of the parity-check matrix is reduced by exactly one and consequently the dimension of the code is $(k-j)m+1$. Let the first $(k-j)m$ columns of $\widetilde{H}$ represent $(k-j)m$ information bits. The structure of the parity-check matrix (3) is such that by specifying the values of these $(k-j)m$ information bits, the next m bits corresponding to the next m columns of $\widetilde{H}$ are determined uniquely from the first row of circulant submatrices; knowing these m bits, the next m bits are then determined from the second row of circulant submatrices. This procedure is repeated through the $(j-3)$th row of circulant submatrices, and in the process $(j-2)m$ parity bits are recovered. The first $(j-2)m$ constraint nodes are now deleted from the graph, the remaining $2m$ constraint nodes are updated with the values of the known adjoining bit nodes, and the known bit nodes are then deleted. The reduced constraint graph contains $2m$ bit nodes and $2m$ constraint nodes and the graph is a single cycle of length $4m$ connecting all the nodes. By the previous lemma, specifying the one remaining information bit node determines the remaining $2m-1$ parity-bit nodes uniquely, thus completing the encoding. Here again, the complexity of encoding is linear in the block length. □

The encoding procedure described above for the block codes can also be extended to the convolutional codes. If the graph-based encoding does not directly apply to a (j,k) regular LDPC code constructed as described (i.e., one for which either $j \neq 2$ or $j = 2$, but the rank drop in the parity-check matrix is larger than one, thus violating the condition for Lemma 3.1), then by a procedure akin to the one described in [24], the parity-check matrix can be reduced to a form where graph-based encoding *almost* works. This form of the parity-check matrix will contain an $h \times h$ matrix (where h is much smaller than the block length N of the associated QC code) that needs to be inverted to complete encoding.

IV. Performance Results

In this section, the performance obtained with the QC and corresponding convolutional LDPC codes introduced in Section II is presented for a binary phase-shift keying (BPSK) modulated AWGN channel. In both cases, the iterative BP algorithm was used for decoding.

A. LDPC QC Block Codes

The performance of a few regular LDPC codes, constructed as described in Section II, with BP decoding is shown in Fig. 7; the results are compared with regular randomly constructed LDPC codes[11] of similar rates and block lengths for a BPSK-modulated AWGN channel with SNR E_b/N_0. The BP decoder was allowed a maximum of 50 decoding iterations. The BP decoder stops when either a valid codeword is found or the maximum number of decoding iterations is reached. All codes in the figure have $j = 3$ and $k = 5$. The algebraic construction[12] is seen to outperform the random construction for short to moderate block lengths (up to 1000 in the figure) [28]. However, at longer block lengths, the algebraic constructions are not as good as the random constructions.

The algebraic constructions show an error floor, which may be due to their limited minimum distance. For $j = 3$, the minimum distance of the algebraically constructed codes is at most $(j+1)! = 4! = 24$ [17]. The minimum distance of random codes, on the other hand, can grow linearly with the block length [10], and the random codes in the figure do not exhibit an obvious error floor behavior.

The performance of the $(3,5)$ regular $[755, 334]$ code $(m = 151)$ of Example 8 (see Appendix A) is shown in Fig. 8. This code has atypical behavior compared to the other $(3,5)$ LDPC codes; despite its good waterfall performance, the code's small minimum distance is reflected in the poor error floor behavior. Because of the linear dependency among rows, there is a rate gain in this case that results in a waterfall performance better than what one would expect from the degree profile of the code. In contrast, the $(3,5)$ regular $[905, 364]$ code $(m = 181)$ has a good minimum distance and its performance at high SNRs, as shown in Fig. 7, is superior to the performance of a block length 905 randomly constructed $(3,5)$ regular LDPC code. We conjecture that the minimum distance of this code is 24 (achieving the upper bound of $(j+1)!$ [17]), since it was observed during simulations that whenever the BP decoder converged to a wrong codeword, that wrong codeword was at a Hamming distance of 24 from the correct codeword.

Fig. 9 compares the performance of the algebraically constructed regular LDPC codes having $j = 5$ and $k = 7$ with regular randomly constructed $(5,7)$ LDPC codes; in this case,

[11]The regular random LDPC matrices were constructed using the online software available at `http://www.cs.toronto.edu/~radford/software-online.html`. This program generates random regular LDPC codes of any specified rate and block length and is capable of expurgating four cycles in the LDPC constraint graph.

[12]It was observed that the algebraic construction yielded codes that perform at least as well as random codes primarily for prime circulant sizes m.

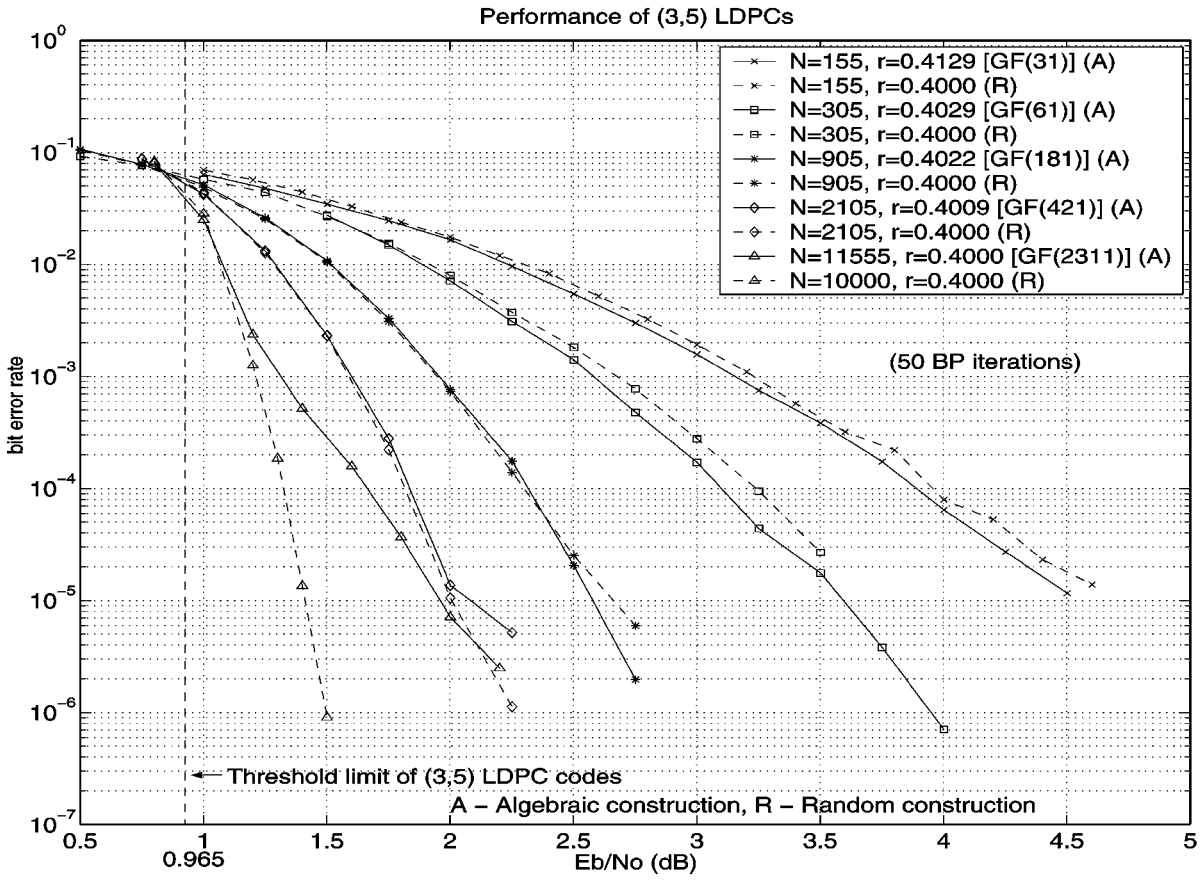

Fig. 7. Algebraic versus random construction of regular $(3,5)$ LDPC codes.

the algebraically constructed codes are found to perform as well as the random codes for block lengths up to around 100 000 and they clearly outperform the random codes at shorter block lengths [28]. Further, we do not see any evidence of the error floor problems observed in the longer $(3,5)$ codes, possibly because the $(5,7)$ codes have larger minimum distances, consistent with the larger bound of $(j+1)! = 6!$.

Fig. 10 compares the performance of the original regular construction with the modified irregular construction for codes derived from $(5,7)$ LDPC matrices. The irregular construction gives rise to LDPC codes for which the BP decoder converges at a smaller channel E_b/N_0. The thresholds of the irregular codes are superior to the thresholds of the regular codes. The threshold of the regular $(5,7)$ codes is 2.65 dB, whereas the irregular codes have a threshold of only 0.87 dB. Hence, the waterfall regions of the BER performance curves for the irregular codes occur at much lower SNR than those of the regular codes. However, the main drawback of the irregular construction is that it can yield codes with poor minimum distance, resulting in high error floors.

B. LDPC Convolutional Codes

The LDPC convolutional codes obtained in this paper typically have large constraint lengths. Therefore, the use of trellis-based decoding algorithms is not feasible. As has been noted, the convolutional code constraint graphs are sparse, and hence the BP algorithm can be used for decoding. As described in [29], BP decoding of LDPC convolutional codes can be scheduled so that decoding results are output continuously after an initial delay. Let m_s denote the memory order, i.e., the largest power of D, in the syndrome former matrix[13] $H^T(D)$. If I decoding iterations are allowed, then the initial decoding delay is $I \times (m_s+1)$. Further, decoding can be scheduled so that the I iterations are carried out independently and in parallel by I identical processors [29].

Consider the convolutional code defined by

$$H(D) = \begin{bmatrix} 1+D & 1 & D \\ D^2 & 1+D & 1 \end{bmatrix}$$
$$G(D) = \begin{bmatrix} 1+D+D^2 & 1+D+D^3 & 1 \end{bmatrix}.$$

Fig. 11 shows a part of the constraint graph of this convolutional code. The memory order of the syndrome former is $m_s = 2$. Suppose now that we allow a maximum of I decoding iterations. Then the decoding window, shown by dotted lines in the figure, for I iterations, is comprised of $(m_s+1) \times I = 3 \times I = 3I$ time units. Decoding results are output after an initial delay equal to the decoding window size i.e., $(m_s+1) \times I = 3I$ time units. Channel values enter the decoding window from the left and decoded results are output from the right. As the bits move through the decoding window, successive iterations are carried out by successive processors, i.e., the ith iteration on a bit is performed by the ith processor. Further, each processor needs to update only the $c-b = 2$ constraint nodes and $c = 3$ variable nodes referred to as "active" nodes in the figure.

[13]We assume here that there are no common factors in any column of $H^T(D)$.

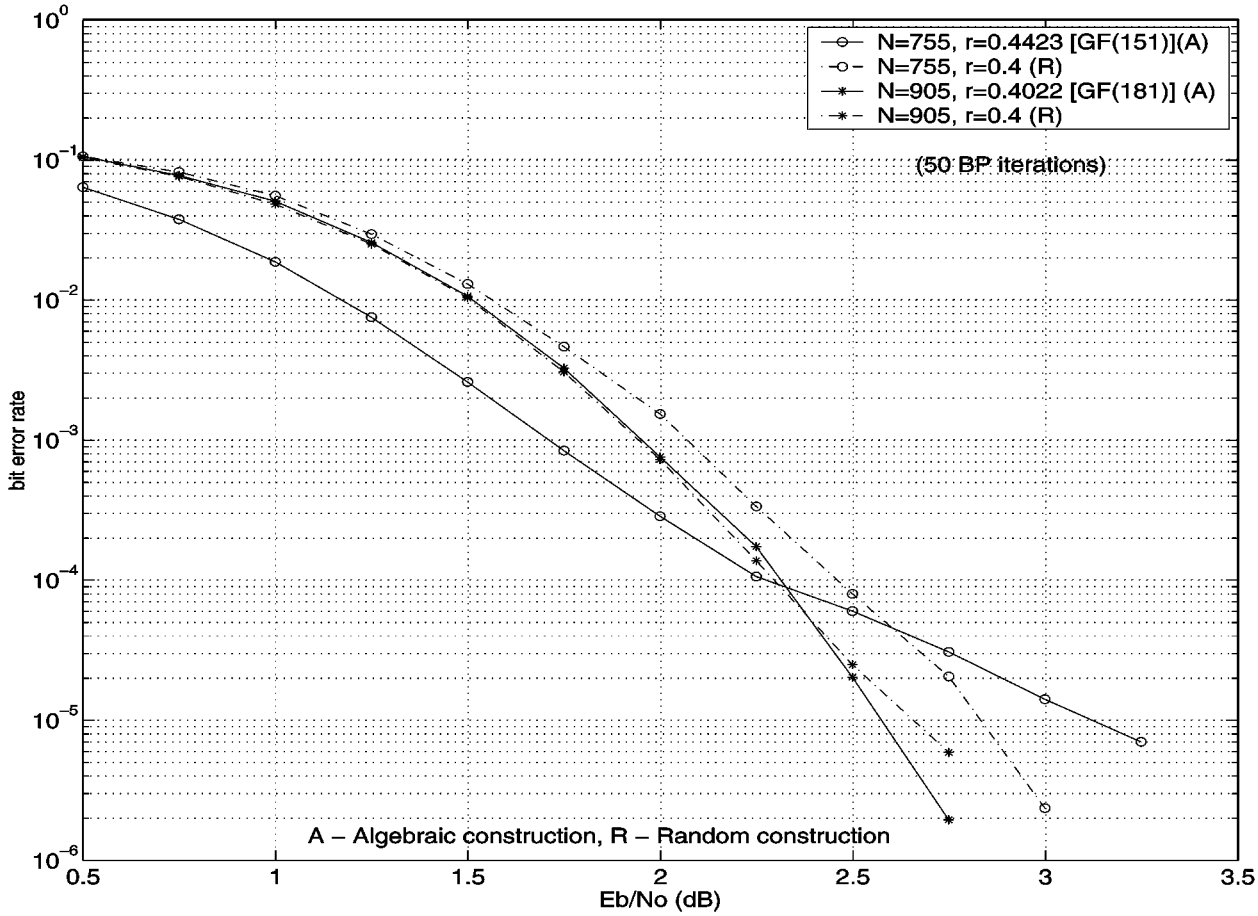

Fig. 8. Performance of two regular $(3, 5)$ algebraically constructed LDPC codes: atypical behavior of the $[755, 334]$ code.

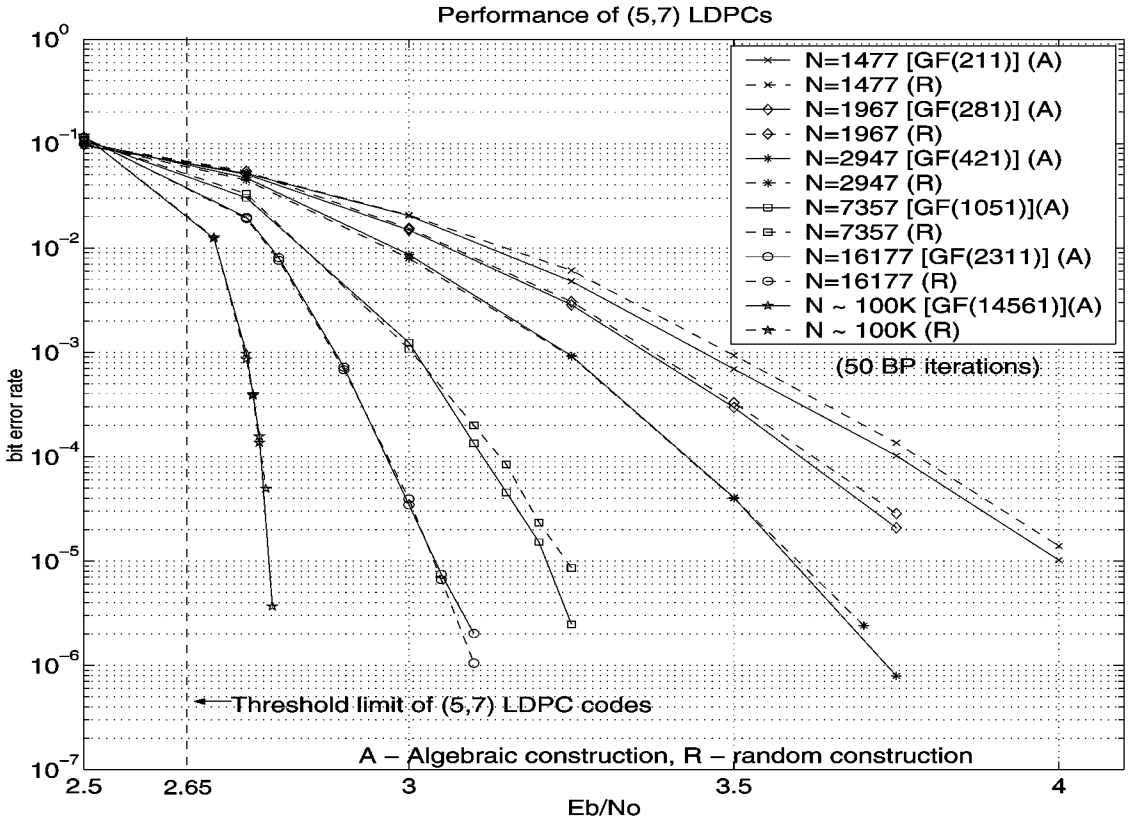

Fig. 9. Algebraic versus random construction of $(5, 7)$ LDPC codes.

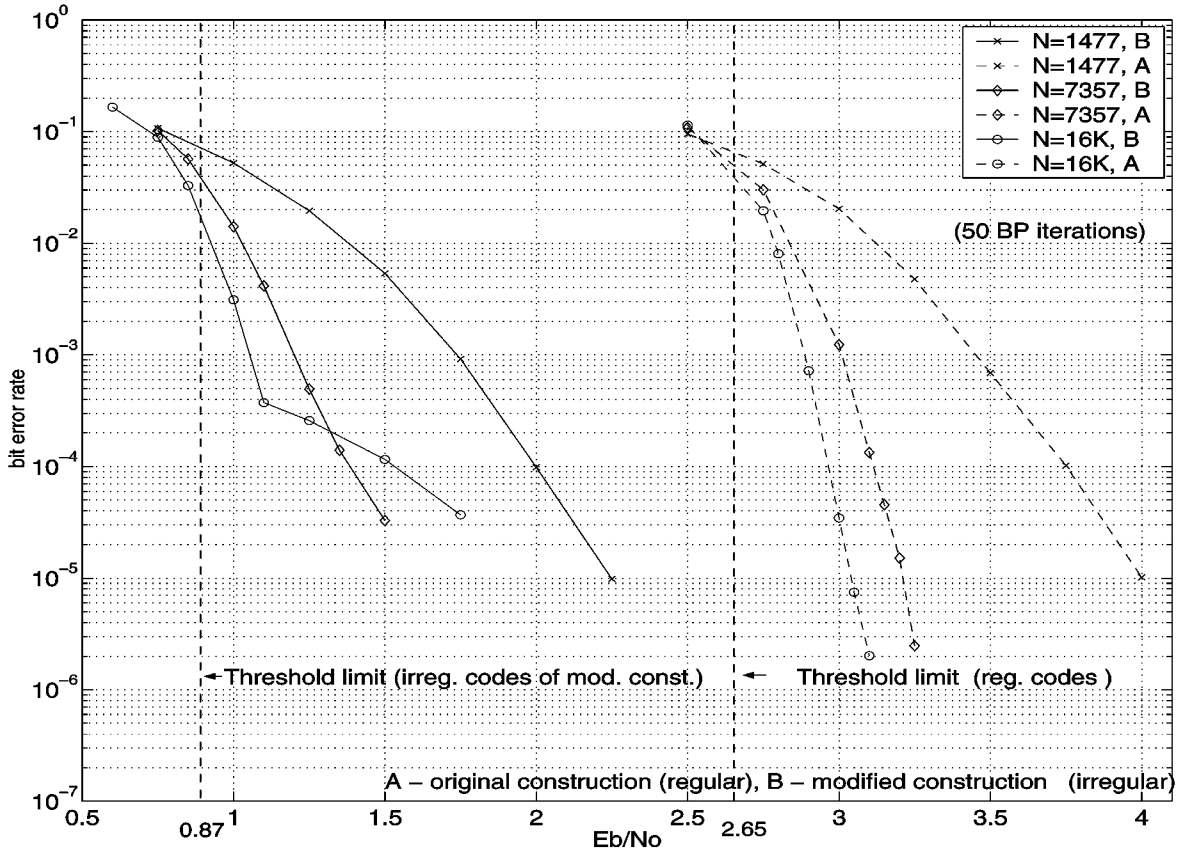

Fig. 10. Rate–$2/7$ LDPC codes: Regular versus irregular constructions.

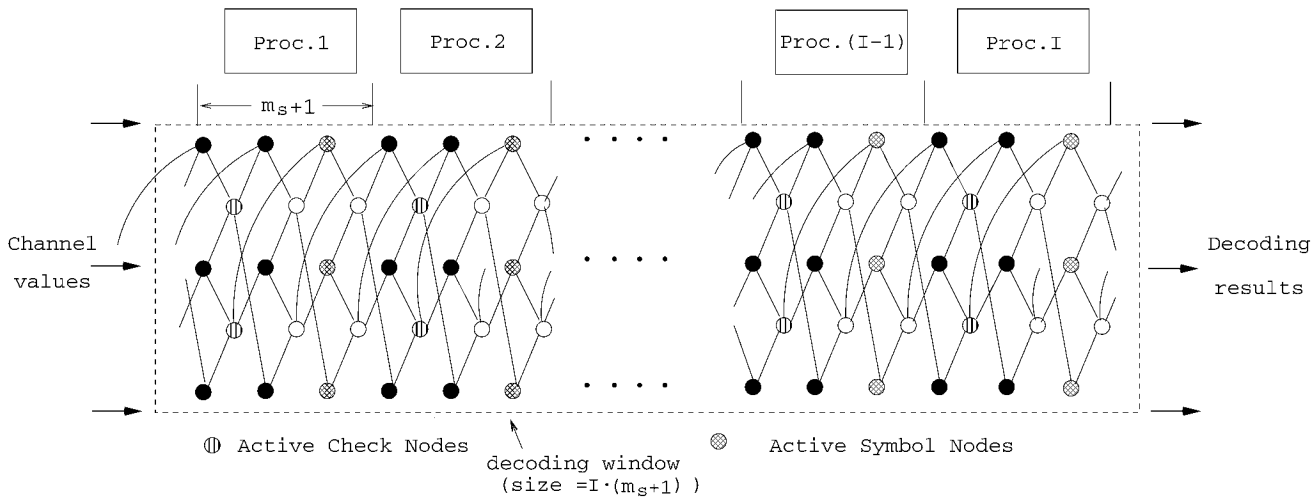

Fig. 11. Continuous decoding of an LDPC convolutional code with I iterations.

Fig. 12 shows the performance of the $(3,5)$ LDPC convolutional codes and that of the corresponding QC codes for a BPSK-modulated AWGN channel assuming 50 BP iterations. We see that the LDPC convolutional codes significantly outperform their block-code counterparts in the waterfall region [30]. At higher SNRs, the convolutional codes exhibit an error floor and the improvement obtained with respect to the block code is reduced. (Occasionally, with the $(3,5)$ convolutional LDPCs, the BP algorithm needed 200–300 iterations to converge. We found that using an *ad hoc* limiting factor on the reliabilities passed in BP decoding insured faster convergence, i.e., within 50 iterations).

Fig. 13 shows the performance of the $(5,7)$ LDPC convolutional codes and the corresponding QC block codes. Again, the convolutional codes outperform the block codes. In this case, there is no apparent error floor exhibited by the convolutional codes. Observe that the waterfall for the $m_s = 966$ convolutional code occurs at a lower SNR than the threshold for regular $(5,7)$ LDPC block codes. This may be because the initial check nodes of the convolutional code have degree less than

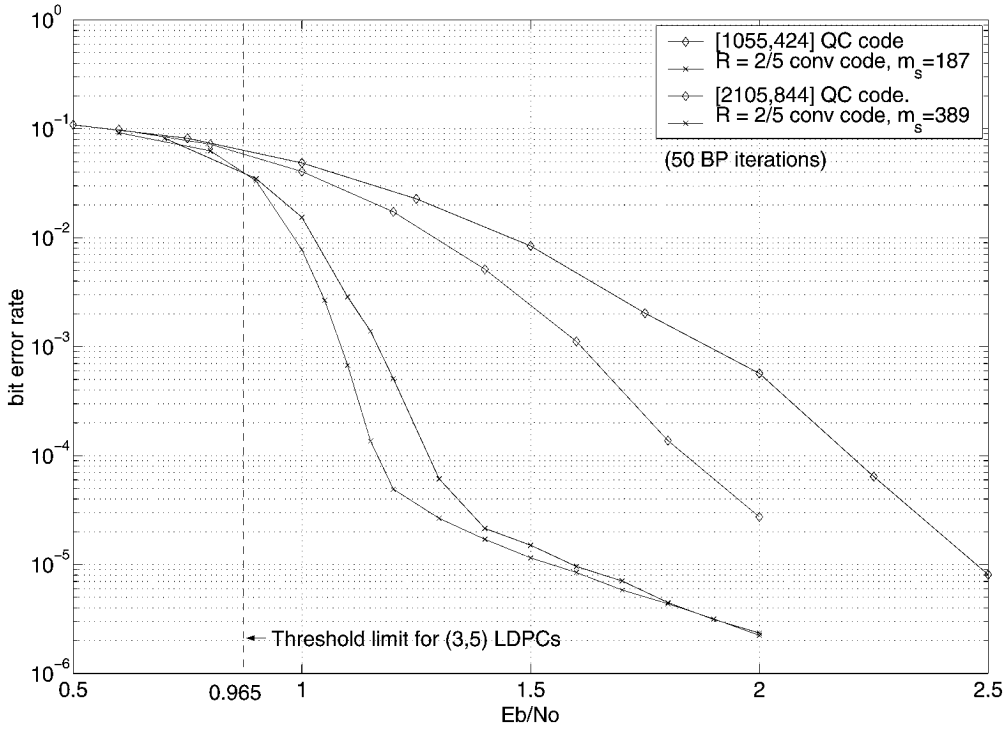

Fig. 12. The performance of convolutional LDPC codes versus the corresponding QC block LDPC codes for $(3, 5)$ connectivity.

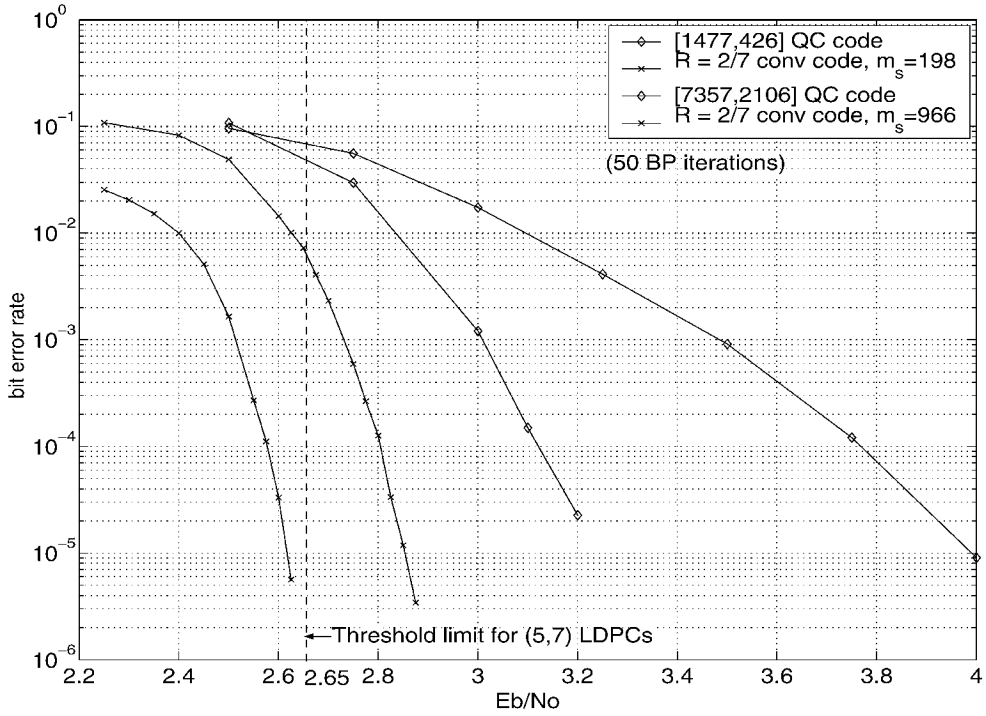

Fig. 13. The performance of convolutional LDPC codes versus the corresponding QC block LDPC codes for $(5, 7)$ connectivity.

k, resulting in a slightly irregular structure (see [31]). Fig. 14 shows the performance of a rate-0.8239, block length 5219, $(3, 17)$ regular QC LDPC code (Example 4) and that of the corresponding rate-$14/17$ LDPC convolutional code derived from this QC code.The figure shows that, for a maximum number of 100 BP iterations, the convolutional code performance again surpasses that of the base QC code.

The convolutional code can be viewed as being "unwrapped" by extending the size of each circulant block in the parity-check matrix of the QC block code to infinity. This "unwrapping" is the principal reason for the performance gain of the convolutional code over the QC block code. To illustrate this, we form a class of block codes with parity-check matrices of increasing circulant sizes, while retaining the same structure within each

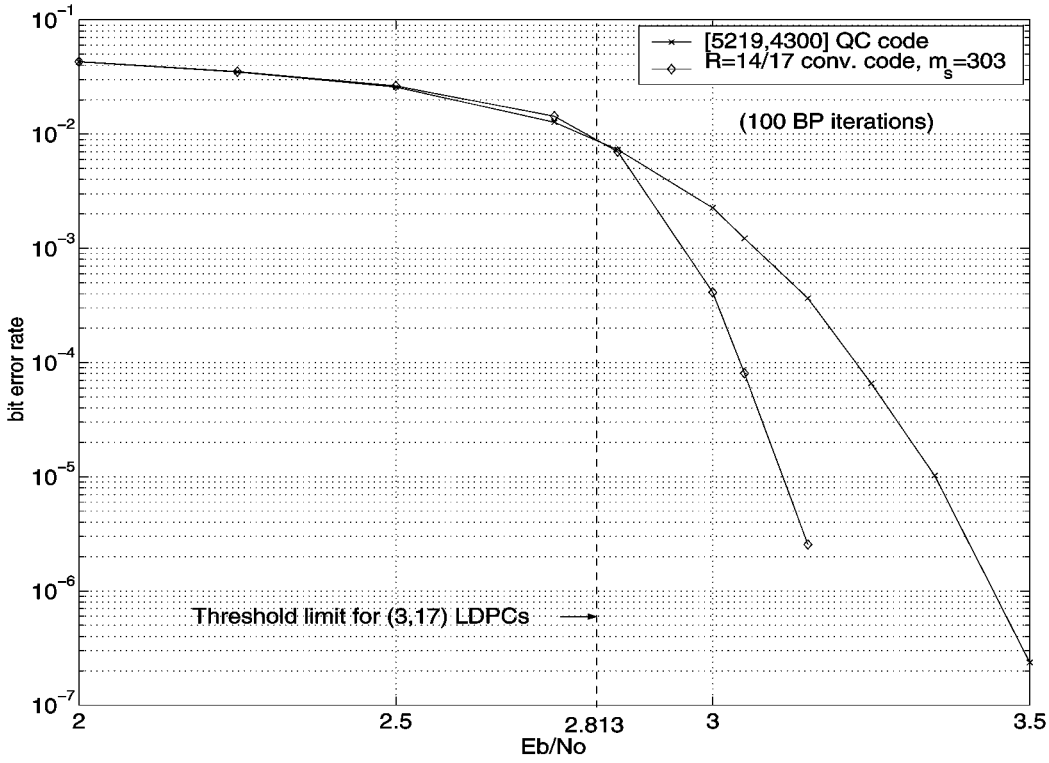

Fig. 14. The performance of convolutional LDPC codes versus the corresponding QC block LDPC codes for $(3, 17)$ connectivity.

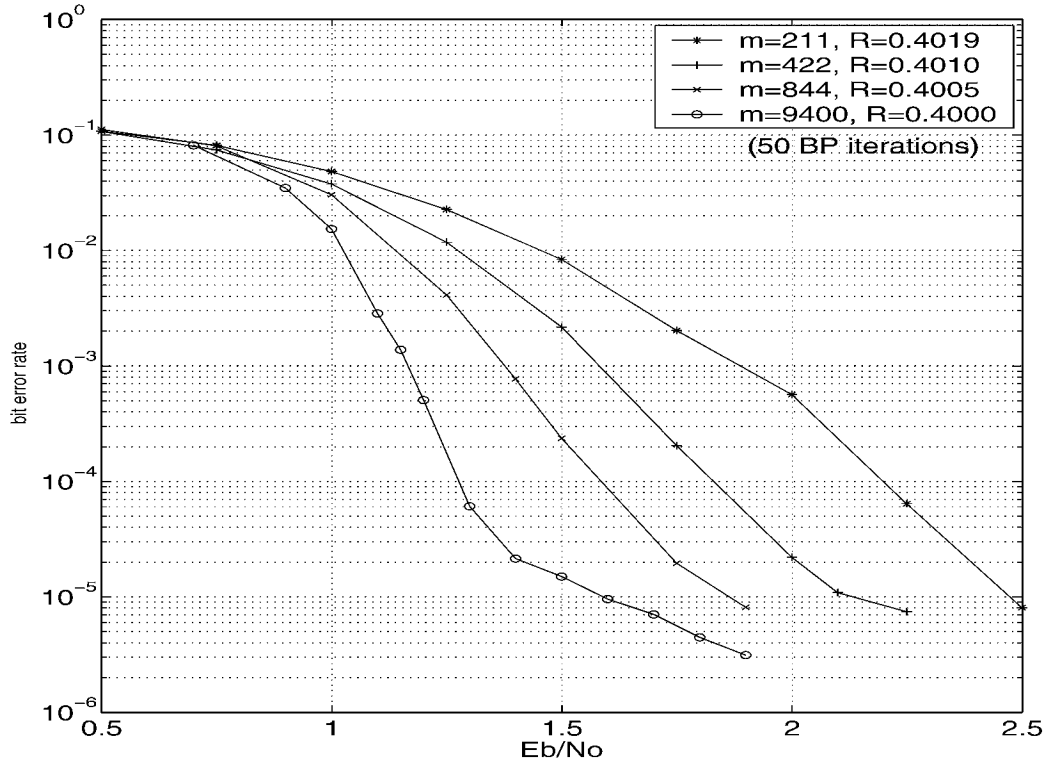

Fig. 15. Performance of regular $(3, 5)$ QC block codes with increasing circulant size m.

circulant. As an example, consider the block length $N = 1055$ $(3, 5)$ regular QC code. The circulant submatrices are of size $m = 211$ in this case. Increasing the circulant size m to 422 and 844 gives QC codes of block lengths $N = 2110$ and $N = 4220$, respectively. The performance of these block codes with BP decoding is shown in Fig. 15. The figure also shows the perfor-

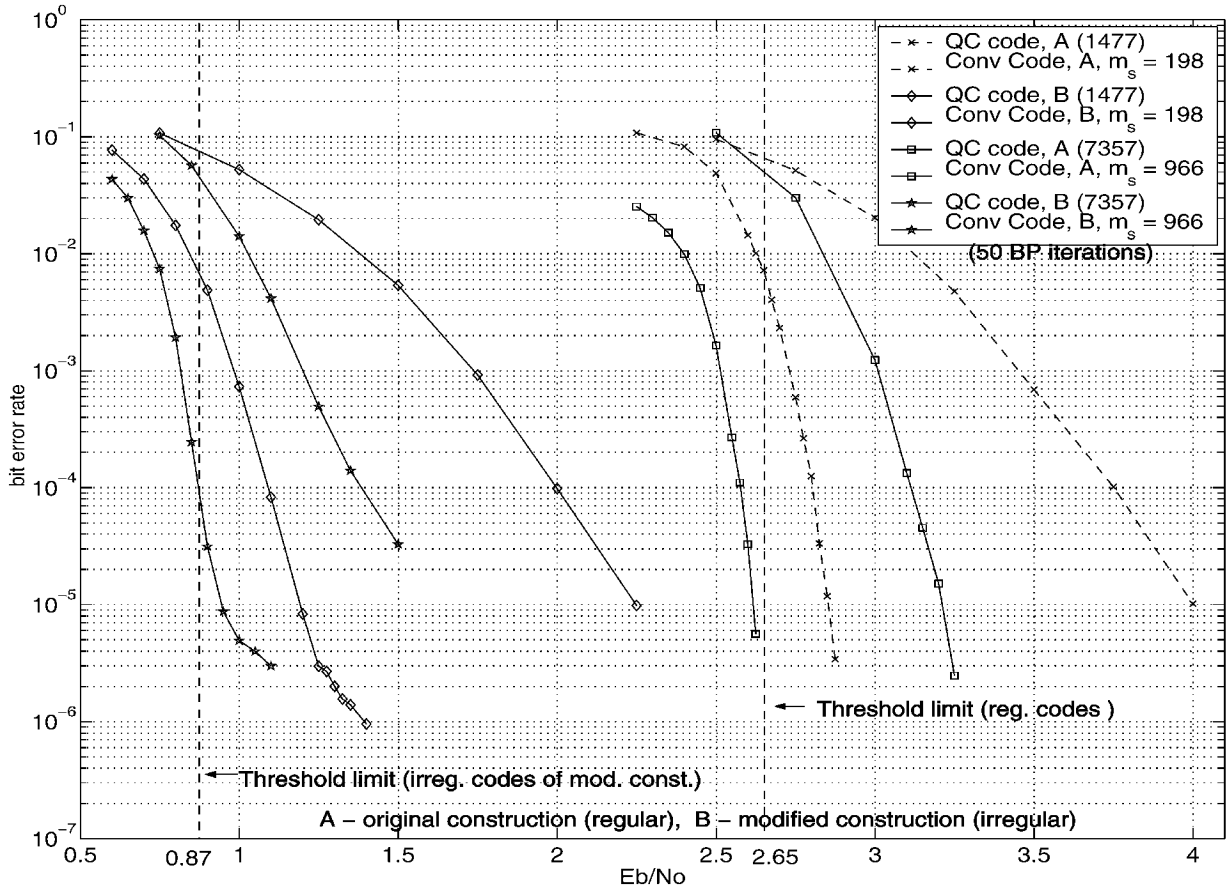

Fig. 16. Performance of convolutional codes versus corresponding QC block codes for $(5, 7)$ regular and irregular constructions.

mance of the corresponding convolutional code. The number of BP iterations is fixed at 50 in all cases.[14] The gradual improvement in performance of the block codes illustrates that the convolutional code represents the limiting case of the block codes as the circulant size increases.

In Fig. 16, the performance obtained with $(5, 7)$ regular and irregular convolutional codes are compared. As for the block codes, the irregular codes show a significant improvement in the waterfall performance. Moreover, as in Fig. 13, the waterfall for the irregular convolutional codes derived from $(5, 7)$ regular convolutional codes appears before the threshold of irregular LDPC block codes having the same degree profile.

V. Conclusion

An algebraic construction based on circulant matrices for designing LDPC codes has been proposed. The properties of the LDPC codes resulting from this construction have been examined. Given the nature of the construction, the codes obtained are QC, which allows for the possibility of deriving corresponding LDPC convolutional code representations. The convolutional representations allow for continuous encoding and decoding.

The proposed codes perform comparably to random LDPC codes at short to moderate block lengths—making them potential candidates for practical applications. The algebraic structure of the codes makes them particularly suitable for high-speed VLSI implementation [32].

A modified construction to yield irregular LDPC codes has also been presented. The irregular LDPC codes perform well in the low-SNR regime, but some suffer from poor distance. Generalizations to the proposed construction may exist for designing irregular codes with good distance that retain the promising features of the original construction.

The LDPC convolutional codes that are obtained from this construction are time-invariant convolutional codes with a variety of rates and constraint lengths. The performance of BP decoding on these codes has been illustrated using a continuous decoding procedure. A significant performance gain is obtained by considering the convolutional representations instead of the original QC versions.

Appendix

The following $(3, 5)$ LDPC code is designed from GF (151).

- Elements $a = 8$, $b = 32$ are chosen from GF (151); then $o(a) = 5$, $o(b) = 3$, and the LDPC matrix is given by

$$H = \begin{bmatrix} I_1 & I_8 & I_{64} & I_{59} & I_{19} \\ I_{32} & I_{105} & I_{85} & I_{76} & I_4 \\ I_{118} & I_{38} & I_2 & I_{16} & I_{128} \end{bmatrix}_{453\times 755}.$$

[14]Since we are limiting the number of iterations for the convolutional code to 50 and m_s in this case is 187, this implies that BP decoding for the convolutional code occurs across nodes up to $50 \times (187 + 1)$ time units away. So its behavior is expected to be equivalent to that of a block code with a circulant size of $m = 9400$.

The code shows atypical behavior in comparison with the other regular $(3, 5)$ LDPC codes constructed here. The rank drop in the parity-check matrix is 32 and not $j - 1 = 2$, as it is for most of the other $(3, 5)$ codes. Further, the code has a small minimum distance ($d_{\min} = 14$) that leads to a poor error floor behavior with belief propagation decoding. Despite the significant rank loss in the QC code, the corresponding LDPC convolutional code has a parity-check matrix with full rank, i.e., it is a rate–$2/5$ convolutional code.

REFERENCES

[1] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, pp. 58–60, Feb. 2001.

[2] R. M. Tanner, "A $[155, 64, 20]$ sparse graph (LDPC) code," presented at the Recent Results Session at IEEE International Symposium on Information Theory, Sorrento, Italy, June 2000.

[3] Y. Kou, S. Lin, and M. Fossorier, "Low density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711–2736, Nov. 2001.

[4] J. Rosenthal and P. O. Vontobel, "Constructions of LDPC codes using Ramanujam graphs and ideas from Margulis," in *Proc. 38th Allerton Conf. Communications, Control, and Computing*, Monticello, IL, Oct. 2000, pp. 248–257.

[5] R. M. Tanner, "On quasi-cyclic repeat accumulate codes," in *Proc. 37th Allerton Conf. Communication, Control and Computing*, Monticello, IL, Oct. 1999, pp. 249–259.

[6] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inform. Theory*, to be published.

[7] J. L. Fan, "Array codes as low-density parity check codes," in *Proc. 2nd Int. Symp. Turbo Codes and Related Topics*, Brest, France, Sept. 2000, pp. 543–546.

[8] R. M. Tanner, "Convolutional codes from quasi-cyclic codes: A link between the theories of block and convolutional codes," Univ. Calif. Santa Cruz, Tech. Rep., 1987.

[9] Y. Levy and D. J. Costello, Jr., "An algebraic approach to constructing convolutional codes from quasi-cyclic codes," *DIMACS Ser. Discr. Math. Theor. Comput. Sci.*, vol. 14, pp. 188–198, 1993.

[10] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[11] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, Feb. 2001.

[12] S. L. Hakimi and J. Bredeson, "Graph theoretic error-correcting codes," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 584–591, July 1968.

[13] R. M. Tanner, D. Sridhara, and T. E. Fuja, "A class of group-structured LDPC codes," in *Proc. Int. Symp. Communications Theory and Applications*, Ambleside, U.K., July 2001.

[14] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.

[15] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of constructions of irregular Gallager codes," *IEEE Trans. Communications*, vol. 47, pp. 1449–1454, Oct. 1999.

[16] S.-Y. Chung, T. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low density parity-check codes using a Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, pp. 657–670, Feb. 2001.

[17] D. J. C. MacKay and M. C. Davey, *Evaluation of Gallager Codes for Short Block Length and High Rate Applications*, 2001, vol. 123, IMA volumes in Mathematics and its Applications, ch. 5, pp. 113–130.

[18] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. Piscataway, NJ: IEEE Press, 1999.

[19] H. H. Ma and J. K. Wolf, "On tail-biting convolutional codes," *IEEE Trans. Commun.*, vol. COM-34, pp. 104–111, Feb. 1986.

[20] G. Solomon and H. C. A. van Tilborg, "A connection between block and convolutional codes," *SIAM J. Appl. Math.*, vol. 37, no. 2, pp. 358–369, Oct. 1979.

[21] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Upper Saddle River, NJ: Prentice-Hall, 2004.

[22] R. M. Tanner, "Minimum distance bounds by graph analysis," *IEEE Trans. Inform. Theory*, vol. 47, pp. 808–821, Feb. 2001.

[23] A. Orlitsky, R. Urbanke, K. Vishwanathan, and J. Zhang, "Stopping sets and the girth of Tanner graphs," in *Proc. 2002 IEEE Int. Symp. Information Theory*, Lausanne, Switzerland, June/July 2002, p. 2.

[24] T. Richardson and R. Urbanke, "Efficient encoding of low density parity check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[25] R. M. Tanner, "A transform theory for a class of group invariant codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 725–775, July 1988.

[26] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical erasure resilient codes," in *Proc. 29th Annu. ACM Symp. Theory of Computing, (STOC)*, 1997, pp. 150–159.

[27] C. Di, D. Proietti, İ. E. Teletar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1570–1579, June 2002.

[28] D. Sridhara, T. E. Fuja, and R. M. Tanner, "Low density parity check codes from permutation matrices," in *Proc. Conf. Information Sciences and Systems*, Baltimore, MD, Mar. 2001, p. 142.

[29] A. J. Feltström and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inform. Theory*, vol. 45, pp. 2181–2191, Sept. 1999.

[30] A. Sridharan, D. J. Costello, Jr., D. Sridhara, T. E. Fuja, and R. M. Tanner, "A construction for low density parity check convolutional codes based on quasi-cyclic block codes," in *Proc. 2002 IEEE Int. Symp. Information Theory*, Lausanne, Switzerland, June 2002, p. 481.

[31] A. Sridharan, M. Lentmaier, D. J. Costello, Jr., and K. S. Zigangirov, "Convergence analysis of a class of LDPC convolutional codes for the erasure channel," in *Proc. 42nd Allerton Conf. Communication, Control and Computing*, Monticello, IL, Oct. 2004.

[32] K. Yoshida, J. Brockman, D. Costello, Jr., T. Fuja, and R. M. Tanner, "VLSI implementation of quasi-cyclic LDPC codes," in *Proc. 2004 Int. Symp. Information Theory and Its Applications*, Parma, Italy, Oct. 10–13, 2004, pp. 551–556.