Design of Structured Irregular LDPC Codes

Peter Trifonov
Saint-Petersburg State Polytechnic University, Russia, petert@dcn.infos.ru

***Abstract*—A novel construction of irregular LDPC codes based on cyclic shift matrices is presented. The construction allows compact specification of LDPC codes of arbitrary length. An optimization algorithm is presented for finding the parameters of the code.**

I. INTRODUCTION

Low density parity check codes have gained recently a lot of interest due to their excellent performance. Despite of substantial progress in the asymptotical analysis of LDPC codes [1], design of good codes of short and moderate length still remains an open problem. The main reason for this is that density evolution, the most widely use LDPC code analysis tool, allows one to obtain only very high-level information about the ensemble of codes, such as node degree distribution of the associated Tanner graph. However, this ignores the important properties of finite-length codes, such as stopping sets and minimum distance. Furthermore, compact representation of code parity check matrix is needed in order to implement LDPC coding in a practical system. On the other hand, most of the existing algebraic LDPC code design techniques lead to structured LDPC codes, which are regular and inherently lack the capacity approaching behaviour.

In this paper a systematic method for finding good structured irregular LDPC codes is presented. The paper is organized as follows. Section II introduces the motivation and necessary background of the new code construction. Section III describes the proposed polynomial based construction and an algorithm for optimization of its parameters. Numeric results are presented in Section IV. Finally, some conclusions are drawn in Section V.

II. MOTIVATION AND BACKGROUND

Low density parity check codes are defined as linear block codes with a sparse parity check matrix. Each $r \times n$ parity check matrix H can be equivalently characterized by a bipartite Tanner graph with n variable and r check nodes, so that the i-th check node is connected to the j-th variable node iff $h_{ij} = 1$. Decoding of LDPC codes is usually performed by the belief propagation algorithm, which operates by passing messages over the network defined by Tanner graph [2].

It was shown in [1] that the performance of an ensemble of infinite-length LDPC codes decoded by the belief propagation algorithm is characterized by the iterative decoding threshold, which depends on the node degree distribution of the associated Tanner graphs. Tables of optimized node degree distributions are available [1]. For finite length codes, the performance depends also on code minimum distance and

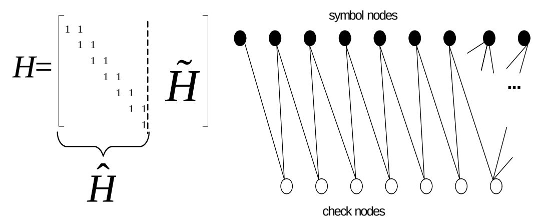

Fig. 1. Zigzag pattern

configuration of stopping sets in the Tanner graph. Short (especially length 4) loops in the Tanner graph are known to degrade the performance of the message-passing decoding algorithm [2]. However, it was shown in [3] that not all short loops are equally harmful, and approximate cycle extrinsic message degree metric (ACE) was suggested for characterization of loop impact on the code performance. It was shown in [4] that by imposing a constraint on minimum ACE of a Tanner graph one can limit the number of small stopping sets and increase the minimum distance of the code. Furthermore, many constructions of good LDPC codes with length-6 loops were proposed recently (see for example [5]).

Numeric optimization reveals that the Tanner graphs of good LDPC codes should have significant number of degree-2 nodes, no degree-1 nodes, and some nodes of degree 3 and higher [1]. It is known that short loops in the Tanner graph involving only degree-2 nodes are particularly harmful for performance of short and moderate length codes. Therefore, it is strongly desired to arrange them into a loop-free configuration.

In practical systems one can usually implement only codes with structured parity check matrices. A widely used approach to construction of structured low-density parity check matrices is expansion of some template matrix [6], [7], [13]. However, it is quite difficult to find a template matrix with any explicit guarantees of code minimum distance. In this paper a different approach is adopted. Namely, the template matrix is used to construct only a part, denoted here by $\tilde{H}$, of the check matrix. The template matrix is optimized in order to maximize the estimated code minimum distance.

III. STRUCTURED LDPC CODES

This section introduces a novel construction of structured LDPC codes and presents some techniques which can be used for optimization of code parameters.

A. Code construction

A simple way to avoid short loops involving degree-2 nodes in the Tanner graph is to arrange them into the zigzag pattern, which is illustrated in Figure 1. It corresponds to bidiagonal submatrix $\hat{H}$ in the parity check matrix. Observe that the zigzag pattern involves one degree-1 node, which should not affect the performance of a sufficiently long code. It will be assumed here that there are no degree-2 nodes in the Tanner graph except those connected to the zigzag pattern. Hence, the parity check matrix of the proposed family of codes can be represented as $H = (\hat{H}|\tilde{H})$, where $\tilde{H}$ is a submatrix containing at least 3 ones in each column. Such codes can be also considered as a generalization of irregular repeat-accumulate codes [8]. Similar construction was also considered in [9].

Since $\hat{H}$ is a square $r \times r$ non-singular matrix, any valid codeword $c = (\hat{c}|\tilde{c})$ of such code must satisfy $\hat{c}^T = \hat{H}^{-1}\tilde{H}\tilde{c}^T$, where $\tilde{c}$ is the vector of information symbols, $\hat{c}$ is the vector of check symbols. The structure of the zigzag pattern allows one to calculate the weight of vector $y = \hat{H}^{-1}x$ as

$$S(x) = \sum_{j=0}^{\lceil l/2 \rceil - 1} (x_{l-2j} - x_{l-2j-1}), \qquad (1)$$

where $x_j : 0 \leq x_j < x_{j+1} < \ldots \leq r-1, j = 1..l$ are positions of non-zero elements in vector x, and $x_0 = -1$.

Submatrix $\tilde{H}$ should be constructed under the constraints of code minimum distance, node degree distribution and stopping set avoidance. We propose to construct this matrix as $\tilde{H} = \Pi A$, where Π is a permutation matrix, and A is a block matrix consisting of cyclic shift matrices and zero matrices. More specifically, let $r = ps$ and $n - r = pt$ be the number of rows and columns in $\tilde{H}$, where p, s and t are some positive integers. Let $\Lambda(x) = \sum_{i \geq 2} \Lambda_i x^{i-1}$ be the target variable node degree distribution, where Λ_i is the fraction of variable nodes of degree i in the Tanner graph to be constructed, and $\Lambda_2 = \frac{r}{n}$. Let $t_i p = \Lambda_i pt$ be the number of variable nodes of degree $i > 2$, so that $\sum_i t_i = t$. Appropriate rounding may be necessary in order to obtain integer values t_i. Let $w_j, j = 0..t-1$ be the sequence of integers such that $|\{j|w_j = i\}| = t_i$ and $w_j \leq w_{j+1}$. Matrix A contains submatrices $\Delta^{p_{ij}}$ in positions $(\phi_{ij}, j), i = 1..w_j$, where $\Delta = \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 & 1 \\ 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{pmatrix}$ is the $p \times p$ permutation matrix corresponding to one-step cyclic shift, and $\phi_{i,j} < \phi_{i+1,j}$. The remaining positions of A are filled with zero matrices.

The permutation matrix Π should be designed so that the vectors obtained as $\Pi A \tilde{c}^T$ for low-weight $\tilde{c}$ do not transform into low-weight vectors $\hat{c}^T = \hat{H}^{-1}\Pi A \tilde{c}^T$. Equation (1) implies that this can happen if $\Pi A \tilde{c}^T$ contains closely located 1's. A possible way to construct Π, which enables some simplifications in the optimization algorithm described below, is to use permutation (block interleaver) $\Pi(i) = (i \bmod p)s + \left\lfloor \frac{i}{p} \right\rfloor, i = 0..sp-1$. Then matrix $\tilde{H}$ consists of columns $(h_{0,0}, h_{0,1}, \ldots, h_{0,p-1}, h_{1,0}, \ldots, h_{1,p-1}, \ldots, h_{s-1,p-1})$, where column $h_{j,k}$ contains 1's in positions $\phi_{ij} + s(p_{ij} + k) \bmod ps, i = 1..w_j$. Pairs (ϕ_{ij}, p_{ij}) completely specify the parity check matrix of the code. Alternatively, the check matrix can be specified by template matrix P containing p_{ij} in positions (ϕ_{ij}, j) and ∞ in other ones. Then A can be defined as a block matrix with elements $\Delta^{P_{ij}}$, assuming that Δ^{∞} is a zero matrix.

B. Optimization algorithm

The idea of the optimization algorithm presented below is that the low-weight information words $\tilde{c}$ after multiplication by $\hat{H}^{-1}\tilde{H}$ should not be transformed into low-weight $\hat{c}$ vectors, obtaining thus low-weight codewords.

In principle, one can consider all possible information vectors $\tilde{c}$, compute for a given (ϕ_{ij}, p_{ij}) set the vector $x = \tilde{H}\tilde{c}^T$, find the weight of $\hat{c}$ using the expression (1), and deduce the minimum distance d_{min} of the code obtained. Then one can perform maximization of d_{min} over all possible (ϕ_{ij}, p_{ij}) values. However, this approach is quite impractical due to large number of input patterns, and huge size of the search space. Therefore one has to limit the search scope to vectors $\tilde{c}$ of sufficiently small weight, as well as impose some constraints on possible (ϕ_{ij}, p_{ij}) values. In this case one may not be able to obtain an exact value of code minimum distance. Nevertheless, this approach allows one to eliminate many low-weight codewords, reducing thus the decoding error probability.

Optimization of code parameters (ϕ_{ij}, p_{ij}) can be performed iteratively according to the following randomized search algorithm:

1) Let $j := 0$.
2) Generate ϕ_{ij} and p_{ij} randomly, so that $0 \leq \phi_{1j} < \phi_{2j} < \ldots < \phi_{w_j j} < s$ and $0 \leq p_{ij} < p$. A number of constraints on these values will be described below, which enable one to exclude some bad configurations.
3) Make sure that the Tanner graph corresponding to the matrix $\tilde{H}_j$, given by (ϕ_{ij}, p_{ij}) values generated up to now, has minimal ACE not less than a given threshold η. This can be implemented efficiently using the algorithms presented in [3], [11]. Go to step 2 in case of failure.
4) Consider all possible vectors $\tilde{c} \in GF(2)^{(j+1)p}$: $\mathrm{wt}(\tilde{c}) \leq w$. Determine the weights of the associated output vectors using (1), and find the minimal weight of the obtained codewords $(\hat{c}|\tilde{c})$, which is denoted here by S_{min}.
5) Repeat steps 2–4 a given number of times, and select (ϕ_{ij}, p_{ij}) values maximizing S_{min}. Let $\hat{d}_j$ be the obtained maximal value of S_{min}.
6) Let $j := j + 1$. If $j < t$, go to step 2.

Observe that $\hat{d} = \min_j \hat{d}_j$ gives an upper bound on the minimum distance of the code obtained. Clearly, increasing w, the weight of input vectors being analyzed, improves the accuracy of this estimate, but increases also the complexity of the algorithm. Increasing η in general improves the performance of the obtained codes, but may also cause the algorithm to fail

$$\begin{pmatrix}
23 & \infty & \infty & \infty & \infty & \infty & 9 & \infty & \infty & \infty & \infty & \infty & 6 & \infty & \infty & 21 & \infty & 24 \\
0 & \infty & \infty & \infty & \infty & \infty & \infty & 26 & \infty & \infty & \infty & \infty & \infty & 11 & \infty & 16 & \infty & 25 \\
27 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 1 & \infty & \infty & \infty & \infty & 24 & \infty & 26 & \infty & 14 \\
\infty & 7 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 5 & \infty & \infty & \infty & \infty & 18 & 19 & 31 & 10 \\
\infty & 22 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 23 & \infty & \infty & \infty & 31 & 24 & 3 & 8 \\
\infty & 19 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 29 & \infty & \infty & 13 & 23 & 28 & 19 \\
\infty & \infty & 18 & \infty & \infty & \infty & 28 & \infty & \infty & \infty & \infty & \infty & 8 & \infty & \infty & 30 & 18 & 0 \\
\infty & \infty & 16 & \infty & \infty & \infty & \infty & 9 & \infty & \infty & 16 & \infty & \infty & \infty & \infty & 11 & 20 & \infty \\
\infty & \infty & 23 & \infty & \infty & \infty & \infty & \infty & 31 & \infty & \infty & 8 & \infty & \infty & \infty & 17 & 15 & 11 \\
\infty & \infty & \infty & 22 & \infty & \infty & \infty & \infty & \infty & 5 & \infty & \infty & 12 & \infty & \infty & 25 & 26 & \infty \\
\infty & \infty & \infty & 16 & \infty & \infty & \infty & \infty & \infty & \infty & 2 & \infty & \infty & 9 & \infty & 7 & 7 & 24 \\
\infty & \infty & \infty & 5 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 26 & \infty & \infty & 19 & 2 & 26 & \infty \\
\infty & \infty & \infty & \infty & 18 & \infty & 20 & \infty & \infty & \infty & \infty & \infty & 7 & \infty & \infty & 5 & 12 & 4 \\
\infty & \infty & \infty & \infty & 31 & \infty & \infty & 7 & \infty & \infty & 1 & \infty & \infty & \infty & \infty & 19 & 16 & 25 \\
\infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & 8 & \infty & \infty & 15 & \infty & \infty & \infty & 12 & 27 & 17 \\
\infty & \infty & \infty & \infty & \infty & 7 & \infty & \infty & \infty & 1 & \infty & \infty & 0 & \infty & 21 & \infty & 3 & 21 \\
\infty & \infty & \infty & \infty & \infty & 1 & \infty & \infty & \infty & \infty & 5 & \infty & \infty & 31 & \infty & \infty & 7 & 20 \\
\infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & 6 & \infty & 25 & \infty & \infty & 31 & 18
\end{pmatrix}$$

Fig. 2. Template matrix for rate $1/2$ code (Code 1)

$$\begin{pmatrix}
22 & \infty & \infty & \infty & \infty & 19 & \infty & \infty & 59 & \infty & \infty & 24 & \infty & \infty & \infty & 49 \\
14 & \infty & \infty & \infty & \infty & \infty & \infty & 42 & \infty & 51 & \infty & 7 & \infty & \infty & \infty & 60 \\
44 & \infty & \infty & \infty & \infty & \infty & \infty & 29 & \infty & 11 & \infty & 51 & \infty & \infty & \infty & \infty \\
\infty & 23 & \infty & \infty & \infty & 2 & \infty & \infty & \infty & 0 & \infty & \infty & 34 & \infty & \infty & \infty \\
\infty & 51 & \infty & \infty & \infty & \infty & \infty & 3 & \infty & 25 & \infty & \infty & 42 & \infty & \infty & 5 \\
\infty & 54 & \infty & \infty & \infty & \infty & \infty & 46 & \infty & 12 & \infty & \infty & 23 & \infty & \infty & 67 \\
\infty & \infty & 6 & \infty & \infty & 29 & \infty & \infty & \infty & 40 & \infty & \infty & 40 & \infty & 43 & \infty \\
\infty & \infty & 68 & \infty & \infty & \infty & \infty & 7 & \infty & \infty & 11 & \infty & 24 & \infty & 14 & \infty \\
\infty & \infty & 71 & \infty & \infty & \infty & \infty & 36 & \infty & \infty & 36 & \infty & 54 & \infty & \infty & 65 \\
\infty & \infty & \infty & 34 & \infty & \infty & 51 & \infty & \infty & \infty & 12 & \infty & \infty & 61 & \infty & 69 \\
\infty & \infty & \infty & 69 & \infty & \infty & \infty & \infty & 34 & \infty & 23 & \infty & \infty & 44 & 29 & \infty \\
\infty & \infty & \infty & 2 & \infty & \infty & \infty & \infty & 69 & \infty & 54 & \infty & \infty & 71 & 4 & \infty \\
\infty & \infty & \infty & \infty & 1 & \infty & 65 & \infty & \infty & \infty & 30 & \infty & \infty & 36 & \infty & \infty \\
\infty & \infty & \infty & \infty & 18 & \infty & \infty & \infty & 23 & \infty & \infty & 46 & \infty & 39 & \infty & \infty \\
\infty & \infty & \infty & \infty & 16 & \infty & \infty & \infty & 14 & \infty & \infty & 20 & \infty & 14 & 36 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & 33 & \infty & 57 & \infty & \infty & 50 & \infty & \infty & 56 & \infty
\end{pmatrix}$$

Fig. 3. Template matrix for rate $1/2$ code (Code 2)

to identify the appropriate values of ϕ_{ij} and p_{ij}). In practice it is sufficient to set $w \approx 5$ in order to obtain good codes.

It is possible to impose additional constaints on the values ϕ_{ij} and p_{ij} generated at step 2 in order to quickly eliminate some bad configurations. In particular, one can eliminate length-4 loops in the subgraph of the Tanner graph corresponding to $\tilde{H}$ by enforcing the constraint [10]

$$(p_{i_1j_1} - p_{i_2j_1}) + (p_{i_3j_2} - p_{i_4j_2}) \not\equiv 0 \bmod p \qquad (2)$$

for all $i_k, j_k : \phi_{i_1j_1} = \phi_{i_4j_2}, \phi_{i_2j_1} = \phi_{i_3j_2}$. Furthermore, let us consider the case of $\tilde{c} : \mathrm{wt}(\tilde{c}) = 1$. In this case the vector $\tilde{H}\tilde{c}^T$ contains non-zero elements in positions $\phi_{ij} + s(p_{ij} + k) \bmod ps, i = 1..w_j$ for some $j : 0 \leq j < t$ and $k : 0 \leq k < p$. The vectors corresponding to different values of k can be obtained as cyclic shifts of each other. This restricts the set of possible weights of $\hat{c}$ vector. If w_j is an even integer and $\forall i : \phi_{i+1,j} + sp_{i+1,j} > \phi_{i,j} + sp_{i,j}$, this property implies that the weight of vector $\hat{c}^T = \hat{H}^{-1}\tilde{H}\tilde{c}^T$ equals either to $W'_j = \sum_{i=1}^{w_j/2}(\phi_{2i,j} + sp_{2i,j} - \phi_{2i-1,j} - sp_{2i-1,j})$ or to $W''_j = ps - W'_j$. One can impose constraints $W'_j \geq S_1$ and $W''_j \geq S_1$ for sufficiently large S_0, excluding thus from the code being constucted all codewords having $\mathrm{wt}(\tilde{c}) = 1$ and $\mathrm{wt}(\hat{c}) < S_1$. Similar, but slightly more involved constraints can be developed for the case of odd w_j. These constraints can be used at step 2 in order to exclude some values ϕ_{ij} and p_{ij} leading to bad codes. Furthemore, one can represent vector $\tilde{c}$ as a block vector consisting of subvectors of length p. If $\tilde{c}^{(l)}$ is a vector consisting of $\tilde{c}$ subvectors cyclically shifted by l positions, then $\tilde{H}\left(\tilde{c}^{(l)}\right)^T$ can be obtained by shifting cyclically $\tilde{H}\tilde{c}^T$ by ls positions. This allows one to obtain the weight of $\hat{c}$ vector by appropriately changing the summation order in (1), reducing thus the number of different vectors to be considered at step 4.

In many cases it appears that the template matrix optimized for one value of p provides quite good performance for other values as well. This enables one specify a large family of codes with different length with a single template matrix.

IV. Numeric results

Figures 2 and 3 present template matrices for rate $1/2$ codes obtained with the above algorithm for different node degree distributions. Optimization of both codes took 40 minutes on an Athlon-2500 computer. The number z of variable nodes connected to the zigzag pattern is assumed to be equal to the number of check nodes.

Figure 4 presents simulation results illustrating the performance of the codes defined by these matrices. Additionally, it presents the curves corresponding to IEEE 802.16 LDPC codes [7]. It can be seen that these codes have no error floor downto $FER = 5 \cdot 10^{-5}$. The proposed family of codes tend to outperform the 802.16 ones, especially at high signal-to-noise ratios. The first family of codes provides up to 0.2 dB gain compared to the IEEE 802.16 codes.

The proposed construction can be used to obtain codes with any rate. Figure 5 illustrates the performance of the proposed family of codes, as well as those defined in IEEE 802.16 specification, at rate $3/4$. It can be seen that the proposed family of codes again provides better performance. However, it must be recognized that the pure template-based LDPC codes, similar to those defined in IEEE 802.16, may have other advantages, like simpler encoding algorithm.

V. Conclusions

In this paper a method for constructing structured irregular LDPC codes was presented. The code construction is based on a block-permutation matrix with additional row permutation. The parameters of this matrix are derived through an optimization algorithm, which attempts to maximize the estimated code minimum distance. Provided that sufficient computational power is available, the proposed algorithm can produce good estimates of the minimum distance of the code obtained, providing valuable feedback to the code designer.

The codes constructed with the proposed method were shown to outperform the pure template-based codes specified in IEEE 802.16 standard. An important advantage of the proposed construction is that there are no artificial restrictions on p_{ij} values coming from the regular algebraic LDPC constructions (cf. [6], [13]), which cannot be justified in the case of irregular codes. However, depending on the implementation, the encoder for the proposed family of codes may have slightly higher latency than the one of pure template-based codes.

References

[1] T. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions On Information Theory*, vol. 47, no. 2, pp. 619–637, February 2001.

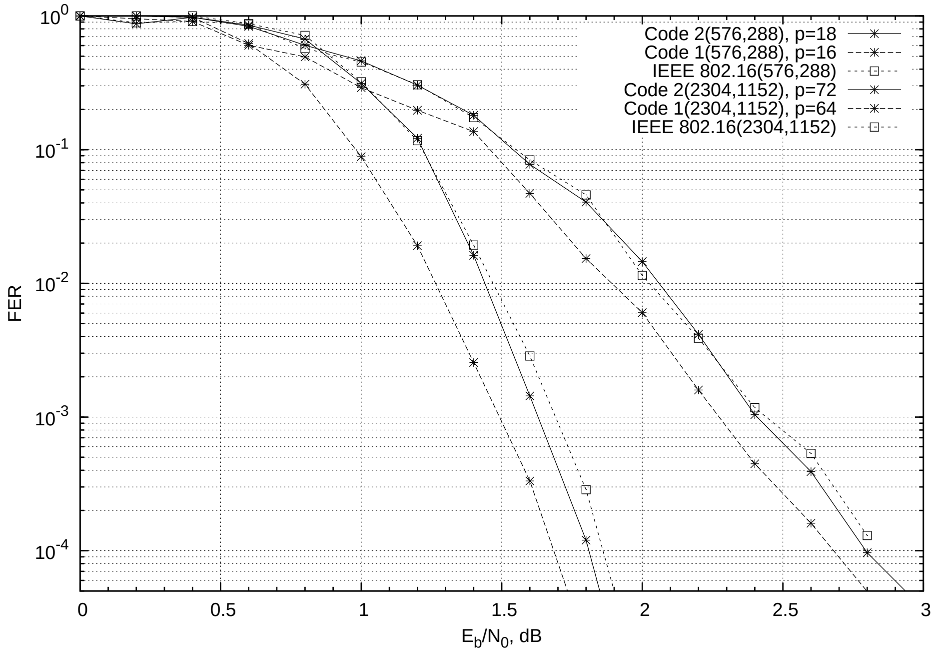

Fig. 4. Performance of rate $1/2$ codes in AWGN channel

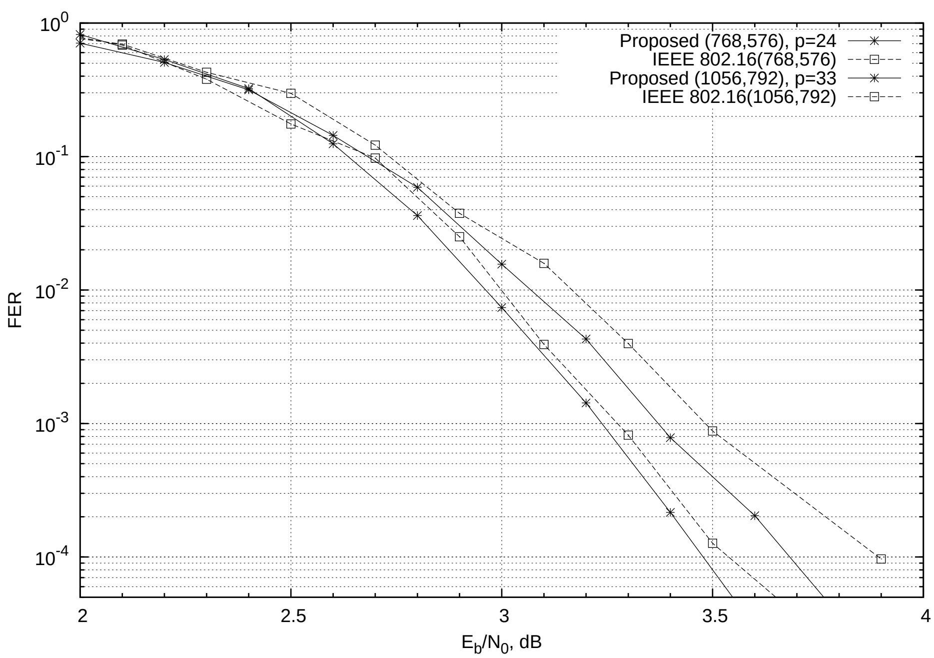

Fig. 5. Performance of rate $3/4$ codes in AWGN channel

[2] W. E. Ryan, “An introduction to LDPC codes,” in *CRC Handbook for Coding and Signal Processing for Recording Systems*, B. Vasic, Ed. CRC Press, 2004.

[3] T. Tian, C. Jones, J. Villasenor, and R. Wesel, “Selective avoidance of cycles in irregular LDPC code construction,” *IEEE Transactions On Communications*, vol. 52, no. 8, August 2004.

[4] A. Ramamoorthy and R. Wesel, “Analysis of an algorithm for irregular LDPC code construction,” in *Proceedings of IEEE International Symposium on Information Theory*, 2004, p. 69.

[5] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, “A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols,” *IEEE Communications Letters*, vol. 7, no. 7, July 2003.

[6] D. E. Hocevar, “LDPC code construction with flexible hardware implementation,” in *Proceedings of IEEE International Conference on Communications*, May 2003, pp. 2708–2711.

[7] B. Classon, Y. Blankenship *et al*, “LDPC coding for OFDMA PHY,” IEEE 802.16 Broadband Wireless Access Working Group, Tech. Rep. C802.16e-05/066r3, 2005.

[8] G. Liva, E. Paolini, and M. Chiani, “Simple reconfigurable low-density parity-check codes,” *IEEE Communications Letters*, vol. 9, no. 3, March 2005.

[9] M. Yang, Y. Li, and W. Ryan, “Design of efficiently-encodable moderate-length high-rate irregular LDPC codes,” *IEEE Transactions on Communications*, vol. 52, no. 4, April 2004.

[10] M. P. Fossorier, “Quasi-cyclic low-density parity-check codes from circulant permutation matrices,” *IEEE Transactions On Information Theory*, vol. 50, no. 8, August 2004.

[11] J. Kang, P. Fan, and Z. Cao, “Flexible construction of irregular partitioned permutation LDPC codes with low error floors,” *IEEE Communications Letters*, vol. 9, no. 6, June 2005.

[12] R. Echard and S.-C. Chang, “Design considerations leading to the development of good π-rotation LDPC codes,” *IEEE Communications Letters*, vol. 9, no. 5, May 2005.

[13] R. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, “LDPC block and convolutional codes based on circulant matrices,” *IEEE Transactions on Information Theory*, vol. 50, no. 12, December 2004.