Design of regular (2,dc)-LDPC codes over GF(q) using their binary images

Charly Poulliat, Marc Fossorier, David Declercq

Design of regular $(2, d_c)$-LDPC codes over GF(q) using their binary images

Charly Poulliat [*†], Marc Fossorier [†], David Declercq [*]

[*] ETIS, ENSEA/UCP/CNRS

6 Avenue du Ponceau

F-95014 Cergy-Pontoise, France.

Email: {charly.poulliat, david.declercq}@ensea.fr

[†] Dept. of Electrical Eng.

University of Hawaii at Manoa

2540 Dole St., Honolulu, HI-96822, USA.

Email: marc@spectra.eng.hawaii.edu

Abstract

In this paper, a method to design regular $(2, d_c)$-LDPC codes over GF(q) with both good waterfall and error floor properties is presented, based on the algebraic properties of their binary image. First, the algebraic properties of rows of the parity check matrix H associated with a code are characterized and optimized to improve the waterfall. Then the algebraic properties of cycles and stopping sets associated with the underlying Tanner graph are studied and linked to the global binary minimum distance of the code. Finally, simulations are presented to illustrate the excellent performance of the designed codes.

Index Terms

channel coding, error correction coding, nonbinary LDPC codes, iterative decoding, binary image.

This work has been partially supported by the Newcom UE Network of Excellence

I. INTRODUCTION

Since their rediscovery in [16], low density parity check (LDPC) codes designed over GF(q) have been shown to approach the Shannon limit performance for $q = 2$ and very long code lengths [15][22]. Some efficient optimization methods of the code profile and the matrix structure have been derived for both long [22][3] and moderate [13] length cases. For fields with parameters $q > 2$, it has been shown that the error performance can be improved for moderate code lengths by increasing q [5][4][11]. It has been shown, especially in [5][11], that as q becomes large ($q \geq 64$) the best performances at finite length are obtained for "ultra-sparse" LDPC codes, that is with the minimum connectivity on the symbol nodes $d_v = 2$. Furthermore, it is shown in [11] that $d_v = 2$ non binary LDPC codes have optimal average Hamming weight spectrum as $q \rightarrow +\infty$ and $N \rightarrow +\infty$ when used on binary input channels. In this paper, we will focus on the finite length optimization of $d_v = 2$ non binary LDPC codes, for which the problem of choosing appropriately the non zero values in the parity check matrix is simplified. Note also that the decoding complexity of codes in GF(q) is a lot larger than for binary codes, but iterative decoding of non binary LDPC codes using the belief propagation (BP) algorithm or its simplified versions has been addressed efficiently by several authors [5][1][6].

The design of non binary LDPC codes can be addressed in order to meet different objectives: (i) performance, by trying to improve the waterfall region and/or to lower the error floor, and (ii) decoding complexity versus performance tradeoff, by trying to ensure good overall performance using only a limited set of parameters for some efficient and practical hardware implementation purposes. For finite length codes, the optimization problem is generally solved in a disjoint manner. First, the positions of the nonzero entries of the parity check matrix H associated with the non binary code are optimized in order to have good girth properties and minimize the impact of cycles, when using the BP algorithm on the associated Tanner graph. This can be efficiently done using the progressive edge growth (PEG) algorithm [13]. Then, the nonzero entries can be selected either randomly from a uniform distribution among nonzero elements of GF(q) [13] or carefully to meet some design criteria as done in [4][17].

In this paper, we address the problem of the selection and the matching of the parity check matrix nonzero entries assuming that the positions of nonzero entries in the parity check matrix H associated with the non binary code have been previously optimized. The proposed method

is based on the binary image representation of the matrix H and of its components. First we address the problem of rows optimization as previously done in [5][17] in order to improve the waterfall region. Then, we address the problem of lowering the error floor: based on the observation that the columns defining the minimum distance in the binary image of H are located on symbols belonging to the shortest length cycles and the associated stopping sets, we propose a method intended to improve the minimum distance of the binary image of the code. To this end, we use the algebraic properties of both cycles and stopping sets, considered as topological substructures inherently present in the underlying Tanner graph of the code. Finally, the complexity-performance tradeoff is addressed: we show for example that for regular $(2, 4)$ and $(2, 8)$ LDPC codes, using only one optimized row of coefficients to generate the parity check matrix, it is possible to have at least the same performance as for a code with randomly selected coefficients and, for some fields, the waterfall and the error floor region can be both improved.

The paper is organized as follows: in Section II, we briefly review the binary image construction of a non binary parity check matrix and the vector representation of the parity check equations. The optimization of the rows of the parity check matrix is addressed for waterfall improvement in Section III. We also study the thresholds under density evolution for random and row optimized code ensembles. Section IV provides a study of the binary representation of both cycles and stopping sets, and establishes links between those topological structures of the Tanner graph and the binary minimum distance property of the code. This study allows us to propose a method to improve the error floor when using the row optimized code ensemble. In Section V, some optimization and simulation results are provided and finally conclusions and perspectives are drawn in Section VI.

II. Binary images of a non binary parity check matrix H

The motivation of using the binary image of the LDPC code is essentially that we address and illustrate the optimization process for the non zero values in the case of binary input additive white gaussian noise (BI-AWGN). In this context, our goal is then to try to maximize the Hamming minimum distance at the bit level of the LDPC code. Note however that using the binary image of the code is not mandatory and one could easily generalize our approach at the symbol level, as it will be notified in sections IV-B, IV-C and IV-E.

Let us consider the parity check matrix H associated with a regular non binary LDPC code

with the parameters (d_v, d_c, N) representing the number of nonzero entries of H for the columns, for the rows and the code length respectively. All the nonzero elements of H are elements of the Galois field GF(q), with $q = 2^p$ and q is the order of the field. Nonzero elements belong to the set $\mathcal{S} = \left\{\alpha^k : k = 0 \dots q-2\right\}$ where α is the primitive element of the field.

A. Representation of the Galois field using matrices

The Galois field GF(q), described usually using a polynomial (or vector) representation, can be also represented using matrices [18, p.106]

Definition 1: If $p(x) = a_0 + a_1 x + \ldots + x^p$ is a polynomial of degree p having its coefficients in GF(2). The companion matrix of $p(x)$ is the $p \times p$ matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ a_0 & a_1 & a_2 & \dots & a_{p-1} \end{pmatrix}$$

The characteristic polynomial of this matrix is given by

$$\det(A - xI) = p(x)$$

where I is the identity matrix.

If $p(x)$ is a primitive polynomial, it can be shown [18] that the matrix A is the primitive element of the Galois field $GF(2^p)$ under a matrix representation and thus the powers of A are the nonzero elements of this field, defining the set $\mathcal{M} = \left\{0, A^k : k = 0 \dots q-2\right\}$. Additions and multiplications in the field correspond to additions and multiplications modulo 2 of these matrices.

B. Vector representation for the parity check equations

Based on the matrix representation of each nonzero entry, we give thereafter the equivalent vector representation of the parity check equations associated with the rows of H.

Let $\mathbf{x} = [x_0 \dots x_{N-1}]$ be a codeword. For the $i-$th parity equation of H, we have

$$\sum_{j:h_{ij}\neq 0} h_{ij} x_j = 0 \tag{1}$$

Translating (1) into the vector domain, we can write

$$\sum_{j:h_{ij}\neq 0} H_{ij}\mathbf{x_j}^t = \mathbf{0}^t$$

where H_{ij} is the transpose of the matrix representation of the Galois field element h_{ij}, $\mathbf{x_j}$ is the vector representation (binary mapping) of the symbol element x_j and t holds for transpose. The vector $\mathbf{0}$ is the all zero component vector.

Considering the i-th parity check equation of H, we define $\mathbf{H_i} = [H_{ij_0} \dots H_{ij_m} \dots H_{ij_{d_c-1}}]$ as the equivalent binary parity check matrix, with $\{j_m : m = 0 \dots d_c - 1\}$ the indexes of the nonzero elements of the $i-$th row. Let $\mathbf{X_i} = [\mathbf{x_{j_0}} \dots \mathbf{x_{j_{d_c-1}}}]$ be the binary representation of the symbols of the codeword $\mathbf{x}$ involved in the $i-$th parity check equation. When using the binary representation, the i-th parity check equation of H, can be written as

$$\mathbf{H_i}\mathbf{X_i}^t = \mathbf{0}^t$$

We define $d_{min}(i)$ as the minimum distance of the binary code associated with $\mathbf{H_i}$.

C. Example

Let $p(x) = x^3 + x + 1$ be the primitive polynomial used to generate the elements of $GF(2^3)$. The primitive element for the matrix representation is given by

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

Thus, $\{A^k : k = 0, \dots, 6\}$ are the nonzero elements of $GF(2^3)$ under this matrix representation and it is readily checked for our example that ${A^k}^t{\alpha_l}^t = {\alpha_{k+l}}^t$.

III. Selecting rows for waterfall improvement

In this section, we investigate the choice of "good" rows for the parity check matrix regardless of the structure of the Tanner graph associated with it. First, we briefly review the method proposed in [5], [17] to select the coefficients row by row. We show that the set of rows provided by [17] can easily be reduced and we give an analysis of these coefficient sets using the binary images of the code considered. Then, since the method of [5], as an instance of density evolution, can be computationally expensive for high field orders, we propose a simpler optimization method

based on the binary image of the code associated with a row. By comparing the results of both methods, we observe that the coefficient sets we obtain may encompass the sets given by [17]. We also give some good sets for fields up to GF(256). Finally, we compare the convergence thresholds for row optimized and random code ensembles for different code parameters and field orders.

A. Optimization using Monte-Carlo simulations

In [5][17], the authors propose a method to optimize the rows of the parity check matrix H. They select the coefficients of the matrices carefully using a monte-carlo method: the proposed method starts with a choice of channel model, after which they search for the d_c-tuples that maximize the marginal entropy of the syndrome after a given number of iterations. They obtain a primitive set of d_c-tuples and then the rows of the matrix H are generated randomly from the d_c-tuples multiplied by constants and from their random permutations. For example, Table I summarizes the best d_c-tuples of coefficients for GF(16) and GF(64) with $d_c = 4$. Using the mapping used in [17], the corresponding powers of the primitive element α are given in these tables.

From Table I, we observe that the given sets can be reduced to only one 4-tuple for GF(16) (resp. two 4-tuples for GF(64)) since the other ones are obtained by multiplying by a constant one of these 4-tuples. Thus, re-interpreting the primitive sets given in [17] using the powers of the primitive element allows us to consider some reduced sets for good d_c-tuples (indicated by "$\bullet$" and "$\diamond$" in Table I).

According to Section II-B, using the equivalent binary parity check matrix associated with each d_c-tuple, we can compute the minimum distance $d_{\min}$ associated with it. In Table I, we reported $d_{\min}$ and the weight enumerator coefficient $W(d_{min})$ associated with each 4-tuple. For GF(16), we obtain a $(N = 16, K = 12, d_{\min} = 2)$ code and for GF(64), we obtain a $(N = 24, = 18, d_{\min} = 3)$ code in both cases. As a result, these codes reach or are close to the best possible $d_{\min}$ for their length and dimension [2].

B. Optimization using binary images

In this section, we present an optimization method that aims to select good rows using the equivalent binary parity check matrix. The optimization idea is that the higher d_{min} is, the more

distinguishable, hence reliable, the messages passed from check nodes to data nodes using BP are. Therefore considering the equivalent binary parity check matrix $\mathbf{H_i}$, we intend to maximize $d_{min}(i)$. Thus, the best d_c-tuples candidates are those with the largest d_{min} and among those d_c-tuples with maximum d_{min}, the best are those with the smallest weight enumerator coefficient $W(d_{\min})$.

1) Search procedure: Since finding good d_c-tuples can be computationally expensive, next we provide some guidelines to accelerate the search procedure of the primitive set of rows:

- $d_{min}(i)$ is the minimum number of columns of $\mathbf{H_i}$ that are dependent, thus the minimum distance of a d_c-tuple is at most the minimum distance associated with any two sub-matrices H_{ij} and $H_{ij'}$ of $\mathbf{H_i}$. The minimum distance associated with these two elements is greater or equal to 2. Whenever possible (*i.e.* when we consider a sufficiently high order compared to the d_c-tuple size we try to optimize), we focus on the d_c-tuples having a minimum distance greater or equal to 3.
- Since the rows of H can be some permutations or multiplication by a constant of d_c-tuples of a set, each and every element of this set can be written as an ordered set with the following structure

$$\underbrace{1 \dots \alpha^i \dots \alpha^j \dots \alpha^k}_{d_c},\ 0 < i < \dots < j \dots < k$$

- Based on the previous remarks, the d_c-tuples can be derived from the $(d_c - 1)$-tuples by adding an element α^l such as

$$\underbrace{1 \dots \alpha^i \dots \alpha^j \dots \alpha^k}_{d_c-1} \alpha^l,\ 0 < i < \dots < j \dots < k < l$$

- Once we have determined a set of good d_c-tuples, as seen in the following example, it can be further reduced since some d_c-tuples can be related by a multiplication by a constant.

Note that, for a field of order $q = 2^p$ and a given d_c, the equivalent binary matrix $\mathbf{H_i}$ defines a code of length $N = pd_c$ and dimension $K = p(d_c - 1)$. Using the tables given in [2], we obtain an upper bound on $d_{\min}(i)$.

2) Comparison of both methods: We consider GF(64) with $d_c = 4$. The equivalent binary code has the parameters $N = 24$ and $K = 18$. In this case, $d_{min} \leq 4$ [2]. Table II summarizes the primitive set of rows and their weight spectrum when using the same binary mapping as in [17]. The best codes found have $d_{min} = 3$. When comparing our results to those of [17], the

following observations can be made :

- (i) Both procedures find the rows with associated binary minimum distance $d_{\min} = 3$ and $W(d_{\min}) = 20$ (best rows). The method based on the binary images is also able to list all rows of [17] with $d_{\min} = 3$ and $W(d_{\min}) = 22$ (not given in Table II).
- (ii) The procedure using binary images records some good candidates not detected by [17] with $W(d_{\min}) = 21$.
- (iii) As observed in Section III-A, we can reduce the elementary set to three primitive rows (indicated by"$\bullet$", "$\diamond$" and "$\star$" in Table II).

When we further compare the results provided by both methods, for GF(16), both the method of [5][17] and the proposed method provide the set of rows with the smallest $W(d_{\min})$. For GF(64) and $d_c = 5$, the same result is obtained. Both methods seem as effective to obtain the best candidates. However, the proposed method allows us to optimize the rows for larger field orders, since its computational complexity is less than that of a search based on an instance of density evolution. In Table III, we enumerate the best rows found for some fields up to GF(256) for $d_c = 4$.

C. Minimum distance properties of rows

In Table IV, we compare the maximum binary minimum distance achievable after selecting rows using the binary image with the upper bound given by [2], for different field orders and different values d_c. When considering for example $d_c = 4$, for GF(16), we obtain a $(N = 16, K = 12, d_{\min} = 2)$ code and for GF(64), we obtain a $(N = 24, K = 18, d_{\min} = 3)$ code. From Table IV, it appears that the codes obtained using the binary image selection reach or are close to the best possible $d_{\min}$ for their length and dimension [2] for a wide range of values d_c, especially for low field orders. Moreover, despite the difficulty to reach the upper bound for high field orders, increasing the order allows to keep the minimum distance greater than 2 for an increasing range of values d_c. Note that $d_{\min}$ is not sufficient to select good rows and that the best are those with the smallest weight enumerator coefficient $W(d_{\min})$.

D. Thresholds for row optimized code ensembles

After selecting some potentially good rows, an interesting issue is to predict and analyze the influence of that choice on the convergence behavior for different code parameters and different

field orders. Using density evolution [22], we study the theoretical thresholds for both random and row optimized code ensembles. Note that the latter can be viewed as an expurgated ensemble of the former. In this study, we focus on the row optimized code ensembles generated with only one optimized row for some different field orders.

Figure 1 depicts a threshold comparison between random and optimized row code ensembles for $(2,3)$, $(2,4)$ and $(2,6)$–LDPC codes as a function of the field order. The thresholds have been computed with a Monte Carlo estimation of the density evolution. As the variance of the estimation of the threshold highly depends on the parameters used in the Monte Carlo approach, we have chosen to keep reasonable values for the density evolution parameters (random interleaving of size $N = 10000$, a maximum of 200 iterations), together with a simple variance reduction technique. The variance reduction used is simply a posteriori averaging of the threshold values for independent initializations of the density evolution. As observed in Figure 1, the row optimized code ensemble exhibits a better threshold behavior than the random one for each field order, suggesting that the waterfall region of the error performance curve can be improved by selecting carefully the rows of the parity check matrix. However, the threshold improvement tends to vanish as the field order increases.

IV. Lowering Error floor

In Section III, we have applied local optimizations on H to help the iterative decoding. In this section, we address the problem of the global optimization of H using some local properties of the graph associated with the code in order to design a good code for maximum likelihood decoding.

A. Notations and motivations

Let H_b denote the equivalent binary matrix of H in $GF(2)$. H_b is obtained by replacing all elements in H by their $p \times p$ binary matrix representation as described in Section II. Let $N_b = N.p$ be the binary codeword length. Since the binary minimum distance is defined by the minimum number of independent columns of H_b, it is also strongly related to the topology of the Tanner graph associated with H, noted $\mathcal{G}_H$.

In this section, we show that, as in the binary case, it is likely that the cycles and the stopping sets of $\mathcal{G}_H$ remain the key elements to lower the error floor on the frame error rate (FER)

performance of the code. Indeed, as developed in the following, the cycles and the stopping sets of $\mathcal{G}_H$ describe specific topological structures, that may provide low weight codewords of H. Our definition of a structure in the graph is a set of nodes which forms a closed topological clique, and therefore defines a sub-code of the global code. First, we analyze the equivalent binary representations of both the cycles and the stopping sets of $\mathcal{G}_H$, in order to link their algebraic properties to the "local" binary minimum distance of the LDPC code. Then, we propose an iterative method to improve the error floor using row optimized LDPC code ensembles. Finally, we derive some bounds for the binary minimum distance of $(2, d_c, N)$ non binary LDPC codes. We also give the binary minimum distances for several codes obtained using our optimization method.

For our analysis, we assume the knowledge of the structure of the graph G_H (randomly designed or optimized using instances of the PEG algorithm [13] or other good construction algorithms [23]).

B. Cycle "cancellation"

For a $(2, d_c, N)$ regular code, the binary representation of a cycle of $\mathcal{G}_H$ is always a square matrix. In order to lower the error floor, we would like to avoid cycles that provide low weight codewords. Therefore, if we consider a cycle of length l, this cycle is not involved in the equivalent binary minimum distance if the rank of the equivalent binary matrix associated with the cycle is full (*i.e.*, the cycle does not provide any codeword). Let $g = l_{\min}$ be the girth of $\mathcal{G}_H$. By successively ensuring full rank condition for each binary matrix representation of the cycles of $\mathcal{G}_H$ with lengths $g \leq l \leq l_{\max}$, we can expect to lower the error floor by eliminating low weight codewords of the equivalent binary code.

1) Matrix representation of a cycle: Let $\tilde{C}_d$ be the block matrix representation of a cycle of length l extracted from $\mathcal{G}_H$ with $d = l/2$. Using row and column permutations, $\tilde{C}_d$ can be related to the representation C_d of an elementary cycle given by the following $d \times d$ block square

matrix:

$$C_d = \begin{pmatrix} B_1 & B_2 & 0 & \dots & \dots & 0 \\ 0 & B_3 & B_4 & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & 0 \\ 0 & & & & B_{l-3} & B_{l-2} \\ B_0 & 0 & \dots & \dots & 0 & B_{l-1} \end{pmatrix}$$

where $B_n, n = 0, \dots, l-1$ are the matrix representations of the non-zeros entries α_n of H (see Section II-B) involved in the cycle $\tilde{C}_d$.

2) Full rank condition (FRC): The determinant of $\tilde{C}_d$ is given by

$$\det(\tilde{C}_d) = (-1)^{\pi_i + \pi_j} \det(C_d)$$

where π_i (resp. π_j) is the number of row permutations (resp. column permutations) which are used to transform any cycle into the form of matrix C_d. Hence, $\tilde{C}_d$ is singular if and only if C_d is singular.

Furthermore, the matrix C_d is itself equivalent to the matrix

$$C'_d = \begin{pmatrix} B_1 & B_2 & 0 & \dots & \dots & 0 \\ 0 & B_3 & B_4 & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & 0 \\ 0 & & & & B_{l-3} & B_{l-2} \\ 0 & 0 & \dots & \dots & 0 & D \end{pmatrix}$$

where $D = \prod_i B_{2i+1} + \prod_i B_{2i}$. The equivalence is obtained using Gaussian elimination. Since C'_d is an upper triangular block matrix, the determinant of C_d is equal to the product of the determinants of the block matrices of the diagonal. Since the matrices $B_k, k = 0, \dots, l-1$ are invertible, the full rank condition reduces to $\det(D) \neq 0$. Using finite field addition properties, this is finally equivalent to

$$(\text{FRC}) : \prod_{i=0}^{d-1} B_{2i+1} \neq \prod_{i=0}^{d-1} B_{2i} \tag{2}$$

Remark: A short derivation shows that the FRC condition for the equivalent binary matrix of a given cycle has its equivalent statement at the symbol level in the field as:

$$(\text{FRCS}) : \prod_{i=0}^{d-1} \alpha_{2i+1} \neq \prod_{i=0}^{d-1} \alpha_{2i} \tag{3}$$

where α_i are the non-zero symbols involved in the cycle.

In the following, a cycle is called "cancelled" when the FRC is fulfilled for that cycle.

C. Topological stopping set mitigation

We have seen that the cycles in the Tanner graph of the code can have a mitigated influence if the FRC condition is fulfilled, ensuring that no low weight codewords can be created by a particular cycle. The global performance is however not only dependent on the cycle structure, but also on the stopping sets (inherently present in the structure of $\mathcal{G}_H$) [7][23] that are not reduced to a single cycle.

For a $(2, d_c, N)$-regular code, a stopping set (defined through $\mathcal{G}_H$) that is not reduced to a single cycle is composed of at least 3 imbricated cycles and describes a topological structure of the Tanner graph that we aim to characterize algebraically. Note that the denomination "stopping set" may be abusive in our case. Actually, we are not interested in the property that a stopping set is a fixed point of the BP decoder for the Erasure channel, but rather in the fact that it defines a specific topological structure of dimension immediately larger than a cycle. To this aim, we adopt the topological definition of a stopping set, as proposed in [7].

Let d_s be the number of symbols involved in a given stopping set, that we refer to "symbol weight" of a stopping set in the rest of the paper. For a $(2, d_c, N)$-regular graph, the minimum symbol weight of a stopping set is $d_{s,\min} = \lceil 3g/4 \rceil$, where g is the girth of the graph $\mathcal{G}_H$. For all stopping sets with symbol weights $d_s \geq d_{s,\min}$, the equivalent binary matrix is no longer a square matrix: its binary representation is at most a $(d_s - 1)p \times d_s p$ rectangular matrix H_{ss}. The minimum distance of the code defined by H_{ss} depends on the choice of the coefficients α_i involved in the stopping set. Furthermore, each codeword associated with H_{ss} is a codeword of the global code defined by H_b. Thus, by nature, the code performance is drastically limited by the smallest stopping sets and their associated binary minimum distance.

Unfortunately, unlike for cycles, there is no way to "cancel" the influence of such stopping sets by proper symbol assignments: since each stopping set has a minimum distance associated with it, the only way to ensure a good minimum distance for the whole code is to try to maximize the minimum distance over all stopping sets (practically over the most exhaustive set of stopping sets we can enumerate). It is also important to note that the cycle cancellation for the smallest cycles is an important pre-requirement to avoid "catastrophic" stopping sets. The reason is that

the stopping sets contain cycles, and therefore ensuring cycle cancellation inherently avoids that some columns of the equivalent binary parity matrix in a stopping set add to zero. Note that for a graph $\mathcal{G}_H$ with minimum variable node degree $d_v = 2$, it is quite simple to identify the set of stopping sets with minimum weight $d_{s,\min}$: this can be achieved in conjunction with the PEG construction by adding a procedure which tests if a group of nodes contains 3 imbricated cycles.

Remark: As for cycles, this strategy can be applied at the symbol level, if the performance criterion to track and improve is the minimum symbol distance.

D. Global optimization

In this section, we develop an iterative and row-wise optimization procedure which capitalizes on the properties described in Sections IV-B and IV-C to lower the error floor of the non binary LDPC codes.

The proposed optimization is based on a successive fulfillment of the FRC for all cycles of length l, as l increases while maximizing the "local" minimum distance associated with the stopping sets. Cycle cancellation is done with priority to avoid low weight codewords induced by non cancelled cycles. For optimization purposes, the knowledge of the cycle (resp. stopping set) distributions is assumed for some l (resp. d_s) from g (resp. $d_{s,\min}$) up to a given length $l_{\max}$ (resp. a given weight $d_{s,\max}$). The initial Tanner graph $\mathcal{G}_H$ is first optimized using the PEG algorithm [13]. We have modified the PEG algorithm in order to have both good girth property and stopping set distribution (the number of stopping sets with minimum weight is minimized). Let $\mathcal{R}$ and $\mathcal{S}$ be the set of optimized rows chosen from Section III and the set of the smallest stopping sets (more generally, a union of stopping set ensembles with different low weights), respectively. The following general procedure is applied:

- **Initialization:** The rows in H are chosen at random from the rows in $\mathcal{R}$ and their random permutations.
- **Initial cycle cancellation:** This step intends to cancel successively all cycles with length l, $g \leq l \leq l_M$, included in the stopping sets in $\mathcal{S}$ in order to have well conditioned stopping sets. We define $\mathcal{I}$ as the set of the row indexes of H to be optimized. The optimization is performed iteratively using the following procedure for $l \geq g$:
  1) Initialize $\mathcal{I}$ with all the row indexes of H.

2) Select at random a row index $m \in \mathcal{I}$.
3) Compute $\Pi_n^{(m)}$, a set of n random permutations derived from $\mathcal{R}$.
4) Select the permutation in $\Pi_n^{(m)}$ that maximizes the number of cancelled cycles of length l, conditioned that all shorter cycles are cancelled.
5) $\mathcal{I} \leftarrow \mathcal{I} - \{m\}$. If all length l cycles are cancelled, $l = l + 1$, go to step 1. Otherwise, if $\mathcal{I}$ is empty and there subsist some non-cancelled cycles, go to step 1, else go to step 2.

- **Cycle cancellation and stopping set mitigation:** In this step the successive cancellations are performed for some $l \geq l_M$ following the same procedure as the above procedure except for step 4. The permutation selection is performed based on the maximization of the number of cancelled cycles and the maximization of the minimum distance over all the stopping sets belonging to $\mathcal{S}$ with which the current row m is connected. This maximization is still performed conditioned that all shorter cycles are cancelled.
- **End of optimization:** the optimization procedure is stopped when cycle cancellation is not possible anymore.

Note that, due to computational complexity, we have to restrict the initial set $\mathcal{S}$. It is checked a posteriori that we have a good minimum distance for some stopping sets with higher weights not contained in the set $\mathcal{S}$. Since it is impossible to cancel the cycles for all lengths l, we expect that large non cancelled cycles have less impact on the minimum distance as well as less dramatic influence on the stopping sets in which they are involved, as the size of the associated submatrix increases with the size of the cycles.

E. Achievable binary minimum distance

In this section, we study bounds on the achievable binary minimum distance of non binary $(2, d_c, N)$ regular LDPC codes. We first derive some bounds on the achievable minimum distance when we consider random and PEG based Tanner graphs. Then we compare the bounds we obtained with the minimum distances of the matrices that we have optimized using the previous optimization method.

We suppose that the cycle cancellation has been efficiently done (*i.e.* the cycles are cancelled for sufficiently long lengths), resulting in that no low weight codewords are produced by cycles. In this context, low weight codewords are supposed to be given by stopping sets.

1) Bounding the binary minimum distance: For a given girth g of $\mathcal{G}_H$, the minimum weight of a stopping set is lower bounded by

$$d_{s,\min} = \lceil 3g/4 \rceil \tag{4}$$

This minimum value is achieved if the 3 imbricated cycles have exactly length g. Then, the matrix H_{ss} associated with the stopping sets with weight $d_{s,\min}$ has dimensions at most $(M_{ss} = (d_{s,\min} - 1).p, N_{ss} = d_{s,\min}.p)$. Using the maximum achievable minimum distance given by [2] for a code with the preceding parameters (M_{ss}, N_{ss}), we can obtain an upper bound on the maximum achievable binary minimum distance for that minimal stopping set with weight $d_{s,\min}$. This upper bound becomes an upper bound for the global code associated with H_b if at least one stopping set of $\mathcal{G}_H$ has the minimal size given in (4). We refer this upper bound to the *worst case upper bound*.

2) Minimum distance versus codeword length: Next, we aim to link the minimum distance with the length N_b of the code defined by H_b, still under the assumption that at least one stopping set of $\mathcal{G}_H$ has the minimum size $d_{s,\min}$.

For a (d_v, d_c)-regular Tanner graph with N variable nodes and M check nodes, an upper bound on the girth of the graph as a function of N has been derived in [13, Lemma 3]. Applying this result for the $(2, d_c, N)$ case, we can derive an upper bound on $d_{s,\min}$:

Lemma 1: Let $\mathcal{G}_H$ be a $(2, d_c)$-regular Tanner graph. The minimum stopping set weight $d_{s,\min}$ is upper bounded by

$$d_{s,\min} \leq \min(d_1, d_2) \tag{5}$$

where

$$d_1 = \begin{cases} 3\lfloor t_1 \rfloor + 2 \text{ if } \mathcal{I}_1 = 0 \\ 3\lfloor t_1 \rfloor + 3 \text{ otherwise} \end{cases}$$

$$d_2 = \begin{cases} 3\lfloor t_2 \rfloor + 2 \text{ if } \mathcal{I}_2 = 0 \\ 3\lfloor t_2 \rfloor + 3 \text{ otherwise} \end{cases}$$

in which

$$t_1 = \frac{\log\left((M-1)(1-\frac{2}{d_c})+1\right)}{\log(d_c - 1)} \tag{6}$$

$$t_2 = \frac{\log\left((N-1)(1-\frac{d_c}{2(d_c-1)})+1\right)}{\log(d_c - 1)} \tag{7}$$

and $\mathcal{I}_1$ is equal to 0 if and only if

$$(d_c-1)^{\lfloor t_1 \rfloor} > M-1-\frac{d_c((d_c-1)^{\lfloor t_1 \rfloor}-1)}{d_c-2}$$

and $\mathcal{I}_2$ is equal to 0 if and only if

$$(d_c-1)^{\lfloor t_2 \rfloor} > N-1-\frac{2.(d_c-1)((d_c-1)^{\lfloor t_2 \rfloor}-1)}{d_c-2}$$

Note that for a code of length N, according to this lemma, $d_{s,\min}$ varies in $\mathcal{O}(\log(N))$. Using the upper bound in [2], we are now able to compute numerically an upper bound on the binary minimum distance versus $N_b = p.N$ for a regular code.

However, since the upper bound in [2] for a code with parameters (M_{ss}, N_{ss}) does not provide an analytical expression of $d_{\min}$ as a function of $d_{s,\min}$, we apply the Elias upper bound for a code with parameters $(M_{ss} = (d_{s,\min}-1).p, N_{ss} = d_{s,\min}.p)$ [21]:

$$d_{\min} \leq 2A.(1-A)d_{s,\min}.p \tag{8}$$

with A solution of

$$1/d_{s,\min} = 1 + A\log_2(A) + (1-A)\log_2(1-A),\ 0 \leq A \leq 1/2$$

Reporting (5)-(7) into (8), we can conclude that $d_{\min}$ scales as $\mathcal{O}(\log(N)) = \mathcal{O}(\log(N_b))$. This can be related to a previous result from [9], where it is shown that the minimum distance of the binary $(2, d_c)$-regular LDPC codes can increase *at most* logarithmically with the codeword length N: this emphasizes the need for efficient methods to design codes with good minimum distance properties.

Recall that the expression given by (8) is actually a worst case upper bound, since we assume that there are stopping sets in the Tanner graph $\mathcal{G}_H$ with the minimum size given by (4). As seen for example in Figure 2 for $N_b = 256$, a specific construction based on a modified version of the PEG algorithm can avoid the stopping sets with the minimum size $d_{s,\min}$. In such cases, the worst case upper bound (8) no longer applies and can be therefore exceeded.

3) Binary minimum distance comparison: When considering regular PEG designed codes, in order to derive a bound, we use the best optimized $(2, d_c)$ graphs $\mathcal{G}_H$ obtained with a PEG construction, and we compute the effective achievable minimum stopping set weight. Then, as above, we can derive a bound on the minimum distance achievable under a PEG construction using the minimum weight stopping sets and the upper bound [2].

In Figure 2, we report the binary minimum distance we have computed for some codes optimized using our method ("Opt. codes" curve) and we present some bounds for $(2, 4)$ LDPC codes over $GF(256)$: "Bound-random" is the bound derived from Lemma 1 and upper bound [2], and "UB- Opt codes" is an upper bound derived from the effective $d_{s,\min}$ of optimized codes in [19] and the upper bound [2]. We observe that $d_{\min} = \mathcal{O}(\log(N_b))$. Note that in our PEG constructions, the effective minimal stopping set weight was either $d_{s,\min}$ or $d_{s,\min} + 1$, explaining why the PEG bound can be higher than the random graph upper bound based on minimal stopping set with weight strictly equal to $d_{s,\min}$. Similar results were obtained for other field orders [20].

4) Estimating the equivalent minimum distance: For a given matrix H of a $(2, d_c)$-regular LDPC code, it is possible to compute the stopping set ensembles $\mathcal{S}$ for some $d_s \geq d_{s,\min}$. Using this ensemble $\mathcal{S}$, we can compute the binary minimum distance of the matrix H_{ss} associated with each stopping set in $\mathcal{S}$. The binary minimum distance of H_b is less or equal to the minimum over all the computed minimum distances. This method appears to be a good tradeoff between accuracy and complexity when compared to the impulse method which has been proposed to estimate the minimum distance of LDPC codes [10]. This method allows us to have quickly a good estimate of the minimum distance, especially if the code structure was previously optimized using a cycle cancellation (no codewords provided by short cycles). When the cycle cancellation has not been used previously, the impulse method may be required to find the low weight codewords that the non cancelled cycles may have introduced.

Remark : This approach can be easily extended to the estimation of the symbol minimum distance.

V. Optimization and simulation results

A. Optimization strategies

The different construction methods to be compared are the following:

(i) **Random method (R):** Given a binary matrix, the nonzero entries are randomly selected from the nonzero elements in the fields GF(q).

(ii) **Davey-Mackay method (DM):** Given a binary matrix and a set of good d_c-tuples (previously optimized based on Section III), the rows of H are generated randomly from these d_c-tuples, from these d_c-tuples multiplied by constants and from their random permutations.

(ii) **Binary image method (B):** Given a binary matrix and a unique d_c-tuples (previously optimized based on Section III), the optimization is performed using the successive cycle cancellation and stopping set influence mitigation described in Section IV. Each row of H is generated randomly from the selected d_c-tuple and its random permutations. By highly constraining the matrix construction using only one primitive row, we intend to obtain a good performance-complexity tradeoff as well.

B. Results

All the comparisons are done using the same matrix structure: first the graph $\mathcal{G}_H$ is optimized in order to have good girth and stopping set distribution properties. Then, the values α^i are chosen using one of the previously described methods. For different frame lengths and field orders, we compare the FER assuming a memoryless binary input additive white Gaussian noise (BI-AWGN) transmission channel and an iterative BP decoder over GF(q) at the receiver [1]. The maximum number of iterations is fixed to 1000 to ensure proper convergence. Much less iterations are performed on average with the help of syndrome calculation as stopping criterion. Figure 3 depicts the FER for a $(d_v = 2, d_c = 4)$-LDPC code, different field orders and almost the same bit length. As predicted by the theoretical thresholds, the waterfall gain for B and DM methods over R method vanishes when field order increases. The error floor is reduced in the order of one decade for GF(64) when we compare the methods B and R/DM, showing the effect of providing attention to the cycle and stopping set configurations. For GF(256), the three methods have almost the same performance up to FER$= 10^{-5}$. Figure 4 depicts the FER for a $(2, 4)$-LDPC code over GF(16) and a $(2, 8)$-LDPC code over GF(64). For the (2,4) code, we observe that the selection of some good rows is very relevant for the small field orders. For high rate codes, the waterfall gain decreases with the rate for a given field order, but the optimization can improve drastically the error floor.

More generally, simulation results underline that the row optimization (DM method) is not sufficient to ensure good performance in the error floor region. Hence, the matching of the matrix coefficients through cycle cancellation and stopping set mitigation (B method) appears as mandatory to address this problem.

C. Comparison with existing codes

In this section, we compare our performance results with some state of the art optimized binary codes with small codeword lengths for the BI-AWGN channel. In Figure 5, we compare the performance of our codes with that of [14] and with optimized irregular LDPC code whose irregularity is taken from [22] (the PEG algorithm is used to build the parity check matrix in order to avoid short cycles) for $R = 1/2$ and $K = 1024$ information bits. We observe that the error floor region is high for the irregular LDPC codes and that we have a gain of about 0.6 dB compared to the code of [14] in the waterfall region.

Since the optimized codes from [14] are designed for rather low coding rates, we further compare our results with that of [8], whose codes are designed for high rates codes. With our codes, the waterfall region is improved of about 0.25 dB, and slightly more for $R = 4/5$ as shown in Figure 6. This gain has to be balanced by the higher decoding complexity of nonbinary codes. Note that the codes presented in [8] have to our knowledge the best available performance for binary codes presented in the literature.

VI. Conclusion

In this paper, we have addressed the problem of the design of non binary $(2, d_c, N)$ regular LDPC codes. Using the binary image of the code, we characterized the algebraic properties of rows, cycles and stopping sets with respect to a local or global minimum distance. Then, we proposed a method for both waterfall and error floor improvements based on these algebraic properties. The results show that the optimization of local topological structures (rows, cycles and stopping sets) is important to design codes with both good waterfall and error floor properties.

Acknowledgment

The authors wish to thank the authors of [8] for providing the comparative results.

References

[1] L. Barnault and D. Declercq, “Fast Decoding Algorithm for LDPC over GF(2^q)”, *IEEE Inf. Th. Workshop*, Paris, France, March 2003.

[2] A.E. Brouwer and T. Verhoeff, “An Updated Table of Minimum-Distance for Binary Linear Codes”, IEEE Trans.on Inf. Theory, Vol. 39, No. 2, pp. 662–677, March 1993.

[3] S.-Y. Chung, T. J. Richardson, R.L. Urbanke, "Analysis of Sum-Product Decoding of Low-Density-Parity-Check Codes using a Gaussian Approximation", *IEEE Trans. on Inf. Theory*, Vol. 47, pp. 657–670, February 2001.

[4] M.C. Davey, *Error-Correction using Low-Density Parity-Check Codes*, PhD thesis, University of Cambridge, 1999.

[5] M.C. Davey and D. MacKay, "Low-Density Parity-Check Codes over GF(q)", *IEEE Commun. letters*, Vol.2, pp. 165–167, June 1998.

[6] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes over GF(q)", submitted to IEEE Trans. on Commmunications.

[7] C. Di, D. Proietti, I.E. Telatar, T.J. Richardson, R.L. Urbanke, "Finite-length Analysis of Low-Density Parity-Check Codes on the Binary Erasure Channel", *IEEE Trans. on Inf. Theory*, Vol. 48, No. 6, pp. 1570 – 1579, June 2002.

[8] D. Divsalar, C. Jones, S. Dolinar and J. Thorpe, "Protograph Based LDPC Codes with Minimum Distance Linearly Growing with Block Size", *in Proc. IEEE GLOBECOM conference*, Vol. 3, pp. 1152–1156, St Louis, MO, USA, November 2005.

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

[10] X.Y. Hu and M. Fossorier, "On the Computation of the Minimum Distance of Low-Density Parity-Check Codes," *IEEE Int. Conf. on Commun.*, Paris, France, June 2004.

[11] X.Y. Hu and E. Eleftheriou, "Binary Representation of Cycle Tanner-Graph GF(2^q) codes, *IEEE Int. Conf. on Commun.*, Paris, France, June 2004.

[12] X.Y. Hu, M. Fossorier and E. Eleftheriou, "Approximate Algorithms for Computating the Minimum Distance of Low-Density Parity-Check Codes", *IEEE Int. Symp. on Inf. Theory*, Chicago, USA, June 2004.

[13] X.-Y. Hu, E. Eleftheriou and D.M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs", *IEEE Trans. on Inf. Theory*, Vol. 51, No. 1, pp. 386–398, January 2005.

[14] G. Liva, W.E. Ryan, M. Chiani, "Design of quasi-cyclic Tanner codes with low error floors", *in Proc. Int. Turbo Code Symposium*, Munich, Germany, April 2006.

[15] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman, "Improved Low Density Parity Check Codes using Irregular Graphs", *IEEE Trans. On Inf. Theory*, Vol. 47, No. 2, pp. 585–598, Feb. 2001.

[16] D. J. C. Mackay, "Good Error Correcting Codes based on Very Sparse Matrices", *IEEE Trans. On Info. Theory*, Vol. 45, No. 2, pp 399–431, 1999.

[17] D. MacKay, "Optimizing Sparse Graph Codes over GF(q)", available at http://www.inference.phy.cam.ac.uk/mackay/CodesGallager.html, August 2003.

[18] F.J. Mac Williams and N. J. A. Sloane, *The theory of Error-Correcting Codes*, North Holland, 1978.

[19] C. Poulliat, M. Fossorier, D. Declercq, "Optimization of non binary LDPC codes using their binary images", *in Proc. Int. Turbo Code Symposium*, Munich, Germany, April 2006.

[20] C. Poulliat, M. Fossorier, D. Declercq, "Design of non binary LDPC codes using their binary images: algebraic properties", *in Proc. IEEE Int. Symp. on Information Theory*, Seattle, WA, USA, July 2006.

[21] J.G. Proakis, *Digital Communications*, McGraw-Hill, 3rd Ed., 1995.

[22] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of Capacity Approching Irregular Low-Density Parity-Check Codes", *IEEE Trans. on Inf. Theory*, Vol. 47, pp.657–670, Feb. 2001.

[23] T. Tian, C.R. Jones, J.D. Villasenor, R.D. Wesel, "Selective Avoidance of Cycles in Irregular LDPC Code Construction", *IEEE Trans. on Commun.*, Vol. 52, No. 8, pp. 1242 – 1247, Aug. 2004.

GF	Row coefficients				d_{min}	$W(d_{min})$
16	11	7	3	0	2	1 $\bullet$
	11	7	4	0	2	1 $\bullet$
	11	8	4	0	2	1 $\bullet$
	12	8	4	0	2	1 $\bullet$
64	48	35	26	0	3	20 $\bullet$
	28	54	13	0	3	20 $\bullet$
	55	28	13	0	3	22 $\diamond$
	27	48	35	0	3	22 $\diamond$
	21	36	8	0	3	22 $\diamond$
	22	37	9	0	3	20 $\bullet$
	50	15	41	0	3	20 $\bullet$
	42	50	15	0	3	22 $\diamond$

TABLE I

ROW COEFFICIENTS FOR GF(16) AND GF(64) AND $d_c = 4$ FROM [17].

Row coefficients				d_{min}	$W(d_{min})$
37	22	9	0	3	20 $\bullet$
54	28	13	0	3	20 $\bullet$
50	41	15	0	3	20 $\bullet$
48	35	26	0	3	20 $\bullet$
44	18	7	0	3	21 $\diamond$
37	19	9	0	3	21 $\star$
54	28	10	0	3	21 $\star$
56	37	11	0	3	21 $\diamond$
53	44	18	0	3	21 $\star$
37	26	19	0	3	21 $\diamond$
45	35	26	0	3	21 $\star$
52	45	26	0	3	21 $\diamond$

TABLE II

ROW COEFFICIENTS FOR GF(64) AND $d_c = 4$ USING BINARY IMAGES.

GF	Row coefficients				d_{min}	$W(d_{min})$
16	11	7	3	0	2	1
32	15	10	5	0	3	38
	20	15	5	0	3	38
	24	15	5	0	3	38
	24	15	6	0	3	38
	21	14	7	0	3	38
	23	15	7	0	3	38
	22	14	7	0	3	38
64	37	22	9	0	3	20
	44	18	7	0	3	21
	37	19	9	0	3	21
128	93	37	18	0	3	5
	94	38	19	0	3	5
	106	75	19	0	3	5
	108	74	18	0	3	6
	93	38	19	0	3	6
	95	38	19	0	3	6
	107	75	19	0	3	6
256	183	172	8	0	4	156
	183	173	8	0	4	159
	182	172	8	0	4	160
	88	80	8	0	4	161
	89	81	9	0	4	161
	167	127	40	0	4	161
	182	173	8	0	4	162
	169	127	40	0	4	162
	169	128	40	0	4	162

TABLE III

ROW COEFFICIENTS FOR DIFFERENT FIELD ORDERS AND $d_c = 4$.

		d_c									
		3	4	5	6	7	8	9	10	11	12
GF	16	3 (3)	2 (2)								
	32	3 (4)	3 (3)			2 (2)					
	64	4 (4)	3 (4)	3 (3)						2 (2)	
	128	4 (4)	3 (4)							3 (3)	
	256	4 (4)		3 (4)							

TABLE IV

MINIMUM DISTANCE ACHIEVABLE FOR ROWS USING THE BINARY IMAGE. THE UPPER BOUND FROM [2] IS REPORTED IN BRACKETS (.).

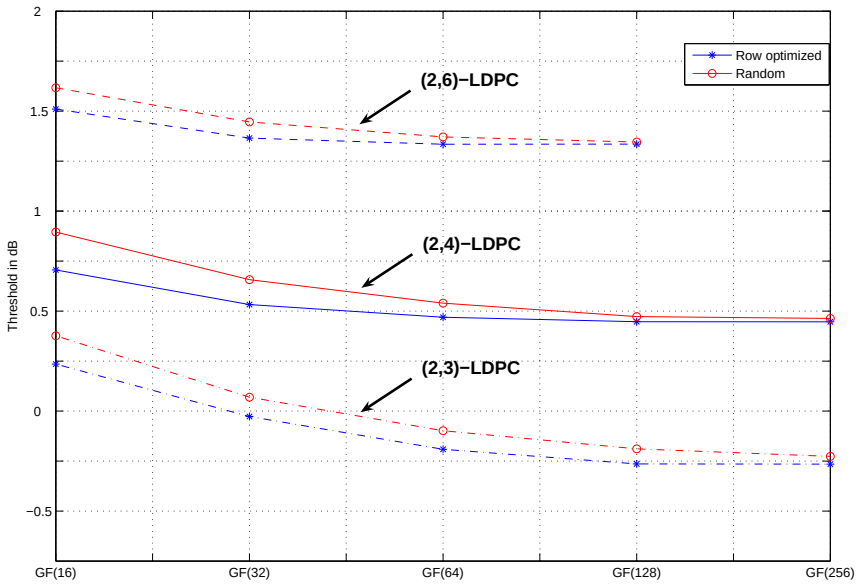

Fig. 1. Theoretical thresholds comparison between random and optimized row code ensembles for $(2,3)$, $(2,4)$ and $(2,6)$-LDPC codes as a function of the field order. The thresholds are computed through density evolution using Monte-Carlo simulations with $N_s = 10000$ symbols per codeword.

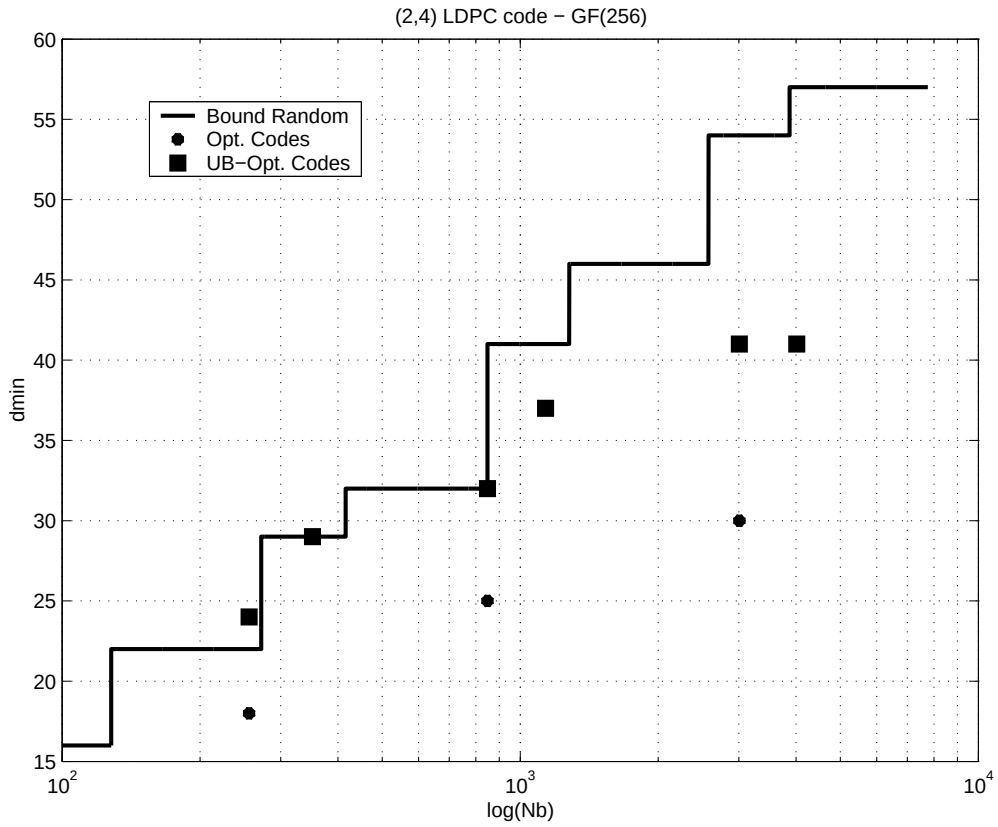

Fig. 2. Bounds on the achievable minimum distance for a $(2, 4)$ non binary LDPC codes over $GF(256)$.

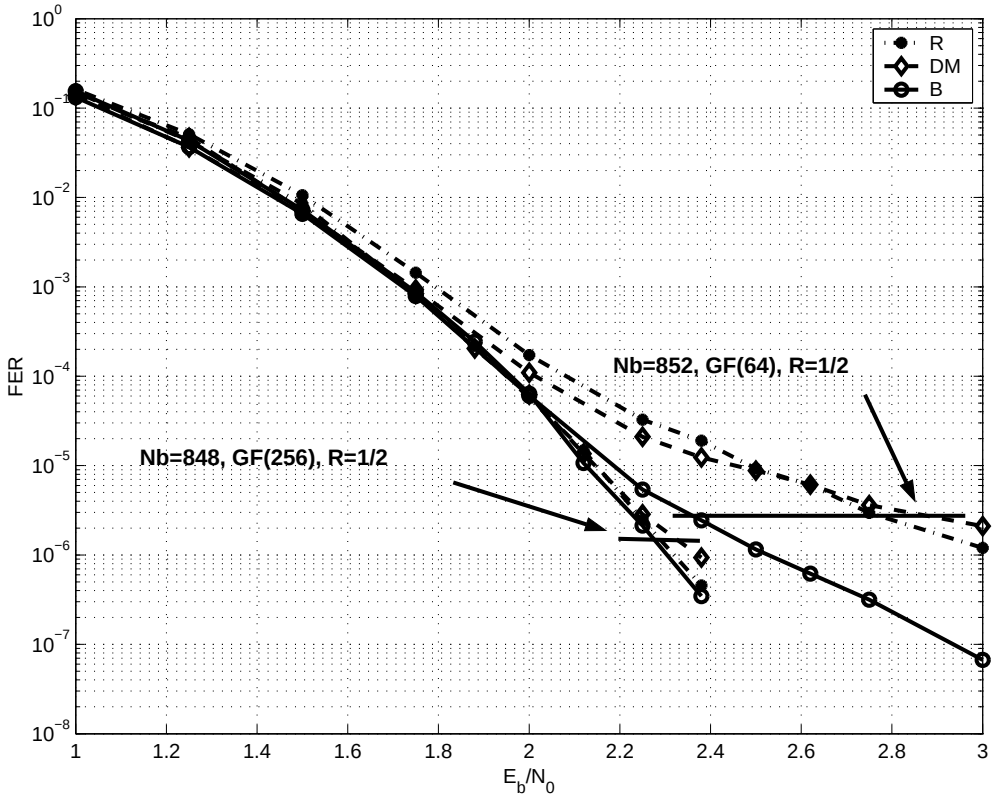

Fig. 3. FER versus E_b/N_0: $GF = \{64, 256\}$, $N_b = \{852, 848\}$ bits, $R = 1/2$.

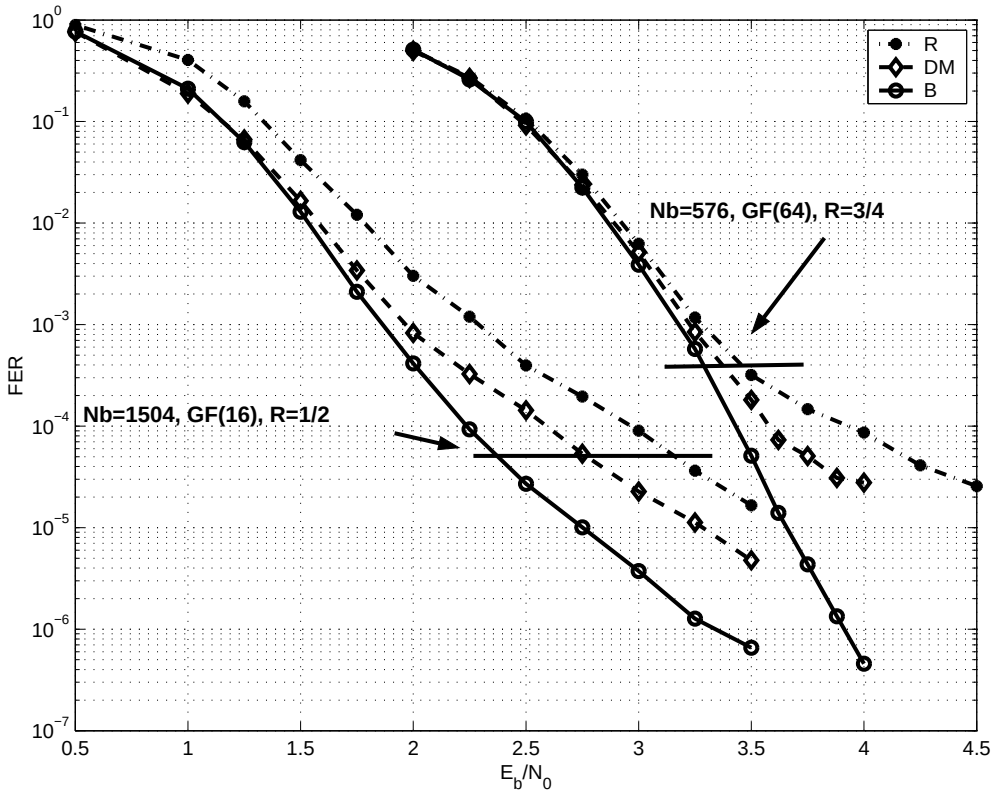

Fig. 4. FER versus E_b/N_0: (i) $GF(16)$, $N_b = 1504$ and $R = 1/2$, (ii) $GF(64)$, $Nb = 576$ and $R = 3/4$.

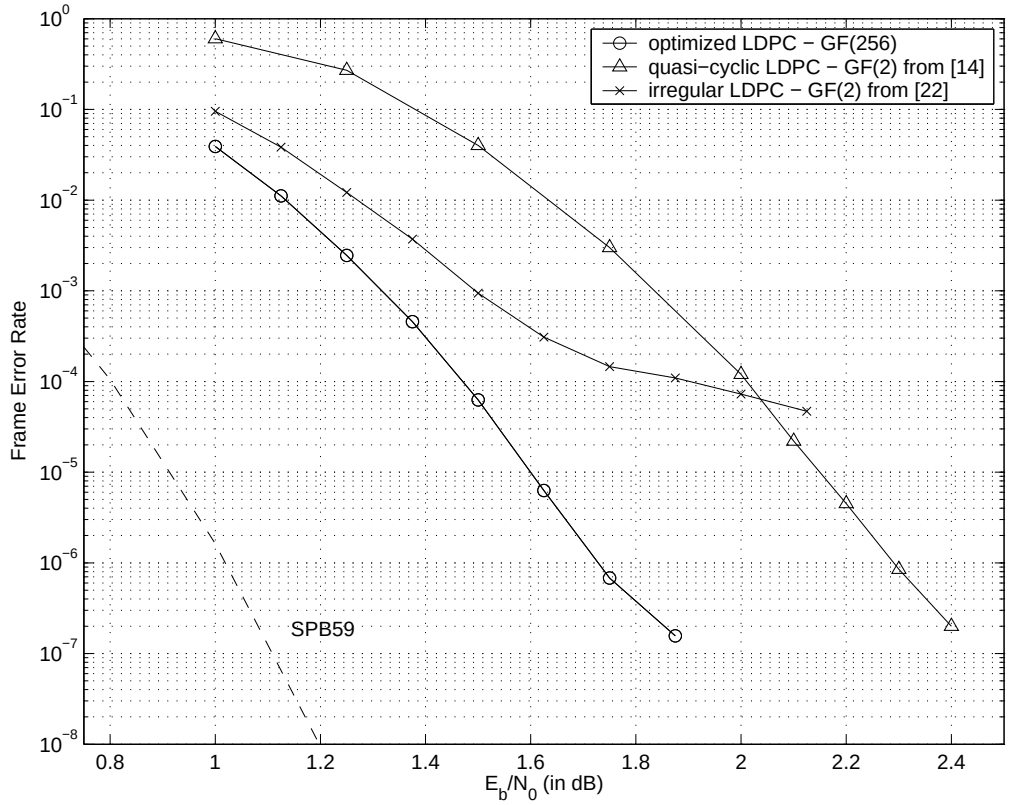

Fig. 5. FER versus E_b/N_0 : comparison with Quasi-cyclic codes from [14] and irregular codes from [22] for $R = 1/2$ and $K = 1024$ information bits. Non binary codes are designed over GF(256).

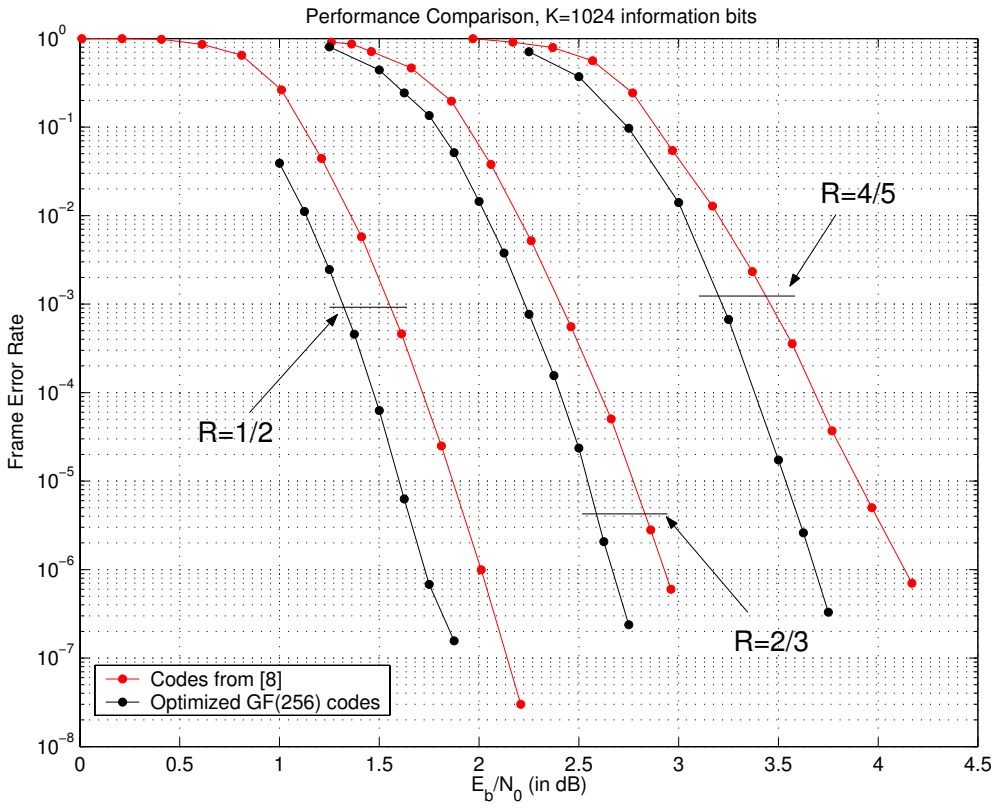

Fig. 6. FER versus E_b/N_0 : comparison with [8]. Non binary codes are designed over GF(256). $K = 1024$ information bits