Analysis of Doubly-Generalized LDPC Codes with Random Component Codes for the Binary Erasure Channel

Enrico Paolini, Marc Fossorier and Marco Chiani

Abstract—A method for the asymptotic analysis of doubly-generalized low-density parity-check (D-GLDPC) codes on the binary erasure channel (BEC) is described. The proposed method is based on extrinsic information transfer (EXIT) charts. It permits to overcome the impossibility to evaluate the EXIT function for the check or variable component codes, in situations where the information functions or split information functions for the component code are unknown. According to the proposed method, D-GLDPC codes where the check and variable component codes are random codes from an expurgated ensemble, are considered. A technique is then developed which permits to obtain the EXIT chart for the overall D-GLDPC code, by evaluating the expected EXIT function for each check and variable component code. This technique is then combined with differential evolution (DE) algorithm in order to generate some optimal D-GLDPC degree distributions. Numerical results on long, random codes, are presented which reveal how D-GLDPC codes can outperform standard LDPC codes in terms of both waterfall performance and error floor.

I. INTRODUCTION

Low-density parity-check (LDPC) codes [1] have been shown to exhibit excellent asymptotic performance over a wide range of channels, under iterative decoding [2] [3]. It has been proved that irregular LDPC codes are able to asymptotically achieve the binary erasure channel (BEC) capacity for any code rate. This means that, for any code rate $R$ and for any small $\epsilon > 0$, it is possible to design an edge degree distribution $(\lambda, \rho)$ corresponding to a code rate $R$, whose threshold is $p^* = (1-\epsilon)(1-R)$ [4]. Examples of capacity achieving (sequences of) degree distributions are the heavy-tail poisson sequence [4] and the binomial sequence [5].

It is well known that this very good asymptotic performance in terms of decoding threshold does not necessarily correspond to a satisfying finite length performance. In fact, finite length LDPC codes with good asymptotic threshold, though characterized by good waterfall performance, are usually affected by high error floors [6] [7]. This phenomenon has been partly explained in [8], where it is proved that, with high probability, the minimum distance of a random LDPC code, with threshold $p^*$ on the BEC close to $1-R$, is a logarithmic function of the codeword length $N$. When considering transmission on the BEC, low weight codewords create small stopping sets [9], resulting in high error floors.

The (so far not overcome) inability in generating LDPC codes, with threshold close to capacity and good minimum distance properties as well, is one of the main reasons for investigating more powerful (and complex) coding schemes. Examples are generalized LDPC (GLDPC) codes and doubly-generalized LDPC (D-GLDPC) codes. In GLDPC codes, generic linear block codes different from single parity-check (SPC) codes can be used as check nodes as well. First introduced in [10], GLDPC codes have been more recently investigated, for instance, in [11]–[17]. Recently introduced in [18], D-GLDPC codes represent a wider class of codes than GLDPC codes. The generalization consists in using generic linear block codes, not only repetition codes, as variable nodes. Linear block codes used as check or variable nodes will be called component codes of the D-GLDPC code.

The threshold analysis of random GLDPC codes and random D-GLDPC codes can be in principle performed through extrinsic information transfer (EXIT) charts [19] [20]. The success of this approach is bound to the knowledge of the EXIT function for each check and variable component code. In [21, Theorem 2] it is proved that, if the communication channel is a BEC, then the EXIT function of a check component code can be related to the code information functions [22], while the EXIT function for a variable component code to the split information functions. This relationship between EXIT function and (split) information functions is very useful for the threshold analysis on the BEC of D-GLDPC codes built up with component codes whose (split) information functions are known. However, for a wide range of linear codes, including most binary double error-correcting and more powerful BCH codes, these parameters are still unknown.

In [16], a possible solution to overcome this problem has been proposed for GLDPC codes. The proposed method consists in considering random check component codes belonging to a certain expurgated ensemble, instead of specific check component codes (like Hamming or BCH codes). A technique was then developed in order to exactly evaluate the expected information functions for each check component code over the considered expurgated ensemble. This permits to evaluate the expected EXIT function for each check component code, supposing transmission on the BEC, and the expected EXIT function for the overall check node decoder. In this paper, this approach is extended to D-GLDPC codes. More specifically, novel formulas are proposed, which permit to exactly evaluate the expected split information functions.

This work was supported in part by the NSF under Grant CCR-04-30576 and in part by the European Commission under project FP6 IST-001812 “PHOENIX”.

E. Paolini and M. Chiani are with DEIS, WiLAB, University of Bologna, via Venezia 52, 47023 Cesena (FC), Italy. Email: {epaolini,mchiani}@deis.unibo.it.

M. Fossorier is with the Department of Electrical Engineering, University of Hawaii at Manoa, 2540 Dole street, Honolulu, HI 96822, USA. Email: marc@spectra.eng.hawaii.edu

Forty-Fourth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
Sept 27-29, 2006
for a random \((n, k)\) linear block code, used as variable node of a D-GLDPC code.

The capability to perform the EXIT chart analysis on the BEC for D-GLDPC codes, possible by combining the results proposed in this paper and those from [16], is then exploited in order to design capacity approaching D-GLDPC distributions. Simulation results obtained on random, long codes, reveal that capacity approaching D-GLDPC codes can be characterized by better threshold and lower error floor than capacity approaching LDPC and GLDPC codes. Moreover, by imposing constraints on the fraction of edges towards the generalized check nodes, the error floor of D-GLDPC codes can be further lowered (at the cost of increased complexity), while preserving good or very good waterfall performance.

II. GLDPC CODES AND D-GLDPC CODES

A traditional LDPC code of length \(N\) and dimension \(K\) is usually graphically represented through a bipartite graph, known as Tanner graph [10], characterized by \(N\) variable nodes and a number \(M \geq N - K\) of check nodes. Each edge in the graph can only connect a variable node to a check node. According to this representation, the variable nodes have a one-to-one correspondence with the encoded bits of the codeword, and each check node represents a parity-check equation involving a certain number of encoded bits. The degree of a node is defined as the number of edges connected to the node. Thus, the degree of a variable node is the number of parity constraints the corresponding encoded bit is involved in, and the degree of a check node is the number of bits involved in the corresponding parity-check equation.

A degree-\(n\) check node of a standard LDPC code can be interpreted as a length-\(n\) SPC code, i.e. as a \((n, n - 1)\) linear block code. Analogously, a degree-\(n\) variable node can be interpreted as a length-\(n\) repetition code, i.e. as a \((n, 1)\) linear block code. A first step towards the generalization of LDPC codes consists in letting some of (or eventually all) the check nodes, be generic \((n, k)\) linear block codes: The corresponding code structure is known as GLDPC code. An \((n, k)\) generalized check node is connected to \(n\) variable nodes. For GLDPC codes, the number of parity check equations is no longer equal to the number of check nodes, since a \((n, k)\) check node corresponds to \(n - k\) parity-check equations. If there are \(N_C\) check nodes, then the number of parity-check equations is \(M = \sum_{i=1}^{N_C} (n_i - k_i)\) (\(n_i\) and \(k_i\) are the length and the dimension of the \(i\)-th check component code). The generalized check nodes are characterized by higher error / erasure correction capability than SPC codes. Moreover, their introduction can be favorable from the point of view of the code minimum distance. However, the generalized check nodes usually produce a code rate loss which makes GLDPC codes with uniform check node structure (composed for instance only of Hamming nodes) quite poor in terms of decoding threshold [14], despite being very good in terms of minimum distance [11].

The second step of the generalization consists in introducing variable nodes different from repetition codes. The corresponding code structure is known as D-GLDPC code [18], and is represented in Fig. 1. An \((n, k)\) generalized variable node is connected to \(n\) check nodes, and receives its \(k\) information bits from the communication channel. Thus, \(k\) of the \(N\) encoded bits for the overall D-GLDPC code are received by the \((n, k)\) generalized variable node, and interpreted by the variable node as its \(k\) information bits. In a D-GLDPC code, the number of parity check equations is still given by \(M = \sum_{i=1}^{N_C} (n_i - k_i)\). Moreover, if \(N_V\) is the number of variable nodes, the codeword length is \(N = \sum_{i=1}^{N_V} k_i\) (\(k_i\) is the dimension of the \(i\)-th variable component code). In the next section, another valid interpretation of the variable nodes, as \((n + k, k)\) codes, will be presented.

The iterative decoding algorithm for D-GLDPC codes on the BEC is a generalization of the iterative decoder for LDPC codes presented in [4]. At each iteration, an \((n, k)\) check node receives \(n\) messages from the variable nodes: Some of them are erasure messages, some others are known messages. Maximum a posteriori (MAP) decoding is performed at the check node in order to correct the unknown encoded bits. After MAP decoding, a known message is sent towards the variable nodes for each known encoded bit, along the corresponding edge, while an erasure message is sent for each encoded bit which is still unknown.

The operation of an \((n, k)\) variable node is analogous, with the only difference that, at each iteration, some of the information bits can be known as well as some of the encoded bits. In order to exploit the partial knowledge of the information bits, MAP decoding is performed considering an extended generator matrix \([G \mid I_k]\), where \(G\) is the chosen \((k \times n)\) generator matrix of the code, and \(I_k\) is the \((k \times k)\) identity matrix. After MAP decoding, the known messages towards the check nodes correspond to the known encoded bits, while the erasure messages correspond to the encoded bits remaining unknown.

III. EXIT FUNCTIONS FOR GENERALIZED VARIABLE AND CHECK NODES ON THE BEC

Consider the general decoding model proposed in [21, Fig. 3], and depicted in part in Fig. 2. The encoder of a linear block code with dimension \(k\) is split into two encoders, namely encoder 1 and encoder 2, generating a codeword \((v, x)\), with length \(|v| + |x|\): The encoded bits \(x\) generated by
the encoder 1 are transmitted on a communication channel, while the encoded bits $v$ generated by the encoder 2 are transmitted on an extrinsic channel. Both the received vector $y$ from the communication channel and the received vector $w$ from the extrinsic channel are exploited by the a posteriori probability (APP) decoder in order to compute the log-likelihood ratios for the encoded bits, and the extrinsic log-likelihood ratios. We refer to the above model in order to describe and analyze each check or variable node of a D-GLDPC code. For a D-GLDPC code, the communication channel is the channel over which the D-GLDPC code encoded bits are transmitted. Moreover, the extrinsic channel represents a model for the channel over which the messages exchanged between variable and check nodes are transmitted, during the iterative decoding process. If the communication channel is a BEC, then the extrinsic channel can also be modelled as a BEC.

For the check nodes of a D-GLDPC code, no communication channel is present. Moreover, any check node representing an $(n, k)$ linear block code interfaces to the extrinsic channel through its $n$ encoded bits. Then, the encoder 1 is not present, while the encoder 2 performs a linear mapping $v = Gu$, where $G$ is one of the several possible generator matrix representations for the $(n, k)$ linear block code. It follows that $|v| = n$. This model is the same proposed in [21, Sec. VII-A]. It includes the check nodes of a standard LDPC code (for which all the check nodes are SPC codes).

According to the concept of D-GLDPC code, the generic variable node, representing an $(n, k)$ linear block code, receives its $k$ information bits from the communication channel, and interfaces to the extrinsic channel through its $n$ encoded bits. For this reason, for the variable nodes of a D-GLDPC code, the encoder 1 is represented by the identity mapping $x = u$ and the encoder 2 performs a linear mapping $v = Gu$. In this case it results $|x| = k$ and $|v| = n$. The $(n, k)$ variable node is indeed modelled as a $(n + k, k)$ code whose generator matrix is in the form $[G \mid I_k]$. This model includes the variable nodes of a standard LDPC code (i.e. repetition codes). An important difference between variable nodes represented by repetition codes and generalized $(n, k)$ variable nodes, with $k > 1$, is that in the latter case different representations of the generator matrix are possible. These different code representations correspond to a different performance of the overall D-GLDPC code. The code representation for the generalized variable nodes is then a degree of freedom for the code design. On the contrary, the code performance is independent of the code representation for the generalized check nodes. This degree of freedom is then not present in GLDPC codes. In Section IV, this point will be better explained.

The EXIT function of the linear code in Fig. 2, with dimension $k$, length $n + k$ and split encoder, without idle components, assuming that the communication channel is a BEC with erasure probability $q$ and that the extrinsic channel is a BEC with erasure probability $p$, has been shown in [21, eq. 36] to be expressed by

$$I_E(p, q) = 1 - \frac{1}{n} \sum_{t=0}^{n-1} p^t (1 - p)^{n-t-1} \sum_{z=0}^{k} q^z (1 - q)^{k-z} \cdot \left[ (n - t) \tilde{e}_{n-t, k-z} - (t + 1) \tilde{e}_{n-t-1, k-z} \right],$$

(2)

which can be easily obtained from (1) by performing the substitutions $t = n - q$ and $z = k - h$. Expressions (1) and (2) are valid under the hypothesis that MAP erasure decoding is performed at the variable node. If applied to a $(n, 1)$ repetition code, (2) leads to $I_E(p, q) = 1 - q p^{n-1}$, i.e. to the well known expression of the EXIT function for a degree-$n$ variable node of an LDPC code on the BEC.

The EXIT function of a generic $(n, k)$ check node of a D-GLDPC code on the BEC can be obtained by letting $q = 1$ in (2) (no communication channel is present). The obtained expression, equivalent to [21, eq. 40], is

$$I_E(p) = 1 - \frac{1}{n} \sum_{t=0}^{n-1} p^t (1 - p)^{n-t-1} \cdot \left[ (n - t) \tilde{e}_{n-t} - (t + 1) \tilde{e}_{n-t-1} \right],$$

(3)

where, for $g = 0, \ldots, n$, $\tilde{e}_g$ is the $g$-th (un-normalized) information function. It is defined as the summation of the dimensions of all the possible codes obtained considering
just \( g \) positions in the code block \( v \) of length \( n \) [22]. As (2), (3) assumes that erasures are corrected at the check node according to MAP decoding\(^1\). If applied to a \((n, n-1)\) SPC code, (3) leads to the expression of the EXIT function for a degree-\( n \) check node of an LDPC code, i.e. \( I_E(p) = (1-p)^{n-1} \). Since the code book is independent of the choice of the generator matrix, different code representations have the same information functions. Thus, different code representations have the same EXIT function for the generalized check node. The performance of the overall D-GLDPC code is then independent of the specific representation of the generalized check nodes.

IV. RANDOM CODE HYPOTHESIS AND EXPURGED ENSEMBLE DEFINITION

Consider a D-GLDPC code with \( \mathcal{I}_V \) different types of variable component codes and \( \mathcal{I}_C \) different types of check component codes. The \( i \)-th variable component code has EXIT function on the BEC \( I_E^{(i)}(p, q) \), corresponding to a specific code representation, and is assumed without idle components. Analogously, the \( i \)-th check component code has EXIT function \( I_C^{(i)}(p) \), and is assumed without idle components. Variable and check nodes are supposed randomly connected through an edge interleaver. The fraction of edges towards the variable nodes of type \( i \) is denoted by \( \lambda_i \), and the fraction of edges towards the check nodes of type \( i \) by \( \rho_i \). Then, the total EXIT function for the variable nodes decoder and for the check nodes decoder are given by respectively

\[
I_{E,V}(p, q) = \sum_{i=1}^{\mathcal{I}_V} \lambda_i I_{E,V}^{(i)}(p, q) \tag{4}
\]

\[
I_{E,C}(p) = \sum_{i=1}^{\mathcal{I}_C} \rho_i I_{E,C}^{(i)}(p). \tag{5}
\]

These relationships can be obtained by reasoning in the same way as [21] for the EXIT functions of the variable and check nodes decoders of an irregular LDPC code.

Next, we introduce the random code hypothesis. Each variable or check component code is assumed a random linear block code. More specifically, the generator matrix of each \((n_i, k_i)\) variable component code of type \( i (i = 1, \ldots, \mathcal{I}_V) \) is assumed uniformly chosen at random from an expurgated ensemble denoted by \( G^{(n_i, k_i)} \). The generator matrix of each \((n_i, k_i)\) check component code of type \( i (i = 1, \ldots, \mathcal{I}_C) \) is assumed uniformly chosen at random from the same expurgated ensemble. Let \( \mathbb{E}[I_{E,V}(p, q)] \) and \( \mathbb{E}[I_{E,C}(p)] \) be, respectively, the expected variable and check EXIT function for such D-GLDPC code ensemble. Then, from (4) and (5) we have

\[
\mathbb{E}[I_{E,V}(p, q)] = \mathbb{E} \left[ \sum_{i=1}^{\mathcal{I}_V} \lambda_i I_{E,V}^{(i)}(p, q) \right] = \sum_{i=1}^{\mathcal{I}_V} \lambda_i \mathbb{E}[I_{E,V}^{(i)}(p, q)] \tag{6}
\]

\[
\mathbb{E}[I_{E,C}(p)] = \mathbb{E} \left[ \sum_{i=1}^{\mathcal{I}_C} \rho_i I_{E,C}^{(i)}(p) \right] = \sum_{i=1}^{\mathcal{I}_C} \rho_i \mathbb{E}[I_{E,C}^{(i)}(p)]. \tag{7}
\]

If the \( i \)-th variable code type is a repetition code, or the \( i \)-th check code type is a SPC code, no expectation is needed.

The reason for an expurgated ensemble of \((n_i \times k_i)\) generator matrices is needed, instead of the ensemble of all the possible \((k_i \times n_i)\) binary matrices, is to ensure a correct application of the EXIT charts analysis, as explained next.

The aim is to perform a threshold analysis of the D-GLDPC codes ensemble through an EXIT chart approach, using the expected EXIT functions expressed in (6) and (7). In order to correctly apply the analysis based on EXIT charts, the following conditions are required:

\[
\lim_{p \rightarrow 0} \mathbb{E}[I_{E,V}(p, q)] = 1 \quad \forall q \in (0, 1) \tag{8}
\]

\[
\lim_{p \rightarrow 0} \mathbb{E}[I_{E,C}(p)] = 1 \tag{9}
\]

\[
\lim_{p \rightarrow 1} \mathbb{E}[I_{E,C}(p)] = 0. \tag{10}
\]

Conditions (8) and (9) guarantee that, for both the variable and the check nodes decoder, the average extrinsic information outcoming from the decoder is equal to 1 when the extrinsic channel (Fig. 2) is noiseless. Condition (10) guarantees that the average extrinsic information from the check nodes decoder is 0 when the extrinsic channel is the useless channel.

Since the conditions \( \sum_{i=1}^{\mathcal{I}_V} \lambda_i = 1 \) and \( \sum_{i=1}^{\mathcal{I}_C} \rho_i = 1 \) must be satisfied, it follows from (6) and (7) that (8) and (9) are satisfied if these relationships hold:

\[
\lim_{p \rightarrow 0} \mathbb{E}[I_{E,V}^{(i)}(p, q)] = 1 \quad \forall q \in (0, 1), \forall i \tag{11}
\]

\[
\lim_{p \rightarrow 0} \mathbb{E}[I_{E,C}^{(i)}(p)] = 1 \quad \forall i. \tag{12}
\]

Analogously, (10) is satisfied if

\[
\lim_{p \rightarrow 1} \mathbb{E}[I_{E,C}^{(i)}(p)] = 0 \quad \forall i. \tag{13}
\]

In order to have (11), (12) and (13) satisfied, it is sufficient that analogue relationships hold for the EXIT function of all
the codes represented by the generator matrices in \( G_{n,k}^* \). This permits to obtain a definition of \( G_{n,k}^* \).

Consider first a check component code. From (3) it follows \( \lim_{p \to 0} I_E(p) = 1 - (\tilde{e}_n - \frac{e_{n-1}}{n}) \). Then, the desired property \( \lim_{p \to 0} I_E(p) = 1 \) is guaranteed by the equality \( n \tilde{e}_n = \tilde{e}_{n-1} \). This equality is always satisfied when the generator matrix \( G \) is full-rank (rank \( G = k \)), and when the \( k \times (n-1) \) matrix obtained by removing any column from \( G \) is full rank. In fact, in this case both sides of the previous equality are equal to \( kn \). We define as independent column any column of a rank-\( r \) \( (m \times n) \) matrix with the following property: Removing that column from the matrix leads to a \( m \times (n-1) \) matrix with rank \( r-1 \). An independent column is linearly independent of all the other columns of the matrix. By reasoning in the same way, it is readily proved from (3) that \( \lim_{p \to 0} I_E(p) = 1 - \tilde{e}_1/n \). Then, \( \lim_{p \to 0} I_E(p) = 0 \) when \( \tilde{e}_1 = n \), i.e. when the generator matrix has no zero columns. This is equivalent to assume that the component code has no idle components, an hypothesis already implicitly considered in (3). Then, (9) and (10) are satisfied if the generator matrix of any check component code is full rank, has no independent columns, and has no zero columns. These conditions, already developed in [16], are next extended to the variable component codes.

Consider a variable component code. From (2):

\[
\lim_{p \to 0} I_E(p, q) = 1 - \sum_{z=0}^{k} q^z (1-q)^{k-z} (\tilde{e}_{n, k-z} - \frac{\tilde{e}_{n-1, k-z}}{n}).
\]

If \( n \tilde{e}_{n, h} = \tilde{e}_{n-1, h} \) for \( h = 0, \ldots, k \), then \( \lim_{p \to 0} I_E(p, q) = 1 \) for any \( q \). This is always true when the \( (k \times n) \) generator matrix is full rank (rank \( G = k \)) and has no independent columns. In fact, in this case \( n \tilde{e}_{n, h} = \tilde{e}_{n-1, h} = \binom{k}{n} n k \). The constraint that the generator matrix has no zero columns must be also considered, since it is a key hypothesis for the validity of (1). Then, (8) is satisfied if the generator matrix of any variable component code is full rank, has no independent columns, and has no zero columns.

Hence, the set of constraints to be imposed on the generator matrix of linear block codes used as check and variable nodes is the same, as assumed at the beginning of this section. We can then define \( G_{n,k}^* \) as the ensemble of all the rank-\( k \) \( (k \times n) \) binary matrices, without independent columns and without zero columns. It follows from (2) and (3) that the problem of evaluating the expected EXIT function on \( G_{n,k}^* \), for each \( (n, k) \) variable (check) node, can be completely solved by evaluating the expected split information functions (information functions) on \( G_{n,k}^* \). In the next section, a technique for solving this problem is proposed.

V. EXPECTED SPLIT INFORMATION FUNCTION COMPUTATION

A technique for the evaluation of the expected information functions \( \tilde{e}_g \) for an \( (n, k) \) linear block code, with generator matrix randomly chosen from \( G_{n,k}^* \), has been presented in [16]. Specifically, it has been proved that

\[
\mathbb{E}_{G_{n,k}^*} [\tilde{e}_g] = \sum_{g=0}^{k} \frac{\text{rank}(S_{g,h})}{\text{rank}(G)}.
\]

The function \( J(m, n, r) \) denotes the number of rank-\( r \) \( (m \times n) \) binary matrices without zero columns and without independent columns. Thus, \( J(k, n, k) \) is the total number of matrices in \( G_{n,k}^* \). Moreover, \( E_{G_{n,k}^*} [\tilde{e}_g] \) is the submatrix of \( G_{n,k}^* \) composed of the last \( g \) columns and \( k \) rows, and such that the rank of the first \( g \) columns is equal to \( u \). This approach is next extended to the split information functions \( \tilde{e}_g \).

Let \( G \) be a binary matrix from \( G_{n,k}^* \), and let \( S_{g,h} \) be a submatrix of \( [G | I_k] \) obtained by selecting \( g \) columns in \( G \) and \( h \) columns in \( I_k \). Then:

\[
\mathbb{E}_{G_{n,k}^*} [\tilde{e}_{g,h}] = \mathbb{E}_{G_{n,k}^*} \left[ \sum_{h=1}^{k} \frac{\text{rank}(S_{g,h})}{\text{rank}(G)} \right]
\]

\[
= \sum_{h=1}^{k} \mathbb{E}_{G_{n,k}^*} \left[ \text{rank}(S_{g,h}) \right] \frac{\text{rank}(G)}{\text{rank}(G)}
\]

\[
= \binom{n}{g} \binom{k}{h} \mathbb{E}_{G_{n,k}^*} \left[ \text{rank}(S_{g,h}) \right].
\]

The last equality is due to the fact that, for the previously defined expurgated ensemble, the expectation of the rank when selecting \( g \) columns in \( G \) and \( h \) columns in \( I_k \) is independent of the specific selected columns. Thus, \( S_{g,h} \) in (15) can be in principle any such submatrix. We suppose that \( S_{g,h} \) in (15) is the submatrix composed of the last \( g \) columns of \( G \), and the first \( h \) columns of \( I_k \) (see Fig. 3).

The probability \( \text{Pr} \{ \text{rank}(S_{g,h}) = u \} \) that, for a randomly chosen matrix \( G \) in \( G_{n,k}^* \), the submatrix \( S_{g,h} \) has rank \( u \), can be expressed as the number of matrices in \( G_{n,k}^* \) for which this property holds divided by the total number of matrices in \( G_{n,k}^* \). It is clear from Fig. 3 that the rank of \( S_{g,h} \) is at least \( h \). In fact, the last \( h \) columns of this submatrix, i.e. the first \( h \) columns of \( I_k \), are linearly independent. Moreover, in order to have \( \text{rank}(S_{g,h}) = u \), it is necessary and sufficient that the \( (k-h) \times g \) submatrix \( \Gamma \) in Fig. 3 has rank \( u-h \). Hence, the binary matrices \( G \in G_{n,k}^* \) satisfying \( \text{rank}(S_{g,h}) = u \) are those ones for which the submatrix intersection of the first \( g \) columns and the first \( k-h \) rows has rank \( u-h \).
The expectation of rank($S_{g,h}$) in (15) can then be further developed as:
\[
E_{g^{(n,k)}}[\text{rank}(S_{g,h})] = \sum_{u=h}^{\min(k,g+h)} u \Pr \{\text{rank}(S_{g,h}) = u\} = \sum_{u=h}^{\min(k,g+h)} \frac{\tilde{K}(k, n, g, k-h, u-h, k)}{J(k, n, k)}.
\] (16)

The function $\tilde{K}(m, n, a, b, t, r)$ represents the number of rank-$r$ $(m \times n)$ binary matrices without zero columns, without independent columns, and such that the submatrix intersection of the first $a$ columns and first $b$ rows has rank $t$.

In the following, recursive formulas for $J(\cdot)$ and $K(\cdot)$ are recalled from [16], and a novel formula for $\tilde{K}(\cdot)$ is derived. In principle, $J(\cdot)$ can be expressed in terms of both $K(\cdot)$ and $\tilde{K}(\cdot)$, as $J(m, n, r) = \sum_{u=1}^{\min(\{g, s\})} K(m, n, g, u, r)$ and $J(m, n, r) = \sum_{t=0}^{\min(a, b)} \tilde{K}(m, n, a, b, t, r)$. However, an independent formula for $J(\cdot)$ is presented.

A. Computation of $J(m, n, r)$ and $K(m, n, g, u, r)$

A technique for the computation of $J(\cdot)$ and $K(\cdot)$ has been developed in [16], and the following results have been proved.

**Lemma 1**: Let $F(m, n, r)$ be the number of rank-$r$ $(m \times n)$ binary matrices without zero columns. Then
\[
F(m, n, r) = \sum_{j=0}^{r-1} \binom{n}{j} \left( \prod_{i=0}^{j-1} (2^m - 2^i) \right) \frac{2^{2(r-j)}}{(2^m - 2^j)} - \sum_{z=1}^{n-r} \binom{n}{z} F(m, n, z, r).
\] (17)

**Theorem 1**: The function $J(\cdot)$ can be recursively evaluated according to
\[
J(m, n, r) = F(m, n, r) - \sum_{j=1}^{r-1} \binom{n}{j} \left( \prod_{i=0}^{j-1} (2^m - 2^i) \right) 2^{2(r-j)} \cdot J(m-j, n-j, r-j).
\] (18)

In (17) and (18), it must be imposed that $F(m, n, 1) = J(m, n, 1) = 2^m - 1$, and $F(m, n, r) = 0$ and $J(m, n, r) = 0$ when at least one of the following conditions is true: $m \leq 0$, $n \leq 0$, $u \leq 0$, $r \leq 0$, $g \leq 0$, $g > n$, $u > \min\{m, g\}$, $r > \min\{m, n\}$, $r - u > \min\{m - g, m - u\}$, $u > r$,$(g > 0, u = 0)$,$(g = 0, u > 0)$,$(g = n, u \neq r)$,$(m = n, r = m, g \neq u)$. Special cases are: $M(m, n, g, u, r) = M(m, n, r)$ if $(g = 0, u = 0)$ or $(g = n, u = r)$. $M(m, n, g, u, r) = M(m, n, g, m)T(m, n - g)$ if $u = m$, $M(m, n, g, u, r) = F(m, g, m)(2^r - 1)^{n-g}$ if $\{u = r, n > g\}$.

**Theorem 2**: The function $K(\cdot)$ can be recursively evaluated according to
\[
K(m, n, g, u, r) = M(m, n, g, u, r) \\
- \sum_{j=1}^{r-1} \sum_{l=0}^{\min(m, u, j-1)} \binom{g}{j} \left( \prod_{i=0}^{j-1} (2^m - 2^i) \right) 2^{j(r-j)} \cdot K(m-j, n-j, g-l, u-l, r-j).
\] (20)

The function $K(\cdot)$ is set to 0 in the same cases as $M(\cdot)$. Special cases are: $K(m, n, g, u, r) = J(m, n, r)$ if $(g = 0, u = 0)$ or $(g = n, u = r)$, $K(m, n, g, u, r) = 2^n - 1$ if $\{u = 1, r = 1, g > 0, n - g > 0\}$, $K(m, n, g, u, r) = F(m, m)$ if $\{m = n, g = u\}$.

B. Computation of $\tilde{K}(m, n, a, b, t, r)$

Let $\tilde{K}(m, n, a, b, t, u, r)$ be the number of rank-$r$ $(m \times n)$ binary matrices without zero columns, without independent columns, such that the submatrix intersection of the first $a$ columns and $b$ rows has rank $t$, and such that the submatrix composed of the first $a$ columns has rank $u$. The function $\tilde{K}(\cdot)$ can be expressed in terms of $K(\cdot)$, as
\[
\tilde{K}(m, n, a, b, t, u, r) = \sum_{u=1}^{\min(m, a)} \hat{K}(m, n, a, b, t, u, r).
\] (21)

The technique for the evaluation of $\tilde{K}(\cdot)$ is based on a recursive formula developed for $\hat{K}(\cdot)$. For some $(m, n, a, b, t, u, r)$, $\hat{K}(m, n, a, b, t, u, r)$ is first evaluated for $u = 1, \ldots, \min(m, a)$, then $\hat{K}(m, n, a, b, t, r)$ is computed according to (21). The proofs of the results presented in the following are only sketched. More details can be found in [23].

**Lemma 3**: Let $\tilde{N}(m, n, p, t, r)$ be the number of rank-$r$ $(m \times n)$ binary matrices, such that the rank of the first $p$ rows is $t$. Then
\[
\tilde{N}(m, n, p, t, r) = 2^{(m-p)} \cdot \prod_{j=0}^{t-1} \frac{(2^p - 2^j)(2^n - 2^j)}{2^{2^j} - 2^j} \cdot \prod_{j=0}^{r-t-1} \frac{(2^m - 2^j)(2^{n-t} - 2^j)}{2^{2^j} - 2^j}.
\] (22)

**Proof**: Let $M$ be the $(m \times n)$ binary matrix. It can be partitioned in a matrix $M_1$, composed of the first $p$ rows of $M$ and a matrix $M_2$, composed of the last $m - p$ rows. The total number of $M_1$ matrices is $\prod_{j=0}^{t-1} (2^p - 2^j)(2^n - 2^j)/(2^{2^j} - 2^j)$. For any choice of $M_1$, the number of $M_2$ matrices is $2^{(m-p)} \prod_{j=0}^{r-t-1} (2^m - 2^j)(2^{n-t} - 2^j)/(2^{2^j} - 2^j)$. ■
The function $\tilde{N}(\cdot)$ is set to 0 if at least one of the following conditions is true: $m \leq 0$, $n \leq 0$, $p < 0$, $r < 0$, $t < 0$, $r > t$, $p > m$, $\{p = 0, t > 0\}$, $\{p = m, t \neq r\}$, $r > \min\{m, n\}$, $t > \min\{p, n\}$. Particular conditions are: $\tilde{N}(m, n, p, 0, r) = \prod_{j=0}^{n-1}(2^{m-2j})(2^{n-2j})(2^{u-2j})$, $\tilde{N}(m, n, r, r) = \prod_{j=0}^{n-1}(2^{m-2j})(2^{n-2j})(2^{u-2j})$.

**Lemma 4:** Let $\tilde{F}(m, n, p, t, r)$ be the number of rank-$r$ $(m \times n)$ binary matrices, such that the rank of the first $p$ rows is $t$, and without zero columns. Then

$$\tilde{F}(m, n, p, t, r) = \tilde{N}(m, n, p, t, r)$$

$$- \sum_{z=1}^{n-r} \binom{n}{z} \tilde{F}(m, n-z, p, t, r). \quad (23)$$

**Proof:** $\tilde{F}(m, n, p, t, r)$ is equal to the total number of rank-$r$ binary matrices such that the rank of the first $p$ rows is $t$, minus the number of such matrices with $z$ zero columns, for $z = 1, \ldots, n-r$. □

The function $\tilde{F}$ is set to 0 in the same cases as $\tilde{N}$, or when $r = 0$. Special conditions are: $\tilde{F}(m, n, p, t, r) = F(m, n, p, r)$ if $t = 0$, and $\tilde{F}(m, n, p, t, r) = F(m, n, r)$ if $\{p = m, t = r\}$.

**Lemma 5:** Let $\tilde{M}(m, n, a, b, t, u, r)$ be the number of rank-$r$ $(m \times n)$ binary matrices without zero columns, such that the submatrix intersection of the first $a$ columns and the first $b$ rows has rank $t$, and such that the submatrix composed of the first $a$ columns has rank $u$. Then

$$\tilde{M}(m, n, a, b, t, u, r) = \tilde{F}(m, a, b, t, u)$$

$$- \sum_{z=0}^{(n-a)-(r-u)} \binom{n-a}{z} F(m-u, n-a-z, r-u)$$

$$\cdot T(u, z) 2^{m(n-a-z)}. \quad (24)$$

**Proof:** Let $M$ be the $(m \times n)$ matrix, and $M_1$ be the submatrix composed of its first $a$ columns. The number of $M_1$ matrices is $F(m, a, b, t, u)$, expressed by Lemma 4. Let $M_2$ be the submatrix of $M$ composed of the last $n-a$ columns. The number of $M_2$ submatrices is independent of the specific choice of $M_1$. A convenient choice of $M_1$ is depicted in Fig. 4, where $M_2$ is partitioned into three submatrices $M_2^{(1)}, M_2^{(2)}, M_2^{(3)}$, and the matrix $A$ is defined. In order to have rank$(M) = r$, it must be rank$(A) = r-u$. Denoting by $z$ the number of zero columns in $A$, the number of $A$ matrices is $\sum_{z=0}^{\max(n-a-z, r-u)(n-a-z, r-u)} F(m-u, n-a-z, r-u)$. Since we need at least $r-u$ non-zero columns for $A$, it must be $z_{\max} = (n-a) - (r-u)$. Since $M$ must have no zero columns, the total number of choices for the $z$ columns of $M_2^{(2)}$, corresponding to the zero columns of some choice of $A$, is $T(u, z)$. Moreover, no constraint exists on the choice of the columns of $M_2^{(3)}$ corresponding to the non-zero columns of $A$. Then, this number is $2^{m(n-a-z)}$. □

The function $\tilde{M}(\cdot)$ is set to 0 if at least one of the following conditions is true: $m \leq 0$, $n \leq 0$, $a < 0$, $b < 0$, $r \leq 0$, $u < 0$, $t < 0$, $t > r$, $t > u$, $u > r$, $a > n$, $b > m$, $r > \min\{m, n\}$, $t > \min\{a, b\}$, $u > \min\{m, a\}$, $\{a = 0, t > 0\}$, $\{b = 0, t > 0\}$, $\{a = 0, u > 0\}$, $\{a = n, u \neq r\}$, $\{b = m, t \neq u\}$, $\{a = n, b = m, t \neq r\}$. Special cases are: $\tilde{M}(m, n, a, b, t, u, r) = F(m, n, r)$ if $\{a = 0, t = 0, u = 0\}$ or $\{a = n, b = m, t = u = r\}$, $\tilde{M}(m, n, a, b, t, u, r) = F(m, n, t, r)$ if $\{a = n, u < r, b < m\}$, $\tilde{M}(m, n, a, b, t, u, r) = F(m, n, a, u, r)$ if $\{b = m, t = u, a < n\}$, $\tilde{M}(m, n, a, b, t, u, r) = F(m, a, b, t, u)T(u, n-a)$ if $\{u = r\}$.

**Lemma 6:** Let $G(m, j, l, b, \gamma, \delta)$ be the number of rank-$j$ $(m \times j)$ binary matrices (necessarily without zero columns), such that the submatrix intersection of the first $l$ columns and the first $b$ rows has has rank $\gamma$, and such that the submatrix composed of the first $b$ rows has rank $\delta$. Then

$$G(m, j, l, b, \gamma, \delta) = \tilde{F}(m, b, \gamma, l)$$

$$\cdot \mathcal{F}(m-l, j-l, b-\gamma, \delta-\gamma, j-l) \cdot 2^{l(j-l)}. \quad (25)$$

**Proof:** Since the rank of the $(m \times j)$ matrix is equal to $j$, all the columns must be linearly independent. Hence, the number of possible choices for the first $l$ columns is $F(m, l, b, \gamma, l)$, with $F(\cdot)$ defined in Lemma 4. The number of possible choices for the last $j-l$ columns is independent of the specific choice of the first $l$ columns. A convenient choice is depicted in Fig. 5, where the $((m-l) \times (j-l))$ submatrix $A$ is defined. In order to have a rank-$j$ $(m \times j)$ matrix, it must be rank$(A) = j-l$. Moreover, in order to have a rank $\delta$ for the first $b$ rows, it must be rank$(M^{(1)}) = \delta-\gamma$. Since all the $j$ columns must be independent, $A$ must have no zero columns. Then, the number of $A$ matrices is $F(m-l, j-l, b-\gamma, \delta-\gamma, j-l)$. Since any choice is allowed for $M^{(2)}$, the number of such submatrices is $2^{l(j-l)}$. □
\[ \hat{K}(m, n, a, b, t, u, r) = \hat{M}(m, n, a, b, t, u, r) - \sum_{j=1}^{r-1} \sum_{l=0}^{u} (a_l)(n-a_j-l) \gamma \sum_{\gamma=0}^{\gamma_{\text{max}}} G(m, j, l, b, \gamma, \delta) 2^{(r-j)(j-(\delta-\gamma))} \cdot \sum_{p=0}^{p_{\text{max}}} \hat{K}(m-j, n-j, a-l, b-\delta, p, u-l, r-j). \]

\[ \cdot \sum_{p=0}^{p_{\text{max}}} \hat{K}(m-j, n-j, a-l, b-\delta, p, u-l, r-j). \]

\[ \prod_{i=0}^{q-1} (2^{(r-j)-2^{i}((r-j)-(u-l-p))}((\delta-\gamma))) . \]

(26)

![Comparison between the EXIT function of a (16,8) variable node, with generator matrix randomly chosen from G_{16}^{16,8} (solid), and the expected EXIT function over G_{16,8} (dotted), for several values of q.](image)

**VI. Numerical Results**

In this section, some numerical results are presented about the asymptotic and finite length performance of \( R = 1/2 \), capacity approaching LDPC, GLDPC and D-GLDPC codes. More specifically, the following case study is considered.

For the LDPC coding scheme, we allow the degree of the variable nodes (repetition codes) to range between 2 and 30, and the degree of the check nodes between 3 and 14. For the GLDPC coding scheme, the check nodes are composed of a mixture of SPC codes, with the same degree range as for the LDPC case, and (31, 21) BCH codes. The (31, 21) BCH code is chosen instead of a random (31, 21) code, since for these values of \( k \) and \( n \) a direct computation of the information functions is still practical. Furthermore, it has been shown in [16] that using check component codes with good minimum distance properties, within GLDPC codes with hybrid check nodes, can be convenient in terms of threshold. For the D-GLDPC coding scheme, the same check nodes types as for the GLDPC case are considered; in addition, hybrid variable nodes are allowed, composed by a mixture of repetition codes with the same degrees as for the LDPC code, plus (31,10) random linear block codes. The choice of (31, 10) codes as variable nodes was dictated by the guideline to use codes with same length and dual dimension at different sides of the bipartite graph. However, there is no proof so far that this is the optimal choice. The expected EXIT function for the (31,10) variable nodes on the BEC was evaluated according to the method presented in Section V.

The differential evolution (DE) algorithm [24] was combined with EXIT chart analysis in order to design irregular distributions, satisfying the above described constraints, and characterized by optimal thresholds on the BEC. Examples of such (edge-oriented) distributions are given in Table I, as well as the corresponding thresholds. The D-GLDPC distribution is denoted by D-GLDPC1. Some considerations about these distributions are provided next.

First, we observe that the GDLPC and D-GLDPC distributions have a better threshold than the LDPC distribution, and that the D-GLDPC distribution has the best threshold. Hence, under the described constraints, using generalized variable nodes permits to increase the threshold with respect to GLDPC codes. This result is even more meaningful, since the LDPC and GLDPC thresholds are already close to capacity. Second, the better threshold of GLDPC and D-GLDPC distributions is achieved with a relatively small fraction of generalized check and variable nodes. In fact,
the fraction of BCH check nodes for the GLDPC code is about 2.70%, while the fraction of BCH check nodes and (31, 10) variable nodes for the D-GLDPC code are about 4.11% and 0.48% respectively. Indeed, it results a small increase in terms of decoding complexity with respect to the LDPC code. The larger fraction of BCH check nodes in the D-GLDPC distribution than in the GLDPC one (4.11% vs 2.70%) suggests the following third consideration. The original idea behind GLDPC codes was to strengthen the check nodes, by introducing powerful generalized check nodes [10]. This approach can provide good minimum distance properties, but the drawback is a lowering of the overall code rate, which reveals unacceptable in many cases [14] [16]. The key idea behind D-GLDPC codes is to overcome this rate loss by introducing weaker component codes at the variable nodes. In the case study under analysis, the introduction of (31, 10) generalized variable nodes is able to partly compensate the rate loss due to the (31, 21) BCH check nodes; then, it is possible to use a larger number of powerful erasure correcting codes among the check nodes, without loosing performance in terms of threshold.

In order to verify these asymptotic performance results, we simulated long and randomly constructed codes, designed according to the distributions presented in Table I. For the D-GLDPC coding scheme, the dual of the (31, 21) BCH code (in systematic representation) was used at the generalized variable nodes. In Fig. 7, the performance in terms of post-decoding bit error rate (BER) is shown for codes of length $N = 128000$. As expected, the LDPC code exhibits bad error floor performance, due to the poor minimum distance of capacity approaching distributions (for this distribution it is $\lambda'(0)/\rho'(1) = 2.015$). This high error floor is not improved when considering the GLDPC code construction. However, the D-GLDPC code exhibits both a slightly better waterfall performance (according to the slightly better threshold) and an error floor which is about one order of magnitude lower than that of LDPC and GLDPC codes. This result suggests that capacity approaching D-GLDPC codes can be constructed, characterized by better properties in terms of both waterfall and error floor performance than LDPC and GLDPC codes, and with limited increase of decoding complexity.

Using generalized variable nodes enables to use a larger number of generalized (powerful) check nodes than for GLDPC codes, suggesting better minimum distance properties, while keeping a better threshold. In order to construct D-GLDPC codes with better minimum distance properties than the D-GLDPC$_1$ code, and still a good threshold, we tried the following approach. We ran the DE algorithm again for the D-GLDPC$_2$ distribution, with the additional constraint of a lower bound on the fraction of edges towards the generalized check nodes. More specifically, we ran the DE optimization with the further constraint $\rho_{BCH}^{\min} \geq \rho_{BCH}^{\min}$. The obtained distribution for $\rho_{BCH}^{\min} = 0.16$ and the corresponding threshold are presented in Table II, denoted by D-GLDPC$_2$ code. The threshold is still better than that of the LDPC distribution.
The performance curve on the BEC obtained for a random $N = 128000$ code, designed according to the D-GLDPC$_2$ distribution, is shown in Fig. 7. We observe an improvement in the error floor region, about one order of magnitude with respect to the D-GLDPC$_1$ code, and about two orders of magnitude with respect to the GLDPC and LDPC codes, with no loss in terms of waterfall performance.

VII. CONCLUSION

In this paper, a technique for the asymptotic analysis of D-GLDPC codes on the BEC has been proposed. This technique assumes that the variable and check component codes are random codes. It permits to evaluate the expected EXIT function for the variable and check node decoders, thus enabling for the EXIT chart analysis. The core of this method is the computation of the expected (split) information functions over an expurgated ensemble of $(n, k)$ linear block codes. The expurgation guarantees a correct application of the EXIT charts analysis. The expected (split) information function computation exploits some formulas for obtaining the exact number of binary matrices with specific properties. The proposed analysis method has been combined with the differential evolution algorithm, in order to develop optimal D-GLDPC distributions. Focusing on random, capacity approaching codes, it has been shown that D-GLDPC codes can be constructed, outperforming LDPC and GLDPC codes in terms of both waterfall and error floor. Moreover, by lower bounding the fraction of edges towards the generalized check nodes, D-GLDPC codes have been designed, which gain two orders of magnitude with respect to the LDPC and GLDPC codes in terms of error floor, with no sacrifice in terms of waterfall performance.

ACKNOWLEDGEMENT

The authors wish to thank Yige Wang for her feedback on this work.

REFERENCES


