Non-Binary Protograph-Based LDPC Codes: Enumerators, Analysis, and Designs

Lara Dolecek, Dariush Divsalar, Yizeng Sun, and Behzad Amiri

***Abstract*—This paper provides a comprehensive analysis of non-binary low-density parity check (LDPC) codes built out of protographs. We consider both random and constrained edge-weight labeling, and refer to the former as the unconstrained non-binary protograph-based LDPC codes (U-NBPB codes) and the latter as the constrained non-binary protograph-based LDPC codes (C-NBPB codes). Equipped with combinatorial definitions extended to the non-binary domain, ensemble enumerators of codewords, trapping sets, stopping sets, and pseudocodewords are calculated. The exact enumerators are presented in the finite-length regime, and the corresponding growth rates are calculated in the asymptotic regime. We then present an EXIT chat tool for computing the iterative decoding thresholds of protograph-based LDPC codes followed by several examples of finite-length U-NBPB and C-NBPB codes with high performance. Throughout the paper, we provide accompanying examples which demonstrate the advantage of non-binary protograph-based LDPC codes over their binary counterparts and over random constructions. The results presented in this paper advance the analytical toolbox of non-binary graph-based codes.**

I. Introduction

Graph-based codes can be divided into two categories: codes where each coded symbol is represented by a single bit (binary codes), and codes where each coded symbol is represented by ℓ, $\ell > 1$ bits (non-binary codes). In his seminal work on low-density parity check (LDPC) codes [21], Gallager studied both binary and non-binary codes. As the graph-based codes resurrected in late 1990s, Davey and MacKay [13] empirically recognized that non-binary low density parity check (LDPC) codes can outperform binary LDPC codes in certain cases. However, a considerable amount of subsequent research effort was devoted to studying binary LPDC codes and their decoders. As pointed out in [23], non-binary LDPC codes can outperform binary LDPC codes on binary channels and can be seamlessly merged with high-order modulation techniques for multiple input, multiple output channels. With these early promising results and the emergence of a range of applications that use non-binary coding schemes, such as optical communication channels [17] and dense data storage [20], the interest in non-binary LDPC codes is being actively renewed.

Generalizing from the binary to the non-binary domain of LDPC codes is often non-trivial, and sometimes even surprising. For example, [39] showed that non-binary LDPC codes with variable degree set to 2 perform quite well, while it is well-known that in the binary setting such codes have rather poor performance. Recent results in [3], [4], [26], and [27] have made an important progress in the non-binary domain in terms of characterizing codeword and pseudocodeword weight distributions of certain regular non-binary codes, both in terms of binary weights and symbol weights.

The works of [11], [25], [46], and [53] have put forth finite-length designs of non-binary LDPC codes with outstanding performance. In a parallel thread, works such as [10] and [49], among others, have explored various aspects of decoding of non-binary LDPC codes. Codes with efficient encoding and decoding were proposed in [52]. The error-floor performance of non-binary LDPC codes was recently investigated in [36] and [38], and non-codeword objects that cause the decoding error under iterative decoding were studied in [37] and [41]. Minimum distance properties of non-binary LDPC codes were recently explored in [29].

Despite the on-going surge of interest in non-binary LDPC codes, many questions regarding *structured* codes remain to be answered. Notable recent results on this topic include the analysis and design of so-called cluster-based LDPC codes, for which bounds on the minimum distance and asymptotic thresholds were derived in [15] and [43]. Hybrid LDPC codes that built upon both binary and non-binary constructions were recently proposed in [42].

In this work we focus our attention on the characterization of non-binary LDPC codes built out of protographs. In particular, we consider novel non-binary code constructions that are based on repeating the nodes and permuting the edges as in the binary case [2], but that are also equipped with the added freedom of choosing the non-binary edge weights (i.e., edge scaling). We refer to resultant codes as *non-binary protograph-based (NBPB) codes*. One can generalize the construction of binary protograph-based LDPC codes by replacing the copy-permute operations with either copy-scale-permute operations [1] or with scale-copy-permute operations. We consider both approaches. The former construction is less restrictive: copies of an edge can receive different non-zero

L. Dolecek, Y. Sun and B. Amiri are with the Department of Electrical Engineering, University of California, Los Angeles (UCLA), Los Angeles, CA, 90095. Their emails are: dolecek@ee.ucla.edu, sunyz@ucla.edu and amiri@ucla.edu. D. Divsalar is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, and with email: Dariush.Divsalar@jpl.nasa.gov. This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA. Research is supported in part by NSF grant CCF-1161822 (JPL Task Plan 82-17473), NSF-CCR grant 1162501, NSF grant CAREER CCF-1150212 and a gift from ASTC. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors. Preliminary results of this work appeared in IEEE Milcom 2011, IEEE ISIT 2011 and 2012, and IEEE ITW 2011 and 2012.

[1] We remark that the copy-permute-scale sequence is equivalent to the copy-scale-permute sequence.

weights, whereas in the latter construction all copies of an edge receive the same weight. We refer to the non-binary LDPC codes obtained from an underlying protograph based on the copy-scale-permute operations as the *unconstrained* NBPB (U-NBPB) codes, and the non-binary codes obtained from an underlying protograph based on the scale-copy-permute operations as the *constrained* NBPB (C-NBPB) codes. We note that C-NBPB codes constitute the first graph-cover style non-binary code construction.

The goal of this paper is multifold: (1) to suitably generalize existing definitions and techniques from the binary to the non-binary domain, (2) to offer novel structured constructions of non-binary codes using guided edge weight assignment on the sequence of replicated protographs, (3) to provide ensemble performance evaluation of the resultant NBPB codes through the explicit computation of codeword enumerator and key non-codeword enumerators, (4) to offer new non-binary EXIT chart evaluation tool for NBPB codes, and (5) to offer explicit non-binary code designs based on NBPB constructions with excellent finite-length and asymptotic performance. Collectively, these results serve to advance the available toolbox of non-binary graph-based codes.

The rest of the paper is organized as follows. In Section II we introduce U-NBPB and C-NBPB codes. In Section III, we present codeword weight enumerators of U-NBPB codes along with illustrative examples. Counterpart enumerators of C-NBPB codes are presented in Section IV. In Section V, we extend the enumeration technique to trapping sets, stopping sets, and pseudocodewords. Iterative decoding thresholds of NBPB codes are derived in Section VI using a new EXIT chart analysis, suitably developed for non-binary protographs. Finite-length examples of U-NBPB and C-NBPB codes with excellent performance are discussed in Section VII. Section VIII concludes the paper and proposes questions for future investigation.

II. U-NBPB and C-NBPB Codes

There is a considerable freedom in choosing the edge weights in constructing protograph-based non-binary LDPC codes. Let us first consider the case where the edges are weighted independently of each other. We refer to resultant codes as *unconstrained* non-binary protograph-based (U-NBPB) codes. We then consider the constructions wherein the edge weights are assigned in bundles, and refer to resultant codes as *constrained* non-binary protograph-based (C-NBPB) codes. Both methods provide natural extensions of binary protograph-based code designs that are described by copy-permute operations, *cf.* [2], however, the U-NBPB construction is a series of copy-scale-permute operations whereas the C-NBPB construction is a series of scale-copy-permute operations.

A protograph $G = (V, C, E)$ [48] consists of the set $V = \{v_1, v_2, \ldots, v_{n_v}\}$ of variable nodes, the set $C = \{c_1, c_2, \ldots, c_{n_c}\}$ of check nodes, and the set $E = \{e_1, e_2, \ldots, e_{|E|}\}$ of edges connecting variable nodes and check nodes. Here, n_v is the total number of variable nodes, n_c is the total number of check nodes, and $|E|$ is the cardinality of the edge set E.

When the graph G is copied N times, each variable node $v_i \in V$ (each check node $c_i \in C$) in this mother protograph produces the set V_i of variable nodes $\{v_{i_1}, \ldots, v_{i_N}\}$ (the set C_i of check nodes $\{c_{i_1}, \ldots, c_{i_N}\}$) in the resultant daughter graph G^N. Likewise, each edge $e_i \in E$ in the mother protograph produces the set E_i of edges in the daughter graph where $E_i = \{e_{i,1}, \ldots, e_{i,N}\}$, and the edge $e_{i,j}$ for $1 \leq j \leq N$ connects the variable node v_{k_j} and the check node c_{l_j} if the edge e_i connects the variable node v_k and the check node c_l in the mother protograph. We denote the resultant daughter graph $G^N = (V^N, C^N, E^N)$.

We first provide the definition of U-NBPB codes and their ensembles. Let π_i denote the edge permutation associated with N copies of edge i.

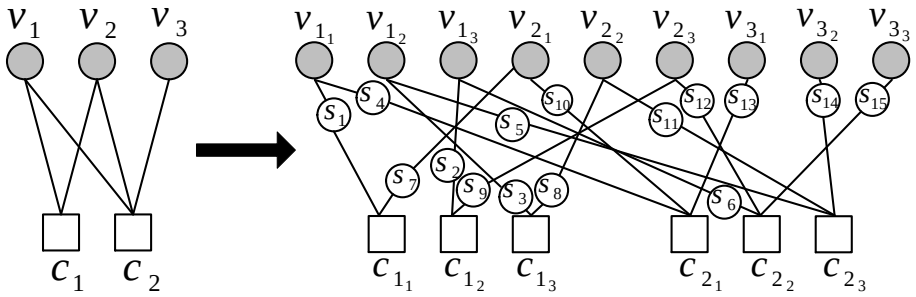
Definition 1 (U-NBPB code)**.** *Given the mother protograph $G = (V, C, E)$, a $(G, N, \{s_k\}_k, \{\pi_i\}_i)$ U-NBPB code is constructed from the daughter graph $G^N = (V^N, C^N, E^N)$ by permuting the edges in the set E_i according to π_i for each $1 \leq i \leq |E|$, followed by scaling each edge k in G^N by a non-zero element s_k of $GF(q)$ for $1 \leq k \leq N \cdot |E|$.* ■

The U-NBPB code construction is illustrated in Figure 1(a) based on the mother protograph with $n_v = 3$, $n_c = 2$ and $N = 3$. The U-NBPB ensemble is defined as follows.

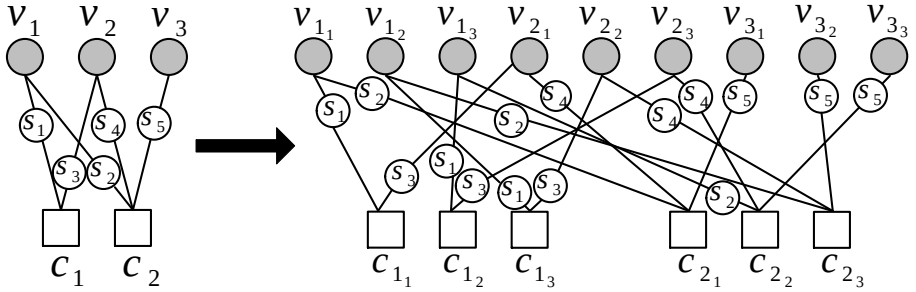
Definition 2 (U-NBPB code ensemble)**.** *The (G, N, q) U-NBPB ensemble is the collection of all $(G, N, \{s_k\}_k, \{\pi_i\}_i)$ U-NBPB codes with all possible choices of s_k's as non-zero elements of $GF(q)$ (for $1 \leq k \leq N \times |E|$) and $\{\pi_i\}$'s as all possible N-permutations (for $1 \leq i \leq |E|$).* ■

Another way of constructing a non-binary code based on a protograph is to first fix the non-binary edge weights of the protograph and then apply copy-and-permute operations to that protograph without altering the edge weights in the resultant graph. Consider again the underlying protograph $G = (V, C, E)$. Let $S_q = \{s_1, s_2, \ldots, s_{|E|}\}$ be the collection of non-zero scales (weights) with one-to-one association with the edges, namely $s_i \in S_q$ is associated with $e_i \in E$, and $s_i \neq 0 \in GF(q)$. Note that $G_q = (V, C, E, S_q)$ fully specifies a q-ary graph-based code with the variable node set V, check node set C, edge set E, and weight set S_q. For notational convenience, G_q will also be referred to as the *scaled* protograph when used to build a larger code.

A C-NBPB code is then constructed by a copy-and-permute procedure (while keeping the edge scalings fixed) applied to the non-binary protograph G_q. Here, the terminology 'constrained' refers to choosing labels for the baseline protograph and keeping them fixed during the subsequent copy-and-permute steps. When the mother graph G_q is copied N times, each variable node $v_i \in V$ (each check node $c_i \in C$) expands into the set V_i of variable nodes $\{v_{i_1}, \ldots, v_{i_N}\}$ (the set C_i of check nodes $\{c_{i_1}, \ldots, c_{i_N}\}$) in the resultant daughter graph G_q^N. Likewise, each edge $e_i \in E$ with its associated scale s_i expands into the set E_i of edges in G_q^N. Note that the elements of E_i, $E_i = \{e_{i,1}, \ldots, e_{i,N}\}$, each have the same scale $s_i \in S_q$ as $e_i \in E$, and the edge $e_{i,j}$ for $1 \leq j \leq N$ connects the variable node v_{k_j} and the check node c_{l_j} if the edge e_i connects the variable node v_k and the check node c_l



(a)



(b)

Fig. 1. Different NBPB code constructions: (a) The original unlabeled protograph and an example of a U-NBPB code construction with $N = 3$ (copy-scale-permute). (b) The original scaled protograph and an example of a C-NBPB code construction with $N = 3$ (scale-copy-permute).

with the same scale $s_i \in S_q$ in the original protograph G_q. We let $G_q^N = (V^N, C^N, E^N, S_q^N)$ denote the resultant graph. C-NBPB codes are then defined as follows.

Definition 3 (C-NBPB code)**.** *Given the mother non-binary protograph $G_q = (V, C, E, S_q)$, a $(G_q, N, \{\pi_i\}_i)$ C-NBPB code is constructed from the daughter graph $G_q^N = (V^N, C^N, E^N, S_q^N)$ by permuting the edges in the set E_i according to π_i for each $1 \leq i \leq |E|$.* ■

An example of a C-NBPB code construction is shown in Fig. 1(b) based on the mother protograph with $n_v = 3$, $n_c = 2$ and $N = 3$. The definition of the C-NBPB code ensemble then follows in the usual sense.

Definition 4 (C-NBPB code ensemble)**.** *The (G_q, N, q) C-NBPB code ensemble is the collection of all $(G_q, N, \{\pi_i\}_i)$ C-NBPB codes with the given choices of $s_i \in S_q$ as non-zero elements of $GF(q)$ (for $1 \leq i \leq |E|$) and $\{\pi_i\}$'s as all possible N-permutations (for $1 \leq i \leq |E|$).* ■

It is worth noting that the C-NBPB construction is a natural extension of the graph cover construction originally proposed to study pseudocodewords of a given code [28]. In contrast, here we use the graph-cover idea to construct a structured larger code based on the original smaller code. For more on graph covers of graph-based codes, please see [50].

It is clear that when the field size $q = 2$, both U-NBPB and C-NBPB constructions naturally reduce to the binary case, previously analyzed in the literature [2].

Lastly, we specify satisfied and unsatisfied check nodes. Let $G_q = (V, C, E, S_q)$ denote a bipartite graph describing a q-ary LDPC code, with the usual notation of V denoting the set of variable nodes, C denoting the set of check nodes, and the set E describing the edges between the nodes in V and C. For notational convenience recall that we use G_q to also denote a scaled protograph; we tacitly assume that parallel edges are not permitted in the graph describing a code but that they may be permitted in the protograph (as in the latter case the parallel edges will be eliminated during the subsequent copy-and-permute operations). In the non-binary case, each variable node v_i in V can have any value u_i in $GF(q)$. The weight of each edge $e_{i,j}$ in E connecting the variable node v_i and the check node c_j is given by $s_{i,j} \in S_q$ and is a non-zero element of $GF(q)$.

For the graph G_q, we say that the check node c_j of degree m is *satisfied* if $\sum_{i=1}^{m} s_{i,j} v_i = 0 \in GF(q)$, where $s_{i,j}$ is the weight given to the edge connecting the check node c_j and the variable node v_i. If $\sum_{i=1}^{m} s_{i,j} v_i \neq 0 \in GF(q)$ we say that the check node c_j is *unsatisfied*.

Codeword weight enumerators are known to be useful for

bounding the performance under the maximum likelihood (ML) decoding, whereas the enumerators of certain non-codeword objects are of interest when evaluating the performance under iterative decoding. In sequel, we will study several enumerators of interest.

III. U-NBPB Weight Enumerators

The section is composed of three parts. We first provide the exact weight enumerator of a code induced by one check node (Subsection III-A). We then discuss non-asymptotic ensemble weight enumerators (Subsection III-B) and the asymptotic ensemble weight enumerators (Subsection III-C). Some of the presented results build upon [2], and generalize these known results to the non-binary set-up. Throughout the section, illustrative examples accompany the derivations.

A. Weight enumerator of a check node and of its replicas

Let us begin building the enumerator result by first considering a check node c_j with degree m_j in the mother protograph G. We first establish the necessary notation.

It is convenient to view this check node c_j as a (m_j, m_j-1) linear block code $\mathcal{C}_j$ over $GF(q)$. Let $K_j = q^{(m_j-1)}$ denote the number of codewords in $\mathcal{C}_j$. Further, let $\mathbf{M}^{\mathcal{C}_j}$ be the $K_j \times m_j$ matrix with the codewords of $\mathcal{C}_j$ as its rows (whose entries are by construction in $GF(q)$), and let $\mathbf{M}_b^{\mathcal{C}_j}$ be the $K_j \times m_j$ *binary matrix* obtained by converting all non-zero entries of $\mathbf{M}^{\mathcal{C}_j}$ to 1. Note that by construction, some rows of $\mathbf{M}_b^{\mathcal{C}_j}$ may be the same. Let the collection $\mathcal{M}_b^{\mathcal{C}_j}$ represent all rows $\mathbf{x}$ of $\mathbf{M}_b^{\mathcal{C}_j}$, where $\mathbf{x} = [x_1, x_2, \ldots, x_{m_j}]$, $x_i \in \{0,1\}$. Define a $K_{j,r} \times m_j$ *binary matrix* $\mathbf{M}_{b,r}^{\mathcal{C}_j}$ as the submatrix of $\mathbf{M}_b^{\mathcal{C}_j}$ that consists of all distinct rows of $\mathbf{M}_b^{\mathcal{C}_j}$. The number of rows in $\mathbf{M}_{b,r}^{\mathcal{C}_j}$ is $K_{j,r} = 1 + \sum_{i=2}^{m_j} \binom{m_j}{i}$. Let the set $\mathcal{M}_{b,r}^{\mathcal{C}_j}$ represent the rows $\mathbf{x}_k = [x_{k,1}, x_{k,2}, \ldots, x_{k,m_j}]$, $x_{k,i} \in \{0,1\}$, for $i = 1, 2, \ldots m_j, k = 1, 2, \ldots, K_{j,r}$ of $\mathbf{M}_{b,r}^{\mathcal{C}_j}$.

Following the proposed construction of U-NBPB codes, we consider the N copies of node c_j in the daughter graph, and call the resultant $(Nm_j,\ N(m_j-1))$ linear block code $\mathcal{C}_j^N$. It is convenient to denote by n_k the number of occurrences of the k^{th} codeword among these N copies of c_j, and to collect them into the vector $\mathbf{n}$, where $\mathbf{n} = [n_1, n_2, \ldots, n_{K_j}]$. Lastly, let $A^{\mathcal{C}_j^N}(\mathbf{w})$ denote the weight-vector enumerator of $\mathcal{C}_j^N$ where $\mathbf{w} = [w_1, w_2, \ldots, w_{m_j}]$ is the weight vector of the input message in $\mathcal{C}_j^N$, where the entry w_i denotes the number of occurrences of a non-zero value in position i, $1 \le i \le m_j$, over the set of input messages.

With the above, the main result of this subsection is provided in the following theorem that characterizes the weight enumerator of the code $\mathcal{C}_j^N$ (in the daughter graph G^N) that is described by N copies of the single check node c_j (in the mother graph G). For the ease of exposition and since we currently focus on the single check node, we suppress the index j in c_j, $\mathcal{C}_j^N$ and $K_{j,r}$, and simply refer to the check node as c, its resultant code as $\mathcal{C}^N$, and reduced row dimension as K_r.

Throughout the analysis

$$C\left(N; x_1, x_2, \cdots, x_L\right) = \frac{N!}{x_1! x_2! \cdots x_L!}, \tag{1}$$

denotes the multinomial coefficient, where x_i's are non-negative integers summing to N.

Theorem 1. *The weight-vector enumerator $A^{\mathcal{C}^N}(\mathbf{w})$ of $\mathcal{C}^N$ is given by,*

$$A^{\mathcal{C}^N}(\mathbf{w}) = \sum_{\{\mathbf{n}\}} C\left(N; n_1, n_2, \ldots, n_{K_r}\right) e^{\mathbf{n}\cdot\mathbf{f}_q^T}, \tag{2}$$

where $C\left(N; n_1, n_2, \ldots, n_{K_r}\right)$ is the multinomial coefficient specified in (1)*, and $\{\mathbf{n}\}$ is the set of integer-vector solutions to $\mathbf{w} = \mathbf{n}\cdot\mathbf{M}_{b,r}^{\mathcal{C}}$, with $n_1, n_2, \ldots, n_{K_r} \ge 0$, and $\sum_{k=1}^{K_r} n_k = N$. The vector $\mathbf{f}_q = [f_{q,1}, f_{q,2} \ldots, f_{q,K_r}]$ has entries $f_{q,k} = \ln g(q, |\mathbf{x}_k|)$, where $\mathbf{x}_k$ is the k-th element of $\mathcal{M}_{b,r}^{\mathcal{C}}$, $|\mathbf{x}_k|$ is the weight of $\mathbf{x}_k$, and $g(q,i) = \frac{q-1}{q}[(q-1)^{i-1} + (-1)^i]$.*

Proof: The weight-vector enumerators $\{A^{\mathcal{C}^N}(\mathbf{w})\}$ may be found as the coefficients of a multi-dimensional generating function of $\{A^{\mathcal{C}^N}(\mathbf{w})\}$. The generating function of the code $\mathcal{C}$ induced by the check node c is $\sum_{\mathbf{x}\in\mathcal{M}_b^{\mathcal{C}}} W_1^{x_1} W_2^{x_2} \cdots W_m^{x_m}$, where the W_i's are indeterminate bookkeeping variables.

From [21], the weight generating function for the code $\mathcal{C}$ induced by a single check node c of degree m, is given by $A^{\mathcal{C}}(W) = \frac{1}{q}[(1+(q-1)W)^m + (q-1)(1-W)^m]$, which also holds for $GF(q)$. This generating function can also be written as $A^{\mathcal{C}}(W) = \sum_{w=0}^{m} \binom{m}{w} g(q,w) W^w$. For our problem, the number $g(q,w)$ represents exactly the number of repeated rows with weight w in $\mathcal{M}_b^{\mathcal{C}}$. Thus, $\sum_{\mathbf{x}\in\mathcal{M}_b^{\mathcal{C}}} W_1^{x_1} W_2^{x_2} \cdots W_m^{x_m} = \sum_{\forall \mathbf{x}_k \in \mathcal{M}_{b,r}^{\mathcal{C}}} g(q, |\mathbf{x}_k|) W_1^{x_{k,1}} W_2^{x_{k,2}} \cdots W_m^{x_{k,m}}$, where $\mathbf{x}_k$ is the k-th element of $\mathcal{M}_{b,r}^{\mathcal{C}}$ and $|\mathbf{x}_k|$ is its weight (that is, the sum of its entries). The generating function for N copies of this check node in the daughter graph is then

$$A^{\mathcal{C}^N}(W_1, W_2, \ldots, W_m) = \left(\sum_{\forall \mathbf{x}_k \in \mathcal{M}_{b,r}^{\mathcal{C}}} g(q, |\mathbf{x}_k|)\ W_1^{x_{k,1}} W_2^{x_{k,2}} \cdots W_m^{x_{k,m}} \right)^N. \tag{3}$$

Applying the multinomial theorem to (3) yields,

$$\begin{aligned} A^{\mathcal{C}^N}(W_1, W_2, \ldots, W_m) = & \sum_{\substack{n_1, n_2, \ldots, n_{K_r} \ge 0 \\ n_1+n_2+\cdots+n_{K_r}=N}} C\left(N; n_1, n_2, \ldots, n_{K_r}\right) \\ & \times \prod_{\forall \mathbf{x}_k \in \mathcal{M}_{b,r}^{\mathcal{C}}} \left(g(q, |\mathbf{x}_k|) W_1^{x_{k,1}} W_2^{x_{k,2}} \cdots W_m^{x_{k,m}} \right)^{n_k}. \end{aligned} \tag{4}$$

Then, (4) can be written as

$$\begin{aligned} A^{\mathcal{C}^N}(W_1, W_2, \ldots, W_m) = & \sum_{\mathbf{w}} \sum_{\{\mathbf{n}\}} C\left(N; n_1, n_2, \ldots, n_{K_r}\right) \\ & \times \left(\prod_{\forall \mathbf{x}_k \in \mathcal{M}_{b,r}^{\mathcal{C}}} [g(q, |\mathbf{x}_k|)]^{n_k} \right) \\ & \times W_1^{w_1} W_2^{w_2} \cdots W_m^{w_m}, \end{aligned} \tag{5}$$

where $\{\mathbf{n}\}$ is the set of integer solutions to $\mathbf{w} = \mathbf{n}\cdot\mathbf{M}_{b,r}^{\mathcal{C}}$, under the constraints $n_1, n_2, \ldots, n_{K_r} \ge 0$ and $\sum_{k=1}^{K_r} n_k = N$, and where $w_l = \sum_{\forall \mathbf{x}_k \in \mathcal{M}_{b,r}^{\mathcal{C}}} x_{k,l} n_k$, $l = \{1, 2, \ldots, m\}$. To see the

last step, note that the product in (4) can be manipulated as follows

$$\prod_{\forall \mathbf{x}_k \in \mathcal{M}_{b,r}^{\mathcal{C}}} \left(W_1^{x_{k,1}} W_2^{x_{k,2}} \cdots W_m^{x_{k,m}}\right)^{n_k} = W_1^{w_1} W_2^{w_2} \cdots W_m^{w_m}. \tag{6}$$

Also, if $\mathbf{w} = \mathbf{n} \cdot \mathbf{M}_{b,r}^{\mathcal{C}}$ has more than one solution for $\mathbf{n}$, the term $W_1^{w_1} W_2^{w_2} \cdots W_m^{w_m}$ will appear as a common factor in all of the terms that are associated with these solutions. This observation explains the presence of the second summation in (5). The generating function of $\{A^{\mathcal{C}^N}(\mathbf{w})\}$ can also be written as

$$A^{\mathcal{C}^N}(W_1, W_2, \ldots, W_m) = \sum_{\mathbf{w}} A^{\mathcal{C}^N}(\mathbf{w}) W_1^{w_1} W_2^{w_2} \cdots W_m^{w_m}. \tag{7}$$

Finally, comparing (7) and (5) leads to (2). ■

Note that if we choose to use $\mathcal{M}_b^{\mathcal{C}}$ (which has repeated elements) then

$$A^{\mathcal{C}^N}(\mathbf{w}) = \sum_{\{\mathbf{n}\}} C\left(N; n_1, n_2, \ldots, n_K\right), \tag{8}$$

where $\{\mathbf{n}\}$ is now the set of integer-vector solutions to $\mathbf{w} = \mathbf{n} \cdot \mathbf{M}_b^{\mathcal{C}}$, with $n_1, n_2, \ldots, n_K \geq 0$, $\sum_{k=1}^{K} n_k = N$, and $K = q^{m-1}$. We now provide an illustrative example.

Example 1. *Consider a $(3, 2)$ linear block code over $GF(q)$ replicated N times, whose weight enumerator we seek to compute. There is only one check node so we simply refer to this node as c and to the code it generates as $\mathcal{C}$. Let $\mathcal{C}^N$ denote the $(3N, 2N)$ code obtained by replicating $\mathcal{C}$ code N times. Our objective is to evaluate $A^{\mathcal{C}^N}(w_1, w_2, w_3)$.*

We now show that if we start with (8) we can in fact obtain (2) with reduced computational complexity. Observe that $M_b^{\mathcal{C}}$ is a $q^2 \times 3$ (binary) matrix with repeated rows. Solving the equation $\mathbf{w} = \mathbf{n} \cdot \mathbf{M}_b^{\mathcal{C}}$ for $K = q^2$ integers n_i, $i \in \{1, 2, \ldots, K\}$, only requires to solve for 5 integers.

In the set of codewords (x_1, x_2, x_3) of this $(3, 2)$ code, apart from the all-zeros codeword, there are $(q-1)$ codewords of Hamming weight 2, where x_i and x_j are non-zero, and x_k is zero element of $GF(q)$, for i, j, k distinct indices from the set $\{1, 2, 3\}$. There are also $(q-1)(q-2)$ codewords of Hamming weight 3. The set $\mathcal{M}_{b,r}^{\mathcal{C}}$ is $\{[0,0,0], [0,1,1], [1,0,1], [1,1,0], [1,1,1]\}$ and the matrix $\mathbf{M}_{b,r}^{\mathcal{C}}$ (the reduced version of the matrix $\mathbf{M}_b^{\mathcal{C}}$) is then the lexicographical ordering of these rows.

Computing the solution to $\mathbf{w} = \mathbf{n} \cdot \mathbf{M}_b^{\mathcal{C}}$ is equivalent to solving the set of equations $\mathbf{w} = \mathbf{k} \cdot \mathbf{M}_{b,r}^{\mathcal{C}}$, $n_1 = k_1$, $\sum_{i=2}^{q} n_i = k_2$, $\sum_{i=q+1}^{2q-1} n_i = k_3$, $\sum_{i=2q}^{3q-2} n_i = k_4$, $\sum_{i=3q-1}^{q^2} n_i = k_5$, where $\mathbf{k} = [k_1, k_2, k_3, k_4, k_5]$, and $\mathbf{w} = [w_1, w_2, w_3]$. An application of the multinomial theorem results in $\sum_{\substack{i_1, i_2, \ldots, i_l \geq 0 \\ i_1 + i_2 + \cdots + i_l = t}} \frac{1}{i_1! i_2! \ldots i_l!} = \frac{l^t}{t!}$. Using this equality, one can show that (8) reduces to

$$A^{\mathcal{C}^N}(\mathbf{w}) = \sum_{\{\mathbf{k}\}} C\left(N; k_1, \ldots, k_5\right) (q-1)^{k_2 + k_3 + k_4 + k_5} (q-2)^{k_5}, \tag{9}$$

where $\{\mathbf{k}\}$ is the set of integer-vector solutions to $\mathbf{w} = \mathbf{k} \cdot \mathbf{M}_{b,r}^{\mathcal{C}}$, with $k_1, k_2, \ldots, k_5 \geq 0$ and $\sum_{i=1}^{5} k_i = N$. Solving this set of equations we get $k_1 = N - s + k_5/2$, $k_2 = s - w_1 - k_5/2$, $k_3 = s - w_2 - k_5/2$, and $k_4 = s - w_3 - k_5/2$, where $s = (w_1 + w_2 + w_3)/2$. Since $k_i \geq 0$, we have $\max\{0, 2(s-N)\} \leq k_5 \leq 2s - 2\max\{w_1, w_2, w_3\}$.

If $w_1 + w_2 + w_3$ is even, then

$$A^{\mathcal{C}^N}(\mathbf{w}) = \sum_l C\left(N; (N - s + l), (s - w_1 - l), (s - w_2 - l), (s - w_3 - l), (2l)\right) \times (q-1)^{(s-l)} (q-2)^{2l}, \tag{10}$$

where $l = k_5/2$ and k_5 is even. If $w_1 + w_2 + w_3$ is odd, then

$$A^{\mathcal{C}^N}(w) = \sum_l C\big(N; (N - s + l + 1/2), (s - w_1 - l - 1/2), (s - w_2 - l - 1/2), (s - w_3 - l - 1/2), (2l+1)\big) \times (q-1)^{(s-l-1/2)} (q-2)^{2l+1}, \tag{11}$$

where $l = (k_5 - 1)/2$ and k_5 is odd. ■

Based on this exact combinatorial count on the per-node basis, in the next section we derive the exact weight enumerator of the U-NBPB ensemble.

B. Weight enumerator of the U-NBPB ensemble

Before stating the enumerator result, we first define the non-binary uniform interleaver.

Definition 5 (Non-binary uniform interleaver)**.** *A length-L non-binary uniform interleaver over $GF(q)$ is a probabilistic device that maps each input of length L and of Hamming weight w into the $(q-1)^w \binom{L}{w}$ distinct weighted permutations of the input, such that it generates each weighted permutation with equal probability, $\frac{1}{(q-1)^w \binom{L}{w}}$.* ■

The notion of Uniform Codeword Selector (UCS) was introduced in [18] in the context of the concatenation of non-binary product codes. This definition is equivalent to the notion of non-binary uniform interleaver in this paper.

With the protograph based set-up, it is convenient to view the resultant code as a serial concatenation of certain component codes (*cf.* [7]). Suppose $\mathcal{C}_1$ and $\mathcal{C}_2$ are two serially concatenated block codes over $GF(q)$ that are connected by a length-L non-binary uniform interleaver over $GF(q)$. For the given codes $\mathcal{C}_1$ and $\mathcal{C}_2$, let $SCC = SCC(\mathcal{C}_1, \mathcal{C}_2)$ be the resultant ensemble over all possible interleavers.

Lemma 1. *Consider two block codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of dimensions K_l and codeword lengths N_l, $l = 1, 2$, that are serially concatenated via a non-binary uniform interleaver, with all system components over $GF(q)$. The average number of codewords of Hamming weight d that are created by inputs of Hamming weight f in the resultant SCC ensemble is given by*

$$A_{f,d}^{SCC} = \sum_w \frac{A_{f,w}^{\mathcal{C}_1} A_{w,d}^{\mathcal{C}_2}}{(q-1)^w \binom{K_2}{w}}, \tag{12}$$

where $A_{f,w}^{\mathcal{C}_1}$ is the number of codewords in $\mathcal{C}_1$ of Hamming weight w corresponding to $\mathcal{C}_1$-encoder inputs of Hamming weight f, and $A_{w,d}^{\mathcal{C}_2}$ is the number of codewords in $\mathcal{C}_2$ of Hamming weight d corresponding to C_2-encoder inputs of Hamming weight w.

Proof: For the constituent code $\mathcal{C}_1$, there are $(q-1)^f\binom{K_1}{f}$ possible encoder inputs of Hamming weight f, and they produce $A^{\mathcal{C}_1}_{f,w}$ codewords of Hamming weight w. Likewise, for the constituent code $\mathcal{C}_2$, there are $(q-1)^w\binom{K_2}{w}$ possible encoder inputs of weight w, and they produce $A^{\mathcal{C}_2}_{w,d}$ codewords of Hamming weight d. When $\mathcal{C}_1$ and $\mathcal{C}_2$ are connected via a length-K_2 non-binary uniform interleaver ($K_2 = N_1$), each of these $A^{\mathcal{C}_1}_{f,w}$ codewords in $\mathcal{C}_1$ maps into one of the $A^{\mathcal{C}_2}_{w,d}$ codewords of $\mathcal{C}_2$ with probability $\frac{1}{(q-1)^w\binom{K_2}{w}}$.

Averaged over the resultant SCC ensemble, there are $A^{\mathcal{C}_1}_{f,w}A^{\mathcal{C}_2}_{w,d}/(q-1)^w\binom{K_2}{w}$ codewords of Hamming weight d corresponding to the SCC encoder inputs of weight f and to the $\mathcal{C}_2$-encoder inputs of weight w. Summing these codewords over all w, (12) follows. ∎

Based on Lemma 1 we derive the exact weight enumerator over the entire U-NBPB ensemble as follows.

Recall that there are n_v variable nodes and n_c check nodes in the mother protograph G, and that m_j denotes the degree of the check node c_j. Let t_i denote the degree of the variable node v_i. Recall that the U-NBPB ensemble consists of all codes obtained by performing all possible weight permutations on the edges of the daughter graph G^N. Let $\mathbf{d}_j = \left[d_{j_1}, d_{j_2}, ..., d_{j_{m_j}}\right]$ be the weight vector which describes the weights of the N symbol words on the edges connected to check node c_j, produced by the variable nodes $\{v_{j_1}, v_{j_2}, ..., v_{j_{m_j}}\}$ neighboring c_j.

It is convenient to specify Kronecker Delta $\kappa_{x,y}$ as

$$\kappa_{x,y} = \begin{cases} 1 & \text{if } x = y, \text{ and} \\ 0 & \text{if} x \neq y. \end{cases} \tag{13}$$

If x and y are vectors, we interpret Kronecker Delta having value 1 only if x and y agree in all components.

Theorem 2. *The weight-vector enumerator of the U-NBPB code averaged over the entire ensemble is*

$$A(\mathbf{d}) = \frac{\prod_{j=1}^{n_c} A^{\mathcal{C}_j^N}(\mathbf{d}_j)}{\prod_{i=1}^{n_v}(q-1)^{d_i(t_i-1)}\binom{N}{d_i}^{t_i-1}}, \tag{14}$$

where $A^{\mathcal{C}_j^N}(\mathbf{d}_j)$ is the weight-vector enumerator of the code $\mathcal{C}_j^N$ induced by the N copies of the check node c_j. Here, the elements of $\mathbf{d}_j$ comprise a subset of the elements of $\mathbf{d} = [d_1, d_2, ..., d_{n_v}]$, and this subset is obtained from the edge connections in the mother protograph G (see Fig. 2 for illustration).

Proof: Consider N copies of each node in the protograph as a constituent code. These constituent codes are then inter-connected through non-binary uniform interleavers each of size $N \times N$. The N copies of each variable node $v_i \in G$ can be treated as a constituent code with one input of weight d_i and t_i outputs $[w_{i,1}, w_{i,2}, \ldots, w_{i,t_i}]$. The input-output weight coefficient for node v_i is then $(q-1)^{d_i}\binom{N}{d_i}\kappa_{d_i,w_{i,1}} \cdots \kappa_{d_i,w_{i,t_i}}$. The N copies of each check node $c_j \in G$ can be treated as a constituent code with m_j input weights $\mathbf{w_j} = [w_{j,1}, w_{j,2}, \ldots, w_{j,m_j}]$ and no output.

Let $A^{\mathcal{C}_j^N}(\mathbf{w}_j)$ be the input weight enumerator of the check node group $\mathcal{C}_j^N$. Let $A(\mathbf{d})$ represent the number of sequences each with weight vector $\mathbf{d} = [d_1, d_2, \ldots, d_{n_v}]$ that is applied to the variable nodes according to the protograph constraints.

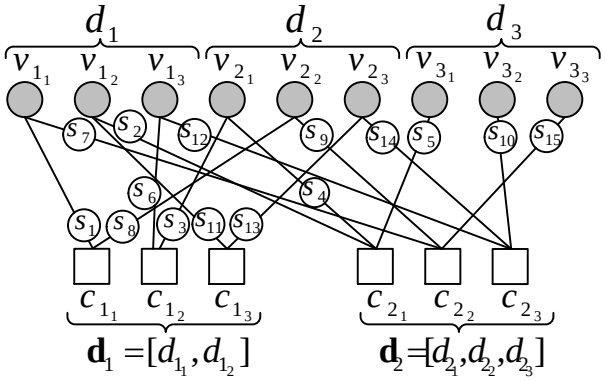


Fig. 2. Illustration of the relationship between vectors $\mathbf{d} = [d_1, d_2, d_3]$ and $\mathbf{d_1} = [d_{1_1}, d_{1_2}]$, $\mathbf{d_2} = [d_{2_1}, d_{2_2}, d_{2_3}]$ for an U-NBPB code with $N = 3$, where $d_{1_1} = d_1$, $d_{1_2} = d_2$, $d_{2_1} = d_1$, $d_{2_2} = d_2$, $d_{2_3} = d_3$.

Then, the result of Lemma 1 is applied to individual concatenations to obtain the average protograph weight-vector enumerator as,

$$A(\mathbf{d}) = \sum_{\substack{w_{m,u} \\ m=1,\ldots,n_v \\ u=1,\ldots,t_m}} \frac{\prod_{k=1}^{n_v}[(q-1)^{d_k}\binom{N}{d_k}g_{d_k,w_{k,1}}\cdots g_{d_k,w_{k,t_k}}]}{\prod_{s=1}^{n_v}\prod_{r=1}^{t_s}(q-1)^{w_{s,r}}\binom{N}{w_{s,r}}} \times \prod_{i=1}^{n_c} A^{\mathcal{C}_j^N}(\mathbf{w}_j). \tag{15}$$

Here, the summation is over all weights $w_{m,u}$, where $w_{m,u}$ is the weight along the u^{th} edge of variable node v_m. Note that $w_{j,l} = w_{i,k}$ if the l^{th} edge of check node c_j is the k^{th} edge of variable node v_i. The vector $\mathbf{d}_j = \left[d_{j_1}, d_{j_2}, ..., d_{j_{m_j}}\right]$ is a weight vector which describes the weights of the N-symbol words on the edges connected to check node c_j, produced by the variable nodes neighboring c_j. The elements of $\mathbf{d}_j$ comprise a subset of the elements of $\mathbf{d}$. Then, (15) reduces to

$$A(\mathbf{d}) = \frac{\prod_{j=1}^{n_c} A^{\mathcal{C}_j^N}(\mathbf{d}_j)}{\prod_{i=1}^{n_v}(q-1)^{d_i(t_i-1)}\binom{N}{d_i}^{t_i-1}},$$

as desired. ∎

The average number of codewords of symbol weight d in the ensemble, denoted by A_d, equals the sum of $A(\mathbf{d})$ over all $\mathbf{d}$ for which $\sum_{\{d_i:v_i\in V\}} d_i = d$.

Example 2. *In this example we calculate the symbol weight enumerator for the three protographs given in Fig. 3. The first protograph describes a regular $(2, 4)$ code, the second and the third protographs are obtained by adding an accumulator to the regular $(2, 4)$ protograph followed by puncturing of a node. We refer to the former as the punctured $(2, 4)$ type 1 protograph and we refer to the latter as the punctured $(2, 4)$ type 2 protograph. Here and in subsequent examples black nodes are punctured. In the calculations, all three code ensembles have 32 transmitted variable nodes and are defined over $GF(8)$, so the total number of bits is 96. As a result, the first code has $N = 16$, and the second and the third code have $N = 8$. The results for the average weight enumerator A_d are*

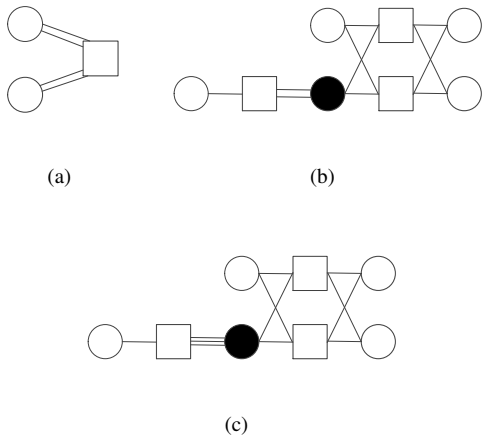


Fig. 3. Three candidate protographs: (a) Regular $(2,4)$ protograph, (b) Punctured $(2,4)$ type 1 protograph, and (c) Punctured $(2,4)$ type 2 protograph. Black nodes are punctured.

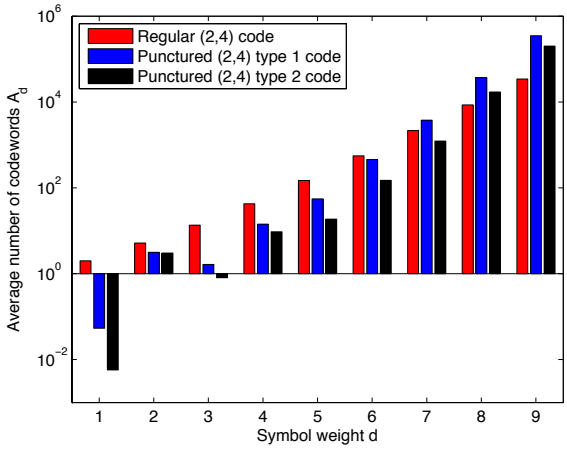


Fig. 4. Weight enumerator for the U-NBPB ensembles of the protographs in Fig. 3 over $GF(8)$ for symbol length 32.

shown in Fig. 4 for the smallest 9 non-zero codeword symbol weights. To further illustrate the enumeration technique, we plot the weight enumerators of the three protograph code ensembles with 80 transmitted variable nodes over $GF(8)$, i.e., $N = 40$ for the first code and $N = 20$ for the second and third codes. The results are shown in Fig. 5, also for the lowest 9 non-zero codeword symbol weights. We note that, relative to the regular $(2,4)$ code, the punctured type 1 code and the punctured type 2 code both have on average fewer low symbol weight codewords, and that the type 2 code has the best distribution of the three codes for small codeword weights. As we shall see later in Section VI, this relative ordering of codes will also be consistent with the threshold calculations computed for the three codes.

Example 3. *Continuing on with the baseline regular $(2,4)$ protograph repeated $N = 20$ times (i.e., with 40 symbols), in Fig. 6, we now plot the average number of codewords, A_d, as a function of the field order q for the first few smallest values of*

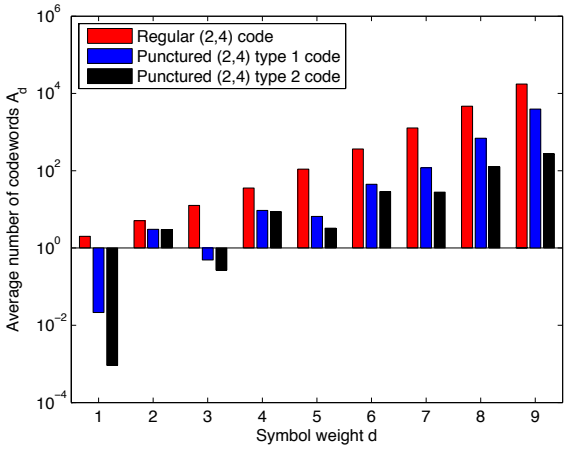


Fig. 5. Weight enumerator for the U-NBPB ensembles of the protographs in Fig. 3 over $GF(8)$ for symbol length 80.

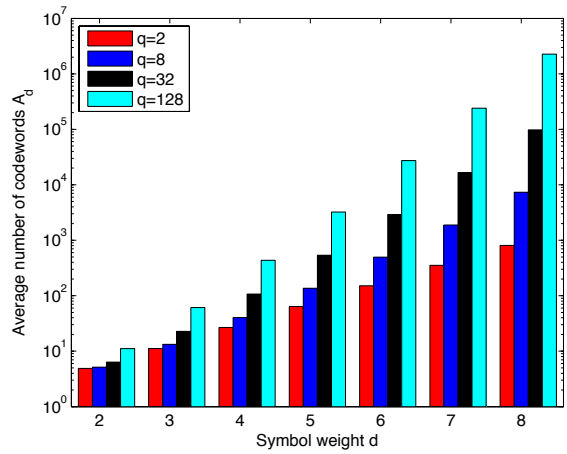


Fig. 6. Weight enumerator for the U-NBPB ensembles of the regular $(2,4)$ protograph in Fig. 3 for symbol length 40 and over different field orders.

the non-zero symbol weight. As expected, the average number of codewords increases with q.

C. Asymptotic ensemble weight enumerators

Given that the formulas in the previous subsection involve the number of copies N, we define the normalized logarithmic asymptotic weight (the growth rate) to be

$$r(\delta) = \limsup_{N\to\infty} \frac{\ln A_d}{N} = \limsup_{N\to\infty} \frac{\ln A_{\delta N}}{N}, \tag{16}$$

where $\delta = d/N$. Note that $n = n_v \cdot N$, so the growth rate in terms of n can be expressed as

$$\tilde{r}(\tilde{\delta}) = \limsup_{n\to\infty} \frac{\ln A_d}{n}, \tag{17}$$

where $\tilde{r}(\tilde{\delta}) = \frac{1}{n_v} r(\tilde{\delta} n_v)$.

From (14), we have

$$\ln A(\mathbf{d}) = \sum_{j=1}^{n_c} \ln A^{\mathcal{C}_j^N}(\mathbf{d}_j) - \sum_{i=1}^{n_v} (t_i - 1) \left[d_i \ln(q-1) + \ln \binom{N}{d_i} \right]. \tag{18}$$

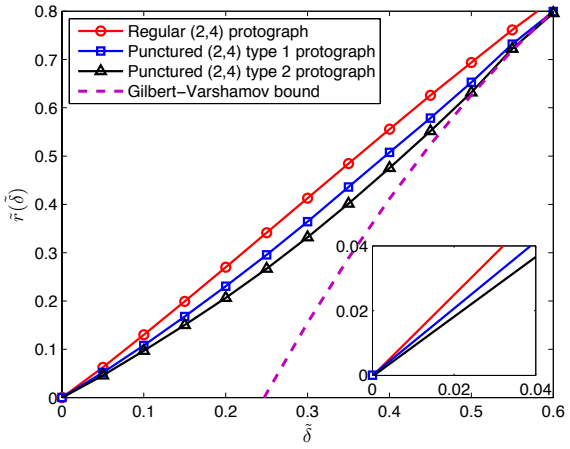


Fig. 7. Asymptotic symbol weight enumerators of protographs in Fig. 3 over $GF(8)$.

Let $\delta_i = d_i/N$, and take the limit as $N \to \infty$. Using $\limsup_{N\to\infty} \ln\binom{N}{d_i}/N = H(\delta_i) = -(1-\delta_i)\ln(1-\delta_i) - \delta_i \ln \delta_i$, [12], we obtain

$$r(\delta) = \max_{\{\delta_l : v_l \in V\}} \left\{ \sum_{j=1}^{n_c} a^{c_j}(\delta_j) - \sum_{i=1}^{n_v} (t_i - 1)[H_q(\delta_i)] \right\}, \tag{19}$$

under the constraint $\sum_{\{\delta_i : v_i \in V\}} \delta_i = \delta$, and $H_q(\delta_i) \triangleq \delta_i \ln(q-1) + H(\delta_i)$. In (19), $a^{c_j}(\delta_j)$ is the asymptotic weight-vector enumerator associated with the check node c_j, defined as

$$a^{c_j}(\omega) = \limsup_{N\to\infty} \frac{\ln A^{\mathcal{C}_j^N}(\mathbf{w})}{N}, \tag{20}$$

where $\omega = \mathbf{w}/N$, and $\delta_j = \mathbf{d}_j/N$.

Let $P_{\boldsymbol{\omega}} = [p_1, p_2, \ldots, p_K]$ be the relative proportion of occurrences of each codeword of constituent check node code $\mathcal{C}$ in a sequence of N codewords, where $p_k = n_k/N$ and n_k is the number of occurrences of the k^{th} codeword. We then let the type class of $P_{\boldsymbol{\omega}}$, $T(P_{\boldsymbol{\omega}})$, be the set of all length-N sequences of codewords in $\mathcal{C}$, each containing n_k occurrences of the k^{th} codeword in $\mathcal{C}$, for $k = 1, 2, ..., K_r$. Observe that $|T(P_{\boldsymbol{\omega}})| = C(N; n_1, n_2, \ldots, n_{K_r})$. From [12, Thm.12.1.3] and [2], $|T(P_{\boldsymbol{\omega}})| \to e^{N\cdot H(P_{\boldsymbol{\omega}})}$, as $N \to \infty$, where $H(P_{\boldsymbol{\omega}}) = -\sum_{k=1}^{K_r} p_k \ln p_k$. As $N \to \infty$ (2) is

$$A^{\mathcal{C}}(\mathbf{w}) = \sum_{\{\mathbf{n}\}} C(N; n_1, n_2, \ldots, n_{K_r})\, e^{\mathbf{n}\cdot\mathbf{f}_q^T}$$
$$= \sum_{\{P_{\boldsymbol{\omega}}\}} |T(P_{\boldsymbol{\omega}})| e^{NP_{\boldsymbol{\omega}}\cdot\mathbf{f}_q^T} \to \sum_{\{P_{\boldsymbol{\omega}}\}} e^{N[H(P_{\boldsymbol{\omega}}) + P_{\boldsymbol{\omega}}\cdot\mathbf{f}_q^T]},$$

under the constraint that $\{P_{\boldsymbol{\omega}}\}$ is the set of solutions to $\boldsymbol{\omega} = P_{\boldsymbol{\omega}} \cdot \mathbf{M}_{b,r}^{\mathcal{C}}$, with $p_1, p_2, \ldots, p_{K_r} \geq 0$ and $\sum_{k=1}^{K_r} p_k = 1$. It follows from (20) that

$$a^{\mathcal{C}}(\boldsymbol{\omega}) = \max_{\{P_{\boldsymbol{\omega}}\}} \left\{ H(P_{\boldsymbol{\omega}}) + P_{\boldsymbol{\omega}} \cdot \mathbf{f}_q^T \right\}. \tag{21}$$

Example 4. *Continuing with the protographs discussed in Example 2, we compute the asymptotic symbol weight enumerators for the three protographs for $q = 8$, as shown in Fig 7. As we can see, in the asymptotic case, the punctured type 1 protograph and the punctured type 2 protograph both have on average fewer low symbol weight codewords than the regular $(2, 4)$ protograph. This result is in agreement with the finite length calculation (and will be later shown to be also consistent with the threshold calculations).*

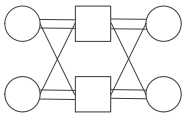


Fig. 8. Regular $(3, 6)$ protograph.

Note that the ensemble of all rate-R, q-ary ("random") linear codes (whose parity-check matrix entries are i.i.d. uniform) has the weight enumerator $A^{\mathcal{C}}(w) = (q-1)^w \binom{n}{w} e^{-n(1-R)\ln(q)}$ and the asymptotic weight enumerator [21]

$$\tilde{r}(\tilde{\delta}) = H_q(\tilde{\delta}) - (1-R)\ln(q), \tag{22}$$

which corresponds to the asymptotic Gilbert-Varshamov bound for the non-binary case. In Fig. 7, we plot the Gilbert-Varshamov bound for $q = 8$. Similar to the binary protograph case studied in [2], the asymptotic symbol weight enumerators converge to the Gilbert-Varshamov bound as $\tilde{\delta}$ gets larger. Here, again, of the three candidate protographs, the punctured type 2 protograph offers the growth rate closest to the Gilbert-Varshamov bound.

Example 5. *In this example, we provide the asymptotic weight enumerator for the regular $(3, 6)$ protograph (presented in Fig. 8) over $GF(q)$, as shown in Fig. 9. We also note that our result for $GF(2)$ is in agreement with [2]. From the figure, we can see that as q increases, there are fewer low weight codewords. In addition, as q increases, the growth rate of high weight codewords increases. We use ν_{min} to denote the second zero crossing of $\tilde{r}(\tilde{\delta})$ (the first zero crossing is $\tilde{r}(0) = 0$). The second zero crossing, if it exists, is called the typical relative minimum distance.*

Fig. 10 shows how the typical relative minimum distance ν_{min} changes with varying q. Consistent with [16], while the Gilbert -Varshamov bound grows monotonically with q, ν_{min} is in fact non-monotonic. In particular, ν_{min} attains maximum value for $q = 64, 128$.

IV. Enumerators of C-NBPB Codes

Building upon the computational machinery developed in the previous section for U-NBPB codes, in this section we derive the corresponding codeword weight enumerators for the C-NBPB codes. As before, we first consider the weight enumerator of a single check node (Section IV-A), followed by the code weight enumerator (Section IV-B) and the asymptotic analysis (Section IV-C). We compare unconstrained and constrained NBPB constructions complexity-wise via an example in Section IV-D.

A. Weight enumerator of a check node and of its replicas

Consider a degree-m_j check node c_j in the scaled protograph G_q defined over $GF(q)$, with neighboring vari-

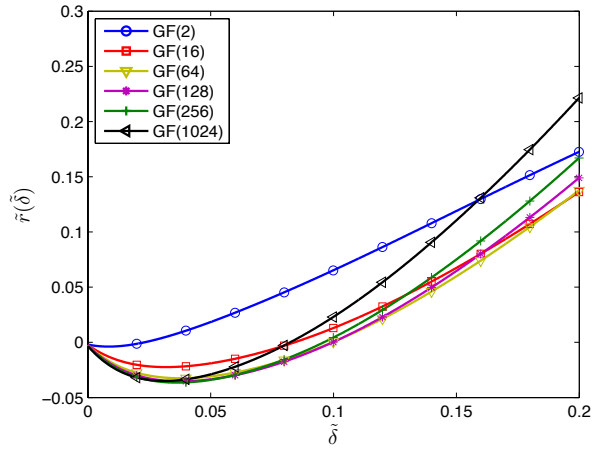


Fig. 9. Asymptotic symbol weight enumerators of regular $(3, 6)$ protograph for different q.

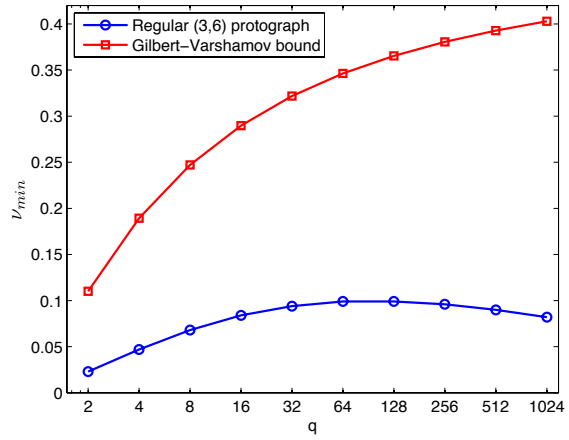


Fig. 10. Typical minimum distance of regular $(3, 6)$ protograph for different q.

able nodes given by the vector $\mathbf{v}_j = (v_{j_1}, v_{j_2}, \ldots v_{j_{m_j}})$ and scalings on the incident edges given by the vector $\mathbf{s}_j = (s_{j_1}, s_{j_2}, \ldots s_{j_{m_j}})$, where s_{j_i}'s are non-zero elements of $GF(q)$ for $i = 1, 2, \ldots, m_j$. Since the edge weights are a priori chosen by construction, we view the node c_j with specified $\mathbf{s}_j$ as a $(m_j, m_j - 1)$ single parity check, linear code $\mathcal{C}_j$ over $GF(q)$.

Recall the notation from the previous section: we again let $K_j = q^{(m_j-1)}$ be the total number of codewords in $\mathcal{C}_j$. Further, we also let $\mathbf{M}^{\mathcal{C}_j}$ be the $K_j \times m_j$ matrix with the codewords of $\mathcal{C}_j$ as its rows.

Consider a $1 \times m_j$ codeword $\mathbf{x} \in \mathcal{C}_j$. Let the mapping $\varphi(\mathbf{x})$ be defined as the symbol indicator,

$$\varphi(\mathbf{x}) = [x_{1,1} \cdots x_{1,(q-1)},\ x_{2,1} \cdots x_{2,(q-1)}, \ldots,\ x_{m_j,1} \cdots x_{m_j,(q-1)}],$$

where $x_{i,\ell} = 1$, if the i-th component of $\mathbf{x}$ is equal to a non-binary symbol with index ℓ, otherwise $x_{i,\ell} = 0$, for ℓ ranging over all $(q-1)$ non-zero symbols in $GF(q)$. We collect the indicators $\varphi(\mathbf{x})$ for all $\mathbf{x}$ as matrix rows of a $K_j \times m_j(q-1)$ binary matrix. This matrix is referred to as $\mathbf{M}_b^{\mathcal{C}_j}$.

We now consider the N copies of the check node c_j in G_q^N. Let the resultant $(Nm_j,\ N(m_j - 1))$ linear block code be denoted as $\mathcal{C}_j^N$.

Let us represent a codeword of $\mathcal{C}_j^N$ as $\mathbf{x}_N = (x_{1,1} \cdots x_{1,N},\ x_{2,1} \cdots x_{2,N},\ \ldots,\ x_{m_j,1} \cdots x_{m_j,N})$, in which $(x_{i,1} \cdots x_{i,N})$ denotes the value of N variable nodes in G_q^N generated from variable node v_{j_i} in G_q. The frequency row vector $\partial_{j_i} = [d_{i,1} d_{i,2} \cdots d_{i,(q-1)}]$ denotes the number of times each non-zero symbol occurs in $(x_{i,1} \cdots x_{i,N})$, for example $d_{i,3}$ is the number of occurrences of the third non-zero element in $(x_{i,1} \cdots x_{i,N})$.

It is convenient to collect the $1 \times (q-1)$ frequency row vectors $\{\partial_{j_1}, \partial_{j_2}, ..., \partial_{j_{m_j}}\}$ of the N non-binary elements on the edges connected to check node c_j, arising from the incident variable nodes $\{v_{j_1}, v_{j_2}, ..., v_{j_{m_j}}\}$, into the protograph check node frequency weight vector $\mathbf{d}_j = \left[\partial_{j_1}, \partial_{j_2}, ..., \partial_{j_{m_j}}\right]$. As in the U-NBPB case, let $A^{\mathcal{C}_j^N}(\mathbf{d}_j)$ denote the weight-vector enumerator of $\mathcal{C}_j^N$. This enumerator is computed according to the following Theorem.

Theorem 3. *The frequency weight matrix enumerator $A^{\mathcal{C}_j^N}(\mathbf{d}_j)$ of $\mathcal{C}_j^N$ is given by,*

$$A^{\mathcal{C}_j^N}(\mathbf{d}_j) = \sum_{\{\mathbf{n}\}} C\left(N; n_1, n_2, \ldots, n_{K_j}\right), \tag{23}$$

where $C\left(N; n_1, n_2, \ldots, n_{K_j}\right)$ is the multinomial coefficient given by (1) *and $\{\mathbf{n}\}$ is the set of integer-vector solutions to $\mathbf{d}_j = \mathbf{n} \cdot \mathbf{M}_b^{\mathcal{C}_j}$. Here, $n_1, n_2, \ldots, n_{K_j} \geq 0$, and $\sum_{k=1}^{K_j} n_k = N$, and n_k is the number of occurrences of the k^{th} codeword among these N copies of c_j.*

Proof: The proof is based on constructing a multi-dimensional generating function $\{A^{\mathcal{C}^N}(\mathbf{d}_j)\}$ and extracting appropriate coefficients from this generating function using a multinomial theorem. The function itself is derived from the generating function of the underlying code $\mathcal{C}_j$ (induced by the check node c_j, and associated scale collection $\mathbf{s}_j$), multiplied N times. Since the proof uses known techniques previously discussed in the proof of Theorem 1, details are omitted for brevity. ■

Note the contrast between the results in Theorem 1 for U-NBPB codes and Theorem 3 for C-NBPB codes. The former treats edge scalings as random whereas the latter treats edge scalings as fixed.

B. Weight enumerator of the C-NBPB ensemble

To obtain the weight enumerator of the C-NBPB ensemble we need the following definition of the frequency uniform interleaver. The frequency uniform interleaver decouples the frequency weight enumeration of component codes. Note that the symbol interleaver, given in Definition 5 and based on Hamming weight, does not represent a frequency uniform interleaver since now the edge weights are a priori fixed. Recall from (1) that $C\left(N; n_1, n_2, \ldots, n_K\right) = \frac{N!}{n_1! n_2! \ldots n_K!}$ denotes the multinomial coefficient.

Definition 6 (Frequency uniform interleaver)**.** *A length-N frequency uniform interleaver is a probabilistic device that maps*

each input of length N with entries as non-zero symbols of $GF(q)$ and with the frequency weight vector $[d_1, d_2, \ldots, d_{q-1}]$ (each d_t denotes the number of occurrences of the t-th symbol in the input) into the $C(N; d_0, d_1, \ldots, d_{(q-1)})$ distinct output sequences of length N. Here $d_0 = N - \sum_{i>0} d_i$. These outputs have the same frequency weight vector as the input, and they are chosen equiprobably. ■

When $q = 2$, the frequency uniform interleaver is the same as the binary uniform interleaver.

Suppose, as usual, that the scaled protograph G_q has n_v variable nodes and n_c check nodes. As in the U-NBPB case, let m_j denote the degree of the check node c_j. Let t_i denote the degree of the variable node v_i. By construction, the C-NBPB code ensemble consists of all codes obtained by performing all possible edge permutations in the derived graph G_q^N.

Theorem 4. *Let $A^{\mathcal{C}_j^N}(\mathbf{d}_j)$ be the frequency weight matrix enumerator of the code $\mathcal{C}_j^N$ induced by the N copies of the check node c_j with the associated scaling $\mathbf{s}_j$. Then, the frequency weight matrix enumerator of the C-NBPB ensemble is*

$$A(\mathbf{d}) = \frac{\prod_{j=1}^{n_c} A^{\mathcal{C}_j^N}(\mathbf{d}_j)}{\prod_{i=1}^{n_v} C(N;\, d_{i,0},\, d_{i,1},\, \ldots,\, d_{i,(q-1)})^{t_i - 1}} \quad (24)$$

Here, the elements of row vector $\mathbf{d}_j$ comprise a concatenation of column vectors of matrix $\mathbf{d}$, written in the transposed form, so that $\mathbf{d}_j$ is an $m_j(q-1)$ row vector $\left[\partial_{j_1}, \partial_{j_2}, \ldots, \partial_{j_{m_j}}\right]$. In this case, transpose of each $(q-1)$-subvector ∂_{j_i}, $1 \le i \le m_j$ is a column vector of matrix $\mathbf{d}$. In particular, the vector $\mathbf{d}_j = \left[\partial_{j_1}, \partial_{j_2}, ..., \partial_{j_{m_j}}\right]$ describes the frequency weights of the N-symbol words on the edges connected to check node c_j, produced by the variable nodes neighboring c_j.

Proof: Consider a concatenation of two codes, one induced by n_v variable nodes and another induced by n_c check nodes (in the protograph G_q), inter-connected by $|E|$ frequency uniform interleavers, each of length N. Node $v_i \in G_q$ can be treated as a constituent code with one input of frequency weight row vector ∂_i and t_i outputs of frequency weight vectors $[w_{i,1}, w_{i,2}, \ldots, w_{i,t_i}]$. The input-output frequency weight enumerator for node v_i is then

$$C(N; d_{i,0}, d_{i,1}, \ldots, d_{i,(q-1)}) \kappa_{\partial_i, w_{i,1}} \cdots \kappa_{\partial_i, w_{i,t_i}}, \quad (25)$$

where $\partial_i = [d_{i,1}, \ldots, d_{i,(q-1)}]$, and (vector) Kronecker Delta $\kappa_{\mathbf{x},\mathbf{y}}$ is defined in (13).

The N copies of each check node $c_j \in G_q$ can be treated as a constituent code with m_j input frequency weights vectors $\mathbf{w_j} = [w_{j_1}, w_{j_2}, \ldots, w_{j_{m_j}}]$ and no output.

Let $A^{\mathcal{C}_j^N}(\mathbf{w}_j)$ be the input frequency weight enumerator of the check node group $\mathcal{C}_j^N$. Let $A(\mathbf{d})$ represent the number of sequences each with frequency weight matrix $\mathbf{d} = [\partial_1^T, \partial_2^T, \ldots, \partial_{n_v}^T]$ that is applied to the variable nodes according to the protograph constraints. (See also Fig. 15 for illustration.)

Then, the result of Lemma 1 is applied to individual concatenations to obtain the average protograph frequency weight matrix enumerator as,

$$A(\mathbf{d}) = \sum_{\substack{w_{m,u} \\ m=1,\ldots,n_v \\ u=1,\ldots,t_m}} \frac{\prod_{k=1}^{n_v} [C(N; d_{k,0}, d_{k,1}, \ldots, d_{k,(q-1)}) \kappa_{\partial_k, w_{k,1}} \cdots \kappa_{\partial_k, w_{k,t_i}}]}{\prod_{s=1}^{n_v} \prod_{r=1}^{t_s} C(N; w_{s,r,0}, w_{s,r,1}, \ldots, w_{s,r,(q-1)})} \times \prod_{i=1}^{n_c} A^{\mathcal{C}_j^N}(\mathbf{w}_j). \quad (26)$$

Here, the summation is over all frequency weight vectors $w_{m,u}$, where $w_{m,u}$ is the frequency weight vector along the u^{th} edge of variable node v_m. Note that $w_{j,l} = w_{i,k}$ if the l^{th} edge of check node c_j is the k^{th} edge of variable node v_i.The elements of $\mathbf{d}_j$ comprise a subset of column vectors of $\mathbf{d}$. Then (26) reduces to

$$A(\mathbf{d}) = \frac{\prod_{j=1}^{n_c} A^{\mathcal{C}_j^N}(\mathbf{d}_j)}{\prod_{i=1}^{n_v} C(N;\, d_{i,0},\, d_{i,1},\, \ldots,\, d_{i,(q-1)})^{t_i - 1}} \quad (27)$$

as desired. ■

Note that the result in (27) is not merely a consequence of the weight enumerator previously computed for the U-NBPB codes: the former assumes fixed edge scalings while the latter considers all possible non-zero scalings in the edge permutations, thus incurring different combinatorial terms (in the denominator) in the expression for the weight enumerator.

We note that an element z in $GF(q)$ can be expressed as a binary vector $(z_0, \cdots, z_{r-1}) \in \{0,1\}^r$, when $q = 2^r$. Such a binary vector is called the binary image of z. For the given Hamming weight d_B of the binary image, the average number of codewords of *binary Hamming weight* d_B in the C-NBPB ensemble, A_{d_B}, is then simply the sum of $A(\mathbf{d})$ over all $\mathbf{d}$ for which $\sum_{\{d_{i,\ell}: v_i \in V\}} d_{i,\ell} w_\ell = d_B$, where w_ℓ denotes the Hamming of the binary image of non-binary symbol ℓ.

We also note that for a given *symbol Hamming weight* d, the average number of codewords of weight d in the C-NBPB ensemble, A_d, is then simply the sum of $A(\mathbf{d})$ over all $\mathbf{d}$ for which $\sum_{\{d_{i,\ell}: v_i \in V\}} d_{i,\ell} = d$.

Continuing on with the binary image representation, our goal is to choose edge weights so that the minimum distance of the binary image of the code is improved. An approach to improve the minimum distance is to maximize the minimum distance of the binary image of each check node, see e.g., [31] and [39]. We will use this approach later in the paper when we discuss the design of finite-length NBPB codes.

Remark 1. *We also remark that if we average the expression in* (23) *over all possible non-zero scales and use it in* (27)*, we then obtain the frequency weight matrix enumerator for the U-NBPB code, denoted by $\bar{A}(\mathbf{d})$. This expression can in turn be used to compute the weight enumerators for the binary image of a U-NBPB code. For the given Hamming weight d_B of the binary image, the average number of codewords of binary weight d_B in the U-NBPB ensemble, $\bar{A}_{d_B}$, is then simply the sum of $\bar{A}(\mathbf{d})$ over all $\mathbf{d}$ for which $\sum_{\{d_{i,\ell}: v_i \in V\}} d_{i,\ell} w_\ell = d_B$, where w_ℓ denotes the Hamming weight of the binary image of a non-binary symbol ℓ.*

Example 6. *Let us consider the binary image of the regular-$(2,4)$ code in Fig. 3(a), now defined over $GF(16)$ and with $N = 4$, as a C-NBPB code. We evaluate C-NBPB ensemble enumerators for different choices of edge weights, by considering two randomly chosen assignments described by the edge*

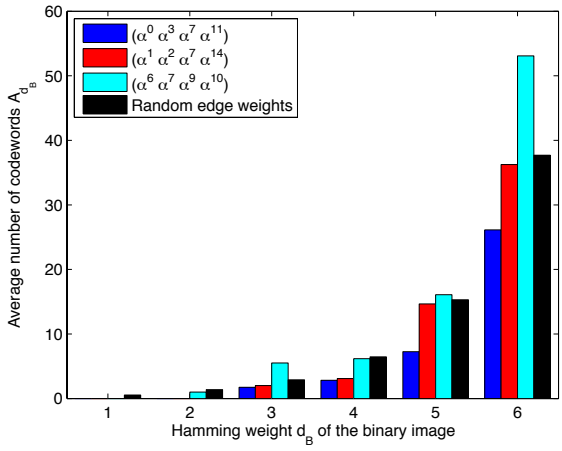


Fig. 11. Weight enumerator for the binary image of various C-NBPB ensembles and for the random edge weight assignment.

vectors $(\alpha^6, \alpha^7, \alpha^9, \alpha^{10})$ *and* $(\alpha^1, \alpha^2, \alpha^7, \alpha^{14})$ *(read top to bottom in the panel) for* α *a primitive element over* $GF(16)$*, and a root of* $x^4 + x + 1$*. We also evaluate the ensemble enumerator for* $(\alpha^0, \alpha^3, \alpha^7, \alpha^{11})$*, an edge weight choice that was proposed in [39] as a good choice for edge weights. Indeed, Fig. 11 shows that the code described with edge weights as proposed in [39] has fewer low weight codewords than the two randomly chosen edge weight assignments for the C-NBPB construction. We also plot the ensemble enumerator under randomly assigned edge scalings in Fig. 11. As we can see, with a good edge weight assignment, the C-NBPB ensemble has fewer low weight codewords than the random ensemble.*

C. Asymptotic ensemble weight enumerators

Equipped with the new weight enumerator, for the Galois field size $q = 2^r$, the asymptotic growth rate can now be derived in the usual sense, either in terms of the number of protograph copies, N,

$$r_B(\delta) = \limsup_{N\to\infty} \frac{\ln A_{d_B}}{N} = \limsup_{N\to\infty} \frac{\ln A_{\delta N}}{N}, \qquad (28)$$

or in terms of the codeword bit length n (where $n = r \cdot n_v \cdot N$),

$$\tilde{r}_B(\tilde{\delta}) = \limsup_{n\to\infty} \frac{\ln A_{d_B}}{n} \qquad (29)$$

where $\tilde{r}_B(\tilde{\delta}) = \frac{1}{rn_v} r_B(\tilde{\delta} r n_v)$. From (27), it follows that,

$$\ln A(\mathbf{d}) = \sum_{j=1}^{n_c} \ln A^{\mathcal{C}_j^N}(\mathbf{d}_j) - \sum_{i=1}^{n_v} (t_i - 1) \ln C(N; d_{i,0}, \ldots, d_{i,(q-1)}), \qquad (30)$$

and, with N tending to infinity,

$$r_B(\delta) = \max_{\{\boldsymbol{\delta}\}} \Big\{ \sum_{j=1}^{n_c} a^{\mathcal{C}_j^\infty}(\boldsymbol{\delta}_j) - \sum_{i=1}^{n_v} (t_i - 1) H\left([\delta_{i,0}, \delta_{i,1}, \ldots, \delta_{i,(q-1)}]\right) \Big\}, \qquad (31)$$

under the constraint $\sum_{\{\delta_{i,\ell}: d_i \in V\}} \delta_{i,\ell} w_\ell = \delta$. Here, $\boldsymbol{\delta} = \mathbf{d}/N$, $\boldsymbol{\delta}_j = \mathbf{d}_j/N$, $\delta_{i,\ell} = d_{i,\ell}/N$, $\tilde{\delta} = \delta/rn_v = d_B/n$, and $H(\cdot, \cdots, \cdot)$ is the multi-dimensional entropy function. The term $a^{\mathcal{C}_j^\infty}(\boldsymbol{\delta}_j)$ stands for the asymptotic frequency weight matrix enumerator of the check node c_j, and it is computed as $a^{\mathcal{C}_j^\infty}(\boldsymbol{\delta}_j) = \max_{\{P_{\boldsymbol{\delta}_j}\}} \{H(P_{\boldsymbol{\delta}_j})\}$. The collection $\{P_{\boldsymbol{\delta}_j}\}$ represents the set of solutions to $\boldsymbol{\delta}_j = P_{\boldsymbol{\delta}_j} \cdot M_b^{\mathcal{C}_j}$, with $P_{\boldsymbol{\delta}_j} = [p_1, p_2, \ldots, p_K]$, $p_1, p_2, \ldots, p_K \geq 0$ and $\sum_{k=1}^K p_k = 1$.

The asymptotic C-NBPB weight enumerators with the same protograph and edge assignments as for the parameters in Example 6 are also calculated. Simulation results show that $r_B(\delta)$ of C-NBPB protographs with good edge scaling assignment and randomly chosen edge scaling assignments are approximately the same.

D. Comparison of computational complexity of U-NBPB and C-NBPB enumerators

Lastly, we compare the computational complexity of enumerators induced by a simple linear code with the single check node for the U-NBPB and the C-NBPB cases.

Example 7. *Assume that a single-parity check code is defined over* $GF(16)$ *with a degree-4 check node* c_j *(i.e.,* $m_j = 4$*). We consider the enumerators of the induced U-NBPB and C-NBPB ensembles in both the finite-length and in the infinite-length regime. In particular, for the finite case, we assume that the single-parity check code is repeated* N *times. (Since we are dealing with one check node, the subscript* $i = 1$ *in* $\delta_{i,\ell}$ *is suppressed.)*

(a) For the U-NBPB case, one computes $\mathbf{w} = \mathbf{n} \cdot \mathbf{M}_{b,r}^{\mathcal{C}_j}$*. The dimension of the matrix* $\mathbf{M}_{b,r}^{\mathcal{C}_j}$ *is* $K_{j,r} \times m_j$*, where* $K_{j,r} = 1 + \sum_{i=2}^{m_j} \binom{m_j}{i} = 12$*, and* $\mathcal{C}_j$ *denotes our single parity-check code. For the finite case, there are* $\binom{N+K_{j,r}-1}{K_{j,r}-1} = \binom{N+11}{11}$ *possible choices for* $\mathbf{n}$*. All possible* $\mathbf{n}$*'s need to be identified to get the symbol weight enumerator for all weights. For the asymptotic case, with fixed* δ*, we need to find the length-4 vector (since* $m_j = 4$*)* $\omega = [\delta_1, \delta_2, \delta_3, \delta_4]$ *under the constraint* $\sum_{\ell=1}^{m_j} \delta_\ell = \delta$ *and* $\delta_\ell \geq 0$ *that maximizes* $a^{c_j}(\omega)$*. This is a search in a 4-dimensional space.*

(b) For the C-NBPB case, the dimension of the matrix $\mathbf{M}_b^{\mathcal{C}_j}$ *is* $K_j \times m_j(q-1)$*, where* $K_j = q^{(m_j-1)} = 16^3 = 4096$ *and* $m_j(q-1) = 60$*. For the finite case, there are* $\binom{N+K_j-1}{K_j-1} = \binom{N+4095}{4095}$ *possible choices for* $\mathbf{n}$*. The total number of possible choices for* $\mathbf{n}$ *is considerably larger in the C-NBPB case than in the U-NBPB case, thus clearly making the overall enumeration much more involved.*

For the asymptotic case, with fixed δ*, we need to find a vector of length* $m_j(q-1) = 60$*, call it* $\boldsymbol{\delta_j} = [\delta_1, \delta_2, \ldots, \delta_{59}, \delta_{60}]$*, that maximizes* $a^{\mathcal{C}_j^\infty}(\boldsymbol{\delta}_j)$ *under the constraint* $\boldsymbol{\delta}_j = P_{\boldsymbol{\delta}_j} \cdot \mathbf{M}_b^{\mathcal{C}_j}$*, with* $P_{\boldsymbol{\delta}_j} = [p_1, p_2, \ldots, p_{K_j}]$*,* $p_1, p_2, \ldots, p_{K_j} \geq 0$ *and* $\sum_{k=1}^{K_j} p_k = 1$*. This is a search in a 60-dimensional space. Obviously, since the dimension of the search space is much higher than in the U-NBPB case, the overall computational complexity is also much higher.*

V. Pseudocodeword, trapping set, and stopping set enumerators

In this section we discuss how the weight enumeration techniques from the previous section can be extended to enumerate certain graphical objects of interest such as trapping sets, stopping sets, and pseudocodewords. We start off with the definitions of non-binary trapping sets, stopping sets and pseudocodewords (Subsection V-A) followed by trapping set enumerators of the U-NBPB and C-NBPB code ensembles (Subsection V-B), stopping set enumerators (Subsection V-C) and the pseudocodeword analysis of the U-NBPB codes (Subsection V-D).

A. Non-binary quantities of interest

Let $G_q = (V, C, E, S_q)$ be the Tanner graph of an LDPC code defined over $GF(q)$, with V and C denoting the variable node set, and the check node set, respectively, and E and S_q denoting the edge set and the associated scalings, respectively. Also, let $|V| = n$.

Definition 7 (Trapping set in $GF(q)$)**.** *For the graph $G_q = (V, C, E, S_q)$, an (a, b) trapping set $T_{a,b}$ is a subgraph of G_q if all a variable nodes in $T_{a,b}$ have values in $GF(q) \setminus 0$, all other variable nodes in V are with value 0, and all of b check nodes in $T_{a,b}$ are unsatisfied.* ■

Note that in the definition of non-binary trapping sets, the number b of unsatisfied checks depends on the input values and the edge weights. That is, two subgraphs with the same topology (and thus the same a) may result in different b depending on the choice of the symbol values on the a variable nodes and depending on the choice of non-zero weights assigned to the edges of the subgraph. We remark that, in the binary case, the value of b is uniquely determined since all edge weights are equal to 1 and all of the a variable nodes have value 1. In contrast to trapping sets, the definition of stopping sets only depends on the topology of the subgraph and therefore is the same as in the binary case.

Definition 8 (Stopping set in $GF(q)$)**.** *For the graph $G_q = (V, C, E, S_q)$, an (a, b) stopping set $S_{a,b}$ is a subgraph of G_q induced by a variable nodes in V, such that there are b check nodes in the induced subgraph, and such that in this subgraph every check node has at least two neighboring variable nodes.* ■

We now turn our attention to pseudocodewords. For M a positive integer, a degree-M *cover* of G_q is a Tanner graph $G_q^{(M)}$ that results from replicating M times each node of G_q, followed by introducing edges in a way that the local adjacency is preserved among the replicated nodes (*cf.* [28]).

As an illustration, the resultant graph in Figure 1(b) can be viewed as a construction of a degree-3 cover of the original protograph of 1(b). For simplicity, permutation matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ was used for each group of $M = 3$ replicated edges in the graph cover.

We let $\mathcal{C}$ denote the code generated by G_q. We let $\hat{\mathbf{c}}_M = (c_{1,1} \cdots c_{1,M},\ c_{2,1} \cdots c_{2,M},\ \ldots,\ c_{n,1} \cdots c_{n,M})$ be an M-cover codeword of $\mathcal{C}^{(M)}$, the code generated by $G_q^{(M)}$. Analogously to the codeword weight enumerator where one is concerned with the number of non-zero symbols per codeword (and not with their exact location), when enumerating the pseudocodewords one keeps track of the frequency of occurrence of each non-zero symbol in each variable of the underlying graph. This observation is the motivation for the following definition of pseudocodeword matrix.

Suppose that the distinct elements of $GF(q)$ form the set $\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$ for α a primitive element of $GF(q)$.

Definition 9 (Pseudocodewords in GF(q))**.** *Following the notation in [45], we let $P = P(\hat{\mathbf{c}}_M)$ be the $(q-1) \times n$ matrix where the entry $P(i, j)$ represents the number of occurrences of symbol α^{i-1} in positions $c_{j,k}$ $1 \le k \le M$ in $\hat{\mathbf{c}}_M$, computed for each i between 1 and $q-1$, and each j between 1 and n. The number of 0 elements then follows from subtracting the total count of non-zero elements of $GF(q)$ from M. Matrix P is called the degree-M pseudocodeword matrix. As a shorthand, P is then referred to as the pseudocodeword.*

Matrix P can be viewed as a concatenation of column vectors, each of length $(q-1)$ that indicate the number (or frequency, when these vectors are normalized) of times each symbol occurs in a particular variable node. We call these $(q-1)$ dimensional vectors *pseudo symbols*.

B. Ensemble trapping set enumerators

In this section we consider the trapping set enumerators of the U-NBPB and C-NBPB ensembles, starting with the former.

1) Trapping set enumerators for a U-NBPB ensemble: Let us consider a $T_{a,b}$ trapping set in a Tanner graph $\tilde{G}_q^N = (V, C, E, S_q)$ specifying a U-NBPB code over $GF(q)$ which is obtained by copying the underlying protograph G N times followed by permuting and scaling. of these a variable nodes to (arbitrary) non-zero elements of $GF(q)$ and set the values of all remaining variable nodes to the zero element of $GF(q)$, so that b neighboring check nodes are unsatisfied. We then attach additional b variable nodes, one to each of these b check nodes in the graph $\tilde{G}_q^N$. The attached nodes are connected via new edges of weight 1 each, and have a non-zero value uniquely chosen to force all check nodes to be satisfied. This suggests to add degree-1 variable nodes to all check nodes in the underlying protograph G. Let this set of nodes be F and call the new graph G'. We can then obtain the trapping set enumerator of the U-NBPB ensemble specified by G from the weight enumerator of the U-NBPB ensemble specified by G'.

In particular, the $T_{a,b}$ trapping set enumerator $A_{a,b}^{(t)}$ is computed as

$$A_{a,b}^{(t)} = \sum_{\{d_i : v_i \in V\}} \sum_{\{d_k : v_k \in F\}} A(\mathbf{d}), \tag{32}$$

under the constraints $\sum_{\{d_i : v_i \in V\}} d_i = a$ and $\sum_{\{d_i : v_i \in F\}} d_i = b$, where

$$A(\mathbf{d}) = \frac{\prod_{j=1}^{n_c} A^{\mathcal{C}'^N_j}(\mathbf{d}_j)}{\prod_{i=1}^{n_v} (q-1)^{d_i(t_i-1)} \binom{N}{d_i}^{t_i-1}}\ . \tag{33}$$

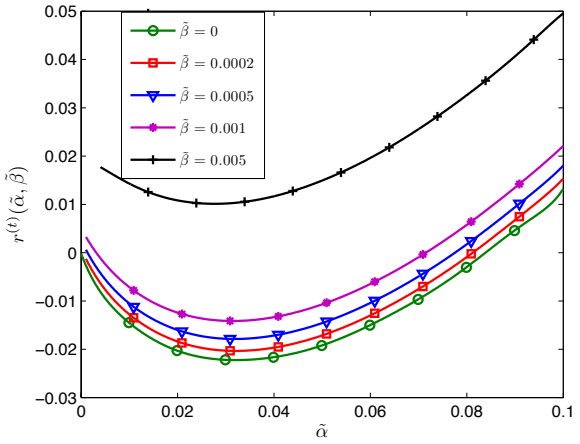


Fig. 12. Asymptotic trapping set enumerators of the regular $(3, 6)$ protograph code ensemble over $GF(16)$.

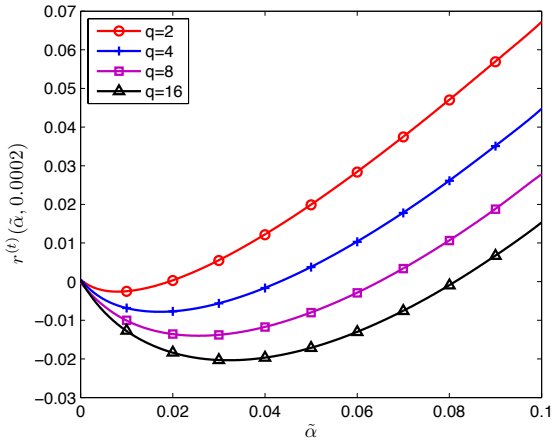


Fig. 13. Asymptotic trapping set enumerators of the regular-$(3, 6)$ protograph code ensemble for different q with $\tilde{\beta} = 0.0002$, for protograph shown in Fig. 8.

We use ${\mathcal{C}'}_j^N$ instead of $\mathcal{C}_j^N$ in (33) to indicate that the weight-vector enumerators in (33) are obtained from the check nodes in G'. These weight-vector enumerators can be evaluated using (2).

As in Section III-C, we define the normalized logarithmic asymptotic trapping set enumerator $\tilde{r}^{(t)}(\tilde{\alpha}, \tilde{\beta})$, as

$$\tilde{r}^{(t)}(\tilde{\alpha}, \tilde{\beta}) = \limsup_{n\to\infty} \frac{\ln A_{a,b}^{(t)}}{n}, \tag{34}$$

where $\tilde{\alpha} = a/n$ and $\tilde{\beta} = b/n$ (recall $n = n_v \cdot N$ is the code length). The derivation of an expression for (34) from (33) uses the same steps used in deriving $\tilde{r}(\tilde{\delta})$, and yields

$$\tilde{r}^{(t)}(\tilde{\alpha}, \tilde{\beta}) = \frac{1}{n_v} r^{(t)}(\tilde{\alpha} n_v, \tilde{\beta} n_v), \tag{35}$$

where

$$r^{(t)}(\alpha, \beta) = \max_{\{\delta_l : v_l \in V\}} \{ \max_{\{\delta_k : v_k \in F\}} \{ \sum_{j=1}^{n_c} a^{c'_j}(\delta_j) - \sum_{i=1}^{n_v} H_q(\delta_i) \} \}, \tag{36}$$

under the constraints $\sum_{\{\delta_i : v_i \in V\}} \delta_i = \alpha$, and $\sum_{\{\delta_i : v_i \in F\}} \delta_i = \beta$. The asymptotic weight-vector enumerator, $a^{c'_j}(\delta_j)$, can be evaluated using (21).

Example 8. *Let us consider the regular $(3, 6)$ protograph code ensemble over $GF(16)$. The asymptotic trapping set enumerators are plotted for different $\tilde{\beta}$ in Fig. 12. Note when $\tilde{\beta} = 0$, by our definition, the curve corresponds to the asymptotic symbol weight enumerator of the regular $(3, 6)$ protograph. In the figure, when $\tilde{\alpha}$ is fixed, $\tilde{r}^{(t)}(\tilde{\alpha}, \tilde{\beta})$ increases with increasing $\tilde{\beta}$. This result is consistent with the trapping set enumerator for binary protograph-based LDPC codes reported in [2].*

Example 9. *In this example, we consider the regular $(3, 6)$ protograph code ensemble for different q's with fixed $\tilde{\beta} = 0.0002$. The asymptotic enumeration results are shown in Fig. 13. In the figure, we can see that when $\tilde{\beta} = 0.0002$, there always exists the second zero-crossing, i.e., there exist the typical relative $\tilde{r}^{(t)}(\tilde{\alpha}, 0.0002)$ smallest trapping sets for different q's. Also, when $\tilde{\alpha}$ is fixed, $\tilde{r}^{(t)}(\tilde{\alpha}, 0.0002)$ decreases as q increases. This indicates that for $\tilde{\beta} = 0.0002$ (and more generally), codes over larger q have fewer trapping sets.*

2) Trapping set enumerators for a C-NBPB ensemble: As in the U-NBPB case, let us enlarge the original scaled protograph G_q with V denoting the set of its variable nodes by additional degree-1 variable nodes (call this set F) to obtain a new scaled protograph G'_q. Let us again denote by F the set of additional degree-1 variable nodes in the resultant graph. Lastly, as in the trapping set analysis of U-NBPB codes, it suffices now to consider the weight enumerator of the C-NBPB ensemble specified by G'_q when calculating the trapping set enumerator of the C-NBPB ensemble specified by G_q.

Based on the results in Section IV, it follows that the trapping set enumerator $A_{a,b}^{(t)}$ can be computed as

$$A_{a,b}^{(t)} = \sum_{\{d_{i,\ell} : v_i \in V\}} \sum_{\{d_{k,\ell} : v_k \in F\}} A(\mathbf{d}), \tag{37}$$

under the constraints $\sum_{\{d_{i,\ell} : v_i \in V\}} d_{i,\ell} = a$ and $\sum_{\{d_{i,\ell} : v_i \in F\}} d_{i,\ell} = b$, where

$$A(\mathbf{d}) = \frac{\prod_{j=1}^{n_c} A^{{\mathcal{C}'}_j^N}(\mathbf{d}_j)}{\prod_{i=1}^{n_v} C(N;\, d_{i,0},\, d_{i,1},\, \ldots,\, d_{i,(q-1)})^{t_i - 1}}. \tag{38}$$

Note that here $A^{{\mathcal{C}'}_j^N}$ refers to the weight-vector check node enumerators for the check nodes in G'_q. These weight-vector enumerators are readily evaluated using (23). The growth rate $r^{(t)}(\alpha, \beta)$ can now be computed in a similar way as (31).

C. Ensemble stopping set enumerators

We first recall that, in contrast to trapping sets, the definition of stopping sets is purely topological, that is we seek structure $S_{a,b}$ with a variable nodes and b check nodes such that each check node has more than one connection to the subset of variable nodes. This constraint, in particular, does not depend on edge scaling. As a result, the problem of enumerating non-binary stopping sets can be simply recast as the problem of

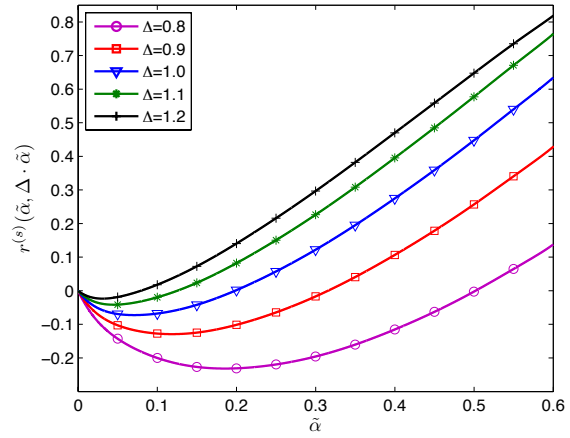


Fig. 14. Asymptotic stopping set enumerators of regular $(3, 6)$ protograph.

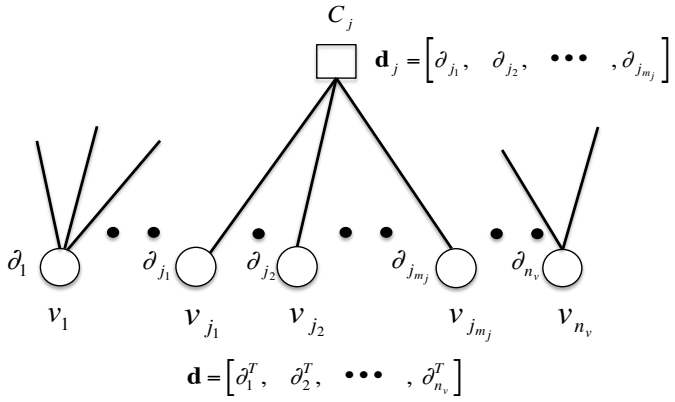


Fig. 15. Illustration of the relation between ∂_i and ∂_{j_i}.

enumerating binary stopping sets. These in turn are enumerated via a weight enumerator of a suitably enlarged graph as in [2]. Let us define $A_{a,b}^{(s)}$ as the average number of $S_{a,b}$ stopping sets of a given ensemble. Similar to the analysis in Section III-C, define the normalized logarithmic asymptotic stopping set enumerator $\tilde{r}^{(s)}(\tilde{\alpha}, \tilde{\beta})$, as

$$\tilde{r}^{(s)}(\tilde{\alpha}, \tilde{\beta}) = \limsup_{n\to\infty} \frac{\ln A_{a,b}^{(s)}}{n}, \tag{39}$$

where $\tilde{\alpha} = a/n$ and $\tilde{\beta} = b/n$.

Example 10. *Let us consider the regular* $(3, 6)$ *protograph in Fig. 8 again. In Fig. 14,* $\tilde{r}^{(s)}(\tilde{\alpha}, \Delta \cdot \tilde{\alpha})$ *is evaluated for several values of* Δ*. For the regular* $(3, 6)$ *protograph, each variable node is connected to three check nodes and each check node is connected to six variable nodes. Thus for* $\tilde{\alpha}$ *variable nodes, there are* $3\tilde{\alpha}$ *edges connected to these variable nodes and* $\frac{3\tilde{\alpha}}{6} \leq \tilde{\beta} \leq \frac{3\tilde{\alpha}}{2}$*, i.e.* $0.5 \leq \Delta \leq 1.5$*. From Fig. 14, we can see that for fixed* $\tilde{\alpha}$*, there tends to be more stopping sets with larger* Δ*.*

D. Ensemble U-NBPB pseudocodeword enumerators

In this section we describe the pseudocodewords arising from the graph covers of U-NBPB codes.

Let us consider a single variable node v_i in the protograph G having n_v variable nodes. Let $\tilde{G}_q^N$ be the graph obtained by copying the graph G N times, followed by edge scaling and permutation. As before, the result of replicating node v_i N times can be viewed as a single constituent code.

Following the notation in Section V-A (with $\tilde{G}_q^N$ playing the role of G_q), we now investigate the degree-M cover of $\tilde{G}_q^N$. One computes the distributions of the pseudocodewords for the constituent code induced by node v_i. Each column vector of length $(q-1)$, P_k, $1 \leq k \leq N$ in the $P = P(v_i)$ matrix of dimension $(q-1) \times N$, represents a pseudo symbol in the degree-M cover of $\tilde{G}_q^N$ corresponding to v_i in G. Note that each entry in the vector P_k, $1 \leq k \leq N$, is an integer between 0 and M. Let M' denote the total number of distinct non-zero pseudo symbols , where each distinct pseudo symbol $\mathbf{f}_\ell = [f_{1,\ell}, f_{2,\ell}, \cdots, f_{(q-1),\ell}]^T$ is a $(q-1)$-vector.

The number M' of possible distinct non-zero pseudo symbols is $M' = \binom{M+q-1}{q-1} - 1$, since each pseudo symbol has $(q-1)$ entries, and considering the count of '0' elements as discussed above, we have q non-negative integers that sum to M. Thus $M'+1$ is just the number of possible partitions of M into q.

It is helpful to express these pseudo symbol vectors via a distribution: let $d_{i,\ell}$ denote the total number of occurrences of the distinct pseudo symbol $\mathbf{f}_\ell$ in pseudocodeword P so that $\sum_{\ell=0}^{M'} d_{i,\ell} = N$, and define distribution row vector $\partial_i = [d_{i,1}, d_{i,2}, \ldots, d_{i,M'}]$ as the pseudoweight vector associated with v_i. Define the matrix of distributions for all n_v variable nodes as $\mathbf{d} = [\partial_1^T\ \partial_2^T\ \ldots\ \partial_{n_v}^T]$.

The relative effect that each pseudo symbol has on the overall performance is a function of its *pseudoweight*, that itself depends on the channel. A representative evaluation of the pseudoweight (*cf.* [27]) for the AWGN channel and the q-ary PAM is:

$$d_{AWGN}(P(\hat{\mathbf{c}}_M)) = \frac{(\sum_{\ell=1}^{M'} d_{i,\ell} \sum_{k=1}^{q-1} f_{k,l} \times k^2)^2}{\sum_{\ell=1}^{M'} d_{i,\ell} (\sum_{k=1}^{q-1} f_{k,l} \times k)^2}. \tag{40}$$

Now, let us consider a particular check node c_j in G of degree m_j. For check node c_j, $\mathcal{P}_{c_j}$ represents the set of pseudocodewords of degree-M cover of the check node c_j. Analogous to the definition of ∂_i for variable node v_i, we define distribution row vector $\partial_{j_k} = [d_{j,k_1} d_{j,k_2} \ldots d_{j,k_{M'}}]$, associated with the k-th edge of check node c_j for $1 \leq k \leq m_j$. Please see Fig. 15 for clarification of notation. Similarly we define a vector of distributions $\mathbf{d}_j = [\partial_{j_1}\ \partial_{j_2}\ \ldots\ \partial_{j_{m_j}}]$ for all m_j edges of check node c_j.

Let $A^{\mathcal{P}_{c_j}^N}(\mathbf{d}_j)$ be the pseudoweight vector enumerator associated with N copies of the check node c_j. This vector enumerator is given by

$$A^{\mathcal{P}_{c_j}^N}(\mathbf{d}_j) = \sum_{\{\mathbf{n}\}} C(N;\, n_1,\, n_2, \ldots,\, n_K), \tag{41}$$

where the sum is over all realizable pseudocodeword weight count configurations each described by the vector $\mathbf{n}$ =

$[n_1\ n_2\ \ldots\ n_K]$, and where K represents the total number of pseudocodewords of the check node c_j. Thus, $\{\mathbf{n}\}$ is the set of integer solutions to $\mathbf{d}_j = \mathbf{n}\cdot\mathbf{M}^{\mathcal{P}_{c_j}}$ with $n_1,\ n_2,\ldots,\ n_K > 0$ and $\sum_{k=1}^{K} n_k = N$, and $\mathbf{M}^{\mathcal{P}_{c_j}}$ is the binary matrix whose rows are obtained as follows. Each row is related to a given $(q-1)\times m_j$ pseudocodeword $P \in \mathcal{P}_{c_j}$ by the nonbinary-to-binary mapping φ defined as $\varphi(P) = [x_{1,1}\ldots x_{1,M'},\ x_{2,1}\ldots x_{2,M'},\ \ldots,\ x_{m_j,1}\ldots x_{m_j,M'}]$, where $x_{i,l} = 1$, if the i-th column of P is equal to $\mathbf{f}_\ell$, otherwise $x_{i,l} = 0$. Note that $\varphi(P)$ can be viewed as an indicator function in the sense that the location of a 1 in $\varphi(P)$ indicates the presence of the corresponding value in P.

Combining the constraints for the check nodes and for the variable nodes, and viewing them as concatenated codes the formula for the ensemble average is given by

$$A^{(p)}(\mathbf{d}) = \frac{\prod_{j=1}^{n_c} \bar{A}^{\mathcal{P}^N_{c_j}}(\mathbf{d}_j)}{\prod_{i=1}^{n_v} C(N;\ d_{i,0},\ d_{i,1},\ \ldots,\ d_{i,M'})^{t_i-1}}, \quad (42)$$

where $\bar{A}^{\mathcal{P}^N_{c_j}}(\mathbf{d}_j)$ is the pseudocodeword enumerator of the check node c_j averaged over all possible scales on those edges connected to this check node. The vector $\mathbf{d}_j$ collects distribution of pseudocodeword symbols over all width-N input variables adjacent to c_j.

We now introduce the binary matrix $\mathbf{M}_{\mathbf{s}}^{\mathcal{P}_{c_j}}$ as the counterpart of $\mathbf{M}^{\mathcal{P}_{c_j}}$ which now also depends on the input scalings $\mathbf{s}$, with $\mathbf{s}$ being the vector representation of scalings s_k on the edges incident to c_j. Again, for a given check node c_j, each row of this binary matrix $\mathbf{M}_{\mathbf{s}}^{\mathcal{P}_{c_j}}$ is related to a $(q-1)\times m$ pseudocodeword P, $P \in \mathcal{P}_{c_j}$ by the same nonbinary-to-binary mapping $\varphi(P)$. For a fixed $\mathbf{s}$, $\mathbf{M}_{\mathbf{s}}^{\mathcal{P}_{c_j}}$ is the same for all degree-M covers.

The generating function of the code $\mathcal{C}$ induced by the M-fold cover of a check node c is $\sum_{\mathbf{x}_i\in\mathbf{M}_{\mathbf{s}}^{\mathcal{P}_c}} \prod_{i=1}^{m} W_{i,1}^{x_{i,1}} W_{i,2}^{x_{i,2}} \cdots W_{i,M'}^{x_{i,M'}}$, where the $W_{i,j}$'s are indeterminate bookkeeping variables, $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,M'}]$, and m is the degree of c. The generating function for the N copies of this check node in the resultant graph is then

$$A^{\mathcal{P}^N_c}(W_{1,1}, W_{1,2}, \ldots, W_{m,M'}) = \prod_{k=1}^{N} \left(\sum_{\mathbf{x}\in\mathbf{M}_{\mathbf{s_k}}^{\mathcal{P}_c}} \prod_{i=1}^{m} W_{i,1}^{x_{i,1}} W_{i,2}^{x_{i,2}} \cdots W_{i,M'}^{x_{i,M'}} \right). \quad (43)$$

Since all edge labels are i.i.d. then

$$\begin{aligned} \bar{A}^{\mathcal{P}^N_c}(W_{1,1}, W_{1,2}, \ldots, W_{m,M'}) &= \left(\mathbb{E}\left[\sum_{\mathbf{x}\in\mathbf{M}_{\mathbf{s_k}}^{\mathcal{P}_c}} \prod_{i=1}^{m} W_{i,1}^{x_{i,1}} W_{i,2}^{x_{i,2}} \cdots W_{i,M'}^{x_{i,M'}} \right] \right)^N \\ &= \left(\sum_{\mathbf{x}\in\mathbf{M}^{\mathcal{P}_c}} h(\mathbf{x}) \prod_{i=1}^{m} W_{i,1}^{x_{i,1}} W_{i,2}^{x_{i,2}} \cdots W_{i,M'}^{x_{i,M'}} \right)^N, \end{aligned} \quad (44)$$

where $\mathbf{M}^{\mathcal{P}_c}$ now includes distinct pseudocodewords of all $\mathbf{M}_{\mathbf{s_k}}^{\mathcal{P}_c}$'s. Note that in going from $\mathbf{M}_{\mathbf{s_k}}^{\mathcal{P}_c}$'s to $\mathbf{M}^{\mathcal{P}_c}$, $h(\mathbf{x})$ accounts for the normalized frequency of occurrence of $\mathbf{x}$ in the underlying graph cover ranging over all $\mathbf{s}$.

Applying the multinomial theorem, we can write

$$\begin{aligned} \bar{A}^{\mathcal{P}^N_c}(W_{1,1}, W_{1,2}, \ldots, W_{m,M'}) &= \sum_{\substack{n_1,n_2,\ldots,n_K\geq 0 \\ n_1+n_2+\cdots+n_K=N}} C(N; n_1, n_2, \ldots, n_K) \\ &\quad \times \prod_{\mathbf{x}\in\mathbf{M}^{\mathcal{P}_c}} \left(h(\mathbf{x}) \prod_{i=1}^{m} W_{i,1}^{x_{i,1}} W_{i,2}^{x_{i,2}} \cdots W_{i,M'}^{x_{i,M'}} \right)^{n_k} \\ &= \sum_{\mathbf{d}} \sum_{\{\mathbf{n}\}} C(N; n_1, n_2, \ldots, n_K) \\ &\quad \times \exp\{\mathbf{n}\cdot\mathbf{b}^T\} \prod_{i=1}^{m} W_{i,1}^{d_{i,1}} W_{i,2}^{d_{i,2}} \cdots W_{i,M'}^{d_{i,M'}}, \end{aligned} \quad (45)$$

where $\mathbf{b} = [b_1\ b_2\ \ldots\ b_K]$ and $b_k = \ln(h(\mathbf{x}_k))$. From (45), $\bar{A}^{\mathcal{P}^N_{c_j}}(\mathbf{d}_j) = \sum_{\{\mathbf{n}\}} C(N; n_1, n_2, \ldots, n_K) \times \exp\{\mathbf{n}\cdot\mathbf{b}^T\}$, where $\mathbf{n} = [n_1\ n_2\ \ldots\ n_K]$, and $\{\mathbf{n}\}$ is the set of integer solutions to $\mathbf{d}_j = \mathbf{n}\cdot\mathbf{M}^{\mathcal{P}_{c_j}}$ with $n_1,\ n_2,\ldots,\ n_K > 0$ and $\sum_{k=1}^{K} n_k = N$. Lastly, the average pseudo-weight enumerator can be computed as

$$A_d^{(p)} = \sum_{\{\mathbf{d}\}} A^{(p)}(\mathbf{d}), \quad (46)$$

where the sum ranges over all matrices $\mathbf{d}$ whose pseudocodeword weight is the channel dependent parameter d. For example, under the channel-dependent constraints provided by the AWGN channel and the q-ary PAM, pseudocodeword weight (40) becomes

$$d = \frac{\left(\sum_{i=1}^{n_v} \sum_{l=1}^{M'} d_{i,l} \sum_{k=1}^{q-1} f_{k,l}\cdot k^2 \right)^2}{\sum_{i=1}^{n_v} \sum_{l=1}^{M'} d_{i,l} \left(\sum_{k=1}^{q-1} f_{k,l}\cdot k \right)^2}. \quad (47)$$

Remark 2. *We quickly remark that using the expression in* (42) *for degree-1 cover ($M = 1$) also provides the frequency weight matrix enumerator for a U-NBPB code, denoted by $\bar{A}(\mathbf{d})$ (see also Remark 1). This enumerator can also be used to obtain binary Hamming weight enumerators for the binary image of U-NBPB codes.*

Remark 3. *We also note that the pseudocodeword enumeration viz. the M-th cover of a C-NBPB code ensemble (with, say, the N-fold copy operation) is similar to the derivation of pseudo codewords of U-NBPB code ensemble (with $N\times M$ copy operation) but now no averaging over scales is required. Details are omitted for brevity.*

The enumeration methods are illustrated with representative examples.

Example 11. *Consider the $(3, 2)$ NB protograph code shown in Fig. 16 over $GF(3)$. Suppose that the base protograph is replicated $N = 2$ times. We seek to compute the ensemble*

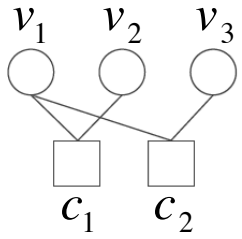


Fig. 16. $(3, 2)$ NB protograph code.

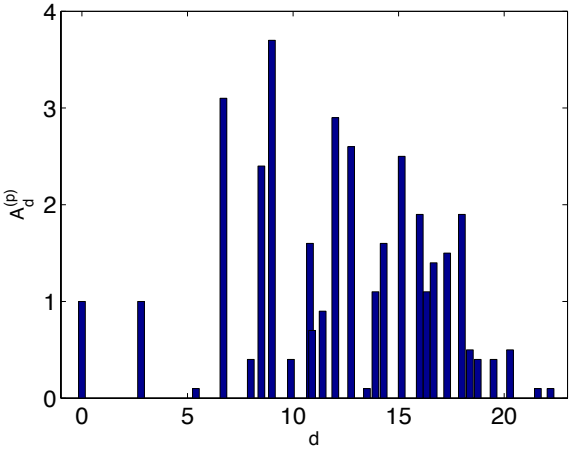


Fig. 17. Pseudo-codeword PAM distance spectrum for protograph in Fig. 16.

pseudo-weight enumerator for the resultant U-NBPB code for the graph cover degree $M = 2$. The number of distinct non-zero pseudocodeword symbols is 5, i.e. $M' = 5$, and the set of all pseudo symbols is $\left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right\}$.

For either check node c_1 or c_2 (both being degree-2 check nodes), the codewords are $\{00, 12, 21\}$, or $\{00, 11, 22\}$ depending on the assigned non-zero scales. In the degree-2 cover of such a check, there are 4 sets: two sets with pseudocodewords $\mathcal{P}_{c_j} = \left\{ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \right\}$, *and two other sets with pseudocodewords* $\mathcal{P}_{c_j} = \left\{ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 2 & 2 \end{bmatrix} \right\}$, *for $j = 1, 2$. The matrix $\mathbf{M}^{\mathcal{P}_{c_j}}$ is obtained by averaging these two sets. The set of pseudocodewords to be considered in the construction of matrix $\mathbf{M}^{\mathcal{P}_{c_j}}$, is* $\left\{ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 2 & 2 \end{bmatrix} \right\}$. *The matrix $\mathbf{M}^{\mathcal{P}_{c_j}}$ is then*

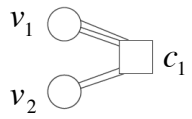


Fig. 18. An RA protograph.

$$\mathbf{M}^{\mathcal{P}_{c_j}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Note that $\mathcal{P}_{c_j}$ is a 10×10 matrix since the total number of pseudocodewords of the check node c_j is $K = 10$ (therefore the number of rows is 10) and $M' \times m_j = 5 \times 2 = 10$ (therefore the number of columns is 10). Except for rows 1 and 8 where $h(\mathbf{x}) = 1$, all other rows of $\mathbf{M}^{\mathcal{P}_{c_j}}$ have $h(\mathbf{x}) = \frac{1}{2}$. To obtain cover-2 pseudocodewords of c_1, $A^{\mathcal{P}_{c_1}^2}([\partial_1, \partial_2])$, we solve $[n_1, n_2, \ldots, n_{10}]\mathbf{M}^{\mathcal{P}_{c_1}} = [\partial_1, \partial_2]$. A similar computation is required for cover-2 pseudocodewords of c_2, $A^{\mathcal{P}_{c_2}^2}([\partial_1, \partial_3])$. Then, we obtain non-zero vector enumerators $A^{(p)}([\partial_1^T, \partial_2^T, \partial_3^T])$ for all possible realizations of $(\partial_1, \partial_2, \partial_3)$.

The final distribution using PAM evaluation in (47) *is shown in Fig. 17.*

Example 12. *We consider a rate-$\frac{1}{2}$ repeat accumulate code over $GF(4)$. The protograph of this code includes one check node with degree 5 (call it c_1) and two variable nodes, one with degree 3, and one with degree 2, as shown in Fig. 18. Suppose that this protograph is copied $N = 3$ times We compute the pseudo-weight enumerator for the graph cover degree $M = 2$ of the resultant code. For this code, the set of all pseudo-symbols is* $\left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right\}$ *which includes 9 distinct non-zero pseudo-symbols, i.e., $M' = 9$.*

For this code, based on the non-zero values of the edges incident to the set of the check node, there are 81 different distinct sets, each including 256 codewords for check node c_1. Now, in order to find $A^{\mathcal{P}_{c_1}^3}([\partial_1, \partial_1, \partial_1, \partial_2, \partial_2])$ for the cover-2 of c_1, similar to Example 11, we seek all the $\mathcal{P}_{c_1}$ sets and as a result, matrix $M^{\mathcal{P}_{c_1}}$ can be computed. Finally, $A^{(p)}([\partial_1^T, \partial_2^T])$ can be computed using (42) *for all choices of (∂_1, ∂_2) (details are omitted). The distribution using PAM is shown in Fig. 19.*

For a finite cover degree M, we now compute the asymptotic ensemble pseudo weight enumerator, as the number of replications N of the original protograph tends to infinity.

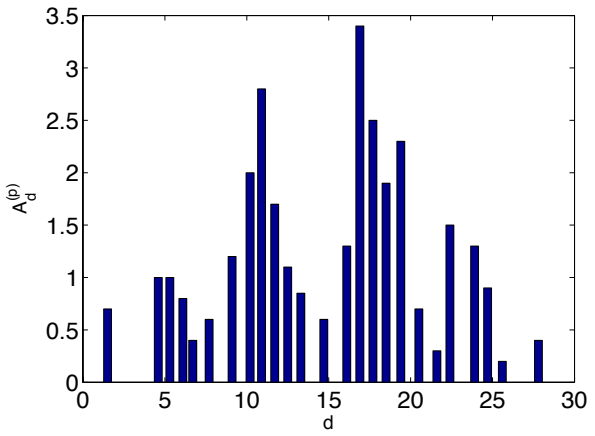


Fig. 19. Pseudo-codeword PAM distance spectrum for the protograph shown in Fig. 18.

We define the normalized logarithmic asymptotic weight (the growth rate) to be

$$r^{(p)}(\delta) \triangleq \limsup_{N\to\infty} \frac{\ln A_d^{(p)}}{N} = \limsup_{N\to\infty} \frac{\ln A_{\delta N}^{(p)}}{N}, \tag{48}$$

where $\delta = d/N$. Note that $n = n_v \cdot N$ so the growth rate in terms of n is then expressed as

$$\tilde{r}^{(p)}(\tilde{\delta}) \triangleq \limsup_{n\to\infty} \frac{\ln A_d^{(p)}}{n}, \tag{49}$$

where $\tilde{r}^p(\tilde{\delta}) = \frac{1}{n_v} r^p(\tilde{\delta} n_v)$. After some computations, it follows that

$$r^{(p)}(\delta) = \max_{\{\boldsymbol{\delta}\}} \left\{ \sum_{j=1}^{n_c} \bar{a}^{\mathcal{P}_{c_j}^{\infty}}(\boldsymbol{\delta}_j) - \sum_{i=1}^{n_v} (t_i - 1) H\left([\delta_{i0},\, \delta_{i1},\, \ldots,\, \delta_{iM'}]\right) \right\}. \tag{50}$$

Under the normalized version of channel-dependent constraints for the AWGN channel with q-ary PAM, δ is

$$\delta = \frac{\left(\sum_{i=1}^{n_v} \sum_{l=1}^{M'} \delta_{i,l} \sum_{k=1}^{q-1} f_{k,l} \cdot k^2 \right)^2}{\sum_{i=1}^{n_v} \sum_{l=1}^{M'} \delta_{i,l} \left(\sum_{k=1}^{q-1} f_{k,l} \cdot k \right)^2}. \tag{51}$$

In (50), $\boldsymbol{\delta} = \mathbf{d}/N$, $\boldsymbol{\delta}_j = \mathbf{d}_j/N$, $\delta_{i,l} = d_{i,l}/N$, $H(\cdot, \cdots, \cdot)$ is the multi-dimensional entropy function, and $\bar{a}^{\mathcal{P}_{c_j}^{\infty}}(\boldsymbol{\delta}_j)$ is the asymptotic vector pseudo-weight enumerator of the check node c_j. This enumerator is computed as

$$\bar{a}^{\mathcal{P}_{c_j}^{\infty}}(\boldsymbol{\delta}_j) = \max_{\{P_{\boldsymbol{\delta}_j}\}} \left\{ H(P_{\boldsymbol{\delta}_j}) + P_{\boldsymbol{\delta}_j} \cdot \mathbf{b}^T \right\}, \tag{52}$$

under the constraint that $\{P_{\boldsymbol{\delta}_j}\}$ is the set of solutions to $\boldsymbol{\delta}_j = P_{\boldsymbol{\delta}_j} \cdot M^{\mathcal{P}_{c_j}}$, with $P_{\boldsymbol{\delta}_j} = [p_1, p_2, \ldots, p_K]$, $p_1, p_2, \ldots, p_K \geq 0$ and $\sum_{k=1}^{K} p_k = 1$.

VI. Iterative Thresholds via EXIT Charts for Non-binary Protographs

In this section we present a novel EXIT chart-based method for computing thresholds of non-binary protograph-based codes with random edge weights. In particular, the method is designed to efficiently evaluate thresholds of U-NBPB codes and generalizes the EXIT-chart methodology previously developed for binary protograph-based codes in [30] and for unstructured non-binary LDPC codes in [5]. We call the proposed technique non-binary PEXIT (NB-PEXIT). Using the NB-PEXIT method, we then offer new non-binary protograph-based codes with capacity-achieving performance for field order as large as $q = 256$. We first revisit the basic concepts underlying the EXIT chart approach in Section VI-A. We then formulate the NB-PEXIT scheme in Section VI-B, and provide threshold evaluations of various codes and modulation schemes in Section VI-C.

A. Preliminaries and Previous Work

The well-known EXIT chart method [8] for computing the decoding threshold of a graph-based code is based on iteratively computing the mutual information (MI) between an edge message and an associated transmitted bit. On the variable node side, we denote the extrinsic MI between the output message of a variable node and the associated transmitted bit as $I_{Ev}(I_{Av}, I_{ch}, d_v)$, where I_{Av} denotes the *a priori* MI between input message of the variable node and the transmitted bit, I_{ch} is the MI between the channel output and the transmitted bit, and d_v is the variable node degree. On the check node side, we denote the extrinsic MI between the output message of a check node and the transmitted bit as $I_{Ec}(I_{Ac}, d_c)$, where I_{Ac} denotes the *a priori* MI between the input message of the check node and the associated transmitted bit, d_c is the check node degree. EXIT chart analysis utilizes the above two functions to examine whether the MI between the edge message and the transmitted bit reaches value 1 by iterative computing. The value 1 of the MI implies that the transmitted bit is decoded correctly by iterative decoding. Since the output message of a variable node is the input message of a check node and vice versa, it follows that $I_{Ac}^{(t)} = I_{Ev}^{(t-1)}$ and $I_{Av}^{(t)} = I_{Ec}^{(t-1)}$ where t denotes the iteration index. At the initial condition $t = 0$, $I_{Ev}^{(0)} = I_{ch}$ and $I_{Ec}^{(0)} = 0$.

The EXIT functions were introduced in [9] for the AWGN channel. We denote by $J(\sigma)$ the MI between a binary random variable $X \in \{+\frac{\sigma^2}{2}, -\frac{\sigma^2}{2}\}$ with equally likely probabilities, and a Gaussian random variable $Y \sim \mathcal{N}(X, \sigma^2)$. Therefore $J(\sigma)$ represents the capacity of a binary-input Gaussian noise channel with the parameter σ and is given in [8]:

$$\begin{aligned} J(\sigma) &= 1 - \mathbb{E}[\log_2(1 + e^{-Y})] \\ &= 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\sigma^2/2)^2}{2\sigma^2}} \log_2(1 + e^{-y}) dy. \end{aligned} \tag{53}$$

Given the AWGN channel with BPSK modulation, the variable node EXIT function for a degree-d_v variable node is

$$I_{Ev}(I_{Av}, I_{ch}, d_v) =$$

$$J(\sqrt{(d_v-1)[J^{-1}(I_{Av})]^2+\sigma_c^2}), \tag{54}$$

where $\sigma_c = \sqrt{8R\frac{E_b}{N_0}}$, R is the code rate and $\frac{E_b}{N_0}$ is the bit signal-to-noise ratio (SNR). The check node EXIT function for a degree-d_c check node can be approximated by the duality property ([9]):

$$I_{Ec}(I_{Ac}, d_c) \simeq 1 - J(\sqrt{d_c-1}J^{-1}(1-I_{Ac})). \tag{55}$$

For the irregular LDPC code ensemble with degree distributions $\lambda(x)$ and $\rho(x)$, the average EXIT functions are

$$I_{Ev} = \sum_{i=1}^{d_v} \lambda_i I_{Ev}(I_{Av}, I_{ch}, i) \tag{56}$$

and

$$I_{Ec} = \sum_{i=1}^{d_c} \rho_i I_{Ec}(I_{Ac}, i), \tag{57}$$

where λ_i and ρ_i are the fractions (edge perspective) of degree-i variable nodes and check nodes, respectively.

For non-binary LDPC (NB-LDPC) code over $GF(q)$, Bennatan and Burshtein proved in [5] the message symmetry and permutation invariance properties given the assumption of uniform random weights and coset vectors. With the support of these two properties, they modeled message distribution as follows.

1) Check-to-variable message distribution: Given the assumption that the edge weight is chosen uniformly over $q-1$ non-zero symbols of $GF(q)$ and given the message symmetry and permutation invariance, the distribution of check-to-variable (c-to-v) messages is formulated as a Gaussian random vector of size $(q-1)$ with mean μ and covariance matrix Σ given by,

$$\mu = \begin{bmatrix} \sigma^2/2 \\ \sigma^2/2 \\ \vdots \\ \sigma^2/2 \end{bmatrix}_{(q-1)\times 1}, \tag{58}$$

and

$$\Sigma = \begin{bmatrix} \sigma^2 & & & \sigma^2/2 \\ & \sigma^2 & & \\ & & \ddots & \\ \sigma^2/2 & & & \sigma^2 \end{bmatrix}_{(q-1)\times(q-1)}, \tag{59}$$

where $\Sigma_{i,j} = \sigma^2$ if $i = j$ and $\sigma^2/2$ otherwise. Note that this distribution is parameterized by σ only.

2) Variable-to-check message distribution: Since the initial message may not be well approximated by a Gaussian random variable, the distribution of variable-to-check (v-to-c) messages is expressed as the superposition of two random vectors: the one representing initial messages from the channel, and another one representing c-to-v messages. The first component depends on the modulation scheme and channel parameter, and the second component is Gaussian with the parameters given in (58) and (59).

The MI between the transmitted symbol S and the random vector W of messages is computed as in [5],

$$I(S;W) = 1 - \mathbb{E}[\log_q(1+\sum_{i=1}^{q-1} e^{-W_i})|S=0], \tag{60}$$

under uniformly distributed transmitted symbols S and edge weights. Here W_i is the i-th entry in W. The MI function between c-to-v messages and transmitted symbols is denoted by $J(\sigma)$, where σ is the parameter of the multivariate Gaussian distribution (see (58), and (59)). The MI function between v-to-c messages and transmitted symbols is denoted by $J_R(\sigma_c, \sigma)$, where σ_c^2 is the variance of the channel output. Unlike in the simpler binary case, a closed form expression is not available for the non-binary MI functions. Monte Carlo method is thus used to suitably approximate these MI functions.

B. NB-PEXIT formulation

The original PEXIT chart analysis [30] is used to evaluate the performance of protograph LDPC code ensembles in the binary case. By evaluating the MI between the *a posteriori* estimation at each variable node and the transmitted codeword bit, PEXIT analysis characterizes the protograph code ensembles by an asymptotic iterative decoding threshold, which is measured as SNR. With the work [5] and the assumption of uniformly distributed edge labels, we generalize the PEXIT analysis to the non-binary setup encompassing different modulations for our U-NBPB codes. For check (variable) node i, we let $N(i)$ denote the set of its neighboring variable (check) nodes. In our non-binary PEXIT (NB-PEXIT) chart analysis there are three components of each iteration:

- **V-to-C update:** Given the transpose of the adjacency matrix B of a protograph (with check nodes indexed by rows and variable nodes indexed by columns), the MI between the v-to-c message from variable node j to check node i and the transmitted symbol v_j is formulated as:

$$I_{Ev}(i,j) = \begin{cases} J(\sigma_{v-to-c}) & \text{if node } j \text{ is punctured,} \\ J_R(\sigma_{v-to-c}) & \text{otherwise.} \end{cases} \tag{61}$$

  Here,

$$J(\sigma_{v-to-c}) = I(S; W = X), \tag{62}$$

$$J_R(\sigma_{v-to-c}) = I(S; W = X+Y), \tag{63}$$

  and

$$\sigma^2_{v-to-c} = \sum_{s\in N(j), s\neq i} b_{s,j}[J^{-1}(I_{Av}(s,j))]^2 + \\ +(b_{i,j}-1)[J^{-1}(I_{Av}(i,j))]^2, \tag{64}$$

  where X is a Gaussian random vector with the parameter σ_{v-to-c} in (58) and (59), Y is the random vector of the initial messages, S is the random variable modeling transmitted symbol v_j, $b_{i,j}$ is the (i,j)-entry of matrix B, and $I_{Ev}(i,j) = 0$ if $b_{i,j} = 0$. Note that for a punctured node there is no information from the channel, and only the c-to-v messages are used to evaluate the MI.

- **C-to-V update:** The MI between the c-to-v message from check node i to variable node j and the transmitted symbol v_j is formulated as a Gaussian random vector with parameter σ_{c-to-v} in (58) and (59):

$$I_{Ec}(i,j) = 1 - J(\sigma_{c-to-v}), \tag{65}$$

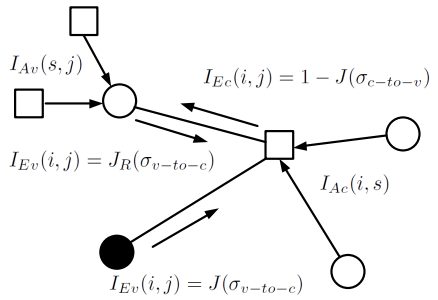


Fig. 20. The MI propagation on a part of a protograph.

where

$$\sigma^2_{c-to-v} = \sum_{s\in N(i), s\neq j} b_{i,s}[J^{-1}(1 - I_{Ac}(i,s))]^2 + (b_{i,j} - 1)[J^{-1}(1 - I_{Ac}(i,j))]^2. \quad (66)$$

- **Convergence evaluation:** Similar to the v-to-c update, the MI between the *a posteriori* message and the transmitted symbol at variable node j is a superposition of two random vectors, which represent initial messages and c-to-v messages, respectively:

$$I_{AP}(j) = \begin{cases} J(\sigma_{AP}) & \text{if node } j \text{ is punctured,} \\ J_R(\sigma_{AP}) & \text{otherwise,} \end{cases} \quad (67)$$

where

$$\sigma^2_{AP} = \sum_{s\in N(j)} b_{s,j}[J^{-1}(I_{Av}(s,j))]^2. \quad (68)$$

The evaluation process ends when either $I_{AP}(j) = 1$ for all variable nodes, or the algorithm reaches the maximum number of iterations.

The MI propagation on a bipartite graph is illustrated in Fig. 20.

C. Threshold evaluation

In this section we present the threshold evaluation results of different NB protographs using their binary images over AWGN, and considering PSK and PAM modulations. The maximum number of iterations is 1000. The value of I_{AP} typically converges to a constant lower than 1 in 100 iterations, if the decoder fails to produce the correct estimate of the transmitted codeword. We compute the SNR threshold with up to two decimal places. First, we present the result for the binary case to demonstrate the consistency with existing works that use complementary methods for computing the SNR threshold. We then describe the underlying variables governing the proposed NB-PEXIT tool, and present several illustrative examples.

1) Binary case: We focus on the RA codes and ARA codes [1] and several popular regular codes to evaluate the consistency of the NB-PEXIT analysis in the binary case. Tables I and II illustrate the SNR thresholds of the AR3A family and AR4A family over the AWGN channel with BPSK modulation, respectively. Table III shows the thresholds of the RA code and the regular-$(2,4)$ binary code. The errors $|\epsilon|$ are less than 0.05 dB for all protographs of different rates. These results demonstrate that NB-PEXIT analysis is consistent with the existing works when reduced to the binary case.

R	DE threshold (dB)	PEXIT threshold (dB)	$\lvert\epsilon\rvert$
1/2	0.516	0.56	0.044
2/3	1.288	1.30	0.012
3/4	1.848	1.88	0.032
4/5	2.277	2.30	0.023
5/6	2.626	2.60	0.026
6/7	2.897	2.91	0.013
7/8	3.129	3.14	0.011

TABLE I
PROTOGRAPHS AND THRESHOLDS FOR THE AR3A FAMILY.

R	DE threshold (dB)	PEXIT threshold (dB)	$\lvert\epsilon\rvert$
1/2	0.560	0.55	0.010
2/3	1.414	1.45	0.036
3/4	1.980	1.96	0.020
4/5	2.396	2.41	0.014
5/6	2.717	2.72	0.003
6/7	2.980	2.97	0.010
7/8	3.197	3.24	0.043

TABLE II
PROTOGRAPHS AND THRESHOLDS FOR THE AR4A FAMILY.

2) Binary image of NBPB codes over AWGN with BPSK: Here, we consider the AWGN transmission for each individual bit in the binary image of the symbols of U-NBPB codes. Let σ_c^2 denote the variance of channel noise.

Let μ_B be an r-length vector with all entries set to $\sigma_c^2/2$. Let Σ_B be an $r\times r$ diagonal matrix with σ_c^2 as diagonal entries.

Following the set-up in the binary case, the random vector of the initial message Y can be represented as a Gaussian random vector with mean

$$\mu_Y = \begin{bmatrix} b_r(1) \\ b_r(2) \\ \vdots \\ b_r(q-1) \end{bmatrix} \mu_B = \begin{bmatrix} d(1) \\ d(2) \\ \vdots \\ d(q-1) \end{bmatrix} \frac{\sigma_c^2}{2}, \quad (69)$$

where $b_r(k)$ is the binary row vector, which contains the binary representation of k, and $d(k)$ is the Hamming weight of the binary representation of k. The covariance Σ_Y of Y is

$$\Sigma_Y = \begin{bmatrix} b_r(1) \\ b_r(2) \\ \vdots \\ b_r(q-1) \end{bmatrix} \Sigma_B \begin{bmatrix} b_r(1)^T & \dots & b_r(q-1)^T \end{bmatrix}. \quad (70)$$

Therefore, for an unpunctured variable node, the distribution

code	DE threshold (dB)	PEXIT threshold (dB)	$\lvert\epsilon\rvert$
rate 1/2 RA code	1.116	1.14	0.024
regular-(2,4) code	3.004	3.04	0.036

TABLE III
PROTOGRAPHS AND THRESHOLDS FOR THE RATE-1/2 RA CODE AND REGULAR-$(2,4)$ CODE.

of v-to-c messages W in (63) is a $(q-1)$-dimensional vector Gaussian random vector with mean μ_W and covariance Σ_W with entries

$$\mu_W(k) = d(k)\frac{\sigma_c^2}{2} + \frac{\sigma_{v-to-c}^2}{2}, \tag{71}$$

and

$$\Sigma_W(k,\ell) = \begin{cases} d(k)\sigma_c^2 + \sigma_{v-to-c}^2 & \text{if } k = \ell, \\ d(k \otimes \ell)\sigma_c^2 + \frac{\sigma_{v-to-c}^2}{2} & \text{otherwise.} \end{cases} \tag{72}$$

Here, $\otimes$ represents the bit-wise AND operation.

We compute the function $J_R(\sigma_c, \sigma_{v-to-c})$ by applying the Monte Carlo method. Given the assumption that the energy is uniformly distributed over the binary image of a symbol, we have the following representation:

$$\Lambda_b = \Lambda_s - 10\log_{10} r, q = 2^r, \tag{73}$$

where Λ_b is the bit SNR, Λ_s is the symbol SNR in the logarithmic decibel scale, and q is the size of the Galois field used to represent the non-binary symbols.

In the following, we discuss several illustrative examples and point out some interesting observations.

Example 13. *We present iterative decoding thresholds in Fig. 21 for candidate protographs shown in Fig. 3, where we consider the binary image of the code used in transmission over an AWGN channel with BPSK modulation. From this figure, we first observe that the threshold of the regular* $(2,4)$ *protograph monotonically decreases as the alphabet size* q *in* $GF(q)$ *increases. Among the three codes, the punctured* $(2,4)$ *type 2 protograph has the best threshold at* $q = 8$*. We recall that this is consistent with the enumeration results previously presented in Examples 2 and 4. Interestingly, the punctured* $(2,4)$ *type 1 protograph has a lower threshold at* $q = 16, 32, 64$ *than the regular* $(2,4)$ *protograph has at* $q = 256$*, and this threshold is only within* $0.2dB$ *of the capacity* $0.187dB$*.*

The next example provides further illustrations.

Example 14. *We consider the RA protograph shown in Fig. 18 and additional two protographs shown in Fig. 22. Black nodes are punctured. The iterative decoding thresholds of their binary image over AWGN channel with BPSK are shown in Fig. 23.*

From Fig. 21 and Fig. 23, one observation is that the function relating the field order q and the decoding threshold depends on the average variable node degree (AVND). Protographs with small AVND (<2.5) tend to have higher thresholds for small q and lower thresholds for large q. In contrast, protographs with large AVND (>2.5) tend to have lower thresholds for small q and higher thresholds for large q.

To further demonstrate this phenomenon and eliminate potential influence of punctured nodes, we compare the thresholds of the regular $(2,4)$ protograph (AVND$=2$) shown in Fig. 3(a), the RA protograph shown in Fig. 18 (AVND=2.5) with the regular $(3,6)$ protograph shown in Fig. 8 (AVND$=3$) and the protograph shown in Fig. 24. This protograph has two variable nodes of degree 4 and two variable nodes of degree 3 so its AVND is 3.5. The results are plotted in Fig. 25. The four protographs have the lowest thresholds at $q = 256$, $q = 16$, $q = 8$, and $q = 2$, respectively which is inversely correlated with the value of AVND. In particular, the thresholds of the regular $(2,4)$ protograph decreases with q and the threshold of the protograph in Fig. 24 increases with q. Observations of this type may be useful when designing NBPB codes over $GF(q)$.

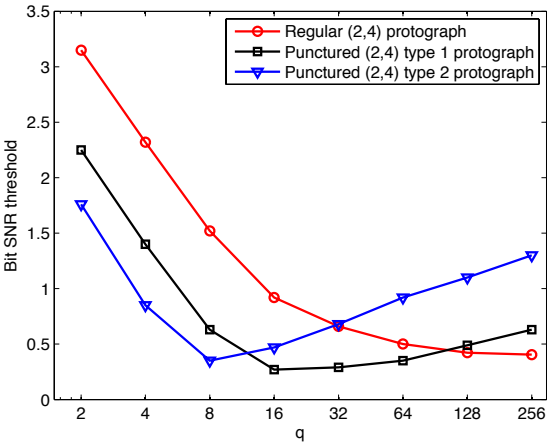


Fig. 21. Bit SNR threshold of the binary image of candidate protograph in Fig. 3 with BPSK over AWGN.

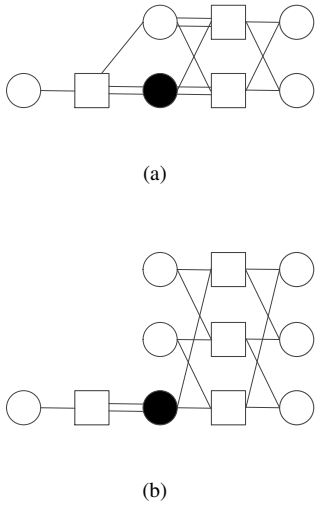


Fig. 22. Candidate protographs. (a) AR3A protograph, and (b) Punctured $(2,4)$ type 3 protograph.

Similar to the analysis for the binary image of NBPB codes suitable for the BPSK modulation, we apply the Monte Carlo method to approximate $J(\sigma)$ and $J_R(\sigma)$ for PAM and PSK modulations over AWGN channel. For PAM we apply a technique in [47], which suggests a non-uniform constellation for PAM. We apply the equation (73) to normalize the symbol SNR with respect to the number of bits in a symbol.

Example 15. *Fig. 26 shows the bit SNR thresholds with PAM modulation over AWGN for protographs in Fig. 3. Fig.*

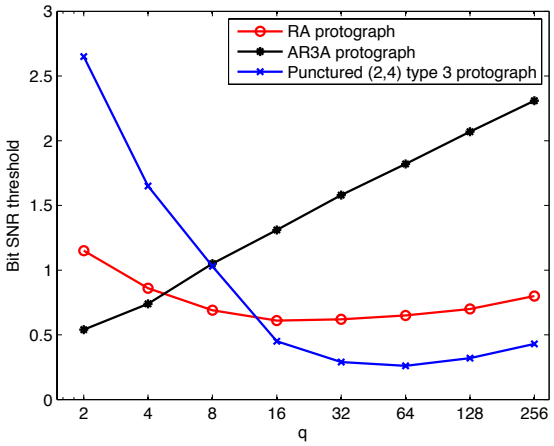


Fig. 23. Bit SNR threshold of the binary image of candidate protographs in Fig. 22 with BPSK over AWGN.

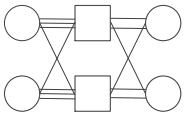


Fig. 24. A candidate protograph with AVND=3.5.

27 shows the thresholds for QPSK modulation over AWGN for protographs in Fig. 3. Similar to the BPSK over AWGN channel transmission, the regular $(2, 4)$ *protograph has the highest threshold when* q *is small and the lowest threshold when* q *is large.*

We remark that for C-NBPB codes, the edge labels are fixed, so the message distribution specified in (58) and (59) does not apply. Further, the MI between the transmitted symbol and edge message does not change with different edge labels. In order to measure the threshold of C-NBPB code, a more accurate method should be considered.

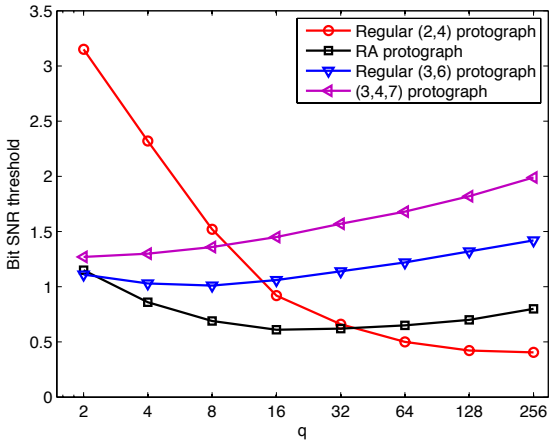


Fig. 25. Bit SNR threshold of the binary image of candidate protograph with different AVND with BPSK over AWGN.

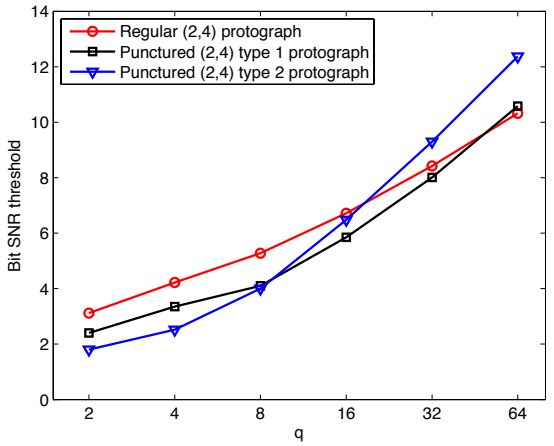


Fig. 26. Bit SNR thresholds of NBPB codes with PAM over AWGN.

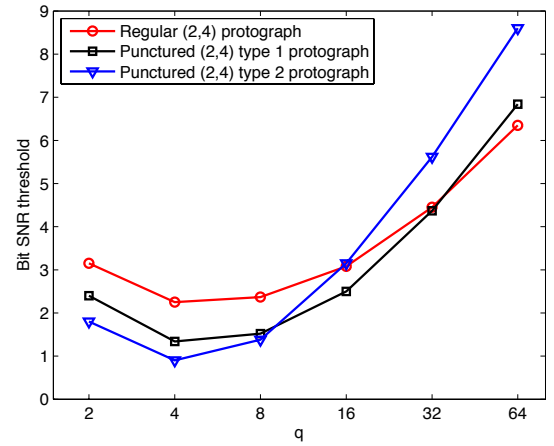


Fig. 27. Bit SNR thresholds of NBPB codes with QPSK over AWGN.

VII. Finite-length U-NBPB and C-NBPB codes

Building upon the analysis from the previous sections, in this section we provide new code designs for finite-length NBPB codes. The proposed codes offer excellent performance, both relative to binary and non-binary constructions. In Section VII-A, we summarize the design guidelines for the code construction. In Sections VII-B and VII-C, we present examples of finite-length C-NBPB codes and U-NBPB codes constructed using our approach. Simulation results are presented for the AWGN channel with an FFT-based decoder [6] with 100 maximum iterations.

A. Design criteria for NBPB codes

The design method for short blocks is based on ensuring a large enough girth and a suitable edge label assignment to achieve a good code minimum distance. Our design criteria is similar to the construction method used by Poulliat, Fossorier, and Declercq in [39]. Here, for a given protograph, we obtain a derived graph by copy-and-permute operations using the circulant PEG algorithm [24] such that the girth is as large as possible. The selection of the original protograph is guided

by our EXIT analysis: we start with a protograph having a competitive threshold (as computed in the previous section) at the prescribed alphabet size. After lifting the protograph to a desired size to obtain a large girth, we compute the cycle distribution of the graph to select the best one among the resulting graph candidates.

The assignment of the non-binary labels from a given field $GF(q)$ is selective rather than random. We select the labels so that the binary image of the check node with the assigned labels produces the largest possible minimum Hamming distance for that check node. Such label optimization was used by MacKay [31] and in [39]. However, the number of labels proposed in these references are somewhat limited. In contrast, for each q, we generate all possible label sets that produce the largest minimum distance for the binary images of the checks with associated labels. Moreover, we also modify the cycle cancellation method proposed in [39] so that, in addition to the shortest cycles, we attempt to eliminate other short cycles as well. In our extensive simulations of non-binary codes, with the all-zero transmitted codewords, the lowest-weight detected codewords were collected. The lowest weight of the binary image of empirically collected codewords is denoted by d^u_{min}. The true minimum weight codeword is of course less than or equal to d^u_{min}.

B. Examples of C-NBPB code construction and performance simulation results

First, we construct regular $(2, 4)$ C-NBPB codes with code rate $1/2$ and block-lengths 128, 256, and 512 in bits, by taking the binary image of constructed non-binary protograph codes over $GF(256)$. The reason for choosing $q = 256$ is influenced by the EXIT analysis as well as experimental observations indicating that iterative non-binary decoders perform better for higher alphabet size.

Here is an example of construction of a $(128, 64)$ C-NBPB code. We start with a scaled protograph with four degree-2 variable nodes and two degree-4 check nodes. We select two sets of optimal labels (each set consists of 4 labels per check node of degree-4) among all sets provided in [39]. This protograph and its edge scalings are shown in Fig. 28(a). In the figure, the labels are non-zero elements of $GF(256)$ and are expressed as powers of a primitive element α of the field, where α is a root of the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$.

We then lift this protograph by a factor of $N = 4$ using circulant $N \times N = 4 \times 4$ permutations that produce the largest possible girth with minimal multiplicity. The circulant permutations are specified by σ^i, where σ denotes the unit left circular shift of the identity matrix. The permutations are also indicated in Fig. 28(a). Distribution of cycles for the resulting lifted graph is shown in Table IV (first row group). The girth of the graph is 8, and d^u_{min} is 15. It is interesting to observe that for the U-NBPB code, there are no codewords associated with single cycles, that is low-weight codewords are all due to interconnected cycles.

Similar constructions produce $(256, 128)$ $(N = 8)$, and $(512, 256)$ $(N = 16)$ codes. Their scaled protographs and

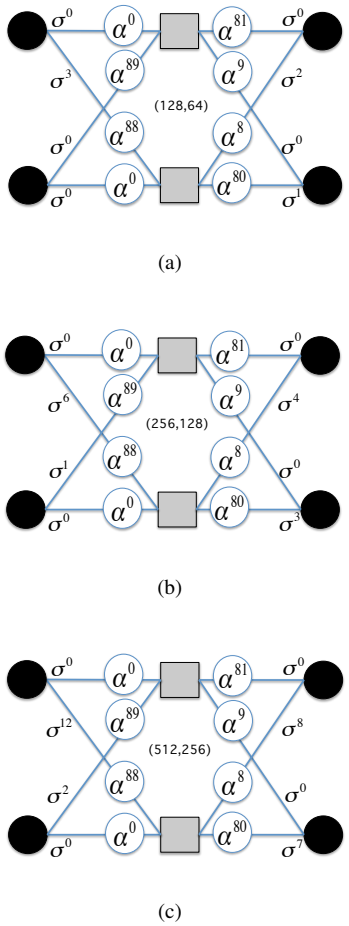


Fig. 28. Non-binary protographs, circulant permutations with optimal assigned labels are shown. The derived graphs are obtained by using graph cover construction.

lifting operations are shown in Fig. 28(b) and Fig. 28(c), respectively. The cycle distributions are also shown in Table IV, in the second and third row groups. (The distribution of cycles is shown for cycles up to size 32.) These two codes both have d^u_{min} of 15.

Note that the derived graphs are obtained by only performing the copy and permute operations, while preserving the same labels as in the underlying graph. We also constructed a $(2048, 1024)$ code using the same method. A slightly better code was proposed in [39]. Performance simulation of the proposed codes is shown in Fig. 29.

C. Examples of U-NBPB code construction and performance simulation results

We also construct regular $(2, 4)$ U-NBPB codes with code rate $1/2$ and block-lengths 128, 256, and 512 in bits, by taking the binary image of constructed non-binary protograph codes over $GF(256)$. We start with a protograph with four degree-2 variable nodes and two degree-4 check node with no scaling attached to the edges. We use the same circulant permutations as in Fig. 28 to lift the graph by a factor of N $(N = 4, 8, 16)$ to obtain the derived graphs. From the available sets of optimal labels, we randomly selected a label set (or its

cycle distribution, no. associated codewords / cycle size	8	12	16	20	24	28	32
N=4	36	96	72	0	0	0	0
C-NBPB (128,64)	0	16	16	0	0	0	0
U-NBPB (128.64)	0	0	0	0	0	0	0
N=8	20	160	634	2304	5184	5632	1464
C-NBPB (256,128)	0	32	48	288	640	1152	1152
U-NBPB (256,128)	0	3	7	21	43	69	71
N=16	0	208	788	5760	28392	146192	614872
C-NBPB (512,256)	0	64	96	864	3488	16448	61248
U-NBPB (512,256)	0	0	4	22	144	729	3134

TABLE IV
DISTRIBUTION OF CYCLES AND ASSOCIATED CODEWORDS FOR THE RESULTANT GRAPH.THE GRAPH IS BASED ON THE PROTOGRAPH WITH 4 VARIABLE NODES AND 2 CHECK NODES EXPANDED BY CIRCULANT PERMUTATION MATRICES OF SIZE $N \times N$.

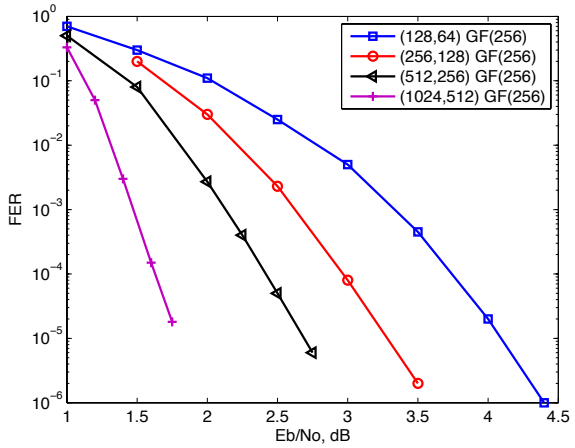


Fig. 29. Simulation of rate-$1/2$, (n, k) binary image of C-NBPB codes, for blocklength $n = 128, 256, 512, 2048$, in bits.

permuted version) and assigned it to the edges of each check node. The parity-check matrix of $(128, 64)$ U-NBPB code is shown in Table V. The table is interpreted as follows: the matrix has 8 rows and 16 columns with entries over $GF(256)$. The non-zero entries are specified by their value and the row and column indices. For example, the non-zero entry in row 1 and column 6 is α^{89}, where α is a primitive element of $GF(256)$ and is a root of $x^8 + x^4 + x^3 + x^2 + 1$. The parity-check matrices of $(256, 128)$ and $(512, 256)$ U-NBPB codes are shown in Tables VI and VII. We obtained $d^u_{min} = 14$ for the $(128, 64)$ U-NBPB code, $d^u_{min} = 16$ for the $(256, 128)$ U-NBPB code, and $d^u_{min} = 23$ for the $(512, 256)$ U-NBPB code.

Performance simulations of these codes are shown in Fig. 30. For comparison, we also plotted the simulation results for our best binary protograph codes with variable node degrees 3 and 5, proposed to the Consultative Committee for Space Data Systems (CCSDS) [54]. The minimum distance of this binary $(128, 64)$ code is 14. It is worthwhile to note that the non-binary $(2, 4)$ protograph codes over $GF(256)$ outperform their binary counterparts by around 1.25, 1.15, and 1.05 dB for $n = 128, 256$, and 512, respectively, at FER of 10^{-5}. Further, we plot non-binary LDPC codes with the same code

Row index	Column index and scaling			
1	$(1,\alpha^0)$	$(6,\alpha^{89})$	$(12,\alpha^{81})$	$(15,\alpha^9)$
2	$(2,\alpha^8)$	$(7,\alpha^0)$	$(9,\alpha^{182})$	$(16,\alpha^{173})$
3	$(3,\alpha^{173})$	$(8,\alpha^8)$	$(10,\alpha^0)$	$(13,\alpha^{183})$
4	$(4,\alpha^8)$	$(5,\alpha^0)$	$(11,\alpha^{88})$	$(14,\alpha^{80})$
5	$(1,\alpha^{183})$	$(5,\alpha^{173})$	$(9,\alpha^8)$	$(16,\alpha^0)$
6	$(2,\alpha^0)$	$(6,\alpha^{88})$	$(10,\alpha^{80})$	$(14,\alpha^8)$
7	$(3,\alpha^0)$	$(7,\alpha^{167})$	$(11,\alpha^{127})$	$(15,\alpha^{40})$
8	$(4,\alpha^0)$	$(8,\alpha^{182})$	$(12,\alpha^{173})$	$(16,\alpha^8)$

TABLE V
PARITY CHECK MATRIX OF A $(128, 64)$ U-NBPB CODE OVER $GF(256)$.

parameters as our U-NBPB designs. For each code length we selected three such codes at random; these curves are labeled with dashed lines with triangles in the plot. It is clear that the proposed codes outperform codes with uninformed edge labeling and unoptimized choice of permutation matrices.

Obviously, the C-NBPB codes are more restrictive and should not perform as well as the U-NBPB codes. Interestingly, our simulation results shows that the performance of the C-NBPB codes (at least for block sizes $n = 128, 256, 512$ in bits) are almost on top of the U-NBPB code performance. Performance of these codes matches with the performance of unconstrained non-binary codes reported in [11].

We also compare our rate-$1/2$ U-NBPB codes over $GF(256)$ with U-NBPB codes over $GF(16)$. The simulation result is shown in Fig. 31. The $(128, 64)$ code over $GF(16)$ is built from the RA protograph shown in Fig. 18, $(256, 128)$ and $(512, 256)$ codes over $GF(16)$ are built from the IRA protograph shown in Fig. 32. Simulation shows that U-NBPB codes over $GF(16)$ outperform their binary counterparts by around 0.80, 0.95, and 0.85 dB for $n = 128, 256$, and 512, respectively, at FER $= 10^{-5}$, but have higher frame error rate than U-NBPB codes over $GF(256)$ for the same SNR.

By puncturing one of the four variable nodes in Fig. 28, we can get a rate-$2/3$ protograph. The NB-PEXIT analysis shows that the rate-$2/3$ protograph has the lowest threshold when $q = 256$ (1.15dB), which is only 0.1dB higher than the channel capacity of 1.059dB. Using the same Tanner graphs and labels with the three U-NBPB codes, we construct rate-$2/3$ U-NBPB codes with block-lengths 96, 192 and 384 bits by taking the binary image of constructed non-binary protograph codes over $GF(256)$. Performance simulations of these codes are shown in Fig. 33. For comparison, we plot simulation results of rate-$2/3$ binary LDPC codes with length 192 bits proposed in [44]. Our non-binary code over $GF(256)$ of the same length outperforms the binary code by around 0.85dB at FER $= 10^{-5}$. We also compare our code of length 384 bits with a binary code proposed in [22]. We remark that the code in [22] has a somewhat lower rate $(3/5)$ and more than double the length of our proposed code. Yet, our code shows similar performance in terms of the bit error rate.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we presented a class of structured non-binary LDPC codes built out of protographs, called NBPB codes, wherein we considered both constrained and unconstrained

Row index	Column index and scaling			
1	(1,α^{0})	(11,α^{8})	(22,α^{80})	(29,α^{88})
2	(2,α^{0})	(12,α^{8})	(23,α^{80})	(30,α^{89})
3	(3,α^{0})	(13,α^{8})	(24,α^{80})	(31,α^{90})
4	(4,α^{0})	(14,α^{8})	(17,α^{80})	(32,α^{91})
5	(5,α^{0})	(15,α^{8})	(18,α^{81})	(25,α^{89})
6	(6,α^{0})	(16,α^{8})	(19,α^{81})	(26,α^{90})
7	(7,α^{0})	(9,α^{8})	(20,α^{81})	(27,α^{91})
8	(8,α^{0})	(10,α^{8})	(21,α^{82})	(28,α^{90})
9	(8,α^{90})	(9,α^{81})	(17,α^{9})	(25,α^{0})
10	(1,α^{91})	(10,α^{81})	(18,α^{9})	(26,α^{0})
11	(2,α^{91})	(11,α^{82})	(19,α^{9})	(27,α^{0})
12	(3,α^{183})	(12,α^{173})	(20,α^{9})	(28,α^{0})
13	(4,α^{86})	(13,α^{72})	(21,α^{10})	(29,α^{0})
14	(5,α^{87})	(14,α^{72})	(22,α^{10})	(30,α^{0})
15	(6,α^{151})	(15,α^{72})	(23,α^{10})	(31,α^{0})
16	(7,α^{86})	(16,α^{72})	(24,α^{11})	(32,α^{0})

TABLE VI
PARITY CHECK MATRIX OF A $(256, 128)$ U-NBPB CODE OVER $GF(256)$.

Row index	Column index and scaling			
1	(1,α^{0})	(21,α^{8})	(42,α^{80})	(57,α^{88})
2	(2,α^{0})	(22,α^{8})	(43,α^{80})	(58,α^{89})
3	(3,α^{0})	(23,α^{8})	(44,α^{80})	(59,α^{90})
4	(4,α^{0})	(24,α^{8})	(45,α^{80})	(60,α^{91})
5	(5,α^{0})	(25,α^{8})	(46,α^{81})	(61,α^{89})
6	(6,α^{0})	(26,α^{8})	(47,α^{81})	(62,α^{90})
7	(7,α^{0})	(27,α^{8})	(48,α^{81})	(63,α^{91})
8	(8,α^{0})	(28,α^{8})	(33,α^{82})	(64,α^{90})
9	(9,α^{0})	(29,α^{8})	(34,α^{82})	(49,α^{91})
10	(10,α^{0})	(30,α^{8})	(35,α^{83})	(50,α^{91})
11	(11,α^{0})	(31,α^{8})	(36,α^{172})	(51,α^{181})
12	(12,α^{0})	(32,α^{8})	(37,α^{172})	(52,α^{182})
13	(13,α^{0})	(17,α^{8})	(38,α^{172})	(53,α^{183})
14	(14,α^{0})	(18,α^{8})	(39,α^{173})	(54,α^{182})
15	(15,α^{0})	(19,α^{8})	(40,α^{173})	(55,α^{183})
16	(16,α^{0})	(20,α^{8})	(41,α^{174})	(56,α^{183})
17	(15,α^{90})	(17,α^{81})	(33,α^{9})	(49,α^{0})
18	(16,α^{91})	(18,α^{81})	(34,α^{9})	(50,α^{0})
19	(1,α^{91})	(19,α^{82})	(35,α^{9})	(51,α^{0})
20	(2,α^{183})	(20,α^{173})	(36,α^{9})	(52,α^{0})
21	(3,α^{86})	(21,α^{72})	(37,α^{10})	(53,α^{0})
22	(4,α^{87})	(22,α^{72})	(38,α^{10})	(54,α^{0})
23	(5,α^{151})	(23,α^{72})	(39,α^{10})	(55,α^{0})
24	(6,α^{86})	(24,α^{72})	(40,α^{11})	(56,α^{0})
25	(7,α^{87})	(25,α^{72})	(41,α^{11})	(57,α^{0})
26	(8,α^{87})	(26,α^{73})	(42,α^{11})	(58,α^{0})
27	(9,α^{88})	(27,α^{73})	(43,α^{11})	(59,α^{0})
28	(10,α^{73})	(28,α^{56})	(44,α^{14})	(60,α^{0})
29	(11,α^{74})	(29,α^{57})	(45,α^{15})	(61,α^{0})
30	(12,α^{166})	(30,α^{126})	(46,α^{40})	(62,α^{0})
31	(13,α^{167})	(31,α^{126})	(47,α^{40})	(63,α^{0})
32	(14,α^{168})	(32,α^{126})	(48,α^{40})	(64,α^{0})

TABLE VII
PARITY CHECK MATRIX OF A $(512, 256)$ U-NBPB CODE OVER $GF(256)$.

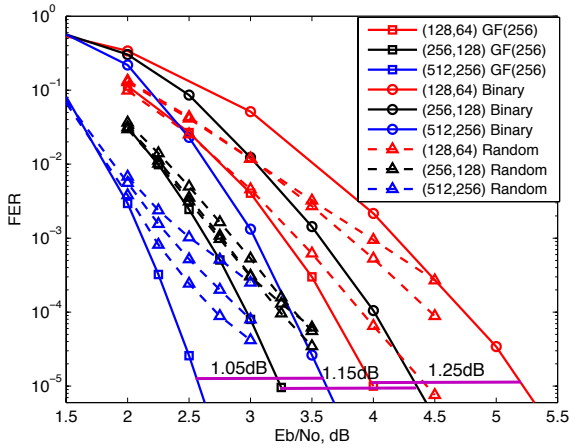


Fig. 30. Comparison of the simulation results of the binary image of rate-$1/2$ U-NBPB codes with various block-lengths with best known binary protograph codes and with codes having random edge labels.

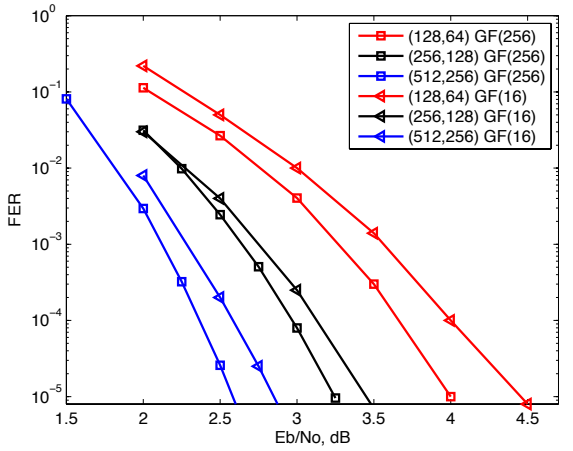


Fig. 31. Comparison of the simulation results of the binary image of rate-$1/2$ U-NBPB codes over $GF(256)$ and $GF(16)$ of various block-lengths.

edge weight selections. In many instances, non-binary constructions were shown to have superior properties to their binary counterparts. Specifically, we computed various enumerators of NBPB ensembles for both the finite and the infinite block-length regimes. We also provided new EXIT chart style analysis for identifying NBPB codes with good thresholds. We also proposed designs of some excellent finite length NBPB codes. Collectively, these results offer a comprehensive framework for designing and analyzing structured non-binary LDPC codes.

Given a considerable freedom in designing non-binary graph-based codes, there are several exciting directions for

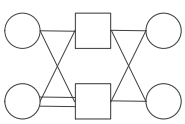


Fig. 32. An IRA protograph.

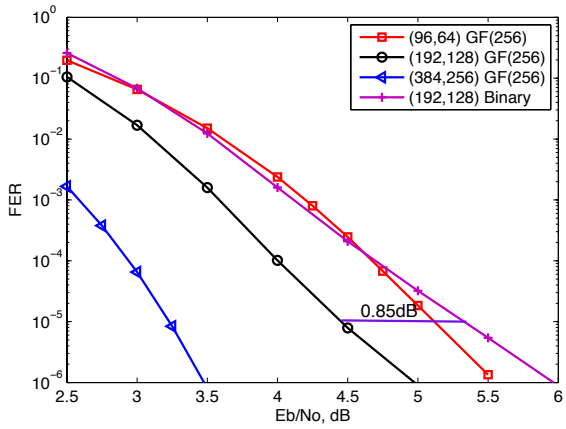


Fig. 33. Simulation results of the binary image of rate-$2/3$ U-NBPB codes and a comparison with a binary code of length 384 bits and rate $2/3$, taken from [44].

future investigation. For example, building upon the results presented here, one may wish to further investigate trapping sets and related objects for finite-length NBPB codes and optimize code design based on the elimination of such objects. Another interesting future direction would to explore a reformulation of the C-NBPB (graph cover-style) constructions in terms of the factor graphs and potentially utilize some of the enumeration methods recently presented in [51]. One may also seek to design NBPB codes in conjunction with a prescribed high-order modulation scheme. While the focus of this work was on the code design and analysis, it is equally important to devise practical decoding algorithms. Given the rich structure of NBPB codes, it may be possible to simplify certain a priori complex decoding steps thus making non-binary LDPC codes designed over very high order fields a reality.

ACKNOWLEDGEMENTS

The authors thank Mr. Ben-Yue Chang for help with the original formulation of NB-EXIT tool and for help with the design methodology of finite-length NBPB codes.

REFERENCES

[1] A. Abbasfar, D. Divsalar and K. Yao, "Accumulate-repeat-accumulate codes," *IEEE Trans. on Commun.*, vol. 55, no. 4, pp. 692-702, Apr. 2007.

[2] S. Abu-Surra, D. Divsalar and W. E. Ryan, "Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes," *IEEE Trans. on Inf. Theory*, vol. 57, no. 2, pp. 858-886, Feb. 2011.

[3] I. Andriyanova and K. Kasai, "Finite-length scaling of non-binary (c, d) LDPC codes for the BEC," in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Austin, TX, Jun. 2010, pp. 714-718.

[4] I. Andriyanova, V. Rathi and J-P. Tillich, "Binary weight distribution of non-binary LDPC codes," in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Seoul, South Korea, Jun. - Jul. 2009, pp. 65-69.

[5] A. Bennatan and D. Burshtein, "Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels," *IEEE Trans. on Inf. Theory*, vol. 52, no. 2, pp. 549-583, Feb. 2006.

[6] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC codes over GF(2^q)," in *Proc. IEEE Inform. Theory Workshop*, Paris, France, Mar. 2003, pp. 70-73.

[7] S. Benedetto, D. Divsalar, G.Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *IEEE Trans. on Inf. Theory*, vol. 44, no. 3, pp. 909-926, May 1998.

[8] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. on Commun.*, vol. 49, no. 10, pp. 1727-1737, Oct. 2001.

[9] S. ten Brink, G. Kramer and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. on Commun.*, vol. 52, no. 4, pp. 670-678, Apr. 2004.

[10] C-Y. Chen, Q. Huang, C-C. Chao and S. Lin, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes," *IEEE Trans. on Commun.*, vol. 58, no. 11, pp. 3140-3147, Nov. 2010.

[11] L. Costantini, B. Matuz, G. Liva, E. Paolini and M. Chiani, "Non-binary protograph low-density parity-check codes for space communications," *Int. Journal of Satellite Commun. and Network* (Wiley), vol. 30, pp. 43-51, Mar. - Apr. 2012.

[12] T. Cover and J. Thomas, *Elements of Information Theory*, 2nd Ed. Wiley-Interscience, 2006.

[13] M. Davey and D. J. C. MacKay, "Low density parity check codes over $GF(q)$," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165-167, Jun. 1998.

[14] D. Declercq and S. Pfletschinger, "Performance comparison of NB-LDPC, NB-root-LDPC, and NB-DGLDPC", *DaVinci Deliverable D*, http://www.ict-davinci-codes.eu, Jun. 2010.

[15] D. Declercq, V. Savin and L. Pham Sy, "Analysis and design of ultra-sparse non-binary cluster-LDPC codes," in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Cambridge, MA, Jul. 2012, pp. 2531-2535.

[16] D. Divsalar and L. Dolecek, "On the typical minimum distance of protograph-based non-binary LDPC codes," in *UCSD Workshop on Inf. Theory and Its Applications (ITA)*, San Diego, CA, Feb, 2012.

[17] I. Djordjevic, W. E. Ryan and B. Vasic, *Coding for Optical Channels*, Springer, 2010.

[18] M. El-Khamy, "New approaches to the analysis and design of Reed-Solomon related codes," Ph.D. Thesis, California Institute of Technology, Sep. 2006.

[19] M. F. Flanagan, E. Paolini, M. Chiani and M. P. C. Fossorier, "Growth rate of the weight distribution of doubly-generalized LDPC codes: general case and efficient evaluation," in *Proc. of IEEE Global Telecommu. Conf. (Globecom)*, Honolulu, HI, Nov. -Dec. 2009, pp. 1-6.

[20] R. Gabrys, E. Yaakobi and L. Dolecek, "Graded bit error correcting codes with applications to flash memory," *IEEE Trans. on Inf. Theory*, 2013, to appear.

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

[22] J. Ha, J. Kim, D. Klinc and S. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *IEEE Trans. on Inf. Theory*, vol. 52, no. 2, pp. 728-738, Feb. 2006.

[23] J. Huang, L. Liu, W. Zhou and S. Zhou, "Large-girth nonbinary QC-LDPC codes of various lengths," *IEEE Trans. on Commun.*, vol. 58, no. 12, pp. 3436-3447, Dec. 2010.

[24] 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, Jan. 2005.

[25] J. Kang, Q. Huang, L. Zhang, B. Zhou and S. Lin, "Quasi-cyclic LDPC codes: an algebraic construction," *IEEE Trans. on Commun.*, vol .58, no. 5, pp. 1383-1396, May 2010.

[26] K. Kasai, C. Poulliat, D. Declercq, T Shibuya and K. Sakaniwa, "Weight distribution of nonbinary LDPC codes," in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Auckland, New Zealand, Dec. 2008, pp. 1-6.

[27] C. Kelley, D. Sridhara and J. Rosenthal, "Pseudocodeword weights for non-binary LDPC codes," in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Seattle, WA, Jul. 2006, pp. 1379-1383.

[28] R. Koetter and P. Vontobel, "Graph-covers and iterative decoding of finite length codes", in *Proc. of IEEE Int. Symp. on Turbo Codes and Apps.*, Brest, France, Sep. 2003.

[29] L. Liu, J. Huang, W. Zhou and S. Zhou, "Computing the minimum distance of nonbinary LDPC codes," *IEEE Trans. on Commun.*, vol. 60, no. 7, pp. 1753-1758, Jul. 2012.

[30] G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," in *Proc. of IEEE Global Telecommu. Conf. (Globecom)*, Washington, DC, Nov. 2007, pp. 3250-3254.

[31] D. MacKay, "Optimizing sparse graph codes over $GF(q)$," available at http://www.inference.phy.cam.ac.uk/mackay/CodesGallager.html, Aug. 2003.

[32] A. Marinoni, P. Savazzi and R.D. Wesel, "Protograph-based q-ary LDPC codes for higher-order modulation," in *Proc. IEEE 6th Int. Symp on Turbo Codes and Related Topics*, Brest, France, Sep. 2010, pp. 6-10.

[33] R. J. McEliece, "Practical codes for photon communication," *IEEE Trans. on Inf. Theory*, vol. 27, no. 4, pp. 393-398, Jul. 1981.

[34] D. G. M. Mitchell, A. E. Pusane and D. J. Costello, "Trapping set analysis of protograph-based LDPC convolutional codes," in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Seoul, South Korea, Jun. - Jul. 2009, pp. 561-565 .

[35] O. Milenkovic, E. Soljanin and P. Whiting, "Asymptotic distributions of trapping sets in random regular LDPC code ensembles," *IEEE Trans. on Inf. Theory*, vol. 53, no. 1, pp. 39-55, Jan. 2007.

[36] T. Nozaki, K. Kasai and K. Sakaniwa, "Analysis of error floors of generalized non-binary LDPC codes over q-ary memoryless symmetric channels" in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Cambridge, MA, Jul. 2012, pp. 2341-2345.

[37] T. Nozaki, K. Kasai and K. Sakaniwa, "Analysis of stopping constellation distribution for irregular non-binary LDPC code ensemble" in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Saint-Petersburg, Russia, Jul. 2011, pp. 1101 - 1105.

[38] T. Nozaki, K. Kasai and K. Sakaniwa, "Error floors of non-binary LDPC codes" in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Austin, TX, Jun. 2010, pp. 729-733.

[39] C. Poulliat, M. Fossorier and D. Declercq, "Design of regular (2,d_c)-LDPC codes over $GF(q)$ using their binary images," *IEEE Trans. on Commun.*, vol. 56, no. 10, pp. 1626-1635, Oct. 2008.

[40] V. Rathi, "On the asymptotic weight and stopping set distribution of regular LDPC ensembles," *IEEE Trans. on Inf. Theory*, vol. 52, no. 9, pp. 4212-4218, Sep. 2006.

[41] V. Rathi and I. Andriyanova, "Some results on MAP decoding of non-binary LDPC codes over the BEC," *IEEE Trans. on Inf. Theory*, vol. 57, no. 4, pp. 2225-2242, Apr. 2011.

[42] L. Sassatelli and D. Declercq, "Nonbinary hybrid LDPC codes," *IEEE Trans. on Inf. Theory*, vol. 56, no. 10, pp. 5314-5334, Oct. 2010.

[43] V. Savin and D. Declercq, "Linear growing minimum distance of ultra-sparse non-binary cluster-LDPC codes," in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Saint-Petersburg, Russia, Jul. 2011, pp. 523-527.

[44] M. Shin, J. Kim and H. Song, "Generalization of Tanner's minimum distance bounds for LDPC codes," *IEEE Commun. Letters*, vol. 9, no. 3, pp. 240-242, Mar. 2005.

[45] V. Skachek, "Characterization of graph-cover pseudocodewords of codes over $\mathbb{F}_3$," in *Proc. of IEEE Inf. Theory Workshop (ITW)*, Dublin, Ireland, Sep. 2010, pp. 1-5.

[46] S. Song, B. Zhou, S. Lin and K. Abdel-Ghaffar, "A unified approach to the construction of binary and nonbinary quasi-cyclic LDPC codes based on finite fields," *IEEE Trans. on Commun.*, vol. 57, no. 1, pp. 84-93, Jan. 2009.

[47] F. Sun and H. van Tilborg, "Approaching capacity by equiprobable signaling on the Gaussian channel," *IEEE Trans. on Inf. Theory*, vol. 39, no. 5, pp. 1714-1716, Sep. 1993.

[48] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," IPN Progress Report, Tech. Rep. 42-154, Aug. 2003.

[49] A. Voicila, D. Declercq, F. Verdier, M. Fossorier and P. Urard,"Low-complexity decoding for non-binary LDPC codes in high order fields" *IEEE Trans. on Commun.*, vol. 58, no. 5, pp. 1365-1375, May 2010.

[50] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," Dec. 2005, Available at http://arxiv.org/abs/cs/0512078.

[51] P. O. Vontobel, "Counting in graph covers: a combinatorial characterization of the Bethe entropy function," submitted to *IEEE Trans. on Inf. Theory, Nov. 2010*. Available on ArXiv. .

[52] S. Zhao, X. Ma, X. Zhang and B. Bai, "A class of nonbinary LDPC codes with fast encoding and decoding algorithms," *IEEE Trans. on Commun.*, vol. 61, no. 1, pp. 1-6, Jan. 2013.

[53] B. Zhou, J. Kang, Y. Tai, S. Lin and Z. Ding, "High performance non-binary quasi-cyclic LDPC codes on Euclidean geometries," *IEEE Trans. on Commun.*, vol. 57, no. 5, pp. 1298-1311, May 2009.

[54] Uplink Coding for New TC Standard, Draft Orange Book, Consultative Committee for Space Data Systems (CCSDS), Newport Beach, CA, Apr. 2008.