BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 70(4), 2022, Article number: e141592
DOI: 10.24425/bpasts.2022.141592

ELECTRONICS, TELECOMMUNICATION AND OPTOELECTRONICS

The design of structured LDPC codes with algorithmic graph construction

Wojciech SUŁEK*

Silesian University of Technology, Gliwice, Poland

Abstract. Low-density parity-check (LDPC) codes are among the most effective modern error-correcting codes due to their excellent correction performance and highly parallel decoding scheme. Moreover, the nonbinary extension of such codes further increases performance in the short-block regime. In this paper, we review the key elements for the construction of implementation-oriented binary and nonbinary codes. These quasi-cyclic LDPC (QC-LDPC) codes additionally feature efficient encoder and decoder implementation frameworks. We then present a versatile algorithm for the construction of both binary and nonbinary QC-LDPC codes that have low encoding complexity and an optimized corresponding graph structure. Our algorithm uses a progressive edge growth algorithm, modified for QC-LDPC graph construction, and then performs an iterative global search for optimized cyclic shift values within the QC-LDPC circulants. Strong error correction performance is achieved by minimizing the number of short cycles, and cycles with low external connectivity, within the code graph. We validate this approach via error rate simulations of a transmission system model featuring an LDPC coder-decoder, digital modulation, and additive white Gaussian noise channels. The obtained numerical results validate the effectiveness of the proposed construction algorithm, with a number of constructed codes exhibiting either similar or superior performance to industry standard binary codes and selected nonbinary codes from the literature.

Key words: channel coding; low-density parity check codes; LDPC; nonbinary codes; quasi-cyclic codes.

1. INTRODUCTION

Error correction coding is a key operation within the physical layer of any data transmission system, and plays an important role in the recent advancements in timing and reliability of broadband mobile networks. Binary low-density parity-check (LDPC) [1] codes provide outstanding error correction performance, and approach the limit set by the Shannon capacity [2]. Such codes can be extended using the nonbinary Galois field. The resulting GF(q) codes [3] have been shown to further improve performance when applied to short or moderate data block lengths. The nonbinary LDPC coding is a highly active research area, covering topics such as code design, decoding algorithms, and hardware implementations [4–9].

The construction and optimization of LDPC codes is a key issue under consideration by LDPC system researchers. The primary research goal is to provide optimized error correction performance for the given use case. However, the code designs are often constrained to implementation-oriented approaches due to the requirement for the efficient implementation of hardware codecs (both encoders and decoders). Quasi-cyclic LDPC (QC-LDPC) codes [10, 11] are particularly suitable for hardware implementation [12, 13], as the block structure of the parity check matrix facilitates both the efficient routing of data paths within the decoder [7, 14, 15], as well as the generation of a compressed matrix representation. The parity check matrix **H** of a QC-LDPC code is composed of a set of submatrices. Each submatrix is either a zero matrix or a circulant permutation matrix (CPM). The error correction performance of QC-LDPC codes can be degraded due to the existence of short cycles within the code graph that represents the parity check matrix. This introduces dependence on the information exchanged during the iterative decoding process, thereby reducing its effectiveness. To avoid this, the code and corresponding graph must be designed appropriately.

Certain algebraic and geometric designs have the potential to support QC-LDPC code development methods [16, 17]. However, such approaches do not produce flexible code lengths and submatrix sizes, and the resultant regular degree distribution limits the waterfall correction performance in comparison with irregular distributions. Alternative approaches, which search for CPMs, can provide more design flexibility. The progressive edge growth (PEG) algorithm [18–20] is a classical search method that achieves excellent results, but only for general (non-QC) LDPC codes.

Important QC-LDPC code advancements have been made by designing graphs with large girth – the length of the shortest cycle within the code graph representation [5, 21, 22]. However, the girth optimization does not exhaust the possibilities of cycle existence minimization. Moreover, typical construction methods do not specifically account for the efficient encoding of **H** [23]. It is also known that, in addition to the shortest cycle length, further characteristics of existing cycles are important for code performance: the external cycle connectivity, represented by the extrinsic message degree (EMD) or approximate cycle EMD (ACE) [24, 25].

*e-mail: wojciech.sulek@polsl.pl

Manuscript submitted 2021-09-13, revised 2022-04-25, initially accepted for publication 2021-05-01, published in August 2022.

In previous publications, we proposed an algorithmic design method for certain constrained subclasses of QC-LDPC codes [6, 26]. We proposed the design of nonbinary, quasi-regular codes based on PEG [26], and a design method based on integer programming that is usable for short to moderate code lengths [6]. In this paper, we present a scalable non-exhaustive search algorithm that can be used for any block length.

Numerous binary LDPC codes already exist, and have been successfully applied in data communication standards such as WiFi (IEEE 802.11n [27]), WiMAX (IEEE 802.16 [28]), and 5G NR ([29]). It is likely that future deployments and non-standard use cases will require additional codes with different parameters such as block length, code rate, and possibly the utilization of nonbinary codes. Therefore, the requirement exists for both a structured review of code development and simple algorithmic methods for the construction of both binary and nonbinary LDPC codes with arbitrary parameters.

In this paper, we highlight the key features of the parity check matrix, for both binary and nonbinary implementations that produce a friendly LDPC code implementation on both the encoding and decoding sides. We further highlight the properties of the associated code graph that influence the code performance. We then provide a systematic, versatile design algorithm that accounts for all such features, and can construct any binary or nonbinary code that can be efficiently implemented.

The proposed algorithm consists of three stages. The first stage is based on PEG, modified to produce QC-LDPC code. Following this, the second stage searches iteratively for the shift values of respective CPMs. The goal of this algorithm is both to maximize the code graph girth, and minimize the number of short cycles of length greater than or equal to the girth. The cycle-length and cycle-number limits are configurable parameters; this allows the algorithm to be scaled easily for different use cases. The generated codes produce efficient encoding and decoding structures. The obtained codes are efficiently implementable and simulation results show that, in some cases, they provide a slight performance increase over standardized codes and those from the literature.

2. TRANSMISSION SYSTEM MODEL WITH LDPC ERROR CORRECTION CODING

We investigate LDPC coding over the field GF(q), where $q = 2^p$, $p \geq 1$. We use 2^X-ary digital modulations, corresponding to a transmission model with potentially high spectral efficiency. Specifically, we consider the following modulation schemes: binary phase shift keying (BPSK), for which $X = 1$; quadrature phase shift keying (QPSK), for which $X = 2$; and 2^X-quadrature amplitude modulation (2^X-QAM) with a square constellation, for $X \in \{4, 6, 8, \ldots\}$. We assume that an LDPC code over any field can be combined with a modulation of any order, which provides spectral efficiency flexibility. For clarification, Figures 1a and 1b show a generalized transmission model block diagram and a specific example using GF(8) and 16-QAM, respectively.

To investigate the coding system performance, we combine the transmitter model with a software model of an additive white Gaussian noise (AWGN) channel, a demodulator, and an LDPC decoder, which uses the belief propagation (BP) [1] and fast Fourier transform BP [4] algorithms for binary and nonbinary codes, respectively.

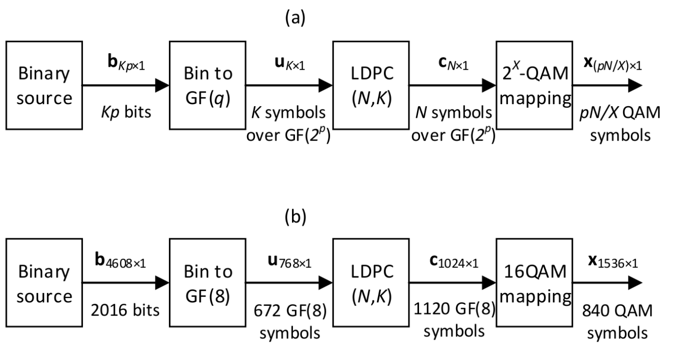

Fig. 1. Transmitter model: a) NB-LDPC coded 2^X-ary modulation; b) an example for $(1120, 672)$ code over GF(8) combined with 16-QAM

3. LDPC CODING DEFINITIONS

LDPC codes are a class of linear block error correcting codes. To encode an (N, K) code, $M = N - K$ redundant elements are inserted into the information vector $\mathbf{u} = \{u_1, u_2, \ldots, u_K\}$, giving the code vector $\mathbf{c} = \{c_1, c_2, \ldots, c_N\}$, where $c_i \in \mathrm{GF}(q) \forall i$, with $q = 2$ for binary codes, and $q > 2$ for nonbinary codes.

An (N, K) LDPC code is defined by its parity check matrix $\mathbf{H}_{M\times N}$, which is sparse and contains GF(q) elements. The decoder confirms a row vector $\hat{\mathbf{c}}$ of length N to be an error-free code word if it satisfies the parity check equation $\mathbf{H}\hat{\mathbf{c}}^T = \mathbf{0}_{M\times1}$ over GF(q). If this equation is not satisfied, the BP error correction decoding procedure is executed [1, 4].

3.1. Degree distribution

The error correction capabilities of a given LDPC code are dependent upon the properties of the corresponding code graph. A key parameter of the code is the node degree distribution of the code graph. This parameter is equivalent to the distribution of column weights of $\mathbf{H}$, which is given by $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_N]$, where λ_n denotes the weight (the number of nonzero elements) of the n-th column of $\mathbf{H}$. If λ_n values are constant for $n = 1, \ldots, N$, the code is a regular LDPC code; otherwise the code is irregular. The node degree distribution influences the waterfall region of the coding performance curve [30].

The first step of a typical LDPC code construction algorithm is to determine the distribution of column weights $\boldsymbol{\lambda}$. For nonbinary codes, a quasi-regular distribution is often favorable [31], for which $\lambda_n \in \{2, 3\} \forall n$. Effective quasi-regular distributions for different nonbinary codes can be found in previous works [32, 33].

3.2. Nonbinary structured codes

The parity check matrix $\mathbf{H}$ of a binary QC-LDPC code consists of a set of square circulant permutation matrices and square zero matrices. A CPM $\mathbf{P}^s$ is a $P \times P$ matrix obtained by cyclically shifting rows of the identity matrix $\mathbf{I}_{P\times P}$ to the right by

$(s \mod P)$ positions. The matrix $\mathbf{H}$ can be presented as follows for a general nonbinary case:

$$\mathbf{H} = \begin{bmatrix} z_{1,1}\mathbf{P}^{s_{1,1}} & z_{1,2}\mathbf{P}^{s_{1,2}} & \cdots & z_{1,J}\mathbf{P}^{s_{1,J}} \\ z_{2,1}\mathbf{P}^{s_{2,1}} & z_{2,2}\mathbf{P}^{s_{2,2}} & \cdots & z_{2,J}\mathbf{P}^{s_{2,J}} \\ \vdots & \vdots & \ddots & \vdots \\ z_{I,1}\mathbf{P}^{s_{I,1}} & z_{I,2}\mathbf{P}^{s_{I,2}} & \cdots & z_{I,J}\mathbf{P}^{s_{I,J}} \end{bmatrix}, \tag{1}$$

where the $s_{i,j} \in \{0,1,\ldots,P-1\}$ define the circulant shift values (CSVs) and $z_{i,j}$ indicates the positions of zero and nonzero submatrices within $\mathbf{H}$. For a binary code, $z_{i,j} \in \{0,1\}$; for a nonbinary code $z_{i,j} \in \mathrm{GF}(q)$ and additionally indicates the GF(q) coefficients of the submatrices.

The matrix $\mathbf{H}$ in equation (1) has a size of $IP \times JP$ and defines a binary QC-LDPC code over GF(2) or a nonbinary structured code over GF$(q > 2)$. The coefficients $z_{i,j}$ form the code base matrix, of size $I \times J$. Similarly, the $s_{i,j}$ form a matrix of CSVs $\mathbf{S}_{I\times J}$. Every QC-LDPC code is fully defined by the base matrix $\mathbf{Z}$, the submatrix size P, and the CSV matrix $\mathbf{S}$.

3.3. Dual diagonal structure for efficient encoding

The structured matrix $\mathbf{H}$ is appropriate for the implementation of fast, semi-parallel decoders. In contrast, efficient LDPC encoding, using the method introduced by Richardson and Urbanke [23], requires the parity check matrix to take an approximately lower triangular form. For a QC-LDPC code with submatrix size $P \times P$, this form can be represented as

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ \mathbf{C} & \mathbf{D} & \mathbf{E} \end{bmatrix}, \tag{2}$$

where $\mathbf{T}$ is a lower triangular matrix of size $(I-g)P \times (I-g)P$, submatrix $\mathbf{A}$ is of size $(I-g)P \times (J-I)P$, $\mathbf{B}$ is of size $(I-g)P \times gP$, $\mathbf{C}$ is of size $gP \times (J-I)P$, $\mathbf{D}$ is of size $gP \times gP$, $\mathbf{E}$ is of size $gP \times (I-g)P$, and g is a positive integer.

3.4. Code graph definitions

An alternative to the matrix $\mathbf{H}$ is the bipartite Tanner graph [1] representation. Such a representation has variable nodes corresponding to the data symbols associated with the columns of $\mathbf{H}$, check nodes corresponding to parity checks associated with the rows of $\mathbf{H}$, and edges corresponding to the positions of nonzero entries within $\mathbf{H}$. Similarly, a base graph representation of the base matrix $\mathbf{Z}$ can be defined.

Let the base graph be defined as $\mathcal{G} = (\mathcal{V}_c \cup \mathcal{V}_v, \mathcal{E})$. This is a bipartite graph, where $\mathcal{V}_c = \{c_1, c_2, \ldots, c_I\}$ is the set of check nodes representing nodes of $\mathbf{Z}$, $\mathcal{V}_v = \{v_1, v_2, \ldots, v_J\}$ is the set of variable nodes representing columns of $\mathbf{Z}$, and $\mathcal{E} \subseteq \mathcal{V}_c \times \mathcal{V}_v$ is the set of edges $\mathcal{E} = \{e_1, e_2, \ldots, e_T\}$. An edge $e_t = (c_i, v_j)$ belongs to $\mathcal{E}$ if and only if there exists an element of the base matrix $z_{ij} \neq 0$. The total number of edges in $\mathcal{G}$, denoted by T, is equal to the number of nonzero elements in $\mathbf{Z}$.

Furthermore, let the code graph representing $\mathbf{H}$ in equation (1) be denoted as $\mathcal{G}^{(P)} = (\mathcal{V}_c^{(P)} \cup \mathcal{V}_v^{(P)}, \mathcal{E}^{(P)})$. The check nodes in $\mathcal{V}_c^{(P)}$ are denoted as $c_{i,p}$, where $i \in \{1,\ldots,I\}$ is the index of the corresponding macro-row in $\mathbf{H}$, and $p \in \{0,\ldots,P-1\}$ is the row position within the macro-row. Similarly, the variable nodes in $\mathcal{V}_v^{(P)}$ are denoted as $v_{j,p}$, where $j \in \{1,\ldots,J\}$ is the index of the corresponding macro-column in $\mathbf{H}$, and $p \in \{0,\ldots,P-1\}$ is the column position within the macro-column. Hence, $\mathcal{G}^{(P)}$ contains the node sets $\mathcal{V}_c^{(P)} = \{c_{i,p} : 1 \le i \le I; 0 \le p \le P-1\}$ and $\mathcal{V}_v^{(P)} = \{v_{j,p} : 1 \le j \le J; 0 \le p \le P-1\}$.

The subset of edges within $\mathcal{G}^{(P)}$ that correspond to the submatrix $\mathbf{P}^{s_{i,j}}$ in equation (1) can be presented as

$$\left\{(c_{i,s_{i,j}}, v_{j,0}), (c_{i,s_{i,j}\oplus 1}, v_{j,1}), \ldots, (c_{i,s_{i,j}\oplus P}, v_{j,P})\right\}, \tag{3}$$
$$\text{where} \quad s \oplus p = (s+p) \bmod P.$$

Since any edge within this subset clearly defines the entire circulant subset, the proposed initial circulant insertion procedure searches for only a single new edge. This edge becomes a generating edge, and is used to generate the entire subset of P edges within a substructure representing a CPM.

3.5. PEG algorithm

The PEG algorithm [18, 19] is a greedy edge placement construction method that is applied to an initially edge-empty graph $\mathcal{G}$ to obtain a Tanner graph representation of an LDPC code. PEG inserts edges into the graph such that when a cycle subsequently arises, its length takes the maximum possible value within the current graph structure. The PEG algorithm is versatile and convenient, but does not support direct QC-LDPC code design.

4. CODE GRAPH STRUCTURAL PROPERTIES

For brevity, we henceforth denote the CSVs within $\mathbf{H}$ as the single-indexed s_t. By $\mathbf{s} = \{s_1, s_2, \ldots s_T\}$, we denote all CSVs corresponding to nonzero submatrices $\mathbf{P}^{s_{i,j}}$ in equation (1). For each $z_{i,j} \neq 0$ in equation (1), the corresponding $s_{i,j}$ is represented by a single-indexed $s_t \in \mathbf{s}$, corresponding to an edge $e_t = (c_j, v_j) \in \mathcal{E}$.

Now consider the key graph structural properties. The presence of short cycles within the code graph substantially influences the performance the corresponding LDPC code. A cycle of length $2l$ is a sequence of adjacent edges, which starts and ends at the same vertex, and satisfies the condition that no edge appears more than once in the sequence. Let such a cycle be denoted as $e_{t_1} \sim e_{t_2} \sim \cdots \sim e_{t_{2l}} \sim$, where $t_1, t_2, \ldots, t_{2l}$ index the constituent edges.

Two consecutive edges of a cycle in $\mathcal{G}^{(P)}$ correspond to distinct CPMs in $\mathbf{H}$. The CPMs must be located within the same macro-row or macro-column. Therefore, every cycle of length $2l$ in a code graph $\mathcal{G}^{(P)}$ is associated with an ordered series of nonzero CPMs:

$$\mathbf{P}^{s_{t_1}} \to \mathbf{P}^{s_{t_2}} \to \cdots \to \mathbf{P}^{s_{t_{2l}}} \to \mathbf{P}^{s_{t_1}}, \tag{4}$$

where $t_i \neq t_{i+1}$ for $1 \le i < 2l$ and consecutive submatrices $\mathbf{P}^{s_{t_i}}, \mathbf{P}^{s_{t_{i+1}}}$ are located within either the same macro-row or the same macro-column of $\mathbf{H}$ [11]. In general, the chain shown in equation (4) can contain repeated submatrices of index i

and j, where $j \geq i+4$. Due to the possibility of edge repetitions, the corresponding sequence of edges in the base graph $e_{t_1} \sim e_{t_2} \sim \cdots \sim e_{t_{2l}} \sim$ does not necessarily form a cycle. Rather, such a closed sequence of adjacent edges, with possible repetitions, is known as a closed walk [6, 34].

Every closed walk within the base graph specifies a chain of CPMs (4). However, a closed chain of CPMs does not necessarily indicate a cycle within code graph $\mathcal{G}^{(P)}$. The ordered CSV series $s_{t_1}, s_{t_2}, \cdots, s_{t_{2l}}$ defines the condition for the existence of a cycle within $\mathcal{G}^{(P)}$ corresponding to a closed walk in $\mathcal{G}$ [10, 11]. Using our presented notation, we now reformulate this condition.

The code graph $\mathcal{G}^{(P)}$ contains P length-$2l$ cycles corresponding to a length-$2l$ closed walk $e_{t_1} \sim e_{t_2} \sim \cdots \sim e_{t_{2l}} \sim$ in the base graph $\mathcal{G}$, if and only if the following condition is satisfied for the CSVs $s_{t_1}, s_{t_2}, \ldots, s_{t_{2l}}$ that correspond to the edges of the closed walk:

$$\sum_{k=1}^{2l} (-1)^{k-1} s_{t_k} \equiv 0 \mod P. \tag{5}$$

Figure 2 presents a base graph fragment that contains a length-4 cycle, a length-6 cycle, and a length-10 closed walk, indicated in red. Following the expansion of the graph into $\mathcal{G}^{(P)}$ with $P = 4$, the CSVs are such that no corresponding length-10 cycle exists within $\mathcal{G}^{(P)}$.

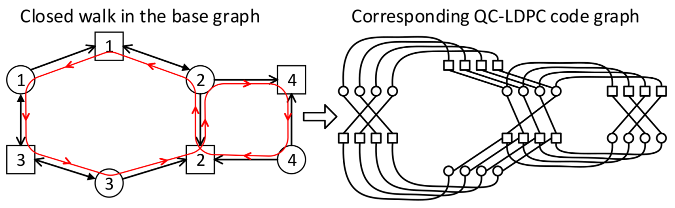
Closed walk in the base graph
Corresponding QC-LDPC code graph
1
2
3
4
1
2
3
4

Fig. 2. Example graph expansion

Cycles do not degrade performance uniformly. Results can be improved with the use of construction methods that account for the external connectivity of cycles. Cycle connectivity can be represented by EMD – the number of check nodes that are singly connected to the cycle variable node subset [24] – and ACE. The ACE of a length-$2l$ cycle is $\sum_i (\lambda_{t_i} - 2)$, where λ_{t_i} is the degree of the variable node v_{t_i} within the cycle [24].

The performance of an LDPC code with a message passing decoding is determined by its graph structure. In addition to cycles, more complex substructures can impact performance. Undesirable error floors in bit error rate curves have been shown to relate to trapping sets – a small number of variable nodes that lock one another into incorrect beliefs. Trapping sets of variable nodes are represented by clusters of short cycles in the corresponding graph. The existence of these clusters can be reduced by avoiding short cycles; particularly those with low external connectivity, as defined by the EMD and ACE metrics [24, 25, 35].

Based on the presented considerations, our **design objective** is to develop a method that minimizes the number of short cycles within the code graph of a structured QC-LDPC code. Accordingly, we present a versatile construction algorithm that can be used for any binary or nonbinary code. The resulting code is efficiently implementable in hardware due to the structured $\mathbf{H}$ and possible dual-diagonal form for efficient encoding.

5. ITERATIVE CONSTRUCTION ALGORITHM

This section presents our versatile design algorithm, which is based on the iterative optimization of the code graph. The process consists of three principal stages:

1) definition of the code base graph or base matrix;
2) base graph cyclic lifting, or expansion of the base matrix into the binary parity check matrix; and
3) in the case of nonbinary codes, replacement of the ones with non-zero entries over GF(q).

For binary non-QC LDPC code graphs, the base graph is typically designed using an algebraic or algorithmic construction method [19, 24]. Our method takes a different approach: we modify the classic PEG algorithm [18] such that during each iteration of the algorithm, P edges (rather than a single edge) are inserted into the graph. These P edges represent a single circulant in the parity check matrix. In this manner, a single procedure both produces a PEG-optimized base graph and initially optimizes the CSVs in the QC-LDPC matrix.

The algorithm is presented as Algorithm 1. In this form, the algorithm uses notation taken from the PEG publication [18], defined as follows. The set $\mathcal{N}^l_{v_{j,p}}$ contains all check nodes that can be reached by a subgraph spreading from symbol node $v_{j,p}$ within depth l. The complementary set is $\bar{\mathcal{N}}^l_{v_{j,p}} = \mathcal{V}^{(P)}_v \backslash \mathcal{N}^l_{v_{j,p}}$. The desired degree of symbol nodes in the j-th macro-column $v_{j,p}$ is denoted as λ_j. The degree vector $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_J]$ is an input parameter to the algorithm. The degree of check node $c_{i,p}$ under the current graph settings is denoted as $d(c_{i,p})$. This value changes as the algorithm progresses. The notation $x := y$ is defined as "x becomes y". For brevity, $\oplus$ denotes $\bmod P$ addition.

Once Algorithm 1 has obtained the initial CSVs in $\mathbf{S}$, they are further optimized to decrease the number of cycles in $\mathcal{G}^{(P)}$. For each cycle of length $2l$ and ACE parameter $\sum_i (\lambda_{t_i} - 2)$, we define a heuristic metric $w = 10^l + 10^{-\mathrm{ACE}}$. The algorithm attempts to minimize the total sum of metrics w over all closed walks within the base graph $\mathcal{G}$ that generate cycles within the optimized graph $\mathcal{G}^{(P)}$. That is cycles for which equation (5) is satisfied.

Formally, we define the optimization goal (or cost function that is minimized) C as

$$C = \mathbf{w} \cdot \mathbb{1}_{\mathcal{P}} \left(\mathbf{A} \cdot \mathbf{s}^T \right). \tag{6}$$

The variables within equation (6) are defined as follows.

- The ordered vector $\mathbf{s} = [s_1, s_2, \ldots, s_T]$ contains all elements within $\mathbf{S}$, in positions corresponding to nonzero elements within the base matrix $\mathbf{Z}$, that is, nonzero CSVs $s_{i,j}$.
- The matrix $\mathbf{A}$ is is of size $D \times T$, where D is the number of identified base graph closed walks. The rows of $\mathbf{A}$ are denoted as $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_D$, where every row $\mathbf{A}_d$ defines a condition (as shown in equation (5)) for a corresponding closed

Algorithm 1: Base graph design and circulant initialization

Input: Dimensions of base matrix (I, J), submatrix P, column weights $\boldsymbol{\lambda}$

Output: Code graph $(\mathcal{V}^{(P)}, \mathcal{E}^{(P)})$, base matrix $\mathbf{Z}_{I\times J}$, set of initialized circulants $\mathbf{S}_{I\times J}$

```
1  {Initialization:}
2  $(\mathcal{V}^{(P)} := \mathcal{V}_c^{(P)} \cup \mathcal{V}_v^{(P)},\ \mathcal{E}^{(P)} := \varnothing)$
3  $\mathbf{Z} := \mathbf{0}_{I\times J}$
4  $\mathbf{S} := \mathbf{0}_{I\times J}$
5  {Insert edges corresponding to the dual-diagonal part of H:}
6  for $(i,j) = (I,J), (I-1,J-1), \dots, (1, J-I+1)$ do
7      for $p = 0,1,\dots,P-1$ do
8          $\mathcal{E}^{(P)} := \mathcal{E}^{(P)} \cup (c_{i,p}, v_{j,p})$ {first diagonal}
9          $\mathcal{E}^{(P)} := \mathcal{E}^{(P)} \cup (c_{i-1,p}, v_{j,p})$ {second diagonal}
10 {Insert edges corresponding to the remaining part of H:}
11 for $j = J-I, J-I-1, \dots, 1$ do
12     {Initial edges for variable nodes $v_{j,p}$:}
13     Select check nodes with minimum $d(c_{i,0})$; from these
         randomly select the $c_{i_g,0}$. Then
14     for $p = 0$ to $P-1$ do
15         $\mathcal{E}^{(P)} := \mathcal{E}^{(P)} \cup (c_{i_g,p}, v_{j,p})$
16     Store the CSV: $s_{i_g,j} := 0$
17     {Subsequent edges for variable nodes $v_{j,p}$:}
18     for $k = 2,\dots,\lambda_j$ do
19         Track the current graph from variable node $v_{j,0}$ up to $l$
             edges deep, such that $\bar{\mathcal{N}}^{l}_{v_{j,0}} \neq \varnothing$ but $\bar{\mathcal{N}}^{l+1}_{v_{j,0}} = \varnothing$ or
             $\bar{\mathcal{N}}^{l+1}_{v_{j,0}} = \bar{\mathcal{N}}^{l}_{v_{j,0}}$.
20         From set $\bar{\mathcal{N}}^{l}_{v_{j,0}}$, select candidate nodes with minimum
             $d(c_{i,p})$; from these select candidate nodes with the
             greatest ACE of the shortest consequent cycles; from
             these randomly select $c_{i_g,p_g}$ as the generating edge
             $(c_{i_g,p_g}, v_{j,0})$.
21         {Place a generated subset of edges in the graph $\mathcal{G}^{(P)}$:}
22         for $p = 0$ to $P-1$ do
23             $\mathcal{E}^{(P)} := \mathcal{E}^{(P)} \cup (c_{i_g,p\oplus p_g}, v_{j,p})$
24         Store the CSV: $s_{i_g,j} := p_g$
25 From the obtained $\mathcal{G}^{(P)}$, determine the corresponding base matrix Z.
```

walk. Each element $A_{d,t}$ is equal to the number of encounters with edge e_t in the d-th closed walk. If edge e_t is not encountered, $A_{d,t} = 0$. For the existence of a cycle due to the d-th closed walk, equation (5) can then be reformulated as

$$\mathbf{A}_d \cdot \mathbf{s}^T \equiv 0 \mod P. \tag{7}$$

- The vector $\mathbf{w} = [w_1, w_2, \dots, w_D]$ contains weights corresponding to D closed walks. Each element is the cycle metric of the d-th length-$2l$ closed walk:

$$w_d = 10^{l(d)} + 10^{-\mathrm{ACE}(d)}. \tag{8}$$

- The function $\mathbb{1}_{\mathcal{P}}(\alpha)$ indicates that $\alpha \in \mathcal{P} = \{\dots, -2P, P, 0, P, 2P, \dots\}$, and can be defined for scalar α as

$$\mathbb{1}_{\mathcal{P}}(\alpha) = \begin{cases} 1, & \alpha \equiv 0 \mod P \\ 0, & \text{otherwise}. \end{cases} \tag{9}$$

For vector α, $\mathbb{1}_{\mathcal{P}}(\alpha)$ is a vector of the same size, to which equation (9) can be applied for each element of α. Making use of $\mathbb{1}_{\mathcal{P}}(.)$, condition in (7) can be reformulated as

$$\mathbb{1}_{\mathcal{P}}\left(\mathbf{A}_d \cdot \mathbf{s}^T\right) = 1. \tag{10}$$

The optimization goal given in equation (6) is therefore equal to the sum of metrics w_d for the closed walks which satisfy equation (10).

The CSV optimization algorithm, presented as Algorithm 2, acts iteratively on the rows $\mathbf{A}_d$ in $\mathbf{A}$. For every d for which equation (10) is satisfied, the algorithm constructs an additive modification $\boldsymbol{\Delta}_d$ to the vector $\mathbf{s}$ such that $\mathbb{1}_{\mathcal{P}}\left(\mathbf{A}_d \cdot (\mathbf{s}^T \oplus \boldsymbol{\Delta}_d^T)\right) = 0$, where $\oplus$ is an element-wise $\bmod P$ addition. Such modifications break the corresponding cycles, but can also create additional cycles. Therefore the total global cost C is calculated during each iteration. If this C is lower than the previous best solution, it is selected as the new best solution.

The maximum size of the closed walks considered by the algorithm is defined as $l_{\max}$, and the maximum number of close walks considered is defined as $D_{\max}$. These parameters are customizable, and allow the algorithm to be scaled to fit the requirements of varied use cases.

Algorithm 2: The optimization of CSVs in S

Input: Base matrix $\mathbf{Z}_{I\times J}$ (corresponding graph $\mathcal{G}$), submatrix size P, set of initialized circulants $\mathbf{S}_{I\times J}$, $D_{\max}$ and/or $l_{\max}$.

Output: Binary matrix $\mathbf{H}_{IP\times JP}$ consisting of $I\times J$ circulants of size $P\times P$

```
1  Arrange nonzero $s_{i,j}$ in $\mathbf{S}_{I\times J}$ into a vector $\mathbf{s}_{1\times T}$
2  {Identify closed walks in the base graph:}
3  $l := 2$
4  while $D < D_{\max}$ and $l < l_{\max}$ do
5      for $j = 1,2,\dots,J$ do
6          Track the base graph $\mathcal{G}$ from node $v_j$ up to depth $l$; for
             nodes met multiple times at depth-$l$, note a length-$2l$
             closed walks from the corresponding paths in the graph.
             Memorize the closed walks.
7      $l := l+1$
8      set $D$ to the total number of identified closed walks up to
         length-$2l$.
9  {Construct matrix A}
10 for $d = 1,\dots,D$ do
11     Create $\mathbf{A}_d$, the $d$-th row of A, corresponding to $d$-th identified
         closed walk in the memorized list. Calculate the metric
         (weight) $w_d$ according to equation (8)
12 $C_{\min} := \mathbf{w} \cdot \mathbb{1}_{\mathcal{P}}(\mathbf{A} \cdot \mathbf{s}^T)$
13 for $d = 1,\dots,D$ do
14     $\mathbf{s}' := \mathbf{s}$
15     if $\mathbb{1}_{\mathcal{P}}(\mathbf{A}_d \cdot \mathbf{s}^T) = 1$ then
16         {Search for modification of s with additive $\boldsymbol{\Delta}_d$:}
17         foreach $\boldsymbol{\Delta}_d : \mathbb{1}_{\mathcal{P}}\left(\mathbf{A}_d \cdot (\mathbf{s}^T \oplus \boldsymbol{\Delta}_d^T)\right) = 0$ do
18             $C := \mathbf{w} \cdot \mathbb{1}_{\mathcal{P}}\left(\mathbf{A} \cdot (\mathbf{s}^T \oplus \boldsymbol{\Delta}_d^T)\right)$
19             if $C < C_{\min}$ then
20                 $\mathbf{s}' := \mathbf{s} \oplus \boldsymbol{\Delta}_d$
21                 $C_{\min} := C$
22         $\mathbf{s} := \mathbf{s}'$
23 With obtained CSVs $\mathbf{s} = [s_1, s_2, \dots, s_T]$ and the base matrix Z,
     construct H as in (1).
```

Using Algorithms 1 and 2, the complete QC-LDPC construction algorithm is formulated and presented as Algorithm 3. We have successfully used the complete algorithm to construct both binary and nonbinary codes, over any GF(q), and with a wide range of code lengths N, coderates R, and submatrix sizes P.

Algorithm 3: The design of QC-LDPC code over GF(q)

```
Input: Code length $(N,K)$, submatrix size $P$, maximum column
       weight $\lambda_{\max}$, order $q$.
Output: Parity check matrix $\mathbf{H}_{(N-K)\times N}$ over GF($q$)
1 Set base matrix size to $(I \times J)$, where $I = (N-K)/P$ and $J = N/P$
2 Determine column degree distribution $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_J]$, where
    $0 \le \lambda_j \le \lambda_{\max}$, which can be done with known optimization
    methods [30] or taken from tables and charts of optimized
    distributions [30,32,33].
3 Construct initial code graph, according to Algorithm 1.
4 Construct binary parity check matrix, according to Algorithm 2
5 if NB code ($q > 2$) then
6     for $i$=1,2,...,$I$ do
7         For $\mathbf{z}_i$, that is $i$-th row of $\mathbf{Z}$, randomly choose and rearrange
            a sequence of coefficients over GF($q$), e.g., from
            a collection of sequences in Table 1, subject to a degree of
            the considered row.
8         Multiply consecutive nonzero submatrices $\mathbf{P}^{s_{i,j}}$ in a
            macro-row of $\mathbf{H}$ corresponding to the row $\mathbf{z}_i$ by the
            selected GF($q$) coefficient sequence.
```

For nonbinary codes, the final step of the construction algorithm is the selection of GF(q) values to replace the ones within the binary matrix. These values can be selected randomly from GF(q), which will produce satisfactory results in some cases. However, ultra-sparse nonbinary LDPC codes obtained using this approach tend to possess an error floor in the performance curve [36].

Poulliat *et al.* [36] propose a superior selection method, by using the binary images of the nonbinary parity-checks. Based on this method, and using an implemented search algorithm, we have computed a collection of GF(q) coefficient sequences for nonzero elements in the rows of the base matrix. Our collection is larger than that provided by Poulliat *et al.* [36], as it includes additional possibilities for the row degrees. This precomputed collection can be used in the final step of Algorithm 3 for nonbinary codes. For every row of $\mathbf{Z}$, a sequence of nonzero coefficients is chosen independently from the precomputed sequence for a given row degree, and rearranged arbitrarily. Table 1 shows an example sequence from the precomputed collection, with GF(8), GF(16), and GF(64), and check degrees up to 6.

Table 1
Example row coefficient sequences for several row degrees (d_c) [6,36]; the provided values are the exponents of the power representation α^i

d_c	GF(16)	GF(32)	GF(64)
3	{0, 4, 8}	{0, 6, 15}	{0, 15, 41}
	{0, 4, 9}	{0, 7, 15}	{0, 18, 44}
	{0, 5, 10}	{0, 5, 13}	{0, 16, 41}
4	{0, 3, 7, 11}	{0, 5, 16, 23}	{0, 9, 22, 37}
	{0, 2, 7, 11}	{0, 5, 17, 23}	{0, 6, 15, 41}
	{0, 2, 6, 10}	{0, 6, 15, 22}	{0, 7, 18, 44}
5	{0, 1, 4, 8, 11}	{0, 5, 10, 17, 22}	{0, 7, 18, 44, 53}
	{0, 1, 5, 7, 11}	{0, 5, 10, 18, 23}	{0, 7, 33, 42, 52}
	{0, 1, 5, 8, 11}	{0, 5, 11, 19, 24}	{0, 7, 33, 42, 55}
6	{0, 1, 4, 6, 8, 11}	{0, 5, 10, 15, 20, 25}	{0, 7, 33, 42, 49, 55}
	{0, 1, 4, 7, 8, 11}	{0, 4, 9, 14, 19, 24}	{0, 7, 18, 34, 44, 53}
	{0, 1, 4, 8, 11, 12}	{0, 4, 9, 14, 20, 25}	{0, 6, 13, 19, 43, 52}

6. RESULTS

To verify our code construction algorithms, we evaluated the error correction performance of binary and nonbinary QC-LDPC codes using Monte Carlo simulations. Numerical results were obtained using the system model developed in Matlab, with 2^X-ary modulation, an AWGN channel, and LLR-BP decoding algorithms. The results are presented as bit error rate or block error rate versus signal-to-noise ratio (SNR).

Figure 3 presents the simulated performance of the constructed binary QC-LDPC codes in comparison with two industrial standard irregular codes: WiFi [27] of rate $R = 1/2$ and WiMAX [28] of rate $R = 2/3$. We constructed irregular $(1296, 648)$ QC-LDPC dual-diagonal codes with the same distribution $\boldsymbol{\lambda}$ as WiFi code and a variety of submatrix sizes. Each constructed code displays a similar performance to that of the WiFi reference code, with a marginal performance improvement visible in the higher SNR region, particularly for the code with submatrix size $P = 18$. We also constructed irregular $(2304, 1536)$ QC-LDPC dual-diagonal codes with the same distribution $\boldsymbol{\lambda}$ as WiMAX code and a variety of submatrix sizes. As with the WiFi code, marginally improved performance is achieved in the higher SNR region.

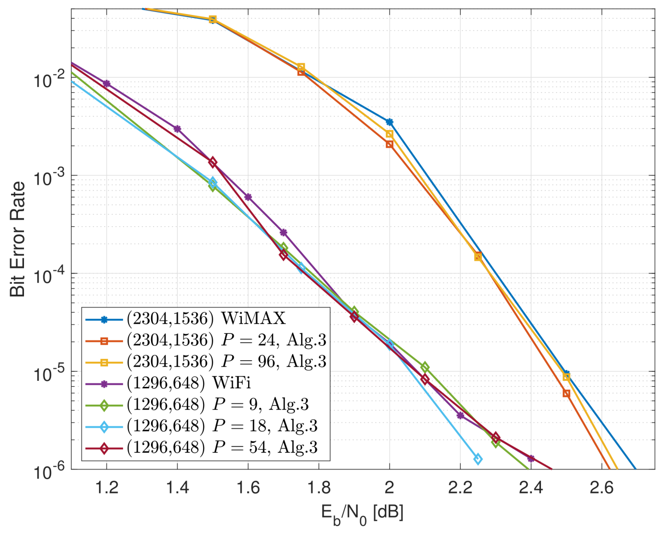

Fig. 3. Performance of the constructed binary QC-LDPC codes with different submatrix sizes, in comparison to two industrial standard irregular codes: WiFi 802.11n Rate-1/2 (1296,648) and WiMAX Rate-2/3 (2304,1536), combined with QPSK modulation

Figure 4 presents the simulated performance of the constructed nonbinary quasi-regular QC-LDPC codes over GF(8) with rate $R = 1/2$ and different block lengths, in comparison with girth-optimized GF(8) LDPC codes of the same block lengths taken from the work of Huang *et al.* [21]. In a similar manner to Huang *et al.* [21], we used the BPSK modulation model. For lengths $N = 492$ and $N = 756$ our codes display similar performance to that of the reference codes. For length $N = 1596$, our code noticeably outperforms the reference code.

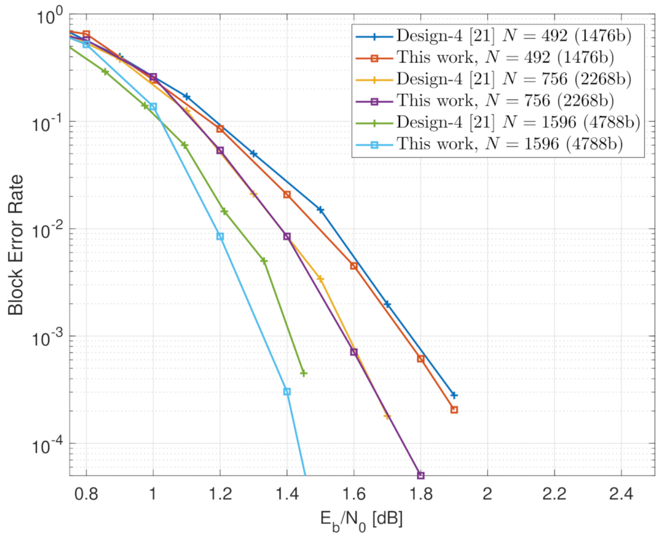

Fig. 4. Performance of the constructed QC-LDPC Rate-1/2 GF(8) codes with different block lengths, in comparison with girth-optimized GF(8) LDPC codes with the same lengths from [21]

Figure 5 presents the performance of the constructed QC-LDPC codes over different fields with rate $R = 0.6$ and block lengths selected such that GF(8), GF(16), and GF(64) are converted into the same number of bits. This demonstrates the coding gains that can be achieved by using nonbinary codes.

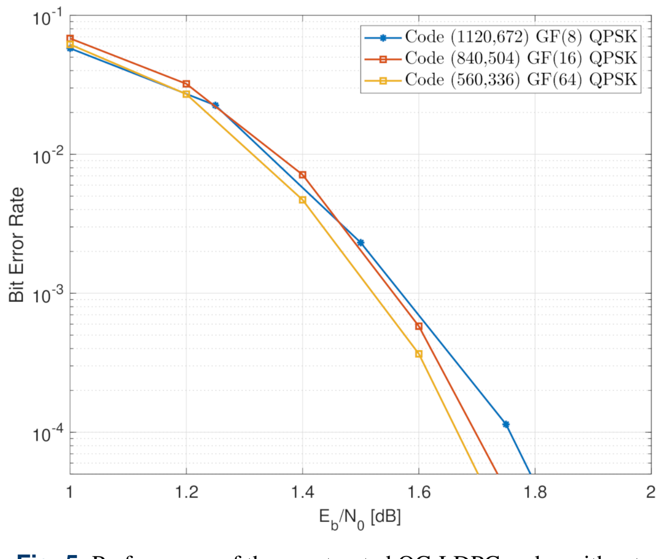

Fig. 5. Performance of the constructed QC-LDPC codes with rate $R = 0.6$ over different fields, combined with QPSK modulation

Finally, the results presented in Fig. 6 demonstrate how the combination of QC-LDPC coding and higher order modulation can be adjusted to a broad range of channel impairment levels, represented by E_b/N_0 in an AWGN model. Combinations with higher spectral efficiency – for example, (2048,1536) codes with rate $R = 0.75$ combined with 16-QAM – require a greater E_b/N_0 than the less efficient combinations of lower-rate codes and lower-order modulation. Figure 6 also demonstrates the gain that can be achieved with increasing block length: for every combination of code rate and modulation order, two different block lengths are presented; the greater block length provides an additional gain of approximately 0.2 dB.

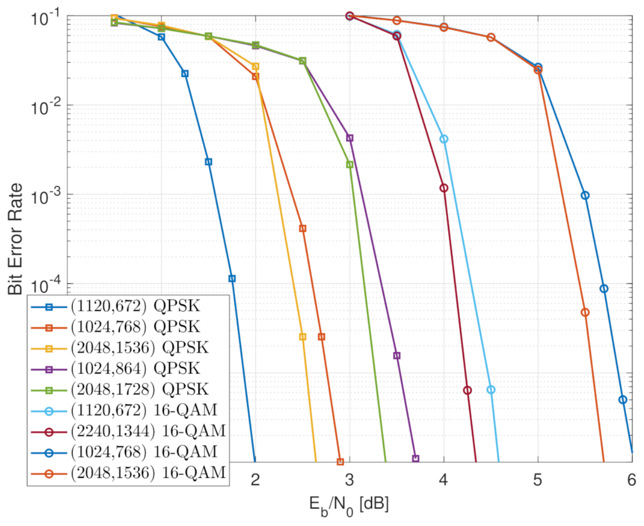

Fig. 6. Performance of the constructed nonbinary GF(8) codes with different rates R and block lengths N, combined with QPSK and QAM-16 modulation

7. CONCLUSIONS

In this paper, we proposed a vectorized formulation for the elimination of cycles from LDPC code graphs, and the definition of a global objective for graph optimization. From this formulation, we developed a versatile QC-LDPC code construction algorithm. The algorithm can construct both binary and nonbinary codes, with a wide range of block lengths, code rates, and irregular distributions. The parity check matrix is structured to provide efficient decodability, and the dual diagonal structure provides efficient encodability.

Numerical results validate the effectiveness of the proposed construction method. The constructed codes exhibit similar or improve performance when compared to industry standard binary codes and codes selected from the literature. We conclude that the developed method can be used to construct strongly performing codes for a broad range of future deployment requirements and non-standardized usage of implementation-oriented LDPC coding systems.

ACKNOWLEDGEMENTS

This work was supported by the Polish Ministry of Science and Higher Education funding for statutory activities.

REFERENCES

[1] D.J.C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 399–431, 1999.

[2] J. Pawelec, "Shannon theory. Myths and reality," *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 56, no. 3, pp. 247–251, 2008.

[3] M.C. Davey and D. MacKay, "Low-Density Parity Check Codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, 1998.

[4] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes Over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, 2007.

[5] I.E. Bocharova, B. Kudryashov, and R. Johannesson, "Searching for Binary and Nonbinary Block and Convolutional LDPC Codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 163–183, 2016.

[6] W. Sulek, "Protograph Based Low-Density Parity-Check Codes Design with Mixed Integer Linear Programming," *IEEE Access*, vol. 7, pp. 1424–1438, 2019.

[7] W. Sulek, "Non-binary LDPC Decoders Design for Maximizing Throughput of an FPGA Implementation," *Circuits Syst. Sig. Process.*, vol. 35, no. 11, pp. 4060–4080, 2016.

[8] V. Wijekoon, E. Viterbo, Y. Hong, R. Micheloni, and A. Marelli, "A Novel Graph Expansion and a Decoding Algorithm for NB-LDPC Codes," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1358–1369, 2020.

[9] T.X. Pham, T.N. Tan, and H. Lee, "Minimal-Set Trellis Min-Max Decoder Architecture for Nonbinary LDPC Codes," *IEEE Trans. Circuits Syst. II*, vol. 68, no. 1, pp. 216–220, 2021.

[10] M.P.C. Fossorier, "Quasi-Cyclic Low-Density Parity-Check Codes from Circulant Permutation Matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, 2004.

[11] S. Myung, K. Yang, and J. Kim, "Quasi-Cyclic LDPC Codes for Fast Encoding," *IEEE Trans. Inf. Theory*, vol. 51, no. 8, pp. 2894–2901, 2005.

[12] W. Sulek, "Pipeline processing in low-density parity-check codes hardware decoder," *Bull. Pol. Acad Sci. Tech. Sci.*, vol. 59, no. 2, pp. 149–155, 2011.

[13] H. Cui, F. Ghaffari, K. Le, D. Declercq, J. Lin, and Z. Wang, "Design of High-Performance and Area-Efficient Decoder for 5G LDPC Codes," *IEEE Trans. Circuits Syst. I*, vol. 68, no. 2, pp. 879–891, 2021.

[14] T.T. Nguyen-Ly, V. Savin, D. Declercq, F. Ghaffari, and O. Boncalo, "Analysis and Design of Cost-Effective, High-Throughput LDPC Decoders," *IEEE Trans.VLSI Syst.*, vol. 26, no. 3, pp. 508–521, 2018.

[15] M. Li, V. Derudder, K. Bertrand, C. Desset, and A. Bourdoux, "High-Speed LDPC Decoders Towards 1 Tb/s," *IEEE Trans. Circuits Syst. I*, vol. 68, no. 5, pp. 2224–2233, 2021.

[16] G. Zhang, R. Sun, and X. Wang, "New Quasi-Cyclic LDPC Codes with Girth at Least Eight based on Sidon Sequences," in *7th International Symposium on Turbo Codes and Iterative Information Processing*, Gothenburg, Sweden, August 2012, pp. 31–35.

[17] J. Li, K. Liu, S. Lin, and K. Abdel-Ghaffar, "Algebraic Quasi-Cyclic LDPC Codes: Construction, Low Error-Floor, Large Girth and a Reduced-Complexity Decoding Scheme," *IEEE Trans. Commun.*, vol. 62, no. 8, pp. 2626–2637, 2014.

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

[19] D. Vukobratovic and V. Senk, "Generalized ACE Constrained Progressive Edge-Growth LDPC Code Design," *IEEE Commun. Lett.*, vol. 12, no. 1, pp. 32–34, 2008.

[20] X. He, L. Zhou, and J. Du, "PEG-Like Design of Binary QC-LDPC Codes Based on Detecting and Avoiding Generating Small Cycles," *IEEE Trans. Commun.*, vol. 66, no. 5, pp. 1845–1858, 2018.

[21] J. Huang, L. Liu, W. Zhou, and S. Zhou, "Large-Girth Nonbinary QC-LDPC Codes of Various Lengths," *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3436–3447, 2010.

[22] A. Tasdighi, A.H. Banihashemi, and M.-R. Sadeghi, "Efficient Search of Girth-Optimal QC-LDPC Codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1552–1564, 2016.

[23] T.J. Richardson and R.L. Urbanke, "Efficient Encoding of Low-Density Parity-Check Codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 638–656, 2001.

[24] T. Tian, C. Jones, J.D. Villasenor, and R.D. Wesel, "Selective Avoidance of Cycles in Irregular LDPC Code Construction," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1242–1247, 2004.

[25] X. Zheng, F.C.M. Lau, and C.K. Tse, "Constructing Short-Length Irregular LDPC Codes with Low Error Floor," *IEEE Trans. Commun.*, vol. 58, no. 10, pp. 2823–2834, 2010.

[26] W. Sulek, "Nonbinary Quasi-Regular QC-LDPC Codes Derived From Cycle Codes," *IEEE Commun. Lett.*, vol. 20, no. 9, pp. 1705–1708, 2016.

[27] *Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput*, IEEE Std. 802.11n, 2009.

[28] *Local and Metropolitan Area Networks Part 16: Air Interface for Broadband Wireless Access Systems*, IEEE Std. 802.16, 2012.

[29] *5G; NR; Multiplexing and channel coding (3GPP TS 38.212 version 15.2.0 Release 15)*, ETSI Std. TS 138 212 V15.2.0, 2018.

[30] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[31] G. Li, I.J. Fair, and W.A. Krzymien, "Density Evolution for Nonbinary LDPC Codes Under Gaussian Approximation," *IEEE Trans. Inf. Theory*, vol. 55, no. 3, pp. 997–1015, 2009.

[32] B. Rong, T. Jiang, X. Li, and M. R. Soleymani, "Combine LDPC Codes Over GF(q) With q-ary Modulations for Bandwidth Efficient Transmission," *IEEE Trans. Broadcast.*, vol. 54, no. 1, pp. 78–84, 2008.

[33] W. Sulek and M. Kucharczyk, "Column Weights Optimization for Semi-Regular Nonbinary LDPC Codes," in *Proc. (IEEE) 38th International Conference on Telecommunications and Signal Processing (TSP)*, Prague, Czech Republic, July 9-11 2015, pp. 172–176.

[34] R. Diestel, *Graph Theory*. Berlin: Springer-Verlag, 2006.

[35] G. Han, Y.L. Guan, and L. Kong, "Construction of Irregular QC-LDPC Codes via Masking with ACE Optimization," *IEEE Commun. Lett.*, vol. 18, no. 2, pp. 348–351, 2014.

[36] C. Poulliat, M. Fossorier, and D. Declercq, "Design of Regular (2,dc)-LDPC Codes over GF(q) Using Their Binary Images," *IEEE Trans. Commun.*, vol. 56, pp. 1626–1635, October 2008.