# Cooperative Decoder Design for Non-binary LDPC Code with Coefficients Selection

Yang Yu, Wen Chen, *Senior Member, IEEE*, Jun Li, *Member, IEEE*, and Benoît Geller, *Senior Member, IEEE*

***Abstract*—In this paper we design novel decoders for non-binary low density parity check (LDPC) codes. For a non-binary LDPC code $\mathcal{C}$ over the field $\mathbb{F}_q$ of size $q$ for some $q > 0$, we propose two novel cooperative decoders, each composed of a binary component decoder and a $q$-ary component decoder in a concatenated manner, to obtain excellent decoding performance. Specifically to reduce the complexity of the cooperative decoders, we design a hybrid $q$-ary component decoder. Then, we propose an algorithm to construct the parity check matrix to eliminate the bit-level cycles. Simulations show that the decoding performance of the proposed algorithm approaches the capacity limit within $0.2$dB at BER$= 10^{-4}$.**

***Index Terms*—Non-binary LDPC, EXIT chart, binary erasure channel, binary Gaussian channel.**

## I. Introduction

Low density parity check (LDPC) codes are mostly designed through the construction of their Tanner graphs [1]. Excellent LDPC codes can be always associated with sparse Tanner graphs in which the small decoding cycles are eliminated. Besides, investigation over finite field, $\mathbb{F}_q$ of size $q = 2^p$ for some $p > 0$, shows that excellent $q$-ary LDPC codes have much sparser Tanner graphs than binary codes. Since small length cycles are more easily to be eliminated, the decoding of non-binary LDPC codes with belief propagation (BP) decoding is easier to achieve high performance [2].

The reason why non-binary LDPC codes have much sparser Tanner-graph has been investigated in [3], [4], where the authors show that the average variable node degree of the threshold-optimized non-binary codes will tend to 2 as the size of the field tends to infinity [5]–[7]. That is, the larger the size of the field the sparser the Tanner-graph. In [5], [6], the authors investigate a particular non-binary code, *i.e.* cycle code, whose variable node degree is 2. The Cayley-graph-based analysis is performed to optimize the encoding and decoding processes of this code in [5]. In [6], complexity-reduced coefficients selection methods are given by using the bit-level representations to optimize the symbol-level performance of the non-binary cycle LDPC codes.

On the other hand, optimal decoders for non-binary LDPC codes require potentially higher decoding complexity as the messages passing through each iterations are vectors. A straight-forward implementation of sum-product algorithm of binary LDPC codes for $q$-ary LDPC codes requires computational complexity of $O(q^2)$ for each check-sum operation. The Fourier transform $q$-ary sum-product algorithm (FFT-QSPA) reduces it to $O(q \log q)$ [4], [8], [9]. As to binary erasure channel (BEC), this complexity is further reduced to $O(q)$ by introducing a simplex constraint in the check nodes [10], [11]. In addition, based on the decoding error probability, [7], [12] show that the minimal decoding complexity exists if the LDPC codes are constructed with proper chosen degree distributions.

Moreover, one essential principle for optimal decoding of non-binary LDPC codes is the *independence* assumption. That is, there is no loop (cycle) within the iterations of the decoding process. As to non-binary LDPC codes, the decoding cycles usually exist in the $q$-ary decoders. Thus, cycle eliminating algorithms for non-binary LDPC codes proposed in [2], [5], [6], [13] are designed to optimize the symbol-level performance by dealing with cycles within the $q$-ary decoders.

## II. Construction of Equivalent Binary Codes from Non-binary LDPC Codes

Assume that the non-binary LDPC code is transmitted over binary channels. Consider the finite field $\mathbb{F}_q$ of size $q = 2^p$. Let $f(x) = f_0 + f_1 x + f_2 x^2 + ... + f_{p-1} x^{p-1} + f_p x^p \in \mathbb{F}_2[x]$ be a primitive polynomial over $\mathbb{F}_2$. Then the finite field $\mathbb{F}_q$ is generated by the root of the primitive polynomial $f(x)$, *i.e.* $\mathbb{F}_q = \{0, 1, \alpha, \ldots, \alpha^{q-2}\}$ where $\alpha$ is a root of $f(x)$ and is also called the cyclic generator of $\mathbb{F}_q$. If $\mathbb{F}_q$ is endowed with a binary vector space structure, then each element $x$ in $\mathbb{F}_q$ can be represented as a binary vector $\bar{\mathbf{x}} = (\bar{x}^{(0)}, \bar{x}^{(1)}, \ldots, \bar{x}^{(p-1)})^T$, that is $x = \sum_{i=0}^{p-1} \bar{x}^{(i)} \alpha^i$.

In order to transform the $q$-ary parity-check matrix into its binary form, we first define $\mathbf{A}$ to be the companion matrix of $f(x)$ over $\mathbb{F}_2$. Then, $\mathbb{F}_q = \{0, \alpha^i, 0 \leqslant i \leqslant q-2\}$ is isomorphic with the set $\{\mathbf{0}, \mathbf{A}^i, 0 \leqslant i \leqslant q-2\}$, *i.e.* $\mathbb{F}_q \cong \{\mathbf{0}, \mathbf{A}^i, 0 \leqslant i \leqslant q-2\}$ with $\alpha^i \leftrightarrow \mathbf{A}^i$. As a result, each entry in the $q$-ary parity-check matrix can be replaced by its binary matrix representation. Moreover, the multiplication of $\alpha^i$ and $x$ is exact the multiplication of $\mathbf{A}^i$ and $\bar{\mathbf{x}}$.

Based on the above facts, we now define the equivalent binary LDPC code corresponding to a non-binary LDPC code.

*Definition 1:* Consider the non-binary LDPC code $\mathcal{C}$ defined over the parity check matrix $\mathbf{H} = (h_{i,j})_{M \times N}, h_{i,j} \in \mathbb{F}_q$. Let $\bar{\mathcal{C}}$ be the *equivalent binary LDPC code* of $\mathcal{C}$ with the *equivalent binary parity check matrix* $\bar{\mathbf{H}} = (\mathbf{A}_{m,n})_{M \times N}$, and $\bar{\mathbf{x}} = (\bar{\mathbf{x}}_1^T, \bar{\mathbf{x}}_2^T, \ldots, \bar{\mathbf{x}}_N^T)$ be the binary vector representation of

Yang Yu, and Wen Chen are with Network Coding and Transmission Laboratory, Shanghai Jiao Tong University, Shanghai, 200240, China, e-mail: {yuyang83, wenchen}@sjtu.edu.cn.

Jun Li is with School of Electrical and Information Engineering, The University of Sydney, NSW, 2006, Australia. e-mail: jun.li1@sydney.edu.au.

Benoît Geller is with the UEI ENSTA, ParisTech, 75015 Paris, France, e-mail: benoit.geller@ensta-paristech.fr.

This work is supported by the National 973 Project #2012CB316106, by the NSF China #61161130529.

the codeword $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \mathbb{F}_q^N$ in $\mathcal{C}$. Then the binary LDPC code $\bar{\mathcal{C}}$ corresponding to $\mathcal{C}$ is defined by

$$\begin{aligned}\bar{\mathcal{C}} &\triangleq \ker(\bar{\mathbf{H}}) \subset \mathbb{F}_2^{N\times p} \\ &= \left\{(\bar{\mathbf{x}}_1^T, \bar{\mathbf{x}}_2^T, \ldots, \bar{\mathbf{x}}_N^T) | \sum_{n=1}^{N} \mathbf{A}_{m,n}\bar{\mathbf{x}}_n = 0, \forall m = 1, \ldots, M\right\}\end{aligned}$$

where $\mathbf{A}_{m,n}$ is the binary matrix representation of $h_{m,n}$. In the following, we denote $\mathcal{G}$ as the Tanner-graph of $\mathcal{C}$, $\bar{\mathcal{G}}$ as the Tanner-graph of $\bar{\mathcal{C}}$, and $\mathbf{A}_{m,n}$ as the *matrix label* along each edge in the Tanner-graph $\mathcal{G}$.

*Example 1:* Let $\mathcal{C}$ be an 8-ary LDPC code and assume that the $m^{th}$ row of its parity-check matrix $\mathbf{H}$ has 3 non-zero entries. Then for the corresponding check node $m$ in $\mathcal{G}$, we get a symbol node set $\{u, v, w\}$. Moreover, suppose that the parity check relation in the check node $m$ can be expressed as

$$\alpha^7 u + \alpha^5 v + \alpha^4 w = 0, \tag{1}$$

then we have its equivalent binary representation as

$$\mathbf{A}^7 \underbrace{\begin{pmatrix}\bar{u}_0\\ \bar{u}_1\\ \bar{u}_2\end{pmatrix}}_{\bar{\mathbf{u}}} + \mathbf{A}^5 \underbrace{\begin{pmatrix}\bar{v}_0\\ \bar{v}_1\\ \bar{v}_2\end{pmatrix}}_{\bar{\mathbf{v}}} + \mathbf{A}^4 \underbrace{\begin{pmatrix}\bar{w}_0\\ \bar{w}_1\\ \bar{w}_2\end{pmatrix}}_{\bar{\mathbf{w}}} = 0, \tag{2}$$

where $\bar{\mathbf{u}}$, $\bar{\mathbf{v}}$ and $\bar{\mathbf{w}}$ are respectively the binary vector representations of the three symbol nodes, which are called *bit-vector nodes* in $\bar{\mathcal{G}}$. Each element in $\bar{\mathbf{u}}$, $\bar{\mathbf{v}}$, or $\bar{\mathbf{w}}$ is called *bit node* in $\bar{\mathcal{G}}$. Let $\bar{\mathbf{m}} = (\bar{m}_1, \bar{m}_2, \bar{m}_3)$ be the vector representation of the check node $m$, which is called *check-vector node* in $\bar{\mathcal{G}}$. Each element in $\bar{\mathbf{m}}$ is called *constituent check node* in $\bar{\mathcal{G}}$.

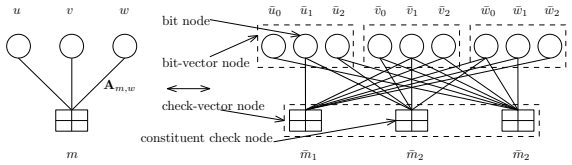


Fig. 1. Equivalent binary check graph.

## III. Novel Decoders Design with Excellent Decoding Performance

In this section, we propose two different types of decoders, namely, the serially cooperative decoder and the paralleled cooperative decoder, to decode a non-binary LDPC code with excellent decoding performance. Each proposed cooperative decoder has two component decoders, namely, a binary SP decoder corresponding to the code $\bar{\mathcal{C}}$ and a $q$-ary SP decoder corresponding to the code $\mathcal{C}$. By performing the binary and $q$-ary decoder in a concatenated way, a $q$-ary symbol is decoded in both bit-vector form (in the binary decoder) and scalar form (in the $q$-ary decoder).

Unlike the classic concatenated codes and their generalizations [16], [17], encoding of the equivalent code $\bar{\mathcal{C}}$ does not need to be concatenated with the encoding of $\mathcal{C}$. Instead, we only need to encode $\mathcal{C}$ over $\mathbb{F}_q$ (the Tanner graph $\mathcal{G}$ can be designed by using progressive edge growth (PEG) [18] or approximate cycle extrinsic message degree (ACE) [19]). Then the equivalent code $\bar{\mathcal{C}}$ is obtained by replacing the $q$-ary symbols and labels with their binary vector and binary matrix representations, respectively. On the decoder side, the binary and $q$-ary decoders are cooperated at the same level, *i.e.* no matter what manner we choose to exchange the messages between them, they both deal with the overall Tanner graph.

We will also show that the proposed decoders are able to deal with different types of error, such as burst error, random error, and so on. In addition, since the $q$-ary decoder is immune to the bit-level cycles, the idea of using two cooperative component decoders is of more practical importance especially when the Tanner graph $\bar{\mathcal{G}}$ is constructed with cycles. In the following we first design a hybrid $q$-ary decoder (HQD) for $\bar{\mathcal{C}}$ which will reduce the computational complexity of the cooperative decoders. Then we illustrate the two cooperative decoders.

### A. Hybrid $q$-ary Decoder

In [20], we have given a hybrid hard-decision decoder for non-binary LDPC codes over BEC channel. In addition, decoding of each coded symbol by the hard-decision decoder is done separately over a dynamic Tanner-graph. Here, we design a hybrid soft-decision decoder for the non-binary LDPC over AWGN channel. Instead of dealing with the dynamic Tanner-graph with graph operations, decoding of each coded symbol by the soft-decision decoder is performed over $\mathcal{G}$. In addition, decoding the bit nodes with the soft-decision decoder can be done both in serial or parallel manner while the hard-decision decoder decodes the bit nodes in serial manner.

More specifically, the proposed Hybrid $q$-ary decoder (HQD) is basically a binary SP decoder with matrix inverse operations. That is, when decoding a $q$-ary symbol, by viewing this $q$-ary symbol as a binary vector, the HQD decodes each bit over a set of binary parity check sub-matrices based on the matrix inverse operations over the corresponding rows of the equivalent binary parity check matrix.

To have a better understanding, we start with the *Example 1* described in the previous section. If the AWGN channel is adopted, and $\{u, v, w\}$ is the symbol set. The equivalent parity check matrix for Eq. (2) is $\bar{\mathbf{H}}_m = (\mathbf{A}^2, \mathbf{A}^4, \mathbf{A}^5)$. The coded bits are $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{w}})$. When decoding a binary vector, say $\bar{\mathbf{w}}$ in Eq. (2), we remove the dependence within the bits in $\bar{\mathbf{w}}$ by matrix inverse operation over the coefficient matrix $\bar{\mathbf{H}}_m$. The resulting parity-check matrix is $\bar{\mathbf{H}}'_{mw} = (\mathbf{A}^4, \mathbf{A}^6, \mathbf{I})$. Note that, in $\bar{\mathbf{H}}'_{mw}$, the matrix coefficient of $\bar{\mathbf{w}}$ is an identity matrix. As a result, the one-step decoding tree regarding $\bar{\mathbf{w}}$ is cycle-free. That is, each bit node of the bit-vector node $\bar{\mathbf{w}}$ in $\bar{\mathcal{G}}$ is only connected to a single constituent check node in one check-vector node. Then the binary SP algorithm over local decoding tree [21] can be adopted to decode the bit nodes of $\bar{\mathbf{w}}$ for one iteration. Specifically, in the check node $m$, the bit-level log-likelihoods (LLRs) are calculated according to $\bar{\mathbf{H}}'_{mw}$ and then sent to the bit nodes in $w$, respectively. To decode $u$ and $v$, we have $\bar{\mathbf{H}}'_{mu}$ and $\bar{\mathbf{H}}'_{mv}$. As a result, bit-level LLRs sent to different symbol nodes are calculated according to different parity check matrices in $m$. The bit-level LLRs sent to $m$ from $w$ are calculated in the following steps. First, let $M(w)$ be the

check nodes set connected to $w$ and $\{\bar{\mathbf{H}}'_{mw}, m \in M(w)\}$ be the corresponding parity check matrices set. Then, in $w$, we place all the $\bar{\mathbf{H}}'_{mw} \in M(w)$ vertically to form a new parity check matrix $\bar{\mathbf{H}}'_w$. At last, the bit-level LLRs sent to $m$ from $w$ are calculated according to $\bar{\mathbf{H}}'_w$. Similarly, bit-level LLRs sent to $m$ from $u$ and $v$ are calculated according to $\bar{\mathbf{H}}'_u$ and $\bar{\mathbf{H}}'_v$, respectively. Based on the above facts, we propose the hybrid $q$-ary decoder by extending the above local decoding process to the overall $\mathcal{G}$.

Step 1 : Let $m_1, \ldots, m_L$ be the row numbers of $L$ non-zero entries of the $n^{th}$ column of $\mathbf{H}$. For each $m \in \{m_1, \ldots, m_L\}$, we have the corresponding row vector $\mathbf{H}_m$. If we perform matrix inverse operation over $\bar{\mathbf{H}}_m$ to diagonalize the $n^{th}$ matrix-element. Then we get the parity check matrix $\bar{\mathbf{H}}'_{mn}$. Considering all the check nodes connected to the symbol node $n$ in $\mathcal{G}$, the binary parity-check matrix regarding $n$ can be expressed as

$$\bar{\mathbf{H}}'_n = (\bar{\mathbf{H}}'^T_{m_1 n}, \bar{\mathbf{H}}'^T_{m_2 n}, \ldots, \bar{\mathbf{H}}'^T_{m_L n})^T.$$

Then, in each symbol node and check node, we associate them with their corresponding binary parity-check matrices $\bar{\mathbf{H}}'_n$ and $\{\bar{\mathbf{H}}'_{mn}, n \in N(m)\}$, respectively, where $N(m)$ is the symbol nodes set connected to $m$.

Step 2 : In each check node $m$ with parity check matrices set $\{\bar{\mathbf{H}}'_{mn}, n \in N(m)\}$, we calculate the bit-level LLRs [22] and then send them to each symbol node $n \in N(m)$ according to the parity check matrix $\bar{\mathbf{H}}'_{mn}$.

Step 3 : In each symbol node $n$, let $M(n)$ be check node set connected to $n$. For each $m \in M(n)$, we calculate the bit-level LLRs according to $\bar{\mathbf{H}}'_n$ and then send them to each check node $m \in N(m)$.

Step 4 : Go to Step 2 until the decision threshold of the LLR value or the maximum number of iterations is reached.

HQD performs bit-level decoding process to decode the non-binary symbols, which makes the HQD equivalent to the $q$-ary maximum *a posteriori* probability (MAP) decoder in general. The concatenations of the HQD and the binary SP decoder lead to the proposed cooperative decoders. The cooperative decoders are able to deal with different types of errors under different channel conditions. The reasons are given in the following. After the matrix inverse operation, every bit node in a bit-vector node is in different binary parity-check functions. For burst error, decoding of the bit node will not be affected by other erroneous bit nodes. Moreover, the same bit node is decoded within different parity check functions under binary decoder and HQD. A bit-error in one component decoder may be corrected in the other component decoder.

The computational complexity for each check-vector-sum operation [8] in the cooperative decoders with HQD is constrained by the complexity of both component decoders. First, we notice that the matrix inverse operation over $\bar{\mathbf{H}}_m$ has a computational complexity of $O(\log^2 q)$ in general (because the matrix labels are $\log q \times \log q$ matrices). In addition, unlike the $q$-ary decoder in [4], HQD use bit-level LLRs to decode each bit node other than the symbol-level LLR vectors (LLRVs). Then, to decode each bit node, the computational complexity of HQD for each check-vector-sum relies linearly on the number of its constituent check nodes, *i.e.* $O(\log q)$. As a result, the computational complexity regarding all the bit nodes in a bit-vector node is $O(\log^2 q)$. Moreover, the processing complexity of the binary component decoder for each bit node is also $O(\log q)$. Then the overall complexity of the cooperative decoder with HQD is dominated by $O(\log^2 q)$ which is smaller than $O(q)$ for large $q$.

### B. Serially Cooperative Decoder

In the serially cooperative decoder (SCD), the binary decoder [23], [24] and the $q$-ary decoder [2], [4] are implemented in a serial order. We first decode the equivalent binary code $\bar{\mathcal{C}}$ with the binary SP decoder. The outputs (in the form of bit-level LLR values) are converted into the corresponding $q$-ary forms, which are utilized to construct the $q$-ary LLR vectors (LLRVs) [4]. Then these LLRVs are directly fed into the $q$-ary SP decoder as extrinsic information. Final decisions are made based on the output of the $q$-ary SP decoder. Converting the binary likelihoods into the $q$-ary forms is shown as follows.

$$P^a_{x_n} = \prod_{i=1}^{p} P^{\bar{a}^{(i)}}_{\bar{x}^{(i)}_n}, a \in \mathbb{F}_q, n = 1, \ldots, N, \tag{3}$$

where $P^a_{x_n}$ is the probability that the transmit symbol $x_n$ equals $a$ and $P^{\bar{a}^{(i)}}_{\bar{x}^{(i)}_n}$ is the probability that the transmit bit $\bar{x}^{(i)}_n$ equals $\bar{a}^{(i)}$ ($\bar{a}^{(i)} \in \{0, 1\}$). We have $P^0_{\bar{x}^{(i)}_n} = e^\lambda/(1+e^\lambda)$ and $P^1_{\bar{x}^{(i)}_n} = 1/(1+e^\lambda)$, where $\lambda$ is the value of the bit-level LLR. Also, we define $P^{\bar{\mathbf{a}}}_{\bar{\mathbf{x}}_n} \triangleq P^a_{x_n}$ with $\bar{\mathbf{x}}_n = \left(\bar{x}^{(0)}_n, \ldots, \bar{x}^{(p-1)}_n\right)^T$ and $\bar{\mathbf{a}} = \left(\bar{a}^{(0)}, \ldots, \bar{a}^{(p-1)}\right)^T$.

When replacing the $q$-ary decoder by the HQD in the cooperative decoders, as shown in Fig. 2, the conversion from binary soft output to $q$-ary soft output is not needed. Bit-level LLRs are exchanged between the component decoders.

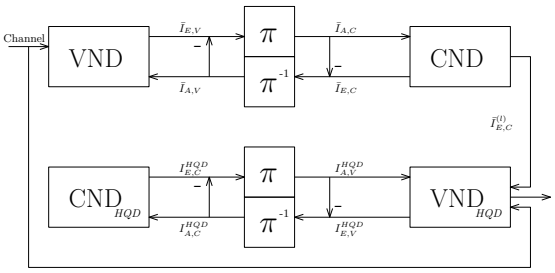


Fig. 2. Serially cooperative decoder with HQD. $\bar{I}_{E,V}$ and $\bar{I}_{A,V}$ are respectively the *extrinsic* information and *a priori* information out of the bit node detector (VND) in the binary decoder; $\bar{I}_{E,C}$ and $\bar{I}_{A,C}$ are respectively the *extrinsic* information and *a priori* information out the constituent check node detector (CND) in the binary decoder; $I^{HQD}_{E,V}$, $I^{HQD}_{A,V}$, $I^{HQD}_{E,C}$ and $I^{HQD}_{A,C}$ are respectively the corresponding information out of the bit node detector $\mathrm{VND_{HQD}}$ and constituent check node detector $\mathrm{CND_{HQD}}$ in the HQD. $\bar{I}^{(l)}_{E,C}$ is the *extrinsic* information in CND after $l$ iterations. $\pi$ and $\pi^{-1}$ are respectively the interleaver and inverse-interleaver.

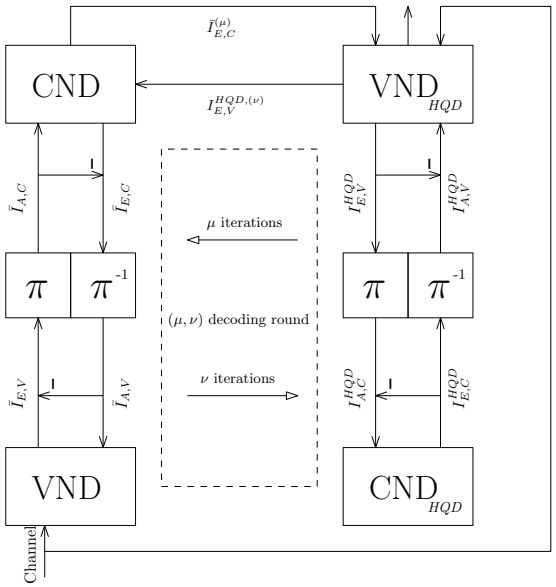


Fig. 3. Paralleled cooperative decoder with HQD. $\bar{I}_{E,C}^{(\mu)}$ is *extrinsic* information out of CND in the binary decoder after $\mu$ iterations. $I_{E,V}^{HQD,(\nu)}$ is the *extrinsic* information out of $\mathrm{VND_{HQD}}$ in the HQD after $\nu$ iterations.

### C. Paralleled Cooperative Decoder

Similar to the decoding principle of Turbo codes, the paralleled cooperative decoder (PCD) exchanges the messages between its component decoders. However, component decoders of PCD operate over different fields. We say one *decoding round* is finished when the binary decoder and $q$-ary decoder have exchanged their messages once. Let $\mu$ be the number of iterations done in the binary decoder and $\nu$ be the number of iterations done in the $q$-ary decoder. A $(\mu, \nu)$ decoding round is a round within which the binary decoder has done $\mu$ times decoding iterations and the $q$-ary decoder has done $\nu$ times decoding iterations before they exchange their massages. In order to do the parallel decoding, the $q$-ary likelihoods are needed to be transformed into their binary form by Eq. (4). The message from binary decoder to $q$-ary decoder is constructed in the same way as done in the SCD.

$$\begin{cases} P^{1}_{\bar{x}_n^{(i)}} = \sum_{\bar{\mathbf{a}} \in \mathbb{F}_q : \bar{a}^{(i)}=1} P^{\bar{\mathbf{a}}}_{\bar{\mathbf{x}}_n}, n = 1, \ldots, N, \\ P^{0}_{\bar{x}_n^{(i)}} = 1 - P^{1}_{\bar{x}_n^{(i)}}, \end{cases} \tag{4}$$

where $P^{\bar{\mathbf{a}}}_{\bar{\mathbf{x}}_n}$ can be obtained by solving the $q$-dimensional equations set regarding to the LLRV.

When the HQD is used in the PCD, as shown in Fig. 3, the conversion of the soft outputs are not needed. Bit-level LLRs are exchanged between the component decoders.

## IV. Construction of Parity Check Matrices

In the above sections, we based our design and optimizations on the *independence* assumption of the cooperative decoders without discussing the imperfect construction of the equivalent parity check matrix, *i.e.* the parity check matrix is constructed with cycles. In the following, we propose an algorithm to have the binary component decoder satisfying the *independence* assumption by avoiding the bit-level Tanner-Graph based cycles introduced by dense matrix labels. Then,

**Algorithm 1:** Candidate set of the sparse matrix labels.

**Data**: given the matrix labels $\{\mathbf{0}, \mathbf{A}^1, \ldots, \mathbf{A}^{q-1}\}$
**Result**: the candidate set $\mathcal{A}$ of the sparse matrix labels
Initialization: calculate the densities and girths of the matrix labels as $\frac{\omega(\mathbf{A}^i)}{p^2}$ and $\zeta(\mathbf{A}^i)$;
Set $\mathcal{T}$ to be the set that includes the combinations of different matrix labels that will be tested. Mostly, $\mathcal{T}$ is set to include all the combinations. Apparently, this algorithm can be accelerated if $\mathcal{T}$ only includes partial label combinations;
Set the density and girth constraints as $d$ and $g$. Then we obtain $\mathcal{A} = \{\mathbf{A}_i | d_i = \frac{\omega(\mathbf{A}_i)}{p^2} \leqslant d, g_i = \zeta(\mathbf{A}_i) \geqslant g$ and if $j \geqslant i$ then $d_j \geqslant d_i\}$;
Set the maximum girth $g_t$ as the end condition and $k = \text{sizeof}(\mathcal{A})$ ;
**for** $h := 2$ to $p$ **do**
  **if** $h \leqslant k$ **then**
    $t = 2h$;
    **while** $t \leqslant g_t$ **do**
      **for** $i := 1$ to $k - h + 1$ **do**
        Let $\mathcal{J} = \{\mathbf{j}_1, \mathbf{j}_2, \ldots\}$ be the $(h-1)$-tuple index set that contains all the $(h-1)$-combinations from the integer set $\{i+1, \ldots, k\}$;
        **for** $j = 1$ to sizeof($\mathcal{J}$) **do**
          **if** $\begin{pmatrix} \mathbf{A}_i \\ \mathbf{A}_{\mathbf{j}_j(1)} \\ \vdots \\ \mathbf{A}_{\mathbf{j}_j(h-1)} \end{pmatrix} \in \mathcal{T}$ **and** $\zeta\left(\begin{pmatrix} \mathbf{A}_i \\ \mathbf{A}_{\mathbf{j}_j(1)} \\ \vdots \\ \mathbf{A}_{\mathbf{j}_j(h-1)} \end{pmatrix}\right) == t$ **then**
            Remove $\{\mathbf{A}_{\mathbf{j}_j(1)}, \ldots, \mathbf{A}_{\mathbf{j}_j(h-1)}\}$ from $\mathcal{A}$;
          **if** $\begin{pmatrix} \mathbf{A}_i & \mathbf{A}_{\mathbf{j}_j(1)} & \cdots & \mathbf{A}_{\mathbf{j}_j(h-1)} \end{pmatrix} \in \mathcal{T}$ **and** $\zeta\left(\begin{pmatrix} \mathbf{A}_i & \mathbf{A}_{\mathbf{j}_j(1)} & \cdots & \mathbf{A}_{\mathbf{j}_j(h-1)} \end{pmatrix}\right) == t$ **then**
            Remove $\{\mathbf{A}_{\mathbf{j}_j(1)}, \ldots, \mathbf{A}_{\mathbf{j}_j(h-1)}\}$ from $\mathcal{A}$;
        $k = \text{sizeof}(\mathcal{A})$;
      $t = t + 2$;

based on the outputs of the proposed algorithm, we show how to obtain the desired parity check matrix for a given code degree distribution.

When we decode the $q$-ary LDPC code $\mathcal{C}$ by the binary SP decoders, there is a different type of cycle compared to those in [2], [5], [6], [13]. This type of cycle is introduced by the dense matrix-labels, *i.e.* the bit-level Tanner-graph-based cycle, even if the symbol-level cycle does not exist. In Fig. 1, we have shown the bit-level cycle for the non-binary parity check equation from *Example 1*. Moreover, avoiding the bit-level cycle is very different from the methods for symbol-level cycle elimination. As a matter of fact, this type of cycle is avoided if we only choose the $q$-ary coefficients associated with sparse matrix representations carefully. That is, with the chosen sparse matrix-labels, the existence of bit-level cycle only depends on the existence of symbol-level cycle. If the $q$-ary LDPC code $\mathcal{C}$ satisfies the *independence* assumption, the equivalent LDPC code $\bar{\mathcal{C}}$ will also satisfy the *independence* assumption.

To show how to choose the $q$-ary coefficients, we first define $\omega(\cdot)$ to be the function that counts the weight of a vector or matrix, and define $\zeta(\cdot)$ to be the function that counts the girth of a parity-check matrix which is the length of its shortest cycle and $\mathcal{A}$ to be the candidate set of the matrix labels with certain density and girth constraints, and $k$ to be the size of $\mathcal{A}$. If $\mathcal{C}$ satisfies the *independence* assumption, and no cycle exists in $\mathbf{H}$, *i.e.* no bit-level cycle in $\bar{\mathbf{H}}$ can be traced back to a symbol-level cycle in $\mathbf{H}$. As a result, only the combinations of different matrix labels placed in a row or in a column should be tested to make sure that no bit-level cycle is formed in $\bar{\mathbf{H}}$ with those chosen labels. For those combinations of matrix labels that will cause cycles (from length-4 to length-$g_t$, $g_t$ is mostly set to $2p$), we eliminate them from $\mathcal{A}$. At last, the program outputs the updated candidate set $\mathcal{A}$. The details have been shown in **Algorithm 1**, where $h$ in Step 5 of the algorithm is the number of matrix labels in a combination. Since the size of a matrix label is $p \times p$ and the length of the Tanner-graph-based cycle is an even number, the maximum length of bit-level cycle caused by the combination is $2p$. Any combination with size larger than $p$, can cause cycles with length less than $2p$. So the maximum number of matrix labels in a combination is set to $p$. For a combination of $h$ matrix labels, we eliminate the matrix labels from $\mathcal{A}$ that will cause bit-level cycles with minimum length of $2h$. Note that, for large $q$, the initial number of combinations that are about to be tested is huge, *i.e.* $\sum_{h=2}^{p} \binom{k}{h}$. Let $\mathcal{T}$ be the set that includes the combinations that will be tested. To accelerate **Algorithm 1** for large $q$, we can set $\mathcal{T}$ only includes partial matrix labels combinations.

After having $\mathcal{A}$, $\bar{\mathbf{H}}$ is constructed by choosing matrix labels from $\mathcal{A}$ with different portions as the threshold-optimized degree distributions suggests which is additionally conditioned on the size of $\mathcal{A}$. Note that we should only choose the labels with the tested combinations, especially when not all the combinations are tested.

*Example 2:* Considering the finite field $\mathbb{F}_8 = \{\mathbf{0}, \mathbf{A}^1, \ldots, \mathbf{A}^7\}$, the candidate set $\mathcal{A}_{d=\frac{5}{9},g=6} = \{\mathbf{0}, \mathbf{A}^7, \mathbf{A}, \mathbf{A}^6, \mathbf{A}^2\}$. If we set the maximum girth to be 6 (because the largest girth of a combination of the $\mathbb{F}_8$-matrix-labels is 6) and $\mathcal{T}$ includes all the combinations, then the updated $\mathcal{A} = \{\mathbf{0}, \mathbf{A}^7, \mathbf{A}, \mathbf{A}^6\}$. Note that $\mathcal{A}\backslash\mathbf{0}$ is a multiplicative group which will be utilized to optimize the performance of the cooperative decoders with HQD to avoid the bit-level cycle introduced by the matrix inverse operations.

**Algorithm 1** further optimizes the coefficients sets used in [5], [6], [10], [11], [25]. The codes constructed over $\mathcal{A}$ will satisfy the independence assumption more easily. The decoding performance is closer to the optimal one (cycle-free decoding). The size of the candidate set $\mathcal{A}$ can be enlarged by increasing the order of the finite field or loosing the the constraints on $\mathcal{A}$ which will allow more choices of $\bar{\mathcal{C}}$. Then, the independence assumption at bit-level will be satisfied more easily and better code may be found for the cooperative decoders proposed in section III by increasing the largest node degree in its code profile.

## V. Numerical Results and Simulations

Consider the rate $R = 1/2$ code over $\mathbb{F}_{512}$. Based the modified EXIT chart, we find the code $\mathcal{C}$ with length 30000-bits of little lower rate $R = 0.4982$ characterized by $\lambda(x) = 0.2498x + 0.1001x^2 + 0.1001x^3 + 0.0998x^4 + 0.0503x^5 + 0.0999x^6 + 0.05x^8 + 0.05x^{12} + 0.05x^{16} + 0.06x^{20} + 0.09x^{50}$ and $\rho(x) = 0.027x^6 + 0.721x^7 + 0.262x^8$. The performance threshold of $\mathcal{C}$ under PCD is $E_b/N_0 = 0.3355dB$ while the capacity limit is $E_b/N_0 = 0.187dB$. Let $\mu = \nu = 1$, we compare the performance of the proposed decoders in Fig. 4 where QSPA stands for Q-ary sum-product algorithm. PCD achieves the lowest error rate in the simulation while QSPA performs a little better than the binary decoder for long block length. Moreover, the SCD achieves little higher error rate than PCD in the simulation as some erroneous LLRs being amplified in the binary component decoder. The decoding performance of the PCD approaches the capacity limit within 0.2dB at BER$= 10^{-4}$. $q$-ary decoders can outperform binary decoders in the short block length regime, but for long block lengths binary decoders still are excellent competitors.

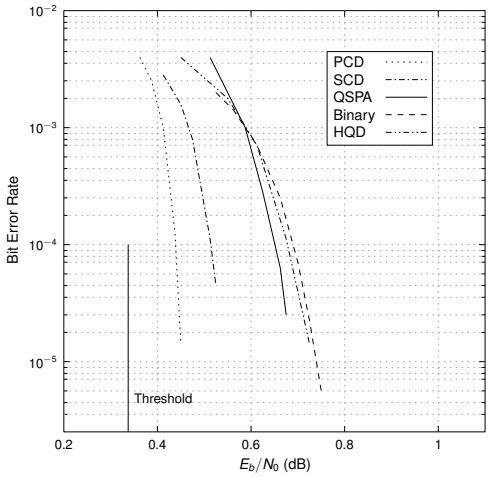


Fig. 4. Performance comparison between different decoders.

For the $q$-ary LDPC code $\mathcal{C}$ of rate $R$, the rate of its equivalent binary code $\bar{R} \geqslant R$, which can be readily obtained

according to the fact that some more rows of the equivalent parity check matrix $\bar{\mathbf{H}}$ with randomly generated matrix labels may not be independent from each other. In order to show the advantage of representing $q$-ary LDPC code with its binary counterpart, we compare the proposed cooperative decoder with some codes decoded by $q$-ary decoders under similar assumptions in [5] and [6]. For the 1/2 rate 4-ary LDPC codes of length 2800-bits, based on the modified EXIT chart, we find the code $\mathcal{C}$, characterized by $\lambda(x) = 0.29x+0.16x^2+0.12x^5+0.08x^8+0.06x^{16}+0.09x^{28}+0.2x^{51}$ and $\rho(x) = 0.7x^7+0.3x^8$, of a little lower rate $R = 0.4917$. The performance threshold of its equivalent binary code $\bar{\mathcal{C}}$ of rate $\bar{R} = 0.4932$ is $E_b/N_0 = 0.4968$dB. In Fig 5, we have shown the comparison results. After coefficients selection, the number of bit-level cycles will be significantly reduced. The codes from [5] are of length 5376-bits. Code from [6] is of length 2048-bits. Our codes decoded by the PCD shows great advantage in our simulation even for the code with much shorter block length than the codes in [5] and with smaller field size than the code in [6]. In addition, in Fig. 5 the code with uniform distributed labels performs the worst because of the inevitable bit-level cycles.

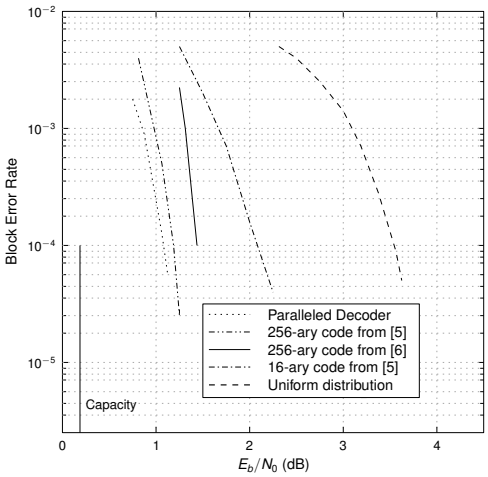


Fig. 5. Comparison with the codes in the literature.

## VI. Conclusion

In this paper, we design novel decoders for non-binary LDPC codes with code profile optimizations and parity-check matrices construction. For a non-binary LDPC $\mathcal{C}$ over $\mathbb{F}_q$, we first show how to construct the equivalent binary code $\bar{\mathcal{C}}$ by using the matrix and vector representations at bit-level. Then novel decoders are proposed to obtain excellent decoding performance with lower decoding complexity. Then we propose an algorithm to construct parity check matrix to satisfy the *independence* assumption. Experimental studies show that the proposed algorithms achieve excellent performance under different channel conditions with lower computational complexity order.

## References

[1] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533 – 547, sep 1981.

[2] M. C. Davey and D. J. MacKay, *Error-correction using LDPC Codes*. Cambridge Press, 1998.

[3] D. J. C. Mackay, "Optimizing sparse graph codes over gf (q)," available at http://www.inference.phy.cam.ac.uk/mackay/CodesGallager.html, 2003.

[4] G. Li, I. Fair, and W. Krzymien, "Density evolution for nonbinary ldpc codes under gaussian approximation," *IEEE Transactions on Information Theory*, vol. 55, no. 3, pp. 997 –1015, march 2009.

[5] J. Huang, S. Zhou, J. Zhu, and P. Willett, "Group-theoretic analysis of cayley-graph-based cycle gf(2p) codes," *IEEE Transactions on Communications*, vol. 57, no. 6, pp. 1560 –1565, june 2009.

[6] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2,dc)-ldpc codes over gf(q) using their binary images," *IEEE Transactions on Communications*, vol. 56, no. 10, pp. 1626 –1635, october 2008.

[7] Y. Yu and W. Chen, "Design of low complexity non-binary ldpc codes with an approximated performance-complexity tradeoff," *IEEE Communications Letters*, vol. 16, no. 4, pp. 514 –517, april 2012.

[8] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary ldpc codes over gf(q)," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633 – 643, april 2007.

[9] C. Vanstraceele, B. Geller, J. P. Barbot, and J. M. Brossier, "Block turbo codes for multicarrier local loop transmission," in *2002 IEEE 56th Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall.*, vol. 3, 2002, pp. 1773–1776 vol.3.

[10] V. Savin, "Binary linear-time erasure decoding for non-binary ldpc codes," in *Information Theory Workshop, 2009. ITW 2009. IEEE*, oct. 2009, pp. 258 –262.

[11] L. P. Sy, V. Savin, and D. Declercq, "Extended non-binary low-density parity-check codes over erasure channels." in *ISWCS,IEEE*, 2011, pp. 121–125.

[12] B. Smith, M. Ardakani, W. Yu, and F. Kschischang, "Design of irregular ldpc codes with optimized performance-complexity tradeoff," *IEEE Transactions on Communications*, vol. 58, no. 2, pp. 489 –499, february 2010.

[13] L. Sassatelli and D. Declercq, "Nonbinary hybrid ldpc codes," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5314 –5334, oct. 2010.

[14] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*. New York, NY, USA: Cambridge University Press, 1986.

[15] X. Ma and B. Bai, "A unified decoding algorithm for linear codes based on partitioned parity-check matrices," in *Information Theory Workshop, 2007. ITW '07. IEEE*, sept. 2007, pp. 19 –23.

[16] D. Forney, *Concatenated Codes*. Massachusetts Institute of Technology, Cambridge, MA, 1966.

[17] A. Barg and G. Zemor, "Concatenated codes: serial and parallel," *IEEE Transactions on Information Theory*, vol. 51, no. 5, pp. 1625 – 1634, may 2005.

[18] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth tanner graphs," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 2, 2001, pp. 995 –1001 vol.2.

[19] T. Tian, C. Jones, J. Villasenor, and R. Wesel, "Construction of irregular ldpc codes with low error floors," in *IEEE International Conference on Communications, 2003. ICC '03*, vol. 5, may 2003, pp. 3125 – 3129 vol.5.

[20] Y. Yu, W. Chen, and L. Wei, "Design of convergence-optimized non-binary ldpc codes over binary erasure channel," *IEEE Wireless Communications Letters*, vol. 1, no. 4, pp. 336 –339, august 2012.

[21] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb.

[22] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670 – 678, april 2004.

[23] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, 1996.

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

[25] V. Savin, "Non-binary ldpc codes over the binary erasure channel: Density evolution analysis," in *Applied Sciences on Biomedical and Communication Technologies, 2008. ISABEL '08. First International Symposium on*, oct. 2008, pp. 1 –5.