Enhancing Iterative Decoding of Cyclic LDPC Codes Using Their Automorphism Groups

Authors: Chao Chen, Baoming Bai, Xinquan Yang, Li Li, Yang Yang
Publication: IEEE T. Comm, June 2013
Speaker: Jeong-Min Ryu

Short summary: For cyclic LDPC codes, they propose to use their automorphism groups to improve the iterative decoding performance. Three types of iterative decoders are devised to take advantage of the code's automor-phism group. Towards exploiting the automorphism group of a code, they propose a new class of cyclic LDPC codes based on pseudo-cyclic MDS codes with two information symbols, for which nonequivalent parity-check matrices are obtained. Simulation results show that for their constructed codes of short lengths, the automorphism group can significantly enhance the iterative decoding performance.

I. INTRODUCTION

- **The use of automorphism group for classical codes**

Most classical codes are defined by high-density parity-check (HDPC) matrices, whose Tanner graphs have a large number of short cycles.

→ Iterative decoding performs rather poorly for these codes.

→To mitigate the deleterious effect of short cycles, Jiang and Narayanan [3] and Kothiyal et al. [4] proposed adaptive versions of iterative decoding, respectively.

→As a result, the performance was greatly improved. However, a significant increase in decoding complexity was incurred.

Classical codes are known to have a very rich algebraic structure.

→ To overcome the adverse effect of short cycles while maintaining a reasonable complexity, the automorphism group, as a code structure, was exploited for iterative decoding

→ For HDPC and moderate-density parity-check (MDPC) codes, the automorphism group aided iterative decoding techniques are applied.

- **In this paper,**

1) they apply automorphism group aided iterative decoding techniques to cyclic LDPC codes.

2) For a cyclic code, two particular subgroups of the automorphism group are well known. They show that for a large class of cyclic LDPC codes [15]-[18], [20], the two subgroups of the automorphism group belong to the same equivalence class and thus cannot be harnessed for iterative decoding.

3) They present a class of cyclic LDPC codes for which the automorphism group can be exploited for iterative decoding.

II. How to Use the Automorphism Group of a Code in Iterative Decoding

A. The Automorphism Group of a Code

Definition: Let C be a binary linear block code of length N. The set of coordinate permutations that map C to itself forms a group under the composition operation. This group is called the automorphism group of C, denoted by Aut(C)

For a permutation $\pi \in \mathrm{Aut}(C)$, let π^{-1} denote its inverse. From the definition we know that for any $c = (c_0, c_1, ..., c_{N-1}) \in C$, $\pi c = \left(c_{\pi^{-1}(0)}, c_{\pi^{-1}(1)}, ..., c_{\pi^{-1}(N-1)}\right) \in C$.

Let $C^{\perp}$ denote the dual code of C, then the following property holds.

Property 1: $\mathrm{Aut}(C) = \mathrm{Aut}(C^{\perp})$.

Property 2: For any $\pi \in \mathrm{Aut}(C)$ and a parity check matrix H of C, πH also forms a parity check matrix of C.

Property 3: For a binary cyclic code with odd length N, the automorphism group contains the following two subgroups:

S_0: The set of permutations $\tau^0, \tau^1, ..., \tau^{N-1}$, where $\tau^k : j \to (j+k) \mod N$.

S_1: The set of permutations $\zeta^0, \zeta^1, ..., \zeta^{N-1}$, where $\zeta^k : j \to (2^k \cdot j) \mod N$ and m_1 is the smallest positive integer such that $2^{m_1} \equiv 1 \mod N$.

B. Two Perspectives and Their Equivalence

Using the automorphism group of a code for decoding has a long history. In the early 1960s, MacWilliams devised a hard-decision decoding procedure, called the *permutation decoding* [14]. Recently, the code's automorphism group was brought into use in the soft-decision iterative decoding of HDPC codes [5]−[9] and MDPC codes [10]. Here, they review two possible perspectives involved and show their equivalence.

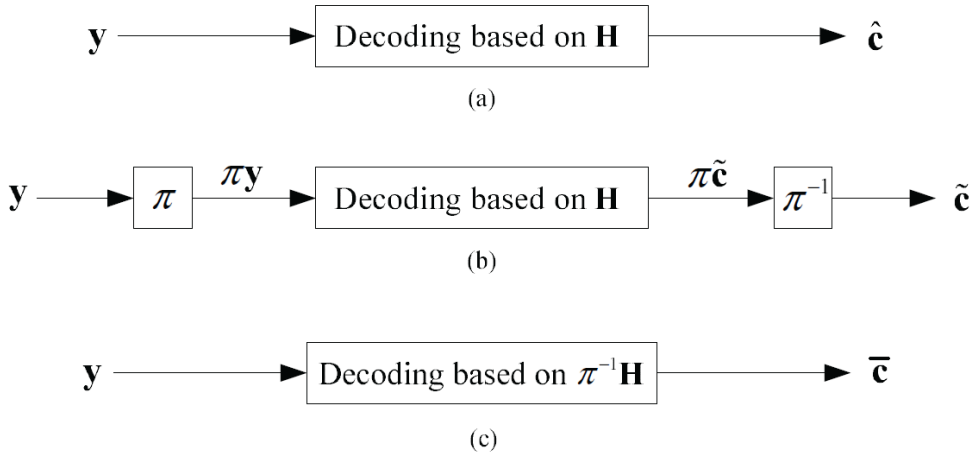

Fig. 1. Iterative decodings of the received sequence based on the SPA with flooding schedule.

(a) Assume that the BPSK signaling is used over the AWGN channel. Let $c \in C$ be the transmitted codeword and $x \in \{\pm 1\}^N$ the corresponding modulated sequence. Then the received signal sequence is given by

$$y = x + n,$$

where n contains N i.i.d. Gaussian noise samples with zero mean and variance σ^2.

(b) Applying a permuation $\pi \in \mathrm{Aut}(C)$, we have

$$\pi y = \pi x + \pi n.$$

Fact 1: In Fig. 1(a) and Fig. 1(b), the outputs $\hat{c}$ and $\tilde{c}$ are not necessarily equal. It is possible that only one of $\hat{c}$ and $\tilde{c}$ is the transmitted codeword.

Fact 2: In Fig. 1(a) and Fig. 1(c), the outputs $\hat{c}$ and $\bar{c}$ are not necessarily equal. It is possible that only one of $\hat{c}$ and $\bar{c}$ is the transmitted codeword.

Fact 3: In Fig. 1(b) and Fig. 1(c), the outputs $\tilde{c}$ and $\bar{c}$ are equal.

C. A partition of the Automorphism Group

We call two parity-check matrices *equivalent* if they can be obtained from each other through row permutations; otherwise, we call them *nonequivalent*. Since the flooding schedule is assumed, we further have

Fact 4: In Fig. 1(a) and Fig. 1(c), if H and πH are equivalent, then the outputs $\hat{c}$ and $\bar{c}$ are equal.

Let $\pi_1, \pi_2 \in Aut(C)$, then π_1 and π_2 belong to the same equivalence class if and only if $\pi_1 H$ and $\pi_2 H$ are equivalent. Note that the partition depends on the selection of H. For a given H, we can construct the same number of nonequivalent parity-check matrices as that of equivalence classes.

D. Design of Three Types of Iterative Decoders

Definition: Let the automorphism group of a code be partitioned based on a given H, a d-order diversity set is a set of d permutations that belong to different equivalence classes.

We choose a d-order diversity set $\{\pi_l : \pi_l \in Aut(C), l = 0,1,...,d-1\}$. Denote by L_y the log-likelihood ratio vector (LLRV) computed from y. Below, they present three types of iterative decoders that use the diversity set in different manners.

- **Decoder-1**: the diversity set is used in a *serial* manner. The decoding procedure is shown in Algorithm 1

Algorithm 1 The Decoding Procedure of Decoder-1

1: **for** $0 \le l \le d-1$ **do**
2: Perform at most I iterations of $\pi_l \mathbf{L_y}$ based on $\mathbf{H}$: if a codeword $\mathbf{c}$ is obtained, stop decoding and output $\pi_l^{-1}\mathbf{c}$.
3: **if** $l = d-1$ and no codeword has been obtained **then**
4: Output the hard-decision of $\mathbf{L_y}$.
5: **end if**
6: **end for**

- **Decoder-2**: the diversity set is used in a *periodic* manner. The decoding procedure is shown in Algorithm 2. In line 5, an inner iteration refers to one time updating of all check nodes and variable nodes of H. For $d = 2$, the decoder works in a Turbo manner [1]. But there are two main differences: 1) The message passing out of a component decoder in the preceding iteration is not subtracted from the a priori information passing to this component decoder in the current iteration; 2) Soft information exchanged between the two component decoders is not limited to information bits.

Algorithm 2 The Decoding Procedure of Decoder-2

1: $\mathbf{s} \leftarrow \mathbf{L_y}$.
2: **for** $0 \le i \le I-1$ **do**
3: **for** $0 \le l \le d-1$ **do**
4: $\mathbf{s} \leftarrow \pi_l \mathbf{s}$.
5: Perform one inner iteration of $\mathbf{s}$ based on $\mathbf{H}$ and update $\mathbf{s}$ as the *a posteriori* LLRV.
6: $\mathbf{s} \leftarrow \pi_l^{-1} \mathbf{s}$.
7: **end for**
8: Use $\mathbf{s}$ to make a hard-decision: if a codeword $\mathbf{c}$ is obtained, stop decoding and output $\mathbf{c}$.
9: **if** $i = I-1$ and no codeword has been obtained **then**
10: Output the hard-decision of $\mathbf{L_y}$.
11: **end if**
12: **end for**

Decoder-3: the diversity set is used in a *parallel* manner. Define $H_l \triangleq \pi_l^{-1} H$. Then by concatenating H_l, we form an augmented parity-check matrix

$$H_{aug} = \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_{d-1} \end{bmatrix}.$$

The decoder performs the SPA with flooding schedule on this highly redundant parity-check matrix, with the maximum number of iterations I.

III. A NEW CONSTRUCTION

We define an $(l\cdot c)\times(l\cdot c)$ binary matrix as

$$B=\begin{bmatrix} A_0 & A_1 & \cdots & A_{c-2} & A_{c-1} \\ A_{c-1}^{(1)} & A_0 & \cdots & A_{c-3} & A_{c-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_2^{(1)} & A_3^{(1)} & \cdots & A_0 & A_1 \\ A_1^{(1)} & A_2^{(1)} & \cdots & A_{c-1}^{(1)} & A_0 \end{bmatrix},$$

where each submatrix is an $l\times l$ circulant and the zeroth row of $A_i^{(1)}$ is the first row of A_i. Define a permutation π as

$$\pi: j\to(j \bmod l)\cdot c+\lfloor j/l\rfloor,\ j=0,1,...,l\cdot c-1. \tag{1}$$

Theorem 4: If we perform row and column permutations on *B*, both using the permutation π given in (1), then we obtain a circulant matrix.

They summarize the construction procedure as follows.

Step 1: Choose a nonzero codeword from an (*n*, 2) pseudo-cyclic MDS code with $a=\alpha$.

Step 2: Use the codeword and its pseudo-cyclic shifts to construct the base matrix W'.

Step 3: Use matrix dispersion on W' to obtain the QC matrix $H_{disp}(W')$.

Step 4: Apply Theorem 4 to $H_{disp}(W')$ to obtain a circulant as the parity-check matrix *H*.

For **Step 1** and **Step 2**, the form of base matrix W' is given by

$$W'=\begin{bmatrix} w_0 & w_1 & \cdots & w_{n-2} & w_{n-1} \\ \alpha w_{n-1} & w_0 & \cdots & w_{n-3} & w_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha w_2 & \alpha w_3 & \cdots & w_0 & w_1 \\ \alpha w_1 & \alpha w_2 & \cdots & \alpha w_{n-1} & w_0 \end{bmatrix}. \tag{2}$$

To construct the base matrix of the form (2), they consider using pseudo-cyclic MDS codes with two information symbols. A pseudo-cyclic code with parameter $a \in GF(q)$ has the property that for any codeword $(c_0, c_1, ..., c_{n-1})$, its pseudo-cyclic shift $(ac_{n-1}, c_0, ..., c_{n-2})$ is also a codeword. If $a = 1$, the pseudo-cyclic code reduces to a cyclic code.

For **Step3**, the Tanner graph corresponding to the matrix $H_{disp}(W)$ has no cycles of length 4 and hence has a girth of at least 6. So the matrix $H_{disp}(W)$ can serve as the parity-check matrix and gives a QC-LDPC code of length $n(q-1)$.

$$H_{disp}(W) = \begin{bmatrix} A(w_{0,0}) & A(w_{0,1}) & \cdots & A(w_{0,n-1}) \\ A(w_{1,0}) & A(w_{1,1}) & \cdots & A(w_{1,n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ A(w_{m-1,0}) & A(w_{m-1,1}) & \cdots & A(w_{m-1,n-1}) \end{bmatrix}.$$

The way to construct the matrix A is given as follow:

Let GF(q) be a finite field with q elements and a be a primitive element of GF(q). Then $\alpha^{-\infty} \triangleq 0, \alpha, \cdots, \alpha^{q-2}$ give all the elements of GF(q). For each non-zero element $\alpha^i, (0 \le i \le q-2)$, define a $(q-1)\times(q-1)$ matrix $A(\alpha^i)$ over GF(2): it is a circulant permutation matrix; the zeroth row is a $(q-1)$-tuple with weight one where the ith component is equal to one and all the other $q-2$ components are equal to zero. The matrix $A(\alpha^i)$ is referred to as the $(q-1)$-fold matrix dispersion of element ai over GF(2). The $(q-1)$-fold matrix dispersion of zero element of GF(q), $A(0)$, is defined as the $(q-1)\times(q-1)$ all-zero matrix.

IV. Simulation Results

They present the simulation results for our constructed cyclic LDPC codes. The BPSK modulated AWGN channel is assumed. In addition to the three decoders presented in Section II, they also simulated a decoder that is not assisted by the automorphism group. The decoder is called Decoder-0, which performs the SPA with flooding schedule on the defining parity-check matrix. For all these decoders, the maximum number of iterations is set to be 100.

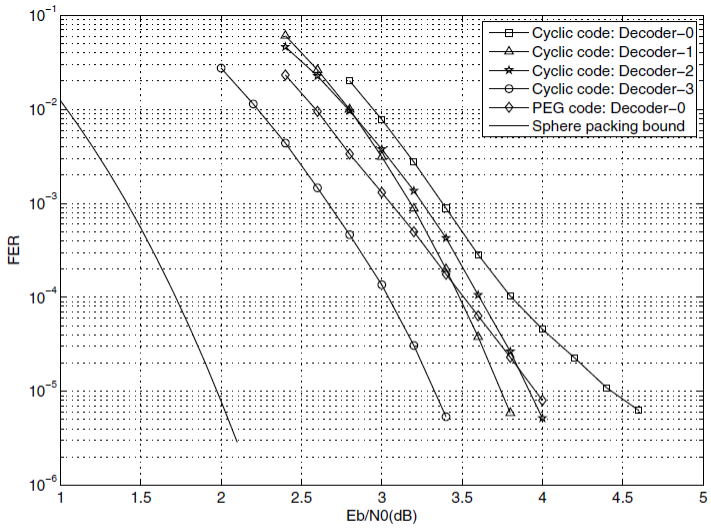

Fig. 2. Performance of the $(341, 160)$ cyclic LDPC code and the $(341, 160)$ PEG-based LDPC code.

Fig. 2 shows the FER performance of the code. The 2-order diversity set $\{\zeta^0, \zeta^1\}$ is used. For comparison, they further simulated a (341,160) LDPC code constructed using the progressive-edge-growth (PEG) algorithm [29]. The parity-check matrix of the code has column weight 3 and row weights 5 and 6. The sphere-packing bound [30] for this length and rate is also included in the figure.

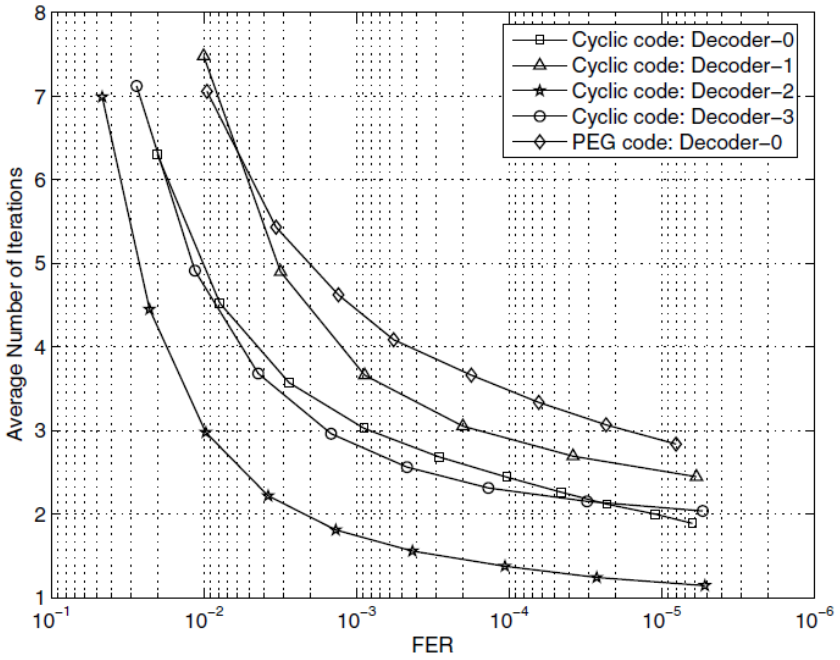

Fig. 3. Average number of iterations for all decoders.

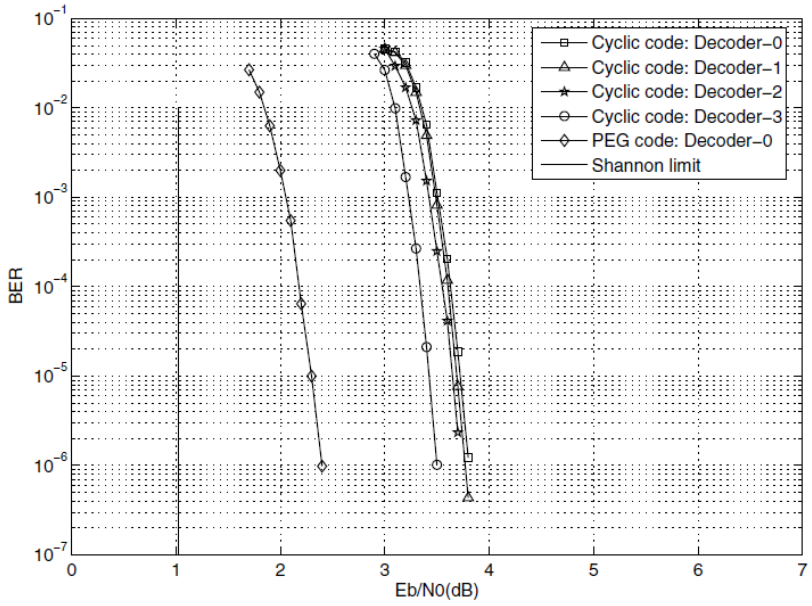

Fig. 4. Performance of the $(5461, 3612)$ cyclic LDPC code and the $(5461, 3612)$ PEG-based code.

- Like HDPC codes, the performance gain for LDPC codes seems more significant for short code lengths. This can be seen by comparing Fig. 2 and Fig. 4 (note that the diversity sets for the two codes have the same order).
- For both HDPC and LDPC with long code lengths, the automorphism group aided iterative decoding does not perform well. In fact, for long HPDC codes, it performs even worse than the hard-decision decoding.
- To obtain a noticeable performance gain, LDPC codes require a smaller diversity set than HDPC codes. This is because that LDPC codes are inherently more suited to iterative decoding than HDPC codes.