Journal of Southeast University (English Edition) Vol. 31, No. 1, pp. 25 - 30 Mar. 2015 ISSN 1003—7985

Efficient construction and encoding of QC-LDPC codes by cyclic lifting of protographs

Liang Yuan Zhang Shulin Gu Pinbiao Wu Lenan

(School of Information Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: Quasi-cyclic low-density parity-check (QC-LDPC) codes can be constructed conveniently by cyclic lifting of protographs. For the purpose of eliminating short cycles in the Tanner graph to guarantee performance, first an algorithm to enumerate the harmful short cycles in the protograph is designed, and then a greedy algorithm is proposed to assign proper permutation shifts to the circulant permutation submatrices in the parity check matrix after lifting. Compared with the existing deterministic edge swapping (DES) algorithms, the proposed greedy algorithm adds more constraints in the assignment of permutation shifts to improve performance. Simulation results verify that it outperforms DES in reducing short cycles. In addition, it is proved that the parity check matrices of the cyclic lifted QC-LDPC codes can be transformed into block lower triangular ones when the lifting factor is a power of 2. Utilizing this property, the QC-LDPC codes can be encoded by preprocessing the base matrices, which reduces the encoding complexity to a large extent.

Key words: low-density parity-check (LDPC) codes; quasi-cyclic LDPC codes; cyclic lifting; protograph LDPC codes

doi: 10.3969/j.issn.1003-7985.2015.01.005

A binary LDPC code is defined by its sparse parity check matrix $\boldsymbol{H} = (h_{i,j})$ and the corresponding Tanner graph. The Tanner graph consists of variable nodes v_j corresponding to the columns of the parity check matrix, check nodes c_i representing the rows, and edges $\{i, j\}$ connecting the two types of nodes. If the entries of 1 in a LDPC parity check matrix $\boldsymbol{H}$ of size $m \times n$ are replaced with non-zero submatrices of size $e \times e$ and the entries of 0 are replaced with zero submatrices of size $e \times e$, the code (base code) develops into an e-lifted code whose parity check matrix is $\widetilde{\boldsymbol{H}} = [\boldsymbol{H}_{i,j}]$ of size $me \times ne$. If each nonzero submatrix $\boldsymbol{H}_{i,j}$ is a circulant permutation matrix with permutation shift $s_{i,j}$, which means that $\boldsymbol{H}_{i,j}$ cyclically shifts the columns of an $e \times e$ identity matrix by $s_{i,j}$ to the bottom, then the lifted code is quasi-cyclic. Such a way to construct QC-LDPC codes is named as cyclic lifting[1-2] of protographs[3-4], where protographs refer to the Tanner graphs of base codes.

The key problem in cyclic lifting is the elimination of short cycles. This paper proposes algorithms to enumerate and eliminate harmful short cycles to construct QC-LDPC codes with good performance. In addition, this paper proposes a novel encoding method for cyclic lifted QC-LDPC codes, which can greatly reduce the complexity of preprocessing parity check matrices in encoding.

1 Enumeration of Cycles

A path in Tanner graph starts from a node and reaches a node through distinct edges. The length of a path is the number of edges in the path. A cycle is a path whose start and end are the same node while other nodes are distinct. Also, extrinsic message degree (EMD)[5] and approximate cycle extrinsic message degree (ACE)[5] are defined to measure the influence of a certain cycle on the performance of the code. Since stopping sets[6] and trapping sets[7] contain cycles with small EMD[5,8], short cycles with small EMD or ACE should be eschewed to improve the performance of the code. The following theorem reveals the relationship between the cycles in the protograph and those in the Tanner graph after cyclic lifting.

Theorem 1 Code $\widetilde{C}$ is a cyclic e-lifted code of base code C, and γ is a cycle of length λ in the Tanner graph of C. The inverse image of γ in the Tanner graph of $\widetilde{C}$ is a union of e/k cycles with length $k\lambda$.

$$k = \frac{e}{\mathrm{GCD}(e,\ s)}$$

is the order of permutation shift of γ and

$$s = \sum_{i}^{\lambda} (-1)^{i} s_i \bmod e$$

is the permutation shift of γ after lifting, where s_i is the permutation shift of the submatrix in the lifted parity check matrix $\widetilde{\boldsymbol{H}}$ that corresponds to the i-th edge in γ.

The proof can be seen in Refs. [2, 9]. It shows that cycles in a protograph can be eliminated if their permutation shifts are non-zero after lifting. Therefore, the first task of lifting is the enumeration of short cycles. We design a cycle enumeration algorithm based on tree expansion[8].

Received 2014-08-24.

Biographies: Liang Yuan(1990—), male, graduate; Zhang Shulin(corresponding author), male, master, associate professor, slzhang@seu.edu.cn.

Foundation item: The National Key Technology R&D Program of China during the 12th Five-Year Plan Period (No. 2012BAH15B00).

Citation: Liang Yuan, Zhang Shulin, Gu Pinbiao, et al. Efficient construction and encoding of QC-LDPC codes by cyclic lifting of protographs[J]. Journal of Southeast University (English Edition), 2015, 31(1): 25 - 30. [doi: 10.3969/j.issn.1003-7985.2015.01.005]

The algorithm traverses all the edges in the Tanner graph and finds the cycles consisting of each edge. Let $\Gamma(v_j)$ denote the check nodes that are incidental to v_j. Therefore, if we want to search for cycles of length λ consisting of edge $\{i, j\}$, the problem equals finding paths of length $\lambda - 2$ from c_i to other check nodes in $\Gamma(v_j)$ and in each of these paths all the nodes are distinct and v_j is not contained. The basic idea of the algorithm is to expand a tree from a root node, c_i for example. Symbol $P(n_k)$ represents the set of valid paths from root to node n_k. At each level of the tree, we only need to expand the tree from the active nodes and whether a node n_k is active depends on whether its $P(n_k)$ is updated at the level. Symbol $d_m(n_k, n_t)$ denotes the minimum distance between node n_k and n_t in the original Tanner graph. The pseudocode is given in Algorithm 1.

Algorithm 1 Enumeration of cycles shorter than λ

```
1) Res←∅//This set stores all the cycles found
2) Calculate d_m(n_k, n_t) for each pair of nodes
for each variable node v_j in the Tanner graph do
  1) Γ(v_j)←{all the check nodes connected to v_j}
  2) Remove v_j from the Tanner graph
  for each check node c_i in Γ(v_j) do
   for each node n_k in the graph do
    P(n_k)←∅
   end
   P(c_i) = {(c_i)}, AN←{c_i}, nAN←∅
   //AN is the set of the active nodes of the last level
   //And nAN is that of the current level
   while |AN| ≠ 0
    for each node n_k in AN do
   Γ(n_k) ← {all the nodes connected to n_k}
    for each node n_t in Γ(n_k) do
     1) P(n_k)\n_t←{the paths in P(n_k) which do not pass n_t}
     2) Append n_t to each path in P(n_k)\n_t
     3) (P(n_k)\n_t)\P(n_t)←{paths in P(n_k)\n_t and shorter than λ
- d_m(n_t, v_j) - 1 but not in P(n_t)}
      if |(P(n_k)\n_t)\P(n_t)| ≠ 0
       1) P(n_t)←(P(n_k)\n_t)\P(n_t) ∪ P(n_t)
       2) nAN←nAN∪{n_t}
      end
    end
    AN←nAN, nAN←∅
  end
  for each check node c_s in Γ(v_j) that s > i do
    1) Add v_j to each path in P(c_s) as start and end
    2) Res←Res∪P(c_s)
   end
  end
end
```

2 Elimination of Cycles

2.1 Assignment of permutation shifts

We prescribe that the lifting factor e should be a power of 2. In this case, the lifted codes can be encoded efficiently, which will be shown in the next section. According to Theorem 1, we can eliminate short cycles by assigning proper permutation shifts. The deterministic edge swapping (DES)[2] algorithm is proposed to search for the good assignment of permutation shifts. However, in the following greedy algorithm, we add more constraints in the assignment of permutation shifts to achieve better performance.

Suppose that there are u non-zero permutation submatrices in $\tilde{\boldsymbol{H}}$, corresponding to the u 1s in $\boldsymbol{H}$. The permutation shifts of these submatrices are denoted by $s_1, s_2, \ldots, s_u$. Therefore, the permutation shift of a length λ cycle can be calculated as

$$s_p = \sum_{i=1}^{\lambda} (-1)^i s_{w_i} \bmod e = \boldsymbol{\delta}^{\mathrm{T}} \boldsymbol{\alpha} \tag{1}$$

where w_i is the index of the submatrix which corresponds to the i-th edge of the cycle; $\boldsymbol{\delta} = [g_1, g_2, \ldots, g_u]^{\mathrm{T}}$ is called the coefficient vector of the path where $g_{w_i} = (-1)^i$ and others are 0s; $\boldsymbol{\alpha} = [s_1, s_2, \ldots, s_u]^{\mathrm{T}}$ is called the permutation vector.

The assignment of permutation shifts is converted to the problem of a matrix equation,

$$\boldsymbol{M\alpha} = \boldsymbol{\beta} \tag{2}$$

where each row in $\boldsymbol{M}$ is the transposition of coefficient vector of a short cycle that we want to eliminate. Notice that the matrix operation here is over the ring of $Z_e = \{0, 1, 2, \ldots, e-1\}$, where the addition and multiplication are both modulo e operations, and -1 in $\boldsymbol{M}$ can be replaced by $e-1$ equivalently. The problem is to find an $\boldsymbol{\alpha}$ to reduce as many 0s in $\boldsymbol{\beta}$ as possible.

By elementary column operations, matrix $\boldsymbol{M}$ can be transformed to a lower triangular matrix $\boldsymbol{L}$,

$$\boldsymbol{MT} = \boldsymbol{L} \tag{3}$$

However, not all the elementary operations can be executed. There are zero divisors in Z_e, namely 2^r where $r = 1, 2, \ldots, q-1$, and the multiplication of a column by a zero divisor is not invertible. Suppose that the dimension of $\boldsymbol{M}$ is $N_{\mathrm{p}} \times u$, and let $i \leftarrow 1$, $j \leftarrow 1$, $\boldsymbol{L} \leftarrow \boldsymbol{M}$. The steps of the Gaussian elimination are as follows:

1) Find the first entry in row i whose column index is greater than or equal to j and which is not a zero divisor. If all the entries' column indices are less than j, let $i \leftarrow i+1$, and repeat step 1). If the entries are all zero divisors, go to step 3).

2) If the element found in step 1) is in column j' and $j' \neq j$, switch column j' with column j. Eliminate all the non-zero elements in row i by adding the multiples of column j to the corresponding columns. This is feasible since the element at (i, j) is a non-zero divisor. Then let $i \leftarrow i+1$, $j \leftarrow j+1$. If $i > N_{\mathrm{p}}$ or $j > u$, end. Otherwise, go to step 1).

3) Since all the non-zero entries in row i whose column indices are greater than or equal to j are zero divisors of $e = 2^q$, one of them will be a divisor of their GCD over

ring Z_e. If this element is in column j' and $j' \neq j$, switch column j′ with column j. Eliminate all the non-zero entries in row i whose column indices are greater than j by adding the multiples of column j to the corresponding columns. Then let $i \leftarrow i+1$, $j \leftarrow j+1$. If $i > N_p$ or $j > u$, end. Otherwise, go to step 1).

An approximate lower triangular matrix $\boldsymbol{L} = [\tau_{i,j}]$ can be obtained. Let $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_u]^{\mathrm{T}}$ and $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_{N_p}]^{\mathrm{T}}$. If j is the greatest index in row i of $\boldsymbol{L}$ satisfying that $\tau_{i,j}$ is non-zero, we call row i a constraint row of ε_j, and $\sum_{j'=1}^{j} \tau_{i,j'} \varepsilon_{j'} = \beta_i$ is the constraint equation of ε_j. For a constraint equation of ε_j, where $\tau_{i,j} = 2^r v$ and v is an odd integer, it may exclude 2^r possible permutation shifts that ε_j can choose at most. The number of these constraint equations is expressed by Ne_r and if $\sum_{r=0}^{q-1} 2^r Ne_r > e$, there will not necessarily be a permutation shift of ε_j satisfying all its constraint rows. Also, ε_j in such case is over constrained.

Suppose that $\sum_{j'=1}^{j} \tau_{i,j'} \varepsilon_{j'} = \beta_i$ is a constraint equation of an over constrained ε_j. If l_i is the greatest index satisfying 1) $l_i < j$ and 2) τ_{i,l_i} is non-zero, ε_{l_i} is called the leading element of the constraint equation. Suppose that ε_{j_s}, $s = 1, 2, \ldots, N_o$, are all the over constrained elements, each of which has at least one constraint equation of the leading element ε_l. When assigning the permutation shift of ε_l, we consider its own constraint equations as well as the over constraint equations of ε_{j_s}. The set of permutation shifts which can satisfy the most constraint equations of ε_l is denoted by Λ_l, and the set of permutation shifts of ε_{j_s} excluded by constraint equations with leading elements' indices less than or equal to l is denoted by $\Psi(j_s, l)$. Since we assign the permutation shifts of ε sequentially, we first calculate $\Psi(j_s, l-1)$. Then we calculate the set $\Psi(j_s, l)$ corresponding to each value in Λ_l and choose the one which can minimize

$$\sum_{s=1}^{N_o} |\psi(j_s, l)| - |\psi(j_s, l-1)| \tag{4}$$

2.2 Simulation

The base code here is a regular code of length 50 and code rate 0.5 whose variable node degree is 3 and check node degree is 6. The base code is constructed randomly by the PEG algorithm[10]. We try lifting factors of 8, 16 and 32, and use both DES and the proposed method to assign the permutation shifts. We aim at eliminating cycles shorter than 10. For comparison, ordinary LDPC codes are also constructed by the PEG algorithm. The distribution of cycles shorter than 10 is listed in Tab. 1.

Fig. 1 displays the bit error rate (BER) performance of the above codes over AWGN channel. The decoding algorithm used is sum product[11] with 30 times iteration. The simulation results reveal that when the code length is relatively short, the three types of codes have very similar BER performances. However, when the lifting factor becomes greater, even though the three codes have similar performances in high BER regions, the lifted QC-LDPC code constructed by the proposed method outperforms the ones constructed by DES and ordinary ones in low BER regions. Their BER performances accord with their cycle distributions: the fewer the short cycles, the lower the BERs. This substantiates the effectiveness of the proposed method.

Tab. 1 Cycle distribution of regular codes

Cycle length	Base code	DES			Proposed method			Randomly constructed		
		$e=8$	$e=16$	$e=32$	$e=8$	$e=16$	$e=32$	$n=400$	$n=800$	$n=1\,600$
4	0	0	0	0	0	0	0	0	0	0
6	204	0	0	0	0	0	0	0	0	0
8	1 438	1 600	768	544	1 088	304	0	1 267	334	0

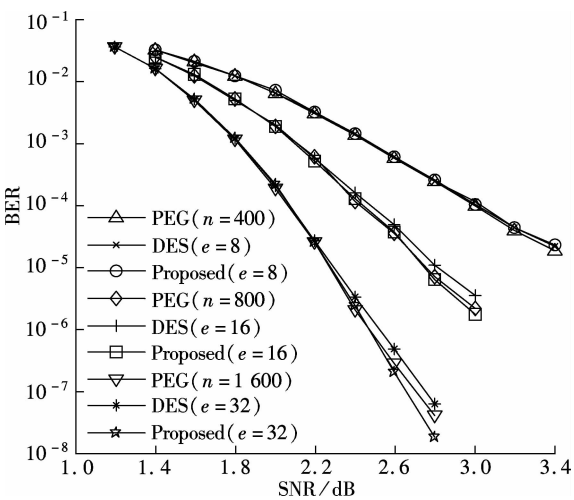

Fig. 1 BER performance of regular codes over AWGN channel

Then, we consider the case of irregular codes. The base code is an irregular code of length 100 and code rate 0.5, which is constructed by PEG. Its variable and check node degree distribution functions are

$$\lambda(x) = 0.204x + 0.509x^2 + 0.037x^5 + 0.086x^6 + 0.025x^7 + 0.139x^8$$
$$\rho(x) = 0.537x^5 + 0.389x^6 + 0.074x^7$$

Lifting factor $e = 32$. We aim at eliminating cycles including 1) All the cycles shorter than 8, 2) The cycles of length 8 whose ACEs $\leqslant 16$, and 3) The cycles of length 10 whose ACEs $\leqslant 8$. The cycle distribution is listed in Tab. 2. The proposed method has fewer cycles of interest

than DES. Fig. 2 displays the BER performance. For comparison, an irregular code of length 3 200 with the same degree distribution is also constructed randomly. We can see that the code constructed by the proposed method decreases the error floor from 10^{-7} to 10^{-8}.

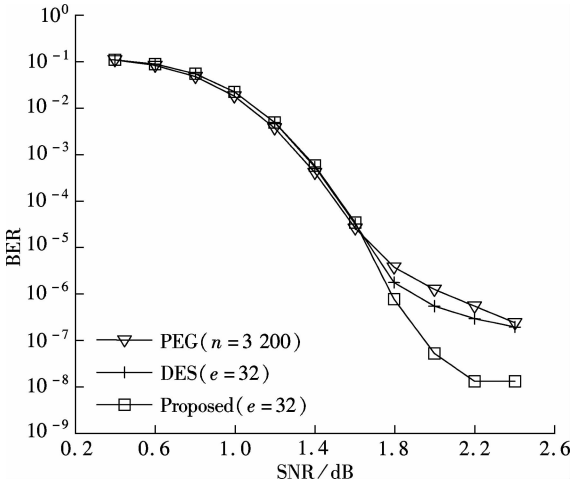

Fig. 2 BER performance of irregular codes over AWGN channel

Tab. 2 Cycle distribution of irregular codes

Cycle length	Base code	DES	Proposed method
4	0	0	0
6	881	0	0
8 (ACE≤16)	6 259	8 608	5 600
10 (ACE≤8)	2 235	0	0

3 Efficient Encoding of QC-LDPC Codes

$\tilde{\boldsymbol{H}}$ can be transformed into a block cyclic matrix $\tilde{\boldsymbol{H}}'$ by row and column permutations[2], which is

$$\tilde{\boldsymbol{H}}' = \begin{bmatrix} \boldsymbol{\mathcal{H}}^0 & \boldsymbol{\mathcal{H}}^{e-1} & \cdots & \boldsymbol{\mathcal{H}}^1 \\ \boldsymbol{\mathcal{H}}^1 & \boldsymbol{\mathcal{H}}^0 & \cdots & \boldsymbol{\mathcal{H}}^2 \\ \vdots & \vdots & & \vdots \\ \boldsymbol{\mathcal{H}}^{e-1} & \boldsymbol{\mathcal{H}}^{e-2} & \cdots & \boldsymbol{\mathcal{H}}^0 \end{bmatrix} \tag{5}$$

where $\boldsymbol{\mathcal{H}}^d = (\hbar_{i,j}^{\,d})$ is an $m \times n$ submatrix, and $\hbar_{i,j}^{\,d}$ is the entry at the position $(d+1, 1)$ of submatrix $\boldsymbol{H}_{i,j}$.

Theorem 2 For a lifted QC-LDPC code, where the lifting factor $e = 2^q$, $\tilde{\boldsymbol{H}}'$ can be transformed into a lower block-triangular matrix $\boldsymbol{\Lambda}$ by proper elementary row and column operations, that is

$$\boldsymbol{\Lambda} = \boldsymbol{R}\tilde{\boldsymbol{H}}'\boldsymbol{C} = \begin{bmatrix} \boldsymbol{H} & & & & \\ \sum_{k=0}^{e-2}\binom{e-2}{k}\boldsymbol{\mathcal{H}}^{e-1-k} & \boldsymbol{H} & & & \\ \sum_{k=0}^{e-3}\binom{e-3}{k}\boldsymbol{\mathcal{H}}^{e-1-k} & \sum_{k=0}^{e-2}\binom{e-2}{k}\boldsymbol{\mathcal{H}}^{e-1-k} & \boldsymbol{H} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \boldsymbol{\mathcal{H}}^{e-1} & \sum_{k=0}^{1}\binom{1}{k}\boldsymbol{\mathcal{H}}^{e-1-k} & \cdots & \sum_{k=0}^{e-2}\binom{e-2}{k}\boldsymbol{\mathcal{H}}^{e-1-k} & \boldsymbol{H} \end{bmatrix} \tag{6}$$

$$\boldsymbol{R} = \begin{bmatrix} \binom{e-1}{0}\boldsymbol{I}_m & \binom{e-1}{1}\boldsymbol{I}_m & \binom{e-1}{2}\boldsymbol{I}_m & \cdots & \binom{e-1}{e-1}\boldsymbol{I}_m \\ & \binom{e-2}{0}\boldsymbol{I}_m & \binom{e-2}{1}\boldsymbol{I}_m & \cdots & \binom{e-2}{e-2}\boldsymbol{I}_m \\ & & \binom{e-3}{0}\boldsymbol{I}_m & \cdots & \binom{e-3}{e-3}\boldsymbol{I}_m \\ & & & \ddots & \vdots \\ & & & & \boldsymbol{I}_m \end{bmatrix} \tag{7}$$

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_n & \binom{1}{0}\boldsymbol{I}_n & \binom{2}{0}\boldsymbol{I}_n & \cdots & \binom{e-1}{0}\boldsymbol{I}_n \\ & \binom{1}{1}\boldsymbol{I}_n & \binom{2}{1}\boldsymbol{I}_n & \cdots & \binom{e-1}{1}\boldsymbol{I}_n \\ & & \binom{2}{2}\boldsymbol{I}_n & \cdots & \binom{e-1}{2}\boldsymbol{I}_n \\ & & & \ddots & \vdots \\ & & & & \binom{e-1}{e-1}\boldsymbol{I}_n \end{bmatrix} \tag{8}$$

where $\boldsymbol{H}$ is the parity check matrix of the base code, and all the matrices are defined over GF(2). Here, $\binom{i}{j} = \frac{i!}{j!(i-j)!} \bmod 2$.

Proof $\boldsymbol{\mathcal{H}}^d, d = 0, 1, \ldots, e-1$, and their summations are expressed by polynomials over GF(2), where x^d denotes $\boldsymbol{\mathcal{H}}^d$. It is clear that $\sum_{d=0}^{e-1}\boldsymbol{\mathcal{H}}^d = \boldsymbol{H}^{[2]}$, so $\sum_{d=0}^{e-1}x^d = \sum_{d=0}^{2^q-1}x^d = (1+x)^{2^q-1} = \boldsymbol{H}$. By induction, it can be proved that after $e-1$ times transformation, $\boldsymbol{\Lambda} = \boldsymbol{R}_{e-1}\ldots\boldsymbol{R}_2\boldsymbol{R}_1\tilde{\boldsymbol{H}}'\boldsymbol{C}_1\boldsymbol{C}_2\ldots\boldsymbol{C}_{e-1}$. In the first transformation, row blocks 2, 3, …, e are added to the row block 1, then column blocks $e-1$, $e-2$, …, 1 are added to column blocks e, $e-1$, …, 2 in sequence, where

$$\tilde{\boldsymbol{H}}'_{(1)} = \begin{bmatrix} \boldsymbol{H} & & & \\ x^1 & x^1+x^0 & \cdots & x^3+x^2 \\ x^2 & x^2+x^1 & \cdots & x^4+x^3 \\ \vdots & \vdots & & \vdots \\ x^{e-1} & x^{e-1}+x^{e-2} & \cdots & x^1+x^0 \end{bmatrix}$$

$$\boldsymbol{R}_1 = \begin{bmatrix} \boldsymbol{I}_m & \boldsymbol{I}_m & \boldsymbol{I}_m & \cdots & \boldsymbol{I}_m \\ & \boldsymbol{I}_m & & & \\ & & \boldsymbol{I}_m & & \\ & & & \ddots & \\ & & & & \boldsymbol{I}_m \end{bmatrix}, \quad \boldsymbol{C}_1 = \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{I}_n & & & \\ & \boldsymbol{I}_n & \boldsymbol{I}_n & & \\ & & \boldsymbol{I}_n & \ddots & \\ & & & \ddots & \boldsymbol{I}_n \\ & & & & \boldsymbol{I}_n \end{bmatrix}$$

Assume that after t times transformation, the matrix is

$$\widetilde{\boldsymbol{H}}'_{(t)}=\begin{bmatrix} \boldsymbol{H} & & & & & & & & \\ x^{1}(1+x)^{e-2} & \boldsymbol{H} & & & & & & & \\ x^{2}(1+x)^{e-3} & x^{1}(1+x)^{e-2} & \boldsymbol{H} & & & & & & \\ \vdots & \vdots & \vdots & \ddots & & & & & \\ x^{t-1}(1+x)^{e-t} & x^{t-2}(1+x)^{e+1-t} & x^{t-3}(1+x)^{e+2-t} & \cdots & \boldsymbol{H} & & & & \\ x^{t} & x^{t-1}(1+x) & x^{t-2}(1+x)^{2} & \cdots & x^{1}(1+x)^{t-1} & (1+x)^{t} & x^{e-1}(1+x)^{t} & \cdots & x^{t+1}(1+x)^{t} \\ x^{t+1} & x^{t}(1+x) & x^{t-1}(1+x)^{2} & \cdots & x^{2}(1+x)^{t-1} & x(1+x)^{t} & (1+x)^{t} & \cdots & x^{t+2}(1+x)^{t} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x^{e-1} & x^{e-2}(1+x) & x^{e-3}(1+x)^{2} & \cdots & x^{e-t}(1+x)^{t-1} & x^{e-t-1}(1+x)^{t} & x^{e-t-2}(1+x)^{t} & \cdots & (1+x)^{t} \end{bmatrix} \tag{9}$$

Since $\dfrac{(1+x)^{e-1}}{(1+x)^{t}}=(1+x)^{e-1-t}=\sum_{k=0}^{e-1-t}\binom{e-1-t}{k}x^{k}$, row blocks $t+2$, $t+3$, …, e are added to row block $t+1$ by coefficients $\binom{e-1-t}{1}$, $\binom{e-1-t}{2}$, …, $\binom{e-1-t}{e-1-t}$ and column blocks $e-1$, $e-2$, …, $t+1$ are added to column blocks e, $e-1$, …, $t+2$ in sequence. The resulting $\widetilde{\boldsymbol{H}}'_{(t+1)}$ conforms to our assumption. Then after $e-1$ times transformation, we have $\boldsymbol{\Lambda}=\boldsymbol{R}_{e-1}\ldots\boldsymbol{R}_2\boldsymbol{R}_1\widetilde{\boldsymbol{H}}'\boldsymbol{C}_1\boldsymbol{C}_2\ldots\boldsymbol{C}_{e-1}$, where $\boldsymbol{R}=\boldsymbol{R}_{e-1}\ldots\boldsymbol{R}_2\boldsymbol{R}_1$ and $\boldsymbol{C}=\boldsymbol{C}_1\boldsymbol{C}_2\ldots\boldsymbol{C}_{e-1}$. In addition, matrix $\boldsymbol{\Lambda}$ is block-Toeplitz. This completes the proof.

Codeword $\boldsymbol{c}=[c_1^{(1)},\ c_1^{(2)},\ \ldots,\ c_1^{(e)},\ c_2^{(1)},\ \ldots,\ c_n^{(e)}]^{\mathrm{T}}$ can be permuted to $\boldsymbol{c}'=[c_1^{(1)},\ c_2^{(1)},\ \ldots,\ c_n^{(1)},\ c_1^{(2)},\ c_2^{(2)}\ \ldots,\ c_n^{(e)}]^{\mathrm{T}}$, which satisfies $\widetilde{\boldsymbol{H}}'\boldsymbol{c}'=\boldsymbol{0}$. Let $\boldsymbol{y}=\boldsymbol{C}^{-1}\boldsymbol{c}'=[y_1^{(1)},\ y_2^{(1)},\ \ldots,\ y_n^{(1)},\ y_1^{(2)},\ y_2^{(2)}\ \ldots,\ y_n^{(e)}]^{\mathrm{T}}$, which satisfies $\boldsymbol{\Lambda}\boldsymbol{y}=\boldsymbol{0}$. Then subcode $\boldsymbol{y}^{(i)}=[\boldsymbol{y}_1^{(i)},\ y_2^{(i)},\ \ldots,\ y_n^{(i)}]^{\mathrm{T}}$ can be viewed as a base code and the encoding of lifted QC-LDPC code is decomposed into the encoding of base codes.

The encoding of each subcode has two steps: 1) The multiplication of each row block by the previous encoded bits, and 2) The encoding of base code. If the permutation shifts are uniformly distributed over Z_e, the expectation of the number of 1s in matrix $\sum_{k=0}^{e-i}\binom{e-i}{k}\boldsymbol{\mathcal{H}}^{e-1-k}$ is $\dfrac{N_{nz}(i)}{e}\times nw$, where w is the average column weight and $N_{nz}(i)$ is the number of non-zero elements in the set $\left\{\binom{e-i}{k}\mid k\in[0,e-i]\right\}$. Matrix $\sum_{k=0}^{e-i}\binom{e-i}{k}\boldsymbol{\mathcal{H}}^{e-1-k}$ can be expressed in the polynomial form as $x^{i-1}(1+x)^{e-i}$, so let a_{e-i} denote the number of non-zero coefficients in $(1+x)^{e-i}$. It can be derived that the numbers of XOR operations sum to $\sum_{i=0}^{e-1}(i+1)a_i nw/e=O(3^q n)$ in step 1). For the second step, we follow the encoding method in Ref. [12], and its complexity is $O(n)+O(g^2)$, where g is the gap of the approximate triangulation of the base matrix. Since y has to be left multiplied by matrix $\boldsymbol{C}$, there is an additional complexity of $O(neq)$.

In this encoding scheme, we only need to preprocess the parity check matrix of the base code, which has a complexity of $O(g^3)$ [12]. However, if we treat the QC-LDPC code as an ordinary one, the complexity of preprocessing becomes $O(g^3e^3)$, and it accounts for much more memories to store the resulting matrix.

4 Conclusion

This paper provides an integrated process of constructing QC-LDPC codes by the cyclic lifting of protographs. The cycle enumeration algorithm can find target cycles efficiently and simulations demonstrate that the proposed greedy algorithm can eliminate more short cycles than the DES algorithm in the assignment of permutation shifts. Additionally, the encoding method introduced greatly reduces the complexity of preprocessing in encoding.

References

[1] Wang C C. Code annealing and the suppressing effect of the cyclically lifted LDPC code ensemble [C]//*IEEE Information Theory Workshop*. Chengdu, China, 2006: 86 – 90.

[2] Asvadi R, Banihashemi A H, Ahmadian-Attari M. Lowering the error floor of LDPC codes using cyclic liftings [J]. *IEEE Transactions on Information Theory*, 2011, **57** (4): 2213 – 2224.

[3] Asvadi R, Banihashemi A H, Ahmadian-Attari M. Design of finite-length irregular protograph codes with low error floors over the binary-input AWGN channel using cyclic liftings [J]. *IEEE Transactions on Communications*, 2012, **60**(4): 902 – 907.

[4] Divsalar D, Dolinar S, Jones C R, et al. Capacity-approaching protograph codes [J]. *IEEE Journal on Selected Areas in Communications*, 2009, **27**(6): 876 – 888.

[5] Tian T, Jones C R, Villasenor J D, et al. Selective avoidance of cycles in irregular LDPC code construction [J]. *IEEE Transactions on Communications*, 2004, **52** (8): 1242 – 1247.

[6] Di C, Proietti D, Telatar I E, et al. Finite-length analysis of low-density parity-check codes on the binary erasure channel [J]. *IEEE Transactions on Information Theory*, 2002, **48**(6): 1570 – 1579.

[7] Richardson T J. Error floors of LDPC codes [C]//41*th*

Allerton Conference on Communication Control and Computing. Urbana, USA, 2003: 1426 - 1435.

[8] Sharon E, Litsyn S. Constructing LDPC codes by error minimization progressive edge growth [J]. *IEEE Transactions on Communications*, 2008, **56**(3): 359 - 368.

[9] Fossorier M P C. Quasi-cyclic low-density parity-check codes from circulant permutation matrices [J]. *IEEE Transactions on Information Theory*, 2004, **50**(8): 1788 - 1793.

[10] Hu X Y, Eleftheriou E, Arnold D M. Regular and irregular progressive edge growth Tanner graphs [J]. *IEEE Transactions on Information Theory*, 2005, **51**(1): 386 - 398.

[11] Richardson T J, Urbanke R L. The capacity of low-density parity-check codes under message-passing decoding [J]. *IEEE Transactions on Information Theory*, 2001, **47**(2): 599 - 618.

[12] Richardson T J, Urbanke R L. Efficient encoding of low-density parity-check codes [J]. *IEEE Transactions on Information Theory*, 2001, **47**(2): 638 - 656.

基于原型图循环提升的高效 QC-LDPC 码构造和编码

梁 原 张树林 顾品标 吴乐南

(东南大学信息科学与工程学院, 南京 210096)

摘要: 通过原型图的循环提升可方便地构造准循环低密度奇偶校验(QC-LDPC)码. 为了保证 QC-LDPC 码的性能,消除 Tanner 图中的短环,首先设计一种算法用于找出原型图中的有害短环,然后提出一种贪婪算法用于对提升后的校验矩阵中的单位循环位移子阵分配适合的循环位移量. 与已有的 DES 算法相比,所提出的贪婪算法在分配循环位移量时施加了更多的限制条件来提升性能,仿真结果表明它比 DES 算法能消除更多的短环. 当提升因子为 2 的整数次幂时,证明了所得 QC-LDPC 码的校验阵可转化成分块下三角阵的形式. 利用该性质,由原型图循环提升得到的 QC-LDPC 码仅需对基矩阵做预处理就可以实现编码,极大地降低了 QC-LDPC 码的编码复杂度.

关键词: 低密度奇偶校验(LDPC)码;准循环 LDPC 码;循环提升;原型图 LDPC 码

中图分类号: TN911. 21