EFFICIENT ENCODING OF CYCLE CODES: A GRAPHICAL APPROACH

Jin Lu and José M. F. Moura and Haotian Zhang

Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213
{Jinlu, moura}@ece.cmu.edu

ABSTRACT

The paper presents a low-complexity encoding algorithm for cycle codes–low-density parity-check (LDPC) codes with column weight $j = 2$. For a cycle code of block length n, the encoding complexity of the algorithm we present is $O(n)$ whereas the conventional encoding method has complexity $O(n^2)$. We achieve the linear-complexity encoder through graph analysis. First, we introduce particular Tanner graphs named pseudo-trees and prove that any pseudo-tree can be encoded in linear time. Second, we show that any cycle code can be represented by the union of pseudo-trees. We show that combining these two ideas, cycle codes are linear-time encodable.

1. INTRODUCTION

LDPC codes [1] are excellent error-correcting codes with performance close to the Shannon Capacity [2]. LDPC codes of column weight $j = 2$ are known as cycle codes [3]. Though distance properties of cycle codes are not as good as LDPC codes of column weight $j \geq 3$ [1], reference [5] shows that cycle codes have low *decoding* complexity and, when concatenated with Reed-Solomon codes, show better error-correcting performance than other LDPC codes. In addition, reference [7] shows that cycle codes with large girth have good error-correcting performance. Hence, cycle codes are promising in the data storage areas and other applications.

The high encoding complexity prevents the application of cycle codes. The conventional way to encode cycle codes is to multiply the data words $\overline{s}$ with a matrix called generator matrix $\mathbf{G}$, i.e., code words $\overline{x} = \mathbf{G} \cdot \overline{s}$. Though the parity-check matrix $\mathbf{H}$ for cycle codes is sparse, the associated generator matrix $\mathbf{G}$ is no longer sparse. Assume the block length of a cycle code is n, the encoding complexity is $O(n^2)$, which is significant. It is known that by matrix manipulation, the coefficient of the quadratic term in the encoding complexity can be made small [4]. In this paper, we prove that cycle codes can be actually encoded in exactly linear time.

This work was supported by the Data Storage Systems Center at Carnegie Mellon University.

2. "LABEL-AND-DECIDE" ENCODING ALGORITHM

LDPC codes can be described by a bipartite graph called Tanner graph [6], in which each bit becomes a bit node and each parity-check equation becomes a check node. If a bit is constrained by a parity-check equation, there is an edge connecting the associated bit node and check node.

Initially Tanner graphs were developed to explain the decoding process for LDPC codes; in fact, they can be used for the encoding of LDPC codes as well. Let $\mathbf{H}$ be an $m \times n$ parity-check matrix for an LDPC code $\mathbf{C}$. To encode $\mathbf{C}$, we need to identify the bits that contain user data, namely, *information bits*, from all the n bits contained in the code word of $\mathbf{C}$. The bits other than the information bits in the code word are called redundant bits or *parity bits*. Given the values of all the information bits, the encoding process is to compute the values of all the parity bits. We identify information bits and parity bits through a labeling process on the Tanner graph. When a bit node is labeled as an information bit, we simply assign its numerical value to it; when a bit node is labeled to be a parity bit, we simultaneously compute its value. This encoding approach is named *Label-and-decide*.

Example. Assume that the parity-check matrix $\mathbf{H}$ for an (8, 3) linear code is given by

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Its associated Tanner graph is cycle-free, as shown in Figure 1.

In Figure 1, initially, all the bit nodes are unlabeled. First, we randomly pick a bit node, say, x_5, to be an information bit. According to the parity-check equation c_2, we know that the value of the bit x_1 depends only on the value of the bit x_5 such that $x_1 = x_5$. Therefore, x_1 should be labeled as a parity bit if we label the bit x_5 as an information

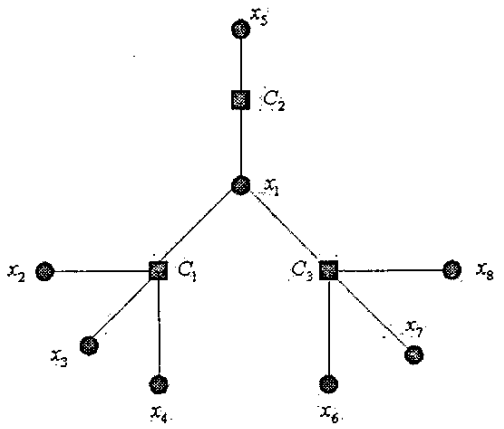

Fig. 1. The Tanner graph for $\mathbf{H}$

bit. After the values of the bits x_5 and x_1 are already fixed, we can not yet determine the value of the bit x_2. Therefore, x_2 ought to be labeled as an information bit. Similarly, the bit x_3 is labeled as an information bit. Next, we label the bit x_4 as a parity bit and compute its value as $x_4 = x_1 \oplus x_2 \oplus x_3$ according to the parity-check equation c_1. In a like manner, the bits x_7 and x_8 are labeled as information bits and the bit x_6 is labeled as a parity bit such that $x_6 = x_1 \oplus x_7 \oplus x_8$.

By the above labeling process, we decide the systematic component of the codeword $(x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8)$ to be $\overline{s} = (x_2\ x_3\ x_5\ x_7\ x_8)$ and the parity component to be $\overline{p} = (x_1\ x_4\ x_6)$.

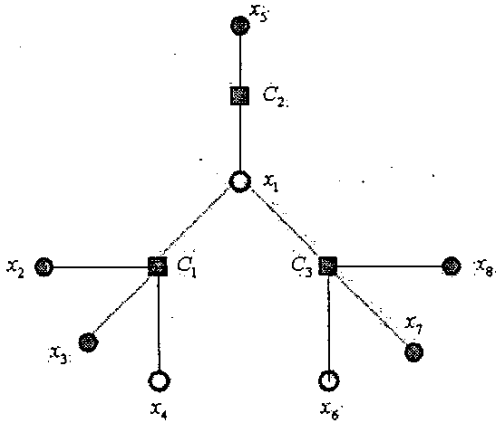

Fig. 2. Labeling bit nodes on the Tanner graph in Fig. 1

The above example is for a cycle-free Tanner graph. In the next section, we will apply it to cycle codes through graph analysis.

3. LINEAR COMPLEXITY ENCODER FOR CYCLE CODES

We achieve linear-complexity encoding for cycle codes as follows: In the first step, we propose particular Tanner graphs named pseudo-trees and prove that any pseudo-tree can be encoded in linear time. Second, we prove that any cycle code can be represented by the union of pseudo-trees. We then show that combining these two ideas, cycle codes can be encoded in linear-time.

We will say that a Tanner graph that satisfies conditions (i) through (iii) is a pseudo-tree.

(i) It is composed of $2P+1$ tiers where P is a positive integer. We number these tiers from 1 to $2P+1$, starting from the top tier to the bottom tier. The $(2i-1)^{\text{th}}$ tier $(i = 1, \ldots, P+1)$ contains only bit nodes while the $(2l)^{\text{th}}$ tier $(l = 1, \ldots, P)$ contains only check nodes.

(ii) Each bit node in the first tier has degree one. Each of them is connected to one and only one check node in the second tier.

(iii) For each check node C_a in the $(2l)^{\text{th}}$ tier where l can take any value from 1 to P, there is only one bit node x_a in the $(2l-1)^{\text{th}}$ tier that connects to C_a and there are no other bit nodes in the upper tiers that connect to C_a. We call x_a the *parent* of C_a and call C_a the *child* of x_a.

Figure 3 on the left shows a pseudo-tree $\mathbf{T_1}$ with seven tiers.

Lemma 1 *Any LDPC code represented by a pseudo-tree is linear time encodable.*

Proof: Let a pseudo-tree contain $2P+1$ tiers, n bit nodes, and m check nodes. Condition (iii) guarantees that each check node has one and only one parent in the immediate upper tier, so there are m parents for the check nodes. We label these m parents as parity bits and the other $n-m$ bit nodes as information bits.

The inputs of the encoder provide the values for all the information bits. The task of the encoder is to compute the values for all the parity bits. Let x_a be an arbitrary parity bit and C_a be the child of x_a. The tier that contains x_a is the $(2i-1)^{\text{th}}$ tier. The value of x_a is determined by the parity-check equation represented by C_a. According to condition (iii), all the bit nodes contained by C_a except for x_a are in tiers below the $(2i-1)^{\text{th}}$ tier. Therefore, the value of x_a depends only on the values of the bit nodes below the $(2i-1)^{\text{th}}$ tier. We compute values of the parity bits tier by tier, starting from the $(2P-1)^{\text{th}}$ tier. This encoding process is repeated until the values of all the parity bits in the 1^{st} tier are known.

Let k_l, $l = 1, 2, \ldots, m$, denote the number of bits contained in the l^{th} parity-check equation. When the l^{th} parity-check equation is used to determine the value of a parity bit,

$(k_l - 2)$ XOR operations are needed. Therefore, overall $\sum_{l=1}^{m}(k_l - 2)$ XOR operations are required to obtain the values for all the m parity bits. Hence, the encoding process above is accomplished in linear time. This completes the proof.

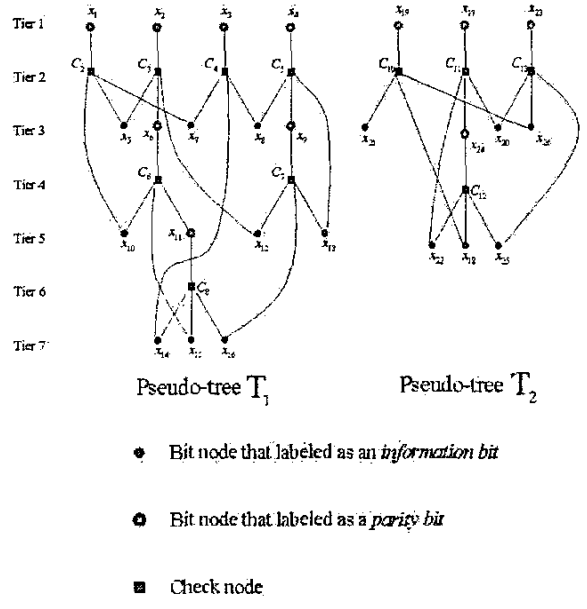

Fig. 3. The associated pseudo-trees of the Tanner graph shown in Fig. 4

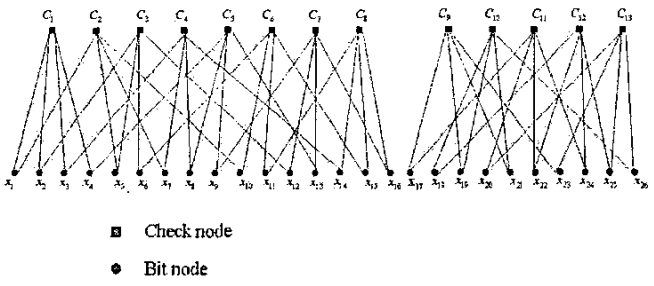

Fig. 4. Tanner graph for a cycle code

Lemma 2 relates general Tanner graphs for cycle codes to pseudo-trees.

Lemma 2 *Any cycle code can be represented as the union of pseudo-trees.*

Proof: Let $\mathbf{C}$ represent an LDPC code of column weight 2. $\mathbf{H}$ denotes the parity-check matrix for $\mathbf{C}$ and $\mathbf{G}$ denotes the Tanner graph for $\mathbf{C}$. $\mathbf{G}$ can be decomposed into q disjoint subgraphs: $\mathbf{G}_1, \ldots, \mathbf{G}_q$. Each subgraph $\mathbf{G}_i$, $i = 1, \ldots, q$, is a connected graph in the sense that every two vertices in $\mathbf{G}_i$ are connected to each other by at least one path. Now we restrict our attention to $\mathbf{G}_i$. Let $\mathbf{G}_i$ contain n_i bit nodes and m_i check nodes. Each of the n_i bit nodes in $\mathbf{G}_i$ is of degree 2 since each column of $\mathbf{H}$ contains exactly two 1's. Each of the m_i check nodes represents a row of $\mathbf{H}$. The sum of all the m_i rows in the binary field is a vector of all 0's because each of the n_i bits appears exactly twice in the summation. Therefore, at least one of the m_i rows is linearly dependent. We can remove the linearly dependent row without changing the underlying code structure of $\mathbf{C}$. Say, we remove the l^{th} row of $\mathbf{H}$. Correspondingly, the associated check node C_l and all the edges that are incident on C_l are removed from $\mathbf{G}_i$ while the remaining Tanner graph still represents the code $\mathbf{C}$. Let $x_1^l, x_2^l, \ldots, x_k^l$ be the k bit nodes that connect to C_l in $\mathbf{G}_i$. After removing C_l from $\mathbf{G}_i$, each x_s^l, $s = 1, 2, \ldots, k$, connects to only one check node in the subgraph $\mathbf{G}_i$. We place $x_1^l, x_2^l, \ldots, x_k^l$ in the first tier and put the check nodes that connect to $x_1^l, x_2^l, \ldots, x_k^l$ in the second tier. Edges connect bit nodes in the first tier and their associated check nodes in the second tier. The bit nodes other than $x_1^l, x_2^l, \ldots, x_k^l$ that connect to the check nodes in the second tier are put in the third tier. In a like manner, except for the check nodes in the second tier, the check nodes that connect to the bit nodes in the third tier are placed in the fourth tier. Successive tiers are constructed in similar manner. As $\mathbf{G}_i$ is a connected graph, the construction goes on until all the n_i bit nodes and all the m_i check nodes but the removed check node C_l in $\mathbf{G}_i$ have been included in the constructed multi-layer structure.

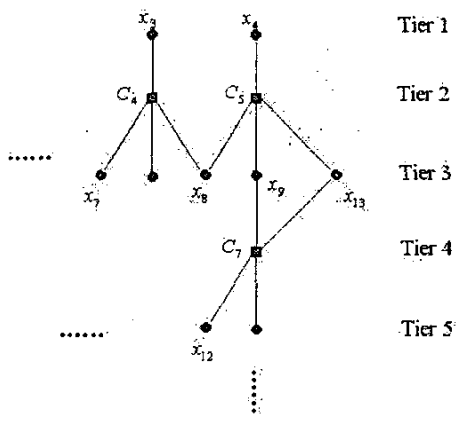

Fig. 5. A multi-layer structure but not a pseudo-tree (C_7 has two parents: x_9 and x_{13})

Up to now, the multi-tier structure constructed conforms

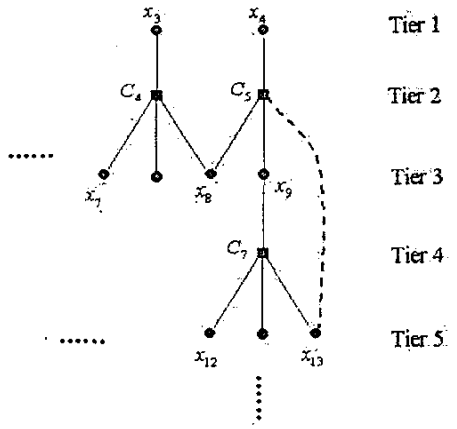

Fig. 6. The pseudo-tree that evolves from the multi-layer structure shown in Fig. 5

to the conditions (i) and (ii), however, condition (iii) may not be satisfied. For example, in Figure 5, the check node C_7 in the 4^{th} tier is connected to two bit nodes x_9 and x_{13} in the 3^{rd} tier, which violates condition (iii). To satisfy condition (iii), we further adjust the positions of the bit nodes. If a check node in the $(2i)^{\text{th}}$ tier is connected to d bit nodes in the upper tiers. We randomly pick one bit node from the d bit nodes and leave its position unchanged. Next, we drag the other $d-1$ bit nodes from their initial positions to the $(2i+1)^{\text{th}}$ tier. To illustrate, let us focus on Figure 5 again. We drag the bit node x_{13} from the 3^{rd} tier to the 5^{th} tier. The newly formed graph is shown in Figure 6, which follows the conditions (i), (ii), and (iii). Hence, a pseudo-tree is formed in Figure 6. By tuning the positions of the bit nodes in this way, we can transform the multi-tier graph constructed into a pseudo-tree. Therefore, each $\mathbf{G}_i$, $i = 1, \ldots, q$, corresponds to an equivalent pseudo-tree. Since the code $\mathbf{C}$ is represented by the union of $\mathbf{G}_1, \ldots,$ up to $\mathbf{G}_q$, $\mathbf{C}$ can also be represented by the union of q pseudo-trees. This completes the proof.

We now show that the encoding complexity of a cycle code is $O(n)$. If the Tanner graph for the cycle code $\mathbf{C}$ is represented as q pseudo-trees: $\mathbf{T}_1, \ldots, \mathbf{T}_q$, $\mathbf{C}$ contains n bits and $\mathbf{T}_i$, $i = 1, 2, \ldots, q$, contains n_i bit nodes. Then $n = \sum_{i=1}^{q} n_i$. Lemma 1 guarantees that the encoding complexity for $\mathbf{T}_i$ is $O(n_i)$. Therefore, the complexity to encode $\mathbf{C}$ is $\sum_{i=1}^{q} O(n_i) = O(\sum_{i=1}^{q} n_i) = O(n)$, i.e., linear complexity.

O(n) Encoding Algorithm

We illustrate by an example the $O(n)$ encoding algorithm. Figure 3 shows two pseudo-trees $\mathbf{T_1}$ and $\mathbf{T_2}$ for the Tanner graph in Figure 4. According to $\mathbf{T_1}$, the bit nodes x_{11}, x_6, x_9, x_1, x_2, x_3, and x_4 are labeled as parity bits; According to $\mathbf{T_2}$, the bit nodes x_{24}, x_{19}, x_{17}, and x_{23} are labeled as parity bits; The other bit nodes are labeled as information bits. Then we encode as follows:

a. Compute x_{11} by the equation $x_{11} = x_{14} \oplus x_{15} \oplus x_{16}$ (The symbol $\oplus$ denotes addition in the binary field);

b. Compute x_6 and x_9 such that $x_6 = x_{10} \oplus x_{15} \oplus x_{11}$ and $x_9 = x_{12} \oplus x_{16} \oplus x_{13}$;

c. Compute x_1, x_2, x_3, and x_4: $x_1 = x_{10} \oplus x_5 \oplus x_7$, $x_2 = x_5 \oplus x_6 \oplus x_{12}$, $x_3 = x_7 \oplus x_{14} \oplus x_8$, and $x_4 = x_8 \oplus x_9 \oplus x_{13}$;

d. Compute x_{24}: $x_{24} = x_{22} \oplus x_{18} \oplus x_{25}$;

e. Compute x_{19}, x_{17}, and x_{23}: $x_{19} = x_{21} \oplus x_{18} \oplus x_{26}$, $x_{17} = x_{22} \oplus x_{24} \oplus x_{20}$, $x_{23} = x_{20} \oplus x_{26} \oplus x_{25}$.

We summarize our results in the main theorem:

Theorem 1 *Any cycle code can be encoded in linear time.*

Proof: By Lemma 2, any cycle code can be represented by the union of several pseudo-trees. Let $\mathbf{C}$ be a cycle code and $\mathbf{C} = \mathbf{T_1} \cup \cdots \cup \mathbf{T_q}$ where $\mathbf{T_i}$, $i = 1, \ldots, q$, is a pseudo-tree. Further, let $\mathbf{T_i}$ contain m_i check nodes and the average number of bits contained in the m_i parity-check equations be $\overline{k_i}$. According to Lemma 1, the computational complexity to encode $\mathbf{T_i}$ is $m_i(\overline{k_i} - 2)$. Therefore, the complexity to encode all the q pseudo-trees is $\sum_{i=1}^{q} m_i(\overline{k_i} - 2)$. For an $(n, 2, k)$ regular LDPC code, the encoding complexity is $(\frac{2n}{k} - q)(k-2) \approx 2n$. Therefore, any cycle code can be encoded in linear time. This completes the proof.

It is of interest to relate Theorem 1 to the parity-check matrix of the code. This is done in the following corollary:

Corollary 1 *The parity-check matrix* $\mathbf{H}$ *of a cycle code can be transformed into an equivalent upper-triangular matrix* $\mathbf{H_{upper}}$ *simply by permuting the rows and columns of* $\mathbf{H}$.

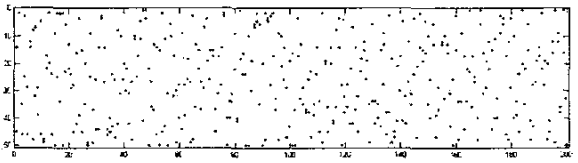

Fig. 7. The parity-check matrix for a (200, 2, 8) LDPC code

For example, by permutation of rows and columns, we transform the parity-check matrix shown in Figure 7 to an upper-triangular matrix shown in Figure 8. Corollary 1 implies that any cycle code can be viewed as a causal system.

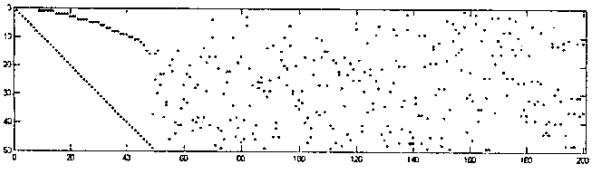

Fig. 8. An equivalent upper-triangular matrix for the LDPC code in Figure 7

4. CONCLUSION

In this paper, we propose an efficient encoding algorithm named "label-and-decide" for cycle codes. By introducing a hierarchical Tanner graph called pseudo-trees and an algorithm to transform Tanner graphs of cycle codes into union of pseudo-trees, we achieve linear encoding. This is quite desirable in many applications.

5. REFERENCES

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

[2] D. J. C. Mackay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions. on Information Theory*, vol. 45, no. 2, pp. 399-431, Mar. 1999.

[3] N. Wiberg. *Codes and Decoding on General Graphs*. PhD thesis, Dept. of Electrical Engineering, Linköping, Sweden, 1996. Linköping studies in Science and Technology. Dissertation No. 440.

[4] T. Richardson and R.Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. on Information Theory*, vol. 47, no. 2, Feb. 2001, pp. 638-656.

[5] H. Song and J. Liu and B. V. K. Vijaya Kumar, "Low complexity LDPC codes for partial response channels," *IEEE Globecom 2002*, vol. 2, pp. 1294-1299, Nov. 2002, Taipei, Taiwan, R.O.C

[6] R. Michael Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Information Theory*, vol. IT-27, no. 5, Sep. 1981, pp. 533-547.

[7] José M. F. Moura, J. Lu and H. Zhang, "Structured Low-Denstiy Parity-Check Codes: Methods to design regular LDPC codes with large girth," to be published, *IEEE Signal Processing Magazine*, vol. 21, no. 1, January 2004.