International Journal of GEOMATE, Nov., 2016, Vol. 11, Issue 27, pp. 2804-2807
Geotec., Const. Mat. & Env., ISSN: 2186-2982(Print), 2186-2990(Online), Japan

MESSAGE PASSING AND LOOPS SOLUTION ALGORITHM BASED ON CUT-NODE TREE

Zhang Huanming[1], Xian Kaiyi[2], Feng Lijun[3] and Hu Chaokang[4]

[1,2,3,4]Electronics and Information Engineering Institute
Foshan University, Foshan, Guangdong, China, 528000

ABSTRACT: Low-density parity-check (LDPC) codes are forward error-correction and linear block codes. An LDPC code can be described by a bipartite graph called Tanner graph[1]. Loops, especially short loops in tanner graph, degrade the performance of LDPC decoder, because they affect the independence of the extrinsic information exchanged in the iterative decoding. In this paper, based on graph theory and Tanner graph, the loop structure in LDPC codes are studied carefully, a new notion, cut-node tree, is proposed to describe LDPC codes. Cut-node tree has full information of Tanner graph. So all loop features in LDPCs can be calculated relatively easy by a computer. Traditional message passing in graph is improved to avoid repeated iteration of information, a new decoding schemes for LDPC codes is proposed and can suppress repeated iteration of information in SPA. The results help to further research on related field.

Keywords: LDPC; belief propagation (BP); iterative decoding; loop; SPA

1. INTRODUCTION

In coding theory, Tanner graph, named after Michael Tanner, is a bipartite graph used to state constraints or equations which specify error correcting codes[2]. See Fig.1. They are used to construct longer codes from smaller ones. Both encoders and decoders employ these graphs extensively[3][4].

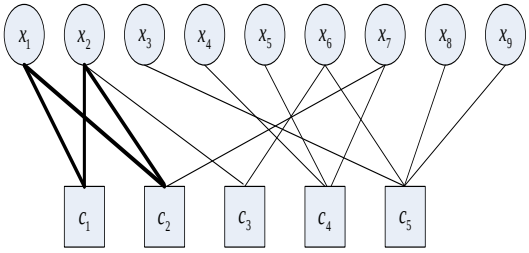

Fig.1 Tanner graph of an LDPC code

In communication system, the transmitted random vector $x=\{x_1,\cdots,x_N\}$ is not observed; instead received noisy vector $y=\{y_1,\cdots,y_N\}$.

N is the length of codeword，Parity check equation vector is $c=\{c_1,c_2,\cdots,c_M\}$, M is the number of equations. $f=\{f_1{}^a,\cdots,f_N{}^a\}$ represents initial information about transmitted codeword.

Where v_i is the ith variable node, and c_j is the jth check equation.

The belief propagation(BP) of LDPC code states as follows[5]-[7]：

R_{ji}^a is check information from check node c_j to variable node v_i, and Q_{ij}^a variable information from v_i to check node c_j.See Fig.2:

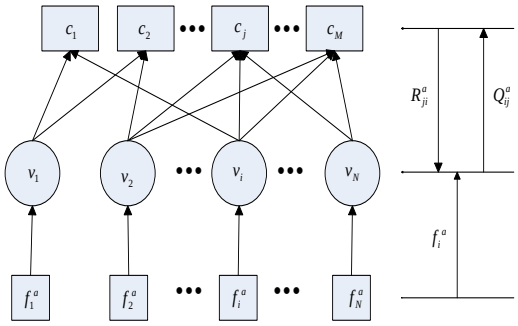

Fig.2 Iterative Decoding Algorithm

$$R_{ji}^0(t+1)=\frac{1}{2}\left(1+\prod_{j'\in N(i)\backslash j}\left(1-2Q_{ij'}^1(t)\right)\right) \quad (1)$$

$$R_{ji}^1(t+1)=1-R_{ji}^0(t+1) \quad (2)$$

$$Q_{ij}^0(t+1)=\alpha_{ij}(1-P_i)\prod_{i'\in M(j)\backslash i}R_{ji}^0(t+1) \quad (3)$$

$$Q_{ij}^1(t+1)=\alpha_{ij}P_i\prod_{i'\in M(j)\backslash i}R_{ji}^1(t+1) \quad (4)$$

Where α_{ij} denotes a normalization constant, $Q_{ij}^0(t+1)+Q_{ij}^1(t+1)=1$. See Fig.3[8].

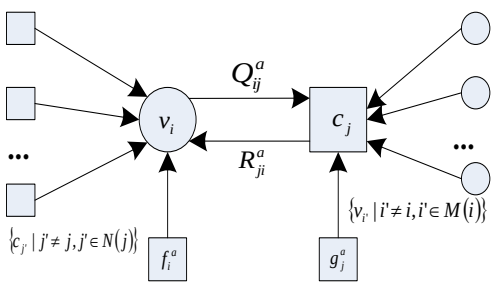

Fig.3 Updating rule for message passing

2. THEORY OF CUT-NODE TREE

In order to solute loops of LDPC code, avoiding repeated information iteration. Tanner graph is re-drew as following principle: Choosing an element '1' in *H*, its variable (or check) node considered root node, check (or variable) nodes connected to the variable (or check) node as 1st order child-nodes, a current node, once appearance in ancestor node or sibling node, will be cut and forbid to grow and become an end node like a leaf, but not a leaf actually. And so forth, at last a cut-node tree can be got. If all nodes are connected, a single cut-node tree can be got, otherwise it is forest. Repeating this process until all end nodes are either cut-nodes or leaf nodes. See Fig.4:

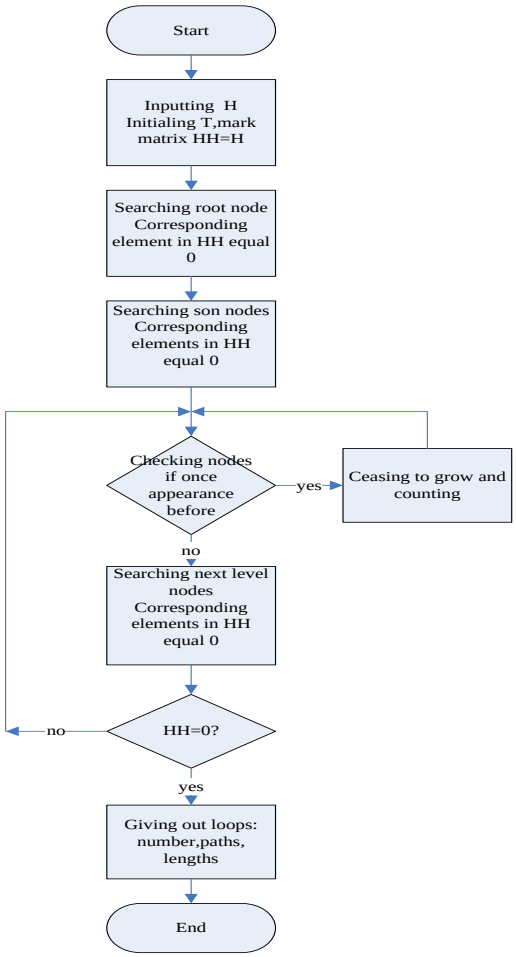

Fig.4 Flow process chart of algorithm

HH is mark matrix, when a node appears in graph, the element in HH changes to zero; every element is zero in HH represents end of algorithm.

3. IMPLEMENT OF PRINCIPLE AND RESULTS

First, a node (variable node or check node), for instance, variable node $v_i, h_{ij} = 1$, should be chosed as root node, its son nodes are those which connect to it, according to this principle, all child-nodes can be obtained. This process can be implemented by computer simulation in matlab's cell array, and express *H* by means of cut-node tree[6].

For example, an LDPC code with check matrix:

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

It has the following cut-node tree graph (See Fig.5):

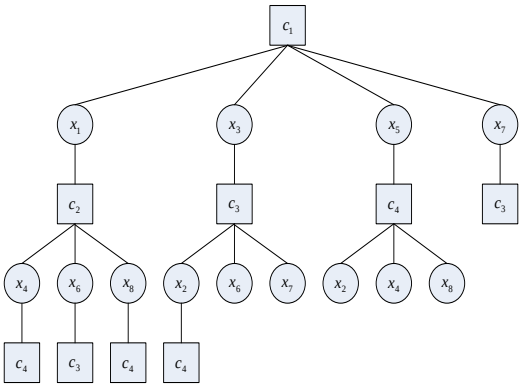

Fig.5 Tree Graph of LDPC codes

For another instance, an LDPC code with check matrix *H*:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad (6)$$

Following tree matrix expressed by matlab can be get, In fact, tree matrix matches along with Tree of Graph, [i j k l m], [i j] states current node, [k l] states father node, m states times being cut.

Root node: [1 1 0 0]
Level1: [2 1 1 1]
Level2: [2 5 2 1] [2 6 2 1] [2 7 2 1]
Level3: [3 5 2 5] [4 6 2 6] [5 7 2 7]
Level4: [3 2 3 5] [3 8 3 5] [3 9 3 5] [4 3 6 3] [4 8 6 8] [4 10 4 6] [5 4 5 7] [5 9 5 7] [5 10 5 7]
Level5: [1 2 3 2] [4 8 3 8 1] [5 9 3 9 1] [1 3 4 3] [3 8 4 8 1] [5 10 4 10 1] [1 4 5 4] [3 9 5 9 1] [4 10 5 10 1]

Level6: [1 1 1 2 1] [1 3 1 2 1] [1 4 1 2 1] [1 1 1 3 2] [1 2 1 3 1] [1 4 1 3 2] [1 1 1 4 3] [1 2 1 4 2] [1 3 1 4 3]

So，searched by a computer, all loops and features of Eq.5 can be get by this algorithm. See Table 1.

Loop 1: [4 8]-[3 8]-[3 5]-[2 5]-[2 6]-[4 6]-[4 8]
Loop 2: [5 9]-[3 9]-[3 5]-[2 5]-[2 7]-[5 7]-[5 10]
Loop 3: [5 10]-[4 10]-[4 6]-[2 6]-[2 7]-[5 7]-[5 10]
Loop 4: [1 1]-[1 2]-[3 2]-[3 5]-[2 5]-[2 1]-[1 1]
Loop 5: [1 3]-[1 2]-[3 2]-[3 5]-[2 5]-[2 6]-[4 6]-[4 3]-[1 3]
Loop 6: [1 4]-[1 2]-[3 2]-[3 5]-[2 5]-[2 7]-[5 7]-[5 4]-[1 4]
Loop 7: [1 1]-[1 3]-[4 3]-[4 6]-[2 6]-[2 1]-[1 1]
Loop 8: [1 1]-[1 3]-[4 3]-[4 6]-[2 6]-[2 5]-[3 5]-[3 2]-[1 2]-[1 1]
Loop 9: [1 4]-[1 3]-[4 3]-[4 6]-[2 6]-[2 5]-[3 5]-[3 2]-[1 2]-[1 4]
Loop 10: [1 4]-[1 3]-[4 3]-[4 6]-[2 6]-[2 7]-[5 7]-[5 4]-[1 4]
Loop 11: [1 1]-[1 4]-[5 4]-[5 7]-[2 7]-[2 1]-[1 1]
Loop 12: [1 1]-[1 4]-[5 4]-[5 7]-[2 7]-[2 5]-[3 5]-[3 2]-[1 2]-[1 1]
Loop 13: [1 1]-[1 4]-[5 4]-[5 7]-[2 7]-[2 6]-[4 6]-[4 3]-[1 3]-[1 1]
Loop 14: [1 2]-[1 4]-[5 4]-[5 7]-[2 7]-[2 5]-[3 5]-[3 2]-[1 2]
Loop 15: [1 2]-[1 4]-[5 4]-[5 7]-[2 7]-[2 6]-[4 6]-[4 3]-[1 3]-[1 2]
Loop 16: [1 3]-[1 4]-[5 4]-[5 7]-[2 7]-[2 5]-[3 5]-[3 2]-[1 2]-[1 3]

Table 1: Features of loops about above example

Parameters	Values
Sparsity	0.4
Loops	16
Total length of loops	108
Average length	6.75
Girth	6
Maximum length	10
Loop relativity	3.18

Here, *H* is just a situation of single tree, with 15 cut-nodes, no leaf node and 16 loops. In a graph with cycle，Cut-node tree graph can be got by cutting all loops, See Fig.5. Cut-node tree has all characters with Tanner graph. Further, Message-passing form for cut-node tree abides by following rule:

In a tree, Message-passing fellows a two-pass form, first sweeping upwards from leaves to a node designated as the root. (See Fig.6)

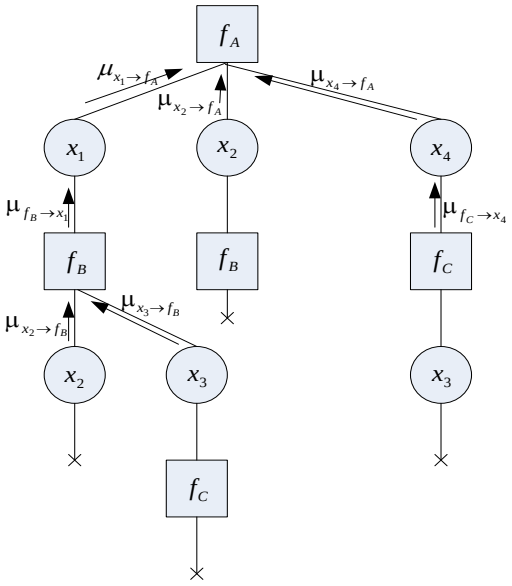

Fig.6 Message passing upwards over cut-node tree graph

and then downwards from the root node to leaves.(See Fig.6)

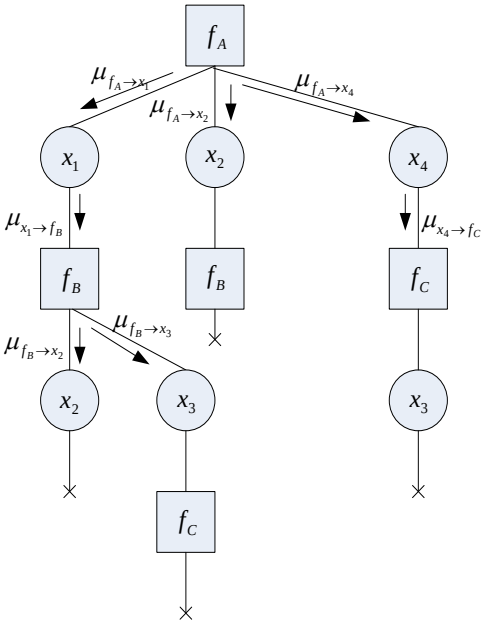

Fig.7 Message passing downwards over cut-node tree graph

4. CONLUSION

A new method to describe graph of LDPC codes is provided in this paper, it aims to solute loops of LDPC codes and message passing algorithm over cut-node tree. For a large matrix *H*, it is difficult to solute all loops and their features, because loops convolve each other. Features of loops have certain relationship with performances. In this paper, a cut-node tree can

be expressed by a certain matrix by matlab's cell array. By computer simulation, cut-node tree can get right answer and gives out method to calculate loops of LDPC codes and cut-node tree graph, and message passing algorithm over cut-node tree can avoid repeated iteration of information. The results assist to further research on related field.

5. REFERENCE

[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 Trans. Inform. Theory, vol. 45, 399-431,Mar. 1999.

[3] R. M. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Inf Theory, pp. 533-547, Sept. 1981

[4] R.Kschischang, B.J.Frey, and H.A.Loeliger, "Factor graphs and the sum-product algorithm" IEEE Trans. Inform. Theory, vol.47, no.2, pp.498–519, Feb 2001.

[5] N.Wiberg, Codes and decoding on general graphs, Linkoping Studies in Science and Technology, Ph.D. dissertation No. 440, Univ. Link oping, Sweden, 1996.

[6] Reinhard D, Graph Theory[M]. New York: Springer-Verlag, 1997

[7] Martin J. Wainwright, Sparse Graph Codes for Side Information and Binning, IEEE SIGNAL PROCESSING MAGAZINE [47], 47-57, SEPTEMBER 2007

[8] Martin Wainwright, Tommi Jaakkola, and Alan Willsky, Tree-based reparameterization analysis of belief propagation and related algorithms for approximate inference on graphs with cycles, ISIT2002, Lausanne, Switzerland, June 30-July 5, 2002

International Journal of GEOMATE, Nov., 2016, Vol. 11, Issue 27, pp.2804-2807.

MS No. 1272 received on July 31, 2015 and reviewed under GEOMATE publication policies.

Corresponding Author: **Zhang Huanming**