Journal of Signal and Information Processing, 2014, 5, 191-197
Published Online November 2014 in SciRes. http://www.scirp.org/journal/jsip
http://dx.doi.org/10.4236/jsip.2014.54020

Scientific Research

Tree Matrix Algorithm of LDPC Codes

Huanming Zhang

Electronics and Information Engineering Institute, Foshan University, Foshan, China
Email: huanmingzhang@126.com

Received 16 September 2014; revised 15 October 2014; accepted 8 November 2014

Abstract

LDPC codes are finding increasing use in applications requiring reliable and highly efficient information transfer over bandwidth. An LDPC code is defined by a sparse parity-check matrix and can be described by a bipartite graph called Tanner graph. Loops in Tanner graph prevent the sum-product algorithm from converging. Further, loops, especially short loops, degrade the performance of LDPC decoder, because they affect the independence of the extrinsic information exchanged in the iterative decoding. This paper, by graph theory, deduces cut-node tree graph of LDPC code, and depicts it with matrix. On the basis of tree matrix algorithm, whole depictions of loops can be figured out, providing of foundation for further research of relations between loops and LDPC codes' performance.

Keywords

LDPC, Belief Propagation (BP), Graph, Iterative Decoding, Loop, Cycle

1. Introduction

Probability distributions defined by graphs are used in literature of various fields, such as coding theory, artificial intelligence, and system theory.

LDPC [1] code can be depicted by its parity-check matrix H [2], $C = \left\{ x \in F_n \middle| xH^{\mathrm{T}} = 0 \right\}$ and its degree distribution of polynomial [3]:

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1} \tag{1}$$

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1} \tag{2}$$

Computing distributions of variable and check nodes in a Tanner graph is relatively efficient without cycle,

but complex with cycles, even difficult when cycles are short for large graph.

The belief propagation algorithm [4], or known as the sum-product algorithm [5] is decoding algorithm for graphical codes, such as convolution code, turbo code and LDPC code.

Belief propagation (BP) algorithm is optimal and produces exact solutions on a tree just in a finite number of iterations. Otherwise, BP algorithm may not converge. Moreover, when it doses converge, it provides approximate solution and the results vary in different conditions.

2. Background

In most common sense of the term, an undirected graph [6] consists of a set of vertices or nodes V together with a set of edges E, which are 2-element subsets of V (*i.e.*, an edge is related with two vertices, and the relation is represented as an unordered pair of the vertices with respect to the particular edge). See **Figure 1**.

$$V = \{1,2,3,4,5,6\}$$

$$E = \{\{1,2\},\{1,5\},\{3,4\},\{2,5\},\{5,6\}\}$$

$n = 6$

$m = 5$

Let $G = (V, E)$ be an undirected graph.

The following statements are equivalent,

1. G is a tree
2. Any two vertices in G are connected by a unique simple path
3. G is connected, but if any edge is removed from E, the resulting graph is disconnected
4. G is connected, and $|E| = |V| - 1$
5. G is acyclic, and $|E| = |V| - 1$
6. G is acyclic, but if any edge is added to E, the resulting graph contains a loop.

In fact, a tree is a connected graph without any cycles; a leaf is a vertex of degree 1. An internal vertex is a vertex of degree at least 2.

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

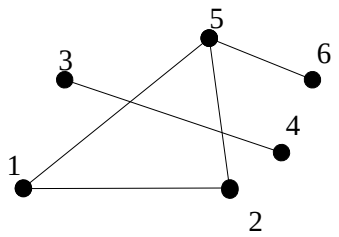

Figure 1. An example of a graph $G = \langle V, E \rangle$.

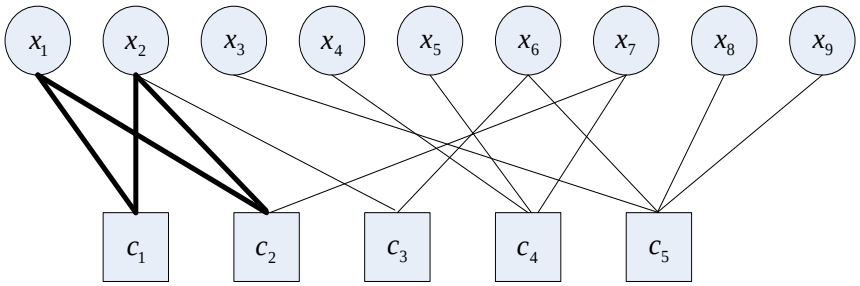

Figure 2. Tanner graph of an LDPC code (a cycle represented by 4 bold lines in figure).

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, c_j, jth check equation.

The belief propagation (BP) of LDPC code states as follows [7]-[11]:

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 **Figure 3**.

$$R_{ji}^0(t+1)=\frac{1}{2}\left(1+\prod_{j'\in N(i)\backslash j}\left(1-2Q_{ij'}^1(t)\right)\right) \tag{3}$$

$$R_{ji}^1(t+1)=1-R_{ji}^0(t+1) \tag{4}$$

$$Q_{ij}^0(t+1)=\alpha_{ij}(1-P_i)\prod_{i'\in M(j)\backslash i}R_{ji}^0(t+1) \tag{5}$$

$$Q_{ij}^1(t+1)=\alpha_{ij}P_i\prod_{i'\in M(j)\backslash i}R_{ji}^1(t+1) \tag{6}$$

where α_{ij} denotes a normalization constant, $Q_{ij}^0(t+1)+Q_{ij}^1(t+1)=1$. And message updating rule see **Figure 4** [12].

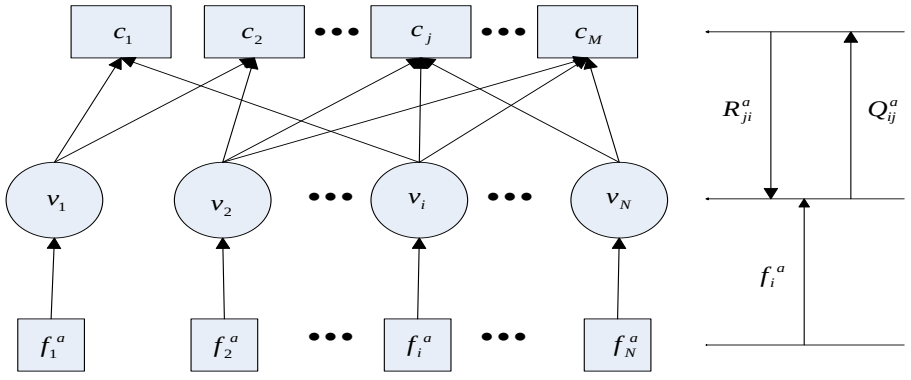

Figure 3. Iterative decoding algorithm.

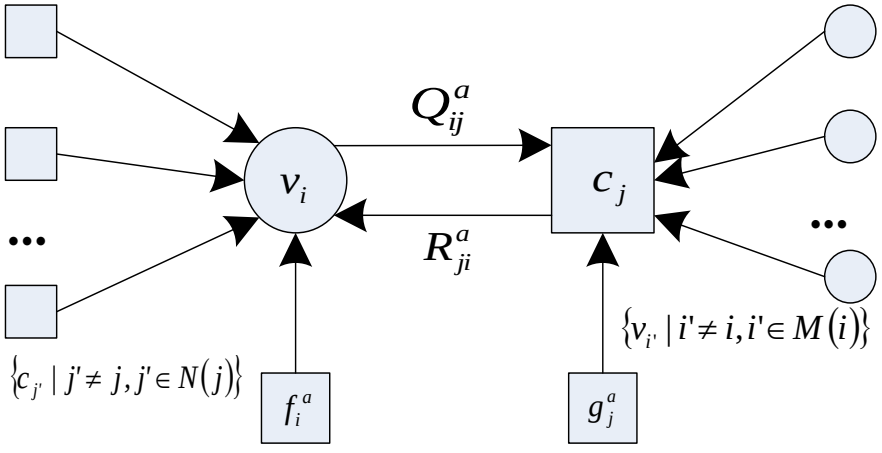

Figure 4. Updating rule for message passing.

3. Principle of Cut-Node Tree

In order to solute loops of LDPC code, 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 a end node like a leaf, but not a leaf actually. And so forth, at last a cut-node tree can be get. If all nodes are connected, a single cut-node tree can be get, otherwise it is forest. Repeating this process until all end nodes are either cut-node or leaf node. See **Figure 5**.

HH is mark matrix, every element in HH is zero represents end of algorithm.

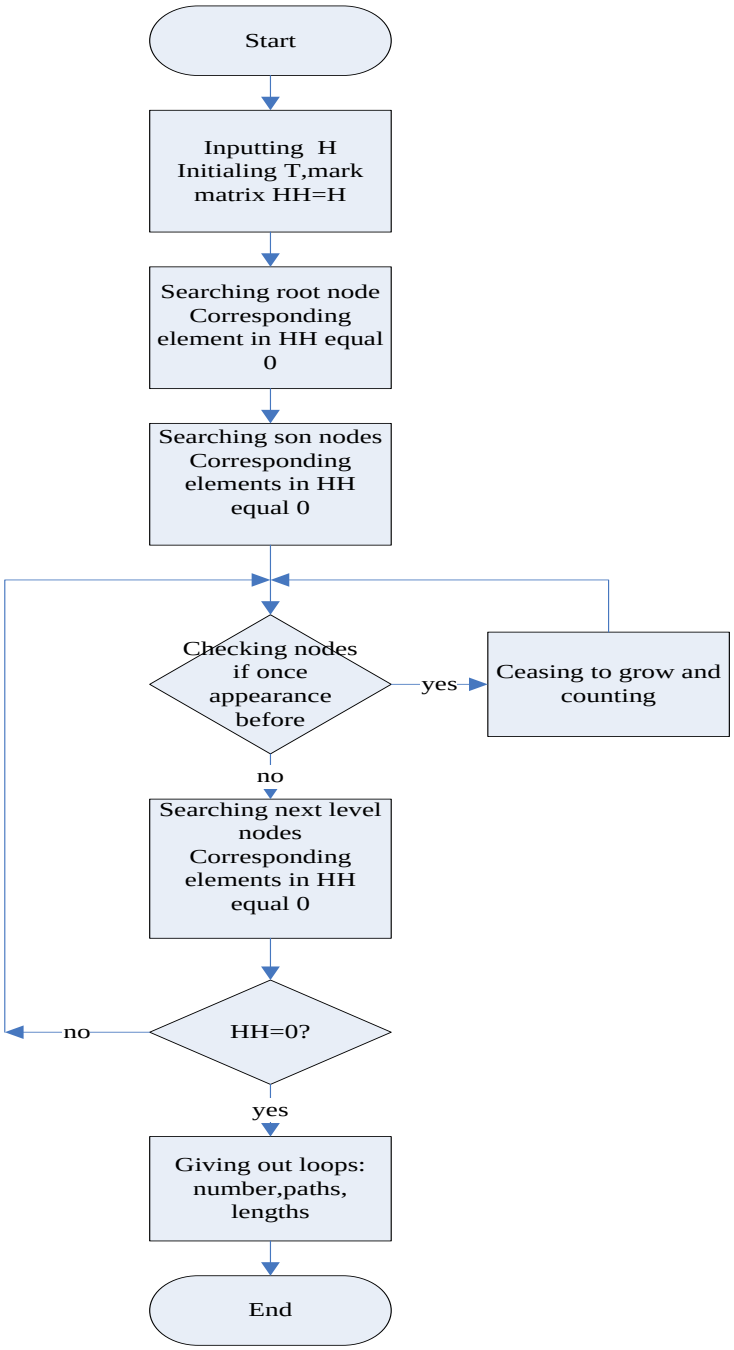

Figure 5. Flow process chart of algorithm.

4. Implement of Principle and Results

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

An example: for

$$H=\begin{bmatrix}1&0&1&0&1&0&1&0\\1&0&0&1&0&1&0&1\\0&1&1&0&0&1&1&0\\0&1&0&1&1&0&0&1\end{bmatrix} \tag{7}$$

has following cut-node tree graph (**Figure 6**).

Another example: For instance, an LDPC code with check matrix H:

$$H=\begin{bmatrix}1&1&1&1&0&0&0&0&0&0\\1&0&0&0&1&1&1&0&0&0\\0&1&0&0&1&0&0&1&1&0\\0&0&1&0&0&1&0&1&0&1\\0&0&0&1&0&0&1&1&0&1\end{bmatrix} \tag{8}$$

Following tree matrix expressed by matlab can be get:

$$\begin{array}{l}[1\ 1\ 0\ 0]\\[2\ 1\ 1\ 1]\\[2\ 5\ 2\ 1]\ [2\ 6\ 2\ 1]\ [2\ 7\ 2\ 1]\\[3\ 5\ 2\ 5]\ [4\ 6\ 2\ 6]\ [5\ 7\ 2\ 7]\\[3\ 2\ 3\ 5]\ [3\ 8\ 3\ 5]\ [3\ 9\ 3\ 5]\ [4\ 3\ 4\ 6]\ [4\ 8\ 4\ 6]\ [4\ 10\ 4\ 6]\ [5\ 4\ 5\ 7]\ [5\ 9\ 5\ 7]\ [5\ 1\ 0\ 5\ 7]\\[1\ 2\ 3\ 2]\ [4\ 8\ 3\ 8\ 1]\ [5\ 9\ 3\ 9\ 1]\ [1\ 3\ 4\ 3]\ [3\ 8\ 4\ 81]\ [5\ 10\ 4\ 10\ 1]\ [1\ 4\ 5\ 4]\ [3\ 9\ 5\ 91]\ [4\ 10\ 5\ 10\ 1]\\[1\ 1\ 1\ 2\ 1]\ [1\ 3\ 1\ 2\ 1]\ [1\ 4\ 1\ 2\ 1]\ [1\ 1\ 1\ 3\ 2]\ [1\ 2\ 1\ 3\ 1]\ [1\ 4\ 1\ 3\ 2]\ [1\ 1\ 1\ 4\ 3]\ [1\ 2\ 1\ 4\ 2]\ [1\ 3\ 1\ 4\ 3]\end{array} \tag{9}$$

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.

So, searched by a computer, all loops can be get:

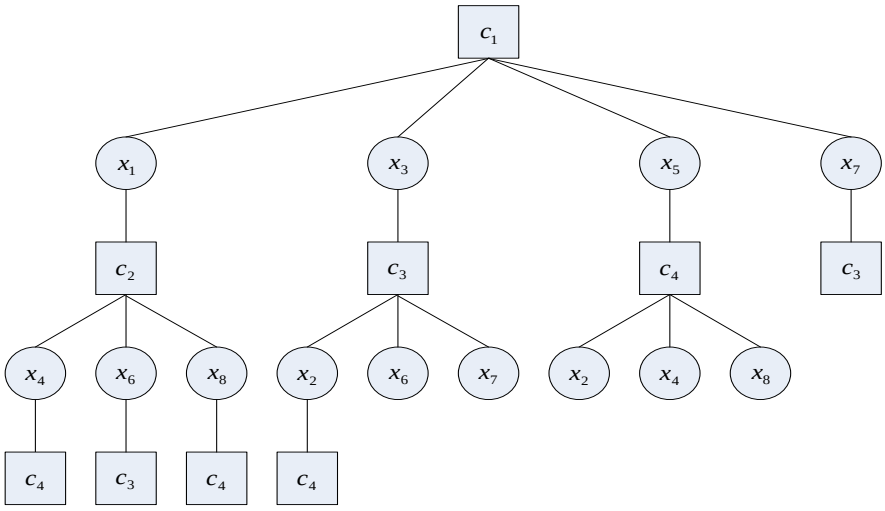

Figure 6. Tree graph of LDPC codes.

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]

Here, *H* is just a situation of single tree, with 15 cut-nodes, no leaf node and 16 loops. See **Table 1**. Further, Message-passing form for cut-node tree has following rule:

In a tree, Message-passing fellows a two-pass form, first sweeping upwards from leaves to a node designated as the root, and then downwards from the root to leaves.

In a graph with cycle, Cut-node tree graph can be get by cutting all loops, See **Figure 7**.

Cut node tree has all same characters with Tanner graph.

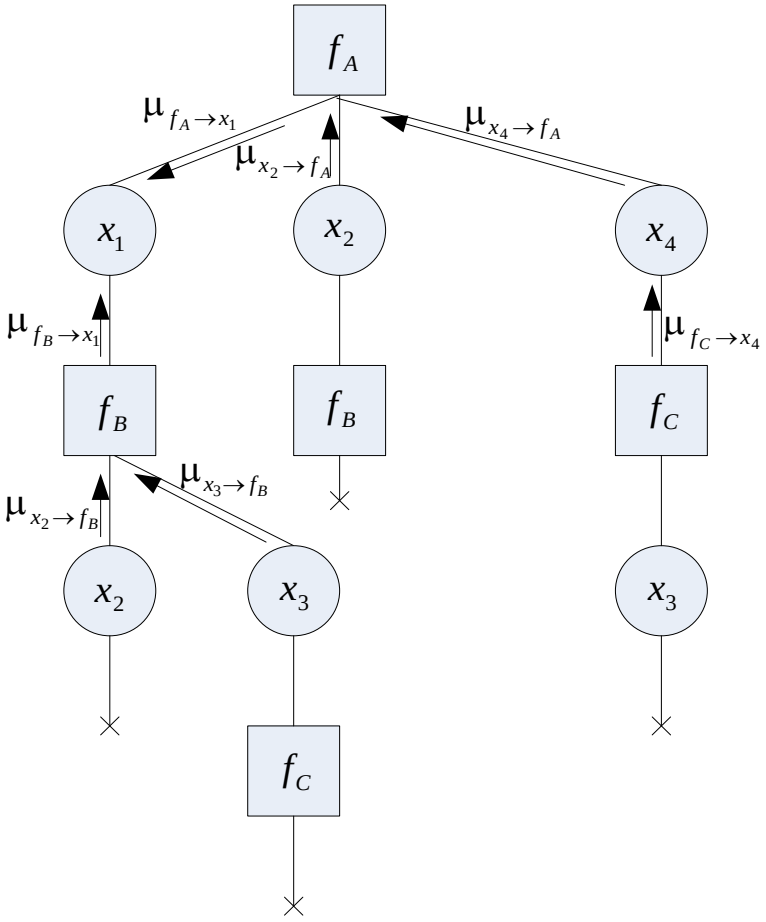

Figure 7. Message passing over cut-node tree graph.

Table 1. Features of loops about above example.

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

5. Conclusion

This paper provides a new method to describe graph of LDPC codes, it aims to solute loops of LDPC codes and message passing over cut node tree. For a large matrix *H*, it is difficult to solute loops, 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 computer simulation, cut-node tree can get right answer and give out method to calculate loops of LDPC codes and cut node tree graph, its results assist to further research this relationship.

References

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

[2] Mackay, D.J.C. (1999) Good Error-Correcting Codes Based on Very Sparse Matrices. *IEEE Transactions on Information Theory*, **45**, 399-431. http://dx.doi.org/10.1109/18.748992

[3] Tanner, R.M. (1981) A Recursive Approach to Low Complexity Codes. *IEEE Transactions on Information Theory*, **27**, 533-547. http://dx.doi.org/10.1109/TIT.1981.1056404

[4] Kschischang, R., Frey, B.J. and Loeliger, H.A. (2001) Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, **47**, 498-519. http://dx.doi.org/10.1109/18.910572

[5] Wiberg, N. (1996) Codes and Decoding on General Graphs, Linkoping Studies in Science and Technology. Ph.D. Dissertation, No. 440, Linköping University, Linköping.

[6] Reinhard, D. (1997) Graph Theory. Springer-Verlag, New York.

[7] Wainwright, M.J. (2007) Sparse Graph Codes for Side Information and Binning. *IEEE Signal Processing Magazine*, **47**, 47-57.

[8] Forney Jr., G.D. (2001) Codes on Graphs: Normal Realizations. *IEEE Transactions on Information Theory*, **47**, 520-548. http://dx.doi.org/10.1109/18.910573

[9] Loeliger, H.A. (2004) An Introduction to Factor Graph. *IEEE Signal Processing Magazine*, 28-41.

[10] Kschischang, F.R. (2003) Codes Defined on Graphs. *IEEE Communications Magazine*, 118-125.

[11] Vontobel, P.O. (2004) A Factor-Graph Approach to the Context-Tree Weighting Method. *Proceedings of the Data Compression Conference* (*DCC*'04), 571.

[12] Wainwright, M., Jaakkola, T. and Willsky, A. (2002) Tree-Based Reparameterization Analysis of Belief Propagation and Related Algorithms for Approximate Inference on Graphs with Cycles. *ISIT*2002, Lausanne, 30 June-5 July 2002.