PAPER

A Cycle Search Algorithm Based on a Message-Passing for the Design of Good LDPC Codes

Sang Hyun LEE[†], *Nonmember*, **Kwang Soon KIM**[††a)], *Member*, **Yun Hee KIM**[†††], *and* **Jae Young AHN**[†], *Nonmembers*

SUMMARY A cycle search algorithm based on a message-passing in a Tanner graph is proposed for designing good LDPC codes. By applying the message-passing algorithm with a message alphabet composed of only two messages to a cycle search, we can perform a cycle search with less computational complexity than tree-based search algorithms. Also, the proposed algorithm can be easily implemented by using an existing message-passing decoder and can easily adopt different kind of criteria for an LDPC code design with a slight modification in the node update equations.

key words: *LDPC codes, message-passing decoding, cycle search*

1. Introduction

A low-density parity-check (LDPC) code is a linear code defined by a sparse parity check matrix which is associated with a corresponding Tanner graph containing two types of node denoted as bit nodes and check nodes. In [1], it was shown that the performance of a LDPC code, associated with a random Tanner graph under the cycle-free assumption, was very close to the Shannon limit. However, a relatively short-length code, whose associated Tanner graph is not cycle-free, suffers from significant performance degradation [2]. Therefore, constructing a Tanner graph for a good code requires a careful elimination of short-length cycles to enhance its decoding capability, especially for that with a relatively short length. Although a cycle search is a well-known problem, existing algorithms are basically tree-based searches whose search trees often grow exponentially. Thus, it would be computationally impractical to use a tree-based search algorithm for a large search depth. There are several girth conditioning methods, such as the Mao's algorithm in [3] and the PEG algorithm in [4], resorting to the tree-based search. The Mao's algorithm uses the breadth-first search over the tree to detect the nodes visited by at least two search paths. While the PEG algorithm has the different edge placement strategy that a new edge is progressively added to the graph to maximize girth in a best-effort way by checking cycles, both algorithms basically have the same computational complexity. Recently, a trellis-based search algorithm is proposed by constructing a trellis with repeated cascades of a Tanner graph and its mirror image [5]. However, this algorithm does not improve the number of required searches and the amount of required storage compared to the tree-based algorithms [3], [4] because the constructed trellis and corresponding path-finding algorithm used in [5] is actually equivalent to the tree-based algorithms. In this paper, we present a novel search algorithm to find the existence and exact length of cycles by using the message-passing algorithm, which requires only simple binary operations. Thus, we can easily obtain a truncated version of the exact cycle distribution of a Tanner graph. Also, the proposed algorithm can be directly implemented by using an existing message-passing decoder with a slight modification. Thus, we may use hardware-implemented message passing decoder to perform extensive search on codes with large length.

Manuscript received September 7, 2004.
Manuscript revised January 17, 2005.
Final manuscript received March 14, 2005.

[†]The authors are with Mobile Communications Lab., Electronics and Telecommunications Research Institute, Daejeon, Korea.
[††]The author is with the Center for Information Technology of Yonsei University (CITY), the Department of Electrical and Electronic Engineering, Yonsei University, Seoul, Korea.
[†††]The author is with the Department of Electronic Engineering, Kyung Hee University, Gyeonggi-do, Korea.
a) E-mail: ks.kim@yonsei.ac.kr

DOI: 10.1093/ietfec/e88–a.6.1599

2. Proposed Algorithm

The basic idea of the proposed algorithm is to run a message-passing algorithm on a Tanner graph with only '1' and '0' as message alphabets. It indicates the length of the shortest path in the Tanner graph that the smallest number of iterations for a message sent by some node to propagate back to the node itself. Then, we can examine the existence of a cycle as follows: i) send a particular message to an edge of a node, ii) check whether that message comes back to the node, and iii) find the length of a cycle by counting the number of iterations until the message comes back. Unlike an ordinary message-passing LDPC decoder, the proposed algorithm requires only a single node update equation defined as

$$y_{ij} = \left(\bigoplus_{j' \in E_i} x_{ij'} \right) \bigotimes x_{ij}, \tag{1}$$

where x_{ij} is the input message of the jth edge entering the ith node, y_{ij} is the output message of the jth edge leaving the ith node, and the operations $\oplus$ and $\otimes$ denote the logical OR and XOR operations, respectively. Also, E_i denotes the set of edges connected to the ith node. For each node, if all input messages are '0,' then all output messages are '0.' If the input messages through some edges are '1,' then the output messages through those edges are '0,' while the output

messages through other edges are '1.'

Now, consider an LDPC code design method by examining the lengths of short cycles on an edge-by-edge basis. In this scheme, edges are sequentially added to construct a Tanner graph. Once a new edge is added to the graph being constructed, a search for cycles caused by the edge is performed. Note that the bit node connected to the new edge is the starting bit node. Initially, the output message of the new edge is set to '1' while the output messages of other edges of the starting bit node and all output messages of other bit nodes are set to '0.'

Definition 1: $M(0)$ is the set containing the starting bit node only. $M(2l-1)$ and $M(2l)$, $l = 1, 2, \ldots$, are the sets of check nodes and bit nodes which are visited by at least one nonzero message at the lth iteration, respectively.

Note that the cardinality of $M(1)$ is 1 because the initial message is sent only through the edge under examination. For easy understanding of the proposed algorithm, Fig. 1 illustrates first several steps of the node processing. The left and right columns mean the bit node processing and the check node processing, respectively. Here, circles and rectangles represent bit nodes and check nodes, respectively. All the output messages of the nodes in the entire Tanner graph that is not shown in Fig. 1 are zero. At the initialization step ((a) in Fig. 1), the output message of the starting bit node $M(0)$ to the neighboring check node $M(1)$ is set to '1.' At the check node processing of the first iteration ((b)), the output messages of $M(1)$ to its neighboring bit nodes except $M(0)$ are '1.' Thus, $M(2)$ is comprised of those bit nodes receiving '1' and the message from $M(1)$ to $M(0)$ is '0.' At the bit node processing of the first iteration ((c)), $M(3)$ is comprised of check nodes receiving '1' and all the check nodes in $M(1)$ receive '0.' Since the output message of $M(2)$ to $M(1)$ and the output message of $M(0)$ are '0,' the output messages of $M(1)$ to all neighboring bit nodes are '0' at the check node processing of the next iteration ((d)). Also, check nodes in $M(3)$ send nonzero messages to new bit node set $M(4)$ and zeros to $M(2)$. Thus, all input messages of $M(2)$ from $M(3)$ and $M(1)$ are '0' and then its output messages are '0' as well. From this observation, one can see that the output messages of $M(1)$ ($M(0) \cup M(2)$) to $M(0)$ ($M(1)$) are fixed to '0' afterward. Then, we can draw the following assertions from the above observations.

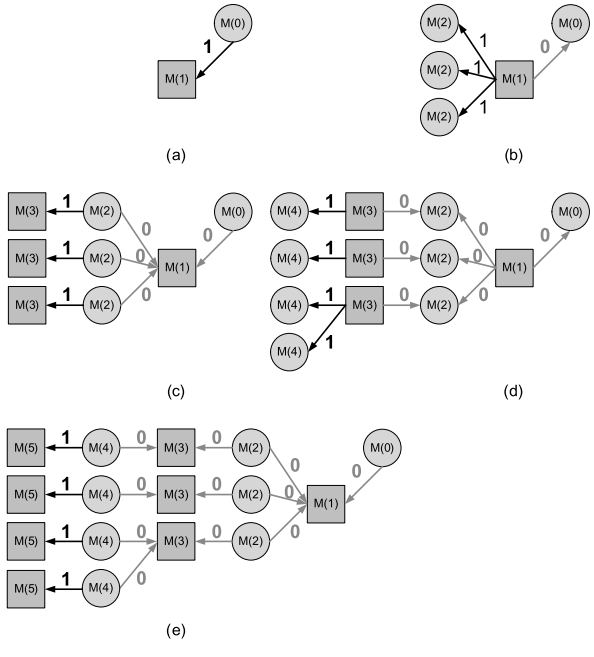


Fig. 1 An example for the proof of Lemma 1.

Lemma 1: If a cycle has not been detected up to the Lth iteration, $M(l)$ for $l = 0, \ldots, 2L$ are disjoint sets. Furthermore, only $M(2L-1)$ ($M(2L)$) has nonzero output messages at check (bit) node processing.

Proof) Here, we will use the mathematical induction. Since we have two different classes (check and bit) of nodes, it is sufficient to prove $M(2l-1)$ ($M(2l)$) for $l = 1, \ldots, L$ are disjoint sets. For induction, suppose that, at the check (bit) node processing of the lth iteration and after, all output messages of $\bigcup_{i=1}^{l} M(2i-1)$ ($\bigcup_{i=0}^{l} M(2i)$) to $\bigcup_{i=0}^{l-1} M(2i)$ ($\bigcup_{i=1}^{l} M(2i-1)$) are '0.' For the check node set $M(2l+1)$, its constituent nodes send '0's and '1's, which are determined by (1), to $M(2l)$ and new bit node set $M(2l+2)$, respectively, at the check node processing of the $(l+1)$th iteration. Also, from the assumption, other check node sets send '0's to their neighboring bit node sets. Since $\bigcup_{i=0}^{l} M(2i)$ received all '0' messages at the check node processing of the $(l+1)$th iteration, their all output messages are '0' at the bit node processing of the $(l+1)$th iteration. Furthermore, $M(2l+2)$ send '0's back to $M(2l+1)$. Since all output messages of $\bigcup_{i=1}^{l+1} M(2i-1)$ ($\bigcup_{i=0}^{l+1} M(2i)$) to $\bigcup_{i=0}^{l} M(2i)$ ($\bigcup_{i=1}^{l+1} M(2i-1)$) are '0,' all messages among them will be '0' afterward. From the mathematical induction, it is proved that $M(2l_1-1)$ and $M(2l_1)$ $(0 \le l_1 \le L)$ receive null messages at the l_2th iteration for any ordered pair $(l_1, l_2)(0 \le l_1 < l_2 \le L)$. Thus, by Definition 1, $M(2l-1)$ ($M(2l)$) for $l = 1, \ldots, L$ are disjoint. Also, the output messages of $M(2L-1)$ ($M(2L)$) to $M(2L)$ ($M(2L+1)$) are '1' while the other output messages of $M(2L-1)(M(2L))$ and the output messages of $\bigcup_{l=1}^{L-1} M(2l-1)$ ($\bigcup_{l=0}^{L-1} M(2l)$) are all '0' at the check (bit) node processing of the Lth iteration. Therefore, only $M(2L-1)$ ($M(2L)$) has nonzero output messages. □

Lemma 2: A nonzero message traverses through an edge at most once until the first cycle is detected.

Proof) Suppose that $M(l), l = 1, 2, \ldots$, receives nonzero messages from $M(l-1)$ for the first time at the lth iteration. Then, from Lemma 1, all messages between $M(l-1)$ and $M(l)$ are zero messages at later iterations. Thus, each edge connecting one node in $M(l)$ to the other node in $M(l-1)$ delivers a nonzero message only once. Since the edges which have not been passed by a nonzero message deliver only zero messages in both directions, all edges are traversed by a nonzero message at most once. □

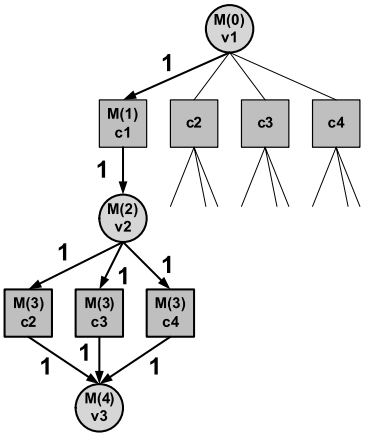


Fig. 2 An example of the case where a message vanishes.

Theorem 1: A nonzero message returns to $M(0)$ at the Lth iteration if and only if the length of the shortest cycle containing $M(0)$ is $2L$.

Proof) Since $M(l), l = 0, 1, 2, \ldots$ are mutually exclusive sets, the number of unvisited edges and bit nodes decreases as nonzero messages propagate in the graph. Traversing the remaining nodes in this way indefinitely, a nonzero message eventually visits the starting node again through unvisited edges unless it vanishes. Nonzero messages vanish only when there exists a node set such that its all constituent nodes receive only nonzero input messages. In Fig. 2, an example of a vanishing message is depicted. The figure shows the trajectory of nonzero messages over a graph for a few iterations. The bit node v_3 in $M(4)$ accepts nonzero messages through its all (three) edges at the check node processing of the second iteration. Since all input messages are '1,' the output messages determined by (1) are all '0' and all output messages of v_3 are '0' thereafter. Thus, this means that the message vanished at v_3. Note that any combination of paths which delivered nonzero messages to the nodes where nonzero message vanish cannot form a cycle containing $M(0)$. Therefore, this case does not affect the search for cycles associated with $M(0)$. Suppose that there exists a node $m(2L)$ such that $m(2L) = M(2L) \cap M(0)$. Then, for each node set $M(l), l = 0, 1, \ldots, 2L$, there exist edges that deliver nonzero messages from its neighboring node set $M(l-1)$. Therefore, by tracking back the edges transferring nonzero messages from $m(2l) \in M(2L)$ to $M(0)$, one can find an ordered set of $2L$ edges forming a cycle with length $2L$. Next, suppose that a bit node $M(0)$ has cycles and the length of the shortest one is $2L$. Let $m(l)$, $l = 0, \ldots, 2L$, denote the nodes constituting the length-$2L$ cycle. Also, let $e(l)$, $l = 1, \ldots, 2L$, denote the edges connecting the node $m(l-1)$ with the node $m(l)$. Then, it is sufficient to conclude the proof that a nonzero message propagates from $m(l)$ to $m(l+1)$ node-by-node at each processing. Suppose that the nonzero message sent from $m(0)$ is passed to the node $m(l^*)$ but it is not passed to the next node $m(l^*+1)$. It means that two nonzero input messages enter $m(l^*)$ through $e(l^*)$ and $e(l^*+1)$ simultaneously at the l^*th processing, which implies that a nonzero message has already arrived at $m(l^*+1)$ at the l^*th processing. Thus, there should exist a path from $m(0)$ to $m(l^*+1)$ of length less than l^*, which implies a cycle containing $M(0)$ with length less than $2L$. However, it contradicts the assumption. □

Theorem 2: The number of length-$2L$ cycles associated with $M(0)$ is lower-bounded by the number of nonzero messages returning to $M(0)$ at the Lth iteration.

Proof) Suppose that nonzero messages returned to the starting bit node at the Lth iteration. The number of returned nonzero messages corresponds to the number of edges connecting $M(2L-1)$ to $M(0)$. From the proof of Theorem 1, for each returned nonzero message, one can find a set of $2L$ edges, each connecting $M(l)$ to $M(l-1), l = 1, \ldots, 2L$, forming a path returning to $M(0)$. Thus, there exist at least as many cycles as nonzero messages arriving at the starting bit node at the Lth iteration. If, in each returning path, there exist a node $m(l^*) \in M(l^*)$ that has received more than one simultaneous nonzero input messages, one can find multiple returning paths from $M(0)$ to $m(2L) \in M(2L) \cap M(0)$. Each of multiple returning paths is formed by the first part of the path where each nonzero message propagated from $M(0)$ to $m(l^*)$ and the last part of common path from $m(l^*)$ to $m(2L)$. Thus, multiple cycles are identified for a nonzero message returning to $m(2L)$ at the Lth iteration. Therefore, actual number of cycles can be more than the number of cycles evaluated by the algorithm. □

The next theorem presents a revised algorithm to determine the exact number of cycles.

Theorem 3: Redefine the operations $\oplus$ and $\otimes$ in (1) as

$$\begin{cases} x \oplus y = x + y & x, y \in \mathbb{Z}^+ \cup \{0\} \\ x \otimes 0 = 0 \otimes x = x & x \in \mathbb{Z}^+ \cup \{0\} \\ x \otimes y = 0 & x, y \in \mathbb{Z}^+ \end{cases} \tag{2}$$

where $+$ is the arithmetic addition and $\mathbb{Z}^+$ is the set of all positive integers. Then, the value of a message represents the number of multiple cycles associated with the current searching path and the exact number of length-$2L$ cycles containing $M(0)$ is the sum of all nonzero input messages returning $M(0)$ at the Lth iteration.

Proof) Initially, all nonzero messages are '1.' If more than one nonzero messages arrive at a node simultaneously, the nonzero output messages of such node are the number of nonzero input messages. Note that each nonzero message is associated with one of the multiple paths which may form a distinct cycle returning to the starting bit node. Thus, the number of nonzero simultaneous input messages indicates the number of such cycles associated with that node. As the iteration proceeds, the number of associated cycles is accumulated and propagated to a neighboring node set. Once a nonzero message returns to the starting node, its value indicates the overall number of nonzero input messages through such returning paths. Thus, the sum of returned nonzero messages is the number of the cycles containing $M(0)$. □

Input : l_0, N_{iter}, E

1: if (the new edge is not the first edge of two nodes which it is connected to) {
2: set k=0, Flag_cycle=0, Length_cycle = 0, $(i_p, j_p) = \mathrm{IND}(l_p)$ $l_p \in E$
3: set $n(l_0)=1$, $n(l)=0$ $l \in E \setminus \{l_0\}$
4: while ($k < N_{iter}$) {
5: for ($l=0; l<|E|; l++$) $m(l) = \left(\bigoplus_{l' \in E(i_l)} n(l')\right) \otimes n(l)$
6: if ($\sum_{l \in E} m(l) == 0$) break
7: for ($l=0; l<|E|; l++$) $n(l) = \left(\bigoplus_{l' \in E(j_l)} m(l')\right) \otimes m(l)$
8: if ($\sum_{l \in E} n(l) == 0$) break
9: if ($\sum_{l \in E(j_0)} n(l) > 0$) Flag_cycle = 1, Length_cycle = $2*k$ break
10: $k = k+1$
11: }
12:}

Output : Flag_cycle, Length_cycle

Fig. 3 The outline of the proposed cycle detection method.

In Fig. 3, the outline of the proposed algorithm is shown. Here, l_0 is the edge index of the newly added edge, N_{iter} is the maximum number of iteration, and E is the set of all edge indices in the graph. The indices of the check node and the bit node connected to the pth edge are denoted as i_p and j_p, respectively. The function IND maps each edge index to its check and bit node indices. Also, $E(i_l)$ and $E(j_l)$ are the set of all edges connected to the check node i_l and the bit node j_l, respectively. Initially, the bit node message $n(l_0)$ of the newly added edge is set to '1' and other messages are set to '0.' During the message-passing processing, the check node message $m(l)$ and the bit node message $n(l)$ of the lth edge are calculated by using (1). If, at each iteration, the number of nonzero input message to the bit node connected to the new edge is nonzero, then a cycle is detected. Note that cycles of length $2L$ are found with L iterations. Lines 6 and 8 check whether all messages on the graph vanish or not. If all messages vanish, no nonzero message can return to the starting bit node. This indicates that there is no cycle containing the starting node. Therefore, the search finishes and reports that there is no cycle. In addition, for detecting larger-length cycles containing the starting bit node, one can set the output messages of the starting bit node at the next iteration to zero and continue the iteration until another nonzero message arrived at the starting bit node.

In Fig. 4, a search example for a length-six cycle associated with the leftmost edge in a (3,2) regular Tanner graph is shown. Here, each row means an iteration. The left (right) column is bit (check) node processing at each iteration. Circles and squares in the graph denote the bit nodes and the check nodes, respectively. Dark gray circles, squares and edges indicate that they have been already included in the searching path. Only nonzero output messages at the nodes are shown. In the initialization (Fig. 4(a)), the starting bit node, v_1, is the only element of $M(0)$. The leftmost edge of $M(0)$ delivers nonzero message to the neighboring check node c_1, which is the only element of $M(1)$. At the check processing (Fig. 4(b)), $M(1)$ sends '1' through the connected edges whose previous input messages to $M(1)$ were '0.' Since the degree of c_1 is 3, $M(2)$ is a two-element set given by $\{v_3, v_5\}$. Likewise, $M(3)$ and $M(4)$ are successively defined as shown in Figs. 4(c)–4(f). The light gray nodes at the third row represent that the multiple searching paths have simultaneously visited. Multiple paths passing through such nodes form another small cycle. Note that the light gray bit node v_6 accepts nonzero inputs through all connected edges. Then, the output messages of v_6 at the next processing are all zero. This corresponds to the case where nonzero messages vanish. Although the vanished messages give rise to smaller cycles than the cycle under examination, such cycles do not contain the starting bit node $M(0)$. Thus, they do not double up the number of cycles containing $M(0)$. It is easily found that two smaller cycles are attached to the cycle of length 6. Thus, there exist two cycles with length 6 in this case. Note that, by using (2) in Theorem 3, the exact number of the cycles is obtained while only one is counted by using (1).

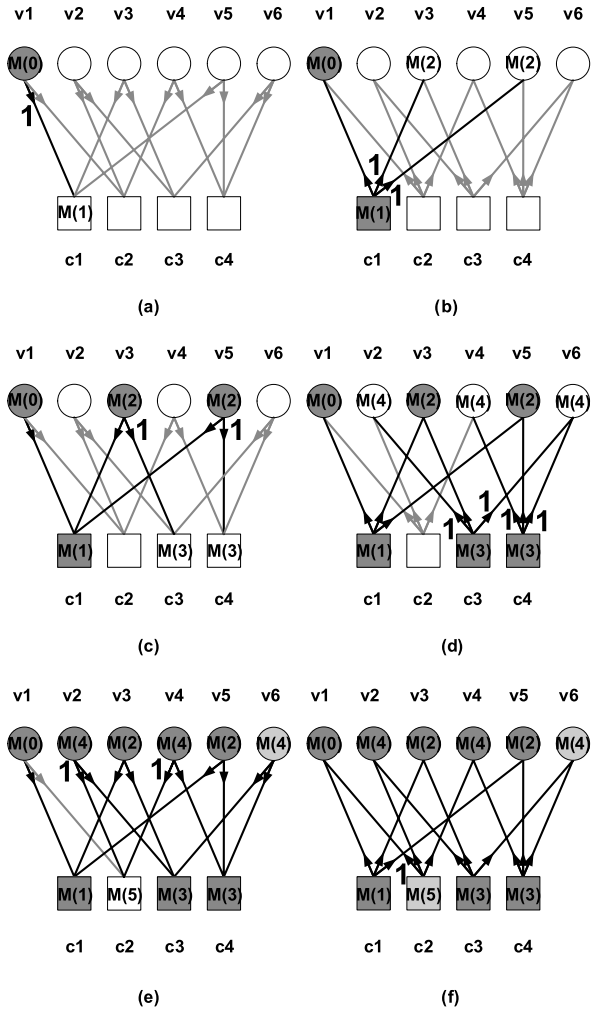


Fig. 4 An example of search for cycle with length 6.

While the above outline of the algorithm is described

in an edge-by-edge manner, it is straightforward to apply the proposed algorithm to a node-by-node LDPC code design. To do so, we employ a message vector of the length equal to the number of edges connected to the bit node under examination. Initially, only the *i*th element of the message vector for the *i*th edge connected to the node under examination is set to '1' for all message vectors sent from the node, while other message vectors sent from other nodes are set to all zero vectors. The operations between two message vectors are defined as the element-wise calculation of (1). Note that a cycle may be counted more than once in this case. Although obtained cycle distribution by using the proposed algorithm may be slightly different from the actual one in this case, we can still use the proposed algorithm to design a good LDPC code.

3. Complexity and Example

We will now roughly analyze the computational complexity of the proposed algorithm. Basically, the complexity of the proposed algorithm is the same as that of the LDPC message-passing decoding. Since a node processing for a single edge requires $O(d_{max})$ operations and the computation result is shared among edges at a node, the entire node processing at a single node requires $O(d_{max})$ operations, where d_{max} is the maximum degree of nodes. Thus, processing for the entire bit nodes requires roughly $O(Ld_{max}m^2)$ operations, where L is the number of iterations and m is the number of bit nodes. Thus, the proposed algorithm has polynomial-time complexity and is quite favorable in the context of complexity compared to the tree-based search algorithms [3], [4].

Table 1 compares the proposed algorithm to the algorithm in [3] with the maximum amount of required memory and the elapsed time for searching all cycles of length up to 12 in a few Tanner graphs. The Mao's algorithm [3] and the PEG algorithm [4] are tree-based girth conditioning algorithms with different edge placement strategies and similar complexity. Also, the trellis-based algorithm [5] forms the list of paths that may form cycles, which is the identical information to that used in [3], [4]. Thus, it requires the same amount of memory as the tree-based algorithms do. Actually, if the branches, which have not been passed by search paths, are removed from the trellis, the trellis becomes identical to what used in the tree-based search. Thus, the trellis-based algorithm [5] is computationally equivalent to the tree-based algorithms. Therefore, it is enough to compare with the algorithm in [3]. Since it is empirically seen that relatively 'dense' short-length cycles in a graph degrade the performance of the message-passing decoding and reduce the minimum distance of the code (also shown in Fig. 5), it would be beneficial for better performance to investigate all cycles up to a given length. Here, we straightforwardly modified the algorithm in [3] to detect multiple-length cycles. From the results, it is shown that the elapsed time increases as the node degree or the code length grows. Also, one can see that, while the same number of cycles is detected, the proposed algorithm takes less searching time and needs smaller amount of memory. Although both algorithms have similar performance for the case of searching only for the shortest cycle, the proposed algorithm improves the searching time significantly in the case of searching for all cycles up to a pre-defined length included in a Tanner graph. This comes from the fact that a tree-based search is more likely to grow exponentially while the proposed algorithm still remains polynomial-time in computational complexity. Figure 5 depicts the performance of the codes designed by both algorithms over AWGN channel by using the message-passing decoder with 50 iterations. The result shows that it is worthwhile to investigate more than girth to design an LDPC code and that the proposed algorithm gives identical performance with less code searching time.

Table 1 Search results of several example graphs.

Aver. bit node degree	3	3	6
Code size (n,k)	(100, 50)	(1000, 500)	(100, 50)
# of detected cycles	1740	12490	3420
Number of required memories			
Proposed	156	1541	468
Algorithm in [3]	172	2953	559
Elapsed time			
Proposed	1.266 s	208 s	1.969 s
Algorithm in [3]	7.375 s	2740 s	14.340 s

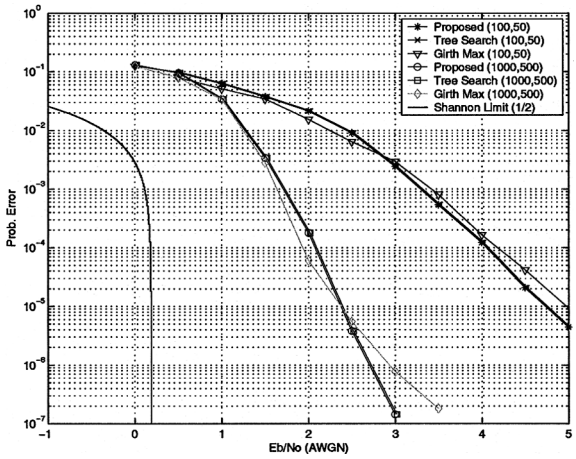


Fig. 5 Simulation results of codes designed by the proposed algorithms.

4. Conclusion

In this paper, a novel cycle search scheme using a message-passing algorithm was proposed for designing a good LDPC code. The proposed scheme can detect the existence of cycles and compute the length of detected cycle by propagating a particular message through the Tanner graph and monitoring messages coming back to the node. The result proves that the proposed algorithm is more efficient than tree-based search because of simpler search based on message-passing with less memory requirement. Also, the proposed algorithm can be implemented by using an existing message-passing decoder without additional cost or by using matrix-

algebra operations without any consideration of making tree or trellis in a computer program. Also, the proposed algorithm can easily adopt different kind of criteria for an LDPC code design, such as ACE in [6], with a slight modification in the node update equations.

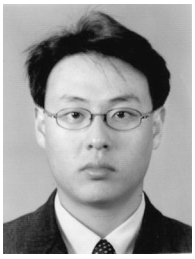
Acknowledgement

The authors are very grateful to the anonymous reviewers for their helpful and constructive comments and suggestions.

References

[1] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," IEEE Trans. Inf. Theory, vol.47, no.2, pp.638–656, Feb. 2001.
[2] D.J.C. MacKay, "Good error correcting codes based on very sparse matrices," IEEE Trans. Inf. Theory, vol.45, no.2, pp.399–431, March 1999.
[3] Y. Mao and A.H. Banihashemi, "A heuristic search for good low-density parity-check codes at short block lengths," Proc. IEEE Int. Conf. Comm., vol.1, pp.11–14, Helsinki, Finland, June 2001.
[4] X. Hu, E. Eleftheriou, and D. Arnold, "Progressive edge-growth Tanner graphs," Proc. 2001 IEEE GlobeCom, vol.2, pp.995–1001, San Antonio, TX, Nov. 2001.
[5] L. Lan, Y.Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "A trellis-based method for removing cycles from bipartite graphs and construction of low density parity check codes," IEEE Commun. Lett., vol.8, pp.443–445, July 2004.
[6] T. Tian, C. Jones, J.D. Villasenor, and R.D. Wesel, "Construction of irregular LDPC codes with low error floors," Proc. IEEE Int. Conf. Comm., vol.5, pp.3125–3129, Anchorage, AK, May 2003.

Sang Hyun Lee received the B.S. and M.S. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1999 and 2001. Since 2001, he has been with Electronics and Telecommunications Research Institute (ETRI) in Daejeon, Korea. His current research interest include channel codes, information theory, wavelet analysis, digital filter design and digital signal processing algorithms for communication systems.



Kwang Soon Kim received the B.S. (*summa cum laude*), M.S., and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1994, 1996, and 1999, respectively. He was a teaching and research assistant at the Department of Electrical Engineering, KAIST from March 1994 to February 1999. From March 1999 to March 2000, he was with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA, U.S.A., as a Postdoctoral Researcher. From April 2000 to February 2004, he was with the Mobile Telecommunication Research Laboratory, Electronics and Telecommunication Research Institute (ETRI), Daejeon, Korea, as a Sennior Member of Rearch Staff. Since March 2004, he has been with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul, Korea, as an Assistant Professor. He was a recipient of the Postdoctoral Fellowship from Korea Science and Engineering Foundation (KOSEF) in 1999. He received the Best Researcher Award from ETRI in 2002. His research interests include statistical signal processing, communication signal processing, adaptive modulation and coding, multiple element antenna technique, channel coding and iterative decoding, cross-layer optimization, and wireless CDMA/OFDM systems.

Yun Hee Kim received the B.S. (*magna cum laude*), M.S., and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1995, 1997, and 2000, respectively. After visiting the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA, U.S.A., during the spring of 2000, she joined the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea, where she worked for the development of the W-CDMA base station MODEM and the air interface for 4G mobile radio access. Since 2004, she has been with School of Electronics and Information, Kyung Hee University, Gyeonggi-do, Korea, where she is now an assistant professor. Her research interests include coding and modulation, signal processing for digital communications, and design and analysis of mobile communication systems such as OFDM and CDMA systems.

Jae Young Ahn received the B.S., M.S., and Ph.D. degrees in electrical engineering from Yonsei University, Seoul, Korea, in 1983, and 1985, and 1989, respectively. Since 1989, he has been with Electronics and Telecommunications Research Institute (ETRI) in Daejeon, Korea. Between 1989 to 2002, he involved in the research and development of satellite communications systems and wireless LAN technologies. From 2003, he has been leading the research of radio transmission technology for next generation mobile systems. His current research interest include radio transmission technology for future mobile and wireless communications.