Design of LDPC Codes Without Trapping Set TS(5,3) with minimizing TS(6,4)

Madiagne Diouf, *Student Member, IEEE*, David Declercq, *Senior Member, IEEE*, Samuel Ouya, *Student Member, IEEE*, and Bane Vasic, *Fellow, IEEE*

Abstract

Design of binary LDPC codes with very low error floors is still a significant problem. Progressive Edge-Growth (P.E.G) original is a well known algorithm for constructing bipartite graphs with good girth properties. In this paper, we propose a characterization of the trapping set TS(5,3) and a cost function in P.E.G that allows to build regular girth 8 LDPC codes free of TS(5,3). We illustrate the interest of eliminating the trapping set TS(5,3) and minimizing TS(6,4) with simulation results.

Index Terms

Error floor, Trapping sets, Progressive Edge-Growth (PEG), Low Density Parity Check (LDPC) codes, Tanner graphs.

I. Introduction

The design of binary LDPC codes with very low error floors is still a significant problem [1], [2], e.g for point to point satellite communication, magnetic or optical recording [3]. On noisy channels (e.g BSC or Bi-AWGN), the error floor of iterative decoder of LDPC codes is due to the presence of trapping sets in the Tanner graph [4], [5]. Therefore, the elimination of the most harmful trapping sets will optimize the code design procedure.

The original PEG algorithm [6] is a well known algorithm for constructing bipartite graphs, it builds up a Tanner graph for LDPC code on an edge-by-edge basis and by maximizing the local girth. There are some improvements in the PEG algorithm that allow avoiding the small stopping set [7], or minimize the number of cycles [8]–[10]. So far no attempt has been made to detect and avoid trapping sets using a PEG-like algorithm.

In this paper we propose a *characterization* of the trapping set TS(5,3) and TS(6,4) in the subgraph called *tree* and a *cost function* in the PEG that allows to build regular girth 8 LDPC codes free of TS(5,3) and minimize

The work of M. Diouf, S. Ouya and D. Declercq are supported by the Seventh Framework Program of the European Union, under Grant Agreement number 309129 (i-RISC project) and B. Vasic work is supported by the NSF Grants CCF-0963726 and NSF CCF-1314147.

Madiagne Diouf and Samuel Ouya are with LIRT/ESP/UCAD, Senegal (email: samuel.ouya@gmail.com), D. Declercq is with ETIS Laboratory, ENSEA/Université de Cergy-Pontoise/CNRS, Cergy-Pontoise, France(email: {madiagne.diouf,declercq}@ensea.fr).

B. Vasic is with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721 USA (e-mail: vasic@ece.arizona.edu).

TS(6,4). As a result our algorithm is able to construct an QC-LDPC code without $TS(5,3)$ with the cycles of length 8.

The remainder of the paper is organized as follows: In Section II we will introduce preliminaries; then the characterization of trapping sets TS(5,3) and TS(6,4) is derived in Section III; furthermore in Section IV we will describe the theorem for detect the trapping set in the tree, In Section V we describe our design algorithm using RandPEG and the build of the QC-LDPC code with increase the number of block that form the cycle of length-8. And finally, we will illustrate the use of our algorithm with simulation results.

II. Preliminaries and Notation

Let G denote the Tanner graph of an (N, K) binary LDPC code $\mathcal{C}$ of rate $R = K/N$, which consists of the set of N variable nodes V and the set of M check nodes C. We will use roman alphabet to denote variable nodes and Greek letters for check nodes. Two nodes are neighbors if there is an edge between them. The degree of a node in G is the number of its neighbors in G. A code $\mathcal{C}$ represented by the graph G is said to be have a regular column-weight d_v if all variable nodes in V have the same degree d_v. Equivalently, such code is said to be d_v-variable-node regular or just d_v-variable regular. The set of neighbors of a node u is denoted as $\mathcal{N}_u$ and $\mathcal{N}_U$ denotes the set of neighbors of all $u \in U$. A path in G is a finite sequence of distinct vertices (variables or checks) $u_0, \ldots, u_k$ such that u_{i-1} and u_i are neighbors for $1 \leq i \leq k$. Two paths are distinct if they differ in at least one node. A k-cycle or cycle of length k in G is a path $u_0, \ldots, u_k$ in G with $u_0 = u_k$. Clearly in a bipartite graph G, k must be even. The girth g of G is the length of shortest cycle in G. $\mathcal{C}_{d_v,g}$ denotes an ensemble of d_v-variable regular codes with girth g.

We begin by providing the definition of a trapping set as originally defined by Richardson in [4].

Definition 1: Given a decoder input $\mathbf{y}$, a *trapping set* (TS) for an iterative decoder denoted by $\mathbf{T}(\mathbf{y})$ is a non-empty set of variable nodes in G that are not corrected at the end of a given number of iterations.

A common notation used to denote a TS is (a, b), where $a = |\mathbf{T}|$, and b is the number of odd-degree check nodes in the subgraph induced by the set of variable nodes $\mathbf{T}$. Let $\mathcal{T}(a, b)$ denote the bipartite graph associated with an (a, b) TS, where a is the number of variable nodes and b is the number of odd-degree check nodes present in the graph. A graph G contains an (a, b) TS of type $\mathcal{T}$ if there exists a subset of variable nodes $\mathbf{T}$ in G whose induced subgraph is isomorphic to $\mathcal{T}(a, b)$. A TS is said to be *elementary* if $\mathcal{T}$ contains only degree-one and/or degree-two check nodes. Otherwise it is non-elementary. Throughout this paper, we restrict our focus to elementary trapping sets, since they are known to be dominant in the error floor [4], [5]. Henceforth, for convenience, whenever we refer to a TS, we will implicitly refer to its topological structure $\mathcal{T}$. Clearly the (a, b) notation is not sufficient to describe a topology of a trapping set as there can be many nonisomorphic graphs with a variable nodes and b odd-degree check nodes. However listing all nonisomorphic graphs is intractable when a is large, thus in this paper we use the notation which enumerates the distinct cycles in a subgraph. Let $\mathcal{T}$ contains g_{2k} $(2k)$-cycles, where $k \geq 2$, then the trapping set associated with $\mathcal{T}$ is said to be of type $TS(a, b, \prod_{k\geq 2}(2k)^{g_{2k}})$. We also use the

notation $\mathcal{T}(a,b,\prod_{k\geq 2}(2k)^{g_{2k}})$ to say that a graph G contains an $(a,b,\prod_{k\geq 2}(2k)^{g_{2k}})$ TS of type $\mathcal{T}$ if there exists a subset of variable nodes $\mathbf{T}$ in G whose induced subgraph is isomorphic to $\mathcal{T}(a,b,\prod_{k\geq 2}(2k)^{g_{2k}})$.

An best method for constructing Tanner graphs having a large girth is Progressive-Edge-Growth (PEG) algorithm [6], [8], [10]. This method is based on edge-by-edge manner. For a given variable node v, a new edge is created by a subgraph spreading from v within *depth-k*. k depends of the target girth [9], and the subgraph is called *depth-k tree*. The graph with new edge is updated for the determination of the next edge. We denote the set of all check nodes neighbors in *depth-k tree* by $\mathcal{M}_v^k$, and $\overline{\mathcal{M}}_v^k$ the complementary set.

Definition 2: Let $T_v^k(G)$ be a depth-k tree for the variable node v of a Tanner graph G. The variable nodes of $T_v^k(G)$ are copies of the variable nodes of G , and the check nodes of $T_v^k(G)$ are copies of the check nodes of G. When there is no ambiguity we will shorten the notation and use T_v^k when the graph is specified, T_v when the depth is specified. A tree of H, a subgraph of a Tanner graph G is denoted by $T_v^k(H)$. We draw a tree of depth k as a tree with $2k+1$ levels, labeled from $[0,2k]$, where the 0^{th} level consists only of the root node, even-numbered levels contains only variable nodes, and odd-numbered levels contains only check nodes. From the definition, different copies of a variable node can be at different levels in T_v. Let T_v contain m copies of a node u, denoted as $u_i^{l_i}$, $1\leq i\leq m$, where l_i is the level of the i-th copy.

Locally, the tree looks like the Tanner graph G. Consider now a depth-k tree, T_v, of the variable node v.

Definition 3: A vertex $w\in T_v(G)$ is said to be an *ascendant* of a vertex $u\in T_v(G)$ if there exists a path starting from the vertex u to the root v that traverses through vertex w. The set of all ascendants *variable nodes* of the vertex u in T_v is denoted as $\mathcal{A}(u)$. For a given vertex set U, $\mathcal{A}_{T_v(G)}(U)$ denotes the set of ascendants variable nodes of all $u\in U$.

When there is no ambiguity, we will shorten the notation and use $\mathcal{A}(u)$ or $\mathcal{A}_{T_v}(u)$ instead of $\mathcal{A}_{T_v(G)}(u)$. We use $\mathcal{A}(U)$ in a similar fashion.

Definition 4: A vertex $w\in T_v$ is said to be a *parent* of a vertex $u\in T_v$ if w is directly connected to u along the path traversing to the root. Again, it is clear that $w\in\mathcal{N}_u$.

Definition 5: A vertex $w\in T_v(G)$ is said to be a *descendant* of a vertex $u\in T_v(G)$ if there exists a path starting from vertex w to the root v that traverses through vertex u. The set of all descendants of the vertex u in T_v is denoted as $\mathcal{D}(u)$. For a given vertex set U, $\mathcal{D}_{T_v(G)}(U)$ denotes the set of descendants of all $u\in U$. A vertex $w\in T_v$ is said to be a *child* of a vertex $u\in T_v$ if w is directly connected to u along the path traversing to the root. Clearly, $w\in\mathcal{N}_u$. A vertex that does not have any child nodes in T_v is called a *leaf node*.

III. Combinatorial Characterization of $(5,3)$ and $(6,4)$ Trapping Sets

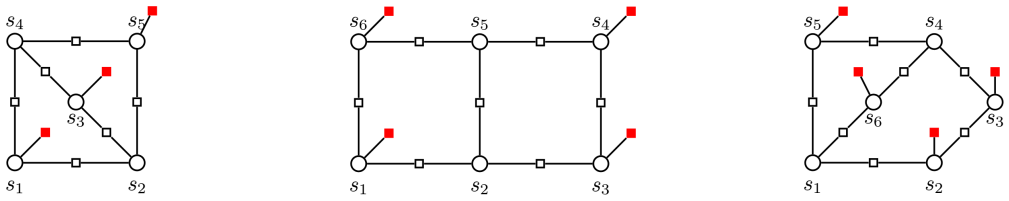
Consider $\mathcal{C}_{d_v,g}$ codes, with $d_v=3$ and $g=8$. [4], [5], [11]. The trapping sets are formed by combination of several cycles. TS can be differentiate by the number of distinct cycles and variable nodes in common on the distinct cycles that form TS.

Lemma 1: There are no non-elementary trapping sets of the type $\mathcal{T}(5,3,8^3)$.

Theorem 1: Let $\mathcal{T}$ be a subgraph associated with an elementary $TS(5,3,8^3)$. Then $\mathcal{T}$ can be only this topologie: any two cycles of 8-cycles in $\mathcal{T}$ have exactly three variable nodes in common.

Theorem 2: Let $\mathcal{T}$ be a subgraph associated with an elementary $TS(6,4,8^2,12)$. Then $\mathcal{T}$ can be only this topologie: any two cycles of 8-cycles in $\mathcal{T}$ have exactly two variable nodes in common.

Theorem 3: Let $\mathcal{T}$ be a subgraph associated with an elementary $TS(6,4,8^1,10^2)$. Then $\mathcal{T}$ can be only this topologie: any two cycles of 10-cycles in $\mathcal{T}$ have exactly four variable nodes in common.



(a) (b) (c)

Fig. 1. TS$(5,3,8^3)$; TS$(6,4,8^2,12)$; TS$(6,4,8^1,10^2)$ respectively

IV. Main Theorem

In PEG algorithm, TS can be predicted for a new edge when we use copies of check nodes c with their apparition levels and the variable nodes in common on the distinct paths $v,.....,c_i^{l_i}$ in T_v .

Theorem 4: Let T_v contain m copies of a check node c, the maximum number of cycles that contains c is $\eta = \binom{m}{2}$. If we chose to connect v and c for the new edge then the maximum number of cycles for a new structure is $\eta + m = \frac{m(m+1)}{2}$.

Lemma 2: For detecting a combination of α cycles in T_v, we need a check node c that have m copies with $m = \lfloor \frac{-1+\sqrt{1+8\alpha}}{2} \rfloor$.

Lemma 3: For detecting a combination of three cycles, we need a check node c that have at least two copies in T_v.

Theorem 5: Let v a variable node of the elementary trapping set and c a check node in T_v with m copies denoted $c_i^{l_i}$ $1 \leq i \leq m$. The length of cycle formed by two distinct paths $v,...,c_1^{l_1}$ and $v,.....,c_2^{l_2}$ in T_v is defined as follows:

- $g = l_1 + l_2 + 6 - 4n$, if $c_1^{l_1}$ and $c_2^{l_2}$ have same *p*arent.
- $g = l_1 + l_2 + 4 - 4n$, if $c_1^{l_1}$ and $c_2^{l_2}$ have not same *p*arent.

Where n is the number of variable-nodes in common to the 2 paths *(ie.* $n = |\mathcal{A}(c_1^{l_1}) \bigcap \mathcal{A}(c_2^{l_2})|$*)*.

Theorem 6: Let $\mathcal{C}_{3,8}$, for each and every variable node v a trapping-set $TS(5,3,8^3)$ is detected in T_v if and only if there exists at least 2 copies of check node c denoted by c_1^7, c_2^7 with the same *p*arent, for which the path $v,...,c_1^7$ and the path $v,.....,c_2^7$ in T_v have three common variable nodes *(ie.* $|\mathcal{A}(c_1^7) \bigcap \mathcal{A}(c_2^7)| = 3$*)*.

Theorem 7: Let $\mathcal{C}_{3,8}$, for each and every variable node v a trapping-set $TS(6,4,8^2,12^1)$ is detected in T_v if and only if we have one of two case:

- there exists at least 2 copies of check node c denoted by c_1^7, c_2^{11} with the same *p*arent, for which the path $v,...,c_1^7$ and the path $v,....,c_2^{11}$ in T_v have four common variable nodes *(ie.* $|\mathcal{A}(c_1^7)\bigcap\mathcal{A}(c_2^{11})|=4$*)*.
- there exists at least 2 copies of check node c denoted by c_1^7, c_2^7 with the same *p*arent, for which the path $v,...,c_1^7$ and the path $v,....,c_2^7$ in T_v have two common variable nodes *(ie.* $|\mathcal{A}(c_1^7)\bigcap\mathcal{A}(c_2^7)|=2$*)*.

Theorem 8: Let $\mathcal{C}_{3,8}$, for each and every variable node v a trapping-set $TS(6,4,8^1,10^1)$ is detected in T_v if and only if we have one of two case:

- there exists at least 2 copies of check node c denoted by c_1^7, c_2^9 with the same *p*arent, for which the path $v,...,c_1^7$ and the path $v,....,c_2^9$ in T_v have three common variable nodes *(ie.* $|\mathcal{A}(c_1^7)\bigcap\mathcal{A}(c_2^9)|=3$*)*.
- there exists at least 2 copies of check node c denoted by c_1^9, c_2^9 with the same *p*arent, for which the path $v,...,c_1^9$ and the path $v,....,c_2^9$ in T_v have four common variable nodes *(ie.* $|\mathcal{A}(c_1^9)\bigcap\mathcal{A}(c_2^9)|=4$*)*.

V. Design Algorithm Using RandPEG

In this section, we briefly review PEG algorithm and the RandPEG algorithm proposed by *Venkiah and al.* [9], then describe our improvement made on the RandPEG algorithm. Our improvement is essentially to add a new constraint in the *objective function* [9] of RandPEG. This constraint itself consists in detecting the tapping-set $TS(5,3)$, $TS(6,4)$, removing all check nodes that create trapping set TS(5,3) and minimizing the trapping-set $TS(6,4)$.

The original PEG algorithm [8] is a procedure for constructing a bipartite graph in an edge by edge manner, where the selection of each new edge aims at minimizing the impact on the girth: at each step the local girth is maximized [6], [8]. The RandPEG algorithm [9] is a improvement of the PEG algorithm, and allows the construction of LDPC code with a target girth g while minimizing the number of cycles created at each stage of construction but it does not take into account the trapping set in the objective function.

Let $\mathcal{C}_{3,8}$, we seek to construct of LDPC codes without trapping sets (5,3). When the construction tree is spread up to a depth truncated $k_{max}\geq 3$, the *cost function* restricts the set of check nodes candidates as follows:

- if $\overline{\mathcal{M}}_v^{k_{max}}\neq\emptyset$, discard all check nodes that are at least once a $depth<3$ (ie. $level<7$) in the tree. We only create cycles of size ≥ 8.
- For each remaining check node c_m, compute $nbtrapping_m[i]_{0\leq i\leq 1}$, the number of trapping set TS(5,3) and TS(6,4) that would be created if c is selected. Discard all check node that would create trapping sets TS(5,3) $i.e\ nbtrapping_m[0]\neq 0$ and all check-nodes that create more than $min_m(nbtrapping_m[1])$.
- For each remaining check node c_m, compute $nbCycles_m$, the number of cycles that would be created if c_m is selected. Discard all check nodes that would create more than $min_m(nbCycles_m)$.
- Compute d_c^{min}, the lowest degree of all remaining check nodes. Discard all check nodes with current degree $d_c>d_c^{min}$.

  At this point, the algorithm randomly samples among the remaining check nodes.

VI. SIMULATION RESULTS

In this section we evaluate the upper-bound L for different values of d_c in QC-LDPC code and we illustrate the efficiency of our improvement by the simulation of error performance. In our simulation, we use $d_v = 3$, $g_t = 8$. Fossorier in [12] suggest a "guest-and-test" and Y. Wang & al. suggest "Hill-climbing" in [13]. These methods do not take into account the trapping-sets. We compare the upper bound L_{min} for regulars QC-LDPC codes without $TS(5,3)$ with "guest-and-test" and "Hill-climbing" in table I.

d_c	4	5	6	7	8	9	10
Guest-and-test	9	14	18	21	26	33	39
Hill-climbing	9	3	18	21	25	30	35
RandPEG	9	13	18	21	25	31	37
RandPEG-TS(5,3)	12	18	23	31	40	52	65

TABLE I

SMALLEST VALUE OF L_{min} FOR (3,d_c)-REGULAR QC-LDPC CODES WITH GIRTH $g_t = 8$

In table II we compare the number of cycles and trapping sets between RandPEG and RandPEG without (5,3) where L_min is constant.

L	12	18	23	31	40	52
d_c	4	5	6	7	8	9
$\#cycle-8$ (RandPEG)	264	756	1704	3379	5820	
$\#cycle-8$ (RandPEG-TS(5,3))	276	720	1764	3255	5820	
$\#TS(5,3)$ (RandPEG)	48	108	216	279	360	
$\#TS(5,3)$ (RandPEG-TS(5,3))	0	0	0	0	0	0
$\#TS(6,4;8^2,12)$ (RandPEG)	864	2142	5568	12307	21200	
$\#TS(6,4;8^2,12)$ (RandPEG TS(5,3))	936	1998	6792	10850	22280	
$\#TS(7,3;8^3,12^4)$ (RandPEG)	408	900	2136	4557	6960	
$\#TS(7,3;8^3,12^4)$ (RandPEG TS(5,3))	624	666	3072	4774	8040	

TABLE II

COMPARISON OF RANDPEG AND RANDPEG WITHOUT TS(5,3): CONSTANT SIZE

For illustrate the performance of the QC-LDPC code, we compare our results with the Tanner code $L = 31$, $d_c = 5$ in table III.

The error performance is better when the LDPC codes have not trapping-set TS(5,3) and the trapping set TS(5,3) is more harmful.

VII. CONCLUSIONS AND REMARKS

In this paper we proposed some improvement on the RandPEG algorithm. Our improvement is to detect the trapping-set $TS(5,3)$ and $TS(6,4)$ then delete $TS(5,3)$ and minimizing $TS(6,4)$. We demonstrated the efficiency

	Tanner	RandPEG	RandPEG without TS(5,3)
$\#cycle-8$	465	620	496
$\#cycle-10$	3720	3348	3596
$\#TS(5,3)$	155	31	0
$\#TS(6,4;8^2,12)$	0	930	651
$\#TS(6,4;8,10^2)$	930	992	713
$\#TS(6,4)$	930	1922	1364
$\#TS(7,3;8^3,12^4)$	0	186	31
$\#TS(7,3;8^2,10^3,12,14)$	0	433	248
$\#TS(7,3;8^2,10^2,14^2)$	930	93	0
$\#TS(7,3)$	930	712	279

TABLE III

COMPARISON TANNER, RANDPEG AND RANDPEG WITHOUT TS(5,3) FOR $L=31$, $d_v=3$, $d_c=5$

of our improvement by simulation with Belief-Propagation algorithm.

APPENDIX A
PROOF LEMMA 1

Proof: Let $\mathcal{C}_{3,8}$, a $8-cycle$ contains 4 variable nodes with 8 neighbors check nodes such as 4 check nodes have degree one and the 4 others check nodes have degree two. Adding a new variable node that gives a new 8-cycle implies 2 check nodes along the 4 check nodes of degree two form edges with new variable. Hence $\mathcal{T}(5,3)$ contains only degree-one and/or degree-two check nodes

■

APPENDIX B
PROOF THEOREM 4

Proof: Let T_v contains m copies of check node c, the maximum number of cycles that contains c is $\eta = \binom{m}{2} = \frac{m(m-1)}{2}$, if we chose to connect v and c we have $\eta + m = \frac{m(m-1)}{2} + m = \frac{m(m+1)}{2}$ cycles. ■

APPENDIX C
PROOF LEMMA 2 & LEMMA 3

Proof: For a combination of α cycles, we have : $\alpha \geq \frac{m(m+1)}{2} \Rightarrow m = \lfloor \frac{-1+\sqrt{1+8\alpha}}{2} \rfloor$.
For a combination of three cycles, $\alpha = 3 \Rightarrow m = \lfloor \frac{-1+\sqrt{1+8*3}}{2} \rfloor = 2$ ■

APPENDIX D
PROOF THEOREM 5

Proof: Assume that there is m copies of check-node c in T_v. Let two copies of c denoted by $c_1^{l_1}$, $c_2^{l_2}$ respectively, $\mathcal{A}(c_i^{l_i})$ the set of all ascendants variable nodes of $c_i^{l_i}$ and n the number of variable nodes in common

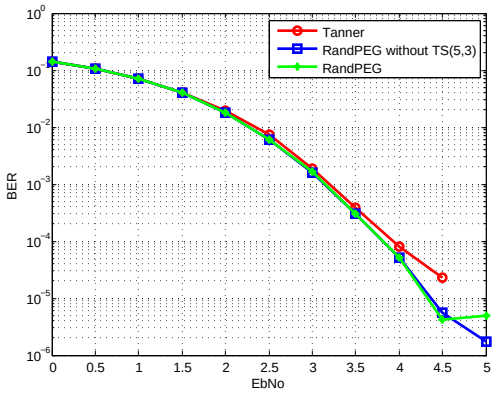


Fig. 2. Bit Error Rate with Belief Propagation decoding with $L = 31$, $d_c = 5$, $d_v = 3$ (100 iterations)

between $\mathcal{A}(c_1^{l_1})$ and $\mathcal{A}(c_2^{l_2})$ ie. $n = |\mathcal{A}(c_1^{l_1}) \bigcap \mathcal{A}(c_2^{l_2})|$. The number of variable nodes in the set $\mathcal{A}(c_1^{l_1}) \bigcup \mathcal{A}(c_2^{l_2})$ is, $|\mathcal{A}(c_1^{l_1}) \bigcup \mathcal{A}(c_2^{l_2})| = \frac{l_1+1}{2} + \frac{l_2+1}{2} - n = \frac{l_1+l_2+2-2n}{2}$

- if $c_1^{l_1}$ and $c_2^{l_2}$ have the same *parent* there is two variable node among the variable nodes in common which belongs to the set of variable nodes that form the cycle hence we have $\frac{l_1+l_2+2-2n}{2} - (n-2)$ variable nodes to the cycle, and the length of this cycle is $g = 2\frac{l_1+l_2+6-4n}{2} = l_1 + l_2 + 6 - 4n$.
- if $c_1^{l_1}$ and $c_2^{l_2}$ have not the same *parent* hence we have $\frac{l_1+l_2+2-2n}{2} - (n-1)$ variable nodes to the cycle, and the length of this cycle is $g = 2\frac{l_1+l_2+4-4n}{2} = l_1 + l_2 + 4 - 4n$

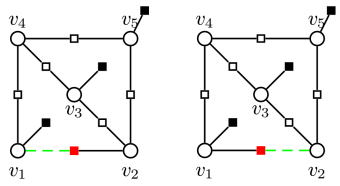
■

APPENDIX E
PROOF THEOREM 6

Proof: A $TS(5,3,8^3)$ is composed by three cycles. Let $\mathcal{C}_{3,8}$ and T_v contain m copies of a check-node c such as two copies are denoted by $c_1^{l_1}$ and $c_2^{l_2}$. For the elementary trapping set all check nodes have two degree two neighboors at maximum hence if $c_1^{l_1}$ and $c_2^{l_2}$ have not the same *parent* the trapping set is not elementary.
Let $g_1 = l_1 + 1$, $g_2 = l_2 + 1$, and n the number of variable nodes in common between $\mathcal{A}(c_1^{l_1})$ and $\mathcal{A}(c_2^{l_2})$ ie. $n = |\mathcal{A}(c_1^{l_1}) \bigcap \mathcal{A}(c_2^{l_2})|$.

- $l_1 \neq 7$ *(resp.* $l_2 \neq 7$*)* $\Rightarrow g_1 \neq 8$ *(resp.* $g_2 \neq 8$*)*. Hence there is no $TS(5,3,8^3)$
- For $l_1 = 7$ and $l_2 = 7$, the length of cycle formed by two distinct paths $v, ..., c_1^7$ and $v, ..., c_2^7$ is:
  $g = 7 + 7 + 6 - 4n$ *Theorem 5*
  $g = 20 - 4n$
  $n \neq 3 \Rightarrow g \neq 8$
  $n = 3 \Rightarrow g = 8$, and $|\mathcal{A}(c_1^7) \bigcup \mathcal{A}(c_2^7)| = 5$

Hence we have a $TS(5,3,8^3)$, if $l_1 = l_2 = 7$, and $n = 3$. In the figure 3 the symbol nodes v_1, v_3, v_5 have the same configuration and v_2, v_4 have the same configuration also. We illustrate the different case to obtain the $TS(5,3)$.



(a) (b)

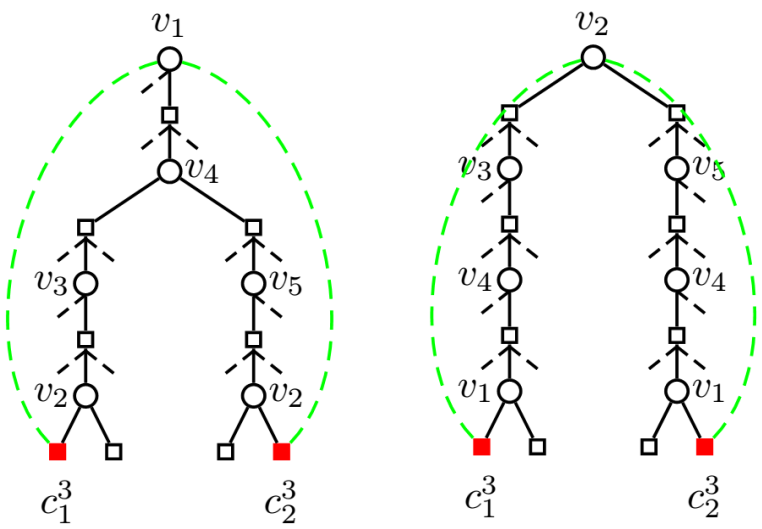
Fig. 3. New edge giving $TS(5,3,8^3)$, - - -

■

APPENDIX F
PROOF THEOREM 7

Proof: A $TS(6,4,8^2,12^1)$ is composed by three cycles. Let $\mathcal{C}_{3,8}$ and T_v contain m copies of a check-node c such as two copies are denoted by $c_1^{l_1}$ and $c_2^{l_2}$. For the elementary trapping set all check nodes have two degree two neighboors at maximum hence if $c_1^{l_1}$ and $c_2^{l_2}$ have not the same *parent* the trapping set is not elementary.
Let $g_1 = l_1 + 1$, $g_2 = l_2 + 1$, and n the number of variable nodes in common between $\mathcal{A}(c_1^{l_1})$ and $\mathcal{A}(c_2^{l_2})$ ie. $n = |\mathcal{A}(c_1^{l_1}) \bigcap \mathcal{A}(c_2^{l_2})|$. $(6,4,8^2,12^1)$ contains two 8-cycles and one 12-cycle hence there is two case levels for c:

- **Case one:** $(l_1 = 7, l_2 = 11)$ hence $(g_1 = 8, g_2 = 12)$ or $(l_1 = 11, l_2 = 7)$ hence $(g_1 = 12, g_2 = 8)$, the length of cycle formed by two distinct path $v, ..., c_1^{l_1}$ and $v, ..., c_2^{l_2}$ is:



(a) (b)

Fig. 4. depth-7 tree of two subgraphs respectively

$g = 7 + 11 + 6 - 4n = 24 - 4n$

$n \neq 4 \Rightarrow g \neq 8$

$n = 4 \Rightarrow g = 8$ and $|\mathcal{A}(c_1^{l_1}) \bigcup \mathcal{A}(c_2^{l_2})| = 6$

- **Case two:** $(l_1 = l_2 = 7)$ hence $(g_1 = g_2 = 8)$, the length of cycle formed by two distinct path $v, ..., c_1^{l_1}$ and $v, ..., c_2^{l_2}$ is:

  $g = 7 + 7 + 6 - 4n = 20 - 4n$

  $n \neq 2 \Rightarrow g \neq 12$

  $n = 2 \Rightarrow g = 12$ and $|\mathcal{A}(c_1^{l_1}) \bigcup \mathcal{A}(c_2^{l_2})| = 6$

■

APPENDIX G
PROOF THEOREM 8

Proof: A $TS(6, 4, 8^1, 10^2)$ is composed by three cycles. Let $\mathcal{C}_{3,8}$ and T_v contain m copies of a check-node c such as two copies are denoted by $c_1^{l_1}$ and $c_2^{l_2}$. For the elementary trapping set all check nodes have two degree two neighboors at maximum hence if $c_1^{l_1}$ and $c_2^{l_2}$ have not the same *parent* the trapping set is not elementary. Let $g_1 = l_1 + 1$, $g_2 = l_2 + 1$, and n the number of variable nodes in common between $\mathcal{A}(c_1^{l_1})$ and $\mathcal{A}(c_2^{l_2})$ ie. $n = |\mathcal{A}(c_1^{l_1}) \bigcap \mathcal{A}(c_2^{l_2})|$. $(6, 4, 8^1, 10^2)$ contains one 8-cycle and two 10-cycles hence there is two case levels for c:

- **Case one:** $(l_1 = 7, l_2 = 9)$ hence $(g_1 = 8, g_2 = 10)$ or $(l_1 = 9, l_2 = 7)$ hence $(g_1 = 10, g_2 = 8)$, the length of cycle formed by two distinct path $v, ..., c_1^{l_1}$ and $v, ..., c_2^{l_2}$ is:

  $g = 7 + 9 + 6 - 4n = 22 - 4n$

$n \neq 3 \Rightarrow g \neq 10$

$n = 3 \Rightarrow g = 10$ and $|\mathcal{A}(c_1^{l_1}) \bigcup \mathcal{A}(c_2^{l_2})| = 6$

- **Case two:** $(l_1 = l_2 = 9)$ hence $(g_1 = g_2 = 10)$, the length of cycle formed by two distinct path $v, ..., c_1^{l_1}$ and $v, ..., c_2^{l_2}$ is:

  $g = 9 + 9 + 6 - 4n = 24 - 4n$

  $n \neq 4 \Rightarrow g \neq 8$

  $n = 4 \Rightarrow g = 8$ and $|\mathcal{A}(c_1^{l_1}) \bigcup \mathcal{A}(c_2^{l_2})| = 6$

■

REFERENCES

[1] T. Tian, C. Jones, J. D. Villasenor, and D. Wesel, "Construction of irregular ldpc codes with low error floors," in *Proc. IEEE Int. Conf. on Communications*, vol. 5, May 2003, pp. 3125–3129.

[2] X. Zheng, F. C. M. Lau, C. K. Tse, and Y. He, "Constructing short-length irregular ldpc codes with low error floor," in *Proc. IEEE Commun.*, vol. 58, Oct. 2010, pp. 2823–2834.

[3] H. Xinde, L. Zongwang, B. V. K. V. Kumar, and R. Barndt, "Error floor estimation of long ldpc codes on magnetic recording channels," in *Mag. IEEE Trans.*, vol. 46, no. 6, June 2010, pp. 1836–1839.

[4] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annual Allerton Conf. on Communications, Control and Computing*, 2003, pp. 1426–1435. [Online]. Available: http://www.hpl.hp.com/personal/Pascal_Vontobel/pseudocodewords/papers

[5] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. Int. Conf. on Communications*, vol. 3, 2006, pp. 1089–1094.

[6] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Progressif edge-growth tanner graphs," in *Proc. IEEE GLOBECOM*, 2001, pp. 995–1001.

[7] G. Richter and A. Hof, "On a construction method of irregular ldpc codes without small stopping sets," in *Proc. IEEE Int. Conf.*, vol. 3, June 2006, pp. 1119–1124.

[8] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressif edge-growth tanner graphs," in *IEEE Trans. Inf. Theory*, vol. 51, no. 01, Jan. 2005, pp. 386–398.

[9] A. Venkiah, D. Declercq, and C. Poulliat, "Design of cages with a randomized progressive edge growth algorithm," in *IEEE Commun. Letters*, vol. 12, April 2008, pp. 301–303.

[10] X. Hua and A. H. Banihashemi, "Improved progressif edge-growth (peg) construction of irregular ldpc codes," in *IEEE Commun. Letters*, vol. 8, no. 12, Dec. 2004, pp. 715–717.

[11] D. Declercq, B. Vasic, S. K. Planjery, and E. Li, "Finite alphabet iterative decoders, Part II: Improved guaranteed error correction of LDPC codes via iterative decoder diversity," in *IEEE Trans. Commun.*, vol. 61, no. 10, Nov. 2013, pp. 4046–4057.

[12] M. P. C. Fossorier, "Quasi-cyclic low density parity check codes from circulant permutation matrices," in *IEEE Trans. Inf. Theory*, vol. 50, Aug. 2004, pp. 1788–1794.

[13] Y. Wang, J. Yedidia, and S. Draper, "Construction of high-girth qc-ldpc codes," in *Turbo Codes and Related Topics, 5th International Symposium on*, Sept. 2008, pp. 180–185.