Asymptotic Trapping Set Analysis of Regular Protograph-based LDPC Convolutional Code Ensembles

David G. M. Mitchell*, Ali E. Pusane†, and Daniel J. Costello, Jr.†
*Institute for Digital Communications, Joint Research Institute for Signal & Image Processing,
The University of Edinburgh, Scotland, David.Mitchell@ed.ac.uk
†Dept. of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana, USA,
{apusane, dcostel1}@nd.edu

***Abstract*—[1] It has been suggested that "near-codewords" may be a significant factor affecting decoding failures of LDPC codes over the AWGN channel. A near-codeword is a sequence that satisfies almost all of the check equations. These near-codewords can be associated with so-called 'trapping sets' that exist in the Tanner graph of a code. In this paper, we analyse the trapping sets of protograph-based LDPC convolutional codes. LDPC convolutional codes have been shown to be capable of achieving the same capacity-approaching performance as LDPC block codes with iterative message-passing decoding. Further, it has been shown that some ensembles of LDPC convolutional codes are asymptotically good, in the sense that the average free distance grows linearly with constraint length. Here, asymptotic methods are used to calculate a lower bound on the trapping set growth rates for several ensembles of regular asymptotically good protograph-based LDPC convolutional codes. This provides us with an estimate of where the error floor will occur for these codes under iterative message-passing decoding.**

I. Introduction

Trapping sets, graphical sub-structures existing in the Tanner graph of Low-Density Parity-Check (LDPC) codes, were first studied in [2]. Known initially as *near-codewords*, they were used to analyse the performance of LDPC codes in the error-floor, or high signal-to-noise ratio (SNR), region of the bit error rate (BER) curve. In [3], Richardson developed these concepts and proposed a two-stage technique to predict the error floor performance of LDPC codes based on trapping sets, and asymptotic results on trapping set enumerators for both regular and irregular LDPC block code ensembles were published in [4].

LDPC convolutional codes were introduced in [5], and their advantages and disadvantages compared to LDPC block codes of the same complexity were discussed in [6], [7]. Further, in [8] and [9], it was shown that several ensembles of both regular and irregular unterminated periodically time-varying LDPC convolutional codes based on protographs are *asymptotically good*. In other words, their average free distance grows linearly with constraint length. Also, a lower bound on the free distance growth rates was obtained, which was shown to exceed the growth rates of minimum distance with block length for corresponding protograph-based LDPC block code ensembles.

The analysis used in [8] and [9] to calculate ensemble average weight enumerators can be extended to the problem of finding ensemble average trapping set enumerators. In this paper, building on work by Abu-Surra, Ryan, and Divsalar [10], asymptotic methods are used to calculate a lower bound on the average trapping set enumerators for several ensembles of regular, asymptotically good, protograph-based LDPC convolutional codes. In particular, we use ensembles of tail-biting LDPC convolutional codes (introduced in [11]) derived from a single protograph-based ensemble of LDPC block codes to obtain a lower bound on the average trapping set enumerators of unterminated, asymptotically good, periodically time-varying LDPC convolutional code ensembles. In the process, we show that the average trapping set enumerators of ensembles of tail-biting LDPC convolutional codes approach the average trapping set enumerator of an associated LDPC convolutional code ensemble as the block length of the tail-biting ensemble increases.

II. LDPC convolutional codes

We start with a brief definition of a rate $R = b/c$ binary LDPC convolutional code $\mathcal{C}$. A code sequence $\mathbf{v}_{[0,\infty]}$ satisfies the equation $\mathbf{v}_{[0,\infty]}\mathbf{H}^{\mathsf{T}}_{[0,\infty]} = \mathbf{0}$, where $\mathbf{H}^{\mathsf{T}}_{[0,\infty]}$ is the syndrome former matrix and

$$\mathbf{H}_{[0,\infty]} = \begin{bmatrix} \mathbf{H}_0(0) & & & & \\ \mathbf{H}_1(1) & \mathbf{H}_0(1) & & & \\ \vdots & \vdots & & \ddots & \\ \mathbf{H}_{m_\mathrm{s}}(m_\mathrm{s}) & \mathbf{H}_{m_\mathrm{s}-1}(m_\mathrm{s}) & \cdots & \mathbf{H}_0(m_\mathrm{s}) & \\ & \mathbf{H}_{m_\mathrm{s}}(m_\mathrm{s}+1) & \mathbf{H}_{m_\mathrm{s}-1}(m_\mathrm{s}+1) & \cdots & \mathbf{H}_0(m_\mathrm{s}+1) \\ & \ddots & \ddots & & \ddots \end{bmatrix}$$

is the parity-check matrix of the convolutional code $\mathcal{C}$. The submatrices $\mathbf{H}_i(t)$, $i = 0, 1, \cdots, m_\mathrm{s}$, $t \geq 0$, are binary $(c-b) \times c$ submatrices, given by

$$\mathbf{H}_i(t) = \begin{bmatrix} h_i^{(1,1)}(t) & \cdots & h_i^{(1,c)}(t) \\ \vdots & & \vdots \\ h_i^{(c-b,1)}(t) & \cdots & h_i^{(c-b,c)}(t) \end{bmatrix}, \qquad (1)$$

that satisfy the following properties:

1) $\mathbf{H}_i(t) = \mathbf{0}$, $i < 0$ and $i > m_s$, $\forall\, t$.
2) There is a t such that $\mathbf{H}_{m_\mathrm{s}}(t) \neq \mathbf{0}$.

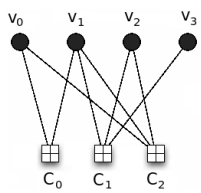
We call m_s the syndrome former memory and $\nu_\mathrm{s} = (m_\mathrm{s}+1)\cdot c$ the decoding constraint length. These parameters determine the width of the nonzero diagonal region of $\mathbf{H}_{[0,\infty]}$. The sparsity of the parity-check matrix is ensured by demanding

[1]This is an extended version of [1] that has been submitted to the 2009 IEEE International Symposium on Information Theory. Much of the background material is repeated here in order to make the paper self-contained.

that its rows have very low Hamming weight, i.e., $w_H(\mathbf{h}_i) << (m_\mathrm{s}+1)\cdot c,\ i>0$, where $\mathbf{h}_i$ denotes the i-th row of $\mathbf{H}_{[0,\infty]}$. The code is said to be regular if its parity-check matrix $\mathbf{H}_{[0,\infty]}$ has exactly J ones in every column and, starting from row $(c-b)m_\mathrm{s}+1$, K ones in every row. The other entries are zeros. We refer to a code with these properties as an (m_s, J, K)-regular LDPC convolutional code, and we note that, in general, the code is time-varying and has rate $R = b/c = 1 - J/K$. A rate $R = b/c$, (m_s, J, K)-regular time-varying LDPC convolutional code is periodic with period T (in submatrices) if $\mathbf{H}_i(t)$ is periodic, i.e., $\mathbf{H}_i(t) = \mathbf{H}_i(t+T), \forall\ i, t$, and if $\mathbf{H}_i(t) = \mathbf{H}_i, \forall\ i, t$, the code is time-invariant. An LDPC convolutional code is called *irregular* if its row and column weights are not constant.

III. Protograph-based LDPC convolutional codes

A protograph is a small bipartite graph. Figure 1 shows a protograph and the associated protograph parity-check matrix.



$$P = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Fig. 1: An example protograph and associated parity-check matrix.

Suppose a given protograph has n_v variable nodes and n_c check nodes. An ensemble of protograph-based LDPC block codes can be created using the copy-and-permute operation [13]. The parity-check matrix H corresponding to a member of the resulting ensemble of protograph-based LDPC block codes can be obtained by replacing ones with $N \times N$ permutation matrices and zeros with $N \times N$ all-zero matrices in the underlying protograph parity-check matrix P, where the permutation matrices are chosen randomly and independently.

Using this construction, the sparsity condition of an LDPC parity-check matrix will be satisfied for large N. The code created by applying the copy-and-permute operation to an $n_c \times n_v$ protograph parity-check matrix P has block length $n = Nn_v$. In addition, the code has the same rate and degree distribution for each of its variable and check nodes as the underlying protograph.

We note that, in the example of Figure 1, the row and column weights of P are not constant, so P represents the parity-check matrix of an irregular protograph. The resulting ensemble of protograph-based block codes will thus also be irregular. Note that it is also possible to consider protograph parity-check matrices P with larger integer entries, which represent parallel edges in the base protograph. In this case, the resulting block in H consists of a sum of $N \times N$ permutation matrices [13].

A. Forming protograph-based convolutional codes

Suppose that we have an $n_c \times n_v$ protograph parity-check matrix P, where $\gcd(n_c, n_v) = y$. We then partition P as a $y \times y$ block matrix as follows:

$$P = \begin{bmatrix} P_{1,1} & \dots & P_{1,y} \\ \vdots & & \vdots \\ P_{y,1} & \dots & P_{y,y} \end{bmatrix},$$

where each block $P_{i,j}$ is of size $n_c/y \times n_v/y$. P can thus be separated into a lower triangular part, P_l, and an upper triangular part minus the leading diagonal, P_u. Explicitly,

$$P_l = \begin{bmatrix} P_{1,1} & & & \\ P_{2,1} & P_{2,2} & & \\ \vdots & \vdots & \ddots & \\ P_{y,1} & P_{y,2} & \dots & P_{y,y} \end{bmatrix}_{n_c \times n_v}$$

and

$$P_u = \begin{bmatrix} & P_{1,2} & \dots & P_{1,y} \\ & & \ddots & \vdots \\ & & & P_{y-1,y} \\ & & & \end{bmatrix}_{n_c \times n_v},$$

where blank spaces correspond to zeros. This operation is called 'cutting' a protograph parity-check matrix.

Rearranging the positions of these two triangular matrices and repeating them indefinitely results in a parity-check matrix P_{cc} of an unterminated, periodically time-varying convolutional code with rate $R = 1 - n_c/n_v$, constraint length $\nu_\mathrm{s} = n_v$, and period $T = y$ given by[2]

$$P_{cc} = \begin{bmatrix} P_l & & & \\ P_u & P_l & & \\ & P_u & P_l & \\ & & \ddots & \ddots \end{bmatrix}. \tag{2}$$

Note that when $\gcd(n_c, n_v) = 1$, we cannot form a square block matrix larger than 1×1 with equal size blocks. In this case, $P_l = P$ and P_u is the all-zero matrix of size $n_c \times n_v$. This trivial cut results in a convolutional code with syndrome former memory zero, with repeating blocks of the original protograph on the leading diagonal. In this case we must use one of two approaches. First, we can create a larger protograph parity-check matrix by applying the copy and permute operation M times to P. This results in an $Mn_c \times Mn_v = n'_c \times n'_v$ parity-check matrix for some small integer M. The $n'_c \times n'_v$ protograph parity-check matrix can then be cut following the procedure outlined above, where $\gcd(n'_c, n'_v) = M$.

Alternatively, we can use a nonuniform cut. Let the protograph parity-check matrix be written as

$$P = \begin{bmatrix} p_{1,1} & \dots & p_{1,n_v} \\ \vdots & & \vdots \\ p_{n_c,1} & \dots & p_{n_c,n_v} \end{bmatrix}.$$

We define a vector ξ consisting of n_c step parameters $\xi = (\xi_1, \xi_2, \dots, \xi_{n_c})$, where $0 \le \xi_1 < \dots < \xi_{n_c} \le n_v$, and each $\xi_{i-1} < \xi_i$ for $i = 2, \dots, n_c$. As in the previous case, we form $n_c \times n_v$ matrices P_l and P_u as follows

[2]Cutting certain protograph parity-check matrices may result in a smaller period $T = y'$, where $y' \in \mathbb{Z}^+$ divides y evenly. If $y' = 1$, then the resulting convolutional code is time-invariant.

- for each ξ_i, $i = 1, \ldots, n_c$, the entries $p_{i,1}$ to p_{i,ξ_i} are copied into the equivalent positions in P_l;
- entries p_{i,ξ_i+1} to p_{i,n_v} are copied, if they exist, into the equivalent positions in P_u;
- the remaining positions in P_l and P_u are set to zero.

An LDPC convolutional code derived from an LDPC block code using a nonuniform cut can be encoded and decoded using conventional encoding and decoding methods with minor modifications. For further details, see [9].

To create an ensemble of time-varying LDPC convolutional codes, we follow the usual protograph construction technique. The ones are replaced with $N \times N$ permutation matrices (or summations of $N \times N$ permutation matrices for parallel edges) and the zeros with $N \times N$ all-zero matrices, where the permutation matrices are chosen randomly and independently. Choosing N to be sufficiently large guarantees the sparsity condition for an LDPC code.

IV. General trapping set enumerators for protograph-based codes

Definition 1: An (a, b) general trapping set $\tau_{a,b}$ of a bipartite graph is a set of variable nodes of size a which induce a subgraph with exactly b odd-degree check nodes (and an arbitrary number of even-degree check nodes).

In order to calculate ensemble average general trapping set enumerators for protograph-based block codes, we make use of the combinatorial arguments previously presented in [14] and [15] for calculating ensemble average weight enumerators. The technique involves considering a two-part weight enumerator for a modified protograph with the property that any (a, b) trapping set in the original protograph is a codeword in the modified protograph. We now briefly describe the procedure introduced in [10]. An auxiliary 'flag' variable node is added to each check node, as displayed in Figure 2.

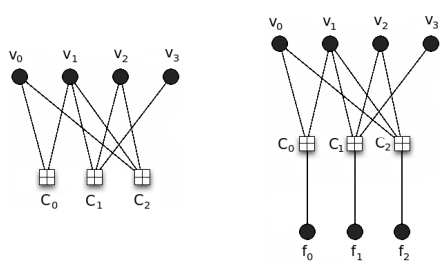


Fig. 2: An example protograph and modified version with auxiliary variable nodes.

Consider a subset S with cardinality a of the variable nodes $V = \{v_0, v_1, v_2, v_3\}$, for example, $a = 3$ and $S = \{v_0, v_1, v_2\}$. We now attach weight 1 to these variable nodes and weight 0 to the remaining nodes in $V/S = \{v_3\}$. We observe that check nodes c_0 and c_1 are satisfied, since they both have input weight 2, but that check node c_2 (with input weight 3) is unsatisfied. Thus there is $b = 1$ odd-degree (unsatisfied) check node. This is an example of a $(3, 1)$ general trapping set. Thus $\tau_{3,1}$ contains the subset $S = \{v_0, v_1, v_2\}$.

For any subset of variable nodes, we can satisfy any odd-degree check nodes by assigning weight 1 to the corresponding auxiliary variable node. Note that the weight of the variable nodes V is $a = 3$ and the weight of the auxiliary nodes is $b = 1$ for this $(3, 1)$ general trapping set, which suggests that the general trapping sets of a protograph can be enumerated by applying a two-part weight enumerator analysis to the modified protograph. We thus consider a two-part weight enumerator over sets of variable nodes $V = \{v_0, v_1, \ldots, v_{n_v-1}\}$ and auxiliary nodes $F = \{f_0, f_1, \ldots, f_{n_f-1}\}$, where n_v is the number of variable nodes in the initial protograph and n_f is the number of auxiliary variable nodes (equal to the number of check nodes n_c). This method of enumerating trapping sets for protograph-based codes is presented in [10]. In the remainder of this section, we summarize the results of [10] and [14] on which our approach is based.

A. Finite length ensemble trapping set enumerators

Suppose that a modified protograph contains n_v variable nodes to be transmitted over the channel. Also, suppose that each of the n_v transmitted variable nodes has an associated weight d_i, where $0 \leq d_i \leq N$ for all i.[3] Let $S_d = \{(d_0, d_1, \ldots, d_{n_v-1})\}$ be the set of all possible weight distributions such that $d_0 + \ldots + d_{n_v-1} = a$. Finally, suppose that $S_f = \{(f_0, f_1, \ldots, f_{n_f-1})\}$ is the set of all weight distributions such that $f_0 + \ldots + f_{n_f-1} = b$, where $0 \leq f_i \leq N$ for all i. Then the two-part ensemble average trapping set enumerator for the modified protograph is given by

$$A_{a,b} = \sum_{\{d_k\} \in S_d} \sum_{\{f_l\} \in S_f} A_{\mathbf{d}}, \tag{3}$$

where $A_{\mathbf{d}}$ is the average number of codewords in the modified ensemble with weight distribution $\mathbf{d} = (d_0, d_1, \ldots, d_{n_v-1}, f_0, f_1, \ldots, f_{n_f-1})$.

B. Asymptotic trapping set spectral shape function

The two-part normalized logarithmic *asymptotic trapping set spectral shape function* of a code ensemble can be written as

$$r(\alpha, \beta) = \lim_{n \to \infty} \sup r_n(\alpha, \beta),$$

where $r_n(\alpha, \beta) = \frac{\ln(A_{a,b})}{n}$, $\alpha = a/n$, $\beta = b/n$, a and b are Hamming weights, n is the block length, and $A_{a,b}$ is the two-part ensemble average weight distribution.

Suppose now we are interested in the ratio of b to a for a general (a, b) trapping set enumerator. Let $\Delta = b/a = \beta/\alpha$, $\Delta \in [0, \infty)$. As proposed in [10], we may now classify the trapping sets as

$$\tau_\Delta = \{\tau_{a,b} | b = \Delta \cdot a\}. \tag{4}$$

For each Δ, we define $d_{ts}(\Delta)$ to be the Δ-*trapping set number*, which is the size of the smallest, non-empty trapping set in τ_Δ. Now consider fixing Δ and plotting the normalized weight α against the two-part asymptotic spectral shape function $r(\alpha, \beta) = r(\alpha, \Delta\alpha)$. Suppose $\alpha > 0$ and the first zero-crossing of $r(\alpha, \beta)$ occurs at $\alpha = \delta_{ts}(\Delta)$. If $r(\delta)$ is negative in the range $0 < \alpha < \delta_{ts}(\Delta)$, then the first zero-crossing $\delta_{ts}(\Delta)$

[3] Since we use N copies of the protograph, the weight associated with a particular variable node in the protograph can be as large as N.

is called the Δ-*trapping set growth rate* of the code ensemble. If $\delta_{ts}(\Delta)$ exists, and if the probability

$$\mathbb{P}(a < \delta_{ts}(\Delta)n) = \sum_{a=1}^{\delta_{ts}(\Delta)n-1} A_{a,b} << 1$$

as the block length n grows, we can say with high probability that the majority of codes in the ensemble have a Δ-trapping set number that increases linearly with n, i.e., $d_{ts}(\Delta) = n\delta_{ts}(\Delta)$. This implies that, for sufficiently large n, a typical member of the ensemble has no small trapping sets.

Experimental results have shown that the failure events that dominate the performance of LDPC codes with iterative decoding in the error floor region of the BER curve can be attributed to small trapping sets [3]. Since small trapping sets contain relatively few variable nodes, it is more likely that all of the variable nodes in a small trapping set have unreliable channel values than in a larger trapping set. Further, examinations of the small trapping sets dominating in the error floor region have shown that the sets with low degree check nodes cause the most trouble. This is because of the low connectivity of these check nodes to the rest of the graph, which results in the trapping set not being able to obtain enough independent (and possibly helpful) messages during decoding iterations. Thus an ensemble that guarantees no small trapping sets on average is highly desirable. General trapping sets that have only degree one or two check nodes form an important subset called *elementary trapping sets*. Extensive simulations (see, e.g., [3]) have shown that in fact most of the decoding failures in iterative decoding correspond to elementary trapping sets. Elementary trapping sets require a slightly different analytical model than the one used here, as described in [10]. Similar results to those presented in this paper for elementary trapping sets are the subject of ongoing research.

V. General trapping set enumerators for LDPC convolutional codes

In this section, we present a method for obtaining a lower bound on the Δ-trapping set number of an ensemble of unterminated, asymptotically good, periodically time-varying LDPC convolutional codes derived from protograph-based LDPC block codes. To proceed, we introduce a family of tail-biting LDPC convolutional codes with incremental increases in block length. The tail-biting codes are then used as a tool to obtain the desired bound on the Δ-trapping set number of the unterminated codes.

A. Tail-biting convolutional codes

Consider the parity-check matrix P_{cc} of the protograph-based, unterminated convolutional code introduced in Section III-A. We now introduce the notion of tail-biting convolutional codes by defining an 'unwrapping factor' λ as the number of times the sliding convolutional structure is repeated before applying tail-biting termination. For $\lambda \geq 1$, the parity-check matrix $P_{tb}^{(\lambda)}$ of the desired tail-biting protograph-based convolutional code with block length λn_v can be written as

$$P_{tb}^{(\lambda)} = \begin{bmatrix} P_l & & & & P_u \\ P_u & P_l & & & \\ & P_u & P_l & & \\ & & \ddots & \ddots & \\ & & & P_u & P_l \end{bmatrix}_{\lambda n_c \times \lambda n_v}. \quad (5)$$

Note that the tail-biting convolutional code for $\lambda = 1$ is simply the original protograph-based block code.

B. Tail-biting LDPC convolutional code ensembles

Given a protograph parity-check matrix P, we generate a family of tail-biting convolutional codes with parity check matrices $P_{tb}^{(\lambda)}$ and increasing block lengths λn_v, $\lambda = 1, 2, \ldots$, using the process described above. Since tail-biting convolutional codes are themselves block codes, we can treat the Tanner graph of $P_{tb}^{(\lambda)}$ as a protograph for each value of λ. Replacing the entries of this matrix with either $N \times N$ permutation matrices or $N \times N$ all-zero matrices, as discussed in Section III, creates an ensemble of LDPC codes that can be analysed asymptotically as N goes to infinity, where the sparsity condition of an LDPC code is satisfied for large N. Each tail-biting LDPC code ensemble, in turn, can be unwrapped and repeated indefinitely to form an ensemble of unterminated, periodically time-varying LDPC convolutional codes[4] with rate $R = 1 - Nn_c/Nn_v = 1 - n_c/n_v$, constraint length $\nu_s = Nn_v$, and, in general, period $T = \lambda y$.

To study the average general trapping set enumerators of these block codes, we add auxiliary flag variables following the procedure detailed in Section IV. The resulting protograph-based modified parity-check matrix is given by

$$\tilde{P}_{tb}^{(\lambda)} = \left[\begin{array}{ccccc|c} P_l & & & & P_u & \\ P_u & P_l & & & & \\ & P_u & P_l & & & \mathbb{I}_{\lambda n_c} \\ & & \ddots & \ddots & & \\ & & & P_u & P_l & \end{array}\right]_{\lambda n_c \times \lambda(n_v + n_c)},$$

where $\mathbb{I}_n$ is the $n \times n$ identity matrix. For any λ, we can now follow the procedure detailed in Section IV to calculate the Δ-trapping set number $d_{ts}^{(\lambda)}$ for the ensemble of LDPC tail-biting convolutional codes based on the protograph parity-check matrix $P_{tb}^{(\lambda)}$.

C. A lower bound on the convolutional Δ-trapping set growth rate

In this section we present a technique for obtaining a lower bound on the convolutional Δ-trapping set growth rate $\delta_{ccts}(\Delta)$ by showing that $\delta_{ccts}(\Delta)$ must be at least as large as any Δ-trapping set growth rate $\delta_{ts}^{(\lambda)}$ of an ensemble of block codes obtained by tail-biting termination of the underlying convolutional code with unwrapping factor λ. The method used to prove the bound is based on a similar minimum distance bound proof first presented in [12].

Theorem 1: Consider forming an unterminated, periodically time-varying convolutional code with rate $R = 1 - n_c/n_v$, constraint length $\nu_s = n_v$, and period $T = y$ from (2), as described in Section III.A. Let $d_{ts}^{(\lambda)}(\Delta)$ be the

[4] In this case, the submatrices of P_l and P_u are of size $Nn_c/y \times Nn_v/y$.

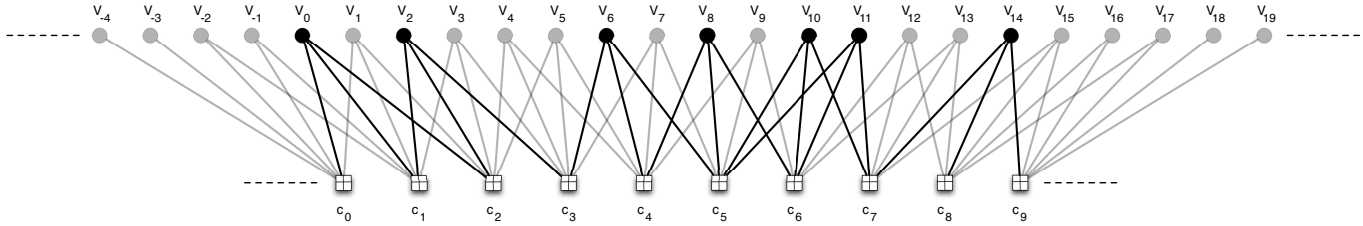


Fig. 3: A $(7,5)$ general trapping set in the infinite Tanner graph of a $(3,6)$-regular convolutional code.

Δ-trapping set number of a tail-biting convolutional code formed by terminating the associated convolutional code with unwrapping factor $\lambda > 0$. Then, for any $\Delta \geq 0$, the Δ-trapping set number $d_{ccts}(\Delta)$ of the unterminated LDPC convolutional code is bounded below by $d_{ts}^{(\lambda)}(\Delta)$ for any unwrapping factor λ, i.e.,

$$d_{ccts}(\Delta) \geq d_{ts}^{(\lambda)}(\Delta). \tag{6}$$

Proof. Consider the unterminated convolutional code whose parity-check matrix is given in (2). Now take any (a,b) general trapping set that consists of a set of variable nodes

$$\{v_{I(0)}, v_{I(1)}, \cdots, v_{I(a-1)}\}$$

and a set of odd-degree check nodes

$$\{c_{J(0)}, c_{J(1)}, \cdots, c_{J(b-1)}\}$$

in the induced subgraph. Here, $\mathbf{I} = (I(0), I(1), \cdots, I(a-1))$ and $\mathbf{J} = (J(0), J(1), \cdots, J(b-1))$ are index vectors of length a and b, respectively, and the elements of each vector are unique. Consider the modulo index vector $\hat{\mathbf{I}}$ (resp. $\hat{\mathbf{J}}$) obtained by letting $\hat{I}(i) = I(i) \bmod \lambda n_v$ for $i = 0, 1, \ldots, a-1$ (resp. $\hat{J}(j) = J(j) \bmod \lambda n_c$ for $j = 0, 1, \ldots, b-1$). This operation corresponds to wrapping the unterminated convolutional code to a tail-biting convolutional code with unwrapping factor λ (whose parity-check matrix is given in (5)) and yields an $(\hat{a}, \hat{b})$ general trapping set in the Tanner graph of the tail-biting code, where the values of $\hat{a}$ and $\hat{b}$ are determined below.

If the elements of the modulo index vectors $\hat{\mathbf{I}}$ and $\hat{\mathbf{J}}$ are still unique, then we say that the original (a,b) general trapping set has not been affected by the wrapping procedure and therefore $\hat{a}$ equals a and $\hat{b}$ equals b. If, however, there are $k > 1$ copies of the same index value in $\hat{\mathbf{I}}$ (resp. $\hat{\mathbf{J}}$), then the value of $\hat{a}$ (resp. $\hat{b}$) is reduced by $2\lfloor \frac{k}{2} \rfloor$. This corresponds to the case when multiple variable or check nodes in the trapping set are wrapped back on top of each other. If there is an even number of such nodes at the same position, none of them survive as a member of the tail-biting $(\hat{a}, \hat{b})$ trapping set, and if the repetition value k is odd, only one node survives. These reductions are cumulative over all the repetitive index values in $\hat{\mathbf{I}}$ and $\hat{\mathbf{J}}$.

This implies that a general trapping set in the unterminated convolutional code induces a general trapping set in any of its tail-biting terminated codes that has at most the same size as the original one. It follows that the smallest general trapping set size of an unterminated convolutional code is at least as large as the smallest general trapping set size of any of its tail-biting terminated convolutional codes, i.e., for any unwrapping factor λ. □

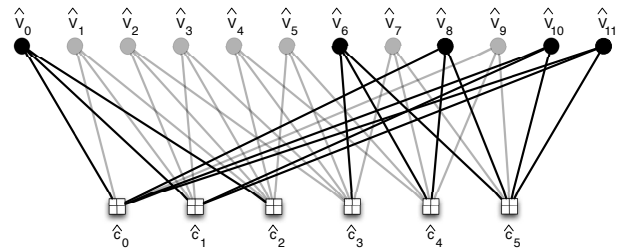


Fig. 4: A $(5,3)$ general trapping set in the Tanner graph of a $(3,6)$-regular tail-biting convolutional code with unwrapping factor $\lambda = 2$.

Example 1: Consider a sequence

$$\{v_0, v_1, \ldots, v_{15}\} = \{1,0,1,0,0,0,1,0,1,0,1,1,0,0,1,0\}$$

in a $(3,6)$-regular convolutional code. Suppose, without loss of generality, that $\mathbf{I} = (0,2,6,8,10,11,14)$. The seven ones in this sequence induce a subgraph with five odd-degree check nodes ($\mathbf{J} = (0,6,7,8,9)$), i.e., the index vectors $\mathbf{I}$ and $\mathbf{J}$ represent a $(7,5)$ general trapping set. This is illustrated in Figure 3. By wrapping the unterminated code onto itself modulo $2n_v = 12$ for variable nodes and modulo $2n_c = 6$ for check nodes, we obtain a sequence in the $(3,6)$-regular tail-biting convolutional code with unwrapping factor $\lambda = 2$. The resulting modulo 12 and 6 index vectors are $\hat{\mathbf{I}} = (0,2,6,8,10,11,2)$ and $\hat{\mathbf{J}} = (0,0,1,2,3)$. Observe that index values $\hat{I}(1) = \hat{I}(6)$, i.e., $k = 2$ for this index value, and we obtain the reduction value of $2\lfloor \frac{2}{2} \rfloor = 2$. Since k is even, the variable nodes corresponding to these index values do not appear in the reduced trapping set, and hence $\hat{\mathbf{I}}$ represents the set of variable nodes $\{\hat{v}_0, \hat{v}_6, \hat{v}_8, \hat{v}_{10}, \hat{v}_{11}\}$ in the Tanner graph of a $(3,6)$-regular tail-biting convolutional code and $\hat{a} = a - 2 = 5$. The same reduction in $\hat{b}$ is obtained as a result of $\hat{J}(0) = \hat{J}(1)$. Hence $\hat{b} = b - 2 = 3$, and the reduced set of check nodes represented by $\hat{\mathbf{J}}$ is $\{\hat{c}_1, \hat{c}_2, \hat{c}_3\}$. The corresponding sequence is

$$\begin{aligned}
&\{\hat{v}_0, \hat{v}_1, \ldots, \hat{v}_{11}\} \\
&\quad = \{(1+0), (0+0), (1+1), (0+0), 0, 0, 1, 0, 1, 0, 1, 1\} \\
&\quad \equiv \{1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1\} \bmod 2,
\end{aligned}$$

and it induces a subgraph in the tail-biting convolutional code graph with three odd-degree check nodes ($\hat{c}_1$, $\hat{c}_2$, and $\hat{c}_3$), i.e., the index vectors $\hat{\mathbf{I}}$ and $\hat{\mathbf{J}}$ represent a $(5,3)$ general trapping set in the Tanner graph of a $(3,6)$-regular tail-biting convolutional code, as illustrated in Figure 4. □

Corollary 2: Consider forming a family of unterminated, periodically time-varying LDPC convolutional codes with rate

$R = 1 - Nn_c/Nn_v = 1 - n_c/n_v$, constraint length $\nu_s = Nn_v$, and period $T = \lambda y$ from (5), as described in Section V.B. Let $d_{ccts}^{(\lambda)}(\Delta)$ be the Δ-trapping set number of the code with $T = \lambda y$, and let $d_{ccts}(\Delta) = \max_{\lambda > 0} d_{ccts}^{(\lambda)}(\Delta)$, which we call the Δ-trapping set number of the unterminated convolutional code family. Then, for any tail-biting termination with unwrapping factor λ, $d_{ccts}(\Delta)$ is bounded below by $d_{ts}^{(\lambda)}(\Delta)$ for any λ.

Proof. Replacing n_v with Nn_v and n_c with Nn_c in the proof of Theorem 1 implies that $d_{ccts}^{(\lambda)}(\Delta) \geq d_{ts}^{(\lambda)}(\Delta)$ for any λ. The result then follows directly from the definition of $d_{ccts}(\Delta)$.□

Intuitively, as λ increases, the tail-biting code becomes a better representation of its associated unterminated convolutional code, with $\lambda \to \infty$ corresponding to a non-periodically time-varying convolutional code. This is reflected in the average general trapping set enumerators, and it is shown in Section VI that increasing λ provides us with Δ-trapping set growth rates $\delta_{ts}^{(\lambda)}(\Delta)$ that converge to a lower bound on $\delta_{ccts}(\Delta)$, which we call the Δ-trapping set growth rate of the unterminated convolutional code family.

As noted in Section IV-B, the Δ-trapping set growth rate of a block code ensemble is defined as its Δ-trapping set number to block length ratio. For the protograph-based tail-biting LDPC convolutional codes defined in Section V-B, this ratio is therefore given as

$$\delta_{ts}^{(\lambda)}(\Delta) = \frac{d_{ts}^{(\lambda)}(\Delta)}{n} = \frac{d_{ts}^{(\lambda)}(\Delta)}{\lambda N n_v} = \frac{d_{ts}^{(\lambda)}(\Delta)}{\lambda \nu_s}. \tag{7}$$

Using (6), we then obtain

$$\delta_{ts}^{(\lambda)}(\Delta) \leq \frac{d_{ccts}^{(\lambda)}(\Delta)}{\lambda \nu_s} \leq \frac{d_{ccts}(\Delta)}{\lambda \nu_s}. \tag{8}$$

We note that, for convolutional codes, the length of the shortest codeword is equal to the encoding constraint length ν_e, which in general differs from the decoding constraint length ν_s. Assuming minimal encoder and syndrome former matrices, the relation between ν_e and ν_s can be expressed as

$$\nu_e = \frac{1-R}{R}\nu_s, \tag{9}$$

which implies that, for code rates less than $1/2$, the encoding constraint length is larger than the decoding constraint length, and vice versa for code rates greater than $1/2$.

Combining (8) and (9) gives us the desired lower bound

$$\delta_{ccts}(\Delta) \geq \frac{R}{1-R}\lambda \delta_{ts}^{(\lambda)}(\Delta), \tag{10}$$

where $\delta_{ccts}(\Delta) = d_{ccts}(\Delta)/\nu_e$ is the Δ-trapping set growth rate[5] of the unterminated convolutional code family.

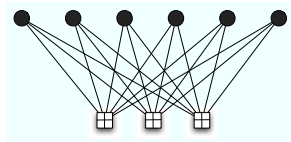
VI. Trapping set analysis

We now present a trapping set analysis for several regular, asymptotically good ensembles of unterminated, periodically time-varying LDPC convolutional codes. As described in Section V, we make use of ensembles of tail-biting LDPC convolutional codes to obtain a lower bound on the desired Δ-trapping set growth rate of the associated unterminated convolutional code family.

[5]If the syndrome former matrix is not in minimal form, (9) results in an upper bound on ν_e, which implies that $\delta_{ccts}(\Delta)$ is underestimated in this case.

A. Regular Ensembles with $gcd(n_c, n_v) > 1$

Example 2 (see also [1]): Consider a $(3,6)$-regular LDPC code with the folowing protograph and associated parity-check matrix:



$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

A family of rate $R = 1/2$ tail-biting LDPC convolutional code ensembles can be generated according to the displayed cut. We now proceed to calculate the Δ-trapping set growth rate $\delta_{ts}^{(\lambda)}(\Delta)$ for the modified tail-biting convolutional code ensembles with base parity-check matrices $\tilde{P}_{tb}^{(\lambda)}$ for various fixed values of Δ and increasing values of the unwrapping factor λ. Note that setting $\Delta = \beta/\alpha = 0$ corresponds to the minimum distance growth rate problem [8]. Thus, for $\lambda = 1$, which corresponds to the $(3,6)$-regular block code ensemble, $\delta_{ts}^{(1)}(0) = \delta_{min} = 0.023$, where δ_{min} is the minimum distance growth rate for the $(3,6)$-regular ensemble, as originally calculated by Gallager [16]. Further, for larger values of λ, the value for $\delta_{ts}^{(\lambda)}(0)$ agrees with the earlier results for minimum distance growth rates of tail-biting convolutional codes given in [8].

As Δ ranges from 0 to ∞, the points $(\delta_{ts}^{(\lambda)}(\Delta), \Delta \cdot \delta_{ts}^{(\lambda)}(\Delta))$ trace out the so-called *zero-contour curve* for a protograph-based block code ensemble [10]. Note that we are particularly interested in small values of Δ, since this means that the ratio of odd-degree check nodes to variable nodes inducing them is small. This implies that there are relatively few odd-degree check nodes available to pass messages to the large number of variable nodes. The zero-contour curves for Example 2 are shown in Figure 5, and the Δ-trapping set growth rates are highlighted for $\Delta = 0.02$.

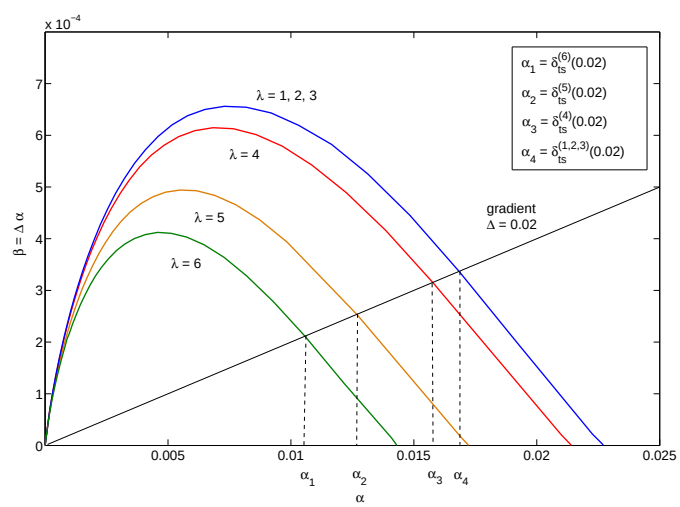


Fig. 5: Zero-contour curves for Example 2.

The zero-contour curve is key to understanding the role of trapping sets in iterative decoding. Code ensembles with large Δ-trapping set numbers $d_{ts}^{(\lambda)}(\Delta)$ are of primary interest, since small trapping sets dominate iterative decoding performance in the error floor [3]. Thus we want the Δ-trapping set growth

rate $\delta_{ts}^{(\lambda)}(\Delta)$ to exist and be as large as possible for each value of Δ. We observe in Fig. 5 that $\delta_{ts}^{(\lambda_1)}(\Delta) \leq \delta_{ts}^{(\lambda_2)}(\Delta)$ for any $\lambda_1 > \lambda_2$. This is analogous to the decrease in the minimum distance growth rate with increasing λ observed in [8]. If a zero-contour curve of ensemble A is always below the zero-contour curve of ensemble B, then, in general, we would expect a code drawn from ensemble A to exhibit poorer error floor performance than one drawn from ensemble B. Thus we would expect worse error floor performance with increasing λ for the tail-biting convolutional code ensembles.[6] For this (3,6)-regular ensemble, the lower bound on $\delta_{ccts}(\Delta)$ is simply $\delta_{ccts}(\Delta) \geq \frac{R}{1-R}\lambda\delta_{ts}^{(\lambda)}(\Delta) = \lambda\delta_{ts}^{(\lambda)}$, since $\frac{R}{1-R} = 1$ in this case. The Δ-trapping set growth rates for Example 2 are plotted in Fig. 6 for $\Delta = 0$, 0.01, and 0.05.

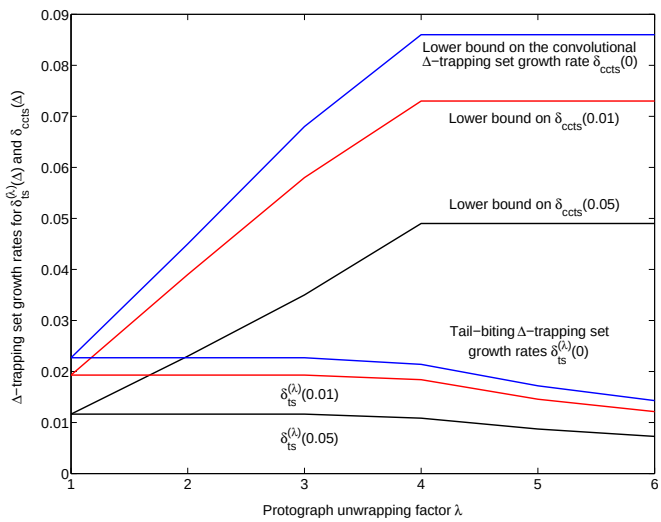


Fig. 6: Δ-trapping set growth rates for Example 2.

We observe that, once the unwrapping factor λ of the tail-biting convolutional code ensemble exceeds 3, the lower bound on $\delta_{ccts}(\Delta)$ levels off for each distinct value of Δ. We also observe a significant increase in the value of $\delta_{ccts}(\Delta)$ compared to $\delta_{ts}^{(1)}(\Delta)$, the Δ-trapping set growth rate of the underlying block code ensemble. □

Example 3: Consider the rate $R = 2/3$, $(3, 9)$-regular LDPC code ensemble. We form a protograph in the usual fashion, creating 3 check nodes, each of which connect to all 9 variable nodes, and we observe that $\gcd(3, 9) = 3$. The protograph parity-check matrix and defined cut are displayed below:

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

For this rate $R = 2/3$ ensemble, the lower bound on $\delta_{ccts}(\Delta)$ is $\delta_{ccts}(\Delta) \geq \frac{R}{1-R}\lambda\delta_{ts}^{(\lambda)}(\Delta) = 2\lambda\delta_{ts}^{(\lambda)}(\Delta)$. We observe that, as in Example 2, the Δ-trapping set growth rates calculated for fixed values of Δ with increasing λ provide us with a lower bound on the convolutional Δ-trapping set growth rate, which exceeds the value of $\delta_{ts}^{(1)}(\Delta)$, the Δ-trapping set growth rate of the protograph-based LDPC block code ensemble. The bounds calculated for several values of Δ are given below in Table 1. □

Δ	$\delta_{ts}^{(1)}(\Delta)$	lower bound on $\delta_{ccts}(\Delta)$
0	0.0054	0.037
0.01	0.0046	0.031
0.05	0.0027	0.018
0.10	0.0014	0.010

Table 1: Block code Δ-trapping set growth rates and lower bounds on the convolutional Δ-trapping set growth rates for several values of Δ for the regular $(3, 9)$-ensemble.

B. Regular Ensembles with $gcd(n_c, n_v) = 1$

For the following $R = 1/4$, $(3, 4)$-regular and $R = 2/5$, $(3, 5)$-regular ensembles, we form the trivial 'all-ones' protograph parity-check matrix of size $n_c \times n_v$. The protograph is then unwrapped according to the nonuniform cutting vector ξ method as described in Section III-A. The nonuniform cuts chosen are consistent with those previously presented in [9], and they are displayed in Fig. 7.

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(a) (b)

Fig. 7: Protograph parity-check matrices with associated nonuniform cut for (a) the $(3, 4)$-regular ensemble and (b) the $(3, 5)$-regular ensemble.

Just as we expect the tail-biting ensemble zero-contour curve values $\delta_{ts}^{(\lambda)}(0)$ to correspond to minimum distance growth rates of the associated block code ensembles, we expect that the lower bound on the free distance growth rate δ_{free} of the regular convolutional codes (found in [9]) based on the displayed cuts correspond to the value $\delta_{ccts}(0)$ of the convolutional lower bound zero-contour curve. This can be seen in Figure 8, where the lower bounds on $\delta_{ccts}(\Delta) = \alpha$ are plotted against $\Delta\alpha = \beta$ to form a convolutional lower bound zero-contour curve for each of the regular convolutional ensembles considered.

We observe that, in general, as the rate increases the convolutional lower bound zero-contour curves become lower, indicating the likelihood of worse error-floor performance. This is consistent with the conclusions drawn from the free distance bounds previously presented in [9].

VII. Conclusions

In this paper, asymptotic methods were used to calculate a lower bound on the Δ-trapping set number that grows linearly with constraint length for several regular, asymptotically good ensembles of unterminated, protograph-based, periodically time varying LDPC convolutional codes. It was shown that the Δ-trapping set growth rates of the LDPC convolutional code ensembles exceed the growth rates of the corresponding LDPC

[6] We observe from the zero-contour curves of Example 2 that increasing λ results in smaller Δ-trapping set growth rates for $\lambda \geq 3$. However, we must be careful in this case to remember that the block lengths also increase and the Δ-trapping set number is $d_{ts}^{(\lambda)}(\Delta) = n\delta_{ts}^{(\lambda)}(\Delta) = N\lambda n_v\delta_{ts}^{(\lambda)}(\Delta)$.

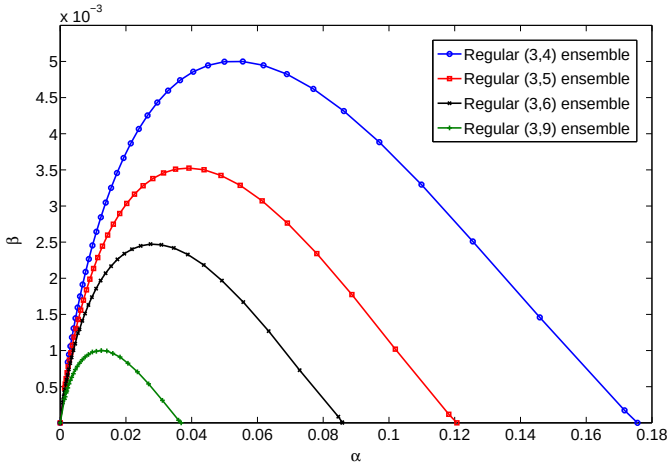


Fig. 8: Lower bounds on $\delta_{ccts}(\Delta)$ for several regular ensembles.

block code ensembles on which they are based. These large trapping set growth rates suggest that the LDPC convolutional codes will exhibit good iterative decoding performance in the error floor.

Acknowledgements

This work was partially supported by NSF Grants CCF05-15012 and CCF08-30650 and NASA Grant NNX07AK536. In addition, the authors acknowledge the support of the Scottish Funding Council for the Joint Research Institute with the Heriot-Watt University, which is a part of the Edinburgh Research Partnership. The authors would also like to thank Prof. K. Sh. Zigangirov and Prof. N. Goertz for their valuable input.

References

[1] D. G. M. Mitchell, A. E. Pusane, and D. J. Costello, Jr., "Trapping set analysis of protograph-based LDPC convolutional codes", *2009 IEEE Int. Symp. on Inform. Theory*, (submitted).

[2] D. J. C. MacKay and M. S. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes", *Elec. Not. in Theoretical Comp. Sci.*, Vol. 74, 2003.

[3] T. Richardson, "Error-floors of LDPC codes", *Proc. Forty-first Annual Allerton Conference*, Allerton, IL, Sep. 2003.

[4] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles", *IEEE Trans. on Inform. Theory*, IT-53, pp.39-55, Jan. 2007.

[5] A. Jiménez-Felström and K. Sh. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrices", *IEEE Trans. Inform. Theory*, IT-45, pp.2181-2191, Sept. 1999.

[6] D. J. Costello, Jr., A. E. Pusane, S. Bates, and K. Sh. Zigangirov, "A comparison between LDPC block and convolutional codes", *Proc. Inform. Theory and App. Workshop*, San Diego, CA, Feb. 2006.

[7] D. J. Costello, Jr., A. E. Pusane, C. R. Jones, and D. Divsalar, "A comparison of ARA- and protograph-based LDPC block and convolutional codes", *Proc. Inform. Theory and App. Workshop*, San Diego, CA, Feb. 2007.

[8] D. G. M. Mitchell, A. E. Pusane, K. Sh. Zigangirov, and D. J. Costello, Jr., "Asymptotically good LDPC convolutional codes based on protographs", *Proc. IEEE Int. Symp. on Inform. Theory*, Toronto, Canada, July 2008.

[9] D. G. M. Mitchell, A. E. Pusane, N. Goertz, and D. J. Costello, Jr., "Free distance bounds for protograph-based regular LDPC convolutional codes", *Proc. Int. Symp. on Turbo Codes and Related Topics*, Lausanne, Switzerland, Aug. 2008.

[10] S. Abu-Surra, W. E. Ryan, and D. Divsalar, "Ensemble trapping set enumerators for protograph-based LDPC codes", *Proc. Forty-fifth Annual Allerton Conference*, Allerton, IL, Sep. 2007.

[11] M. Tavares, K. Sh. Zigangirov, and G. Fettweis, "Tail-Biting LDPC Convolutional Codes", *Proc. IEEE Int. Symp. on Inform. Theory*, Nice, France, June 2007.

[12] R. Smarandache, A. E. Pusane, P. O. Vontobel, and D. J. Costello, Jr., "Pseudo-codeword analysis for LDPC convolutional codes", *IEEE Trans. on Inform. Theory* (accepted for publication). Available online at `http://arxiv.org/abs/cs/0609148v1`.

[13] J. Thorpe, "Low-Density Parity-Check (LDPC) codes constructed from protographs", *JPL INP Progress Report*, Vol. 42-154, Aug. 2003.

[14] D. Divsalar, "Ensemble weight enumerators for protograph LDPC codes", *Proc. IEEE Int. Symp. on Inform. Theory*, Seattle, WA, Jul. 2006.

[15] S. L. Fogal, R. McEliece, and J. Thorpe, "Enumerators for protograph ensembles of LDPC codes", *Proc. IEEE Int. Symp. on Inform. Theory*, Adelaide, Australia, Sept. 2005.

[16] R. G. Gallager, "Low-density parity-check codes", *IRE Trans. Inform. Theory*, IT-8: 21-28, Jan. 1962.