On the Exhaustion and Elimination of Trapping Sets: Algorithms & The Suppressing Effect

Chih-Chun Wang
Center for Wireless Systems and Applications (CWSA)
School of ECE, Purdue University, West Lafayette, IN 47906
chihw@purdue.edu

***Abstract*— This paper studies a systematic treatment of trapping sets in finite-length LDPC codes and its related theoretic properties. It is proven that the complexity of deciding the minimal trapping distance is NP-complete. Furthermore, exhausting minimal trapping sets can be achieved by using any good stopping set exhaustion algorithm as a building block. The *suppressing effect* of cyclic lifting for trapping sets is also studied in this work, which characterizes the probability that a base code trapping sets *survives* after applying the cyclic lifting technique. The corresponding quantitative knowledge about the origin of small trapping sets in the cyclically lifted codes helps provide definite guidelines for the base code optimization to lower the error-floor over non-erasure channels. Extensive numerical experiments are provided to demonstrate the various techniques discussed in this work.**

I. Introduction

Analogous to the stopping sets (SSs) [1] for binary erasure channels (BEC), the error floor performance of arbitrary, fixed, finite-length, LDPC codes over non-erasure channels is dominated by the error bits forming a special substructure in the corresponding Tanner graph representation, which is termed differently under various motivations including trapping sets (TSs) [2], near-codewords [3], pseudo-codewords [4], and instantons [5]. In this work, we study systematically the TS exhaustion, elimination, and the associated theoretical properties, ranging from the complexity analysis and efficient implementation of TS exhaustion to the *suppressing effect* for the cyclically lifted (CL) code [2], [6].

In contrast with the graph-theoretic definition of SSs, the definition of TS is purely operational. Therefore, a concrete definition is necessary before any theoretical discussion. We define and classify the TS by its corresponding numbers of degree one check nodes and show that deciding the minimal size of TSs in any given class is NP-complete, similar to the complexity of deciding the minimal stopping distance [7]. As an asymptotic result, the NP-completeness does not prevent relatively efficient algorithms for problem of small finite sizes. Since the delay constraint of practical communications generally requires the codeword length to be pretty small $n = 500$–2000, one particularly interesting subject is to design a good algorithm that is efficient enough for exhausting minimal TSs for small n using reasonable amount of resources. In this work, a good TS exhaustion algorithm is constructed using the SS exhaustion algorithm in [8] as a building block, and is capable of exhausting minimal TSs for codes of length $n \approx 500$, ten times larger than the searchable range of the best existing algorithms of this type. Our results serve as a starting point of designing even better algorithms for codes of length $n \approx 2000$. Other existing results focus on non-exhaustively enumerating error-prone patterns [2], [9], [10].

In [11], the *suppressing effect* of cyclic lifting is identified for the first time, which characterizes the probability that a base code SSs *survives* after applying the cyclic lifting technique. Quantifying the suppressing effect for SSs leads to an analytic expression of the ensemble BEC error floor for CL codes with variable lengths, which were previously computable only for the simplest graph ensemble [1]. The suppressing effect suggests the use of the *suppressing weight* $W_{\sup} = 0.5\#E - \#V + \#C_{\mathrm{odd},\geq 3}$ as an objective function for base-code optimization, where $\#E$, $\#V$, and $\#C_{\mathrm{odd},\geq 3}$ are the numbers of edges, variable nodes, and check nodes with odd, ≥ 3 degrees of any SSs in the base code. This work generalizes the suppressing effect for TSs and shows that the suppressing weight $W_{\sup}$ is indeed a critical performance measure when both the SSs and TSs are taken into accounts. Extensive numerical experiments are then conducted to demonstrate the exhaustion and elimination of TSs based on the techniques studied herein. For demonstration, we also construct a short code $n = 512$ with minimal 1-out trapping distance 10, the definition of which is relegated to Section II. The superior efficiency of our algorithms is illustrated by the fact that it requires 3.1×10^{20} trials to determine the minimal 1-out trapping distance by a brute force search. The error correcting performance of our code is then compared to the irregular PEG code in [12]. Other related work for the finite code analysis and construction can be found in [12], [13], [14] and the references therein.

II. Definitions

Let $G = (V, C, E)$ denote the bipartite Tanner graph of any given LDPC code with an $m \times n$ parity-check matrix H, where all valid codewords, represented by a column vector x, satisfy $Hx = 0$. The letters G and H are sometimes used interchangeably for the Tanner graph and for the parity-check matrix since the mapping between them are bijective.

The notion of TSs stems from an operational definition referring to a subset of $V = \{v_1, \ldots, v_n\}$ that is "susceptible to errors under message-passing decoding," thus depends on

1-4244-1429-6/07/$25.00 ©2007 IEEE

Algorithm 1 Reduction from SD(H,t) to k-OTD(H,t)

1: **Input:** the parity-check matrix H and t.
2: Construct the corresponding graph G from H.
3: Construct G' by duplicating $(k+2)$ copies of G, and denote the $(k+2)n$ var. nodes as $\{v_{i,l} : \forall i = 1,\ldots,n, \forall l = 1,\ldots,(k+2)\}$.
4: **for** $i = 1$ to n **do**
5: Form a $(k+2)$-clique among all the var. node copies $\{v_{i,l}\}_l$ by adding $(k+1)(k+2)/2$ new edges.
6: Break each new edge into two by inserting a new check nodes of degree 2 in between.
7: **end for**
8: Denote the final graph as G''.
9: **for** $i = 1$ to n **do**
10: Construct $G^{(i)}$ from G'' by adding k check nodes of degree 1, and connecting them to $v_{i,1}$.
11: Let $H^{(i)}$ denote the parity check matrix of $G^{(i)}$.
12: **end for**
13: **Output:** If k-OTD$(H^{(i)}, t(k+2))$ outputs 1 for any i, then output 1. Otherwise, output 0.

Algorithm 2 A minimum k-OTS exhaustion algorithm

1: **Input:** k, H, and the corresponding $G = (V, C, E)$.
2: **Initialization:** $\mathbf{T}_{\min,k} \leftarrow \{V\}$.
3: **repeat**
4: Based on G, select k edges incident to k distinct check nodes and denote them as $(v_{i_1}, c_{j_1}), \cdots, (v_{i_k}, c_{j_k})$.
5: **if** there is no edge between $v_{i_{l_1}}$ and $c_{j_{l_2}}$ for any $l_1 \neq l_2$, **then**
6: Construct a new H' from H by removing those columns i in H simultaneously satisfying (1) $i \neq i_l, \forall l = 1, \cdots, k$, and (2) $\exists l \in \{1, \cdots, k\}$ such that $H_{j_l i} = 1$.
7: Construct a new H'' from H' by removing rows $j_l, \forall l \in \{1, \cdots, k\}$.
8: Let $\mathbf{v} \leftarrow \{v_{i_l} : \forall l \in \{1, \cdots, k\}\}$
9: $\mathbf{S}_{\text{temp}} \longleftarrow$ MSSE$(H'', \mathbf{v})$.
10: **if** $\mathsf{dist}(\mathbf{S}_{\text{temp}}) < \mathsf{dist}(\mathbf{T}_{\min,k})$ **then**
11: $\mathbf{T}_{\min,k} \leftarrow \mathbf{S}_{\text{temp}}$.
12: **else if** $\mathsf{dist}(\mathbf{S}_{\text{temp}}) = \mathsf{dist}(\mathbf{T}_{\min,k})$ **then**
13: $\mathbf{T}_{\min,k} \leftarrow \mathbf{T}_{\min,k} \cup \mathbf{S}_{\text{temp}}$.
14: **end if**
15: **end if**
16: **until** all possible selections of k distinct edges are exhausted.
17: **Output:** $\mathbf{T}_{\min,k}$.

the underlying channel model and the decoding algorithm [2]. Empirically, almost all TSs are *near-codewords* [3].

We now redefine the k-out TSs from a pure graph-theoretic perspective, which is independent of the underlying channel model and the decoding algorithm.

Definition 1 (k-out TSs): A subset $\mathbf{t} \subseteq V$ is a k-out TS if in the sub-graph induced by $\mathbf{t}$, there are exactly k check nodes of degree one. A SS is simply a 0-out TS.

The definition of the k-out TS is different from "near-codeword" [3] in the following way. A near-codeword focuses on the number of check nodes of *odd degrees*, and thus emphasizes its resemblance to a valid codeword. A k-out TS focuses on how many extrinsic messages can be used to break the loopy behavior and recover the "trapped" errors, which stresses its connection to a valid SS. All near-codewords are k-out TSs for some small k, so removing minimal k-out TSs for small k will in turn remove small near-codewords. The error floor performance is thus improved.

III. Exhaustively Searching For Trapping Sets

A. Complexity

We consider the complexity of deciding the minimal k-out trapping distance (k-OTD) of a given G (or equivalently H), which is defined as the minimal size of non-empty k-out TSs and is denoted by $d_{\mathrm{t},k}$. With the input being an arbitrary parity-check matrix H and an integer t, the decision problem "k-OTD(H,t)" proposes the question whether the corresponding $d_{\mathrm{t},k} \leq t$.

Theorem 1: For any fixed integer k, the decision problem k-OTD(H,t) is NP-complete.

Sketch of the proof: We prove Theorem 1 by reduction from the minimal stopping distance decision problem SD(H,t), which is NP-complete [7]. The corresponding poly-time reduction is described in Algorithm 1. The correctness of this reduction can be proved by showing that for any i, j, l_0, if v_{i,l_0} is in a k-out TS of $G^{(j)}$, then $\{v_{i,l}\}_{l=1,\cdots,k+2}$ must be in the same k-out TS. Otherwise, the min-cut max-flow theorem guarantees that there are $\geq k+1$ check nodes of degree 1. From the above reasoning, any k-out TS must be a parallel collection of variable nodes in $G^{(j)}$. For any k-out TS that is a parallel collection of variable nodes, any check node of degree 1 in the parallel collection of check nodes will result in a total number of $k+2$ check nodes of degree 1, which is again impossible. So the k check nodes of degree 1 in a k-out TS must be the additional check nodes added in Line 10. Projecting the k-out TSs of $G^{(i)}$ on the original graph G gives us a SS. The converse direction that any SS in G leads to a k-out TS in $G^{(i)}$ is straightforward.

■

B. Exhaustion Algorithm

In [8], we have devised a very good minimal SS exhaustion algorithm: MSSE$(H, \mathbf{v})$, which takes input of an arbitrary parity-check matrix H and an arbitrary set of variable nodes $\mathbf{v} \subseteq V$. The output of MSSE$(H, \mathbf{v})$ is an exhaustive list of smallest non-empty SSs $\mathbf{S}_{\min} = \{\mathbf{s}_i\}$ such that $\mathbf{v} \subseteq \mathbf{s}_i$. We use $\mathsf{dist}(\mathbf{S}_{\min}) \triangleq |\mathbf{s}_i|$ to denote the corresponding minimal stopping distance.[1] The MSSE$(H, \mathbf{v})$ will then serve as a building block of the minimal k-out TS exhaustion algorithm depicted in Algorithm 2. The correctness of Algorithm 2 is self-explanatory from the definition of k-out TSs, in which the k selected edges in Line 4 are exactly the k extrinsic edges incident to check nodes of degree 1 and any SS in H'' is thus a k-out TS in H.

The complexity of Algorithm 2 grows up in the order of $\mathcal{O}\left(n^k\right)$, which makes it less appealing for cases of large k. Using the good MSSE$(H, \mathbf{v})$ proposed in [8] and a personal computer, the minimal k-out TS can be exhausted for practical codeword length $n \approx 500$ for the cases of $k = 0, 1, 2$. Since the k-out TSs with $k \leq 2$ generally contribute to more than 50% of the error floor [Figure 5, [2]], Algorithm 2 is a successful first attempt on this asymptotically NP-complete exhaustion problem for finite, practical sizes of n.

[1] Here the minimal stopping distance is with respect to the variable nodes $\mathbf{v}$. When $\mathbf{v} = \emptyset$, $\mathsf{dist}(\mathbf{S}_{\min})$ is the minimal stopping distance of the entire code.

Remark: For large k, exhausting k-out TSs becomes tricky when the minimum variable node degree is $\geq k$. For a node v_0 with $\deg(v_0) = k$, the single node collection $\{v_0\}$ itself is a trivial k-out TS and would thus dominate the output of Algorithm 2. A simple remedy is to carefully select the $\{(v_{i_1}, c_{j_1}), \cdots, (v_{i_k}, c_{j_k})\}$ of interest in Line 4 and preclude uninteresting combinations. A more effective method would require unambiguous definitions of 'non-trivial" k-out TSs and is currently under investigation.

C. Exhausting k-out TSs in Algebraically Constructed Codes

We use the following well-studied codes to demonstrate Algorithm 2: the (155,64,20) Tanner code [15], the Ramanujan-Margulis (2184,1092) code with $q = 13$ and $p = 5$ [16], and the (672,336) Margulis code with $p = 7$ [16]. The results for $k = 0, 1, 2$ are summarized in Table I, in which the minimal stopping distance is the minimal 0-out trapping distance. Algorithm 2 either returns an exhaustive list of minimal SSs/TSs or provide a rigorous lower bound d on the minimal distances by exhausting all combinations of $<d$ variable nodes. The upper bound is provided by enumerating at least one instance [10].

In addition to generating rigorous lower bounds on the minimal distances of these algebraically constructed codes for the first time, at least two new important implications can be made as follows.

1) The (155,64,20) Tanner Code: We are able to exhaustively locate all 465 minimal 2-out TSs of size 8, all of which can be described by the following 5 representatives using the automorphisms discussed in [15].

$$7, 17, 19, 33, 66, 76, 128, 140$$
$$7, 31, 33, 37, 44, 65, 100, 120$$
$$1, 19, 63, 66, 105, 118, 121, 140$$
$$44, 61, 65, 73, 87, 98, 137, 146$$
$$31, 32, 37, 94, 100, 142, 147, 148.$$

Recently, the instanton analysis in [5] identified some dominating error patterns, termed as instantons. Each of the reported instantons contains one aforementioned minimal 2-out TS as a substructure. These results again confirm that the dominating error patterns are generally k-out TSs with small k.

2) The Ramanujan-Margulis (2184,1092) Code w. $q = 13, p = 5$: MacKay *et al.* [3] and Hu *et al.* [10] identified independently 1092 valid codewords of size 14 via algebraic arguments and via the error impulse method respectively. The remaining question is whether there is any other codeword of equal or smaller sizes. Algorithm 2 shows that there are "only" 1092 SSs of size 14 and there is no SS of smaller size. Since any codeword must be a SS, the minimal Hamming distance of the Ramanujan-Margulis (2184,1092) code must be 14 and its multiplicity must be 1092. What MacKay *et al.* and Hu *et al.* found are indeed the entire collection of minimal codewords.

	Tanner(155,64)	R-M(2184,1092)	Marg(672,336)
Stopping Dist.			
by enum.	$(\leq 18, ?)$	$(\leq 14, ?)$	$(\leq 16, ?)$
by exhaust'n	$(\geq 13, ?)$	$(14, 1092)$	$(\geq 14, ?)$
1-Out Trap. Dist.			
by exhaust'n	$(\geq 12, ?)$	$(\geq 13, ?)$	$(\geq 13, ?)$
2-Out Trap. Dist.			
by exhaust'n	$(8, 465)$	$(\geq 10, ?)$	$(\geq 10, ?)$

TABLE I

Lower and upper bounds on the minimal stopping, 1-out, and 2-out trapping distances. The pair (d, m) in each entry represents the minimal distance (or the range of the minimal distance) and how many SSs/TSs are of weight d.

IV. The Suppressing Effects

A. Preliminary

Cyclic lifting is a technique of generating bigger graphs/codes from a small base graph/code, of which the advantages include compact description, efficient encoding and decoding implementation, and reported strong error-correcting performance [2]. Starting from a base $m \times n$ parity check matrix H and a lifting factor $K \in \mathbb{N}$, bigger graphs can be constructed by replacing zero entries of H by $K \times K$ zero matrices and each non-zero entry by a $K \times K$ cyclic permutation matrix, which results in a new $mK \times nK$ parity check matrix H'. The cyclically lifted (CL) code ensemble is obtained by assuming the cyclic permutation matrices are drawn uniformly randomly. In many cases, the CL code has a much less number of SSs/TSs compared to that of its base code. It is as if the existing SSs/TSs in the base code is *suppressed* during the lifting procedure.

We consider the problem that given a bad error-prone substructure in the base code, what is the probability that the same bad substructure remains in the CL code ensemble. Or equivalently, how effective is the CL construction in terms of suppressing bad substructures in the base code. The survival probability of a bad structure under cyclic lifting is thus termed the *suppressing effect*.

The suppressing effect of SSs analytically quantifies the ensemble-averaged BEC error floor [11]. For non-erasure channels, the error floor cannot be determined by the minimal TSs, but the order and the multiplicity of minimal TSs are still very important metrics gauging the error correcting capability in the high SNR regime. For the following, we generalize the suppressing effect for k-out TSs.

B. Different Types of Suppressing Effects

Following the construction described in the previous subsection, let G_L denote the lifted code obtained from a base code G_B with lifting factor K. To be more explicit, for any i, the variable nodes $v_{(i-1)K+1}, \cdots, v_{iK}$ in G_L corresponds to the i-th variable node in G_B. For any k_L-out TS $\mathbf{t}_L$ of G_L, rename all $v_i \in \mathbf{t}_L$ by $v_{\lfloor \frac{i-1}{K} \rfloor + 1}$ and use $\mathbf{t}_{L/K}$ to denote the result after this conversion. $\mathbf{t}_B$ is constructed from $\mathbf{t}_{L/K}$ by removing the repeated nodes in $\mathbf{t}_{L/K}$. We then have

Lemma 1: $\mathbf{t}_B$ is a k_B-out TS of G_B for some $k_B \leq k_L$.

Lemma 1 prompts two types of suppressing effects: the direct and the cross suppressing effects. The former is the cases in which $k_B = k_L$ while the latter is for $k_B < k_L$. In sum, any SS (0-out TS) of the lifted code must be resulted from a base-code SS, but a 1-out TS in the lifted code may be resulted either from a 1-out TS or from a SS of the base code. If $\mathbf{t}_{L/K}$ contains no repeated nodes, we define such t_L as the first order survival of $\mathbf{t}_B$ and the corresponding survival probability is the *first order suppressing effect*. The high order suppressing effect is defined as the survival probability of $\mathbf{t}_L$ that is not the first order survival. Both the first order and the high order suppressing effects for SSs are discussed quantitatively in [11].

In this work, only the first order suppressing effect is considered due to the following two reasons. First, the repeated nodes in $\mathbf{t}_{L/B}$ imply that the size of such $\mathbf{t}_L$ is strictly larger than that of their first-order counterpart for the same $\mathbf{t}_B$ and thus has less influence on the error floor. Second, $\mathbf{t}_L$ with repeated nodes in $\mathbf{t}_{L/B}$ is much rare due to the additional graph-based constraints [11]. In all our experiments of exhausting minimal TSs in CL codes, we have never seen a single instance of $\mathbf{t}_L$ with repeated nodes in $\mathbf{t}_{L/B}$.

The First Order, Direct Suppressing Effect

Theorem 2: Define the following auxiliary function:

$$f(K,d) = \sum_{t=0}^{\min(K,d)} (-1)^t \binom{d}{t}\binom{K}{t} t!(K-t)^{d-t}. \quad (1)$$

For any base-code k-out TS $\mathbf{t}_B$, let $\mathbf{T}_L$ denote the collection of all first order survivals $\mathbf{t}_L$ of $\mathbf{t}_B$ that are also k-out TSs. We then have

$$\begin{aligned}\mathsf{E}\{|\mathbf{T}_L|\} &= K^k K^{\#V-\#E} \prod_{c_j:\mathsf{deg}(c_j)\geq 2} f(K,\mathsf{deg}(c_j)) \\ &\propto K^{0.5k} K^{-(0.5\#E-\#V+0.5\#C_{\mathrm{odd},\geq 3})}, \quad (2)\end{aligned}$$

where $\#V$, $\#E$ are the numbers of variable nodes and edges in the subgraph induced by $\mathbf{t}_B$, while c_j and $\mathsf{deg}(c_j)$ represent the individual check node and its degree. $\#C_{\mathrm{odd},\geq 3}$ is the number of odd degree check nodes of degrees ≥ 3.

Theorem 2 quantifies the first order direct suppressing effect and implies that for a SS and a TS of the same values of $\#V$ and $\#E$, the latter is comparably harder to suppress than the former due to the multiplication term $K^{0.5k}$ in (2).

Corollary 1: Consider any G_B in which all variable nodes are of degree d_v. To ensure that random cyclic lifting can take care of all small k-out TSs automatically, or equivalently, to ensure (2) goes to zero as K tends to infinity, G_B must satisfy one of the following conditions depending on the value of d_v.

1) If $d_v = 2$, then the base-code optimization has to remove as many base-code SSs as possible.
2) If $d_v = 3$, "(2)$\rightarrow 0$" implies that G_B cannot have any 2-cycle and cannot have any 4-cycle.
3) If $d_v \geq 4$, "(2)$\rightarrow 0$" implies that G_B cannot have any 2-cycle. Nonetheless, 4-cycles are tolerable in this case.

Sketch of the proof: Substitute $\#E = d_v \#V$ into (2), and discuss the minimal values of $\#V$ for any k-out TS with $k \leq d_v - 1$. This corollary can then be obtained straightforward. ■

The First Order Cross Suppressing Effect

Here we investigate the first order cross suppressing effect from a k-out base-code TS $\mathbf{t}_B$ to a $(k+1)$-out lifted-code TS $\mathbf{t}_L$. The same principles are applicable to the cases in which $k_L - k_B > 1$.

Theorem 3: In addition to the auxiliary function $f(K,d)$ in (1), define another auxiliary function $g(K,d)$:

$$g(K,d) = \sum_{t=1}^{\min(K,d)} (-1)^{t-1} t \binom{d}{t}\binom{K}{t} t!(K-t)^{d-t}.$$

Given any base-code k-out TS $\mathbf{t}_B$, let $\mathbf{T}_L^{+1}$ denote the collection of all first order survivals $\mathbf{t}_L$ of $\mathbf{t}_B$ that are $(k+1)$-out TSs. We then have

$$\begin{aligned}\mathsf{E}\{|\mathbf{T}_L^{+1}|\} &= K^k K^{\#V-\#E} \\ &\times \prod_{c_j:\mathsf{deg}(c_j)\geq 2} f(K,\mathsf{deg}(c_j)) \left(\sum_{c_j:\mathsf{deg}(c_j)\geq 2} \frac{g(K,\mathsf{deg}(c_j))}{f(K,\mathsf{deg}(c_j))} \right) \\ &\propto K^{0.5k} K^{-(0.5\#E-\#V+0.5\#C_{\mathrm{odd},\geq 3})} \\ &\times (K\#C_{\mathrm{odd},\geq 3} + \#C_{\mathrm{even},\geq 4}),\end{aligned}$$

where $\#C_{\mathrm{even},\geq 4}$ denote the total number of even degree check nodes of degree ≥ 4.

We notice that the combinatorial bias of forming a SS/TS ensures that even for a base code of medium size, almost all SSs/TSs have only check nodes of degrees 1 or 2, as observed in [17], [11]. Theorem 3 shows that for those SSs/TSs, there will be no survival of the "+1" cross suppressing effect. Consequently, similar to the high-order survival, the survival of the cross suppressing effect is very rare.

Contrary to the minimum stopping/trapping distance analysis provided herein, a related approach analyzing the codeword weight distribution of the cyclically lifted protograph code is considered in [6].

V. Numerical Results

The analysis of the direct and the cross suppressing effects shows that the suppressing weight, defined as $W_{\mathrm{sup}} = 0.5\#E - \#V + 0.5\#C_{\mathrm{odd},\geq 3}$, plays a central role in the base-code optimization for both the SS and the TS elimination. Using the *code annealing* algorithm in [11] to maximize the minimal W_{sup} of the base code, we construct a CL code of $n = 512$ from a base code of size 128 and degree distributions $\lambda(x) = 0.320x + 0.276x^2 + 0.0145x^5 + 0.390x^6$ and $\rho(x) = 0.5095x^5 + 0.4915x^6$. The corresponding minimal distances and the multiplicities for SSs and 1-out TSs are $(11, 12)$ and $(10, 24)$ respectively. Its performance is compared to codes of the same degree distributions in Fig. 1, including a random code and a code optimized for BECs with the minimal stopping distance being 13. As expected, by jointly optimizing both the stopping and the trapping distances, the error floor over additive white Gaussian channel is further lowered.

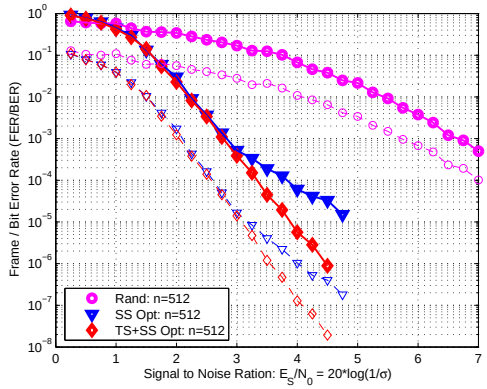

Fig. 1. Comparison of cyclically lifted codes. The distances and multiplicities of SSs and 1-out TSs are: Rand (2, 1), (2, 8); SS Opt: (13, 40), (5, 4); TS+SS Opt: (11, 12), (10, 24).

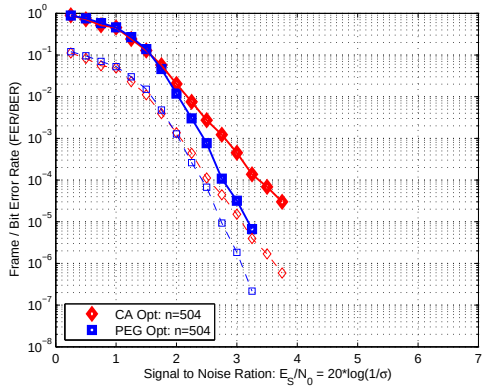

Fig. 2. Comparison of the $n = 504$ irregular PEG code [12] and an $n = 504$ random code optimized through CA.

We also compare the $n = 504$ irregular PEG code [12] with a code of the same length and degree distribution but with its minimal 1-out trapping distance maximized by CA. The irregular PEG code has two 1-out TS of size 7:

$$52, 53, 122, 136, 178, 229, 348$$
$$5, 42, 100, 131, 187, 199, 374$$

while the minimal 1-out trapping distance of our code is optimized to 8. The PEG code outperforms our code while neither codes experience hard error floors before FER=10^{-5}. This is the first concrete observation that maximizing the trapping distance alone is not leading to the optimal performance, a phenomenon different from the BEC case. The cause of the error floor lies deeper in the code structure, and we may need a new characterization of the error-prone events than the commonly accepted model of the TS.

VI. Conclusion

We have studied systematically the trapping sets (TSs) for LDPC codes with finite codeword length. The behavior of TSs in the cyclically lifted code ensemble has been quantified, which helps develop base-code optimization criteria so that the error floor of the lifted code can be lowered. The complexity of exhausting minimal TSs has been proven to be NP-complete. A new exhaustion algorithm has been devised, the efficiency of which is demonstrated on well-studied codes including algebraically and PEG constructed codes. The numerical results in this work show that removing small TSs can definitely improve the error floor but unlike the case of stopping sets for binary erasure channels, removing the small TSs alone does not guarantee the best performance. The combination of TS exhaustion/analysis and the pseudo-codeword framework is currently under investigation, which will lead to better understanding of the source of the error floor for non-erasure channels.

References

[1] C. Di, D. Proietti, E. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, no. 6, pp. 1570–1579, June 2002.

[2] T. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annual Allerton Conf. on Comm., Contr., and Computing*. Monticello, IL, 2003.

[3] D. MacKay and M. Postol, "Weakness of Margulis and Ramanujan-Margulis low-density parity check codes," *Electronic Notes in Theoretical Computer Science*, vol. 74, 2003.

[4] P. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes." *IEEE Trans. Inform. Theory*, submission preprint - arXiv:cs.IT/0512078.

[5] M. Stepanov and M. Chertkov, "Instanton analysis of low-density parity-check codes in the error-floor regime," in *Proc. IEEE Int'l Symp. Inform. Theory*. Seattle, WA, July 2006, pp. 552–556.

[6] D. Divsalar, "Ensemble weight enumerators for protograph LDPC codes," in *Proc. IEEE Int'l Symp. Inform. Theory*. Seattle, WA, July 2006.

[7] K. Krishnan and P. Shankar, "On the complexity of finding stopping distance in Tanner graphs," in *Proc. 40th Conf. Inform. Sciences & Systems*. Princeton, NJ, March 2006.

[8] C.-C. Wang, S. Kulkarni, and H. Poor, "Upper bounding the performance of arbitrary finite LDPC codes on binary erasure channels," in *Proc. Int'l. Symp. Inform. Theory*. Seattle, WA, July 2006.

[9] C. Cole, S. Wilson, E. Hall, and T. Giallorenzi, "Analysis and design of moderate length regular LDPC codes with low error floors," in *Proc. 40th Conf. Inform. Sciences and Systems*. Princeton, NJ, March 2006.

[10] X. Hu, M. Fossorier, and E. Eleftheriou, "On the computation of the minimum distance of low-density parity-check codes," in *Proc. IEEE Int'l. Conf. Commun.* Paris, France, 2004, pp. 767–771.

[11] C.-C. Wang, "Code annealing and the suppressing effect of the cyclically lifted LDPC code ensemble," in *2006 IEEE Information Theory Workshop*. Chengdu, China, October 2006.

[12] X. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386–398, January 2005.

[13] A. Montanari, "The asymptotic error floor of LDPC ensembles under BP decoding," in *Proc. 44th Annual Allerton Conf. on Comm., Contr., and Computing*. Monticello, Illinois, Sept. 2006.

[14] E. Sharon and S. Litsyn, "A method for constructing LDPC codes with low error floor," in *Proc. IEEE Int'l. Symp. Inform. Theory*, July 2006, pp. 2569–2573.

[15] R. Tanner, D. Sridhara, A. Sridharan, T. Fuja, and D. Costello, Jr., "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inform. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.

[16] J. Rosenthal and P. Vontobel, "Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis," in *Proc. of 38th Allerton Conf. Commun. Control & Computing*. Monticello, IL, 2000, pp. 248–257.

[17] S. Ländner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. 2005 Int'l. Conf. Wireless Networks, Commun. and Mobile Computing*. Maui, HI, USA, June 2005, pp. 630–635.