On The Error-Prone Substructures for The Binary-Input Ternary-Output Channel and Its Corresponding Exhaustive Search Algorithm

Gyu Bum Kyung
Samsung Advanced Institute of Technology
Samsung Electronics, Yongin, Korea
E-mail: gyubum.kyung@samsung.com

Chih-Chun Wang
School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907, USA
E-mail: chihw@purdue.edu

***Abstract*—The error floor performance of a low-density parity-check (LDPC) code is highly related to the presence of error-prone substructures (EPSs). In general, existing characterizations of the EPSs are inspired by the LDPC decoding behavior under simple binary erasure channel (BEC) and binary symmetric channel (BSC) models. In this work, we first introduce a new class of EPSs: *the 1-shot EPSs and static EPSs for the binary-input ternary-output channel (BITOC)*. By focusing on BITOCs, which are a step closer to additive white Gaussian channels (AWGNC), the proposed EPS would better characterize the decoding behavior of the AWGNC than the existing BEC- or BSC-based definitions. We then develop an efficient search algorithm that can exhaustively enumerate all small BITOC EPSs. The new exhaustive algorithm enables us to order the harmfulness of the EPSs and distinguish within a given EPS which bits are more prone to what types of errors. The proposed algorithm can also be regarded as a unified search method for the existing EPSs such as cycles, codewords, stopping sets, and fully absorbing sets. The proposed methodology is potentially generalizable to the binary-input m-ary output channel.**

I. Introduction

Low-density parity-check (LDPC) codes can be decoded by the efficient belief propagation (BP) decoder, which is, nonetheless, still suboptimal. One drawback of LDPC codes is the *error floor* phenomenon in the high signal-to-noise ratio (SNR) regime [1]. The cause of the error floor is due to the presence of the *error-prone substructures* (EPSs) [2], [3]. Some examples of the EPSs are stopping sets (SSs) [2], near-codewords [3], trapping sets (TSs) [1], and fully absorbing sets (FASs) [4].

Most existing works on error floor characterization started from considering the decoding error events on the simplest channel models, such as the binary erasure channel (BEC) and the binary symmetric channel (BSC) models. More specifically, SSs are the EPSs when performing erasure decoding on the BEC. FASs are the EPSs when performing bit-flipping decoding on the BSC [4]. By focusing on simple channel models, the corresponding EPSs can be characterized in a rigorous graph-theoretic manner. Efficient algorithms can then be devised to enumerate and later eliminate small EPSs, which further lowers the error floor. The EPS search algorithms can be classified into two categories: Exhaustive and non-exhaustive search algorithms. Exhaustive search algorithms provide a complete list of all EPSs in a given LDPC code

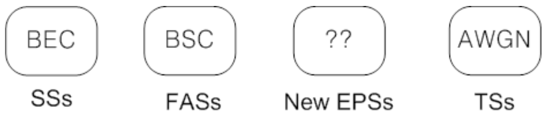

Fig. 1. The relationship of existing EPSs and the operational-definition of trapping sets (TSs) on AWGNCs.

that meet the search criteria, which can be used for thorough diagnosis of the cause of the error floor and for the subsequent low-error-floor code design. [5] proposed the first such algorithm that exhaustively searches for small SSs. A more efficient exhaustive SS search algorithm was later proposed in [6]. Motivated by [6] and [7], new efficient exhaustive search algorithms for FASs for LDPC codes are devised in [8] and [9] based on the branch-&-bound algorithm and integer-programming (IP) solvers. It is worth noting that the common goal of the above approaches is *to understand the notoriously challenging problem of characterizing/lowering the error floor of BP decoding over the realistic additive white Gaussian channel (AWGNC) model by studying the EPSs that are rigorously defined on the much simpler channel models.*

Although initial success has been obtained [5], [6], [8], there are problems in the above SS- and FAS-based approaches. First, the BEC and the BSC models are drastically simplified versions of the AWGNC. Second, the definition of the FAS is motivated by the bit-flipping decoder, which is a much simplified version of the BP decoder. As a result, BP decoding over AWGNCs is not fully captured by the SS- and FAS-based approaches. To mitigate such problems, a solution is to characterize *the EPS based on a channel model that is closer to the AWGNC and for a decoding method that better mimics the floating-point BP decoder (also see Fig. 1).*

In this paper, we provide the first step along this direction. We first define new types of EPSs based on the binary-input ternary-output channel (BITOC). We then show that even with the new EPS definitions that better capture the AWGNC decoding behavior, one can design efficient search algorithms that exhaustively enumerate all such EPSs with small sizes. In particular, we cast the *bounding step* of the existing branch-&-bound exhaustive algorithms as a new IP problem. Our results can also be viewed as the unification of existing search methods for the existing EPSs such as cycles, codewords, SSs, and FASs. Other benefits of our results include the ordering of 'harmfulness' of EPSs, and distinguishing within a given EPS which bits are more prone to what types of errors. The proposed methodology is also potentially generalizable to the binary-input m-ary output channels.

II. Binary-Input Ternary-Output Channels (BITOCs) and the Corresponding EPSs

A. BITOCs

The BITOC $\boldsymbol{X} \rightarrow \boldsymbol{Y}$ is defined as a channel with binary input set $\boldsymbol{X} = \{+1, -1\}$ and ternary output set $\boldsymbol{Y} = \{+1, 0, -1\}$ such that

$$P(Y = x|X = x) = 1 - p_1 - p_2 \tag{1}$$

$$P(Y = 0|X = x) = p_1 \tag{2}$$

$$P(Y = -x|X = x) = p_2 \tag{3}$$

where X and Y are the input and output, respectively, and p_1 and p_2 are prespecified channel parameters. We then apply the following 3-level quantized version of the BP decoding: Each message $m_{v_i \rightarrow c_j}$ (or $m_{c_j \rightarrow v_i}$) takes values in $\{-1, 0, 1\}$. The variable node message map is defined as

$$m^{(l)}_{v_i \rightarrow c_j} = \mathsf{sgn}\left(m^{(0)}_{v_i} + \sum_{c_k \in \mathcal{N}(v_i) \backslash c_j} m^{(l-1)}_{c_k \rightarrow v_i} \right) \tag{4}$$

where $m^{(0)}_{v_i} \triangleq y_i$ is the received channel output for variable node v_i, and the signum function $\mathsf{sgn}(x)$ is defined by

$$\mathsf{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} . \tag{5}$$

The message from a check node c_j to a variable node v_i is

$$m^{(l)}_{c_j \rightarrow v_i} = \prod_{v_k \in \mathcal{N}(c_j) \backslash v_i} m^{(l)}_{v_k \rightarrow c_j}. \tag{6}$$

The final decision $m^{(l)}_{v_i}$ is calculated by

$$m^{(l)}_{v_i} = \mathsf{sgn}\left(m^{(0)}_{v_i} + \sum_{c_k \in \mathcal{N}(v_i)} m^{(l-1)}_{c_k \rightarrow v_i} \right). \tag{7}$$

B. 1-shot EPSs and Static EPSs

In this section, we define new EPSs called 1-shot EPSs and static EPSs for the BITOC.

1) 1-shot EPSs : Suppose we transmit an all-one codeword that contains only channel inputs '+1'. Then, after passing through a BITOC, we call the received channel output a 1-shot EPS if it satisfies the following condition:

$$m^{(1)}_{v_i} = m^{(0)}_{v_i} \text{ and at least one } m^{(0)}_{v_i} \neq +1 \quad \text{for all } v_i \in V \tag{8}$$

where V is the set of variable nodes. Namely, the 1-shot EPS refers to the error pattern such that the variable node decisions stay the same after the first iteration of the 3-level BP decoding.

2) Static EPSs : We define the static EPS as follows:

$$m^{(l)}_{v_i \rightarrow c_j} = m^{(0)}_{v_i} \text{ and at least one } m^{(0)}_{v_i} \neq +1 \quad \text{for all } v_i \in V, c_j \in \mathcal{N}(v_i) \text{ and } l. \tag{9}$$

The static EPS refers to the error patterns for which the message $m^{(l)}_{v_i}$ at every iteration l stays the same value for all v_i.

III. A New Exhaustive Search Algorithm for The BITOC EPSs

A. The Main Branch-&-Bound Structure

Let $n \triangleq |V|$ denote the total number of variable nodes. We say that the i-th coordinate of an n-dimensional quaternary vector $\mathrm{s} \in \{-1, 0, 1, *\}^n$ is an "unconstrained position" if the value of its i-th coordinate is "$*$". We say that a quaterary vector $\mathrm{s}_1 = (s_{11}, \cdots s_{1n})$ is compatible with $\mathrm{s}_0 = (s_{01}, \cdots s_{0n})$ if $\mathrm{s}_{1i} = \mathrm{s}_{0i}$ for all i that is *not* an unconstrained position of s_0. For example, $\mathrm{s}_1 = (1, -1, 0, 1, 0)$ is compatible to $\mathrm{s}_0 = (*, -1, *, 1, 0)$. Let $\mathsf{Obj}(\mathrm{s})$ be the corresponding *objective value* of a vector s, which will later be used in an IP problem during the bounding step. In addition, we use $\mathcal{A}$ to denote a collection of 1-shot EPSs. The overall branch-&-bound structure is described as follows.

Algorithm Exhaustively searching for 1-shot EPSs.

```
1:  Input: the parity check matrix $\boldsymbol{H}$ and a scalar $s_{\max}$, which
    is an upper bound of the objective values of interest.
2:  Set $\mathbf{S} \leftarrow \{(*, *, \cdots, *)\}$ and set $\mathcal{A} \leftarrow \emptyset$.
3:  while $\mathbf{S} \neq \emptyset$ do
4:    Take one vector $\mathrm{s}_0$ from $\mathbf{S}$, and set $\mathbf{S} \leftarrow \mathbf{S} \backslash \mathrm{s}_0$.
5:    Using an IP solver to compute a lower bound $b$ for the
      objective values of all vectors $\mathrm{s}'$ that satisfy both (i)
      being a 1-shot EPS and (ii) compatible to $\mathrm{s}_0$.
6:    if $b \leq s_{\max}$ then
7:      if during the computation of $b$, the IP solver also finds
        a vector $\mathrm{s}^*$ with $\mathsf{Obj}(\mathrm{s}^*) = b$ then
8:        $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathrm{s}^*\}$.
9:      end if
10:     Choose one unconstrained position of $\mathrm{s}_0$ and generate
        three new vectors $\mathrm{s}_1$, $\mathrm{s}_2$, and $\mathrm{s}_3$ by setting the uncon-
        strained position of $\mathrm{s}_0$ to $-1$, $0$ and $1$, respectively.
11:     $\mathbf{S} \leftarrow \mathbf{S} \cup \{\mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3\}$.
12:   end if
13: end while
14: Output: $\mathcal{A}$ is the exhaustive list of all 1-shot EPSs with
    objective values ($\leq s_{\max}$).
```

B. The IP Problem in Line 5

For the following, we first describe how to formulate an IP problem that finds the minimum objective value of *all* 1-shot EPSs without the condition "being compatible to s_0." As will be seen shortly after, the compatibility condition can be added readily by slightly modifying the following IP problem. The proposed IP problem can be easily modified to search for static EPSs as well. The detailed modification is omitted due to space constraints.

The proposed IP problem consists of 4 classes of *binary integer variables*, denoted by x_{v_i}, y_{v_i}, z_{e_l}, and u_{e_l}, and one class of non-negative integer variables w_{e_l}, where the subscripts indicate whether the variable is associated to a variable node $v_i \in V$ or an edge $e_l \in E$, E being the set of all edges. For simplicity, we use x_i, y_i, w_l, z_l, and u_l as shorthand when there is no ambiguity. *The binary integer pair (x_i, y_i) represents the value of the variable node messages $m_{v_i \to c}$. That is, (x_i, y_i) being $(0,0)$, $(0,1)$, and $(1,0)$ corresponds to $m_{v_i \to c}$ being 1, 0, -1, respectively.* Similarly, for a given edge e_l connecting variable node v and check node c, the pair (z_l, u_l) represents the value of the check to variable node messages, i.e., (z_l, u_l) being $(0,0)$, $(0,1)$, and $(1,0)$ corresponds to $m_{c \to v}$ being 1, 0, -1, respectively. The following is an IP problem of finding the "smallest objective value" BITOC 1-shot EPSs for the BITOC.

$$\text{Minimize} \quad \log \frac{1-p_1-p_2}{p_2} \sum_{v_i \in V} x_i + \log \frac{1-p_1-p_2}{p_1} \sum_{v_i \in V} y_i \quad (10)$$

subject to the following groups of conditions:
Regularity conditions:

$$x_i, y_i, z_l, u_l \in \{0,1\}, \forall v_i \in V, \forall e_l \in E. \quad (11)$$

$$w_l \text{ is a non-negative integer }, \forall e_l \in E. \quad (12)$$

$$x_i + y_i \leq 1, \forall v_i \in V \text{ and } z_l + u_l \leq 1, \forall e_l \in E. \quad (13)$$

Check node message map - Erasure consideration: For all edges $e_l = (v, c)$, we must have

$$\left(\max_{v_k \in \mathcal{N}(c) \backslash v} y_k \right) \leq u_l \leq \left(\sum_{v_k \in \mathcal{N}(c) \backslash v} y_k \right) \quad (14)$$

Check node message map - Parity consideration: For all edges $e_l = (v, c)$, we must have

$$0 \leq \left(\sum_{v_k \in \mathcal{N}(c) \backslash v} x_k - 2w_l - z_l \right) \leq u_l \quad (15)$$

Variable node message constraints for the 1-shot EPS: For all $v_i \in V$, its *adjacent* edges e_l is of the form $e_l = (v_i, c)$ for some check node c. Use $E(v_i)$ to denote all adjacent edges of v_i. The degree of v_i is thus $d_{v_i} \triangleq |E(v_i)|$. For all $v_i \in V$, we must have

$$d_{v_i}(x_i + y_i) \leq \left(\sum_{e_l \in E(v_i)} u_l \right) + \left(2 \sum_{e_l \in E(v_i)} z_l \right) \leq d_{v_i}(x_i + 1) \quad (16)$$

Avoid searching for the trivial all-1 pattern:

$$\sum_{v_i \in V} (x_i + y_i) \geq 1 \quad (17)$$

We first explain the objective function in (10). Assume that we transmit an all-one codeword, which means that without any noise, we will have $x_i = 0$ and $y_i = 0$ for all $v_i \in V$. However, with errors, the number of -1's (bit-flipping errors) can be computed by $\sum_{v_i \in V} x_i$ and the number of 0's (erasures) can be computed by $\sum_{v_i \in V} y_i$. As a result, the likelihood of the received channel output can be calculated by

$$(1-p_1-p_2)^{n - \sum_{v_i \in V} x_i - \sum_{v_i \in V} y_i} (p_2)^{\sum_{v_i \in V} x_i} (p_1)^{\sum_{v_i \in V} y_i}. \quad (18)$$

Our search algorithm thus finds the most likely error pattern that maximizes (18), which is equivalent to minimizing (10).

Regularity conditions (11) and (12) specify the types of integers for the variables. Since when mapping a binary integer pair to a 3-level message we did not use the case $(x_i, y_i) = (1,1)$, we use (13) to disallow the case $x_i = y_i = 1$. Similarly, we also disallow the case $z_l = u_l = 1$.

For the check node message map (6), the output of check node message is erasure if any one of the inputs is erasure. If none of the inputs is erasure, the output is the parity sum of all inputs. (14) guarantees that if $y_k = 0$ (i.e., no erasure) for all $v_k \in \mathcal{N}(c) \backslash v$ then the u_l value over edge $e_l = (v, c)$ is zero. If at least one y_k is 1, then (11) and (14) jointly imply $u_l = 1$. At the same time, when $u_l = 0$, the value of z_l has to be the parity-sum of all input messages. This is achieved by (15). Namely, when $u_l = 0$, (15) collapses to

$$\left(\sum_{v_k \in \mathcal{N}(c_j) \backslash v_i} x_k \right) = 2w_l + z_l. \quad (19)$$

The variable z_l is thus the parity-sum. Note that when $u_l = 1$, we have $z_l = 0$ by (13) and (15) can always be satisfied by some integer w_l

The intuition behind (16) is as follows. For example, the output variable node message is 1 (i.e., $x_i = y_i = 0$) when the sum of the input check-to-variable message is positive. When $x_i = y_i = 0$, (16) ensures that $0 \leq$ Inner term $\leq d_{v_i}$, which is related to the sum of input messages being positive (not too many e_l with $z_l = 1$). We omit the proof of (16) because of the space. To further accommodate the condition "compatible to s_0", we simply add the following "hardwiring conditions":

- for all the unconstrained positions of s_0, we allow the corresponding x_i, y_i to be free variables $\in \{0,1\}$.
- for all other x_i, y_i, set their values according to the value of the i-th coordinate of s_0 and the binary-pair to ternary-message map.

Remark: In general, solving the above IP problem is very time consuming. However, since we are only interested in finding a lower bound b (see Line 5 of the algorithm), we usually solve the relaxed mixed IP problem instead, which greatly enhances the efficiency.

IV. The Exhaustive Lists of BITOC EPSs for Several Representative Codes

We apply our new search algorithm to search for 1-shot EPSs and static EPSs of two regular $(3,6)$ LDPC codes (M204 and M504), and a PEG-type regular LDPC code (PEGR504).

TABLE I
THE NUMBER OF 1-SHOT EPSS, STATIC EPSS, AND FASS FOR DIFFERENT LDPC CODES.

	1-shot EPSs			Static EPSs			FASs			
Code	Type	Obj	Num	Type	Obj	Num	Type	Num	Type	Num
M204	(-1,0,0,0)	8.56	5	(0,0,0,0,0,0,0,0)	10.86	1	$(3,3)$	134	$(4,2)$	13
	(-1,0,0,0,0)	9.91	142	(0,0,0,0,0,0,0,0,0,0)	12.22	2	$(4,4)$	730	$(5,1)$	1
	(0,0,0,0,0,0,0,0)	10.86	1	(-1,-1,-1)	13.45	134	$(5,3)$	361	$(5,5)$	3895
	(-1,-1,0,0)	11.68	4624	(0,0,0,0,0,0,0,0,0,0,0)	13.57	3	$(6,2)$	63	$(6,4)$	4937
	(0,0,0,0,0,0,0,0,0,0)	12.22	2	(0,0,0,0,0,0,0,0,0,0,0,0)	14.93	6	$(6,6)$	20114	$(7,1)$	7
	(-1,0,0,0,0,0,0,0)	12.63	91	(0,0,0,0,0,0,0,0,0,0,0,0,0)	16.29	14	$(7,3)$	1720	$(8,0)$	1
	(-1,-1,0,0,0)	13.04	351	(0,0,0,0,0,0,0,0,0,0,0,0,0,0)	17.65	59	$(8,2)$	329	$(9,1)$	26
	(-1,-1,-1)	13.45	134	(-1,-1,-1,-1)	17.94	744	$(10,0)$	1	$(10,2)$	1867
	(0,0,0,0,0,0,0,0,0,0,0)	13.57	3				$(11,1)$	180		
	(-1,0,0,0,0,0,0,0,0)	13.98	50							
M504	(-1,0,0,0)	8.56	10	(-1,-1,-1)	13.45	159	$(3,3)$	159	$(4,2)$	5
	(-1,0,0,0,0)	9.91	60				$(4,4)$	1056	$(5,3)$	180
	(-1,-1,0,0)	11.68	5074				$(6,2)$	20	$(7,3)$	380
PEGR 504	(-1,0,0,0,0)	9.91	28	(-1,-1,-1,-1)	17.94	760	$(4,4)$	760	$(5,3)$	14
	(-1,-1,0,0)	11.68	3208				$(6,4)$	849	$(7,3)$	47

The codes are of lengths 204, 504, and 504, respectively, and their parity-check matrices are available in [10]. We set the BITOC parameter values by $p_1 = 0.2028$ and $p_2 = 0.0089$.[1]

The exhaustive lists of 1-shot and static EPSs of the four representative codes are summarized in Table I. Column 'Type' describes how many bit-flipping bits ('-1') and how many erasures ('0') in the 1-shot EPS (resp. static EPSs). Column 'Obj' describes the objective value of the EPS. Column 'Num' describes the number of such 1-shot EPSs (resp. static EPSs). For example, among all 1-shot EPSs of M504, the smallest objective value is 8.56. All "smallest" 1-shot EPSs of M504 have 1 flipped bit and three erasures, and there are 10 such 1-shot EPSs. As can be seen in the table, the proposed algorithms are very efficient and can exhaustively enumerate many types of 1-shot and static EPSs.

A. *The topological structure of BITOC EPSs*

The topologies of 1-shot and static EPSs are closely related to other existing FASs. For example, the type-$(0,0,0,0,0,0,0,0)$ 1-shot EPS of M204 has 8 erasure bits. It turns out that these 8 bits are also the support set of the smallest non-zero codeword. Another example is the type-$(-1,0,0,0,0)$ 1-shot EPS of PEGR504. In [8] all 14 $(5,3)$ FASs of PEGR504 have been exhaustively enumerated (also see the right-most columns of Table I). It turns out that the 28 type-$(-1,0,0,0,0)$ 1-shot EPSs correspond to the 14 $(5,3)$ FASs in a way that the position of the flipped bit (-1) of a 1-shot EPS can be chosen from two possible locations in a given $(5,3)$ FAS.

By (9), the static EPS is a much more restrictive definition when compared to that of the 1-shot EPS (8). Therefore there is a far smaller number of static EPSs. For example, the smallest static EPS of M204 is of type-$(0,0,0,0,0,0,0,0)$, which turns out to be same error pattern as the "third smallest" 1-shot EPS of M204. From our observations, all BITOC static EPSs are of one of the following forms: being a codeword, being a SS, or being a FAS.

B. *Properties of the proposed exhaustive search algorithms*

- One advantage of the new EPS definition is now we can order the harmfulness of EPSs using their objective values instead of their Hamming weights. For example, the type-$(-1,0,0,0)$ 1-shot EPS of M204 has smaller objective value than the type-$(0,0,0,0,0,0,0,0)$ one, the latter of which consists of erasure bits on the minimum codeword. This shows that the error probability contributed by the minimum codeword might not be the dominant cause of the error floor when compared to the type-$(-1,0,0,0)$ 1-shot EPS.
- We are able to further distinguish which bits (within a given EPS) are more prone to error. For example, for type-$(-1,0,0,0,0)$ 1-shot EPSs of PEGR504, the 1-shot error occurs when the bit corresponding to '-1' needs to be flipped but the other four bits corresponding to 0 need only to be erased. Therefore the four other bits are more susceptible to error.
- By using pairs of binary integer variables to represent the quantized 3-level messages in BP decoding, we are able to convert the non-linear sgn and parity-sum functions into linear integer constraints. Such a problem can then be solved efficiently by a linear IP solver. We believe such an approach could be extended to the more general binary-input m-ary output channel.
- In our implementation, the new exhaustive search algorithm runs slower than the existing SS and FAS exhaustive algorithms [6], [8]. This is probably inevitable since our algorithm searches for more intricate decoding structures over BITOCs. It is likely that if we generalize the results to binary-input m-ary output channels, the resulting exhaustive search algorithm will run even slower. On the other hand, the search of EPSs only needs to be performed in the code design and analysis stage.

[1]All three codes we examined have error floors at $\mathsf{SNR} = 5.0$dB. Suppose the input is $x \in \{-1,1\}$, the output $y = x + n/\sqrt{\mathsf{SNR}}$, and the quantized output of y with thresholds -0.5 and 0.5 is denoted by $\hat{y}$. Then with probabilities 0.2028 and 0.0089 we will observe $\hat{y} = 0$ and -1, respectively, which gives our p_1 and p_2 values.

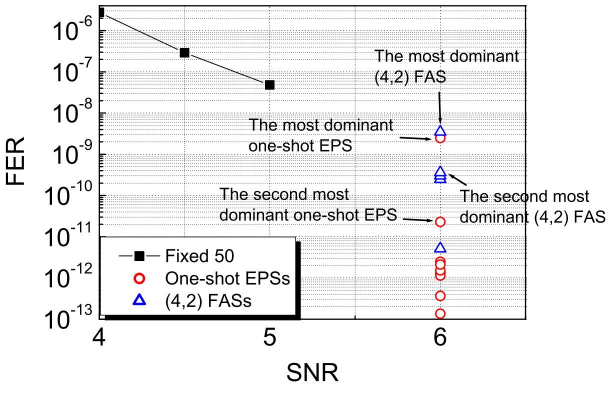

Fig. 2. FER contribution of different FASs and 1-shot EPSs for the M504 code. The curve "Fixed 50" corresponds to the 5-bit fixed point decoder with 50 iterations. The error contribution of the 10 type-$(-1,0,0,0)$ 1-shot EPSs and 5 $(4,2)$ FASs is also plotted.

V. Simulation Results

The goal of this work is by no means to provide a complete solution to the error-floor characterization problem but rather to propose a new methodology that could better approximate and explain the BP decoding behavior and thus enhance our understanding of the problem. In this section, we discuss several observations of ours.

Even with the sorted objective values, some EPSs are more harmful than the others. It is well known [11] that among all (s,t) FASs of the same topology, say among all $(4,2)$ FASs, some contribute to the error floor more than the others. We first observe that even among the BITOC EPSs of the same topology (and thus the same objective value), again some are much more detrimental than the others when considering BP decoding over AWGNCs. To illustrate this observation, we use the importance sampling method developed in [1] to estimate the error floor contribution of different EPSs of the same topology. In addition, we use the 5-bit decoder of [12] to accelerate the simulation so that we can look into the low Frame Error Rate (FER) with range 10^{-9}–10^{-13}. In Fig. 2, we plot the error floor contribution of 5 $(4,2)$ FASs and 10 1-shot EPSs of type-$(-1,0,0,0)$ for the M504 code. As can be seen, among FASs and 1-shot EPSs of the same topologies, some of them contribute much more to the error floor (around 10^{-9}) than the others (around 10^{-13}). This thus prompts further investigation of the harmfulness of the EPSs of the same topology, which will be discussed later.

On the other hand, **significant overlaps between a 1-shot EPS and the FASs seem to be a good indicator of how dominant a 1-shot EPS could be.** We observe that a 1-shot EPS overlaps significantly with a $(4,2)$ FASs, which are the dominant 1-shot EPS and FAS in the top-right corner of Fig. 2. All other 1-shot EPSs do not overlap much with any small FASs and their contribution is much smaller. The same phenomenon is also observed in several other benchmark LDPC codes, which are not reported herein due to space constraint. One heuristic explanation is that *the 1-shot EPSs are the structures for which the refinement of the decisions proceeds only slowly (not able to correct the error after one iteration). Therefore if there are a lot of bad FASs structures that overlap with a 1-shot EPS, then with high probability the refinement of the decision is not fast enough and the decision will thus "fall into one of the overlapped FAS structures" and cause errors.*

VI. Conclusion

We have introduced two new types of EPSs, termed the 1-shot EPSs and static EPSs, based on the simplified BP decoder over the BITOC, which is closer to the AWGNC model. We have also proposed a new algorithm that exhaustively enumerates such EPSs. The proposed algorithm has been applied to several LDPC codes and new exhaustive lists of BITOC EPSs have been found. The exhaustive lists of small BITOC EPSs enables us to order the harmfulness of EPSs and also distinguish which bits are susceptible to what type of errors. The proposed algorithm can also be regarded as a unified search method for the existing EPSs including codewords, SSs, and FASs. The proposed solution is potentially generalizable to the binary-input m-ary output channels.

Acknowledgment

This work is supported by NSF Grant No. CCF-0845968.

References

[1] T. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annu. Allerton Conf. on Commun. Contr. and Computing*. Monticello, IL, Oct. 2003.

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

[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] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inform. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[5] C.-C. Wang, S. R. Kulkarni, and H. V. Poor, "Finding all small error-prone substructures in LDPC codes," *IEEE Trans. Inform. Theory*, vol. 55, no. 5, pp. 1976–1998, May 2009.

[6] E. Rosnes and Ø. Ytrehus, "An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices," *IEEE Trans. Inform. Theory*, vol. 55, no. 9, pp. 4167–4178, Sept. 2009.

[7] A. Keha and T. M. Duman, "Minimum distance computation of LDPC codes using a branch and cut algorithm," *IEEE Trans. Commun.*, vol. 58, no. 4, pp. 1072–1079, Apr. 2010.

[8] G. B. Kyung and C.-C. Wang, "Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder," in *Proc. IEEE Int'l. Symp. Inform. Theory*. Austin, TX, USA, Jun. 2010, pp. 739–743.

[9] G. B. Kyung, *Design & Analysis for Practical LDPC-Coded Systems from Broadcast Channel to Low Error-Floor Applications*. Ph.D. thesis, Purdue University, West Lafayette, IN, Aug. 2011.

[10] D. J. C. MacKay, "Encyclopedia of spase graph codes," [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html.

[11] D. V. Nguyen, S. K. Chilappagari, M. W. Marcellin, and B. Vasić, "LDPC codes from Latin squares free of small trapping sets," Aug. 2010, preprint-arXiv:cs.IT/1008.4177.

[12] T. J. Richardson, H. Jin, and V. Novichkov, "Methods and apparatus for decoding LDPC codes," *United States Patent 6,633,856*, Oct. 2003.