Towards Maximum Likelihood Decoding of Finite Length LDPC Codes via Iterative Decoder Diversity

David Declercq *Senior Member, IEEE*, Bane Vasić, *Fellow Member, IEEE*,
Erbao Li and Shiva Planjery

Abstract

We introduce a method for improving the guaranteed error correction capability of regular low-density parity check codes. The proposed algorithm operates not one but a set iterative decoders, for which the message-passing update rules are judiciously chosen to ensure that decoders have different dynamics on a specific finite-length code. The idea is that if an error pattern cannot be corrected by one particular decoder, there another decoder in the set of decoders which can correct this pattern. The decoders are from the class of finite-alphabet iterative decoders. A systematic construction of the decoder set is proposed, based on knowledge of the most harmful trapping sets. Different dynamics of the selected decoders on the for the same error patterns is interpreted as "decoding diversity." We show that a the use of a plurality of message update rules can collectively correct more errors patterns than the stand-alone decoders, and therefore increase significantly the guaranteed error correction capability of regular, column-weight three codes. This improvement is illustrated in detail on the particular example of the $(N = 155, K = 62, d_{min} = 20)$ Tanner code.

I. Introduction

It is now well established that iterative decoding (ID) approaches the performance of maximum likelihood decoding (MLD) of the low density parity check (LDPC) codes, asymptotically in the block

D. Declercq and E. Li are with the ETIS, ENSEA/University of Cergy-Pontoise/CNRS UMR 8051, 95014 Cergy-Pontoise, France (email: declercq@ensea.fr)

B. Vasić and S. Planjery are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85719 USA (e-mail: vasic@ece.arizona.edu).

Manuscript received January 30, 2012. The authors would like to acknowledge generous support of the NSF (Grants CCF-0830245 and CCF-0963726), and the "Intitut Universitaire de France" grant.

length. However, for finite length sparse codes, iterative decoding fails on specific subgraphs of a Tanner graph, generically termed as trapping sets [4, 18]. Trapping sets (TS) give rise to an error floor, described as an abrupt degradation of the code error performance in the high signal to noise ratio regime. This performance degradation is also characterized by the notion of pseudo-codewords [3], which represent attractor points of iterative message passing decoders, analogous to codewords which represent solutions of the MLD. A precise structural relationship between trapping sets and pseudo-codewords of a given Tanner graph and a decoding algorithm is not yet fully established, but it is known that pseudo-codewords on the binary symmetric channel (BSC) are typically supported by the small TS of the LDPC code in the sense that the bits which form a pseudo-codeword are contained in the topological structure which define a TS. Many authors have pointed out that the minimum Hamming weight of pseudo-codewords is generally smaller that the minimum distance for most LDPC codes [27]. Thus, the presence of trapping sets prevents, in principle, iterative decoders to approach or reach the performance of MLD for finite lengths LDPC codes.

An error correcting code $\mathcal{C}$ is said to have t-guaranteed error correction capability using a decoder $\mathcal{D}$ over the BSC channel if $\mathcal{D}$ can correct all error patterns of weight t or less. Following the discussion of the previous paragraph, increasing the guaranteed error correction of an LDPC code improves the frame error rate (FER) performance in the high signal to noise ratio region, as error patterns with small Hamming weights are dominant in this region. The problem of guaranteed error correction is critical for applications like magnetic, optical and solid-state storage, flash memories, optical communication over fiber or free-space, as well as an important open problem in coding theory. Guaranteed error correction is usually achieved by using Bose-Chaudhuri-Hocquenghem (BCH) or Reed-Solomon (RS) codes and hard-decision decoders such as the Berlekamp-Massey decoder [28], but very little is known about the guaranteed error correction capability of LDPC codes under iterative decoding. The main reason for this comes from the fact that even though the weaknesses of an LDPC code are relatively well characterized through identification of its trapping sets, it is an arduous task to determine wether a particular ID succeeds in correcting all t-error patterns localized on all the trapping sets. As an example, for strictly regular LDPC codes with $d_v = 3$ connections per coded bit, and girth $g = 8$, it has been shown that the guaranteed error correction capability differs when different ID are used [19, 20].

In this paper, we introduce a general approach to improve the guaranteed error correction capability of regular LDPC codes, by using a set of carefully chosen ID which are tuned to have different dynamical behaviors on a specific finite length LDPC code. The idea is that if an error pattern located in a dominant

TS cannot be corrected by one particular ID, there is another decoder in the set of considered decoders that can correct this pattern. By choosing carefully a set of IDs, it is then possible to correct all error events located in the dominant trapping sets and therefore increase the guaranteed error capability of the LDPC codes, compared to classical ID such as belief propagation (BP) or its derivatives (BP-based, min-sum). The capability of the set of ID to collectively correct a diverse set of error patterns is termed *"decoding diversity"*. Throughout this paper, we will restrict the choice of decoders to the recently introduced class of finite alphabet iterative decoders (FAID) [7, 8], for which the messages may be represented only by three bits. The reason of this choice is that FAID have been demonstrated to be more powerful than usual IDs in the error floor region [7, 16], and also because the FAID framework allows to define a plurality of different iterative decoders [21] tailored to correct a desired set of error patterns.

The principle of our approach is described as follows: given a fixed LDPC code, we want to find a set of good decoders that — when used sequentially or parallely — can correct a fixed number of errors, say t. A brute force approach would rely on checking all possible error patterns of weight t for every possible decoder, and then choosing the set of decoders that correct all the patterns. However, this brute force approach is prohibitively complex. Instead, we focus on the error patterns associated with the smallest TS present in the code. The methodology then involves searching for such TS, build the corresponding error pattern sets, and then find a combination of decoders that can correct all these particular patterns. Although each FAID in the decoder set is very simple, we show that using increasingly larger sets of decoders, one can significantly increase the guaranteed error correction of iterative decoding for short length LDPC codes. We apply this procedure to the $(N = 155, K = 62, d_{min} = 20)$ Tanner code, and demonstrate that using the framework of FAID decoding diversity allows to increase the guaranteed error correction capability from $t = 5$ for achievable by the best existing iterative decoders, to $t = 7$ using our new framework, inducing thereby a significant improvement in the slope of the error floor in the FER curve.

The paper is organized as follows. In a first section, we recall the main notions presented in the literature that are useful for the understanding of our methodology of iterative decoding diversity. Then in a second section of the paper, we describe the general methodology, based on a precise knowledge of the TS distribution of a given code and an iterative decoder selection algorithm. Finally, in a third section, we apply our procedure to the specific example of the $(N = 155, K = 62, d_{min} = 20)$ Tanner code.

II. PRELIMINARIES

In this section, we introduce the main notations and concepts introduced in the literature which are required for the decoding procedure that we propose in this paper. After basic notations on LDPC codes and channels assumptions, we present the general notion of iterative message passing decoders and the concept of Trapping sets, which will be used in the rest of the paper.

A. Regular LDPC Codes

LDPC codes belong to the class of linear block codes which can be defined by sparse bipartite graphs called Tanner graphs [1]. The Tanner graph G of a binary LDPC code $\mathcal{C}$ is a bipartite graph with two sets of nodes: the set of variable nodes $V = \{v_1, \cdots, v_N\}$ and the set of check nodes $C = \{c_1, \cdots, c_M\}$. The check nodes (variable nodes resp.) connected to a variable node (check node resp.) are referred to as its neighbors. The set of neighbors of a node v_i is denoted as $\mathcal{N}(v_i)$, and the set of neighbors of node c_j is denoted by $\mathcal{N}(c_j)$. The degree of a node is the number of its neighbors, and for regular LDPC codes, we denote by $d_v = |\mathcal{N}(v_i)| \ \forall i$ the constant variable node degree, and by $d_c = |\mathcal{N}(c_j)| \ \forall j$ the constant check-node degree.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ be a vector such that x_i denotes the value of the bit associated with variable node v_i. The vector $\mathbf{x}$ is a codeword if and only if for each check node, the modulo two sum of the bit values of its neighbors is zero, which means that each and every parity-check equation is verified. An (N, d_v, d_c) regular LDPC code is then a code with rate $R \leq 1 - d_v/d_c$ that has a Tanner graph with N variable nodes each of degree d_v and Nd_v/d_c check nodes each of degree d_c.

B. Channel Assumptions

We assume that a binary codeword $\mathbf{x}$ is transmitted over a noisy channel and is received as $\mathbf{y}$. We consider only binary-input, memoryless, output-symmetric channels, so that

$$\Pr(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{N} \Pr(y_i|x_i).$$

$\Pr(y_i|x_i)$, the probability that y_i is received given that x_i was sent, depends only on the transmitted bit and the value of channel output, bit not on i.

The log-likelihood ratio (LLR) corresponding to each variable node v_i is given by

$$\gamma_i = \log\left(\frac{\Pr(y_i|x_i = 0)}{\Pr(y_i|x_i = 1)}\right).$$

We shall restrict our discussion to the Binary Symmetric Channel (BSC). On the BSC with transition probability α, every transmitted bit $x_i \in \{0,1\}$ is flipped with probability α and is received as $y_i \in \{0,1\}$. Hence, we have

$$\gamma_i = \begin{cases} \log\left(\frac{1-\alpha}{\alpha}\right) & \text{if } y_i = 0 \\ \log\left(\frac{\alpha}{1-\alpha}\right) & \text{if } y_i = 1 \end{cases} \tag{1}$$

The BSC channel can also be modeled as $\mathbf{y} = \mathbf{x} \oplus \mathbf{e}$, where the binary vector $\mathbf{e} = (e_1, e_2, \ldots, e_N)$ is referred to as an *error pattern*, and $\oplus$ is the modulo-two sum.

Throughout the rest of the paper, the support of an error vector $\mathbf{e} = (e_1, e_2, \ldots, e_N)$, denoted by $\text{supp}(\mathbf{e})$, is defined as the set of all positions i such that $e_i \neq 0$. A *weight* of the error pattern $\mathbf{e}$, denoted by $w(\mathbf{e})$ is the cardinality of $\text{supp}(\mathbf{e})$.

C. Iterative Message Passing Decoders

Iterative decoders operate by passing messages along the edges of the Tanner graph representation of the code. In each iteration, the messages on the edges are computed as a function of only local messages coming from their neighbors. More precisely, it excludes the incoming message on the edge for which the outgoing message is being determined at a particular node.

Let $m^{(k)}_{v_i \to c_j}$ denote the message passed by a variable node $v_i \in V$ to its neighboring check node $c_j \in C$ during the k^{th} iteration and $m^{(k)}_{c_j \to v_i}$ denote the message passed by a check node c_j to its neighboring variable node v_i. The set $m^{(k)}_{\mathcal{N}(v_i) \to v_i}$ denotes the set of all incoming messages to variable node v_i and $m^{(k)}_{\mathcal{N}(v_i) \setminus c_j \to v_i}$ denotes the set of all incoming messages to variable node v_i except from check node c_j. The set $m^{(k)}_{\mathcal{N}(c_j) \setminus v_i \to c_j}$ is defined similarly.

Let moreover $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ be the input to the decoder, which are used for the computation of the LLR values as indicated in equation (1). Let the output channel values take their values in the set $\mathcal{Y}$ and the messages of the ID take their values in the set $\mathcal{M}$ (note that in general, both $\mathcal{Y}$ and $\mathcal{M}$ could be continuous or discrete sets). A general iterative decoder is defined by two functions Φ_v and Φ_c, which determine the message update functions through the variable nodes and the check-nodes, respectively. The function $\Phi_v : \mathcal{Y} \times \mathcal{M}^{d_v - 1} \to \mathcal{M}$, denotes the update function used at the variable node, and $\Phi_c : \mathcal{M}^{d_c - 1} \to \mathcal{M}$ denotes the update function used at the check node. At each iteration $k > 0$, the iterative decoder updates each and every message in the graph as follows:

$$\begin{aligned} m_{v_i \to c_j}^{(0)} &= y_i & v_i \in V,\ c_j \in \mathcal{N}(v_i) \\ m_{c_j \to v_i}^{(k)} &= \Phi_c(m_{\mathcal{N}(c_j)\setminus v_i \to c_j}^{(k-1)}) & c_j \in C,\ v_i \in \mathcal{N}(c_j) \\ m_{v_i \to c_j}^{(k)} &= \Phi_v(y_i, m_{\mathcal{N}(v_i)\setminus c_j \to v_i}^{(k)}) & v_i \in V,\ c_j \in \mathcal{N}(v_i) \end{aligned}$$

Message update rules are symmetric functions, *i.e.*, they remain unchanged by any permutation of the incoming messages. In other words, the order of arguments is insignificant. Note that Φ_c is a symmetric function on all $d_c - 1$ arguments, and Φ_v is a partially symmetric function that is symmetric only on the $d_v - 1$ incoming messages, *i.e.* the output is independent of the ordering of the $d_v - 1$ incoming messages as arguments. In typical message-passing algorithms though, y_i and the $d_v - 1$ messages belong to the set of reals, $\mathbb{R}$, and the function Φ_v is symmetric. This is not the case for the class of finite alphabet iterative decoders that we focus on in this paper.

A function $\Psi : \mathcal{Y} \times \mathcal{M}^{d_v}$, called the decision function is used to take a decision on the bit value associated with each variable node $v_i \in V$ at the end of each iteration. The result of decoding after k iterations $\hat{\mathbf{x}}^{(k)} = (\hat{x}_1^{(k)}, \hat{x}_2^{(k)}, \ldots, \hat{x}_N^{(k)})$ is calculated as

$$\hat{x}_i^{(k)} = \Psi(y_i, m_{\mathcal{N}(v_i)\to v_i}^{(k)}) \tag{2}$$

In traditional message passing algorithms, especially when the messages passed are in the log domain, the decision function is simply implemented by looking at the sign of the sum of its arguments. In other words, $\hat{x}_i^{(k)} = 0$ if $y_i + \sum m_{\mathcal{N}(v_i)\to v_i}^{(k)} > 0$, $\hat{x}_i^{(k)} = 1$ if $y_i + \sum m_{\mathcal{N}(v_i)\to v_i}^{(k)} < 0$, and $\hat{x}_i^{(k)} = (\text{sgn}(y_i) - 1)/2$ otherwise, where the sgn function denotes the standard signum function.

The decoder also produces the syndrome vector $\mathbf{s}^{(k)} = (s_1^{(k)}, s_2^{(k)}, \ldots, s_M^{(k)})$, defined as

$$\hat{s}_j^{(k)} = \bigoplus \hat{x}_{\mathcal{N}(c_j)}^{(k)}$$

where $\hat{x}_{\mathcal{N}(c_j)}^{(k)}$ are the estimated bit values assigned to variable nodes neighboring to c_j, and the operator $\bigoplus$ denotes their modulo-two sum. The computation of the syndrome vector and its comparison to the all-zero vector serves as a halting criterion for the iterative decoder. The iterative algorithm runs until a maximum number of iterations, N_I, is reached or until the syndrome vector becomes the all-zero vector, which means that all parity-check equations are satisfied and a codeword has been detected.

During analysis of message-passing decoders throughout this paper, we shall assume that the all-zero codeword was transmitted. This is a valid assumption as explained in [13] since the decoder update equations as well as the channel outputs we are considering are symmetric.

D. Belief propagation algorithm: sum-product and min-sum

Message passing decoders can be broadly classified into: (i) hard decoding algorithms such as the Gallager A and the Gallager B algorithms (used for decoding over the BSC [23]), and (ii) soft decoding algorithms such as belief propagation (BP) or the min-sum algorithm [24, 25]. While the messages for the BP-based algorithms are real-valued, the quantized BP algorithms and algorithms with erasures such as Gallager-E [23] belong to the class of finite alphabet iterative decoders (FAID) [8] because their messages take values in a finite alphabet. In this paper, we shall focus on a new type of FAID which are different from existing quantized decoders primarily due to their difference in the definition of their variable node update functions as well as in the interpretation of their messages. These decoders are presented in detail in Section III.

For the sake of clarity and comparison purposes, we provide the update rules of the BP and min-sum algorithms. For both algorithms, the decoder input $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ is a real-valued vector of log-likelihoods ($\mathcal{Y} = \mathbb{R}$) such that $y_i = \gamma_i \ \forall i \in \{1, \ldots, N\}$, and γ_i is determined for the BSC using (1). Also since the messages are real-valued ($\mathcal{M} = \mathbb{R}$), $\Phi_v : \mathbb{R} \times \mathbb{R}^{d_v - 1} \to \mathbb{R}$ is the update rule used at a variable node of degree d_v, and $\Phi_c : \mathbb{R}^{d_c - 1} \to \mathbb{R}$ is the update rule used at a check node with degree d_c.

Let $m_1, m_2, \ldots, m_{l-1}$ denote the $l-1$ incoming messages of a node of degree l which are used in the calculation of the outgoing message. The update rules for the belief propagation algorithm are given as follows [25]:

$$\Phi_v(y_i, m_1, \ldots, m_{d_v - 1}) = y_i + \sum_{j=1}^{d_v - 1} m_j \tag{3}$$

$$\Phi_c(m_1, m_2, \ldots, m_{d_c - 1}) = 2 \tanh^{-1} \left(\prod_{j=1}^{d_c - 1} \tanh \left(\frac{m_j}{2} \right) \right) \tag{4}$$

As one can see on equation (4), the check node update equation of the BP decoder is quite complicated and it is in practice not trivial to have a good implementation-friendly approximation of this update equation. To simplify the implementation of the check node update on hardware, the min-sum algorithm is usually prefered and described, for a parity-check node of degree d_c, as:

$$\Phi_c(m_1, \ldots, m_{d_c - 1}) = \left(\prod_{j=1}^{d_c - 1} \operatorname{sgn}(m_j) \right) \min_{j \in \{1, \ldots, d_c - 1\}} (|m_j|) \tag{5}$$

where sgn denotes the standard signum function.

The decision function Ψ used for these algorithms is the same as the one defined in previous Section. It has been further shown that when the output of the min-sum check node update equation (5) is corrected

by a simple constant offset, the performance of the min-sum algorithm approach closely the performance of BP in the high-SNR regime [24].

E. Trapping sets and trapping set ontology

Now that we have introduced the generalities about iterative decoders, let us dscuss in details what has been identified as their main weakness in terms of error correction, using the concept of trapping sets (TS) [4]. For a given decoder input $\mathbf{y} = \{y_1, y_2, \cdots, y_N\}$, a TS $\mathbf{T}(\mathbf{y})$ is a non-empty set of variable nodes that are not eventually corrected by the ID. A variable node v_i is said to be eventually correct if there exists a positive integer q such that when the decoder is run for all $l \geq q$ iterations, v_i is always decoded to the correct value. Note that if $\mathbf{T}(\mathbf{y})$ is empty, then the decoding is declared as successful. A standard notation commonly used to denote a trapping set is (a, b), $0 \leq a \leq A$, $0 \leq b \leq B$, where $a = |\mathbf{T}(\mathbf{y})|$, and b is the number of odd-degree check nodes present in the subgraph induced by $\mathbf{T}(\mathbf{y})$.

The Tanner graph representation of an (a, b) TS denoted by $\mathcal{T}$ is a subgraph induced by $\mathbf{T}(\mathbf{y})$ containing a variable nodes and b odd-degree check nodes. A code $\mathcal{C}$ is said to contain a TS of type $\mathcal{T}$ if there exists a set of variable nodes in G whose induced subgraph is isomorphic to $\mathcal{T}$, seen as a topological structure. Let $N_{\mathcal{T}}$ denote the number of trapping sets of type $\mathcal{T}$ that are contained in the code $\mathcal{C}$. Also for convenience we shall simply use $\mathbf{T}$ (instead of the more precise notation $\mathbf{T}(\mathbf{y})$) to refer to a particular subset of variable nodes in a given code that form a trapping set. Finally, let $\{\mathbf{T}_{i,\mathcal{T}} \mid i = 1, \ldots, N_{\mathcal{T}}\}$ be the collection of trapping sets of type $\mathcal{T}$ present in code $\mathcal{C}$. In other words, $\{\mathbf{T}_{i,\mathcal{T}}\}_i$ is a collection of all distinct subsets of variable nodes whose induced subgraphs are isomorphic to trapping sets of type $\mathcal{T}$.

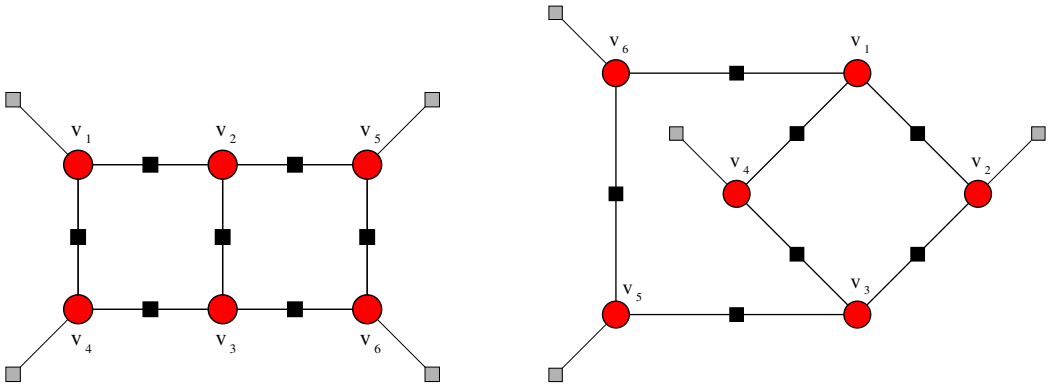
A TS is said to be elementary if $\mathcal{T}$ contains only degree-one or/and degree-two check nodes. It is now well established that the error floor phenomenon is dominated by the presence of elementary trapping sets in the Tanner graph of the code [4, 9]. Hence, we shall restrict our focus on elementary trapping sets throughout the paper.

Although the (a, b) notation is typically used in literature, this notation is not sufficient to uniquely denote a particular trapping set as there can be many trapping sets with different topological structures that share the same values of a and b. Distinguishing different TS with same values of a and b is an important issue, since the topological structure of a TS determines how harmful the TS is for the error floor of a given decoder [18]. In this paper, we propose to modify the TS notation by including additional parameters into the existing notation so that we are able to distinguish between distinct trapping sets that share the same values a and b. Since a notation which includes complete topological description of a

subgraph would be extremely complicated and too precise for our purpose, we introduce a simplified notation which only captures the cycle structure of the subgraph, and thus gives a cycle inventory of a trapping set.

Definition 1. *A trapping set is said to be of type* $(a, b; \prod_{k\geq 2}(2k)^{g_k})$ *if the corresponding subgraph contains exactly* g_k *distinct cycles of length* $2k$.

As an example, Fig.1 shows the two non-isomorphic $(a, b) = (6, 4)$ trapping sets for girth $g = 8$, $d_v = 3$ regular codes: (6,4;$8^2 12^1$) and (6,4;$8^1 10^2$).



(a) (6,4;$8^2 12^1$) (b) (6,4;$8^1 10^2$)

Figure 1. Two non-isomorphic trapping sets $\mathcal{T}$ with $(a, b) = (6, 4)$. The (6,4;$8^2 12^1$) trapping set (Fig. 1(a)) is composed of two 8-cycles ($v_1 - v_2 - v_3 - v_4 - v_1$ and $v_3 - v_4 - v_5 - v_6 - v_3$) and one 12-cycle ($v1 - v_2 - v_5 - v_6 - v_3 - v_4 - v_1$), while the (6,4;$8^1 10^2$) trapping set (Fig. 1(b)) has one 8-cycle ($v_1 - v_2 - v_3 - v_4 - v_1$) and two 10-cycles ($v_1 - v_2 - v_3 - v_5 - v_6 - v_1$ and $v_1 - v_4 - v_3 - v_5 - v_6 - v_1$).

Our choice of notation appears to be sufficient for differentiating between the topological structures of mutiple (a, b) trapping sets, and also includes the definition of codewords of $\mathcal{C}$. The $(a, 0)$ trapping sets corresponds to a codeword of weight a. In a given code there may be different types of $(a, 0)$ codewords having different cycle structure. For example, as we discuss later, the $(N = 155, K = 62, d_{min} = 20)$ Tanner code [1] contains minimum weight codewords with three different cycle inventories: $(20, 0; 8^4 10^{11} 12^8 14^{12} 16^{15})$, $(20, 0; 8^6 10^7 12^{10} 14^{17} 16^{31})$, and $(20, 0; 8^5 12^5 16^{40})$.

The trapping sets are typically fixed points of iterative decoders, in the sense that if the decoder is initialized on the BSC with errors exactly located in $\mathbf{T}$, the decoder is stuck and cannot converge to a codeword (right or wrong codeword). Moreover, following the interpretation of iterative decoders as dynamical systems [32, 33], trapping sets act as *attractor points* for an ID, and error patterns which are

close but not identical to $\mathbf{T}$ could also be attracted to $\mathbf{T}$ [4]. Although the trapping sets are always the support of patterns for which the ID fails to converge in the low error-floor region, on the other hand, the number of bit errors and the locations of the errors which lead to an iterative decoding failure depend on the ID update equations [16].

Instead of viewing this dependence as a disadvantage, we may exploit it for designing improved iterative decoders as well as for constructing codes with better error floor performance as shown in [7] and [19, 18]. The concept of trapping set ontology (TSO) was introduced in order to form a database of trapping sets constructed systematically, and to provide a classification of the trapping sets based on their topological relations. The TSO can be used for easy enumeration of certain types trapping sets present in a given code, for estimation of the error floor performance of a given code, for constructing good codes, and for designing better decoders [18].

III. Finite Alphabet Iterative Decoders

The methodology preseted in this paper relies on the recently introduced type of finite alphabet iterative decoders [7, 8]. These new message passing decoders have much lower complexity compared to existing message passing decoders, while having the potential to surpass — in the error floor region — the classical decoders like the floating-point BP or the min-sum decoder. Additionnally, the FAID framework allows us to define a plurality of powerful iterative decoders having different decoding dynamics for particular error pattern realizations, which we make use in this paper to introduce the concept of *decoder diversity*.

A. General Definition of FAID

An N_s-level FAID $\mathbf{D}$ is a 4-tuple given by $\mathbf{D} = (\mathcal{M}, \mathcal{Y}, \Phi_v, \Phi_c)$. The messages are levels confined to a finite alphabet $\mathcal{M} = \{-L_s, \ldots, -L_2, -L_1, 0, L_1, L_2, \ldots, L_s\}$ consisting of $N_s = 2s + 1$ levels, where $L_i \in \mathbb{R}^+$ and $L_i > L_j$ for any $i > j$. The number of precision bits required for representation of the messages during implementation is $\lfloor \log_2(N_s) \rfloor$. For instance, a FAID with $N_s = 7$ levels requires 3 bits.

As in the previous sections, we interpret the sign of a message $x \in \mathcal{M}$ to represent the value of the bit (positive for zero and negative for one) associated with the variable node, and the magnitude $|x|$ to reflect a measure of how reliable this value is. It must be noted however that $|x|$ does not represent the quantized value of a log-domain reliability (*a posteriori* log-likelihood ratio) nor does it represent a quantized probability. The message 0 in the alphabet can be interpreted as an erasure message.

The set $\mathcal{Y}$ denotes the set of possible channel values, which reduces to only two values $\mathcal{Y} = \{\pm \mathrm{C}\}$ in the case of BSC. The channel value $y_i \in \mathcal{Y}$ is determined by $y_i = (-1)^{r_i}\mathrm{C}$ where r_i is the value received from the channel at variable node v_i, i.e., we use the mapping $0 \rightarrow \mathrm{C}$ and $1 \rightarrow -\mathrm{C}$. Let $m_1, m_2, \ldots, m_{l-1}$ denote the $l-1$ incoming messages of a node of degree l which are used in the calculation of the outgoing message, either check node or variable node.

The function $\Phi_c : \mathcal{M}^{d_c-1} \rightarrow \mathcal{M}$ is used as update rule at a check node with degree d_c and is defined as

$$\Phi_c(m_1, \ldots, m_{d_c-1}) = \left(\prod_{j=1}^{d_c-1} \mathrm{sgn}(m_j) \right) \min_{j \in \{1,\ldots,d_c-1\}} (|m_j|) \tag{6}$$

Note that this function is the same as the one used in the min-sum decoder except that the messages belong to a finite alphabet $\mathcal{M}$.

The function $\Phi_v : \mathcal{Y} \times \mathcal{M}^{d_v-1} \rightarrow \mathcal{M}$ is a map used as update rule at a variable node with degree d_v. Φ_v can be described as a closed form function (as given in [7]) or simply as a multidimensional array (look-up table). As a closed form function, Φ_v is defined as

$$\Phi_v(y_i, m_1, m_2, \cdots, m_{d_v-1}) = Q\left(\sum_{j=1}^{d_v-1} m_j + m_c \cdot y_i \right) \tag{7}$$

where the function $Q(.)$ is defined as follows based on a threshold set $\mathscr{T} = \{T_i : 1 \leq i \leq s+1\}$ such that $T_i \in \mathbb{R}^+$ and $T_i > T_j$ if $i > j$, and $T_{s+1} = \infty$.

$$Q(x) = \begin{cases} \mathrm{sgn}(x) L_i, & \text{if } \ T_i \leq |x| < T_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

The additive parameter m_c is computed using a symmetric function $\Omega : \mathcal{M}^{d_v-1} \rightarrow \{0, 1\}$, which can be linear or non-linear. Based on this definition, Φ_v can be described as a linear-threshold (LT) or non-linear-threshold (NLT) function. If Ω=constant, then it is an LT function, else it is an NLT function. Note that due to possible non-linearity in Ω, the FAID are different — and more general — from existing quantized BP or min-sum decoders. Alternatively Φ_v can be described as map in the form of a look-up table (LUT), which is more convenient to describe them and for practical hardware implementation.

Definition 2 (Symmetric decoder)**.** *If the update function Φ_v of a FAID satisfies the following condition*

$$\Phi_v(y_i, m_1, ..., m_{d_v-1}) = -\Phi_v(-y_i, -m_1, ..., -m_{d_v-1}) \tag{8}$$

then the FAID is a symmetric decoder.

From the above definition as well as from (7) that defines the map Φ_v, it is clear that FAID are symmetric decoders.

The decision function Ψ is again the same as the one defined for the sum-product and min-sum algorithms, i.e. decide whether the bit value of the variable node is 0 or 1 based on the sign of the sum of its arguments.

B. FAID for column-weight $d_v = 3$ LDPC codes

For column-weight three codes, the function Φ_v can be conveniently represented as a two dimensional array. Let us alternatively define $\mathcal{M}$ to be $\mathcal{M} = \{M_1, M_2, \cdots, M_{N_s}\}$ where $M_1 = -L_s$, $M_2 = -L_{s-1}$,$\cdots$, $M_s = -L_1$, $M_{s+1} = 0$, $M_{s+2} = L_2$,$\cdots$, $M_{N_s} = L_s$. In this case, the function Φ_v can be represented by an array $[l_{i,j}]_{1\leq i,j\leq N_s}$, $l_{i,j} \in \mathcal{M}$, so that $\Phi_v(M_i, M_j, -\mathrm{C}) = l_{i,j}$ for any $M_i, M_j \in \mathcal{M}$. The values for $\Phi_v(M_i, M_j, +\mathrm{C})$ can be deduced from the symmetry of Φ_v. Note that given a set $\mathcal{M}$ and $\mathcal{Y}$, since Φ_c is fixed for any N_s-level FAID, the function Φ_v uniquely specifies the N_s-level FAID.

In the array representation of Φ_v, a particular choice of $[l_{i,j}]_{1\leq i,j\leq N_s}$ gives rise to a particular Φ_v. It is clear from this representation that there is a huge number of N_s-level FAIDs to select from. To make it managable, we first disregard the rules that do not exhibit a "typical" behavior observed in soft message passing decoders. This typical behavior implies that for a given y_i at a node v_i, the magnitude of the outgoing message should always increase with the increase of magnitude of the incoming messages. Intuitively, this makes sense since we would expect a node to send messages with an increased reliability if it is receiving messages with higher reliability as long as all of them are conforming to the same sign. For instance, if $\Phi_v(L_1, L_2, -\mathrm{C}) = L_2$, then all the FAIDs with $\Phi_v(L_2, L_2, -\mathrm{C}) = L_1$ are forbidden since $\Phi_v(L_2, L_2, -\mathrm{C})$ should be no less than $\Phi_v(L_1, L_2, -\mathrm{C})$. To incorporate this property into FAID rules, we specify an ordering constraint, called *lexicographic ordering* as follows.

Definition 3 (Lexicographic ordering)**.** *A lexicographically ordered FAID for column weight three codes is a decoder whose variable node update function Φ_v satisfies the following property.*

$$|\Phi_v(-\mathrm{C}, |m_1|, |m_2|)| \geq \left|\Phi_v(-\mathrm{C}, \left|m_1'\right|, \left|m_2'\right|)\right| \tag{9}$$

for all $|m_1| \geq |m_1'|$ and $|m_2| \geq |m_2'|$.

The ordering significantly reduces the number of FAID considered, but still leaves many good FAIDs as candidates in the selection procedure.

Definition 4 (Class-A FAID)**.** *A Class-A FAID is a symmetric lexicographically ordered FAID for column weight three codes.*

In this paper, we shall only consider the set of FAID consisting of all possible Class-A FAID. A natural question that arises at this point is the number of Class-A FAIDs. The total number of all possible Class-A FAID with N_s levels is given by the following theorem.

Theorem 1 (Number of Class-A FAID)**.** *The total number $K_A(N_s)$ of symmetric lexicographically ordered N_s-level FAID is given by*

$$K_A(N_s) = \frac{H_2(3N_s)H_1(N_s)H_2(N_s-1)}{H_2(2N_s+1)H_1(2N_s-1)} \tag{10}$$

where $H_k(n) = (n-k)!\,(n-2k)!\,(n-3k)!\ldots$ is the staggered hyperfactorial function.

Proof: The proof of the theorem is given in Appendix A and follows from the isomorphism between the Class-A FAID and symmetric boxed plane partitions. Kuperberg [34] gave an elegant formula for the enumeration of symmetric plane partitions contained in a box of dimensions (r, r, t) leading to Eq. 10.

The total number of Class-A FAID for $N_s = 5$, $N_s = 7$, and $N_s = 9$ levels are:

$$K_A(5) = 28\,314$$
$$K_A(7) = 530\,803\,988$$
$$K_A(9) = 230\,316\,871\,499\,560$$

Such an enormous number of decoders requires an efficient systematic procedure to separate good decoders from bad ones. By "good" we mean good in the the error floor region. The existence of such a procedure is difficult to conjecture, and in our previous work we have shown with counter-examples that the selecton process cannnot rely neither on density-evolution thresholds alone [16, 21] nor on the sole analysis of trapping sets viewed as combinatorial objects. Indeed, for the FER analysis on the BSC, one needs to consider the notion of a critical number [31, 31] of a trapping set that was introduced to characterize the contribution of a given TS to the error floor. The concept of a critical number is motivated by the fact that not all variable nodes in a TS need to be in error for decoder to fail on this TS. Thus, the critical number corresponds to the minimum number of errors located inside the TS, leading to a decoding failure. The computation of the critical number on a (a, b) trapping set is done under a so-called *isolation assumption* [7]. Loosely, the isolation assumption provides a conditions under which a TS isolated from rest of the Tanner graph has the same dynamics as when it is not isolated, and is a generalization of the Gallager's *independence* assumption. Although the isolation assumption provides a

convenient way to analyze the ID decoding failures of trapping sets, it does not take into account the messages coming from the neighborhood of the trapping set. Consequently, the critical numbers of the smallest TS may not adequately reflect the behavior of a decoder on the whole graph, not even in the error floor region. In [16], we presented some evidence that the sole use of critical numbers to select good decoders could be misleading.

In [21], we have proposed a method for selecting good Class-A FAID decoders based on a heuristic discriminating between good and bad FAID rules. We introduced a concept of *noisy trapping sets*, i.e., trapping sets containing deliberately introduced noisy messages in their direct neighborhood. The associated heuristic is defined as a vector of *noisy critical numbers*, which better reflects the average error correction capability of a decoder when a noisy neighborhood of the TS is assumed. Using statistics on the noisy critical numbers, we proposed an iterative greedy selection algorithm to select good FAID and reject bad FAID. From this selection algorithm, we were able to select 5 FAID decoders with $N_s = 5$ levels among the $K_A(5) = 28314$ decoders, and $N_f = 70$ FAID decoders with $N_s = 7$ levels among a subset of $6\,575\,972$ decoders. Each of the selected decoders were found to have better error correction performance than the floating point BP in the error floor for a range of regular $d_v = 3$ LDPC codes.

IV. Decoder Diversity

A. Decoder Diversity Principle

In this section, we rigorously introduce the concept of *decoder diversity*, which is based on the fact that we have at our disposal a large number of FAID rules, each of which could serve as an iterative decoding algorithm. From section III-B, we have seen that the concept of FAID allows the definition of a very large number of possible decoders, which could be combined to increase the error correction capability of an LDPC code under iterative decoding.

Let us assume that we have at our disposal a set of Class-A N_s-level FAID denoted by

$$\mathcal{D} = \left\{ \left(\mathcal{M}, \mathcal{Y}, \Phi_v^{(i)}, \Phi_c \right) \mid i = 1, \ldots, N_{\mathcal{D}} \right\} \tag{11}$$

where each $\Phi_v^{(i)}$ is defined as in Equation (7), and Φ_c is the check node rule defined in Equation (6). We refer to this set $\mathcal{D}$ as a *decoder diversity set*, and an element of this set is denoted by D_i where $\mathrm{D}_i = \left(\mathcal{M}, \mathcal{Y}, \Phi_v^{(i)}, \Phi_c \right)$.

The question we address is to determine whether the decoders in the set $\mathcal{D}$ could be used in combination (either sequentially or parallelly) in order to guarantee the correction of all error patterns up to a certain

weight t. We need first to put notations on the collection of error patterns that each decoder is able to correct separately. Let $\mathcal{E}$ denote an arbitrary set of error patterns on a code $\mathcal{C}$ whose Tanner graph is G, *i.e.* a set of non-zero vectors $\mathbf{e}$ which serve as initialization of the decoders. Now, let $\mathcal{E}_{\mathrm{D}_i} \subseteq \mathcal{E}$ be the subset of error patterns that are correctable by a given FAID D_i, alone.

Definition 5. *We say that the set of error patterns $\mathcal{E}$ is correctable by a decoder diversity set $\mathcal{D}$ if*

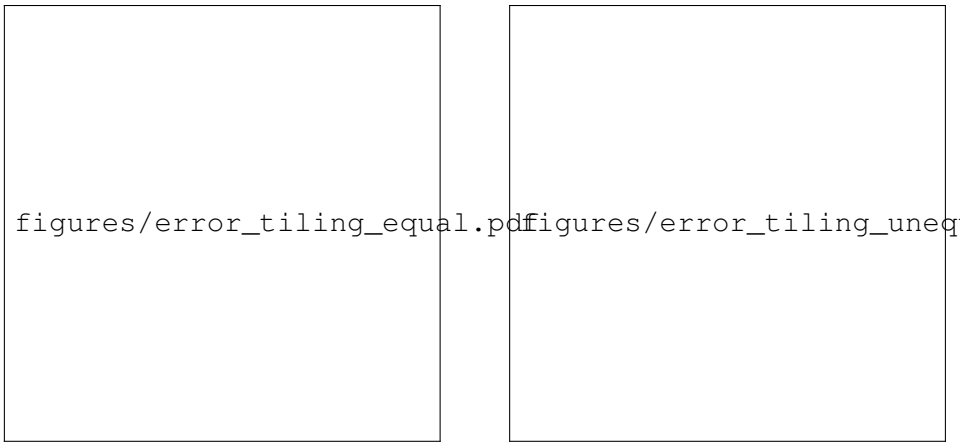
$$\mathcal{E} = \bigcup_{i=1}^{N_{\mathcal{D}}} \mathcal{E}_{\mathrm{D}_i}$$

Note that here we do not put a limit on the maximum number of decoding iterations of each decoder D_i, this issue will be discussed for the practical example of the Tanner code in section V. Given a set of error patterns up to a certain weight t on a code, one would like to determine the smallest decoder diversity sets that can correct all such error patterns. This problem is known as the Set Covering Problem, and is NP-hard [29]. In this paper, instead of looking for the smallest number of decoders which covers the error patterns set, we propose a algorithm for the selection of the decoder divesity set, based on some heuristics, which results in $\mathcal{D}$ larger than the minimal set, but still improves greatly the guaranteed error correction of a given code.

Note that in the definition of a decoder diversity set, we do not make any assumption on the cardinalities of each correctable subset $\mathcal{E}_{\mathrm{D}_i}$. Typically, strong decoders have large correctable subsets $\mathcal{E}_{\mathrm{D}_i}$, while other decoders which are selected to correct very specific error patterns, could have a small correctable subset. There are different ways to compose a set $\mathcal{D}$ from $\mathcal{D}_i$'s in order to cover the set $\mathcal{E}$ with the sets $\mathcal{E}_{\mathrm{D}_i}$. Two distiguished ways are illustrated in Figure 2. Figure 2(a) shows a case where the set of error events $\mathcal{E}$ (represented as a big square) is paved with nearly equally powerful decoders (smaller overlapping squares of similar sizes). Figure 2(b) shows another type of covering corresponding to using one strong decoder and a number of weaker decoders (smaller rectangles) dedicated to "surgical" correction of specific error patterns not correctable by the strong decoder.

B. Error Sets

As mentioned previously, our main goal is to increase the guaranteed error correction of a code using decoder diversity, that is for a given LDPC code $\mathcal{C}$ whose Tanner graph is G, we would like to find a small decoder diversity set $\mathcal{D}$ which guarantees correction of a fixed number of errors t. Since we are required to determine the correctable subsets for each Class-A N_s-level FAID that is considered



(a) (b)

Figure 2. Typical ways in which decoder diversity can correct all error patterns from a pre-determined set $\mathcal{E}$.

for possible inclusion into the set $\mathcal{D}$, it is important to address the issue of which error sets should be considered for assessing the error correction capability of a given FAID.

Let G' be a subgraph of G with the set of variable nodes $V' \subseteq V$. Denote by $\mathcal{E}^k(G')$ the set of all error patterns of weight k whose support lies entirely in the variable node set of subgraph G':

$$\mathcal{E}^k(G') = \{\mathbf{e} : w(\mathbf{e}) = k,\ \text{supp}(\mathbf{e}) \subseteq V'\} \tag{12}$$

Note that $\mathcal{E}^k(G)$ denotes the set of all error patterns of weight k on the code $\mathcal{C}$. For simplicity, we shall denote this particular set as $\mathcal{E}^k$ instead of $\mathcal{E}^k(G)$. Also let $\mathcal{E}^{[t]} = \bigcup_{k=1}^{t} \mathcal{E}^k$ denote the set of all error patterns whose weight is no larger than t.

A brute force approach to ensure a t-guaranteed error correction capability requires to consider all the error patterns in the set $\mathcal{E}^{[t]}$, the target error set that we wish eventually to "pave", and find a decoder diversity set $\mathcal{D}$ which corrects the errors in $\mathcal{E}^{[t]}$. Obviously, the cardinality of such error pattern sets are too large for a practical analysis. Instead, let us consider smaller error pattern sets, based on the knowledge of the trapping set distribution of the code $\mathcal{C}$. It is reasonable to assume that the errors patterns that are the most difficult to correct for the iterative decoders are patterns whose support is concentrated in the topological neighborhood of trapping sets.

Recall that $\{\mathbf{T}_{i,\mathcal{T}} \mid i = 1, \ldots, N_{\mathcal{T}}\}$ denotes the collection of all (a, b) trapping sets of type $\mathcal{T}$ that are present in code $\mathcal{C}$. Let $\mathcal{E}^k(\mathcal{T})$ denote the set of error patterns of weight k whose support lies in a (a, b) trapping set $\mathbf{T}_{\mathbf{i},\mathcal{T}}$ of type $\mathcal{T}$. More precisely,

$$\mathcal{E}^k(\mathcal{T}) = \{\mathbf{e} : w(\mathbf{e}) = k, \text{supp}(\mathbf{e}) \subseteq \mathbf{T}_{i,\mathcal{T}} \quad i \in \{1, \ldots, N_{\mathcal{T}}\}\} \tag{13}$$

The cardinality of $\mathcal{E}^k(\mathcal{T})$ is given by

$$|\mathcal{E}^k(\mathcal{T})| = \begin{pmatrix} a \\ k \end{pmatrix} N_{\mathcal{T}} \tag{14}$$

Now, let $\Lambda_{a,b}$ denote the set of all TS of different types present in the code $\mathcal{C}$ that have the same parameters (a, b). The error sets $\mathcal{E}^k(\Lambda_{a,b})$ and $\mathcal{E}^{[t]}(\Lambda_{a,b})$ associated with $\Lambda_{a,b}$ are defined as follows:

$$\mathcal{E}^k(\Lambda_{a,b}) = \bigcup_{\mathcal{T} \in \Lambda_{a,b}} \mathcal{E}^k(\mathcal{T}) \tag{15}$$

$$\mathcal{E}^{[t]}(\Lambda_{a,b}) = \bigcup_{k=1}^{t} \mathcal{E}^k(\Lambda_{a,b}) \tag{16}$$

Finally, $\Lambda^{\{A,B\}}$ is the set containing all (a, b) trapping sets of different types for different values of $a \leq A$ and $b \leq B$

$$\Lambda^{(A,B)} = \bigcup_{0 \leq a \leq A,\ 0 \leq b \leq B} \Lambda_{a,b}$$

and its associated error set is

$$\mathcal{E}^{[t]}(\Lambda^{(A,B)}) = \bigcup_{0 \leq a \leq A,\ 0 \leq b \leq B} \mathcal{E}^{[t]}(\Lambda_{a,b}) \tag{17}$$

Clearly, $\mathcal{E}^{[t]}(\Lambda^{(A,B)}) \subseteq \mathcal{E}^{[t]}$, and the cardinality of the latter error set can be further reduced by taking into account homomorphism properties imposed by a specific LDPC code design. Quasi-cyclic codes are the prime example. To enable parallelization of the decoder hardware, most LDPC codes that are used in practice are build from protographs expansions, and are therefore quasi-cyclic in nature. As a result, Tanner graph of such codes possess many TSs that are not only isomorphic in the sense of their topological structure, but also have identical neighborhoods. Therefore suffice to consider error patterns associated with any one of these isomorphic topologies rather than considering all of them. Some authors have proposed LDPC code constructions which have even more structural properties than just the quasi-cyclicity. An example of this construction is the Tanner code we mentioned in the Introduction [12]. For this particular design, the existence of three homorphism groups reduces by several orders of magnitude the number of TSs of maximum size (A, B) that need to be considered. We will study in detail a code of this type in section V.

To conclude this section on error sets, we provide a conjecture that we found valid for the test cases that we have analyzed, and which gives a criterion to choose the values of A and B that one wants to consider for the final error set that will be used for the diversity decoder set selection algorithm. From the standpoint of the computational complexity, it is indeed important to appropriately limit the maximum size of the trapping sets that are included in the set $\Lambda^{(A,B)}$. The first remark concerns the choice of B. Typically it has been observed that — in the case of regular $d_v = 3$ LDPC codes — most harmful (a, b) trapping sets have small values of b. Note that this is not anymore the case for LDPC codes with $d_v = 4$, as explained with the concept of absorbing sets in [26]. In addition, the value of A should be chosen depending on the value of guaranteed error correction capability t that we are trying to achieve, and based on the following conjecture.

Conjecture 1. *If there exists a decoder diversity set $\mathcal{D}$ that corrects all patterns in the set $\mathcal{E}^{[t]}(\Lambda^{(A,B)})$ on the isolated trapping sets $\Lambda^{(A,B)}$, with $A = 2t$, and sufficiently large B, then the decoder diversity set $\mathcal{D}$ will also correct all error patterns up to weight t on the code $\mathcal{C}$ with high probability.*

The above conjecture is analogous to the condition for correcting t errors by the MLD, which requires that the Hamming weight of error patterns shell be lower than $\lfloor d_{min}/2 \rfloor$. Put differently, if a decoder $\mathrm{D}_i \in \mathcal{D}$ cannot correct t errors which are concentrated on a TS of size smaller than $2t$, then it cannot correct more scattered weight-t error patterns either. That is, the decoder cannot correct the error patterns at a Hamming distance larger than t from the support of a considered TS.

At the present stage of this work, we have not found any counter-example, but have not been able to prove the conjecture. We have analyzed numerous codes, and in this paper we present the the results for the $(N = 155, K = 62, d_{min} = 20)$ Tanner code (in Section V).

Based on the above conjecture, we now see that considering the set $\mathcal{E}^{[t]}(\Lambda^{(A,B)})$ instead of $\mathcal{E}^{[t]}$ may be sufficient for determining the decoder diversity set ensuring t-guaranteed error correction capability, while at the same time reducing sufficiently the cardinality of the error sets the selection procedure begins with. The proposed method has reasonable computational complexity and is applicable to medium-length LDPC codes and FAIDs up to 9 levels.

C. Generation of FAID diversity sets

We now present the procedure for selecting the FAID diversity set $\mathcal{D}^{[t]}$ capable of correcting all error patterns in the set $\mathcal{E}^{[t]}(\Lambda^{(A,B)})$. In other words, we would like to determine a small (possibly smallest) decoder diversity set that can ensure a guaranteed correction of t errors on a given code.

Our approach for selecting the diversity sets is as follows. Let us assume that we are given a large set $\mathcal{D}_{base}$ which contains candidate FAID that are potentially good on any given code (i.e. they have potential to surpass BP in the error floor). This set could be obtained from simulations on different codes or by using a selection technique such as the methodology based on *noisy trapping sets* presented in [21]. Our goal is to select $\mathcal{D}^{[t]}$ from $\mathcal{D}_{base}$. In essence, the procedure runs over all error patterns up to weight t and tests their correctability when decoded by different decoders from $\mathcal{D}_{base}$. The algorithm iteratively expands the sets $\mathcal{D}^{[1]}, \mathcal{D}^{[1]} \dots \mathcal{D}^{[t]}$ by including in the diversity set the decoders capable of correcting larger and larger error patterns. At the same time, the algorithm is bookkeeping the set of $\mathcal{E}^r$ of error patterns whose correctability is undecided. The algorithm halts when $\mathcal{E}^r = \emptyset$, which means that for the given choice of t and N_I, the decoder diversity set $\mathcal{D}^{[t]}$ guarantees correction of t errors on the error set $\mathcal{E}^{[t]}(\Lambda^{(A,B)})$, which in turns guarantees correction of t errors on the LDPC code $\mathcal{C}$ in a maximum of N_I iterations — assuming the Conjecture 1 is valid. As a side result, the selection algorithm gives also the diversity sets $\mathcal{D}^{[k]}$ for $k < t$, which are obtained at each iterative stage of the algorithm.

To formally define the algorithm, let us denote by $\mathcal{E}^r \subset \mathcal{E}^{[t]}(\Lambda^{(A,B)})$ the set of unresolved error patterns during the selection algorithm. If $\mathcal{E}^{[t]}_{\mathrm{D}_i}$ is the set of error patterns that are correctable by the decoders in $\mathrm{D}_1, \mathrm{D}_1, \dots, \mathrm{D}_l$, that are selected and included in $\mathcal{D}^{[t]}$, then the set of unresolved error patterns set is

$$\mathcal{E}^r = \mathcal{E}^{[t]}(\Lambda^{(A,B)}) \backslash \bigcup_{1 \le i \le l} \mathcal{E}^{[t]}_{\mathrm{D}_i}$$

The iterative selection algorithm is now described as follows:

Note that in addition to D_{base}, the algorithm takes also the maximum number of iterations, N_I, as an input. The number of iterations required to to correct a trapping set varies across error patterns $\mathcal{E}^{[t]}(\Lambda^{(A,B)})$, and different choices of N_I lead to different identified decoder diversity sets. Determining optimal number of iterations is beyond scope of this paper, but it will be discussed in details in the next section for our test case.

In any case, if for some t and N_I, the algorithm ends with $\mathcal{E}^r = \emptyset$ after all error patterns have been tested for correctability, i.e., when $k = t$ is reached, then the decoder set $\mathcal{D}^{[t]}$ does not contain a single decoder that guarantees correction of all t errors on the error set $\mathcal{E}^{[t]}(\Lambda^{(A,B)})$. This means that assuming that the assumptions of the Conjecture 1 are valid, the diversity set cannot correct all t errors in the LDPC code $\mathcal{C}$ in a maximum of N_I iterations. As a side result, the selection algorithm gives also the diversity sets $\mathcal{D}^{[k]}$ for $k < t$, which are obtained at each iterative stage of the algorithm.

For example, consider the case of $t = 7$-guaranteed error correction capability on a code $\mathcal{C}$. Suppose the selection algorithm has already tested correctabilty of shorter error patterns, and is currently in the

Algorithm 1 Decoder Diversity Selection Algorithm

1) Given $\mathcal{D}_{base}$ and N_I, set $\mathcal{D}^{[k]} = \emptyset$, $1 \leq k \leq t$.
   Initialize k, set parameters $(A, B) = (2k, B)$ based on conjecture 1. Set $\mathcal{E}^r = \mathcal{E}^k(\Lambda^{(A,B)})$,
2) set $\mathcal{D}^k = \emptyset$ and $i = 1$,
   a) run all FAID in $\mathcal{D}_{base}$ initialized with error patterns in $\mathcal{E}^r$ for a maximum of N_I iterations and select the FAID D_i which corrects the largest number of error patterns, *i.e.* $\left|\mathcal{E}^r_{\mathrm{D}_i}\right|$ is maximum. Put D_i in $\mathcal{D}^k$, i.e., $\mathcal{D}^k = \mathcal{D}^k \cup \mathrm{D}_i$,
   b) remove all error patterns corrected by D_i from the set $\mathcal{E}^r$, i.e. $\mathcal{E}^r = \mathcal{E}^r \backslash \mathcal{E}^r_{\mathrm{D}_i}$.
   c) Set $i = i + 1$, go to 2) a) and repeat until $\mathcal{E}^r = \emptyset$:
3) set $\mathcal{D}^{[k]} = \mathcal{D}^{[k]} \bigcup \mathcal{D}^k$,
4) set $k = k + 1$, and $(A, B) = (2k, B)$. Set $\mathcal{D}^{[k]} = \mathcal{D}^{[k-1]}$,
   a) $\forall\ \mathrm{D}_i \in \mathcal{D}^{[k]}$, determine the correctable subsets of k-error patterns by decoders D_i denoted by $\mathcal{E}^k_{\mathrm{D}_i}(\Lambda^{(A,B)})$,
   b) set $\mathcal{E}^r = \mathcal{E}^k(\Lambda^{(A,B)}) \backslash \bigcup_{\mathrm{D}_i \in \mathcal{D}^{[k]}} \mathcal{E}^k_{\mathrm{D}_i}(\Lambda^{(A,B)})$,
   c) go to 2) and repeat until $k = t$

stage when $k = 5$. The algorithm is now testing the 5-error patterns and, based on the Conjecture 1, sets $(A, B) = (10, 4)$. After one iteration of step 2) in the selection algorithm, we get a set $\mathcal{D}^{[5]}$ of FAIDs, that corrects all 5-error patterns in $\mathcal{E}^5(\Lambda^{(10,4)})$, for which the support is included in TS with parameters $(a \leq 10, b \leq 4)$. Then, for $k = 6$, step 4) removes from $\mathcal{E}^6(\Lambda^{(12,4)})$ the error patterns correctable by the decoders already selected in $\mathcal{D}^{[5]}$. We then proceed with the second iteration of the selection algorithm with the remaining undecided error patterns in $\mathcal{E}^r$, *i.e.* the error patterns whose correctability is to be tested by other decoders $\mathrm{D}_i \notin \mathcal{D}^{[5]}$. After the second instance of step 2), we obtain a larger set of decoders $\mathcal{D}^{[6]}$ ($\mathcal{D}^{[5]} \subset \mathcal{D}^{[6]}$) which corrects all error patterns in $\mathcal{E}^{[6]}(\Lambda^{(12,4)})$. Finally, after the third iteration, we eventually obtain a set of decoders $\mathcal{D}^{[7]}$, such that $\mathcal{D}^{[5]} \subset \mathcal{D}^{[6]} \subset \mathcal{D}^{[7]}$, which corrects all error patterns in $\mathcal{E}^{[7]}(\Lambda^{(14,4)})$. Based on the conjecture 1, the set of decoders $\mathcal{D}^{[7]}$ ensures correction of all k-error patterns with $k \leq t = 7$.

Note that during the iterations of the selection algorithm, if $\mathcal{E}^r \neq \emptyset$ even after $i = |\mathcal{D}_{base}|$, then it implies that even the entire set of candidate FAID $\mathcal{D}_{base}$ with N_I being the maximum number of iterations, is not sufficient to correct all k-error patterns. Under such a scenario, in order to allow the algorithm to progress, one can increase the maximum number of iterations N_I allowed for each decoder in $\mathcal{D}_{base}$ or consider a larger set of candidate decoders $\mathcal{D}_{base}$ in the diversity selection algorithm. We

used both these strategies in order to obtain a diversity set with $t = 7$ guaranteed error correction on the case study presented in the next section.

V. Case Study: Guaranteed error correction on the $(N = 155, K = 62, D_{min} = 20)$ Tanner Code

In this paper, we shall use the $(N = 155, K = 62, D_{min} = 20)$ Tanner code [11, 12], as an example to illustrate how the concept of decoder diversity can be used to increase the guaranteed error-correction capability of the code with reasonable complexity.

This code is a regular LDPC code with the column weight $d_v = 3$, row weight $d_c = 5$, and is a particularly good test case for the following reasons. First, the difference between its minimum distance $d_{min} = 20$ and its minimum pseudo-distance $w_p^{min} \simeq 10$ is very large, which means that the difference in the error capability between standard iterative decoders (Gallager-B, min-sum, BP) and MLD from one side and our diversity framework on the other is expected to be large. Another reason is that the $(N = 155, K = 62, d_{min} = 20)$ Tanner code is sufficiently small and structured (the code is quasi-cyclic with bloc-cyclicity equal to 31) such that brute force search for uncorrectable patters can be used to verify some claims by Monte Carlo simulations.

Table I shows the t-guaranteed error correction capability of the existing algorithms on the Tanner code.

Table I
A t-guaranteed error correction on the $(N = 155, K = 62, D_{min} = 20)$ Tanner Code

t	Algorithm	Reference
3	Gallager A and Gallager B	[19]
4	Min-Sum and Belief Propagation	[7]
5	5-level and 6-level FAID	[16]

We also found by simulations [21] that there are no 7-level FAID among the set of all possible decoders that can guarantee a correction of more than $t = 5$ on this particular code. However, using the approach of decoder diversity, we show that it is possible to increase the guaranteed error correction capability of the code to $t = 7$ with an appropriate choice of the decoder diversity set.

As mentioned in the previous section, we only consider error patterns belonging to $\mathcal{E}^{[t]}(\Lambda^{(A,B)})$ where $A = 2t$ and B small enough. We verified by simulations that the value of $B = 4$ was sufficient to determine the decoder diversity set $\mathcal{D}^{[t]}$.

For the case of the Tanner code, the structure of its Tanner graph satisfies certain structural properties — apart from the block-cyclicity property which comes from the design based on circulants — which allows further reduction in the number of error patterns considered. The different homomorphism groups that the Tanner graph of this code follows are presented in [11], and are reported in this paper in the Appendix B. Following notations of [11], the transformations σ, π, and ρ act on the indices of the variable nodes and preserve the topological structures. The transformation σ comes from the quasi-cyclicity of the code and allows then a constant reduction factor of $L = 31$ for all the TS topologies, while the other transformations π and ρ can bring another factor of reduction, depending on the type and location of the TS. The full enumeration of TS with $a \leq 14$ and $b \leq 4$ in presented in Table II. The first column of the table gives the (a, b) parameters, and the second column indicates the TS cycle inventory of different (a, b) TS types (the cycle inventory is omitted for the parameters that allow too many cycle-inventory types). The last three columns show the numbers of TS that need to be checked by the Algorithm **??** when the code automorphisms are exploited. $\sigma(\mathcal{T})$, corresponds to the naive approach which uses no knowledge of the code structure. $\sigma(\mathcal{T})$ corresponds to knowledge of cyclic transformation σ, and $\sigma(\pi(\rho(\mathcal{T})))$ to all three transformations σ, π and ρ. The small section of the table at the bottom shows the structure and number of the lowest weight codewords of different types.

These trapping sets have been enumerated using the modified impulse algorithm, which is known as the most efficient algorithm to find low-weight codewords or near-codewords of a given short length LDPC code [14, 22]. When the number of types was too large, we did not indicate the details of the TS notation. It is clear from the Table that the number of topologies which needs to be considered to characterize the behavior of an iterative decoder on the Tanner code could be greatly reduced. Actually, the number of structures (including isomorphic) of given type $\mathcal{T}$ present in the code could be either $L\,d_c\,d_v = 465$, $L\,d_c = 155$ or $L\,d_v = 93$ and this number is reduced for the analysis by the transformations $\sigma(\pi(\rho(\mathcal{T})))$. The TS of type (5,3;$8^3$) is an example where there are $L\,d_c = 155$ such structures in the Tanner code , while (20,0)-type-III codewords is an example where there are $L\,d_v = 93$ such structures.

A. Error Sets for the Tanner Code

The error sets that we have considered for the Tanner code are shown in Table III. We have indicated in this table the cardinalities of the error sets, which have been reduced by the structural properties σ, π, and ρ of the Tanner code to:

$$|\mathcal{E}^k(\mathcal{T})| = \left(\begin{array}{c} a \\ k \end{array} \right) \; N_{\sigma(\pi(\rho(\mathcal{T})))}. \tag{18}$$

Table II

TRAPPING SET SPECTRUM AND LOW-WEIGHT CODEWORDS SPECTRUM OF THE $(N = 155, K = 62, D_{min} = 20)$ TANNER CODE

$\mathcal{T}$	TS-label	$N_{\mathcal{T}}$	$N_{\sigma(\mathcal{T})}$	$N_{\sigma(\pi(\rho(\mathcal{T})))}$
(5,3)	$(5,3;8^3)$	**155**	**15**	**1**
(6,4)	$(6,4;8^1 10^2)$	**930**	**30**	**2**
(7,3)	$(7,3;8^3 10^2 14^2)$	**930**	**30**	**2**
(8,2)	$(8,2;8^3 10^4 12^2 14^4 16^2)$	**465**	**15**	**1**
(8,4)	4 types	**5012**	**165**	**11**
	$(8,4;8^3 12^2 16^2)$		45	3
	$(8,4;8^1 10^2 12^2 14^2)$		15	1
	$(8,4;8^1 10^3 12^1 14^1 16^1)$		90	6
	$(8,4;8^1 10^4 16^2)$		15	1
(9,3)	3 types	**1860**	**60**	**4**
	$(9,3;8^1 10^4 12^4 14^2 16^2 18^2)$		15	1
	$(9,3;8^1 10^5 12^2 14^2 16^4)$		30	2
	$(9,3;8^3 10^2 12^2 14^4 16^2 18^2)$		15	1
(10,2)	2 types	**1395**	**45**	**3**
	$(10,2;8^1 10^6 12^5 14^4 16^6 18^5 20^2)$		15	1
	$(10,2;8^3 10^5 12^2 14^6 16^6 18^2 20^4)$		30	2
(10,4)	27 types	**29295**	**945**	**63**
(11,3)	11 types	**6200**	**200**	**14**
(12,2)	2 types	**930**	**30**	**2**
	$(12,2;8^1 10^6 12^6 14^6 16^9 18^9 20^8 22^7 24^4)$		15	1
	$(12,2;8^4 10^2 12^4 14^4 16^8 18^1 220^6 22^8 24^2)$		15	1
(12,4)	170 types	**196440**	**6240**	**416**
(13,3)	53 types	**34634**	**1155**	**79**

$\mathcal{T}$	TS-label	$N_{\mathcal{T}}$	$N_{\sigma(\mathcal{T})}$	$N_{\sigma(\pi(\rho(\mathcal{T})))}$
(20,0)	3 types	**1023**	**33**	**3**
	type-I	465	15	1
	type-II	465	15	1
	type-III	93	3	1
(22,0)	14 types	**6200**	**200**	**14**
(24,0)	97 types	**43865**	**1415**	**97**

where $N_{\sigma(\pi(\rho(\mathcal{T})))}$ is the value obtained from Table II.

One can further reduce the number of error patterns in each error set, since for example, a 5-error pattern on one of the TS $(9, 3)$ could be the same of one listed in the 5-error patterns in the TS $(8, 2)$. In order to limit at most the computational complexity of the decoder selection algorithm, we only put in the error sets $\mathcal{E}^{[k]}(\Lambda_{a,b})$ the error patterns which are distinct from all patterns in $\mathcal{E}^{[k]}(\Lambda_{a',b'})$ with $a' < a$

and $b' < b$. The final number of error patterns considered is reported at the bottom of Table III, together with the complexity reduction factor due to the use of $\mathcal{E}^{[t]}(\Lambda^{(A,B)})$ instead of $\mathcal{E}^{[t]}$ in the decoder selection algorithm. As we can see, the complexity reduction factor is in each case is of the order of 10^6, which is very large and in any case sufficient to reduce the complexity of finding the decoder diversity set to a reasonnable computational time.

Table III

CARDINALITIES OF ERROR SETS CONSIDERED FOR THE $(N = 155, K = 62, D_{min} = 20)$ TANNER CODE.

5-errors			6-errors			7-errors		
$\mathcal{E}^{[5]}(\Lambda_{5,3})$	1	1						
$\mathcal{E}^{[5]}(\Lambda_{6,4})$	12	12	$\mathcal{E}^{[6]}(\Lambda_{6,4})$	2	2			
$\mathcal{E}^{[5]}(\Lambda_{7,3})$	36	23	$\mathcal{E}^{[6]}(\Lambda_{7,3})$	14	11	$\mathcal{E}^{[7]}(\Lambda_{7,3})$	2	2
$\mathcal{E}^{[5]}(\Lambda_{8,2})$	56	20	$\mathcal{E}^{[6]}(\Lambda_{8,2})$	28	15	$\mathcal{E}^{[7]}(\Lambda_{8,2})$	8	6
$\mathcal{E}^{[5]}(\Lambda_{8,4})$	510	398	$\mathcal{E}^{[6]}(\Lambda_{8,4})$	308	240	$\mathcal{E}^{[7]}(\Lambda_{8,4})$	88	79
$\mathcal{E}^{[5]}(\Lambda_{9,3})$	448	100	$\mathcal{E}^{[6]}(\Lambda_{9,3})$	336	110	$\mathcal{E}^{[7]}(\Lambda_{9,3})$	144	72
			$\mathcal{E}^{[6]}(\Lambda_{10,2})$	630	416	$\mathcal{E}^{[7]}(\Lambda_{10,2})$	360	277
			$\mathcal{E}^{[6]}(\Lambda_{10,4})$	13230	7860	$\mathcal{E}^{[7]}(\Lambda_{10,4})$	7560	5421
			$\mathcal{E}^{[6]}(\Lambda_{11,3})$	6468	1980	$\mathcal{E}^{[7]}(\Lambda_{11,3})$	4620	1894
						$\mathcal{E}^{[7]}(\Lambda_{12,2})$	1584	857
						$\mathcal{E}^{[7]}(\Lambda_{12,4})$	329472	187360
						$\mathcal{E}^{[7]}(\Lambda_{13,3})$	135564	31890

5-errors		6-errors		7-errors	
$\mathcal{E}^{[5]}(\Lambda^{(9,4)})$	554	$\mathcal{E}^{[6]}(\Lambda^{(11,4)})$	11 829	$\mathcal{E}^{[7]}(\Lambda^{(13,4)})$	227 858
$\mathcal{E}^{[5]}$	698 526 906	$\mathcal{E}^{[6]}$	17 463 172 650	$\mathcal{E}^{[7]}$	371 716 103 550
Comp. Reduction Factor		Comp. Reduction Factor		Comp. Reduction Factor	
1 260 879		1 476 301		2 293 227	

B. Error Correction Results for the Tanner Code

Let us recall that we consider only 7-level FAID which require only 3 bits of precision for their message representation. Our main results can be summarized in the Table IV. Correcting all 7-error patterns on a Tanner code requires using $N_{\mathcal{D}} = 343$ FAIDs with run for no more than $N_I = 120$ iterations.

We were able to verify by brute force Monte Carlo simulations that each of the above diversity sets guarantees a correction of all error patterns of weight atmost t (for $t = 5, 6, 7$) on the Tanner code even though only error patterns in $\mathcal{E}^{[t]}(\Lambda^{(A,B)})$) were used in the algorithm, thus providing further evidence to the validity of the conjecture stated in Section IV-B.

Due to the huge complexity reduction in the consideration of error sets (as shown out in Table III), we were able to identify the decoder diversity set for $t = 6$ in less than one hour, and for $t = 7$ in a few

Table IV
A t-GUARANTEED ERROR CORRECTION DIVERSITY SCHEMES ON THE $(N = 155, K = 62, D_{min} = 20)$ TANNER CODE

t	$N_{\mathcal{D}}$	N_I
5	1	15
6	9	50
7	243	120

days. Note that decoder diversity does not require any post-processing, as it is still an iterative message passing decoder with the additional feature that the the variable node update rule Φ_v changes after N_I iterations (and the decoder is restarted). Nota also that additional complexity reduction can be achieved by exploiting the similarity of the update rules of the decoders in the diversity set.

Now let us have a look at how the decoder diversity set behaves on different error sets. We have reported in Table V-B the statistics of some FAID by computing the number of correctable error patterns associated with each decoder. The FAID rules of these decoders are reported on Table VI in Appendix B-A. The first part of the Table shows the values of $\left|\mathcal{E}^{[6]}(\Lambda^{(11,4)})\right|, \forall \mathrm{D}_i \in \mathcal{D}^{[6]}$. For convenience, we have noted $\mathcal{D}^{[5]} = \{\mathbf{D}_0\}$ and $\mathcal{D}^{[6]} = \mathcal{D}^{[5]} \bigcup \{\mathbf{D}_1, \ldots, \mathbf{D}_8\}$. Recalling that the total number of error patterns in $\mathcal{E}^{[6]}(\Lambda^{(11,4)})$ is $\left|\mathcal{E}^{[6]}(\Lambda^{(11,4)})\right| = 11829$, we can see that all decoders in $\mathcal{D}^{[6]}$ are in fact almost equally powerfull with respect to 6-error patterns. No decoder alone dominates the other decoders, but when the $N_{\mathcal{D}} = 9$ decoders are combined in the decoder diversity framework, they — altogether — ensure that all 6-error patterns are corrected. We also indicate the number of remaining error patterns after the sequential use of each decoder.

Another example, also presented in Table V-B, shows how very specific decoders — that we call *"surgeon"* decoders — are necessary to correct error events that could not be corrected with usual iterative decoders. We took the example of the eight 7-error patterns concentrated in the smallest trapping set, *i.e.* in the set $\mathcal{E}^{[7]}(\Lambda_{7,3}) \cup \mathcal{E}^{[7]}(\Lambda_{8,2})$. For these eight error patterns, we had to rely on eight different FAID (labeled D_{10} to D_{17} for convenience) to seperately correct the eight patterns. Moreover, in comparison with the statistics obtained from the decoders belonging to $\mathcal{D}^{[6]}$ on the 6-error patterns, these decoders are not as strong as the first nine decoders $\mathbf{D}_0$ to $\mathbf{D}_8$. Five of them especially have very poor behaviors on the 6-error events.

These two examples show clearly that in order to guarantee t error correction, the decoder diversity sets

pave the sets of considered error patterns in very different manners. In short, for $t = 6$ error correction, the decoder diversity set behaves roughly like in Figure Fig.2(a), while for $t = 7$ error correction, the decoder diversity set behave more like in Figure Fig.2(b), using both powerful and surgeon decoders. In order to be able to reproduce these results, we give in appendix B-A the decoders $\mathbf{D}_0$ to $\mathbf{D}_8$ and $\mathbf{D}_{10}$ to $\mathbf{D}_{17}$ that we have identified.

Table V

STATISTICS ON THE ERROR CORRECTION OF SEVERAL FAID DECODER IN THE DIVERSITY SETS.

Decoder D_i	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
$\left\|\mathcal{E}^{[6]}(\Lambda^{(11,4)})\right\|$	11724	11779	11777	11782	11784	11770	11759	11623	11724
remaining errors	105	16	10	7	4	3	2	1	0

Decoder D_i	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}
$\left\|\mathcal{E}^{[7]}_{\mathcal{D}_i}(\Lambda_{7,3}) \cup \mathcal{E}^{[7]}_{\mathcal{D}_i}(\Lambda_{8,2})\right\|$	1	1	1	1	1	1	1	1
$\left\|\mathcal{E}^{[6]}_{\mathcal{D}_i}(\Lambda^{(11,4)})\right\|$	10781	110	83	208	10143	3726	164	321

Figure Fig.3 shows the FER performance of the decoder diversity set $\mathcal{D}^{[7]}$, when simulated on the Tanner code over the BSC channel with cross-over error probability α and with a maximum of $N_I = 120$ decoding iterations for each decoder. One can see that — especially in the error floor region — increasing the number of decoders increases the slope of the FER curve, and reaches eventually a slope of $t = 8$, which corresponds to the minimal error-event which is not corrected by our approach of decoder diversity.

VI. CONCLUSION

[TO DO] BLA BLA BLA ... Comments on the results and application to other codes than the Tanner Code.

APPENDIX A
PLANE PARTITIONS AND FAID RULES ENUMERATION

In this section, we establish the link between FAID rules and symmetric plane partitions. We then use combinatorial results on plane partition enumeration to count the number of possible FAID algorithms. We consider only the case of $d_v = 3$.

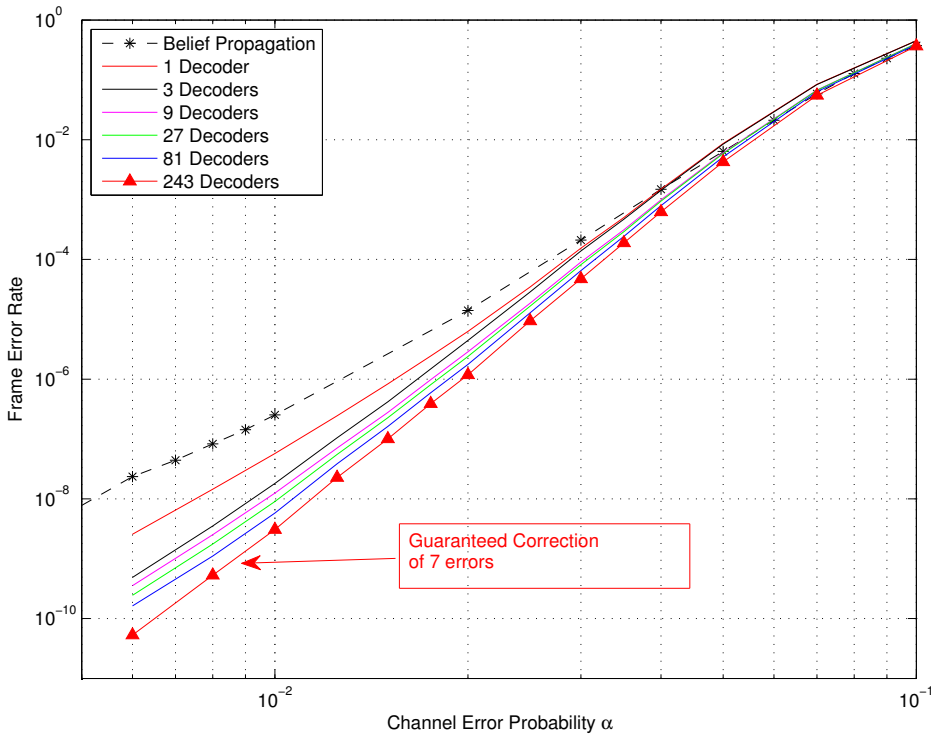
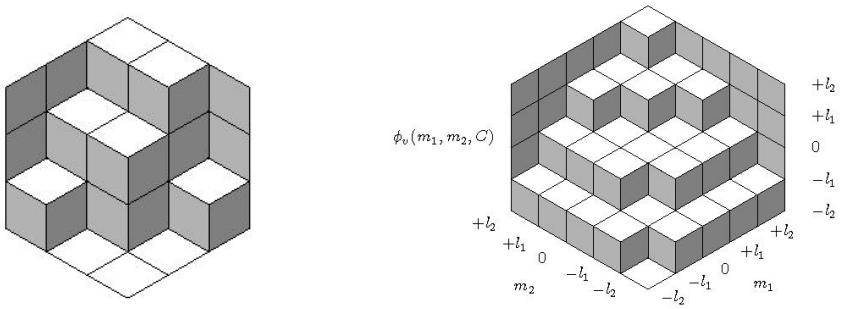


Figure 3. FER results on the Tanner Code with Guaranteed error correction of 7 errors.

A plane partition π is an array of nonnegative integers $(\pi_{i,j})_{i\geq 1, j\geq 1}$ such that $\pi_{i,j} \geq \pi_{i+1,j}$ and $\pi_{i,j} \geq \pi_{i,j+1}$ for all $i, j \geq 1$. If $\pi_{i,j} = 0$ for all $j > s$, a plane partition is *s-columned*. If $\pi_{i,j} = 0$ for all $i > r$, it is *r-rowed*. If $\pi_{i,j} \leq t$, we say that the parts *do not exceed* t. An r-rowed, s-columned plane partition whose parts do not exceed t is said to be *contained* in a box with sidelengths (r, s, t).

Consider the plane partition given by: $\pi_{3,3} = 0$, $\pi_{3,2} = 0$, $\pi_{3,2} = 1$, $\pi_{2,3} = 0$, $\pi_{2,2} = 2$, $\pi_{2,1} = 2$, $\pi_{1,3} = 1$, $\pi_{1,2} = 3$, $\pi_{1,1} = 3$. The value $\pi_{i,j}$ is represented as a box of height $\pi_{i,j}$ positioned at (i, j) coordinate on a horizontal plane. This plane partition is contained in the $(3\times 3\times 3)$ box as shown in figure Fig.4(a). Figure Fig.4(b) shows the actual plane partition for a 5-levels FAID rule that we have designed. The box of height $\pi_{i,j}$ is positioned on the horizontal plane at the position (i, j). For convenience the box *walls* corresponding to the planes $i = 0$ and $j = 0$ are shown. The figure (a) represents the case of Example 1, and figure (b) represents an actual FAID rule with 5 levels.

A plane partition is symmetric if $\pi_{i,j} = \pi_{j,i}$. Since $\Phi_v(M_i, M_j, -C) = l_{N_s+1-i,N_s+1-j}$, the assignment $l_{N_s+1-i,N_s+1-j} = M_{N_s+1-(\pi_{i,j}+1)}$ establishes a bijection between a plane partitions con-



(a) (b)

Figure 4. A vizualisation of the plane partition as stacked boxes.

tained in a $(N_s \times N_s \times N_s - 1)$ box and an array $l_{i,j}$ defining the update function Φ_v. Clearly, since $\pi_{i,j}$ is non-increasing with both i and j, it follows that $\Phi_v(L_{i+1}, L_j, -C) \geq \Phi_v(L_i, L_j, -C)$ and $\Phi_v(L_i, L_{j+1}, -C) \geq \Phi_v(L_i, L_j, -C)$ for all $1 \leq i, j \leq N_s$. The non-increasing property of plane partitions matches the condition of lexicographic ordering defined by Equation 9.

Due to symmetry of the update functions, the corresponding plane partition is symmetric. In [34], Kuperberg gave an elegant formula for the enumeration of symmetric plane partitions contained in a box of dimensions (r, r, t). The number of symmetric plane partitions is given by:

$$N_2(r, r, t) = \frac{H_2(2r + t + 1)\, H_1(r)\, H_2(t)}{H_2(2r + 1)\, H_1(r + t)} \tag{19}$$

where $H_k(n) = (n - k)!\,(n - 2k)!\,(n - 3k)!\ldots$ is called the staggered hyperfactorial function.

Therefore, the total number of plane partitions contained in the $(N_s \times N_s \times N_s - 1)$ box corresponds to the total number of valid FAID rules for Φ_v, defined on N_s-levels. From Equation (19) this number is given by:

$$N_2(N_s, N_s, N_s - 1) = \frac{H_2(3N_s)H_1(N_s)H_2(N_s - 1)}{H_2(2N_s + 1)H_1(2N_s - 1)} \tag{20}$$

APPENDIX B
TOPOLOGIES OF THE TANNER CODE

As explained in [11], there are three types of homomorphisms which preserve the topological structures in the graph of the Tanner code, due to the fact that Tanner's design of the parity-check matrix is based on an array of (d_v, d_c) circulants of size L, and that the values of shifts for the circulant matrices are chosen from two multiplicative sub-groups of the Galois field GF(L). For easy understanding, we shall

instead present the homorphisms as simple transformations acting on the indices of the variable nodes in the code. Let α (respectively β) be two elements of GF(L) with multiplicative order d_c (respectively d_v). The parity check matrix is defined by an array of circulants with shift orders $\{\alpha^t\beta^r\}_{0\leq t\leq d_c-1, 0\leq r\leq d_v-1}$. Now, let the index of a variable node v_i be expressed as $i = k*L+l$. We now define the three following transformations acting on the indices of $\mathbf{T}$ that preserve the topology as well as the neighborhood of the TS.

- block-cyclicity: Let $\sigma:\ V\times\{0,\ldots,L-1\}\rightarrow V$. Then

$$\sigma(v_i,t) = v_j \quad \text{where } j = (k*L)+(l+t \mod (L))$$

- row-wise transformation: Let $\pi:\ V\times\{0,\ldots,d_c-1\}\rightarrow V$. Then

$$\pi(v_i,t) = v_j \quad \text{where } j = ((k+t \mod (d_c))*L)+(\alpha^t l \mod (L))$$

- column-wise homomorphism: Let $\rho:\ V\times\{0,\ldots,d_v-1\}\rightarrow V$. Then

$$\rho(v_i,t) = v_j \quad \text{where } j = (k*L)+(\beta^t l \mod (L))$$

Consider a trapping set of size a bits denoted by $\mathbf{T} = \{v_{n_1},\ldots,v_{n_a}\}$. By applying the transformation σ on $\mathbf{T}$ such that $\sigma(\mathbf{T},t) = \{\sigma(v_{n_1},t),\ldots,\sigma(v_{n_a},t)\}$ where $t\in\{0,\ldots,L-1\}$, the induced subgraphs of $\sigma(\mathbf{T},t)$ and $\mathbf{T}$ are *isomorphic* to each other in the code $\forall t\in\{0,\ldots,L-1\}$, i.e., they have exactly the same topology and neighborhood. This implies that one has to only consider error patterns associated with one of the isormorphic structures instead of all of them. The same applies for the transformations π and ρ. By applying all three transformations $\sigma(\pi(\rho(\mathbf{T})))$, the number of trapping sets of a certain type $\mathcal{T}$ that need to be considered for the analysis of an iterative decoder is significantly reduced.

A. *List of 7-level FAID used in the paper*

In the Table VI, we list some of the FAID variable node rules with 7-levels that have been used in this paper. We have only indicated the values of the indices in the alphabet $\mathcal{M}$, and for the array entries which cannot be deduced by symmetry (see section III for notations).

Remind the reader again what $l_{1,1}, l_{1,2},\ldots,l_{1,7}$ are. Give the full table for D_0 as an example.

REFERENCES

[1] M. Tanner, "A recursive approach to low complexity codes," *Information Theory, IEEE Transactions on*, vol. 27, no. 5, pp. 533–547, 1981.

Table VI

LIST OF SOME 7-LEVEL FAID USED IN THIS PAPER. THE FIRST 9 FAID GUARANTEE AN ERROR CORRECTION OF 6 BITS FOR THE TANNER CODE.

FAID	$l_{1,1}$	$l_{1,2}$	$l_{1,3}$	$l_{1,4}$	$l_{1,5}$	$l_{1,6}$	$l_{1,7}$	$l_{2,2}$	$l_{2,3}$	$l_{2,4}$	$l_{2,5}$	$l_{2,6}$	$l_{2,7}$	$l_{3,3}$	$l_{3,4}$	$l_{3,5}$	$l_{3,6}$	$l_{3,7}$	$l_{4,4}$	$l_{4,5}$	$l_{4,6}$	$l_{4,7}$	$l_{5,5}$	$l_{5,6}$	$l_{5,7}$	$l_{6,6}$	$l_{6,7}$	$l_{7,7}$
$\mathbf{D}_0$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	L_1	$-L_2$	$-L_2$	$-L_1$	$-L_1$	L_1	$-L_1$	0	0	L_1	0	L_1	L_2	L_1	L_3	L_3
$\mathbf{D}_1$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	0	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	L_1	$-L_2$	$-L_1$	$-L_1$	0	L_2	$-L_1$	0	0	L_2	0	L_1	L_2	L_1	L_3	L_3
$\mathbf{D}_2$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_2$	L_1	$-L_2$	$-L_1$	$-L_1$	0	L_1	$-L_1$	0	0	L_3	0	L_1	L_3	L_1	L_3	L_3
$\mathbf{D}_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_1$	L_2	$-L_2$	$-L_1$	$-L_1$	0	L_2	$-L_1$	0	0	L_2	0	L_1	L_3	L_1	L_3	L_3
$\mathbf{D}_4$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_1$	L_1	$-L_2$	$-L_2$	$-L_1$	$-L_1$	L_2	$-L_1$	0	0	L_2	0	L_1	L_2	L_1	L_2	L_3
$\mathbf{D}_5$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	0	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_1$	L_1	$-L_2$	$-L_2$	$-L_1$	$-L_1$	L_2	$-L_1$	0	0	L_2	0	L_1	L_2	L_1	L_2	L_3
$\mathbf{D}_6$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	L_1	$-L_2$	$-L_2$	$-L_1$	L_1	L_2	$-L_1$	0	L_1	L_2	0	L_1	L_2	L_1	L_2	L_3
$\mathbf{D}_7$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	L_1	$-L_2$	$-L_2$	$-L_1$	$-L_1$	L_1	$-L_1$	$-L_1$	0	L_3	0	L_1	L_3	L_2	L_3	L_3
$\mathbf{D}_8$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	0	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	L_1	$-L_2$	$-L_1$	$-L_1$	0	L_2	$-L_1$	0	0	L_2	L_1	L_1	L_2	L_3	L_3	L_3
$\mathbf{D}_{10}$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	0	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	L_2	$-L_3$	$-L_2$	$-L_1$	$-L_1$	L_2	$-L_1$	0	L_1	L_2	L_1	L_1	L_3	L_1	L_3	L_3
$\mathbf{D}_{11}$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	$-L_3$	$-L_3$	$-L_1$	$-L_1$	$-L_1$	L_1	$-L_3$	$-L_1$	0	0	L_2	$-L_1$	L_1	L_2	L_3	L_2	L_3	L_3	L_3	L_3	L_3
$\mathbf{D}_{12}$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	0	$-L_3$	$-L_3$	$-L_3$	$-L_2$	0	L_2	$-L_3$	$-L_3$	0	L_1	L_2	$-L_1$	L_1	L_2	L_3	L_1	L_2	L_3	L_2	L_3	L_3
$\mathbf{D}_{13}$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	0	L_1	$-L_3$	$-L_2$	$-L_2$	0	L_2	$-L_2$	0	L_2	L_2	L_2	L_2	L_3	L_3	L_3	L_3
$\mathbf{D}_{14}$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	0	L_1	$-L_2$	$-L_2$	$-L_1$	L_1	L_2	$-L_2$	$-L_1$	L_1	L_2	0	L_2	L_3	L_3	L_3	L_3
$\mathbf{D}_{15}$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_2$	$-L_1$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	0	L_2	$-L_3$	$-L_2$	$-L_2$	L_1	L_2	$-L_2$	$-L_1$	L_1	L_3	0	L_2	L_3	L_3	L_3	L_3
$\mathbf{D}_{16}$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	L_1	$-L_3$	$-L_3$	0	L_1	L_1	$-L_1$	L_1	L_1	L_2	L_1	L_1	L_2	L_2	L_2	L_3
$\mathbf{D}_{17}$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_1$	0	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_1$	L_2	$-L_3$	$-L_2$	$-L_1$	L_1	L_2	$-L_2$	L_1	L_1	L_3	L_1	L_2	L_3	L_2	L_3	L_3

[2] D. J. MacKay and M. S. Postol, "Weaknesses of Margulis and Ramanujan-Margulis Low-Density Parity-Check Codes," in *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003, p. 2003.

[3] C. A. Kelley and D. Sridhara, "Pseudocodewords of Tanner Graphs," *IEEE Transactions on Information Theory,*, vol. 53, no. 11, pp. 4013–4038, Nov. 2007.

[4] T. Richardson, "Error Floors of LDPC Codes," *Proc. 41st Annual Allerton Conf on Communications Control and Comuting*, 2003.

[5] B. Smith, F. R. Kschischang and W. Yu, "Low-density parity-check codes for discretized min-sum decoding," in *Proc. 23rd Biennial Symp. on Commun.*, pp. 14–17, 2006.

[6] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[7] S.K. Planjery, D. Declercq, S.K. Chilappagari, and B. Vasic, "Multilevel decoders surpassing belief propagation on the binary symmetric channel," 2010, Preprint. [Online]. Available: http://arxiv.org/abs/1001.3421

[8] S.K. Planjery, D. Declercq, L. Danjean, and B. Vasic, "Finite Alphabet Iterative Decoders,", Electronics Letters, vol. 47, no. 16, pp. 919–921, Aug. 2011.

[9] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. IEEE Int. Conf. on Commun. (ICC '06)*, vol. 3, Istanbul, Turkey, pp. 1089–1094, 2006.

[10] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. Wainwright, "Lowering LDPC error floors by postprocessing," in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM '08)*, New Orleans, LA, pp. 1–6, Nov.30-Dec. 4 2008.

[11] M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," 2001. [Online]. Available: citeseer.ist.

psu.edu/tanner01class.html

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

[13] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[14] D. Declercq and M. Fossorier, "Improved Impulse Method to Evaluate the Low Weight Profile of Sparse Binary Linear Codes", in the proc. of ISIT'08, Toronto, Canada, July 2008.

[15] S. K. Chilappagari and B. Vasic, "Error correction capability of column-weight-three LDPC codes," *IEEE Trans. Inform. Theory*, vol. 55, no. 5, pp. 2055–2061, May 2009.

[16] D. Declercq, L. Danjean, S. K. Planjery and B. Vasić, "Finite alphabet iterative decoding (FAID) of the (155,64,20) Tanner code," in *Proc. 6th International Symposium on Turbo-Codes & Iterative Information Processing*, Brest, France, 2010.

[17] D. Nguyen, B. Vasić, M. Marcellin and S.K. Chilappagari, "Structured LDPC codes from permutation matrices free of small trapping sets", *Proc. of the IEEE Inf. Theory Workshop*, Dublin, Ireland, Sept. 2010, .

[18] B. Vasic, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," *Proc. 47th Annual Allerton Conf. on Commun., Control, and Computing*, Sept. 2009.

[19] S. K. Chilappagari, A. R. Krishnan, and B. Vasic, "LDPC codes which can correct three errors under iterative decoding," in *Proc. IEEE Inform. on Theory Workshop*, pp. 406–410, May 2008.

[20] L. Sassatelli, S. K. Chilappagari, B. Vasic, and D. Declercq, "Two-bit message passing decoders for LDPC codes over the binary symmetric channel," in *Proc. IEEE Int. Symp. on Inform. Theory (ISIT 2009)*, Seoul, Korea, pp. 2156–2160, July 2009.

[21] L. Danjean, S. K. Planjery, D. Declercq and B. Vasić, "On the Selection of Finite Alphabet Iterative Decoders for LDPC codes on the BSC" in *Proc. Inform. Theory Workshop (ITW 2011)*, Praty, Brazil, October 2011.

[22] S. Abu-Surra, D. Declerq, D. Divsalar, and W. Ryan, "Trapping Set Enumerators for Specific LDPC Codes", in *Proc. of the ITA workshop*, San Diego, CA, USA, February 2010.

[23] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: M.I.T. Press, 1963.

[24] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[25] T. Richardson and R. Urbanke, "Modern Coding Theory," *Cambridge University Press*, 2008.

[26] L. Dolecek, Z. Zhang, V. Anatharam, M. Wainwright and B. Nikolic, "Analysis of Absorbing Sets and Fully Absorbing Sets of Array-Based LDPC Codes," *IEEE Transactions on Information Theory*, vol. 56, no 1, pp. 181–201, 2010.

[27] R. Koetter and P. Vontobel, "Graph-covers and Iterative Decoding of Finte Length Codes", *in the Proc. of Turbo-coding symposium*, Brest, France, 2003.

[28] F. Mac-Williams and N. Sloane, "The Theory of Error-Correcting Codes", *North-Holland Mathematical Library*, 1978.

[29] Richard M. Karp (1972), *Reducibility Among Combinatorial Problems*, R. E. Miller and J. W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum. pp. 85103.

[30] S. K. Chilappagari, D. V. Nguyen, B. Vasic, M. W. Marcellin,"On trapping sets and guaranteed error correction capability of LDPC codes and GLDPC codes", *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1600–1611, Apr. 2010.

[31] S. K. Chilappagari, D. V. Nguyen, B. Vasic, M. W. Marcellin, "Error correction capability of column-weight-three LDPC codes under the Gallager A algorithm - PART II", IEEE Trans. Inf. Theory, vol. 56, no. 6, pp. 2626–2639, Jun. 2010.

[32] L. Kocarev, F. Lehman, G.M. Maggio, B. Scanavino, Z. Tasev and A. Vardy, "Nonlinear Dynamics of Iterative Decoding Systems: Analysis and Applications", *IEEE Trans. on Info. Theo.*, vol. 52, pp. 1366-1384, April 2006.

[33] X. Zheng, F. Lau, C. Tse, S.C. Wong, "Study of Binfurcation Behavior of LDPC Decoders", *International Journal of Bifurcation and Chaos*, Vol. 16, No. 11 (2006) 34353449.

[34] G. Kuperberg, "Symmetries of plane partitions and the permanent-determinant method," *J. Comb. Theory, Ser. A*, vol. 68, no. 1, pp. 115-151, 1994.