Abstract—We introduce a generic approach for improving the guaranteed error correction capability of regular low-density parity check codes. The method relies on operating (in serial or in parallel) a set of finite alphabet iterative decoders. 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 for the Binary Symmetric Channel, if some error pattern cannot be corrected by one particular decoder, there exists in the set of decoders, another decoder which can correct this pattern. We show how to select a plurality of message update rules so that the set of decoders can collectively correct error patterns on the dominant trapping sets. We also show that a set of decoders with dynamic re-initializations can approach the performance of maximum likelihood decoding for finite-length regular column-weight three codes.

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 length. However, for finite length sparse codes, iterative decoding fails on specific subgraphs of a Tanner graph, generically termed as trapping sets [1], [2]. Trapping sets (TS) give rise to an error floor which is an abrupt degradation in the error-rate performance of the code in the high signal to noise ratio regime. This performance degradation is also characterized by the notion of pseudo-codewords [4], 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 has been observed that the support of pseudo-codewords on the Binary Symmetric Channel (BSC) are typically contained in topological structures which are trapping sets. Many authors have pointed out that the minimum Hamming weight of pseudo-codewords is generally smaller that the minimum distance for most LDPC codes [5]. Thus, the presence of trapping sets prevents, in principle, iterative decoders to approach or reach the performance of MLD for finite length LDPC codes.

An error correcting code $C$ is said to have $t$-guaranteed error correction capability using a decoder $D$ over the BSC channel if $D$ 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 such as 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 [6], 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 the identification of the corresponding trapping sets of an ID in the code, it still remains an arduous task to determine whether a particular ID succeeds in correcting all $t$-error patterns localized on all the trapping sets. As an example, for strictly regular LDPC codes whose Tanner graphs have girth $g = 8$, it has been shown that the guaranteed error correction capability differs when different ID are used [7], [8].

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 ID, 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) [9], [10], for which the messages may be represented only by three bits. The reason for this is that FAID have been demonstrated to be more powerful than traditional ID in the error floor region [9], [11], and also the FAID framework allows us to define a plurality of different iterative decoders [3] with manageable complexity that are 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 in parallel, 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 small trapping sets present in the code. 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 in parallel, 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 small trapping sets present in the code. The methodology then involves searching for such trapping sets, 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 by using increasingly larger sets of decoders, one can significantly increase the guaranteed error correction of iterative decoding for short length LDPC codes. As an example, we apply this procedure to the \((N = 155, K = 62, d_{\text{min}} = 20)\) Tanner code. We show in particular that using the framework of FAID decoding diversity on the Tanner code can increase the guaranteed error correction capability from \( t = 5 \) for existing iterative decoders to \( t = 7 \), thereby leading to a significant improvement in the slope of the error floor in the frame error-rate (FER) curve.

In a second part of the paper, we extend the approach of decoder diversity and combine it with random dynamical re-initializations of each FAID. By using random dynamical re-initializations, we advantageously make use of the inherent oscillating behavior of iterative decoders around attractors defined by the trapping sets to avoid converging to trapping sets attractors. By combining decoder diversity and dynamically re-initialized decoders, we were able to approach very close to the performance of MLD on several finite lengths regular LDPC codes on the BSC channel. For this paper, we provide numerical results only for the \((N = 155, K = 62, d_{\text{min}} = 20)\) Tanner code.

II. NOTATIONS AND FAID DECODERS

A. LDPC Codes and Channel Assumption

The LDPC codes are defined by their Tanner graphs comprised of two sets of nodes: the set of variable nodes \( V = \{v_1, \ldots, v_N\} \) and the set of check nodes \( C = \{c_1, \ldots, c_M\} \). 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 \( v_i \) is denoted by \( \mathcal{N}(c_j) \). We will consider only regular LDPC codes, so that 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 \( x = (x_1, x_2, \ldots, x_N) \) be a vector such that \( x_i \) denotes the value of the bit associated with variable node \( v_i \). The binary codeword \( x \) is transmitted over a binary symmetric channel (BSC) with channel error probability \( \alpha \) and is received as \( y \). The BSC channel can also be modeled as \( y = x \oplus e \), where the binary vector \( e = (e_1, e_2, \ldots, e_N) \) 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 \( e = (e_1, e_2, \ldots, e_N) \), denoted by \( \text{supp}(e) \), is defined as the set of all positions \( i \) such that \( e_i \neq 0 \).

B. Finite Alphabet Iterative Message Passing Decoders

Iterative decoders operate by passing messages, denoted along the edges of the Tanner graph representation of the code, with the goal of estimating the a posteriori distribution of the codeword bits, and taking a decision on the most reliable codeword. In each iteration, the messages on the edges are computed as a function of only local messages coming from their neighbors, using two local update rules \( \Phi_s \) and \( \Phi_e \), which determine the message output through the variable nodes and the check-nodes, respectively. The methodology presented in this paper relies on the recently introduced type of finite alphabet iterative decoders (FAID) [9], [10], for which the messages are 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, and for which the update functions \( \Phi_s \) and \( \Phi_e \) are defined as follows:

\[
\Phi_s(m_1, \ldots, m_{d_c-1}) = \left( \prod_{j=1}^{d_c-1} \text{sgn}(m_j) \right)^{\min_{j\in\{1,\ldots,d_c-1\}} |m_j|} (1)
\]

\[
\Phi_e(y_i, m_1, m_2, \ldots, m_{d_c-1}) = Q \left( \sum_{j=1}^{d_c-1} m_j + m_c \cdot y_i \right) (2)
\]

where \( Q(.) \) is a non-uniform quantization function and the additive parameter \( m_c \) is computed using a non-linear symmetric function of the incoming messages. Refer to [10] for more details on the definition of a FAID. Note that the function \( \Phi_e \) is the same as the one used in the min-sum decoder [12]. 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. Since the update functions treat zeros and ones in the same way, we can assume in our analysis that the all-zero codeword was transmitted. 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 \( \hat{x}^{(k)} = (\hat{x}_1^{(k)}, \hat{x}_2^{(k)}, \ldots, \hat{x}_N^{(k)}) \) is calculated as \( \hat{x}_i^{(k)} = 0 \) if \( y_i + \sum m_{N(v_i)\rightarrow v_i}^{(k)} > 0 \), \( \hat{x}_i^{(k)} = 1 \) if \( y_i + \sum m_{N(v_i)\rightarrow v_i}^{(k)} < 0 \), and \( \hat{x}_i^{(k)} = (\text{sgn}(y_i) - 1)/2 \) otherwise, where the \text{sgn} function denotes the standard signum function.

These new message passing decoders have much lower complexity compared to existing message passing decoders, while having the potential to surpass classical decoders such as the floating-point BP or the min-sum decoder in the error floor.
region. Additionally, the FAID framework allows to define a plurality of powerful iterative decoders having different decoding dynamics for particular error pattern realizations, which we make use of in this paper to introduce the concept of decoder diversity. For column-weight $d_v = 3$ codes, the function $\Phi_v$ can be conveniently represented as a two dimensional array, or look-up table (LUT). We have shown in [10] that the number of non-trivial $N_v$-level FAID for $d_v = 3$ codes (denoted Class-A FAID) is given by

$$K_A(N_v) = \frac{H_2(3N_v)H_2(N_v)H_2(N_v - 1)}{H_2(2N_v + 1)H_2(2N_v - 1)}$$

(3)

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

This leaves us with a number of possible FAID equal to $K_A(5) = 28314$ and $K_A(7) = 530803988$ for FAID that use 3 bits of precision for messages. Such an enormous number of decoders requires an efficient systematic procedure to separate good decoders from bad ones. By “good” we mean decoders which perform well in 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 cannot rely neither on density-evolution thresholds alone [11], [3] nor on the sole analysis of 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 $N_v = 5$ levels among the $K_A(5) = 28314$ decoders, and 70 FAID decoders with $N_v = 7$ levels among a subset of $6575972$ 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.

C. Trapping sets and trapping set ontology

Now that we have introduced the generalities about iterative decoders, we shall now discuss in detail the main weaknesses of iterative decoders in terms of error correction using the notion of trapping sets (TS) [1]. For a given decoder input $y = \{y_1, y_2, \ldots, y_N\}$, a TS $T(y)$ is a non-empty set of variable nodes that are not eventually corrected by the ID. A standard notation commonly used to denote a trapping set is $(a, b)$, where $a = |T(y)|$, and $b$ is the number of odd-degree check nodes present in the subgraph induced by $T(y)$. The Tanner graph representation of an $(a, b)$ TS will be denoted by $T$. A code $C$ is said to contain a TS of type $T$ if there exists a set of variable nodes in $G$ whose induced subgraph is isomorphic to $T$. Let $N_T$ denote the number of trapping sets of type $T$ that are contained in the code $C$. Also for convenience we shall simply use $T$ (instead of the more precise notation $T(y)$) to refer to a particular subset of variable nodes in a given code that form a trapping set. Finally, let $\{T_{i,T} \mid i = 1, \ldots, N_T\}$ be the collection of trapping sets of type $T$ present in code $C$.

A TS is said to be elementary if $T$ 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 [1], [17]. 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 [2]. We will extend this notation, which 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 $(a, b; \Pi_{k \geq 2}^{g_k}(2k)^{a_k})$ if the corresponding subgraph contains exactly $g_k$ distinct cycles of length $2k$.  

As an example, for $d_v = 3$ and girth $g = 8$ codes, there are two possible non-isomorphic TS with $(a, b) = (6, 4)$, one with two 8-cycles and one 12-cycle $(6,4;8^{2}12^{1})$ and one with one 8-cycle and two 10-cycles $(6,4;8^{1}10^{2})$. Our choice of notation appears to be sufficient for differentiating between the topological structures of multiple $(a, b)$ trapping sets.

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 $T$, the decoder is stuck and cannot converge to a codeword (right or wrong codeword). Moreover, following the interpretation of iterative decoders as dynamical systems [20], [21], trapping sets act as attractor points for an ID, and error patterns which are close but not identical to $T$ could also be attracted to $T$ [1]. Although the trapping sets are usually the support of patterns for which the ID fails to converge in the low error-floor region, the number of bit errors and the particular locations of the errors that cause an iterative decoder to fail depend on the ID update equations [11]. 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 [9], [7], or exploit this dependence to combine several FAID to improve the performance, as done in this paper.

III. DECODER DIVERSITY

A. Decoder Diversity Principle

In this section, we formally 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 II-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 $N_x$-level FAID denoted by

$$D = \left\{ \left( \mathcal{M}, \mathcal{Y}, \Phi_v^{(i)}, \Phi_e \right) \mid i = 1, \ldots, N_D \right\}$$

(4)

where each $\Phi_v^{(i)}$ is defined as in Equation (2), and $\Phi_e$ is the check node rule defined in Equation (1). We refer to this set $D$ as a decoder diversity set, and an element of this set is denoted by $D_i$ where $D_i = \left( \mathcal{M}, \mathcal{Y}, \Phi_v^{(i)}, \Phi_e \right)$.

The question we address is to determine whether the decoders in the set $D$ could be used in combination (either sequentially or in parallel) in order to guarantee the correction of all error patterns up to a certain weight $t$. We first introduce notations to denote the set of error patterns correctable by each decoder. Let $E$ denote an arbitrary set of error patterns on a code $C$ whose Tanner graph is $G$, i.e. a set of non-zero vectors $e$ which serve as initialization of the decoders. Now, let $E_{D_i} \subseteq E$ be the subset of error patterns that are correctable by a given FAID $D_i$, alone.

**Definition 2.** We say that the set of error patterns $E$ is correctable by a decoder diversity set $D$ if

$$E = \bigcup_{i=1}^{N_D} E_{D_i}$$

Given a set of error patterns up to a certain weight $t$ on a code, one would like to determine the smallest decoder diversity set that can correct all such error patterns. This problem is known as the Set Covering Problem, and is NP-hard [19]. 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 diversity set, based on some heuristics, which results in $D$ larger than the minimal set, but still greatly improves 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 $E_{D_i}$. Typically, strong decoders have large correctable subsets $E_{D_i}$, 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 $D$ from $D_i$’s in order to cover the set $E$ with the sets $E_{D_i}$. Two distinguished ways are illustrated in Figure 1. Figure 1(a) shows a case where the set of error events $E$ (represented as a big square) is paved with nearly equally powerful decoders (smaller overlapping squares of similar sizes). Figure 1(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 $C$ whose Tanner graph is $G$, we would like to find a small decoder diversity set $D$ which guarantees correction of a fixed number of errors $t$. Since we are required to determine the correctable subsets for each Class-A $N_x$-level FAID that is considered for possible inclusion into the set $D$, 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 $G'$:

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

(5)

Note that $\mathcal{E}^k(G)$ denotes the set of all error patterns of weight $k$ on the code $C$. For simplicity, we shall denote this particular set as $\mathcal{E}^k$ instead of $\mathcal{E}^k(G)$. Also let $\mathcal{E}[i] = \bigcup_{k=1}^{w(G)} \mathcal{E}^k$ 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 $D$ 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 $C$. 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 $\{ T_{i,T} \mid i = 1, \ldots, N_T \}$ denotes the collection of all $(a,b)$ trapping sets of type $T$ that are present in code $C$. Let $\mathcal{E}^k(T)$ denote the set of error patterns of weight $k$ whose support lies in a $(a,b)$ trapping set $T_{i,T}$ of type $T$. More precisely,

$$\mathcal{E}^k(T) = \{ e : w(e) = k, \ \text{supp}(e) \subseteq T_{i,T} \ i \in \{ 1, \ldots, N_T \} \}$$

(6)

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

$$|\mathcal{E}^k(T)| = \binom{a}{k} N_T$$

(7)

Figure 1. Typical ways in which decoder diversity can correct all error patterns from a pre-determined set $E$. 

<table>
<thead>
<tr>
<th>$E$</th>
<th>$D$</th>
<th>$E$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${ (a,b) } \subseteq V' \subseteq V$</td>
<td>$\bigcup_{i=1}^{N_D} E_{D_i}$</td>
<td>${ (a,b) } \subseteq V' \subseteq V$</td>
<td>$\bigcup_{i=1}^{N_D} E_{D_i}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$E$</th>
<th>$D$</th>
<th>$E$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${ (a,b) } \subseteq V' \subseteq V$</td>
<td>$\bigcup_{i=1}^{N_D} E_{D_i}$</td>
<td>${ (a,b) } \subseteq V' \subseteq V$</td>
<td>$\bigcup_{i=1}^{N_D} E_{D_i}$</td>
</tr>
</tbody>
</table>

[Figure 1(a) and 1(b) depict two types of decoder diversity sets: (a) a set of Class-A $N_x$-level FAID where each decoder is a small subset of error patterns correctable by $D_i$, and (b) a set of Class-A $N_x$-level FAID where each decoder is a large subset of error patterns correctable by $D_i$.]

[Equation (1) displays the check node rule defined in Equation (1). We refer to this set $D$ as a decoder diversity set, and an element of this set is denoted by $D_i$ where $D_i = \left( \mathcal{M}, \mathcal{Y}, \Phi_v^{(i)}, \Phi_e \right)$.]

[Equation (2) 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.]
Now, let $\Lambda_{a,b}$ denote the set of all TS of different types present in the code $C$ that have the same parameters $(a,b)$. The error sets $E^k(\Lambda_{a,b})$ and $E^{[t]}(\Lambda_{a,b})$ associated with $\Lambda_{a,b}$ are defined as follows:

$$E^k(\Lambda_{a,b}) = \bigcup_{T \in \Lambda_{a,b}} E^k(T)$$  \hspace{1cm} (8)$$

$$E^{[t]}(\Lambda_{a,b}) = \bigcup_{k=1}^{t} E^k(\Lambda_{a,b})$$  \hspace{1cm} (9)$$

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

$$E^{[t]}(\Lambda^{(A,B)}) = \bigcup_{0 \leq a \leq A, 0 \leq b \leq B} E^{[t]}(\Lambda_{a,b})$$  \hspace{1cm} (10)$$

Clearly, $E^{[t]}(\Lambda^{(A,B)}) \subseteq E^{[t]}$, 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 built from protograph expansions, and are therefore quasi-cyclic in nature. As a result, Tanner graph such of codes possess many TS that are not only isomorphic in the sense of their topological structure, but also have identical neighborhoods. Therefore, it suffices 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-cyclicality. An example of this construction is the Tanner code we mentioned in the Introduction [14]. For this particular design, the existence of three homomorphism groups reduces by several orders of magnitude the number of TS of maximum size $(A,B)$ that need to be considered. We will study in detail a code of this type in section IV.

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 necessarily the case for LDPC codes with $d_v = 4$, as explained with the concept of absorbing sets in [18]. 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 $D$ that corrects all patterns in the set $E^{[t]}(\Lambda^{(A,B)})$ on the trapping sets $\Lambda^{(A,B)}$, with $A = 2t$, and sufficiently large $B$, then the decoder diversity set $D$ will also correct all error patterns up to weight $t$ on the code $C$ 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 shall be lower than $\lceil d_{\min}/2 \rceil$. Put differently, if a decoder $D_i \in D$ 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. 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 results for the $(N = 155, K = 62, d_{\min} = 20)$ Tanner code (in Section IV). Based on the above conjecture, we now see that considering the set $E^{[t]}(\Lambda^{(A,B)})$ instead of $E^{[t]}$ may be sufficient for determining the decoder diversity set that ensures a $t$-guaranteed error correction capability, while at the same time sufficiently reducing the cardinality of the error sets that need to be considered in the selection procedure.

**C. Generation of FAID diversity sets**

We now present the procedure for selecting the FAID diversity set $D^{[t]}$ capable of correcting all error patterns in the set $E^{[t]}(\Lambda^{(A,B)})$. 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 $D_{\text{base}}$ 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 [3]. Our goal is to select $D^{[t]}$ from $D_{\text{base}}$. In essence, the procedure runs over all error patterns up to weight $t$ included in $E^{[t]}(\Lambda^{(A,B)})$ and tests their correctability when decoded by different decoders from $D_{\text{base}}$.

The algorithm iteratively expands the sets $D^{[1]}, D^{[2]}, \ldots D^{[t]}$ by including the decoders capable of correcting larger and larger error patterns into the diversity set. At the same time, the algorithm is bookkeeping the set of $E^\prime$ of error patterns whose correctability is undecided. The algorithm halts when $E^\prime = \emptyset$. This means that for the given choice of $t$ and $N_1$, the decoder diversity set $D^{[t]}$ guarantees correction of $t$ errors on the error set $E^{[t]}(\Lambda^{(A,B)})$, which in turns guarantees correction of $t$ errors on the LDPC code $C$ in a maximum of $N_1$ iterations assuming the Conjecture 1 is valid. As a side result, the selection algorithm gives also the diversity sets $D^{[k]}$ for $k < t$, which are obtained at each iterative stage of the algorithm. To describe the algorithm, let $E^\prime \subset E^{[t]}(\Lambda^{(A,B)})$ denote the set of unresolved error patterns during the selection algorithm. If $E^{[t]}_{D_i}$ is the set of error patterns that are correctable by the decoders $D_1, D_2, \ldots, D_t$, which are included in $D^{[t]}$. 

then the set of unresolved error patterns is
\[ \mathcal{E}^r = \mathcal{E}[i](\Lambda(A,B)) \setminus \bigcup_{1 \leq i \leq t} \mathcal{E}_{D_i}^{[i]} \]

The iterative selection algorithm is now described as follows:

**Algorithm 1 Decoder Diversity Selection Algorithm**

1) Given \( D_{base} \) and \( N_I \), set \( D^{[k]} = \emptyset \), \( 1 \leq k \leq t \). Initialize \( k \), set parameters \((A, B) = (2k, B)\) based on conjecture 1. Set \( \mathcal{E}^r = \mathcal{E}^{k}(\Lambda(A,B)) \).
2) Set \( D^{k} = \emptyset \) and \( i = 1 \).
   a) run all FAID in \( D_{base} \) 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. \(|\mathcal{E}^{D_i}_{D_i}\)| is the maximum.
   b) remove all error patterns corrected by \( D_i \) from the set \( \mathcal{E}^r \), i.e. \( \mathcal{E}^r = \mathcal{E}^r \setminus \mathcal{E}^{D_i}_{D_i} \).
   c) Set \( i = i + 1 \), go to 2) a) and repeat until \( \mathcal{E}^r = \emptyset \).
3) Set \( D^{[k]} = D^{[k]} \cup D^{k} \).
4) Set \( k = k + 1 \), and \( (A, B) = (2k, B) \). Set \( D^{[k]} = D^{[k-1]} \).
   a) \( \forall D_i \in D^{[k]} \), determine the correctable subsets of \( k \)-error patterns by decoders \( D_i \) denoted by \( \mathcal{E}_{D_i}^{[k]}(\Lambda(A,B)) \).
   b) set \( \mathcal{E}^r = \mathcal{E}^{k}(\Lambda(A,B)) \setminus \bigcup_{D_i \in D^{[k]}} \mathcal{E}_{D_i}^{[k]}(\Lambda(A,B)), \)
   c) go to 2) and repeat until \( k = t \).

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 correct a trapping set varies across error patterns \( \mathcal{E}^{[i]}(\Lambda(A,B)) \), and different choices of \( N_I \) lead to different identified decoder diversity sets. Determining optimal number of iterations is beyond the scope of this paper, but it will be discussed in the next section for our test case.

If, during the iterations of the selection algorithm, \( \mathcal{E}^r \neq \emptyset \) even after \( i = D_{base} \), then it implies that even the entire set of candidate FAID \( D_{base} \) 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 \( D_{base} \) or consider a larger set of candidate decoders \( D_{base} \) 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.

**IV. 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 [13], [14], 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 \( d_c = 3 \), row weight \( d_r = 5 \), 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 patterns 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>
<thead>
<tr>
<th>( t )</th>
<th>Algorithm</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Gallager A and Gallager B</td>
<td>[7]</td>
</tr>
<tr>
<td>4</td>
<td>Min-Sum and Belief Propagation</td>
<td>[9]</td>
</tr>
<tr>
<td>5</td>
<td>5-level and 7-level</td>
<td>[11]</td>
</tr>
</tbody>
</table>

We also found by simulations [3] 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 \( \mathcal{E}^{[i]}(\Lambda(A,B)) \) 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 \( D^{[i]} \).

For the case of the Tanner code, the structure of its Tanner graph satisfies certain structural properties in addition to the block-cyclicity property which comes from the design based on circulants and 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 [13]. Following notations of [13], the transformations \( \sigma, \pi \), and \( \rho \) act on the indices of the variable nodes and preserve the topological structures. The transformation \( \sigma \) 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 \( \pi \) and \( \rho \) 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 number of TS that need to be checked by the Algorithm 1 when the code automorphisms are exploited. \( N_T \) corresponds to the number of trapping sets of type \( T \) that are present in the code, without taking into account the code structure. \( N_{\sigma(T)} \) corresponds to number of trapping sets after the cyclic transformation \( \sigma \) is applied, and \( N_{\pi(\sigma(T))} \) corresponds to number of trapping sets after all three transformations \( \sigma, \pi \) and \( \rho \) are applied.
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 [16], [15]. 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 a given type \( T \) that can be present in the code would have to be a multiple of either \( L_d \), \( d_v = 465 \), \( L_d = 155 \) or \( d_v = 93 \) and this number is further reduced for the analysis by the transformations \( \sigma(\pi(\rho(T))) \).

<table>
<thead>
<tr>
<th>( T )</th>
<th>TS label</th>
<th>( N_T )</th>
<th>( N_{\sigma(T)} )</th>
<th>( N_{\sigma(\pi(T))} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,3)</td>
<td>(5,3,4)</td>
<td>155</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>(6,4)</td>
<td>(6,4,4)</td>
<td>930</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>(7,3)</td>
<td>(7,3,4)</td>
<td>930</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>(8,2)</td>
<td>(8,2,4)(12,2)(13,4)(6,1)</td>
<td>465</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>(8,4)</td>
<td>7412</td>
<td>165</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>(9,3)</td>
<td>1660</td>
<td>60</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>(9,4)</td>
<td>(9,4,4)(10,2)(12,4)(14,10)(16,8)</td>
<td>15</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(10,2)</td>
<td>1309</td>
<td>45</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>(10,4)</td>
<td>(10,4,4)(12,4)(14,3)(16,20)(18,20)</td>
<td>15</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(10,5)</td>
<td>(10,5,4)(12,4)(14,10)(16,20)(18,20)</td>
<td>30</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(11,3)</td>
<td>13150</td>
<td>485</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>(12,2)</td>
<td>20926</td>
<td>20926</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>(12,4)</td>
<td>196440</td>
<td>6240</td>
<td>416</td>
<td></td>
</tr>
<tr>
<td>(13,3)</td>
<td>34038</td>
<td>11519</td>
<td>79</td>
<td></td>
</tr>
</tbody>
</table>

### Table II

**Trapping Set Spectrum of the \((N = 155, K = 62, d_{min} = 20)\) Tanner Code**

The error sets that we have considered for the Tanner code are shown in Table III. In this table the first value represents the cardinality of each error set, which has been reduced by the structural properties \( \sigma \), \( \pi \), and \( \rho \) of the Tanner code to:

\[
|E^{c}(T)| = \binom{a}{k} N_{\sigma(\pi(\rho(T)))}. \tag{11}
\]

where \( N_{\sigma(\pi(\rho(T)))} \) 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 as one listed in the 5-error patterns in the TS (8, 2). In order to limit the computational complexity of the decoder selection algorithm, we only include in the error sets \( E^{c}(k)(A_{a,b}) \), the error patterns which are distinct from all patterns in \( E^{c}(k)(A_{a',b'}) \) with \( a' < a \) and \( b' < b \). This reduced number is indicated as the second value for each error set in Table III. 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 \( E^{c}(k)(A_{a,b}) \) instead of \( E^{c}(k) \) 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 sufficient to enable reasonable computational time for finding the decoder diversity set.

### B. Error Correction Results for the Tanner Code

Let us recall that we consider only 7-level FAID which requires 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 the Tanner code requires using \( N_D = 343 \) FAID with a run for no more than \( N_I = 120 \) iterations.

**Table III**

**Cardinalities of Error Sets Considered for the \((N = 155, K = 62, d_{min} = 20)\) Tanner Code.**

<table>
<thead>
<tr>
<th>7-errors</th>
<th>6-errors</th>
<th>5-errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>E^{c}(0)(A_{1,3})</td>
<td>)</td>
</tr>
<tr>
<td>(</td>
<td>E^{c}(1)(A_{1,3})</td>
<td>)</td>
</tr>
<tr>
<td>(</td>
<td>E^{c}(2)(A_{1,3})</td>
<td>)</td>
</tr>
</tbody>
</table>

**Table IV**

**A 1-tolerant Error Correction Diversity Schemes on the \((N = 155, K = 62, d_{min} = 20)\) Tanner Code**

\[
\begin{array}{c|cc|c|c|c|c|c}
\hline
\text{Dec. Selection Algorithm} & \text{No. of FAID} & \text{No. of Iterations} & \text{Min. V. Node} & \text{Max. V. Node} & \text{Min. C. Node} & \text{Max. C. Node} \\
\hline
\text{ODFAID} & 343 & 120 & 20 & 20 & 15 & 25 \\
\text{ODFAID} & 343 & 120 & 20 & 20 & 15 & 25 \\
\text{ODFAID} & 343 & 120 & 20 & 20 & 15 & 25 \\
\text{ODFAID} & 343 & 120 & 20 & 20 & 15 & 25 \\
\text{ODFAID} & 343 & 120 & 20 & 20 & 15 & 25 \\
\end{array}
\]

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 utmost \( t \) (for \( t = 5, 6, 7 \)) on the Tanner code even though only error patterns in \( E^{c}(k)(A_{1,3}(B)) \) were used in the algorithm, thus providing further evidence to the validity of the conjecture 1 stated in Section III-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 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 \( \Phi_c \) changes after \( N_I \) iterations (and the decoder is restarted).

Now let us have a look at how the decoder diversity set behaves on different error sets. We have reported in Table IV-B the statistics of some FAID by computing the number of correctable error patterns associated with each decoder. The first part of the Table shows the values of \( |E^{c}(6)(A_{11,4})| \), \( \forall D_{i} \in D^{[6]} \). For convenience, we have noted \( D^{[5]} = \{D_0 \} \) and \( D^{[6]} = D^{[5]} \bigcup \{D_1, \ldots, D_5 \} \). Recalling that the total number
of error patterns in $\mathcal{E}^6(\Lambda^{(11,4)})$ is $|\mathcal{E}^6(\Lambda^{(11,4)})| = 11829$, we can see that all decoders in $\mathcal{D}^6$ are in fact almost equally powerful with respect to 6-error patterns. No decoder alone dominates the other decoders, but when the $N_D = 9$ 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 IV-B, shows how very specific decoders, which 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 $\mathcal{D}_{10}$ to $\mathcal{D}_{17}$ for convenience) to separately 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 $\mathcal{D}_0$ to $\mathcal{D}_8$. 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 Fig. 1(a), while for $t = 7$ error correction, the decoder diversity set behave more like in Fig. 1(b), using both powerful and surgeon decoders. The details on the decoding rules which were used to obtain those results will be reported in a future publication.

Table V

<table>
<thead>
<tr>
<th>Decoder $D_i$</th>
<th>Remaining 0</th>
<th>Remaining 1</th>
<th>Remaining 2</th>
<th>Remaining 3</th>
<th>Remaining 4</th>
<th>Remaining 5</th>
<th>Remaining 6</th>
<th>Remaining 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_0$</td>
<td>243</td>
<td>81</td>
<td>27</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D_1$</td>
<td>243</td>
<td>81</td>
<td>27</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D_2$</td>
<td>243</td>
<td>81</td>
<td>27</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D_3$</td>
<td>243</td>
<td>81</td>
<td>27</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D_4$</td>
<td>243</td>
<td>81</td>
<td>27</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D_5$</td>
<td>243</td>
<td>81</td>
<td>27</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D_6$</td>
<td>243</td>
<td>81</td>
<td>27</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D_7$</td>
<td>243</td>
<td>81</td>
<td>27</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 2 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 $\alpha$ and with a maximum of $N_I = 120$ decoding iterations for each decoder. One can see that increasing the number of decoders increases the slope of the FER curve especially in the error floor region, 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.

V. APPROACHING MAXIMUM LIKELIHOOD DECODING USING DECODER DIVERSITY

In the first part of the paper, we have shown that the concept of FAID decoder diversity can significantly improve the error correction performance of a LDPC code. The approach presented however, has its limits since the number of TS that needs to be considered for larger and larger $t$ become prohibitively large. Also, FAID algorithms that use only 3 bits of precision might not be sufficient to correct error patterns with Hamming weights close to the MLG error correction capability. For example, we were not able to find a 7-level FAID which would correct the 8-error pattern on the (8,2)TS present in the Tanner code even with $N_I = 500$ iterations. In order to further improve and try to approach MLD performance with FAID, we present in this second part of the paper a new approach, which combines the use of decoder diversity with dynamical re-initialization of each decoder.

A. STATE-SPACE REPRESENTATION

As pointed out in the literature [20], [21], an LDPC iterative decoder can be seen as a dynamical system with possible chaotic behavior for which the dynamical equations are given by (2)-(1). In the case of a FAID algorithm which uses messages belonging to a finite alphabet $\mathcal{M}$, the dynamical system is not chaotic since its state space dimension is finite. However, it still can have pseudo-chaotic behaviors in the sense that the convergence of FAID is either to stable fixed points or infinite oscillations around attractor points in the state space. The state space for an ID of an LDPC code can be defined over the state variables referred to as state vectors, where each state vector denoted by $\{m^{(k)}\}$ is a collection of messages on all edges of the Tanner graph of the code after $k$ iterations. This could be the vector $(m_{c_j \rightarrow v_i})_{c_j \in C, v_i \in \mathcal{N}(c_j)}$, or a vector of messages in the other direction $(m_{v_i \rightarrow c_j})_{v_i \in V, c_j \in \mathcal{N}(v_i)}$, or both. The series of state vectors for all considered iterations $k$ is called a trajectory. A chaotic system is a deterministic system in which trajectories remain bounded in the state space and show strong sensitivity to small changes in initial conditions, which is the case of an ID of LDPC codes. For example, on the BSC, flipping a particular bit in the error pattern may totally change the behavior of the decoder. Chaotic dynamical systems either diverge or oscillate, or converge to a fixed point in the state space. A fixed point $\{m^{(k)}\}^*$ is reached when the values taken by the state vector remain stable with the iterations,
or equivalently when:
\[
\left\{ m^{(k')} \right\} = \left\{ m^{(k)} \right\} \quad \forall k' \geq k
\]

In order to accurately represent the dynamical behaviors of IDs, we need to find a scalar parameter that characterizes all the messages \( \left\{ m^{(k)} \right\} \). In existing literature, either the number of bit errors at each iteration or the entropy of the decision vector \( \mathbf{z}^{(k)} \), is used as the single parameter to characterize the dynamical behaviors. However, either of these two parameters do not make use of all the messages transiting in the graph, and therefore can only partially represent the behavior of the dynamical system. In this paper, we propose to use the graph free-entropy also denoted in statistical physics as Bethe free-entropy [22], as the single parameter which is a function of all the messages in the Tanner graph.

To compute the Bethe free-entropy for a FAID algorithm, we first transform the messages belonging to the finite alphabet to probabilities by writing them as log-likelihoods with convention \( \log(p(b_i = 0)/p(b_i = 1)) \) where \( b_i \) is the bit being estimated corresponding to variable node \( v_i \). Let \( \mu_{c_j \rightarrow v_i}(0) \) denote the probability of the bit \( b_i \) being zero which is evaluated by its neighboring check node \( c_j \). Let \( \mu_{c_j \rightarrow v_i}(1) \) be denoted similarly. Let the message on this edge be denoted as \( m_{c_j \rightarrow v_i} \). Then the probabilities are evaluated as follows
\[
\mu_{c_j \rightarrow v_i}(0) = \frac{e^{m_{c_j \rightarrow v_i}}}{1 + e^{m_{c_j \rightarrow v_i}}}, \quad \mu_{c_j \rightarrow v_i}(1) = 1 - \mu_{c_j \rightarrow v_i}(0)
\]

We first determine the free-entropy \( E_i \) corresponding to each variable node \( v_i \) as \( E_i = \log(\sum_1^d \prod_{j=1}^M \mu_{c_j \rightarrow v_i}(u)) \). We then determine the free-entropy \( E_j \) corresponding to each check node \( c_j \) as \( E_j = \log(\sum_{i_1+i_2+\cdots+i_{d_c}=0}^{d_c} \prod_{n=1}^I \mu_{v_i \rightarrow c_j}(i_n)) \). Finally, for each edge, we calculate the free-entropy \( E_{ij} \) as \( E_{ij} = \log(\sum_{u=0}^{|E|} \mu_{c_j \rightarrow v_i}(u)\mu_{v_i \rightarrow c_j}(u)) \). At the \( k \)-th iteration, the Bethe free-entropy is defined as the sum of these three entropies as
\[
E^k = \sum_{i=1}^N E_i + \sum_{j=1}^M E_j + \sum_{i=1}^N \sum_{j=1}^M E_{ij}
\]

where \(|E| = d_v N = d_c M\) is the number of edges in the Tanner graph \( G \). The Bethe free-entropy, which is negative definite, is maximal when the decoder converges to a codeword.

**B. FAID Decoder Diversity with Randomized Dynamics**

In this section, we introduce a method based on random re-initialization of a decoder diversity set that enabled us to approach the MLD performance. More details on the analysis of the typical behaviors of FAID dynamical algorithms with Jacobian matrices and computation of Lyapunov exponents will be provided in our future publications. The basic principle for our approach is based on the following observation. If for some error pattern all FAID fail to converge to the right codeword, they are either stuck on a fixed point in the vicinity of some large TS or oscillate around an attractor. In order to break the infinite oscillations, we propose to stop the decoding process regularly after a certain number of iterations and re-initialize the decoder with some function of the state vector. This is similar to a method known in chaotic systems [23] that introduces random jumps in system phase-space when the trajectory reaches some Poincaré plane cut. By introducing appropriately chosen random jumps, one can hope to break the oscillations due to the chaotic attractor and eventually converge to a stable fixed point which is expected to be the right codeword.

We will denote the jumps as random re-initializations. The first random parameter corresponds to the periodicity at which we perform the jumps. Indeed, oscillations of ID around attractors have most of the time a typical periodicity, and one needs to perform the random re-initializations irregularly in time (here time is the iteration index) in order to break the oscillatory behavior. Based on Monte-Carlo experiment, we could estimate the typical periodicity \( N_I^* \) of a given code and select at random the iteration index at which we perform the random re-initialization, following a uniform distribution around \( N_I^*: U[N_I^* - \varepsilon, N_I^* + \varepsilon] \) (for some integer \( \varepsilon > 0 \)). The second random aspect of the proposed jumps is related to the way one wants to avoid the attractor by forcing the decoder to be pushed away from the associated TS. The approach we propose is to compare the hard-decision at iteration \( N_I^* \) with the original channel values, and randomly flip the bits which do not agree, with some probability \( \delta \), as explained in Algorithm 2. Figure 3 shows the trajectories of the graph entropy for the Tanner code and a FAID algorithm, with and without the technique of random re-initialization. The re-initialization jumps in the phase-space are shown in bold on the right Figure 3(b). The FAID alone is trapped around an attractor and oscillates indefinitely (we have tested until \( N_I = 10^5 \) iterations), while the technique we propose performs random jumps in the phase-space, which eventually leads the decoder to converge to the right codeword (last bullet on the right) and correct the error event.

![](image)

**Figure 3.** Dynamics of the Entropy function (12) for a 8-error pattern on the \( (N = 155, K = 62, d_{min} = 20) \) Tanner code.

The FAID diversity set \( D_{base} \) has been obtained as described in the first part of this paper, using the FAID selection technique of [3]. In order to try and reach the best possible performance in the error floor region, we did not limit the number of decoders in \( D_{base} \) as much as in the first part of the
paper, and we selected $N_D = 9239$ FAID in the diversity set. Using the algorithm 2, with a choice of $(N_f = 20, \varepsilon = 5)$ and $\delta = 0.05$, we were able to reach the performance of MLD on the $(N = 155, K = 62, d_{\text{min}} = 20)$ Tanner Code, as shown on Figure 4. One can see in particular that the gap between the BP and the MLD on this code is particularly large and reaches almost 5 decades at channel error probability $\alpha = 10^{-2}$.

Here a maximum of $N_f = 4000$ iterations was used for each decoder in $D_{\text{base}}$. Note that although the number of FAID in $D_{\text{base}}$ is very large, the decoders do not operate in parallel but in serial. As a consequence, a large percentage of the error patterns converge with the first FAID in a small number of iterations, and the rest of the diversity set and the random re-initializations are used only for the most problematic error events. Note that we have applied this approach to various regular $d_c = 3$ LDPC codes, with different rates and lengths, and observed similar results as for the Tanner code.

**Figure 4.** Diversity Results on the rate $R = 0.4$ $(N = 155, K = 62, d_{\text{min}} = 20)$ Tanner Code

**REFERENCES**


[16] 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.


