# Finite Alphabet Iterative Decoding (FAID) of the (155,64,20) Tanner Code

David Declercq, Ludovic Danjean, Shiva K. Planjery, Erbao Li, Bane Vasic

# Finite Alphabet Iterative Decoding (FAID) of the (155,64,20) Tanner Code

David Declercq, Ludovic Danjean, Erbao Li
ETIS
ENSEA / UCP / CNRS UMR 8051
95000 Cergy-Pontoise, France
{declercq,danjean,erbao.li}@ensea.fr

Shiva K. Planjery, Bane Vasić
Dept. of Electrical and Computer Eng.
University of Arizona
Tucson, AZ 85721, USA
{shivap,vasic}@ece.arizona.edu

***Abstract*—It is now well established that iterative decoding approaches the performance of Maximum Likelihood Decoding of sparse graph codes, asymptotically in the block length. For a finite length sparse code, iterative decoding fails on specific subgraphs generically termed as trapping sets. Trapping sets give rise to error floor, an abrupt degradation of the code error performance in the high signal to noise ratio regime. In this paper, we will study a recently introduced class of quantized iterative decoders, for which the messages are defined on a finite alphabet and which successfully decode errors on subgraphs that are uncorrectable by conventional decoders such as the min-sum or the belief propagation. We will especially study the performance of the proposed finite alphabet iterative decoders on the famous (155,64,20) Tanner code.**

## I. Introduction

Low-density parity-check (LDPC) codes have received much attention in the past several years owing to their exceptional performance under iterative decoding. A wide spectrum of iterative decoders of varying complexity have been developed ranging from simple hard-decision algorithms such as Gallager-A/B algorithms to the more sophisticated belief propagation (BP) algorithm. On codes defined with sparse parity-check matrices or their equivalent Tanner graphs [1], the BP decoder performs close to the Maximum Likelihood Decoding (MLD) defined as the optimal decoder under the assumption of a cycle-free Tanner graph. This last assumption stands only for an infinite blocklength code.

For finite length LDPC codes, the presence of unavoidable cycles breaks the BP optimality, and the decoder could converge to fixed points or loopy attractors which are not codewords, leading to the so called error floor region [2, 3]. It has been observed by many authors that for finite length LDPC codes and especially for small lengths, other decoders than the BP decoder tend to have better performance in the error floor region. This is the case for example of the corrected min-sum decoder [4, 5]. More recently, Planjery *et al.* have introduced a new class of message passing decoders, called *Finite Alphabet Iterative Decoding* (FAID) decoders [6], with the goal of surpassing the BP decoder in the error floor region.

Richardson has introduced the notion of *trapping sets* in [3] to characterize error floors. A $(a, b)$ trapping set (TS) is a subgraph of the whole Tanner graph, which represents a small set of $a$ bit nodes whose induced subgraph has $b$ odd degree check nodes. A TS is then a small topological structure which prevents the correction of the $a$ bits when initally in error by an iterative decoder, which will be defined as a decoding failure. The error floor corresponds to the decoding failures induced by low-weight error patterns located on TS. Trapping sets can be present in any finite-length code irrespective of how good the decoding threshold obtained by density evolution is and hence, codes optimized for good decoding thresholds can still exhibit high error floors. Characterization of error floors and design of LDPC codes with low error floors has recently been a subject of wide interest [7, 8]. Note that in principle, a TS is defined by a particular topological structure, and does not depend on the chosen decoder. As a consequence, a well chosen iterative decoder could in fact *not be trapped* by a TS, and this is one of the key feature of our appraoch to design good FAID decoders.

In this paper, we present a detailed study of FAID decoders of the (155,64,20) Tanner code [9], in the case the decoders use messages stored only on 3 quantization bits. The (155,64,20) Tanner code is a particularly nice LDPC code and a 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 error correction difference between standard iterative decoders (Gallager-B, uncorrected min-sum, BP) and MLD is expected to be large, which also means that improving iterative decoders should be easier on this code than on other, larger codes. Another reason is that the (155,64,20) Tanner code is sufficiently small and structured (the code is quasi-cyclic with bloc-cyclicity equal to 31) such that brute force simulation can be used to verify some claims.

The paper is organized as follows. In section II, we describe in details the concept of FAID decoders, and give 4 decoders using messages only on 3 quantization bits that we will study in details on the Tanner code. In section III, we depict the main sub-structures that we focus on, which are the smallest TS of the Tanner code, and the smallest codewords of weight $d_{min} = 20$. We show in particular that there are 3 different topologies for the minimal codewords on the Tanner code, and that two of them contain one and only one minimal TS. In section IV, we explain that looking at the minimal TS in an isolated way is not sufficient to predict the behavior of the iterative decoder, and propose a simulation based strategy to discriminate between good and bad FAID decoding rules. We present the results of the obtained FAID decoders for the (155,64,20) Tanner code on the binary symmetric channel (BSC).

## II. Finite alphabet Iterative Decoders (FAID)

In the general case of a binary LDPC code $\mathcal{C}$ of length $N$ and $M$ constraints, the related Tanner graph contains $N$ bit nodes and $M$ parity-check nodes. We consider only regular LDPC whose bit nodes have degree $d_v$, and check nodes have degree $d_c$. The FAID decoding algorithm is an iterative decoder presented in [6] in which the messages propagated along the edges of the Tanner graph belong to a finite alphabet $\mathcal{M}$. In the case of $N_s$ levels $\mathcal{M} = \{0, \pm l_k : 1 \leq k \leq \lfloor \frac{N_s}{2} \rfloor\}$ where the sign of $l_k$ represents the value of the bit to zero or one, and the magnitude $|l_k|$ represents the reliability of the bit value. The value of the observation from the channel $\{y_i\}_{i=0,1,\ldots,N-1}$ belongs to the channel output set which is simply $\mathcal{Y} = \{-C, +C\}$ in the case of the BSC. As for all message-passing algorithm, update rules are defined on both bit nodes and check nodes of the Tanner graph. The update rules consist

in computing the outgoing messages from the different extrinsic messages entering a node (except the message on the edge for which the output is computed); let $\Phi_v$ and $\Phi_c$ be the functions representing respectively the rules for the bit nodes and for the check nodes.

In this paper, we consider the check node update function $\Phi_c$ to be

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

where sgn(.) denotes the standard signum function. This particular update function $\Phi_c$ corresponds to the same update function in the min-sum decoder. For all decoders considered in this paper, $\Phi_c$ will be unchanged, such that the variability in the definition of decoders will come only from different choices of bit node rules $\Phi_v^{(k)}$.

For a regular column-weight $d_v$ code the update rule $\Phi_v$ on the $i^{th}$ bit node will depend on the incoming messages $m_1, m_2, ..., m_{d_v-1}$ and the channel value $y_i$. The function $\Phi_v$ is in general a non-linear function of the $m_i$'s, and can be expressed as:

$$\Phi_v(m_1, m_2, \cdots, m_{d_v-1}, y_i) = Q\left( \sum_{j=1}^{d_v-1} m_j + \omega_c \cdot y_i \right) \quad (2)$$

where $\omega_c = \Omega(m_1, m_2, ..., m_{d_v-1})$ is symmetric non-linear function $\Omega : \mathcal{M}^{d_v-1} \rightarrow \{0, 1\}$, and the function $Q(.)$ is a — potentially non-uniform — quantization function, defined by a set of thresholds (see [6] for more details).

The function $\Phi_v$ follows the symmetry condition:

$$\Phi_v(m_1, ..., m_{d_v-1}, -C) = -\Phi_v(-m_1, ..., -m_{d_v-1}, C) \quad (3)$$

If the condition (3) is fulfilled, then the decoder is symmetric, and treats zeros and ones in the codeword in the same way.

For general codes, the function $\Phi_v$ lives in a discrete space of dimension $2.|\mathcal{M}|^{d_v-1}$, *i.e.* the number of possible values for its entries. In this paper, since we consider only the Tanner code which is regular with $d_v = 3$, the function $\Phi_v$ can be conveniently represented by a single 2-D look-up table with the values $\{\Phi_v(m_1, m_2, -C)\}_{m_1,m_2}$ tabulated in it. The values for the output function $\Phi_v(m_1, m_2, +C)$ can be deduced from the 2-D LUT with equation (3). We give in the next section four examples of those functions that will be studied in details on the Tanner code.

### A. Decoding rules using 3 quantization bits

A FAID decoder will be said to use $k$ quantization bits when the messages realization set $\mathcal{M}$ has cardinality $N_s = |\mathcal{M}| < 2^k$. We will use in this paper only decoding rules such that the number of levels $N_s$ is odd, and such that the value $m = 0$ belongs to $\mathcal{M}$. The value $m = 0$ corresponds to an erasure message, while the signum of the message indicates the value of the corresponding bit. By convention, we choose negative values to represent a bit equal to 0 and positive values to represent a bit equal to 1.

We designed 4 different rules, which are reported on tables V to VIII. The first 3 rules on tables V to VII are defined over an alphabet of $N_s = 5$ levels, while the fourth rule on table VIII is defined with $N_s = 7$ levels. The function outputs corresponding to $\Phi_v(m_1, m_2, +C)$ can be deduced by symmetry. All the 4 rules will use the check node update defined by eq. (1). At each iteration, the decision on each coded bit $c_i$ is made with the following rule:

$$\begin{cases} \sum_{j=1}^{d_v} m_j(i) + y_i < 0 & \Rightarrow \quad c_i = 0 \\ \sum_{j=1}^{d_v} m_j(i) + y_i > 0 & \Rightarrow \quad c_i = 1 \\ \sum_{j=1}^{d_v} m_j(i) + y_i = 0 & \Rightarrow \quad c_i = y_i \end{cases}$$

| (155,64,20) Tanner code | | |
|---|---|---|
| TS(5,3) 0-3-0-0-0-0 | → | **155** |
| TS(6,4) 0-1-2-0-0-0 | → | **930** |
| TS(7,3) 0-3-2-0-2-0 | → | **930** |
| TS(7,5) 0-1-1-0-1-0 | → | **11160** |
| TS(7,5) 0-1-0-2-0-0 | → | **2790** |

Table I: Trapping Set spectrum of the Tanner Code.

For each rule, we indicated the decoding threshold of the decoder for the regular $(d_v = 3, d_c = 5)$ family, which corresponds to the connectivity of the Tanner code. The decoding threshold corresponds to the maximum value of the channel error probability $\alpha^*$ such that density evolution converges to the noiseless case [10]. We can see that all these decoders have similar decoding thresholds, and the gap to the Shannon limit $\alpha_{SL} = 0.1461$ can be explained by the very small number of bits which quantize the messages of the decoder.

## III. Topologies of the (155,64,20) Tanner code

### A. Trapping Sets Distributions

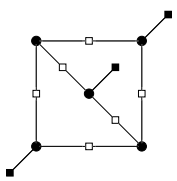
Let $\mathbf{e} = (e_1, e_2 \ldots, e_n)$ be an error pattern at the input to the decoder obtained from the BSC. A trapping set $\mathbf{T}(\mathbf{e})$ is a non-empty set of variable nodes that are eventually not corrected by the decoder [3]. A standard notation for a TS is $TS(a, b)$ where $a$ is the number of bits in errors in $\mathbf{e}$ and $b$ is the number of odd-degree check nodes in the sub-graph induced by $\mathbf{T}(\mathbf{e})$.
This standard notation is however not sufficient to describe in details the topologies which are the supports of the TS. In particular, there could be several different topologies which have the same values $a$ and $b$. To circumvent this problem, we extend the notation of TS by adding to the first two parameters, an additionnal topological information which allows to distinguish between different structures with the same $a$ and $b$. We propose the following notation:

$$TS \ : \ (a, b) \ n_{c_3} - n_{c_4} - n_{c_5} - n_{c_6} - n_{c_7} - n_{c_8}$$

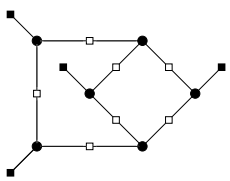
where $n_{c_k}$ represents the number of cycles containing $k$ bit nodes.

Let us focus on the topologies of the Tanner code which are supposed to be dominant in the error floor region of the frame error rate curves, namely the smallest structures in terms of number of bits involved in it. Using expansion of the neighboring tree from each bitnode, it is quite easy to derive an algorithm which detects and counts the small closed topologies, and therefore TS, in a graph. We have reported on table I the distribution of TS up to $a = 7$ bits which are present in the Tanner Code. Two examples of those TS are drawn on figure 1(a) and figure 1(b).

The TS(5,3) is the corner point of the weakness of iterative decoders on the Tanner code. This *very small* TS makes several iterative decoders fail when the bits in error are located on the 5 bits which compose the TS. More details are given in section IV.



(a) The TS(5,3) trapping set.



(b) A TS(6,4) trapping set.

| (155,64,20) Tanner code | | |
|---|---|---|
| weight 20 | → | **1023** |
| weight 22 | → | **6200** |
| weight 24 | → | **43865** |
| weight 26 | → | **259918** |

Table II: Distance spectrum of the Tanner Code.

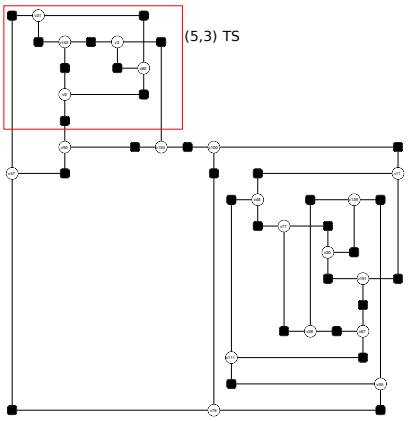



Figure 1: Topological structure of a Type-I minimal codeword of the Tanner code

### B. Minimal Codeword Structures

The knowledge of the dominant TS could be sufficient to predict the behavior of usual decoders in the error floor region [7, 3]. However, as demonstrated in the next section, when the iterative decoder is more general, which is the case of the FAID decoders studied here, looking at the TS alone is not sufficient. The natural and obvious thing to do is then to look at bigger structures which are also attractor points of the decoders. Instead of considerering larger and larger TS, we propose to study the behavior of the decoders on the closed structures which form the codewords of the Tanner code. The main reason is that the multiplicity of TS with constant $b > 1$ becomes rapidly cumbersome with increasing $a$. It is our belief that looking at error events located inside a codeword gives a lot of information about iterative decoding convergence points, although a TS is not necessarily nested in a codeword.

The Hamming distance spectrum of the Tanner code is given on table II. This spectrum has been obtained with the impulse algorithm presented in [11], and the multiplicities are assumed exact. We will focus on the minimum codewords of weight $d_{min} = 20$. Note that those minimal codewords are actually TS(20,0) trapping sets.

By analysis of the topologies of these codewords, we have identified that there are only 3 types of structures for the minimal codewords, which we will denote Type-I, Type-II and Type-III. This means that each and every codeword of weight 20 belongs to one of the automorphism group of the subgraph induced by one of the 3 types of codewords. This is especially interesting since we can restrict the study of the decoders on 3 sub-graphs instead of 1023 sub-graphs. Another observation is that only 2 out of the 3 types contain the minimal TS, *i.e.* the TS(5,3) trapping set. We have drawn on figure 1 the structure of Type-I codewords, which contain the TS(5,3) trapping set. For lack of space, we do not represent the other types.

## IV. The Issue of Predicting Decoder Behavior based on Minimal Trapping Sets

### A. Trapping Sets Critical Numbers

In [7] the concept of *critical number* has been introduced to characterize the contribution to the error floor of a given trapping set. Given a TS$(a, b)$ trapping Set $\mathbf{T}(\mathbf{e})$, the critical number $m(\mathbf{T}(\mathbf{e}))$ is the minimal number of bits received in error inside the trapping set leading to a decoding failure. Although the notion of critical number was originally developed for Gallager-A/B algorithms in [12], it is still applicable in our current framework of FAID decoders since we are considering decoding over the BSC.

The critical number is then more representative than the TS itself to measure its impact in the error floor region. For example, a TS(7,3) with critical number 4 will have a larger contribution to the error floor than a TS(5,3) with critical number 5. Additionnally, we choose the convention $m(\mathbf{T}(\mathbf{e})) = \infty$ when all combinations of $a$ errors or less are corrected by the iterative decoder.

In the next section, we show that computing the critical numbers of the TS when the TS are considered in an isolated way could be misleading.

### B. Limitations of the Isolation Assumption

In [6], the concept of *isolated structures* was defined and used to analyze local decoding behaviors in order to derive good update rules for FAID decoders. In short, the isolation assumption ensures that the decoder updates inside a TS are not corrupted by propagation of messages along external closed path to the TS, and this for a given number of iterations $k$. In other words, the TS is *isolated* from the rest of the graph for at least $k$ iterations. This assumption is necessary to interpret correctly the values of the critical numbers and to be able to predict the performance of one iterative decoder in the error floor region based on these critical numbers. Please refer to [6] for a detailed discussion about the advantages of the isolation assumption.

As a consequence, the computation tree of an isolated structure is equivalent if the structure is simulated alone, or if it is simulated when embedded in a global larger graph. Under the isolation assumption, the critical numbers therefore represent exactly the typical error correcting behavior of a decoder on the whole graph of a code. We have computed and indicated on table III the critical numbers of the 4 different FAID decoders, and for all TS present in the Tanner code up to 8 bits.

Without big surprise, the TS(8,2) is the most difficult TS to cope with for all decoders. Note that all 4 FAID decoders have infinite critical number on the 2 smallest TS, the TS(5,3) and the TS(6,4), which seems to indicate that it is possible to derive quantized decoders, even with very few quantization bits, which are not trapped by the TS of usual decoders (Gallager-B, Min-sum).

Those critical numbers are however not predictive at all when the isolation assumption is not fulfilled. We have verified some error correction properties with extensive Monte Carlo simulations on the whole Tanner code. It turns out that although the rule $\Phi_v^{(3)}$ has the exact same statistics as rule $\Phi_v^{(2)}$ and even better statistics than rule $\Phi_v^{(1)}$ in terms of critical numbers, rule $\Phi_v^{(3)}$ fails on 110 five-error patterns when we simulate the rule on the whole Tanner code, while rule $\Phi_v^{(1)}$ and rule $\Phi_v^{(2)}$ correct all five-errors patterns in less than 100 iterations. Another contradiction is that the critical number for rule $\Phi_v^{(2)}$ on TS(8,2) is 5, which means that there are five-error patterns such that decoder $\Phi_v^{(2)}$ fails in an isolated way, but successfully corrects the five errors when the TS(8,2) is simulated in the whole Tanner code. As we can see,

| Number of bits | Trapping Set Label | rule $\Phi_v^{(1)}$ | rule $\Phi_v^{(2)}$ | rule $\Phi_v^{(3)}$ | rule $\Phi_v^{(4)}$ |
|---|---|---|---|---|---|
| 5 bits | (5,3) 0-3-0-0-0-0 | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 6 bits | (6,4) 0-1-2-0-0-0 | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 7 bits | (7,3) 0-3-2-0-2-0 | 7 | $\infty$ | $\infty$ | 6 |
| | (7,5) 0-1-1-0-1-0 | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| | (7,5) 0-1-0-2-0-0 | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 8 bits | (8,2) 0-3-4-2-4-2 | 6 | 5 | 5 | 6 |
| | (8,4) 0-3-0-2-0-2 | $\infty$ | $\infty$ | $\infty$ | 6 |
| | (8,4) 0-1-3-1-1-1 | $\infty$ | $\infty$ | $\infty$ | 7 |
| | (8,4) 0-1-2-2-2-0 | $\infty$ | $\infty$ | $\infty$ | 7 |
| | (8,6) 0-1-0-1-0-1 | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| | (8,6) 0-1-0-0-2-0 | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| Decoding Threshold $\alpha^*$ | | 0.09781 | 0.09778 | 0.09777 | 0.10155 |

Table III: Critical numbers on the trapping sets of the Tanner code for the selected decoding rules. The decoding threshold of each rule is also shown.

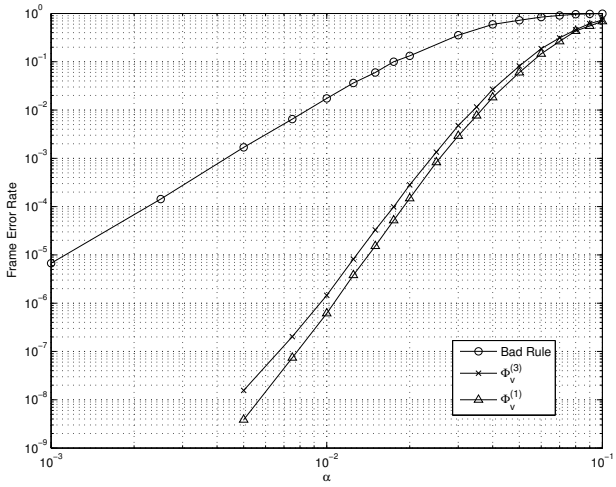



Figure 2: Performance comparison of different FAID decoders on the (155,64,20) Tanner Code

contradictions in the analysis of the isolated critical numbers are in both positive and negative directions, which makes it difficult to make use in the goal of choosing a good decoder for a particular code.

On figure 2, we compared the frame error rate (FER) performance of the FAID rules $\Phi_v^{(1)}$ and $\Phi_v^{(3)}$ on the Tanner code. All curves have been plotted with a maximum of 100 iterations, and at least 300 frame errors have been recorded for each simulation points. The difference between the two rules is not large, although the curves start to split apart in the error floor region due to the fact that $\Phi_v^{(1)}$ corrects all five-errors patterns while $\Phi_v^{(1)}$ does not. As a catastrophic counter-example of using only isolated critical numbers for designing FAID rules, we have also plotted the performance of a FAID rule which have *all its critical numbers* equal to $+\infty$ (labelled as 'Bad Rule' in the figure). The decoding threshold for this rule is only $\alpha^* = 0.07778$, and then is a lot worse than the tresholds of the rules tabulated at the end of this paper.

Of course, brute force simulations on the whole code would give the desired ordering between rules, but at the price of a too large computational burden. The problem of finding the best decoding rules for a specific code cannot be solved with the knowledge of critical numbers alone, and remains an open issue. In this paper, we propose a first approach to partially solve this issue which is still based on Monte Carlo simulations, but on larger structures than the smallest TS.

### *C. Selection of Rules by Monte-Carlo Simulations on Larger Structures*

We propose to make decisions with respect the ordering between FAID rules, by simulations of $n_e$-errors patterns on the sub-graphs induced by the minimal codewords of weight 20. Although we do not claim that simulation on these sub-graphs are strictly predictive, looking at the codeword structures makes sense with respect to the isolation assumption described in the preceding section. As a matter of fact, a codeword is a particular TS(a,0) trapping set, and then is connected to the rest of the graph only by edges which have already even degree inside the TS. More importantly, there is no edge which connects the codeword to the rest of the graph, and which ouputs from a bitnode. From our own observations on TS, the isolation assumption is more often *'broken'* when the external paths go through a bitnode than when then go only through check nodes of the TS. It seems that codewords are *almost isolated*, at least more than other types of TS. A more formal study of the isolation assumption will be reported in a future paper.

We have simulated all 5-errors patterns and all 6-errors patterns on the 3 types of codewords, for a large number of FAID decoders. We report on Table IV the results for the 5-levels decoders of tables V-VII. The numbers in the table indicate the number of error patterns which are *not* corrected by the decoders, and in the case all error events are corrected, we indicate in brackets the maximum number of iterations needed to correct all events.

| | 5-errors patterns | | | 6-errors patterns | | |
|---|---|---|---|---|---|---|
| | $\Phi_v^{(1)}$ | $\Phi_v^{(2)}$ | $\Phi_v^{(3)}$ | $\Phi_v^{(1)}$ | $\Phi_v^{(2)}$ | $\Phi_v^{(3)}$ |
| type-I | 3 | 2 | $> 10$ | 172 | 138 | $> 500$ |
| type-II | $0^{(7)}$ | $0^{(8)}$ | $0^{(9)}$ | $0^{(16)}$ | $0^{(21)}$ | $> 21$ |
| type-III | $0^{(4)}$ | $0^{(4)}$ | $0^{(4)}$ | $0^{(4)}$ | $0^{(4)}$ | $0^{(5)}$ |

Table IV: Statistics of correction for small error events on the codewords sub-graphs.

As a first observation, we can see that the 3 types of codewords have completely different behaviors. Type-I codewords seem to be the most problematic ones, and Type-III codewords the easiest to decode. Remember that Type-III codewords do not contain TS(5,3) trapping sets, which could explain why they have the best behaviors with iterative decoding. This is a very interesting differentiation of codewords which have although the same Hamming weight, and therefore cannot be differentiated with MLD.

In terms of ordering of the different rules, those statistics are in better accordance with the simulations on the whole Tanner code than the critical numbers of table III. It is readily seen on these statistics that rule $\Phi_v^{(3)}$ is worse than rules $\Phi_v^{(1)}$ and $\Phi_v^{(2)}$. Since we verified that rule $\Phi_v^{(3)}$ does not correct all five-error patterns on the Tanner code while $\Phi_v^{(1)}$ and $\Phi_v^{(2)}$ do, we can see that the ordering of rules made with simulations on the codewords is somewhat more predictive than the isolated critical numbers. A more important result is that we performed those statistics for all possible 5-levels decoders (there are 28314 possible FAID decoders), and rules $\Phi_v^{(1)}$ and $\Phi_v^{(2)}$ have the best overall statistics of all decoders. Since this approach of simulating error patterns on codewords appears to be predictive, we conjecture that we have found the best 5-levels decoders for the (155,64,20) Tanner code, that is decoders $\Phi_v^{(1)}$ and $\Phi_v^{(2)}$. The FER curves for these two decoders are very close, and we report on figure 3 only the performance of rule $\Phi_v^{(1)}$ together with the 7-levels rule $\Phi_v^{(4)}$. The 7-levels rule $\Phi_v^{(4)}$ has been obtained with similar techniques as described in details for the 5-levels rules in this paper. However,

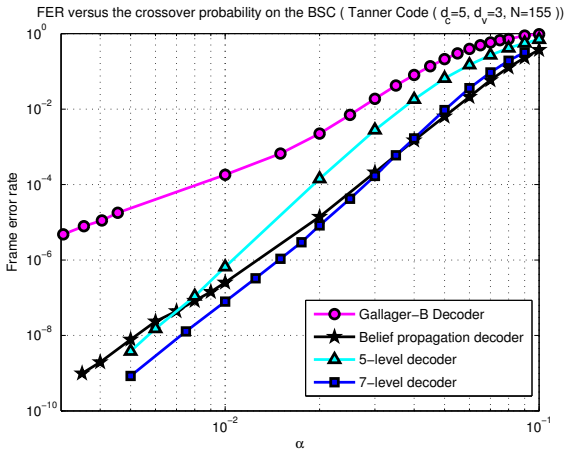



Figure 3: Performance of the best FAID rules found compared to Gallager-B and BP

| $m_1 \backslash m_2$ | $-l_2$ | $-l_1$ | $0$ | $+l_1$ | $+l_2$ |
|---|---|---|---|---|---|
| $-l_2$ | $-l_2$ | $-l_2$ | $-l_2$ | $-l_2$ | $0$ |
| $-l_1$ | $-l_2$ | $-l_2$ | $-l_1$ | $-l_1$ | $+l_1$ |
| $0$ | $-l_2$ | $-l_1$ | $-l_1$ | $0$ | $+l_1$ |
| $+l_1$ | $-l_2$ | $-l_1$ | $0$ | $+l_1$ | $+l_2$ |
| $+l_2$ | $0$ | $+l_1$ | $+l_1$ | $+l_2$ | $+l_2$ |

Table V: Table Look-up for $\Phi_v^{(1)}$ - the channel value is set to $-C$. The decoding threshold for this rule is $\alpha^{(1)^*} = 0.09781$.

we do not claim that rule $\Phi_v^{(4)}$ is the best 7-levels decoder for the Tanner code. All curves have been simulated with a maximum of 100 decoding iterations on the BSC channel with probability of error $\alpha$. As we can see, both FAID decoders $\Phi_v^{(1)}$ and $\Phi_v^{(4)}$ beat the BP decoder in the error floor region, as expected.

## V. Conclusion

The problem of looking for the best decoder for a particular code is not usual. Most research directions actually fix the iterative decoder, try to characterize its behavior and then propose to add special constraints to the code design such that the code is adapted to the decoder (both asymptotically with density evolution or for finite length cases using topological constraints). Here, we look at the problem the other way around, and by defining a large number of iterative FAID decoders, try to find the best one for a particular code. With the approach presented in this paper, we have obtained as an interesting result the best 5-levels decoder for the (155,64,20) Tanner Code. Also interesting are the error performance results of the proposed 7-levels decoder. Indeed, the curves of $\Phi_v^{(4)}$ and the BP decoder cross at FER=$10^{-3}$, and then $\Phi_v^{(4)}$ becomes better than the BP quite rapidly, although it requires messages stored only on 3 bits, which is by far smaller than the number of bits usually used in hardware implementations of the log-BP or the min-sum decoders (usually a number of 6 quantization bits is adviced). Our approach of FAID decoders could then eventually lead to reduced silicium area in hardware implementations of LDPC decoders. Those good results need to be verified for the AWGN channel and for other codes than the Tanner code.

## Acknowlegement

This work is partially funded by the NANO2012 project.

| $m_1 \backslash m_2$ | $-l_2$ | $-l_1$ | $0$ | $+l_1$ | $+l_2$ |
|---|---|---|---|---|---|
| $-l_2$ | $-l_2$ | $-l_2$ | $-l_2$ | $-l_2$ | $0$ |
| $-l_1$ | $-l_2$ | $-l_2$ | $-l_1$ | $-l_1$ | $+l_1$ |
| $0$ | $-l_2$ | $-l_1$ | $-l_1$ | $0$ | $+l_2$ |
| $+l_1$ | $-l_2$ | $-l_1$ | $0$ | $+l_1$ | $+l_2$ |
| $+l_2$ | $0$ | $+l_1$ | $+l_2$ | $+l_2$ | $+l_2$ |

Table VI: Table Look-up for $\Phi_v^{(2)}$ - the channel value is set to $-C$. The decoding threshold for this rule is $\alpha^{(2)^*} = 0.09778$.

| $m_1 \backslash m_2$ | $-l_2$ | $-l_1$ | $0$ | $+l_1$ | $+l_2$ |
|---|---|---|---|---|---|
| $-l_2$ | $-l_2$ | $-l_2$ | $-l_2$ | $-l_2$ | $0$ |
| $-l_1$ | $-l_2$ | $-l_1$ | $-l_1$ | $-l_1$ | $+l_2$ |
| $0$ | $-l_2$ | $-l_1$ | $-l_1$ | $0$ | $+l_2$ |
| $+l_1$ | $-l_2$ | $-l_1$ | $0$ | $+l_2$ | $+l_2$ |
| $+l_2$ | $0$ | $+l_2$ | $+l_2$ | $+l_2$ | $+l_2$ |

Table VII: Table Look-up for $\Phi_v^{(3)}$ - the channel value is set to $-C$. The decoding threshold for this rule is $\alpha^{(3)^*} = 0.09777$.

| $m_1 \backslash m_2$ | $-l_3$ | $-l_2$ | $-l_1$ | $0$ | $+l_1$ | $+l_2$ | $+l_3$ |
|---|---|---|---|---|---|---|---|
| $-l_3$ | $-l_3$ | $-l_3$ | $-l_3$ | $-l_3$ | $-l_3$ | $-l_3$ | $-l_1$ |
| $-l_2$ | $-l_3$ | $-l_3$ | $-l_3$ | $-l_3$ | $-l_2$ | $-l_1$ | $+l_1$ |
| $-l_1$ | $-l_3$ | $-l_3$ | $-l_2$ | $-l_2$ | $-l_1$ | $0$ | $+l_1$ |
| $0$ | $-l_3$ | $-l_3$ | $-l_2$ | $-l_1$ | $-l_1$ | $+l_1$ | $+l_2$ |
| $+l_1$ | $-l_3$ | $-l_2$ | $-l_1$ | $-l_1$ | $0$ | $+l_1$ | $+l_2$ |
| $+l_2$ | $-l_3$ | $-l_1$ | $0$ | $+l_1$ | $+l_1$ | $+l_1$ | $+l_2$ |
| $+l_3$ | $-l_1$ | $+l_1$ | $+l_1$ | $+l_2$ | $+l_2$ | $+l_2$ | $+l_3$ |

Table VIII: Table Look-up for $\Phi_v^{(4)}$ - the channel value is set to $-C$. The decoding threshold for this rule is $\alpha^{(4)^*} = 0.10155$.

## References

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

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

[4] 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.

[5] 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.

[6] 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

[7] 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.

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

[9] R. Tanner, D. Srkdhara, and T. Fuja, "A class of group-structured LDPC codes," 2001. [Online]. Available: citeseer.ist.psu.edu/tanner01class.html

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

[11] 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.

[12] 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.