Non-Binary Decoder Diversity for Dense or Locally-Dense Parity-Check Codes

David Declercq

Non-binary Decoder Diversity for Dense or Locally-Dense Parity-Check Codes

David Declercq, *Member, IEEE*

Abstract

In this paper, a new and promising framework, called "non-binary decoder diversity", is presented based on the observation that different non-binary Tanner graphs of the same binary code, decoded with a non-binary belief-propagation decoder, can have distinct convergence behaviors and fixed points. The goal of this work is to propose a decoder with linear complexity in the blocklength, and with performance close to maximum-likelihood decoding. This framework is especially interesting for binary codes which are dense or locally-dense, and for which the usual binary iterative decoders perform far from the optimum curves. By using the diversity brought by decoding distinct Tanner graphs of the same code, the proposed technique has very good decoding performance for three very different test cases which are known to be complex decoding problems: *(i)* near maximum-likelihood decoding (MLD) of BCH codes on the BPSK-AWGN channel, *(ii)* performance results which outperform bounded distance decoding of BCH codes over a binary symmetric channel (BSC), and finally *(iii)* decoding performance better than the BCJR-based turbo-decoder for parallel duo-binary turbo-codes.

Index terms- non-binary Tanner graphs, non-binary belief-propagation, iterative decoding of block codes, near MLD decoders.

I. Introduction

Iterative decoders have completely changed the use of error-correcting codes in modern digital communications. The famous paper describing an iterative decoder to decode the parallel concatenation of interleaved convolutional codes [1] has led to new and rediscovered code families (mainly Turbo-Codes and Low-Density Parity-Check codes) with capacity-approaching performance, all of them decoded with an instance of the iterative Belief propagation (BP) algorithm [2], [3].

Part of this work has been presented at the ISTC, Lausanne, 2008

David Declercq is with ENSEA/univ. Cergy-Pontoise/CNRS-UMR-8051, Cergy-Pontoise, France

Earlier introduced by Gallager [4] and used since then in other scientific fields, BP-based decoders can be interpreted as dynamical systems, with possible chaotic behavior [6]. For such iterative decoding process, it is convenient to make use of a graphical representation of the code. The representation of the code is usually referred to as Tanner graph (TG) [5], or Factor graph when the code is not defined entirely from parity-check constraints [2].

The BP-based decoders are very efficient on Turbo-Codes and LDPC codes because their factor graph representation is very sparse, in that they have a small number of connections between the nodes. Note that the nodes in an LDPC code are meant as parity-check nodes and symbol nodes while in a Turbo-code, the factor graph also contains state nodes (see *e.g.* [2] for examples). The sparseness of the graphs ensures that the optimal Bayesian update rule which is performed locally, *i.e.*, only with the knowledge of the closest neighborhood, does not suffer from correlation propagation effects due to the presence of small cycles in the graph. It is well known that iterative decoders lose their attractiveness when the graph becomes too dense, and especially, no efficient iterative decoder has been proposed yet for dense block codes such as BCH or Reed-Solomon codes.

The question of whether there exists a practical iterative decoder which can approach or reach maximum-likelihood decoding (MLD) performance for dense —or even locally dense— error-correcting codes is still an open problem. Towards this objective, one can think of two different but compliant approaches. The first one consists in taking into account the correlation of the messages in the local Bayesian computations by using more elaborate equations than the BP ones, *i.e.*, using generalized BP decoders [13]. The second approach is to modify the structure of the Tanner graph without changing the code space, in order to get a sparser representation or to adapt the Tanner graph to the noise realization [7], [8], [14]. In the current literature, the methods based on graph transformation use a binary version of BP, and the proposed decoders do not exhibit very good performance when the code becomes too long typically more than N=100 coded bits.

In this paper, we will consider the second approach, that is by making use of Tanner graph transformations, but with a non-binary BP decoder. In our work, we will use message passing decoders which operate in a finite set of high order $q \gg 2$, typically defined by binary vectors of size $p = \log_2(q)$. By doing so, we could represent a binary code with a non-binary Tanner graph, which itself represents a non-binary parity-check code defined in the finite group $\mathbb{F}_q = \mathbb{F}_2^p$. Our main motivation is that, although more complex than the binary BP, a non-binary BP decoder has still a linear complexity in the codeword length and has been shown to perform close to MLD and is in any case better than binary BP decoders for

short block codes [12]. Furthermore, there are many possible ways of modifying the non-binary Tanner graph of the same code, with very different topological properties (edge density and cycle distribution). The non-binary framework has then more degrees of freedom than the binary one.

By changing the non-binary Tanner graph representation of a given code, we have observed that the convergence behavior of the non-binary BP decoder can be different from one Tanner graph to another, when the decoders are initialized with the same noise realization. Based on this observation, we conjecture that —for a given noise realization— among the huge number of non-binary Tanner Graphs representing the same code, there exists a specific graph which is well matched to this noise realization and such that the non-binary BP decoder converges to the ML codeword. Finding the best Tanner graph might be as complex as using an ML decoder, and, as a first step, we present in this paper a related framework called *decoder diversity*. Instead of looking for an optimum Tanner-graph we build a collection of $d \gg 1$ different Tanner graphs of the same code, and capitalize on the diversity of the d decoder convergence behaviors to improve the error-correction performance.

The paper is organized as follows. In section II, we briefly recall the necessary bases of the proposed approach, that is how to define non-binary codes in $\mathbb{F}_q$ and how to deduce them from a binary parity-check code. We also recall the update equations of a non-binary BP decoder. In section III, we present the concept of non-binary decoder diversity (NB-DD), and justify the usefulness of the approach. In section IV, we present in detail three different cases for which the non-binary decoder diversity has shown to be useful. In particular, we present performance results close to MLD for the decoding of BCH codes on the BPSK-AWGN channel, performance results which outperform algebraic decoding of BCH codes over a BSC, and finally decoding results better than the BCJR based turbo-decoder for parallel duo-binary turbo-codes.

II. Background on Non-binary Parity Check Codes and Related Non-binary Decoder

A. Non-binary Parity Check Constraints formed from Binary Vectors

In this paper, we restrict the study to codes built from parity-checks defined in the same finite set, of the type $\mathbb{F}_q = \mathbb{F}_2^p$. In other words, the different symbols of the finite set $\{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$, with $\alpha_0 = 0$, have a binary vector representation of size p bits. Let $\{c_j\}_{j=0\ldots N-1}$ be the codeword non-binary symbols. For the $i-th$ parity-check equation of the code, of degree d_c, we denote by $\{k_j\}_{j=0\ldots d_c-1}$ the symbol indices which contribute to the parity-check equation. A general parity-check equation in $\mathbb{F}_q$ is

expressed as:

$$\sum_{j=0}^{d_c-1} f_{ij}\left(c_{k_j}\right) = 0 \quad \text{in } \mathbb{F}_q \tag{1}$$

where $f_{ij}(.)$ is any linear function from $\mathbb{F}_q$ to $\mathbb{F}_q$.

The equation (1) can be seen as a generalization of a parity-check defined in a Galois field, and it has been shown that fast implementation of the BP decoder is a simple generalization of the BP decoder for classical codes in Galois fields [12]. The existence of a practical decoder for general LDPC codes on $\mathbb{F}_q$, denoted in the following *NB-BP decoder*, allows to consider a wider class of parity-check classes compared to codes defined in a field, in particular:

- for parity-checks in fields $GF(q)$, the functions $f_{ij}(.)$ are bijections which perform cyclic permutations (or rotations) of the symbols α_k, and correspond to the multiplication with a non-zero element $h_{ij} \in GF(q)$. So there are only $(q-1)$ possible choices for the functions,
- for parity-checks in groups $\mathbb{F}_q$, any function from $\mathbb{F}_q$ to $\mathbb{F}_q$ (linear or non-linear, bijection or not) can be used in equation (1), which raises the number of possible functions to $q^q - 1$,
- a subset of codes of particular interest is the linear case, that is when $f_{ij}(.)$ has a binary matrix representation of size $(p \times p)$, which is the case we will consider in the rest of the paper. This case is more general than the field case since the total number of possible functions in this subcase is $q^p - 1$. We show some examples of such functions in the next section.

B. Binary Representations and Bit Clustering

We restrict the class of codes to the binary vector space case $\mathbb{F}_q = \mathbb{F}_2^p$ since the goal of this paper is to decode binary parity-check codes with a non-binary decoder. To this aim, it is necessary to consider parity-check equations following equation (1), which can be represented in a binary vector form. Let us discuss first the structure of the binary vector representation of a single parity-check equation. Let $\{b_{k_j}[i]\}_{i=1\ldots p}$ be the set of bits representing the symbol c_{k_j}, and group those bits in a vector denoted $\mathbf{b}_{k_j}$. The binary representation of a linear function $f_{ij}(.)$ from $\mathbb{F}_q$ to $\mathbb{F}_q$ is a $(p \times p)$ binary matrix that we denoted H_{ij}. Note that in the field case, the matrix H_{ij} is defined as the power of the companion matrix of the primitive element of the field [10]. With these notations, the binary representation of a parity-check equation (1) is, in vector form:

$$\sum_{j=0}^{d_c-1} H_{ij}\, \mathbf{b}_{k_j} = \mathbf{0}_p \tag{2}$$

with $\mathbf{0}_p$ being the all zero vector of size p.
The binary representation of such a parity-check equation is then a matrix of size $(p \times d_c p)$ which forms a local component code. This code can be locally treated as a binary code and decoded optimally using a BCJR decoder on its time-varying trellis, or treated as a single parity-check in a finite non-binary group, and decoded optimally with BP equations.

The operation of computing a non-binary representation of a binary code is obtained through a bit clustering operation, which is essentially the same as the one described in [12]. Let a binary linear code be defined by its parity-check matrix H_b. Every adjacent non-overlapping square matrix of size $(p \times p)$ in H_b is transformed during the clustering process into a linear function from $\mathbb{F}_q$ to $\mathbb{F}_q$, which is used in the NB-BP decoder, defined in the next section. The parameter p is called the clustering order. In general, the size of the parity-check matrix is not necessarily an integer multiple of p. In order to allow any clustering order, one can add redundant rows and all-zero columns to H_b so that the matrix can be divided in clusters of size $(p \times p)$ without changing the code space. It has been shown in [12] that the NB-BP decoder of a clustered binary code is very similar to a BP decoder on fields $GF(2^p)$, and that dealing with the more general functions $f_{ij}(.)$ is quite easy, as explained in the next section.

A cluster is defined as a sub-matrix H_{ij} of size $(p \times p)$ in H_b. For any non-zero cluster, we associate an edge in the Tanner graph which connects the corresponding group of rows and group of columns. Linked to this edge, a linear function $f_{ij}(.)$ is associated, which has H_{ij} as matrix representation. A clustering example is plotted in section III.

In order to fully understand the different types of functions that we get from the clustering process, we have depicted in Fig. 1 two clusters of size (4×4) with their associated function, one for the full rank case, and the other one for the rank deficient case. The function corresponding to example *(a)* is a bijection which implements a permutation of the symbols and the cluster H_{ij} is full rank, while the function corresponding to example *(b)* is an injective function (only half of the symbol values are reached through $f_{ij}(.)$), and the cluster H_{ij} has rank 3. When clustering a general linear block code, with the technique explained in detail in section III, the clusters do not have the regular structure of a companion matrix of a Galois field element. That is to say, a clustered binary code defines a non-binary code on $\mathbb{F}_q$, the vector space of size $p = \log_2(q)$ binary vectors, but does not define a code on the Galois field GF(q). The update equations of the message passing decoder have to take the structure of the clusters into account, and we explain in the next section how to deal with these particular functions in the NB-BP decoder.

C. Non-binary Belief Propagation Decoder on $\mathbb{F}_q$

The Tanner graph obtained by clustering a binary code with $(p \times p)$ clusters defines a code on the finite group $\mathbb{F}_q$ of the order $q = 2^p$. We will refer to the non-binary belief-propagation decoder on $\mathbb{F}_q$ as NB-BP decoder. The NB-BP decoder is very similar in nature than regular BP on finite fields. The only difference is that the non-zero values of a parity-check equation are replaced with more general linear functions from $\mathbb{F}_q$ to $\mathbb{F}_q$, defined by the binary matrices which form the clusters. In particular, is is shown in [12] that NB-BP can be implemented in the Fourier domain with a reasonable decoding complexity.

We briefly review the main steps of the NB-BP decoder and its application to the non-binary Tanner Graph. For a more detailed presentation of belief-propagation decoder over non-binary sets, see *e.g.* [9]. The non-binary Tanner graph of a parity-check code over $\mathbb{F}_q$ is depicted in Fig. 2, in which we indicated the notations we use for the vector messages, $\{U_{vp}, U_{pc}, U_{cp}, U_{pv}\}$. Additionally to the classical variable and check nodes, we add function nodes to represent the effect of the linear transformations induced from the clusters as explained in the previous section.

The NB-BP decoder has four main steps which use q-dimensional probability messages:

- *Data node update:* for each edge connected to a symbol node, the ouput extrinsic message is equal to the term by term product of all input messages and the channel likelihood message, excluding the message carried on the same branch of the Tanner graph.
- *Function node update:* the messages are updated through the function nodes $f_{ij}(.)$. This message update is reduced to a cyclic permutation in the case of a finite field code, but in the case of a more general linear function from $\mathbb{F}_q$ to $\mathbb{F}_q$ denoted $\beta = f_{ij}(\alpha)$, the update operation is:

$$U_{pc}[\beta_j] = \sum_i U_{vp}[\alpha_i] \qquad j = 0 \dots q-1, \;\; \beta_j = f_{ij}\left(\alpha_i\right)$$

  the summation is performed over all values α_i which have image β_j through the linear function f_{ij}.
- *Check node update:* this step is identical to BP decoder over finite fields and can be efficiently implemented using a Fast Fourier Transform. See *e.g.* [9], [12] for more details.
- *Inverse function node update:* with the use of the function $f_{ij}(.)$ backwards, *i.e.*, by identifying the values α_i which have the image β_j, the update equation is:

$$V_{pv}[\alpha_i] = V_{cp}[\beta_j] \qquad \forall \alpha_i \in \mathbb{F}_q \, : \, \beta_j = f_{ij}\left(\alpha_i\right)$$

These four steps define one decoding iteration of a general parity-check code on a finite group. Note

that, both in the finite field case and when the cluster defining the function $f_{ij}(.)$ is full rank, the function node update is simply a re-ordering of the values. When the cluster has deficient rank $r < p$, which is often the case when clustering a binary block code, only 2^r entries of the message U_{pc} are filled and the remaining entries are set to zero.

The purpose of this work is to show that with an iterative decoding procedure, it is possible to get close to the ML decoding performance of dense or locally-dense linear block codes. Therefore, we will not particularly focus on studying practical constraints such as decoding latency or implementation complexity. However, we still want our approach to be implementable if one accepts to increase the decoder complexity in order to get better error-correction results. For this reason, we will only consider the non-binary decoder in reasonable group orders, such as $\mathbb{F}_{64}$, $\mathbb{F}_{256}$ or $\mathbb{F}_{512}$ at most. Although non-binary decoders have a much higher computational complexity and storage complexity than binary decoders, a lot of work has been published recently that introduce sub-optimal non-binary decoders with complexity close to binary decoders (see [11] and the references within). The extended min-sum (EMS) algorithm studied in [11] has both computational complexity and storage requirements close to binary min-sum decoders, and with a performance loss within 0.1dB-0.5dB from the full-complexity non-binary BP decoder. This shows that the non-binary decoder diversity framework proposed in this paper, combined with a practical sub-optimal decoder could indeed be implemented in hardware.

III. Notion of Non-binary Decoder Diversity

A. Preprocessing and Tanner graph Diversity

Now that we have introduced all the necessary bases of our approach, we present in this section the main objective of this research, which relies on the diversity of behaviors of the NB-BP decoding when it is applied to different non-binary Tanner graphs representing the same code.

Let a block code be defined by a binary parity-check matrix H_b of size $(M \times N)$. Any linear transformation A on the rows and any column permutation Π applied to H_b does not change the code space[1], but changes the topology of the clustered Tanner graph. The application of A and Π is called pre-processing, and let us denote the pre-processed binary parity-check matrix by $H'_b = \mathcal{P}(H_b) = A\, H_b\, \Pi$.

In order to be able to use H'_b in the decoder when a codeword $\mathbf{m}$ corresponding to H_b is sent, it is necessary to re-order the received noisy codeword according to the inverse permutation Π^{-1}. Indeed,

[1]Only the binary mapping of the codewords change. The codes obtained with these transformations are equivalent codes, and we will refer to "equivalent representation of the *same code*" in the rest of the paper.

since $H'_b \left(\Pi^{-1}\,\mathbf{m}\right) = A\,H_b\,\Pi\,\left(\Pi^{-1}\,\mathbf{m}\right) = A\,H_b\,\mathbf{m} = \mathbf{0}$, if $\mathbf{y}$ is the output of a noisy memoryless channel with $\mathbf{m}$ as input, then $\mathbf{y}' = \Pi^{-1}\,\mathbf{y}$ should be used to compute the channel likelihoods that initialize a decoder based on H'_b.

The concept of *NB-BP decoder diversity* (NB-DD) relies on the fact that the NB-BP decoder is a dynamical system with sensitivity to initial conditions. As a consequence, if two representations of the same code are used in a NB-BP decoder, the dynamics of the iterative decoder applied to the two representations might be different and the convergence points also might be distinct. Please note that re-ordering of the columns and rows implies different dynamics of the decoder only if clustering is used, or equivalently if a *non-binary* message-passing decoder is considered. The re-ordering, with $A = \Pi_A$, would have no impact on the decoder behavior if a binary decoder is used. Let us illustrate this phenomenon through a small example.

As an example, we depict in Fig. 3 a binary random block code with clustering order $p = 2$, and the associated Tanner graph. We did not represent the linear function nodes associated with the linear functions $f_{ij}(.)$ in the graph for sake of simplicity. By applying a permutation of the columns, or equivalently of the bit positions in the codeword, one can see that the resulting Tanner graph has a completely different structure. In particular, the upper matrix has more all-zero clusters, and the Tanner graph has therefore fewer edges and fewer small cycles. We do not claim here that one case is necessarily a better graph representation for the NB-BP decoder than the other case. With this example, we only point out that the same code could have very different non-binary Tanner graph representations, which can possibly result in some diversity behavior in the decoder convergence.

In order to verify that the NB-BP decoder can exploit Tanner graph diversity, we have computed on the two graphs of figure 3 with 10000 AWGN noise vectors at $(E_b/N_0)_{dB} = 0dB$ the following statistics:

- 5019 cases where both decoders converge to the right codeword,
- 384 cases where both decoders fail to converge after 50 iterations,
- 589 cases where both decoders converge to a wrong codeword,
- 1213 cases where one decoder fails to converge, and the other one converges to the right codeword,
- 1041 cases where one decoder fails to converge, and the other one converges to a wrong codeword,
- 1754 cases where one decoder converges to a wrong codeword, and the other one converges to the right codeword.

Of course this is a basic example which corresponds to a very small code, but this shows quite clearly that all the possible decoder behaviors could happen when two decoders are used to decode the same

noisy codeword, and with the possibility of correcting up to 2967 extra cases, for which at least one decoder has converged to the right codeword.

The NB-DD approach presented in this paper relies on the following conjecture. We assume that -for a given noise realization- among the huge number of non-binary Tanner Graphs representing the same code, there exists a specific one which is well matched to this noise realization and such that the NB-BP decoder converges to the ML codeword.

This assumption is obviously true if there is no constraint on the complexity of the non-binary decoder. For any code, there exist a pre-processing function $\mathcal{P}$ and a sufficiently large clustering order p such that the non-binary Tanner graph is a tree and the decoder is optimal (for all noise realizations). The question of whether a well matched Tanner graph exists for a clustering order p small enough for practical implementation (namely $p \leq 10$) is more complicated and requires further research. In this paper, as a first step illustrating this conjecture, we will show by simulations that for dense or locally-dense block codes, if one uses several different representations of the same code, one can significantly improve the error-correction performance as compared to existing approaches.

B. Non-binary Decoder Diversity Sets

Let a block code be defined by a binary parity-check matrix H_b with size $(M \times N)$. Let $A^{(i)}$ be a binary matrix of full rank corresponding to a change of basis on the code, and let $\Pi^{(i)}$ be a permutation of the columns of H_b, or equivalently a permutation of the bits in the codewords. By applying the pre-processing $H_b^{(i)} = \mathcal{P}^{(i)}(H_b) = A^{(i)} H_b \Pi^{(i)}$ we get another valid binary parity check matrix of the same code. Now, let us denote by $\mathcal{G}_{p_i}^{(i)}$ the non-binary Tanner graph obtained from the clustering operation of order p_i on $H_b^{(i)}$. The Tanner graph $\mathcal{G}_{p_i}^{(i)}$ is composed of N/p_i symbol nodes, M/p_i general parity-check nodes, and a certain number of edges associated with the non-zero clusters. In general, the size of the parity-check matrix is not necessarily an integer multiple of p_i. In order to allow any clustering order, one can add redundant rows and all-zero columns to H_b so that the matrix can be divided in clusters of size $(p_i \times p_i)$ without changing the code space.

Definition

A diversity set $\mathcal{G}$ is defined as a collection of d distinct non-binary Tanner graphs associated with the same code and obtained through pre-processing $\mathcal{P}^{(i)}$ and clustering of order p_i, $i = 1, \ldots, d$:

$$\mathcal{G} = \left\{ \mathcal{G}_{p_1}^{(1)}, \ldots, \mathcal{G}_{p_i}^{(i)}, \ldots, \mathcal{G}_{p_d}^{(d)} \right\} \tag{3}$$

■

This definition is very general and we give hereafter some specific examples.

- *Clustering diversity only*:
  for all i, we choose $A^{(i)} = I_M$ and $\Pi^{(i)} = I_N$, where I_N is the identity matrix of dimension $(N \times N)$. In other words, we do not apply any pre-processing, and only p varies in the definition of the diversity set. For example, $\left\{\mathcal{G}_1^{(1)}, \ldots, \mathcal{G}_9^{(9)}\right\}$ is a diversity set with $d = 9$ Tanner graphs obtained from a *unique* binary parity-check matrix H_b. Note that $\mathcal{G}_1^{(1)}$ is the usual binary Tanner graph. This shows that since the diversity is defined in terms of graphs from clustered matrices, we can define a diversity set even without changing the binary parity-check matrix of the code.
- *Diversity with constant clustering order and common code basis*:
  for all i, we choose $A^{(i)} = \Pi_1^{(i)}$ and $\Pi^{(i)} = \Pi_2^{(i)}$, that is we consider only a re-ordering of the rows and columns of H_b. Furthermore, with the same clustering order p for all Tanner graphs, $\left\{\mathcal{G}_p^{(1)}, \ldots, \mathcal{G}_p^{(d)}\right\}$ is a diversity set such that all the graphs have the same number of nodes, and the same basis is used for all the code representations. This example of diversity set has the property that the binary density of $H_b^{(i)}$ remains the same since no linear transformation other than permutations are used (which is useful if the initial H_b has low density). However, the different Tanner graphs could have different densities as shown in Fig. 3. Note that the choice of $p = 1$ does not bring any diversity since all the graphs would have the exact same topology.
- *Diversity set with binary Tanner graphs*:
  with the choice of $p = 1$ (no clustering) and both $A^{(i)}$ and $\Pi^{(i)}$ can be general, the diversity set $\left\{\mathcal{G}_1^{(1)}, \ldots, \mathcal{G}_1^{(d)}\right\}$ defines a collection of binary Tanner graphs of the same code, represented in different basis. This special case has been recently studied in the literature [16], where the authors have conducted a performance/complexity comparison when multiple bases of a block code are used. Note that using different binary basis is also very similar to the methods proposed in [7], [8], in which the authors propose to extend the binary representation of the code by introducing redundant rows, with the goal of breaking the correlation effects which can prevent the binary BP decoder from converging to a fixed point.

These few examples show that there are many possible ways of defining a diversity set and we emphasize here that in our opinion, the choice of the diversity set is very important in order to effectively have a gain in terms of error-correction. For each of the three simulation studies that we performed (see section IV) a specific diversity set has been carefully chosen, adapted to each case.

C. Merging Strategies

We discuss now the different strategies that we conducted in order to capitalize on the NB-DD. There are various possible merging methods that one can think of in order to use the outputs of each decoder, with associated performance complexity tradeoffs. Aside from the two natural merging strategies depicted below, one can think of more elaborate choices.

- *Serial merging*:

  the d decoders are potentially used in a sequential manner. Assuming that we check the value of the syndrome at each iteration, when a decoder fails to converge to a codeword after a given number of iterations, we switch to another decoder. That is, another Tanner graph is computed with a different pre-processing and we restart the decoder from scratch with the new graph and the permuted likelihood. The process stops when one decoder converges to a codeword (either the sent codeword or another one). This strategy is different from the one proposed in [8] where the authors iterate the message-passing decoder iteratively between different sets of redundant nodes. In our case, when we switch to another decoder, the messages in the new graph are reset to zero and we start the next decoder from scratch.

- *Parallel merging*:

  the d decoders are used in parallel and a MLD decision is taken among the ones that have converged to a codeword. If $nb \leq d$ decoders have converged to a codeword in less than the maximum number of iterations, the nb associated likelihoods are computed and the one with the maximum likelihood is selected. Note that the nb candidate codewords are not necessarily distinct. A similar merging strategy with different binary bases has been proposed in [16].

- *Lower bound* on merging strategies:

  in order to study the *potential* of the NB-DD approach regardless of the merging strategy, we define the following lower bound. Among the d decoders in the diversity set, we check if at least one decoder converges to the right codeword. A decoder failure is decided if all d decoders have not converged to the right codeword after the maximum number of iterations. Note that this method does not exhibit any undetected error. This is called a lower bound on merging strategies, because it assumes that if there exists at least one Tanner graph which converges to the right codeword, one can think of a smart procedure to select this graph. This is of course not always possible, especially if the codeword sent is different than the codeword decoded with MLD, and the lower bound can therefore be better than the MLD itself. This lower bound is however useful since it allows to have

a possibly tight estimation on the parallel merging case, without having to simulate all d decoders.

The extra complexity induced by the serial merging is negligible since the diversity is to be used only when the first decoder fails to converge. That is, at a FER=10^{-3} for the first decoder, the decoder diversity will be used only 0.1% of the time. The parallel merging is much more complex since it uses d times more computations than a single decoder, but eventually leads to better performance.

Note that in this paper, the choice of a particular merging is not studied. Only the use of several binary representations is defended. However, it should be mentioned that the merging strategy is linked to the additional complexity of the NB-DD approach compared to the use of only one decoder. The parallel merging will not be used in simulations, and we will only compare the serial merging and the lower bound, which correspond to the extreme situation for a system using NB-DD.

Finally, let us point out that an interesting development would be to combine the approach presented in this paper with the idea of extending the graph representation with redundant rows, as proposed in [8], and run a message-passing decoder on the global graph with a well chosen scheduling. This is although a non-trivial generalisation since there is no simple way to capitalize on the decoding output of a particular graph in the diversity set and propagate a message to another graph representation. Indeed, since we consider binary pre-processing functions on the binary parity-ckeck matrices of the code, one would need to marginalize the non-binary messages at the binary level and then re-combine them in the correct non-binary symbols before the decoder can start in another part of the graph. This strategy did not work at all in our simulations, at least when random pre-processing functions were considered. It appears that running a BP-based decoder on a redundant representation of the code, and therefore on a denser Tanner graph, seems to work better for binary decoders than for non-binary decoders.

IV. Simulation Results for Various Examples

In this section, we present in detail three examples for which the NB-DD technique has shown its advantage. The three examples are very different, both with respect to the transmission model (channel, code families), and with respect to the chosen diversity set.

Before studying the different cases, it should be noted that the performance gain brought by NB-DD must be fairly compared to the performance obtained with a single decoder. The fair comparison is ensured when NB-DD with $d = 1$ corresponds to a single decoder with reasonable performance. In order to do so, and in all the examples in this section, we have carefully chosen the diversity sets such that all decoders in the set have comparable average performance but more importantly, they have each very

good performance when used separately. This particularly means that the observed performance gains are indeed due to the diversity of the NB-DD approach.

A. BCH codes on the BI-AWGN Channel

1) Diversity Set: In this section, we consider BCH codes sent over the BSPK-AWGN channel. For such a case, it has been shown that it is necessary to adapt the parity-check matrix so that an iterative decoder has acceptable performance. The method developed in [14] is to change adaptively the binary Tanner graph such that the least reliable bits are connected only to one check node, and therefore do not propagate much noise in one decoding iteration. The first step of this method is to sort the channel likelihoods (or *a posteriori* probabilities after the first iteration) in ascending order and diagonalize the binary parity-check matrix in such a way that one obtains the parity-check matrix represented in its most reliable basis (MRB). This method is computationally intensive since it usually requires a Gaussian elimination at each and every decoding iteration. It has been recently proposed in [15] to perform the Gaussian elimination and obtain the MRB only at the initialization of the decoder, which is sufficient to get good performance if the block length is not too large ($N \leq 64$ bits). This latter algorithm is called modified adaptive BP (m-ABP).

In this section, we define the diversity set with the knowledge of the MRB. We first perform a MRB Gaussian elimination based on the channel likelihood values, and store the transformation matrices A_{MRB} and Π_{MRB}. Therefore, the binary matrix $A_{MRB}\, H_b\, \Pi_{MRB}$ is diagonalized in its MRB. Then, we wish to build a diversity set such that for all candidate Tanner graphs, *(i)* the bit indices which form the MRB basis are localized in the same part of the codeword, *(ii)* the least reliable symbols are connected only to one check node of the non-binary Tanner graph, that is only one non-zero cluster in the first part of the parity-check matrix $H_b^{(i)}$ is allowed.

The most general pre-processing which follows these constraints is of the type:

$$H_b^{(i)} = \mathcal{P}^{(i)}\left(H_b\right) = \left(A^{(i)}\, A_{MRB}\right)\, H_b\, \left(\Pi_{MRB}\, \Pi^{(i)}\right) \tag{4}$$

where A_{MRB} and Π_{MRB} are obtained from the MRB Gaussian elimination, and therefore are the same

for all pre-processing in the diversity set. The other transformations have the following structure:

$$A^{(i)} = \begin{pmatrix} \mathbf{a}_1^{(i)} & 0 & \dots & 0 \\ 0 & \mathbf{a}_2^{(i)} & \dots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{a}_{M/p}^{(i)} \end{pmatrix} \qquad \Pi^{(i)} = \begin{pmatrix} I_M & 0 \\ 0 & \Pi_K^{(i)} \end{pmatrix}$$

That is $A^{(i)}$ is block diagonal with full rank clusters $\mathbf{a}_k^{(i)}$ of size $(p \times p)$, and $\Pi^{(i)}$ permutes only the $K = N - M$ most reliable bits. In the diversity set, $A^{(i)}$ and $\Pi^{(i)}$ are generated at random.

Using a diversity set based on these pre-processing strategies ensures that all the decoders have comparable average performance. This is also a very special use of the NB-DD approach since the diversity sets differ from one noise realization to another, because of the MRB Gaussian elimination. We will see in the next examples more natural uses of NB-DD, for which the diversity set is built regardless of the noise realization.

2) Frame Error Rate Results: We consider the following codes as case studies: eBCH$(N = 128, K = 92, D_{min} = 12)$, BCH$(N = 127, K = 71, D_{min} = 19)$ and eBCH$(N = 256, K = 131, D_{min} = 38)$.

In Fig. 4, we plot the NB-DD results using the proposed diversity set and a maximum diversity order $d = 200$, with two clustering orders $p = 6$ and $p = 9$. The maximum number of decoding iterations was set to 100. The performance of the m-ABP of [15] is shown and the tangential sphere bound (TSB) computed with the code distance spectrum is used as maximum-likelihood decoding (MLD) lower bound. There are several important comments to make based on this figure:

- First, the proposed NB-DD approach has the potential to reach MLD performance on this code since the lower bound with parameters $(p = 9, d = 200)$ sticks to the TSB curve. This shows that with a diversity order of $d = 200$ Tanner graphs, there exist at least one of these graphs for which the NB-BP would converge to the ML codeword. We verified that for all noisy codewords, there is no dominant graph and that the use of the diversity set is therefore necessary to be able to reach these results. Indeed, the graphs in the diversity set which converge for a particular noise realisation are not the same as those which converge for a different noise realisation.
- The performance obtained with the serial merging, which corresponds to the minimum complexity NB-DD scheme, is only $0.35dB$ away from the TSB at a $FER = 10^{-5}$ which is still the best performance that we have observed in the literature with iterative decoders on block codes with comparable length and rates. For instance, the binary m-ABP has a $1.4dB$ loss compared to the TSB at $FER = 10^{-5}$. Note that although close to the lower bound, the performance loss of the

serial merging is relatively big, and this is due to the poor minimum distance of this BCH code, which result in an large number of undetected errors.

- The diversity gain in this case is not really great since we gain less than one decade by using a maximum of $d = 200$ decoders instead of a single one (remember that the 200 decoders are seldom used in the decoding process). This is due to the fact that the MRB Gaussian elimination adapts the graph to the channel statistics. In this sense, the graphs selected in the diversity set are already well matched to the noise realization, and the NB-DD gain is not impressive. The fact that the lower bound sticks to the TSB curve shows however that NB-DD could bring a non-negligible performance gain with a proper merging strategy or a proper pre-processing choice.
- Finally, we can see that with a smaller clustering order $p = 6$, the results are close to the $p = 9$ results, which shows that reducing the complexity of the NB-BP decoders can result in a interesting complexity/performance tradeoff.

The same remarks can be made about the BCH$(N = 127, K = 71, D_{min} = 19)$ code, for which we plot the same set of curves in Fig. 5. The clustering orders that we considered for this code are $p = 4$ and $p = 8$. This code has almost the same length as the eBCH$(N = 128, K = 92, D_{min} = 12)$, but has a larger minimum distance and a rate closer to $R = 0.5$. For this code, the lower bound of NB-DD is very close to the TSB, which we think is a very promising result. At a $FER = 10^{-5}$, the lower bound on NB-DD is 1.9dB better than the m-ABP solution of [15], and with the serial merging the gain is still important (1.2 dB better).

Although not shown in the figure, we have studied the performance evolution with increasing values of the diversity order d for the BCH$(N = 127, K = 71, D_{min} = 19)$ code. For the serial merging, the same performance can be obtained with a lot smaller diversity order, as we observed only a negligible gain from $d = 80$ to $d = 200$. However, when considering the lower bound on merging strategies, the performance are still improving beyond $d = 80$. A number of $d = 200$ decoders was the smallest number such that the NB-DD reaches the TSB bound. For example, NB-DD with $d = 200$ has a FER roughly twice better than with $d = 100$ and four times better than with $d = 50$.

Finally, we plot in Fig. 6 the performance of NB-DD on the eBCH$(N = 256, K = 131, D_{min} = 38)$ code which is a very challenging code for which no decoder with reasonable complexity approaches the performance of MLD. We can see that the gap between NB-DD and the TSB is approximately 1.4dB which means that MLD performance for this code is for the moment out of the reach of NB-DD. Our method has however a gain of about 1.8dB compared to binary m-ABP. Note that in this case, the gap between the lower bound NB-DD and the serial merging is very small, and this is due to the fact that

the large minimum distance of the code limits the number of undetected errors in the NB-BP.

B. BCH codes on the BSC Channel

1) Diversity Set: In this section, we consider BCH codes sent over the binary symmetric channel. For this channel where the errors are flipped bits, there is no possibility of having a well chosen matrix representation, matched to the channel outputs, as we did with the MRB Gaussian elimination in the previous section.

This is a good example which shows the great advantage of NB-DD. Since there is no possibility to choose *a priori* a good Tanner graph for a particular noise realization, we rely on the possibility that among d distinct graphs, there exist one graph which is best matched to the noise. In order to show that this is indeed the case, we will only simulate the performance of NB-DD with the lower bound on merging strategies approach, which means that we only focus on finding a Tanner graph among the d candidates, such that the NB-BP decoder converges to the right codeword.

The only thing we can do to help the decoding process is to diagonalize the parity-check matrix (in any basis), in order to maximize the number of all-zero clusters in the non-binary Tanner graph. Let us consider H_b, the binary parity check matrix of a BCH code in its systematic form. The diversity set is composed of the pre-processing functions of the type:

$$H_b^{(i)} = \mathcal{P}^{(i)}\left(H_b\right) = A_{GE}\, H_b\, \left(\Pi_N^{(i)}\, \Pi_{GE}\right) \tag{5}$$

where $\Pi_N^{(i)}$, generated at random, is used to permute the N columns of H_b, and (A_{GE}, Π_{GE}) are obtained from the Gaussian elimination. Once more, all the candidates in the diversity set have similar average performance.

2) Frame Error Rate Results: We plot the frame error rate with respect to a normalized signal to noise ratio $(E_b/N_0)_{dB} = 10\,\log_{10}(E_b/N_0)$ from which the probability of error p_{err} for the BSC is computed with,

$$p_{err} = \frac{1}{2}\text{erfc}\left(\sqrt{R\,\frac{E_b}{N_0}}\right) \tag{6}$$

where erfc is the complementary error function, and R the rate of the code. This representation has the advantage that we can measure the performance loss due to hard decision before decoding in a Gaussian channel.

In Fig. 7 and Fig. 8, we plot only the lower bound of NB-DD, that is the best performance we can expect from our method, and compared it to the bounded distance decoding performance, that is assuming an algebraic decoder which corrects all noise patterns of Hamming weight less than $\lfloor \frac{D_{min}-1}{2} \rfloor$. We used

a clustering order $p = 9$ for the eBCH$(N = 128, K = 92, D_{min} = 12)$ code and a clustering order of $p = 8$ for the BCH$(N = 127, K = 71, D_{min} = 19)$ code.

We can see on both figures that using one decoder is by far not sufficient to have good error-correction, since the performance is outperformed by a simple bounded distance decoder. However, the NB-DD approach shows an impressive performance gain, especially at high E_b/N_0. Indeed, at $(E_b/N_0)_{db} = 6.5dB$ in Fig. 7, using a diversity order of $d = 2000$ allows a gain of almost 5 decades compared to a single decoder. Moreover, the NB-DD is $1dB$ better than a bounded distance decoder for all frame error rates below 10^{-5}, which is a very important gain for this problem, that is decoding dense block codes over the BSC.

These good performance results are even enhanced for the BCH$(N = 127, K = 71, D_{min} = 19)$ code in Fig. 8. One can see also that for this case, the NB-DD is fully used even with a large diversity order since there is an important gap between a diversity order $d = 200$ and a diversity order $d = 2000$.

C. Turbo-codes on the BI-AWGN Channel

1) Diversity Set: In this last example, we consider the problem of decoding a turbo-code on a BPSK-AWGN channel with a NB-BP decoder, as introduced in [17].

Binary parity-check matrices of convolutional codes are typically concentrated on the diagonal of the matrix and are obtained through algebraic arguments [18]. That is, the parity-check matrix is sparse, but locally dense, which is a good configuration for using with a NB-BP decoder on the clustered matrix. Of course, when considering turbo-codes, we still get some locally dense parts in the binary parity-check matrix of the code, except for the parts which correspond to the turbo interleaver. In [17], the authors have proposed a pre-processing function specific to turbo-codes optimized in order to have the most concentrated diagonal in the parity-check matrix, which is also equivalent to minimizing the number of non-zero clusters that will be created on the diagonal after the clustering process. The binary representation that is obtained with the technique proposed in [17] has the structure depicted in Fig. 9.

Binary representations such as the one in Fig. 9 are supposed to be a good choice since the clusters on the diagonal are the more dense in the Tanner graph, and are assumed to participate the most to the performance degradation of the BP decoder when these clusters contribute to cycles. Indeed, we have verified by simulations on several turbo-codes that the number of non-zero clusters of a given size is minimized when we perform a pre-processing resulting in a binary representation similar to figure 9. Note that by properly choosing the columns to be permuted, several representations of this type could be created. In this paper we will consider $d = 5$ such parity-check matrices which form the diversity set.

Contrary to the two previous examples, here the diversity order is very small, but each Tanner graph is supposed to exhibit very good performance under NB-BP decoding since they are very sparse.

2) Frame Error Rate Results: We present in this section simulation results for the duo-binary turbo-codes of the DVB-RCS standard, which were first presented in [19]. The considered code parameters are $R = 0.5$ and size $N = \{3008\}$ coded bits, with tail-biting trellis termination. The minimum distance of this code is $D_{min} = 19$.

The NB-DD results are plotted in Fig. 10. If we focus on the maximum performance gain that one can hope for by looking at the lower bound curves, it is clear that using several decoders can improve significantly the performance, both in the waterfall region and the error floor region. Using NB-BP decoding with decoder diversity can gain between 0.25dB to 0.4dB compared to the turbo-decoder using BCJR component decoders, which was up to now considered as the best decoder proposed for turbo-codes. This result shows in particular that it is possible to consider iterative decoders which are more powerful, and therefore which are closer to the maximum-likelihood decoder, than the classical turbo-decoder. This claim was known from asymptotic density evolution results, but to the best of our knowledge, our decoder is the first that could practically, that is with complexity reasonable enough to be implemented on a chip, beat the turbo-decoder.

Interestingly, the serial merging which is the more simple merging strategy, achieves full decoder diversity gain in the waterfall region, *i.e.*, above FER$= 10^{-3}$. This is particularly useful for wireless standards which use ARQ based transmission and therefore hardly require error-correction below FER$= 10^{-3}$. In the error floor region though, we can see in Fig. 10 that more elaborate merging solutions should be used to achieve full diversity gain and obtain a substantial gain compared with turbo decoder.

V. Conclusion

In this paper, we have presented a new framework, called “non-binary decoder diversity” based on the observation that different non-binary Tanner graphs of the same code, decoded with a non-binary BP decoder can have distinct convergence behaviors and fixed points. This framework is especially interesting for binary codes which are dense or locally-dense, and for which the usual binary iterative decoders perform far from the optimum curves. We focused our study on deriving an adapted strategy for three very different test cases which are known to be complex decoding problems, and in all cases, the proposed approach showed promising results, either close to the theoretical bound or outperforming existing decoders.

The results presented in this paper show the potential of the non-binary decoder diversity approach to

get close to MLD performance. This is however the first initial step and further work will be dedicated to the characterization and the optimization of the non-binary Tanner graph, in order to match the graph to a particular noise realization.

ACKNOWLEDGMENT

The author would like to thank Charly Poulliat and Marc Fossorier for interesting discussions related to this work and Marc Fossorier for providing the TSB curves. The author thanks also the reviewers for their careful reading of this paper, and for their remarks which helped to improve the quality of the presentation.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shanon limit error-correcting coding and decoding : Turbo codes", *in the proc. of ICC 1993*, Geneva, Switzerland, 1993.

[2] F. Kschischang, B. Frey, and H-A. Loeliger, "Factor Graphs and the Sum Product Algorithm", *IEEE Trans. on Info. Theo.*, vol. 47, pp. 498-519, February 2001.

[3] R. McEliece, D. MacKay and J-F. Cheng, "Turbo decoding as an instance of Pearl's belief-propagation algorithm", *IEEE Journal on Selected Areas in Communications*, vol. 16(2), pp. 140-152, February 1998.

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

[5] R. M. Tanner, "A Recursive Approach to Low Complexity Codes", *IEEE Trans. on Info. Theo.*, vol. 27, pp. 533-547, September 1981.

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

[7] M. Fossorier, R. Palanki and J. Yedidia, "Iterative Decoding of Multi-Step Majority Logic Decodable Codes", *in the proc. of Turbo-codes 2003*, Brest, France, 2003.

[8] T. Halford and K. Chugg, "Random Redundant Iterative Soft-in Soft-out Decoding", *IEEE Trans. on Info. Theo.*, vol. 56(4), pp. 513-517, April 2008.

[9] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes over GF(q)," *IEEE Trans. on Commun.*, vol. 55(4), pp. 633-643, April 2007.

[10] C. Poulliat, M. Fossorier and D. Declercq, "Design of regular (2,dc)-LDPC codes over GF(q) using their binary images", *IEEE Trans. on Commun.*, vol. 56(10), pp. 1626-1635, October 2008.

[11] A. Voicila, D. Declercq, F. Verdier, M. Fossorier and P. Urard, "Low Complexity Decoding for Non-Binary LDPC Codes in High Order Fields", *to appear in IEEE Trans. on Commun.*, 2010.

[12] A. Goupil, M. Colas, G. Gelle and D. Declercq, "FFT-based BP Decoding of General LDPC Codes over Abelian Groups," *IEEE Trans. on Commun.*, vol. 55(4), pp. 644-649, April 2007.

[13] J. Yedidia, W. Freeman and S. Weiss, "Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms," *IEEE Trans. on Info. Theo.*, vol. 51(7), pp. 2282-2312, July 2005.

[14] J. Jiang and K.R. Narayanan, "Iterative Soft-Input-Soft-Output Decoding of Reed-Solomon Codes by Adapting the Parity Check Matrix", *IEEE Trans. on Info. Theo.*, vo. 52(8), pp. 3746-3756, August 2006.

[15] C. Jego and W. Gross, “Turbo Decoding of Product Codes based on the Modified Adaptive Belief Propagation Algorithm”, *in the proc. of IEEE ISIT*, Nice, France, 2007.

[16] T. Hehn, J. Huber, O. Milenkovic and S. Laendner , “Multiple-Bases Belief-Propagation Decoding of High-Density Cyclic Codes”, *IEEE Trans. on Commun.*, vol. 58(1), pp. 1-8, January 2010.

[17] C. Poulliat, D. Declercq and T. Lestable, “Preprocessing for an efficient decoding of Turbo-codes with non-binary Belief Propagation” *in the proc. of ISTC*, Lausanne, Switzerland, 2008.

[18] R. Johannesson and K. Sh. Zigangirov, “Fundamentals of Convolutional Coding”, ser. Digital, Mobile Communication, ch. 1-2, New York: IEEE Press, 1999.

[19] C. Douillard and C. Berrou, “Turbo Codes with rate $m/(m+1)$ Constituent Convolutional Codes”, *IEEE Trans. on Commun.*, vol. 53(10), pp. 1630-1638, Oct. 2005.

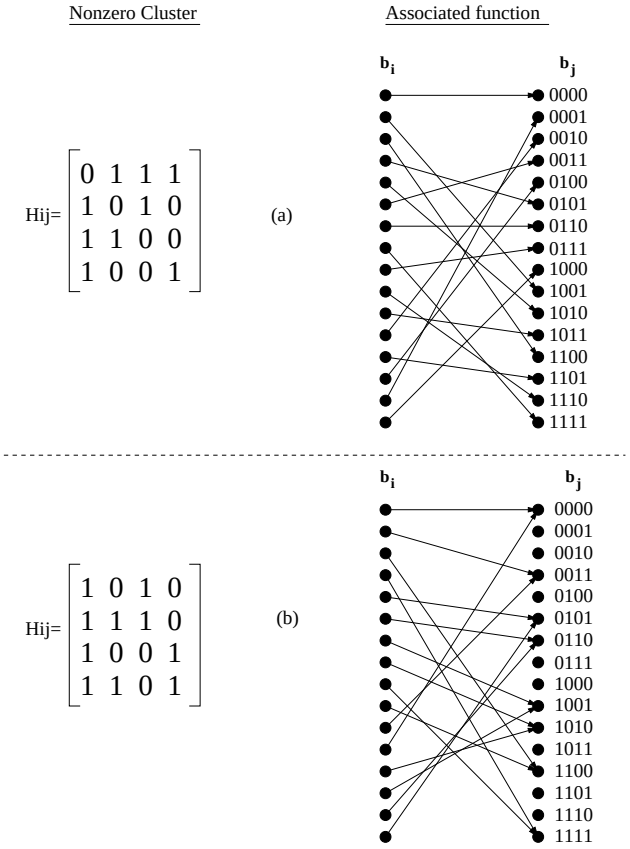

Fig. 1. A binary cluster H_{ij} and its associated linear function $f_{ij}(.)$: (a) the full rank case and (b) the rank deficient case. The output vector is obtained by $\mathbf{b}_j = H_{ij}\,\mathbf{b}_i$.

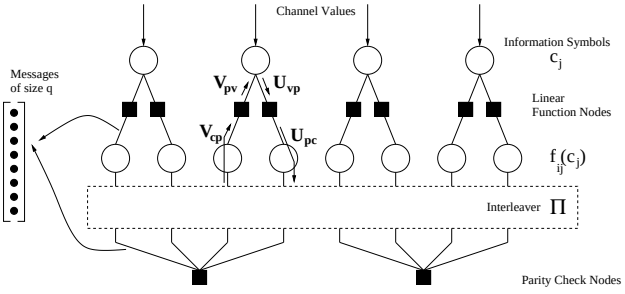

Fig. 2. Tanner graph of a non-binary LDPC code defined over a finite group $\mathbb{F}_q$.

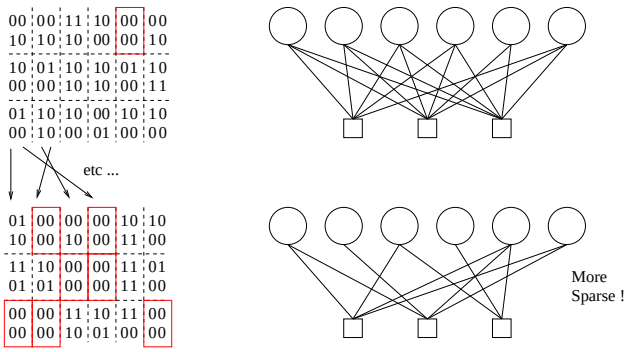

Fig. 3. Diversity of non-binary Tanner graph representations for the same code.

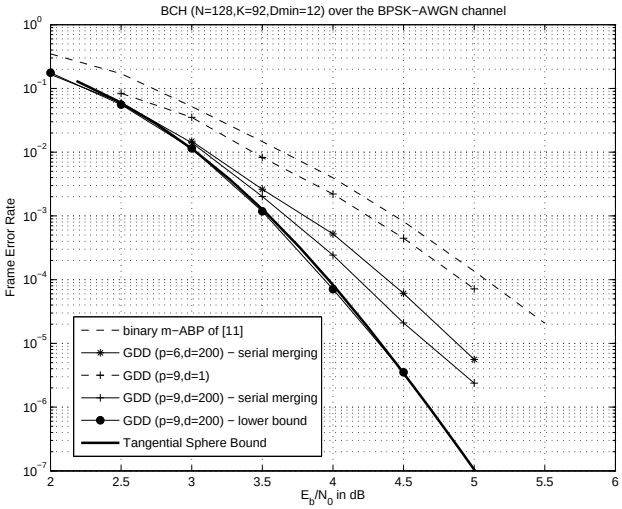

Fig. 4. Performance of NB-DD for the eBCH$(N = 128, K = 92, D_{min} = 12)$ code on the BPSK-AWGN channel

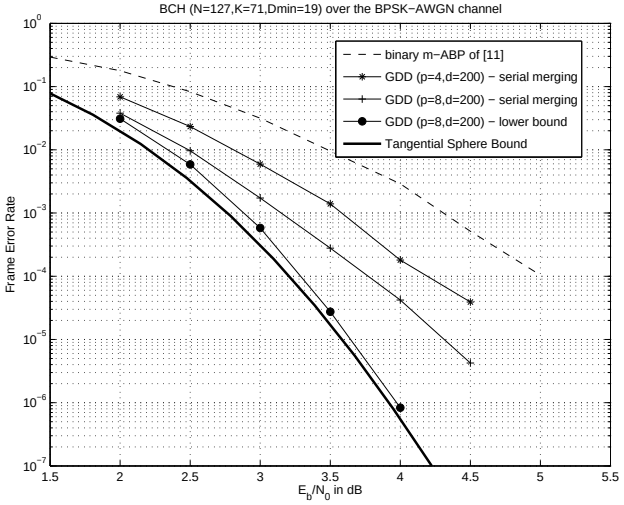

Fig. 5. Performance of NB-DD for the BCH$(N = 127, K = 71, D_{min} = 19)$ code on the BPSK-AWGN channel

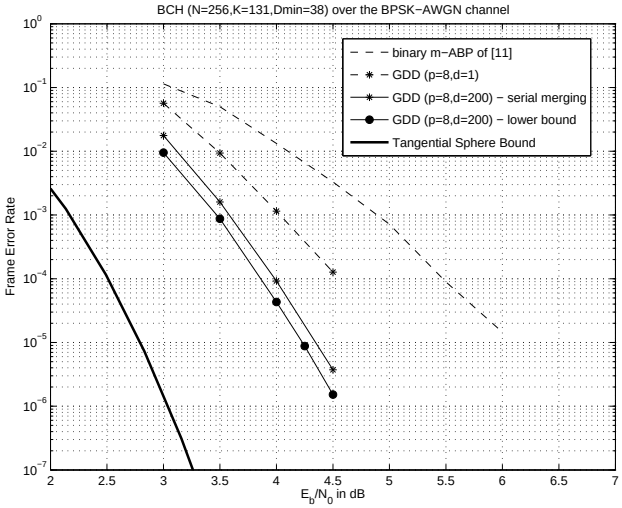

Fig. 6. Performance of NB-DD for the eBCH($N = 256, K = 131, D_{min} = 38$) code on the BPSK-AWGN channel

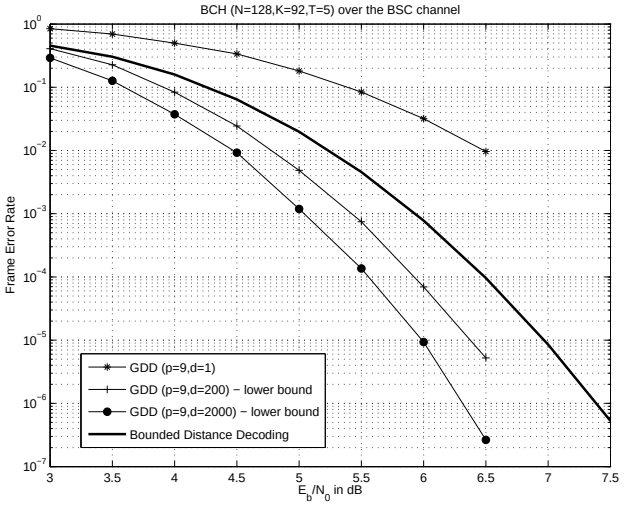

Fig. 7. Performance of NB-DD for the eBCH($N = 128, K = 92, D_{min} = 12$) code on the BSC

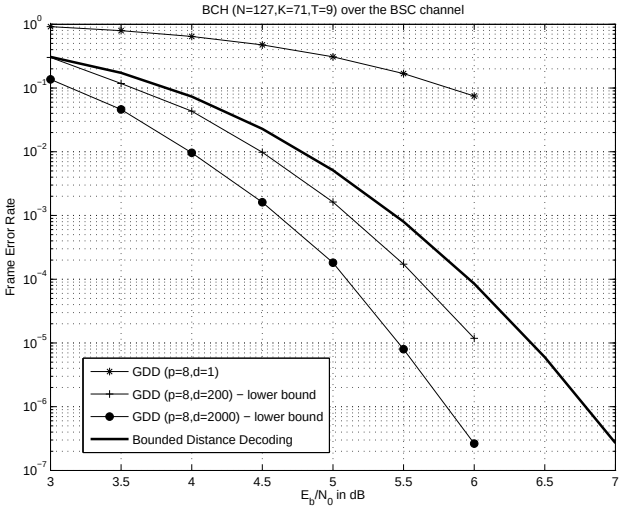

Fig. 8. Performance of NB-DD for the BCH$(N = 127, K = 71, D_{min} = 19)$ code on the BSC

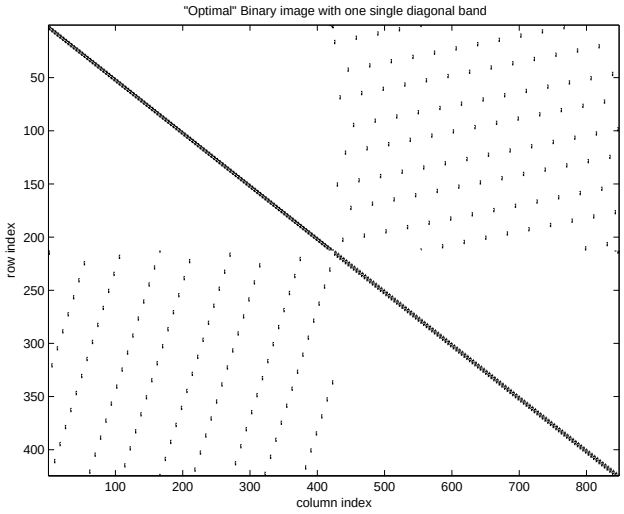

Fig. 9. Optimized binary representation of a parallel Turbo-code

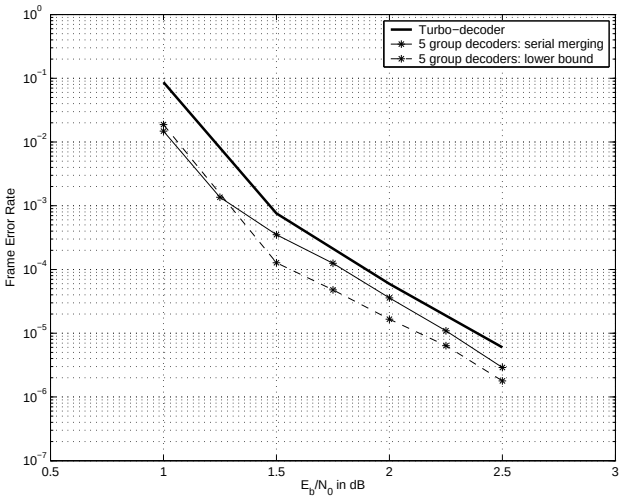

Fig. 10. Performance of decoder diversity applied to the (R=0.5,N=3008) duobinary Turbo Code.