Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2008, Article ID 473613, 10 pages
doi:10.1155/2008/473613

Research Article

Efficient Decoding of Turbo Codes with Nonbinary Belief Propagation

Charly Poulliat,[1] David Declercq,[1] and Thierry Lestable[2]

[1] *ETIS laboratory, UMR 8051-ENSEA/UCP/CNRS, Cergy-Pontoise 95014, France*
[2] *Samsung Electronics Research Institute, Communications House, South Street, Staines, Middlesex TW18 4QE, UK*

Correspondence should be addressed to Charly Poulliat, charly.poulliat@ensea.fr

Received 31 October 2007; Revised 25 February 2008; Accepted 27 March 2008

Recommended by Branka Vucetic

This paper presents a new approach to decode turbo codes using a nonbinary belief propagation decoder. The proposed approach can be decomposed into two main steps. First, a nonbinary Tanner graph representation of the turbo code is derived by clustering the binary parity-check matrix of the turbo code. Then, a group belief propagation decoder runs several iterations on the obtained nonbinary Tanner graph. We show in particular that it is necessary to add a preprocessing step on the parity-check matrix of the turbo code in order to ensure good topological properties of the Tanner graph and then good iterative decoding performance. Finally, by capitalizing on the diversity which comes from the existence of distinct efficient preprocessings, we propose a new decoding strategy, called decoder diversity, that intends to take benefits from the diversity through collaborative decoding schemes.

1. INTRODUCTION

Turbo codes and low-density, parity-check (LDPC) codes have long been recognized to belong tothe family of modern error correctingcodes. Although often opponents in standards and applications, these two classes of codes share common properties, the most important one being that they have a sparse graph representation that allows to decode them efficiently using iteratively whether the maximum a posteriori (MAP) algorithm [1] for turbo codes, or the belief propagation (BP) algorithm for LDPC codes [2], as well as their low-complexity iterative decoders.

Moreover, LDPC and turbo codes are two coding candidates which are often options within the same system [3, 4]. It is thus interesting to investigate common architecture/algorithm at the receiver side to enable switching easily among them, whilst still maintaining reasonable cost and area size.

Even if turbo codes effectively exhibit a sparse factor graph representation for which the BP decoder is equivalent to the so-called turbo decoder [5, 6], this factor graph representation is composed of different types of nodes, both for variable and for function nodes, which are not reduced to parity-check constraints (see [5] for more details). Later, some researchers have tried to use a factor graph representation of the turbo code based only on parity-check equations [7]. We will refer to a factor graph with only parity-check constraints for the function nodes (binary or not) as Tanner graph in the rest of the paper [8].

The classical BP algorithm (sometimes called sum-product) on the Tanner graph of a turbo code does not perform sufficiently well to compete with the turbo decoder performance [7]. This is mainly due to the inherent presence of many short cycles of length 4, that lead to a poor convergence behavior inducing loss of performance. In order to solve the problem of these short cycles, in [9, 10] the authors propose to use special convolutional codes as components of the turbo code, called *low-density convolutional codes*, for which an iterative decoder based ontheir Tanner graph experiences has less statistical dependence, and therefore exhibits very good performance.

Our approach is different from [10] since we aim at having a generic BP decoder which performs close to the best performance, without imposing any constraint on the component code. In this paper, we present a new approach to decode parallel turbo codes (i.e., binary, duobinary, punctured or not, etc.) using a nonbinary belief propagation decoder. The generic structure of the proposed iterative decoder is illustrated in Figure 1. The general approach can be decomposed into two main steps: the first step consists

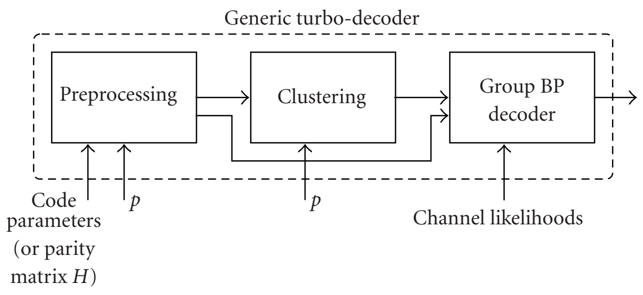


FIGURE 1: Block representation of the generic turbo decoder based on group BP decoder.

in building a nonbinary Tanner graph of the turbo code using only parity-check nodes defined over a certain finite group, and symbol nodes representing groups of bits. The Tanner graph is obtained by a proper clustering of order p of the binary parity-check matrix of the turbo code, called "binary image." However, the clustering of the commonly used binary representation of turbo codes appears to be not suitable to build an nonbinary Tanner graph representation that leads to good performance under iterative decoding. Thus, we will show in the paper that there exist some suitable preprocessing functions of the parity-check matrix (first block of Figure 1) for which, after the bit clustering (second block of Figure 1), the corresponding nonbinary Tanner graphs have good topological properties. This preliminary two-round step is necessary to have good Tanner graph representations that outperform the classical representations of turbo codes under iterative decoding. Then, the second step is a BP-based decoding stage (last block in Figure 1) and thus consists in running several iterations of group belief propagation (group BP), as introduced in [11], on the nonbinary Tanner graph. Furthermore, we will show that the decoder can also fully benefit from the decoding diversity that inherently raises from concurrent extended Tanner graph representations, leading to the general concept of decoder diversity. The proposed algorithms show very good performance, as opposed to the binary BP decoder, and serve as a first step to view LDPC and turbo codes within a unified framework from the decoder point of view, that strengthen the idea to handle them with a common approach.

The remaining of the paper is organized as follows. In Section 2, we describe how to decode turbo codes based on group BP decoder. To this end, we review how to derive the binary representation of the parity-check matrix H_{tc} of a parallel turbo code. Then, we explain how to build the non-binary Tanner graph of a turbo code based on a clustering technique and describe the group BP decoding algorithm based on this representation. In Section 3, we discuss how to choose a posteriori good matrix representations and how to take advantage of the inherent diversity that is offered by concurrent preprocessing in the decoding process. To this end, we present some choices for the required preprocessing of the matrix H_{tc} before clustering to build a Tanner graph with good topological properties, that performs well under group BP decoding. Then, we introduce in Section 4 the concept of decoder diversity and show how it can be used to further enhance performance. Finally, conclusions and perspectives are drawn in Section 5.

2. DECODING A TURBO CODE AS A NONBINARY LDPC CODE

In this Section, we present the different key elements that enable to decode turbo codes as nonbinary LDPC codes defined over some extended binary groups. First, we briefly review how to derive the binary representation of the parity-check matrix H_{tc} of a parallel turbo code based on the parity-check matrix of a component code. Then, we explain how to build the nonbinary Tanner graph of a turbo code based on a clustering technique and describe how the group BP decoding algorithm can be used to efficiently decode turbo codes based on this extended representation.

2.1. *Binary parity-check matrix of a turbo code*

The first step in our approach consists in deriving a binary parity-check matrix representation of the turbo code. We will only focus in this paper on parallel turbo codes with identical component codes.

2.1.1. *Parity-check matrix of convolutional codes*

The binary image of the turbo code is essentially based on the binary representation of the parity-check matrices of its component codes. Following the derivations presented in [12], the parity-check matrix for both feedforward convolutional encoders and their equivalent recursive systematic form is generally derived using the Smith's decomposition of its polynomial generator matrix $G(D)$, where $G(D)$ is a $k \times n$ matrix that gives the transfer of the k inputs into the n outputs of the convolutional encoder and D is defined as the delay operator (please refer to [12] for more details about this decomposition). From this decomposition, the polynomial syndrome former matrix $H^T(D)$ [12], of dimensions $n \times (n-k)$, can be derived and it can be expanded as

$$H^T(D) = H_0^T + H_1^T D + \cdots + H_{m_s}^T D^{m_s}, \tag{1}$$

where H_i^T, $0 \le i \le m_s$ is a matrix with dimensions $n \times (n-k)$, and m_s is the maximum degree of polynomials in $H^T(D)$. For both feedforward convolutional encoders and their recursive systematic form, it is possible to derive the binary image from the semi-infinite matrix H^T given by

$$H^T = \begin{pmatrix} H_0^T & H_1^T & \dots & H_{m_s}^T & & \\ & H_0^T & H_1^T & \dots & H_{m_s}^T & \\ & & \ddots & \ddots & & \ddots \end{pmatrix}. \tag{2}$$

When direct truncation is used, it is possible to derive from H^T the finite length binary parity-check matrix with dimension $(N-K)\times N$, given by

$$H=\begin{pmatrix} H_0 & & & & \\ H_1 & H_0 & & & \\ \vdots & \ddots & H_0 & & \\ H_{m_s} & & & & \\ & H_{m_s} & & & \\ & & \ddots & \ddots & \\ & & & H_{m_s} \ \dots & H_0 \end{pmatrix}, \tag{3}$$

where N and K are the codeword and information block lengths, respectively.

Under some length restrictions for the recursive case [13, 14], it is also possible to derive the binary image of the parity-check matrix of the tail-biting code H_{tb} from the parity-check matrix H [15] for feedforward convolutional encoders and their recursive systematic form. This can finally be represented as follows using the so-called "wrap around" technique:

$$H_{tb}=\begin{pmatrix} H_0 & & & H_{m_s} & \dots & H_1 \\ H_1 & H_0 & & & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & H_{m_s} \\ H_{m_s} & & H_0 & & & \\ & H_{m_s} & & & & \\ & & \ddots & & \ddots & \\ & & & H_{m_s} & H_{m_s-1} \ \dots & H_0 \end{pmatrix}. \tag{4}$$

Note that, in each case, both systematic and nonsystematic encoders give the same codewords and thus share the same parity-check matrix [12, 16].

2.1.2. Parity-check matrix of turbo codes

For recursive systematic convolutional codes of rate $k/(k+1)$, that mainly compose classical turbo codes in the standards, the matrix $H^T(D)$ is simply given by [12]

$$H^T(D)=\begin{pmatrix} h_1^T(D) \\ h_2^T(D) \\ \vdots \\ h_{k+1}^T(D) \end{pmatrix}, \tag{5}$$

where in fact $h_i^T(D)$, $1\le i\le k$, are the feedforward polynomials and $h_{k+1}^T(D)$ is the feedback polynomial defining the recursive systematic convolutional code. Then, for this kind of components codes, the binary parity-check matrix can be simply derived using (2)–(4).

As recursive component codes of turbo codes are systematic, the columns of the associated parity-check matrix H with dimension $(N-K)\times N$ can be assigned to information bits and to redundancy bits. Note that when using the preceding expressions of H, the output bits of the convolutional encoder are supposed to be ordered alternatively within the codeword. After some column permutations, we can rewrite H as $\widetilde{H}=[H_iH_r]$, where H_i and H_r contain columns of H relative to information and redundancy bits, respectively. Using this notation, we can derive easily the parity-check matrix of a turbo code as follows for the case of two-component codes in parallel [17, 18]:

$$H_{tc}=\begin{bmatrix} H_i & H_r & 0 \\ H_i\Pi^T & 0 & H_r \end{bmatrix}, \tag{6}$$

where Π^T is the transpose of the interleaver permutation matrix at the input of the second component encoder. In that case, H_{tc} has dimensions $2(N-K)\times 2N-K$. Of course, this technique can be easily generalized to more than two components.

2.1.3. Example

To illustrate this section, we consider an $R=1/3$ turbo code with two rate one-half code components with parameters in octal given by $(1, 23_8/35_8)$. Under direct truncation, the parity-check matrix of a component code and a corresponding turbo code are, respectively, given by the matrices H and H_{tc} as follows:

$$H=\begin{pmatrix}
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&1&1&0&0&0&0&0&0&0&0&0&0\\
1&0&0&1&0&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&0&0&1&0&1&1&1&0&0&0&0&0&0\\
0&0&1&1&1&0&0&1&0&1&1&1&0&0&0&0\\
0&0&0&0&1&1&1&0&0&1&0&1&1&1&0&0\\
0&0&0&0&0&0&1&1&1&0&0&1&0&1&1&1
\end{pmatrix},$$

$$H_{tc}=\left(\begin{array}{c|c|c}
10000000 & 10000000 & 00000000\\
01000000 & 11000000 & 00000000\\
00100000 & 11100000 & 00000000\\
10010000 & 01110000 & 00000000\\
11001000 & 10111000 & 00000000\\
01100100 & 01011100 & 00000000\\
00110010 & 00101110 & 00000000\\
00011001 & 00010011 & 00000000\\
\hline
00100000 & 00000000 & 10000000\\
01000000 & 00000000 & 11000000\\
00010000 & 00000000 & 11100000\\
00101000 & 00000000 & 01110000\\
01100100 & 00000000 & 10111000\\
11010000 & 00000000 & 01011100\\
00011001 & 00000000 & 00101110\\
00001110 & 00000000 & 00010111
\end{array}\right). \tag{7}$$

2.2. Clustering and preprocessing

Once the parity-check matrix H of a turbo code has been derived, we obtain a nonbinary Tanner graph by applying a clustering technique, which is essentially the same as the one described in [11].

The matrix H is decomposed in groups of p rows and p columns. Each group of p rows represents a generalized parity-check node in the Tanner graph, defined in the finite group $G(2^p)$, and each group of columns represents a symbol node, build from the concatenation of p bits (p-tuples) defining elements in $G(2^p)$.

A cluster is then defined as a submatrix h_{ij} of size $p \times p$ in H, and each time a cluster contains nonzero values (ones in this case) in it an edge connecting the corresponding group of rows and group of columns is created in the Tanner graph. To each nonzero cluster is associated a linear function $f_{ij}(\cdot)$ from $G(2^p)$ to $G(2^p)$ which has h_{ij} as matrix representation. Using this notation, the ith generalized parity-check equation defined over $G(2^p)$ can be written as

$$\sum_j f_{ij}(c_j) \equiv 0, \tag{8}$$

where c_j is the jth coordinate of a codeword having symbols defined in $G(2^p)$.

To illustrate the clustering impact on the Tanner graph representation and to give some insights that can motivate to extend the representation from the binary domain to a nonbinary one, we will consider as a simple example the clustering of the recursive systematic convolutional codes with polynomial representation in octal basis given by $(1, 5_8/7_8)$. We assume that 12 information bits have been sent using direct truncation. Then, a 4×4 clustering is applied to the binary parity-check matrix. Using representation of (3), the resulting clustered matrix is given by

$$H = \left(\begin{array}{c|c|c|c|c|c}
1100 & 0000 & 0000 & 0000 & 0000 & 0000 \\
0111 & 0000 & 0000 & 0000 & 0000 & 0000 \\
1101 & 1100 & 0000 & 0000 & 0000 & 0000 \\
0011 & 0111 & 0000 & 0000 & 0000 & 0000 \\
\hline
0000 & 1101 & 1100 & 0000 & 0000 & 0000 \\
0000 & 0011 & 0111 & 0000 & 0000 & 0000 \\
0000 & 0000 & 1101 & 1100 & 0000 & 0000 \\
0000 & 0000 & 0011 & 0111 & 0000 & 0000 \\
\hline
0000 & 0000 & 0000 & 1101 & 1100 & 0000 \\
0000 & 0000 & 0000 & 0011 & 0111 & 0000 \\
0000 & 0000 & 0000 & 0000 & 1101 & 1100 \\
0000 & 0000 & 0000 & 0000 & 0011 & 0111
\end{array}\right). \tag{9}$$

We are now able to associate a nonbinary Tanner graph representation of H with generalized parity-check constraints applying now to 4-tuples binary vectors. The Tanner graph corresponding to our example is finally given in Figure 2(b) and it is compared with the Tanner graph associated with the binary image defined by H (Figure 2(a)).

Through this example, we can see that, for convolutional codes, when using the representation given in (3), we can still ensure a sparse graph condition and even reach a tree representation when increasing the order of the representation. In fact, for rate one-half codes, it has been observed that there exists a minimum value of P for which we can have a tree. This induces that using a BP-like decoder

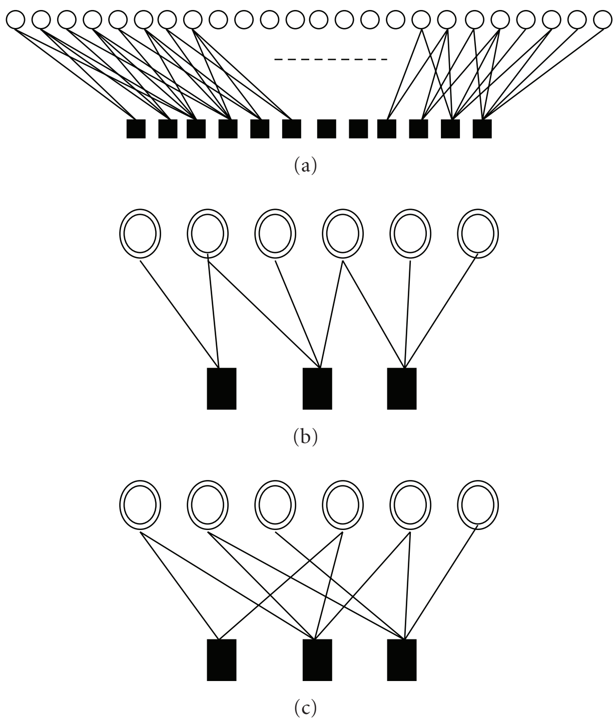


FIGURE 2: Comparison of different Tanner graph representations of the recursive systematic convolutional code with bit clustering of order $p = 4$ versus those of the binary image defined by H.

will lead to a maximum a posteriori symbol decoding and, in that case, it has been verified that BP and the MAP have the same performance. Unfortunately, this tree condition does not hold anymore when we use the alternative representation $\widetilde{H}$ of the parity-check matrix of a convolutional code as used in turbo codes parity-check matrix as it can be seen for the Tanner graph representation of our previous example in Figure 2(c). This representation introduces cycles even in the extended representation of the convolutional code using bit clustering, and as a result, in the extended representation of the turbo codes. Moreover, when tail biting is used, there is no possibility to ensure a tree condition due to the nonzero elements in the right-hand corner of the tail-bited, parity-check matrix of the component code. Thus, a remaining issue is how to derive a "good" extended Tanner graph representation. To this end, we will present in Section 3 how to overcome these problems to ensure fair performance under BP decoding by applying an efficient preprocessing of the parity-check matrix of the turbo code.

2.3. Nonbinary group belief propagation decoding

The Tanner graph obtained by preprocessing and clustering the binary image does not correspond to a usual code defined over a finite field $GF(q = 2^p)$ but can be defined on a finite group $G(2^p)$ of the same order (see [11] for more details). We will refer to the belief propagation decoder on group codes as *group BP* decoder. The group BP decoder is very similar in nature to regular BP in finite fields. The

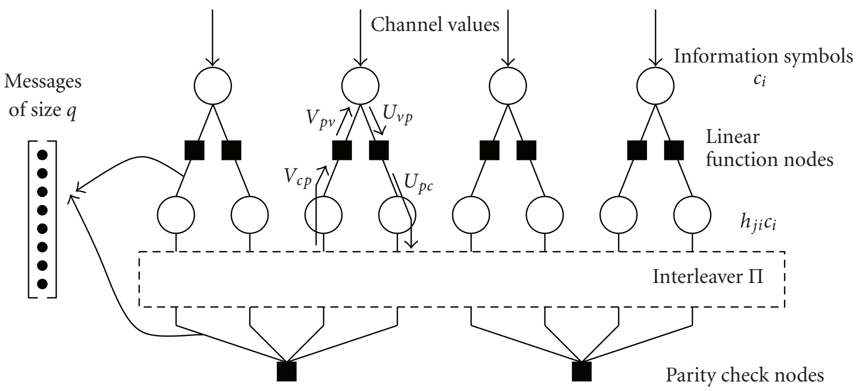


Figure 3: Tanner graph of a nonbinary LDPC code defined over a finite group $G(q)$.

only difference is that the nonzero values of a parity-check equation are replaced with more general linear functions from $G(2^p)$ to $G(2^p)$, defined by the binary matrices which form the clusters. In particular, it is shown in [11] that group BP can be implemented in the Fourier domain with a reasonable decoding complexity.

We briefly review the main steps of the group BP decoder and its application to the nonbinary Tanner Graph of a turbo code. The modified Tanner graph of an LDPC code over a finite group is depicted in Figure 3, in which we indicated the notations we use for the vector messages. Additionally, to the classical variable and check nodes, we add function nodes to represent the effect of the linear transformations deduced from the clusters as explained in the previous section.

The group BP decoder has four main steps which use $q = 2^p$ dimensional probability messages.

(i) *Data node update:* the output extrinsic message is obtained from the term by term product of all input messages including the channel-likelihood message, except the one carried on the same branch of the Tanner graph.

(ii) *Function node update:* the messages are updated through the function nodes $f_{ij}(\cdot)$. 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 $G(2^p)$ to $G(2^p)$ denoted $\beta = f_{ij}(\alpha)$ the update operation is

$$U_{pc}[\beta_j] = \sum_i U_{vp}[\alpha_i] \quad j = 0, \ldots, q-1,\ \beta_j = f_{ij}(\alpha_i). \tag{10}$$

(iii) *Check node update:* this step is identical to BP decoder over finite fields and can be efficiently implemented using a fast Fourier transform. See, for example, [11, 19] for more details.

(iv) *Inverse function node update:* with the use of the function $f_{ij}(\cdot)$ backwards, that is, by identifying the values α_i which have the same image β_j, the update equation is

$$V_{pv}[\alpha_i] = V_{cp}[\beta_j] \quad \forall \alpha_i : \beta_j = f_{ij}(\alpha_i). \tag{11}$$

These four steps define one decoding iteration of a general parity-check code on a finite group, which is the case of a clustered convolutional or turbo code as described previously. Note that the function node update is simply a reordering of the values both in the finite field case, and when the cluster defining the function $f_{ij}(\cdot)$ is full rank. When the cluster has deficient rank $r < p$, which is often the case when clustering a turbo code, only 2^r entries of the message U_{pc} are filled and the remaining entries are set to zero.

Note that we do not discuss in this paper the decoding complexity issues, but we rather focus on the feasibility of the decoding using a BP decoder. Of course, a nonbinary BP decoder is naturally much more computationally intensive than a binary BP or a turbo decoder. However, reduced complexity nonbinary decoders have been recently proposed, which exhibit good complexity/performance tradeoff even compared to binary decoders [20]. The reduced complexity decoder can be easily adapted to codes on finite groups, since the function node update is not more complex in the group case than in the field case.

3. COMPARISON OF BINARY IMAGES OBTAINED WITH DIFFERENT PREPROCESSINGS

In this Section, we discuss some relevant issues related to the improvement of the performance when group BP decoder is used. We show in particular that some preprocessing functions can lead to interesting Tanner graph topologies and good performance under iterative decoding.

3.1. Selection of preprocessing for an efficient sparse graph representation

It should be noted that the performance of the group BP decoder depends highly on the structure of the nonbinary Tanner graph. In our framework, it is possible to apply some specific transformations on the binary image H before the clustering operation, so that the Tanner graph has desirable properties. Indeed, any row linear transformation A and column permutation Π applied to H do not change the code space but change the topology of the clustered Tanner graph. Let us denote by $H' = \mathcal{P}_c(H) = A \cdot H \cdot \Pi$ the preprocessed binary parity-check matrix. We propose in this paper two preprocessing techniques that we found attractive in terms of Tanner graph properties and described below and depicted in Figure 4.

($\mathcal{P}_{c_1}$)

This preprocessing is defined by alternating the information bits and the redundancy bits of the first convolutional code of the parallel turbo code. We obtain with this technique two parts in the parity-check matrix. Each of them has an upper triangular form with a diagonal (or near diagonal for the rectangular part of H'), therefore, reducing the number of nonzero clusters in the nonbinary Tanner graph deduced from H'. Note that a second preprocessing of this type can be considered by alternating the information bits and the redundancy bits of the second convolutional encoder.

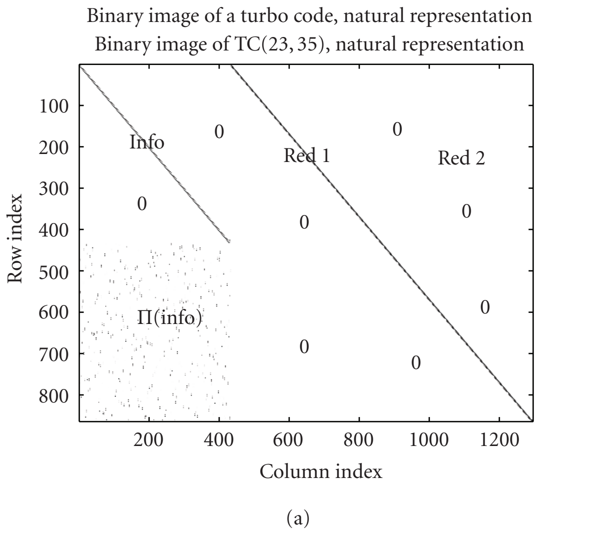
($\mathcal{P}_{c_2}$)

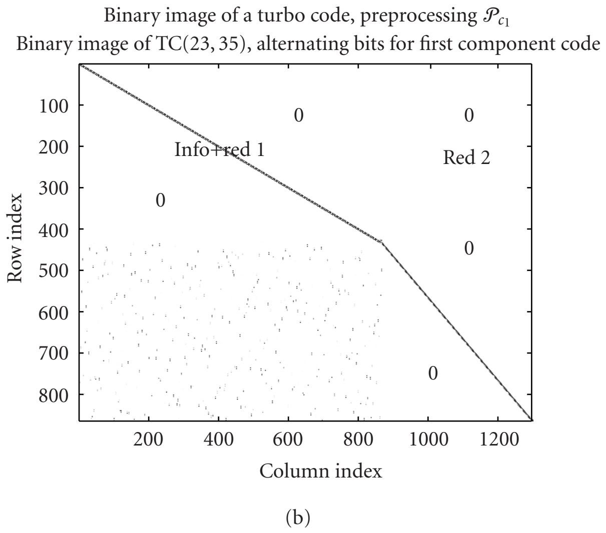
This preprocessing is obtained by column permutations with the aim of having the most concentrated diagonal in the parity-check matrix, that is, minimizing the number of clusters that will be created on the diagonal. This is supposed to be a good choice since the clusters on the diagonal are the more dense in the Tanner graph and are assumed to participate the most to the performance degradation of the BP decoder when they contribute to cycles. Indeed, we have verified by simulations on several turbo codes that the number of nonzero clusters of a given size has less in the preprocessing $\mathcal{P}_{c_2}$ than in the preprocessing $\mathcal{P}_{c_1}$ on the diagonal. Note that by properly choosing the columns to be permuted, several images of this type could be created.

Note that the two proposed preprocessing techniques are restricted to column permutations, that is, with the special case of $A = \mathrm{Id}$, where Id corresponds to the identity transformation. This case is the simplest one; the transformation keeps the binary Tanner graph of the code unchanged, but the nonbinary clustered Tanner graph is modified after preprocessing. We will show through simulations that this has an important impact on the decoder performance. Although Figure 4 plots examples of rate $R = 1/3$ turbo codes, the exact same preprocessing strategies can be applied to any type of turbo code, that is, any rate for punctured and/or multibinary turbo codes.

3.2. Simulation results with different preprocessings

In this section, we apply the different preprocessing techniques presented in the previous section to duobinary turbo codes with the parameters $R = 0.5$ and size $N = \{848, 3008\}$ coded bits taken from the DVB-RCS standard [21, 22]. The frame sizes we used correspond to ATM and MPEG frame sizes with $K = \{53, 188\}$ information bytes, respectively. Note that these turbo codes have sizes which are not particularly well suited to clustering. A size of $N = 864$ would have been preferable for cluster size $p = 8$ to ensure a proper clustering of *each part* of the turbo code parity-check matrix corresponding to each component codes, but we wanted to





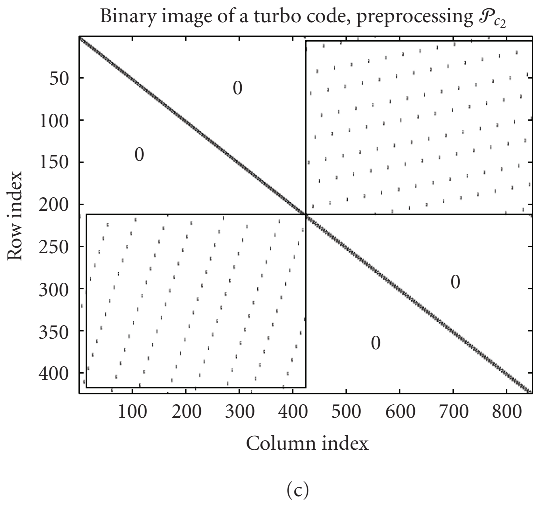


FIGURE 4: Three different binary representations of the same rate $R = 1/3$ turbo code. The first one is the natural representation (see (6)), the second one corresponds to the clustering $\mathcal{P}_{c_1}$, and the third one to the clustering $\mathcal{P}_{c_2}$.

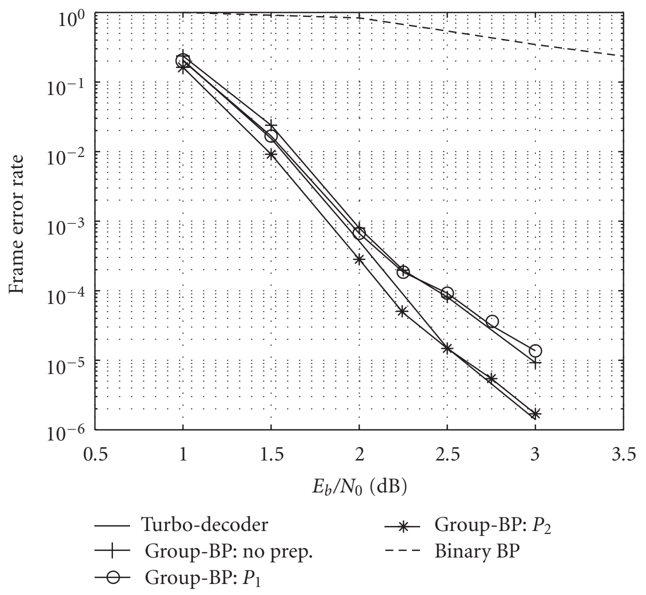


Figure 5: Performance of the group BP decoding algorithm based on different preprocessing functions for the ($R = 0.5$, $N = 848$) duobinary turbo code and comparison with the turbo decoder as used in the DVB standard.

keep the original interleaver and frame size from the standard [21, 22]. These codes have been terminated using tail biting, and their minimum distances are $d_{\min} = \{18, 19\}$. For both duobinary turbo codes, $H^T(D)$ is given by

$$H^T(D) = \begin{pmatrix} 1 + D^2 + D^3 \\ 1 + D + D^2 + D^3 \\ 1 + D + D^3 \end{pmatrix}. \tag{12}$$

In the following, we will consider the additive white Gaussian noise channel (AWGN) for our simulations. For this channel, we compare the group decoder performance with various preprocessing functions, a clustering size of $p = 8$, and a floating point implementation of the group BP decoder using shuffle scheduling [23]. As a reference, we simulated the turbo decoder based on MAP component decoders in floating-point precision in order to have the best results that one can obtain with a turbo decoding strategy.

The curves plotted in Figure 5 are related to the $R = 1/2$ turbo code with parameter $N = 848$ and correspond to the natural representation of the code and two preprocessings (one type $\mathcal{P}_{c_1}$ and one type $\mathcal{P}_{c_2}$).

In order to illustrate that the preprocessing has influence on the nonbinary factor graph, we have counted the number of nonzero clusters and also the number of full-rank clusters in the cases of the two simulated matrices tested in this section, and for the two types of preprocessings $\mathcal{P}_{c_1}$ and $\mathcal{P}_{c_2}$. We reported the statistics on Table 1. Remember that a nonzero cluster corresponds to an edge in the Tanner graph, and that a full-rank cluster corresponds to a permutation function, while a rank deficient cluster corresponds to a projection. We can see that the number of nonzero clusters is much lower in the case of the proposed preprocessing,

Table 1: Cluster Statistics on the turbo codes from the DVB standard, with a clustering size of $P = 8$.

	$\mathcal{P}$	Total clusters	Nonzero clusters	Full-rank clusters
Turbo $R = 1/2$	$\mathcal{P}_{c_1}$	5618	506	26
$N = 848$ bits	$\mathcal{P}_{c_2}$	5618	426	0
Turbo $R = 1/2$	$\mathcal{P}_{c_1}$	70688	1786	94
$N = 3008$ bits	$\mathcal{P}_{c_2}$	70688	1504	0

but also that there is no full-rank clusters. This indicates that the preprocessing $\mathcal{P}_{c_2}$ has concentrated the ones of the parity-check matrix H_b in a better way than $\mathcal{P}_{c_1}$. Our simulation results show that this better concentration has a direct influence on the error-correction capability of the group BP decoder.

All group BP simulations have used a maximum of 100 iterations, but the average number of iteration is as low as 3-4 iterations for frame error rates below 10^{-3}. Simulations were run until at least 100 frames have been found in error. As expected, the preprocessing of type $\mathcal{P}_{c_2}$ is far better than the other preprocessings, which is explained by the fact that the corresponding Tanner graph has less nonzero clusters. It can be seen that with a good preprocessing function, a turbo code can be efficiently decoded using a BP decoder, and even can slightly beat the turbo decoder in the waterfall region. The turbo decoder remains better in the error floor region, which is due to the fact that the group BP decoder has much more detected errors (due to decoder failures) in this region than the turbo decoder. Although we are aware that the group BP decoder is much more complex than the turbo decoder, this result is quite encouraging since it was long thought that turbo codes could not be decoded using an LDPC-like decoder. As a drastic example, we have plotted the very poor performance of a binary BP decoder on the binary image of the turbo code, which does not converge at all for all the SNRs under consideration.

We also simulated the same curves for a longer code with $N = 3008$ in order to show the robustness of our approach. The results are shown in Figure 6, and the same comments as for the $N = 848$ code apply with an even larger performance gain when using the best preprocessing function.

4. IMPROVING PERFORMANCE BY CAPITALIZING ON THE PREPROCESSING DIVERSITY

As there exist more than one possibility to build a nonbinary Tanner graph from the same code through different preprocessing functions, this raises the question whether if it is possible to improve the decoding performance by using this diversity of graph representation. Actually, we have noticed that with the same noise realization, the group BP decoder on a specific Tanner graph can either (i) converge to the right codeword,or (ii) converge to a wrong codeword (undetected error),or (iii) diverge after a fixed maximum number of iterations. If we accept some additional complexity, using several instances of iterative decoding based on several preprocessing functions and a

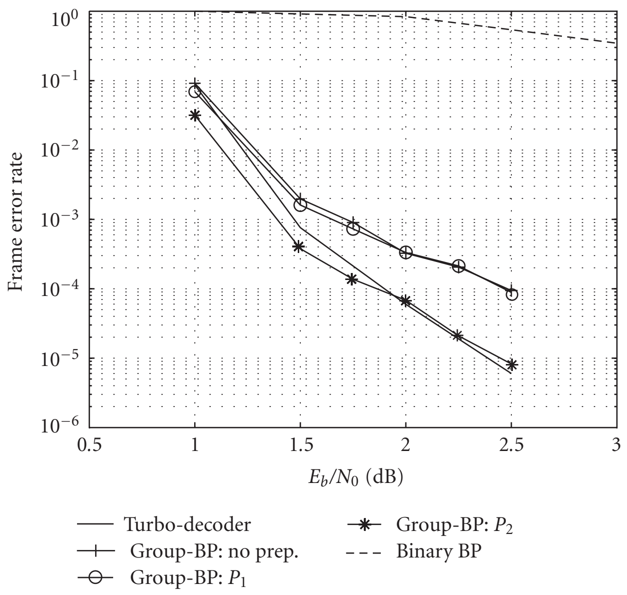


FIGURE 6: Performance of the group BP decoding algorithm based on different preprocessing functions for the ($R = 0.5$, $N = 3008$) duobinary turbo code and comparison with the turbo decoder as used in the DVB standard.

proper results merging strategy is likely to improve the error-correction performance.

In this paper, we will not address the problem of finding a good set of preprocessing functions, and we restrict ourselves to $N_d = 5$ different Tanner graphs obtained with preprocessing functions of type $\mathcal{P}_{c_2}$. There are various possible merging methods 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 N_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 the right codeword or to a wrong codeword after a given number of iterations, we switch to another decoder, that is, another Tanner graph is computed with a different preprocessing 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 not).

Parallel merging

The N_d decoders are used in parallel and a maximum likelihood (ML) decision is taken among the ones that have converged to a codeword. If nb, with $nb \leq N_d$, is the number of decoders that have converged to a codeword in less than the maximum number of iterations, then the nb associated likelihood is computed and the one with the maximum likelihood is selected. Note that the nb candidate codewords are not necessarily distincts.

Lower bound on merging strategies

In order to study the potential of the decoder diversity approach regardeless of the merging strategy, we define the following lower bound. Among the N_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 N_d decoders have not converged 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 not the ML codeword. This lower bound allows also to have a possibly tight estimation on the parallel merging case, without having to simulate all N_d decoders.

The extra complexity induced by the serial merging is negligible since the other Tanner graphs will be used only when the first one fails to converge, that is, at an 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 N_d times more computations, but one can argue that it is simpler to parallelize on a chip. We did not simulate the parallel merging in this work. In the worst case, the extra latency of the serial merging is obviously linearly dependent on the number N_d of different Tanner graphs.

In Figures 7 and 8, we report simulation results for the AWGN channel for the two turbo codes that have been studied in the previous section. Of course, the results with no diversity are similar to those observed in Figures 5 and 6 for the preprocessing of type $\mathcal{P}_{c_2}$, and we do not plot them in the new figures. 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 and the error floor regions. For the small code as well as for the longer code, using group BP decoding with decoder diversity can gain between 0.25 dB to 0.4 dB compared to the turbo decoder using MAP 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.

Interestingly, the serial merging which is the more obvious merging strategy, and also requires the least additional complexity, achieves full decoder diversity gain in the waterfall region, that is, 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 both Figures 7 and 8 that more elaborate merging solutions should be used to achieve full diversity gain and obtain a substantial gain compared with turbo decoder. Note, however, that with the serial merging

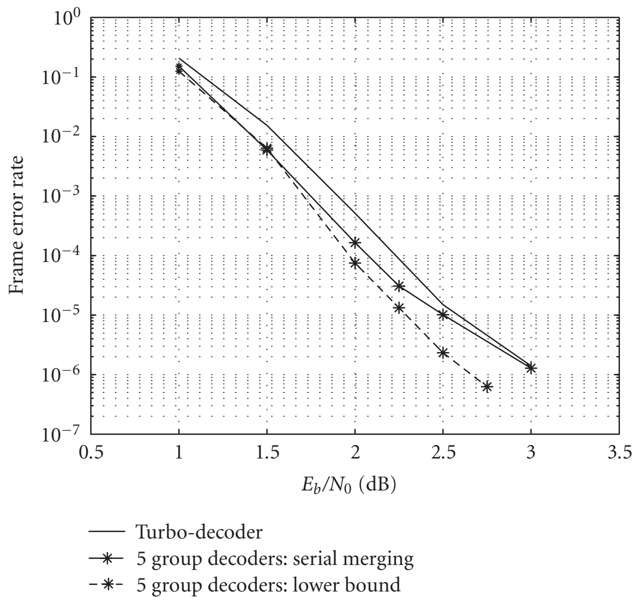


Figure 7: Decoding performance when diversity is applied to the ($R = 0.5$, $N = 848$) duobinary turbo code.

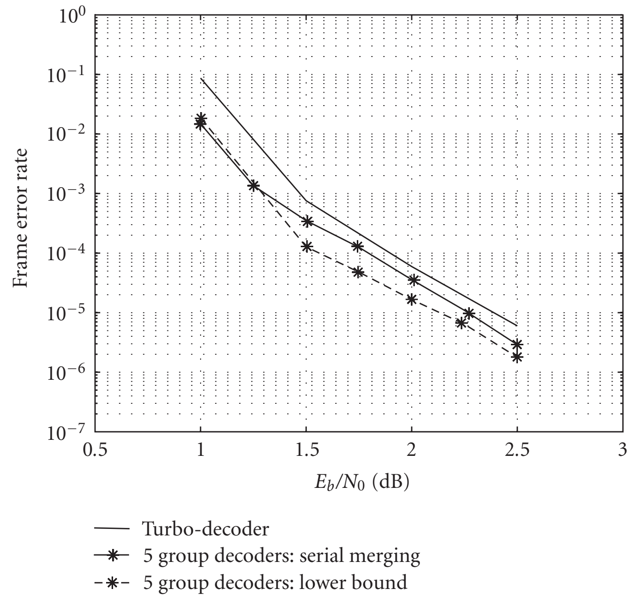


Figure 8: Decoding performance when diversity is applied to the ($R = 0.5$, $N = 3008$) duobinary turbo code.

and for the $N = 3008$ turbo code the results are better than the turbo decoder for all SNRs, even in the error floor region.

5. CONCLUSION

In this paper, we have proposed a new approach to efficiently decode turbo codes using a nonbinary belief propagation decoder. It has been shown that this generic method is fully efficient if a preprocessing step on the parity-check matrix of the code is added to the decoding process in order to ensure good topological properties of the Tanner graph and then good iterative decoding performance. Using this extended representation, we show that the proposed algorithm exhibits very good performance in both the waterfall and the error regions when compared to a classical turbo decoder. Moreover, using the inherent diversity induced by the existence of several concurrent extended Tanner graph representations, we show that the performance can be further improved and we introduce the concept of decoder diversity. This study shows that this decoding strategy (i.e., joint use of preprocessing, group BP and diversity decoding) appears as a key step that enables to consider LDPC and turbo codes within a unified framework from the decoder point of view.

REFERENCES

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[2] R. G. Gallager, *Low Density Parity Check Codes*, Number 21 in Research Monograph SeriesNumber 21 in Research Monograph Series, MIT Press, Cambridge, Mass, USA, 1963.

[3] IEEE 802.16-2004, “IEEE standard for local and metropolitan area networks, air interface for fixed broadband wireless access systems,” October 2004.

[4] IEEE 802.16e, February 2006, IEEE standard for local and metropolitan area networks, air interface for fixed broadband wireless access systems, amendment 2: physical and medium access control layers for combined fixed and mobile operation in licensed bands and corrigendum 1.

[5] F. R. Kschischang and B. J. Frey, “Iterative decoding of compound codes by probability propagation in graphical models,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 219–230, 1998.

[6] R. J. McEliece, D. J. C. 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, no. 2, pp. 140–152, 1998.

[7] L. Zhu, J. Wang, and S. Yang, “Factor graphs based iterative decoding of turbo codes,” in *Proceedings of the IEEE International Conference on Communications, Circuits and Systems and West Sino Expositions (ICCCAS & WeSino Expo ’02)*, vol. 1, pp. 46–50, Chengdu, China, June-July 2002.

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

[9] K. Engdahl and K. Sh. Zigangirov, “On the statistical theory of turbo codes,” in *Proceedings of the 6th International Workshop on Algebraic and Combinatorial Coding Theory (ACCT ’98)*, pp. 108–111, Pskov, Russia, September 1998.

[10] A. J. Felström and K. Sh. Zigangirov, “Time-varying periodic convolutional codes with low-density parity-check matrix,” *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2181–2191, 1999.

[11] A. Goupil, M. Colas, G. Gelle, and D. Declercq, “FFT-based BP decoding of general LDPC codes over Abelian groups,” *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 644–649, 2007.

[12] R. Johannesson and K. Sh. Zigangirov, *Fundamentals of Convolutional Coding*, Digital, Mobile CommunicationDigital, Mobile Communication, chapter 1-2, IEEE Press, New York, NY, USA, 1999.

[13] P. Stahl, J. B. Anderson, and R. Johannesson, "A note on tail-biting codes and their feedback encoders," *IEEE Transactions on Information Theory*, vol. 48, no. 2, pp. 529–534, 2002.

[14] C. Weiss, C. Bettstetter, and S. Riedel, "Code construction and decoding of parallel concatenated tail-biting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 1, pp. 366–386, 2001.

[15] H. Ma and J. Wolf, "Binary unequal error-protection block codes formed from convolutional codes by generalized tail-biting," *IEEE Transactions on Information Theory*, vol. 32, no. 6, pp. 776–786, 1986.

[16] S. Lin and D. J. Costello, *Error Control Coding*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2nd edition, 2004.

[17] O. Pothier, *Compound codes based on graphs and their iterative decoding*, Ph.D. thesis, ENST, Paris, France, January 2000.

[18] R. E. Blahut, *Algebraic Codes for Data Transmission*, Cambridge University Press, Cambridge, UK, 2003.

[19] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633–643, 2007.

[20] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low complexity, low memory EMS algorithm for non-binary LDPC codes," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 671–676, Glasgow, UK, June 2007.

[21] C. Douillard and C. Berrou, "Turbo codes with rate-$m/(m+1)$ constituent convolutional codes," *IEEE Transactions on Communications*, vol. 53, no. 10, pp. 1630–1638, 2005.

[22] Digital Video Broadcasting (DVB), "Interaction channel for satellite distribution systems," 2000, ETSI EN 301 790, v 1.2.2.

[23] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209–213, 2005.