Decoding Algorithms for Nonbinary LDPC Codes over $GF(q)$

David Declercq and Marc Fossorier

September 19, 2006

Abstract

In this paper, we address the problem of decoding nonbinary LDPC codes over finite fields GF(q), with reasonnable complexity and good performance. In a first part of the paper, we recall the original Belief Propagation (BP) decoding algorithms and its Fourier domain implementation. We show that the use of tensor notations for the messages is very convenient for the algorihtm description and understanding. In the second part of the paper, we introduce a simplified decoder which is inspired from the min-sum decoder for binary LDPC codes. We called this decoder Extended Min-Sum (EMS). We show that it is possible to greatly reduce the computationnal complexity of the check node processing by computing approximate reliability measures with a limited number of values in a message. By choosing appropriate correction factors or offsets, we show that the EMS decoder performance are quite good, and in some cases better than the regular BP decoder. The optimal values of the factor and offset correction are obtained asymptotically with simulated density evolution. Our simulations on ultra-sparse codes over very high order fields show that non-binary LDPC codes are promising for applications which require low FER for small or moderate codeword lengths. The EMS decoder is a good candidate for practical hardware implementations of such codes.

1 Introduction

Low density parity check (LDPC) codes designed over GF(q) (also referred to as GF(q)-LDPC codes) have been shown to approach Shannon limit performance for $q = 2$ and very long code lengths [1, 2, 4, 5]. On the other hand, for moderate code lengths, the error performance can be improved by increasing q [6, 7]. However this improvement is achieved at the expense of increased decoding complexity.

A straightforward implementation of the belief propagation (BP) algorithm to decode GF(q)-LDPC codes has computational complexity dominated by $O(q^2)$ operations for each check sum processing [6]. As a result, no field of order larger than $q = 16$ was initially considered. Extending the ideas presented in [3], a more efficient approach using Fourier transforms over GF(2^p) was presented in [8]. The description of this algorithm in the log-domain has been given in [9]. Note that the Fourier transform is easy to compute only when the Galois field is a binary extension field with order $q = 2^p$. In that case, this approach allows to reduce the computational complexity of the BP algorithm to $O(p2^p)$. Consequently, results for $2^p = 256$ were reported with this method [10, 7]. In [10], the formulation of this algorithm was further elegantly and conveniently modified based on the introduction of a tensoral representation. With this representation, the generalization of BP decoding over GF(2) to any field of order $q = 2^p$ becomes very natural. We present in details the BP algorithm using tensoral notations in the first part of this paper.

Simplified iterative decodings of GF(q)-LDPC codes have also been investigated. For $q = 2$, the min-sum (MS) algorithm with proper modification has been shown to result in negligible performance degradation (less than 0.1 dB for regular LDPC codes) while performing additions only, and becoming independent of

the channel conditions [11]. Extension of this approach to any value q seems highly attractive. Unfortunately, such extensions are not straightforward as many simplifications can not be realized in conjunction with Fourier transforms. In [9], the authors present a log-domain BP decoder combined with a FFT at the check node input. However combining log-values and FFT requires a lot of exponential and logarithm computations, which may not be very practical. To overcome this issue, the authors propose the use of a look-up table (LUT) to perform the required operations. Although simple, this approach is of limited interest for codes over high order fields since the number of LUT accesses grows in $q \log_2(q)$ for a single message. As a result, for fields of high order, unless the LUT has a prohibitively large size, the performance loss induced by the LUT approximation is quite large. In [9], the authors present simulation results for LDPC codes over fields up to GF(16), in which case the LUT approach remains manageable. In [12], the MS algorithm is extended to any finite field of order q. Although only additions are performed and no channel information is necessary, its complexity remains $O(q^2)$. As a result, only small values of q can be considered by this algorithm and for $q = 8$, a degradation of 0.5 dB over BP decoding is reported. Simplifications of BP decoding of GF(q)-LDPC codes have also been considered in [13] for non binary signaling.

In this paper, we develop a generalization of the MS algorithm which not only performs additions without the need of channel estimation, but also with the two following objectives: *(i)* a complexity much lower than $O(q^2)$ so that finite fields of large order can be considered; and *(ii)* a small performance degradation compared with BP decoding. The first objective is achieved by introducing configuration sets, which allow to keep only a small number of meaningful values at the check node processing (note that while the check node processing is $O(q^2)$, that of the variable node processing is only $O(q)$ [12]). The second objective is achieved by applying at the variable node processing the correction techniques of [11] to the proposed algorithm.

The paper is organized as follows. In Section 2, the tensoral representation of the messages for GF(2^p)-LDPC codes [10] is further developed and we discuss the importance of such notation for the simplification of the decoding algorithm in the Fourier domain. The extended MS algorithms in the logarithm domain are derived in Section 3. We present the basic proposed algorithm (EMS), and its improved corrected versions using a scaling factor or an offset. The correction values are obtained by minimizing the decoding thresholds computed with density evolution techniques. Simulation results for small and moderate block lengths are discussed in Section 4 and concluding remarks are given in Section 5.

2 Decoding GF(2^p)-LDPC codes with Belief Propagation

In this section, we present the BP decoding algorithm of non-binary GF(2^p)-LDPC codes. This decoding algorithm has been already presented in the literature [6, 9, 10], but the presentation proposed in this paper is original and could be useful to a deeper understanding and analysis of GF(2^p)-LDPC decoders. The key points of a clear presentation of GF(2^p)-LDPC BP are *(i)* the use of a tensoral representation of the messages along the edges in the factor graph representation of the code, and *(ii)* transformations of the usual LDPC factor graph so that the BP equations can be written in a simple way.

2.1 Tensorial representation of the messages

In a GF(q)-LDPC code, the parity check matrix H of size $M \times N$ defining the kernel of the code $\mathcal{C}_H$ is a sparse matrix, such that:

$$\mathcal{C}_H = \left\{ \underline{c} \in GF(q)^N \ \mid \ H.\underline{c} \stackrel{\text{GF}}{=} \underline{0} \right\} \tag{1}$$

As in the binary case, the rank of H is $N-K$ with $K \geq N-M$ and the code rate is $R = K/N \geq 1-M/N$.

The factor graph representation of the code [15, 14] consists of a set of variable nodes belonging to GF(q) fully connected to a set of parity check nodes. The edges connecting the two sets of nodes carry messages that represent probability density functions (pdf's) of the codeword symbols. Since the codeword symbols are random variables over GF(q), the messages are discrete pdf's of size q. When the Galois field is an extension field of GF(2), that is when $q = 2^p$, the messages can be conveniently represented by tensors of size 2 and dimension p [10]. Let us briefly describe this tensoral structure of the messages.
In the Galois field GF(2^p), the elements of the field can be represented by a polynomial $i(x) = \sum_{l=1}^{p} i_l\, x^{l-1}$ of degree $p-1$ with binary coefficients. Any set of p binary values $\{i_l\}_{l=1..p}$ thereby determines a unique field element $i(x)$. In the following, we use these polynomial notations to indicate when the operations (sums, products, etc) are performed in a Galois field.
Using this representation, a message on the edge connected to a variable node denoted $i(x)$ is a tensor $\{U[i_1..i_p]\}_{i_1..i_p}$ indexed by the binary coefficients of $i(x)$. For example, $U[0,1,1]$ corresponds to the probability $p(i(x) = x + x^2|.)$ in GF(8), conditionnaly to the random variables which depend on $i(x)$ in the factor graph.

2.2 Belief Propagation decoding over GF(2^p)

In order to simplify the notations used in the derivation of the equations, we omit temporal indices to describe the BP algorithm. The BP algorithm for non-binary LDPC codes is not a direct generalization of the binary case because the nonzero values of the matrix H are not binary. There is however a way to link tightly the non-binary case to the binary case by modifying the factor graph. Let us take the example of a single row (check sum) with check degree d_c. A parity node in a LDPC code over GF(2^p) represents the following parity equation:

$$\sum_{k=1}^{d_c} h_k(x) i_k(x) = 0 \quad \mathrm{mod} p(x) \tag{2}$$

where $p(x)$ in the modulo operator is a degree $p-1$ primitive polynomial of GF(q). Equation (2) expresses that the variable nodes needed to perform the BP algorithm on a parity node are not the codeword symbols alone, but the codeword symbols multiplied by nonzeros values of the parity matrix H. The corresponding transformation of the graph is performed by adding variable nodes corresponding to the multiplication of the codeword symbols $i_k(x)$ by their associated nonzero H-values. The transformation is depicted in Figure 1.
The function node that connects the two variable nodes $i_k(x)$ and $h_k(x)i_k(x)$ performs a permutation of the message values. The permutation that is used to update the message corresponds to the multiplication of the tensor indices by $h_k(x)$ from node $i_k(x)$ to node $h_k(x)i_k(x)$ and to the division of the indices by $h_k(x)$ the other way. Using this transformation of the factor graph, the parity node update is indeed a convolution of all incoming messages, just as in the binary case.
We use the following notations for the messages in the graph (see Figure 1). Let $\{V_{pv}\}_{v=1..d_v}$ be the set of messages entering a variable node of degree d_v, and $\{U_{vp}\}_{v=1..d_v}$ be the output messages for this variable node. The index 'pv' indicates that the message comes from a *permutation* node to a *variable node*, and 'vp' is for the other direction. We define similarly the messages $\{U_{pc}\}_{c=1..d_c}$ (resp. $\{V_{cp}\}_{c=1..d_c}$) at the input (resp. output) of a degree d_c check node.
The initialization of the decoder is achieved with the channel likelihoods denoted $L[i_1..i_p]$. If the channel has binary input and Gaussian additive noise for example, then, the symbol likelihood has the form $L[i_1..i_p] = \prod_{l=1}^{p} l(i_l)$, where $l(i_l) = Prob(y_l|b_l = i_l)$ with b_k being the $l-th$ bit of the GF(q) symbol, and y_l is a Gaussian noisy version of b_l.
The three steps of one decoding iteration are then:

$(\boldsymbol{\mathcal{S}_1})$: product step, variable node update for a degree d_v node:

$$\begin{aligned} U_{tp} &= L \times. \prod_{v=1, v\neq t}^{d_v} . V_{pv} \qquad\qquad t = 1 \dots d_v \\ \text{or } U_{tp}[i_1..i_p] &= L[i_1..i_p] \prod_{v=1, v\neq t}^{d_v} V_{pv}[i_1..i_p] \qquad\qquad (i_1..i_p) \in \{0,1\}^p, t = 1 \dots d_v \end{aligned} \tag{3}$$

■

where $\times.$ is defined as the term by term product of tensors, and L is the tensor of likelihoods computed from the channel output. Note that since the messages are probability density functions, one needs to normalize the messages after the product in (3) so that $\sum_{i_1..i_p} U_{tp}[i_1..i_p] = 1$.

$(\boldsymbol{\mathcal{S}_2})$: permutation step (from variable to check nodes):

$$\begin{aligned} \text{vec}\,(U_{pc}) &= P_{h(x)}\ \text{vec}\,(U_{vp}) \\ \text{or } U_{pc}[i_1..i_p] &= U_{vp}[j_1..j_p] \qquad\qquad (i_1..i_p) \in \{0,1\}^p \text{ with } i(x) = h(x)j(x) \end{aligned} \tag{4}$$

■

where $P_{h(x)}$ is a $(q \times q)$ permutation matrix corresponding to $h(x)$ and $\text{vec}\,(U)$ collects in a column vector all the values of the tensor U. Note that since finite Galois fields are cyclic fields, the permutation $P_{h(x)}$ is actually a cyclic shift (rotation) of all the values in the message, except the value labeled by 0. As for the permutation step in the other direction, that is from check node to variable node, it is performed with the inverse permutation $P_{h(x)}^{-1}$.

$(\boldsymbol{\mathcal{S}_3})$: sum-product step: parity check node update for a degree d_c node:
Using the secondary nodes $h(x)i(x)$, all the parity nodes have the same behavior, and no longer depend on the non-zeros entries of H. The BP update for a parity node of degree d_c is effectively a convolution of probability densities on GF(2^p):

$$\begin{aligned} V_{tp} &= \circledast_{c=1, c\neq t}^{d_c} U_{pc} \qquad\qquad t = 1 \dots d_c \\ \text{or } V_{tp}[i_{t_1}..i_{t_p}] &= \sum_{\{i_c(x)\}_{c\neq t}} \prod_{c=1, c\neq t}^{d_c} U_{pc}[i_{c_1}..i_{c_p}]\ \mathbb{1}_{\sum_{c=1}^{d_c} i_c(x)=0} \qquad\qquad t = 1 \dots d_c \end{aligned} \tag{5}$$

where $\mathbb{1}_S$ is the indicator function equal to 1 if and only if condition S is fullfiled. ■

We could also express the sum in (5) without the help of an indicator function, by using a *configurations set*. We will see in the presentation of the EMS algorithm (Section 3) that the concept of *configurations set* is very useful. Let us consider the following set:

$$\text{Conf}_{i_t(x)} = \left\{ \{i_c(x)\}_{c\neq t} \ : \ \sum_{c=1}^{d_c} i_c(x) = 0 \right\} \tag{6}$$

Using (6), step $(\mathcal{S}_3)$ becomes:

$(\boldsymbol{\mathcal{S}_3})$: sum-product step: parity check node update for a degree d_c node:

$$V_{tp}[i_{t_1}..i_{t_p}] = \sum_{\{i_c(x)\}\in \text{Conf}_{i_t(x)}} \prod_{c=1, c\neq t}^{d_c} U_{pc}[i_{c_1}..i_{c_p}] \tag{7}$$

■

2.3 Fast Fourier Transform based Belief Propagation decoding

In the BP algorithm presented in the previous section, the computational complexity of step $(\mathcal{S}_3)$ can rapidly become prohibitively large. The number of basic operations required to compute V_{tp} varies exponentially with both the field order q and the parity check degree d_c. This complexity can be reduced to $O(d_c\, q^2)$ by performing (5) recursively [6], but this remains too complex to allow the decoding of LDPC codes in very high order fields, and in many works the maximum field order is often restricted to $q = 16$ with this approach.
It has been proposed in [10] to perform step $(\mathcal{S}_3)$ in the frequency domain, which transforms the convolution into a simple product. The computation of step $(\mathcal{S}_3)$ in the frequency domain has also been pointed out in [8, 9].
For densities of random variables in GF(2^p) represented by tensors, the Fourier transform is formally very simple to explain since it corresponds actually to p second order Fourier transforms, one in each dimension of the tensor.

Proposition

Let $U_{i_1..i_p}$ be a tensor of order 2 and dimension p, representing a density function of the random variable $i(x) \in GF(2^p)$. The Fourier transform of $U_{i_1..i_p}$ is given by:

$$W = \mathcal{F}(U) = U \times_1 F \times_2 F \ldots \times_p F \tag{8}$$

where "$\times_k$" represents the outer tensoral product in the k-th dimension of the tensor, and F is the 2×2 matrix of second order Fourier transform given by:

$$F = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

■

As pointed out in [17], the Fourier transform $\mathcal{F}(.)$ with the correct symbol-bit labeling reduces to the Hadamard transform. For more details about Fourier transforms on finite sets (groups, rings or fields), one can refer e.g. to [25] and the references within.
The outer tensoral product $Z = U \times_k F$ is defined as, for $(i_1..i_{k-1}, i_{k+1}..i_p) \in \{0,1\}^{p-1}$:

$$Z[i_1..i_{k-1}, 0, i_{k+1}..i_p] = \frac{1}{\sqrt{2}}(U[i_1..i_{k-1}, 0, i_{k+1}..i_p] + U[i_1..i_{k-1}, 1, i_{k+1}..i_p])$$
$$Z[i_1..i_{k-1}, 1, i_{k+1}..i_p] = \frac{1}{\sqrt{2}}(U[i_1..i_{k-1}, 0, i_{k+1}..i_p] - U[i_1..i_{k-1}, 1, i_{k+1}..i_p])$$

Using the Fourier transform (8), one can change the parity check node into a product node in the factor graph of the GF(2^p)-LDPC code. The whole factor graph in the case of a regular $(d_v, d_c) = (2, 4)$ is depicted in Figure 2. This leads to a modified version of step $(\mathcal{S}_3)$:

$(\boldsymbol{\mathcal{S}_3})_{\boldsymbol{FFT}}$: frequency domain message update for a degree d_c check node:

$$V_{tp} = \mathcal{F}\left(\prod_{c=1, c\neq t}^{d_c} . \mathcal{F}(U_{pc})\right) \qquad t = 1 \ldots d_c \tag{9}$$

■

The computational complexity of step $(\mathcal{S}_3)_{FFT}$ is reduced to $O(d_c\, pq)$. Moreover, we can remark that the special form of the Fourier transform in GF(2^p) only involves additions and no multiplication, which increases even more the decoding speed. Note that the use of tensoral notations has another advantage since the Fourier transform (8) is already expressed in its fast implementation form (FFT). This can be seen in the definition of the outer tensoral product which represents exactly the equations used in a *butterfly* implementation of the Fourier transform. This reduced complexity BP decoding allows to decode nonbinary LDPC codes in very high order fields and for large values of d_c, or equivalently high code rates since $R \geq 1 - d_v/d_c$.

3 Extended Min-Sum Algorithm in the Logarithm Domain (EMS)

In this section, we present a reduced complexity decoding algorithm for LDPC codes over GF(q), based on a generalization of the MS algorithm used for binary LDPC codes [11, 14, 16, 18, 26]. It is now well known that reduced complexity decoding algorithms are built in the logarithm domain, and this is for two main reasons. First, it transforms the products in (3)-(5) into simple sums and the normalization of the messages in (3) is no longer required. The second reason is that log-domain algorithms are usually more robust to the quantization effects when the messages are stored on a small number of bits [19, 24]. The practical decoding algorithms for turbo-codes (max-log-map, max*-log-map) and LDPC codes (MS, logBP) are all expressed in the logarithm domain.
The simplified decoding algorithm that we propose also uses log-density-ratio (LDR) representations of the messages. In the binary case, only two probabilities define a message, and the LDR is defined as $LDR(u) = \log(\text{prob}(u=0)/\text{prob}(u=1))$. In the q-ary case, the message is actually composed of $q-1$ nonzero LDR values. By convention, we consider the following LDR messages, for any tensorial message U:

$$\overline{U}[i_1..i_p] = \log\left(\frac{U[i_1..i_p]}{U[0..0]}\right) \qquad \forall (i_1..i_p) \in \{0,1\}^p \tag{10}$$

Therefore, we get $\overline{U}[0..0] = 0$.

The purpose of the EMS algorithm is to simplify the message update only at the check node output, by performing step $(\mathcal{S}_3)$ with a reduced number of operations. By replacing the FFT in $(\mathcal{S}_3)_{FFT}$ with another message update rule, it is no longer important that the field order is a power of 2[1]. Therefore, the EMS algorithm could be applied to any Galois field order q as long as q is a prime number or a power of a prime number. We have chosen to keep the tensoral notations used for GF(2^p) fields in order to have coherent notations with Section 2. It is straightforward to adapt the notations used in this section to any field order.
As in the previous section, we omit the temporal indices in the equations. Also, for ease of presentation and without loss of generality, we assume that the output message is labeled by d_c (the last branch of the check), which means that the incoming messages of a degree d_c check node are labeled $\{U_{pc}, \forall c \in 1 \ldots d_c - 1\}$, and the output message is denoted $V_{d_c p}$.

3.1 Configuration Sets

Similarly to the binary case, the goal of any message passing algorithm is to assign a *reliability measure* to each and every entry in the ouput message $V_{d_c p}$, using the values of the incoming messages. In the

[1] We recall that the Fourier transform has the form (8) only for densities over GF(2^p) and this motivated the choice of $q = 2^p$ in Section 2

logBP algorithm, this reliability measure is the local *a posteriori* probability, but it is hard to compute efficiently. With a suboptimal EMS algorithm, we aim at computing a suboptimal reliability measure, with low computational complexity. To make it happen, we try not only to avoid any complex operations (multiplications, exponentials, logarithms, etc), but also to compute the output reliability using only a limited number of incoming values (obviously the largest of such values).

We start by selecting in each incoming message $\overline{U}_{pc}$ the n_m largest values that we denote $\overline{u}_{pc}^{(k_c)}$, $k_c = 1 \ldots n_m$. We use the following notation for the associated field elements $\alpha_c^{(k_c)}(x)$, so that we have (according to (10))

$$\overline{u}_{pc}^{(k_c)} = \log\left(\frac{\text{Prob}\left(h_c(x)i_c(x) = \alpha_c^{(k_c)}(x)\right)}{\text{Prob}\left(h_c(x)i_c(x) = 0\right)}\right) = \overline{U}_{pc}\left[\alpha_{c_1}^{(k_c)}..\alpha_{c_p}^{(k_c)}\right]$$

In principle, the number of values selected by the EMS algorithm could differ from one message to another, depending for exemple on the dynamic of the reliability. This issue is not discussed in this paper, and for simplicity of the presentation we select *exactly* n_m values in each and every incoming message.

With these largest values, we build the following set of *configurations*:

$$\text{Conf}(n_m) = \left\{\boldsymbol{\alpha}_{\mathbf{k}} = \left[\alpha_1^{(k_1)}(x), \ldots, \alpha_{d_c-1}^{(k_{d_c-1})}(x)\right]^T \ : \ \forall \mathbf{k} = [k_1, \ldots, k_{d_c-1}]^T \in \{1 \ldots n_m\}^{d_c-1}\right\}$$

Any vector of $d_c - 1$ field elements in this set is called a *configuration*. The set $\text{Conf}(n_m)$ corresponds to the set of configurations built from the n_m largest probabilities in each incoming message. Hence its cardinality is

$$|\text{Conf}(n_m)| = {n_m}^{d_c-1}$$

Note that $\text{Conf}(1)$ contains only one configuration that we call *order-0* configuration.

In some cases, such as for large values of d_c or n_m, the number of considered configurations in $\text{Conf}(n_m)$ remains too large. In order to further reduce the number of chosen configurations, we consider the following subset of $\text{Conf}(n_m)$, for $n_c \leq d_c - 1$:

$$\text{Conf}(n_m, n_c) = \text{Conf}(n_m)^{(0)} \cup \text{Conf}(n_m)^{(1)} \cup \ldots \cup \text{Conf}(n_m)^{(n_c)}$$

where $\text{Conf}(n_m)^{(l)}$ is the subset of configurations that differ from the *order-0* configuration in *exactly* l entries. The set $\text{Conf}(n_m, n_c)$ is therefore the subset of configurations belonging to $\text{Conf}(n_m)$ that differ *at most* in n_c entries from the *order-0* configuration. The number of elements in $\text{Conf}(n_m, n_c)$ is

$$|\text{Conf}(n_m, n_c)| = \sum_{k=0}^{n_c} \begin{pmatrix} d_c - 1 \\ k \end{pmatrix} (n_m - 1)^k \simeq \begin{pmatrix} d_c - 1 \\ n_c \end{pmatrix} {n_m}^{n_c} \tag{11}$$

The idea of using $\text{Conf}(n_m, n_c)$ instead of $\text{Conf}(n_m)$ is that it is much smaller in many cases, and it is built from configurations with large probabilities since the *order-0* configuration is the one with the highest probability. We finally remark that $\text{Conf}(n_m) = \text{Conf}(n_m, d_c - 1)$.

To each configuration, we need to assign a reliability that is straightforward to compute. We have chosen the following function $L(\boldsymbol{\alpha}_{\mathbf{k}})$ as reliability measure:

$$L(\boldsymbol{\alpha}_{\mathbf{k}}) = \sum_{c=1 \ldots d_c-1} \overline{u}_c^{(k_c)} \tag{12}$$

Note that this choice is arbitrary, but is based on an analogy with the binary case. Moreover, this choice of reliability seems natural since the EMS with parameters $(n_m, n_c) = (q, d_c - 1)$ is similar to the algorithm based on the approximation of the $max^*(.)$ operator by a $max(.)$ operator presented in [12]. With these notations, we are now able to build a EMS algorithm from the reliabilities $L(\boldsymbol{\alpha_k})$ of the configurations in $\text{Conf}(n_m, n_c)$. To this aim, we denote $\text{Conf}_{i_{d_c}(x)}(n_m, n_c)$ the subset of $\text{Conf}(n_m, n_c)$ defined by the check node constraint:

$$\text{Conf}_{i_{d_c}(x)}(n_m, n_c) = \left\{ \boldsymbol{\alpha_k} \in \text{Conf}(n_m, n_c) \quad : \quad h_{d_c}(x) i_{d_c}(x) + \sum_{c=1}^{d_c-1} \alpha_c^{(k_c)}(x) = 0 \right\}$$

Finally, we remark that for some choices of the parameters (n_m, n_c), the set $\text{Conf}_{i_{d_c}(x)}(n_m, n_c)$ could be *empty*. If $\text{Conf}_{i_{d_c}(x)}(n_m, n_c)$ is empty for some values of $i_{d_c}(x)$, this could create a problem of convergence for the EMS algorithm since each and every entry in the output message has to be filled by a reliability. This statement has been verified by simulations. However, this problem is readily overcome by using the subsets $\text{Conf}_{i_{d_c}(x)}(q, 1)$ which are non-empty for any value of $i_{d_c}(x) \in GF(q)$.

3.2 EMS Algorithm

Using the notations of the preceding section, the complete EMS algorithm is given thereafter:

$(\boldsymbol{\mathcal{S}_1})_{\boldsymbol{EMS}}$: sum step, variable node update for a degree d_v node:

$$\begin{aligned} \overline{U}_{tp} &= \overline{L} + \sum_{v=1, v\neq t}^{d_v} \overline{V}_{pv} \qquad\qquad t = 1 \dots d_v \\ \text{or } \overline{U}_{tp}[i_1..i_p] &= \overline{L}[i_1..i_p] + \sum_{v=1, v\neq t}^{d_v} \overline{V}_{pv}[i_1..i_p] \qquad\qquad (i_1..i_p) \in \{0,1\}^p \end{aligned} \tag{13}$$

■

$(\boldsymbol{\mathcal{S}_2})_{\boldsymbol{EMS}}$: permutation step (from variable to check nodes):

$$\begin{aligned} \text{vec}\left(\overline{U}_{pc}\right) &= P_{h(x)} \text{ vec}\left(\overline{U}_{vp}\right) \\ \text{or } \overline{U}_{pc}[i_1..i_p] &= \overline{U}_{vp}[j_1..j_p] \qquad\qquad (i_1..i_p) \in \{0,1\}^p \text{ with } i(x) = h(x)j(x) \end{aligned} \tag{14}$$

The permutation step from check to variable nodes is performed using $P_{h(x)}^{-1}$. ■

$(\boldsymbol{\mathcal{S}_3})_{\boldsymbol{EMS}}$: message update for a degree d_c check node:

- from the $d_c - 1$ incoming messages $\overline{U}_{pc}$, build the sets $\mathcal{S}_{i_{d_c}(x)} = \text{Conf}_{i_{d_c}(x)}(q, 1) \cup \text{Conf}_{i_{d_c}(x)}(n_m, n_c)$, then

$$\overline{V}_{d_c p}[i_{d_{c_1}}..i_{d_{c_p}}] = \max_{\boldsymbol{\alpha_k} \in \mathcal{S}_{i_{d_c}(x)}} \{L(\boldsymbol{\alpha_k})\} \qquad (i_{d_{c_1}}..i_{d_{c_p}}) \in \{0,1\}^p \tag{15}$$

- post-processing

$$\overline{V}_{cp}[i_1..i_p] = \overline{V}_{cp}[i_1..i_p] - \overline{V}_{cp}[0..0] \qquad (i_1..i_p) \in \{0,1\}^p \quad c = 1 \dots d_c \tag{16}$$

■

The post-processing in step $(\mathcal{S}_3)_{EMS}$ is necessary for numerical reasons, to ensure the non-divergence of the algorithm. Without this post-processing, the values of the LDR messages would converge to the highest achievable numerical value in a few iterations. Subtracting the value of the messages at index 0 has also another meaning since it forces the output of the check nodes to have the same LDR structure as defined in (10).
It is readily seen that the EMS algorithm reduces to the MS algorithm when it is applied to GF(2) (the proof is given in Appendix A). This is why we named our algorithm the extended MS algorithm. Another way to see it is to express the check node update with the help of $max^*(.)$ operators, and then approximate the expressions with $max(.)$ operators, just as it is done in the MS algorithm [12].

3.3 Computation Complexity

For $n_m = q$ and $n_c = d_c - 1$, the EMS algorithm becomes the MS algorithm proposed in [12]. As a result, the algorithm proposed in [12] has complexity per iteration $O(Nd_vq)$ at the variable nodes and $O(Md_cq^2)$ at the check nodes, assuming the recursive updating proposed in [6]. In our approach, the main differences in complexity are the additional cost to determine the configuration sets $\text{Conf}(n_m, n_c)$ and the resulting reduced cost to evaluate the values $\overline{v}_{cp}$.
At each node, all configuration sets can be determined by identifying the n_m largest values for each of the d_c incoming branches. Based on a binary tree representation, the resulting cost becomes $O(Md_c(q + (n_m - 1)\log q)$.
The recursive approach of [6] can still be implemented in conjunction with configuration sets. For simplicity, we consider all representations in $\text{Conf}(n_m)$. Both the forward and backward recursions start with n_m values, and at their first step, evaluate up to ${n_m}^2$ intermediary reliability values (in which case no operations are needed). However, as soon as all q possible intermediary reliability values have been evaluated, the number of operations becomes n_mq for the remaining steps. As a result, the complexity of this approach can be evaluated as $O(Md_cn_mq)$.
This recursive approach requires d_c serial steps. Another possible approach with a higher level of parallelism if $n_c < d_c - 1$ is to compute the values in $\text{Conf}(n_m)^{(i)}$ from those in $\text{Conf}(n_m)^{(i-1)}$ for $i = 1...n_c$. Since given a representation in $\text{Conf}(n_m)^{(i)}$, it is always possible to find a representation in $\text{Conf}(n_m)^{(i-1)}$ which differs in at most one position, the resulting complexity becomes based on (11): $O(Md_c\binom{d_c-1}{n_c}{n_m}^{n_c})$. In general, the complexity of this direct non recursive approach is slightly higher than that of the previous recursive one, but the number of serial steps is reduced from d_c to $n_c + 1$.
We observe that both approaches significantly reduce the complexity $O(Md_cq^2)$ of the algorithm proposed in [12]. In fact, for values of n_m and n_c which allow to approach the performance of BP decoding (with the enhancements discussed in the next section), these complexities are of the same order of $O(Md_cq\log q)$ which corresponds to the implementation of BP with FFTs. However only additions are performed in our simplified approach.

3.4 Scaled and offset EMS Algorithms

The EMS algorithm is suboptimal and naturally introduces a performance degradation compared to the BP algorithm. The main reason (although not necessarily the only reason) of this degradation is that the reliabilities computed in the EMS algorithm are *over-estimated*. This causes the suboptimal algorithm to converge too rapidly to a local minimum, which is most often a pseudo-codeword. This behavior has also been observed for binary LDPC codes decoded with MS as well as for turbo-codes decoded with max*-log-map (either parallel or block turbo-codes). For binary LDPC codes, a simple technique that is used to compensate for this over-estimation is to *reduce* the magnitude of the messages at the variable

node input by means of a factor or an offset [19]. We propose to apply these correction techniques to the EMS algorithm, either using a factor correction (17) or an offset correction (18).

$(\mathcal{S}_1)_{EMSfactor}$: sum step with factor correction, variable node update for a degree d_v node:

$$\overline{V}_{pv}[i_1..i_p] = \overline{V}_{pv}[i_1..i_p]/\alpha_{fac} \qquad (i_1..i_p) \in \{0,1\}^p \tag{17}$$

$$\begin{aligned} \overline{U}_{tp} &= \overline{L} + \sum_{v=1, v\neq t}^{d_v} \overline{V}_{pv} \qquad t = 1 \dots d_v \\ \text{or } \overline{U}_{tp}[i_1..i_p] &= \overline{L}[i_1..i_p] + \sum_{v=1, v\neq t}^{d_v} \overline{V}_{pv}[i_1..i_p] \qquad (i_1..i_p) \in \{0,1\}^p \end{aligned}$$

■

$(\mathcal{S}_1)_{EMSoffset}$: sum step with offset correction, variable node update for a degree d_v node:

$$\begin{aligned} \overline{V}_{pv}[i_1..i_p] &= \max\left(\overline{V}_{pv}[i_1..i_p] - \alpha_{off}, 0\right) \qquad if\ \ \overline{V}_{pv}[i_1..i_p] > 0 \\ \overline{V}_{pv}[i_1..i_p] &= \min\left(\overline{V}_{pv}[i_1..i_p] + \alpha_{off}, 0\right) \qquad if\ \ \overline{V}_{pv}[i_1..i_p] < 0 \end{aligned} \tag{18}$$

$$\begin{aligned} \overline{U}_{tp} &= \overline{L} + \sum_{v=1, v\neq t}^{d_v} . \overline{V}_{pv} \qquad t = 1 \dots d_v \\ \text{or } \overline{U}_{tp}[i_1..i_p] &= \overline{L}[i_1..i_p] + \sum_{v=1, v\neq t}^{d_v} \overline{V}_{pv}[i_1..i_p] \qquad (i_1..i_p) \in \{0,1\}^p \end{aligned}$$

■

Of course, the values of the factor (or offset) have to be chosen carefully, by optimizing an ad-hoc cost function. In this paper, we have chosen the values of the correction factor by minimizing the decoding threshold of the LDPC code, computed with estimated density evolution. The density evolution is simulated by means of Monte Carlo estimations of the densities, using a sufficiently large number of samples in order to get a low variance of estimation on the threshold value. In our simulations, a number of $N_{MC} = 20000$ message samples was sufficient to get good threshold variances. Note that this number is rather small since the LDPC codes that we consider all have $d_v = 2$, which means that they are the sparsest LDPC codes possible, and for ultra-sparse LDPC codes with $d_v = 2$ the asymptotic behavior of the decoding algorithms is reached very rapidly. The decoding threshold δ is defined as the minimum signal to noise ratio $(E_b/N_0)_{dB}$ such that the bit error probability goes to zero as the number of decoding iterations goes to infinity. In practice, we choose a maximum of 200 decoding iterations and we stop the density evolution when the bit error probability is lower than 10^{-3}. Note that we use density evolution techniques only to choose the correction values and not to compute accurate thresholds. In Figure 3 the decoding thresholds $\delta = (E_b/N_0)_{dB}$ are depicted as a function of the correction factor (or offset), for the $(d_v, d_c) = (2, 4)$ LDPC code over GF(256), and for two different EMS complexities $(n_m, n_c) = (7, 2)$ and $(n_m, n_c) = (13, 3)$. We notice that the threshold is actually a good cost function in order to choose the correction values since the curves are convex-$\cup$ and have a unique global minimum. We observe as in the binary case that the optimum values for the offset correction are slightly worse than for the factor correction [11]. The thresholds and the corresponding correction values for various codes and complexities are tabulated and discussed in the next section.

4 Performance comparison

4.1 Code thresholds and correction factors

The optimum threshold values of the scaled and offset EMS algorithms with $(n_m, n_c) = (7, 2)$ and $(n_m, n_c) = (13, 3)$ and different field orders have been reported in Table 1 for the rate $R = 1/2$ ($d_v = 2, d_c = 4$) code and the rate $R = 3/4$ ($d_v = 2, d_c = 8$) code. For comparison purpose, the capacity limit is $(E_b/N_0)_{min}$=0.18 dB for $R = 1/2$ and $(E_b/N_0)_{min}$=1.63 dB for $R = 3/4$. As expected, the threshold values improve as n_m and n_c increase, and can approach the threshold of BP closely with much less complexity. For $(n_m, n_c) = (13, 3)$, we observe that the best values are obtained for $q = 128$ in both cases. This can be explained by the fact that as q increases, the gap between BP decoding and the simpliest corrected EMS algorithm obtained for $(n_m, n_c) = (2, 1)$ increases too. For the ($d_v = 2, d_c = 4$) code and $(n_m, n_c) = (2, 1)$, the threshold values of the factor correction (respectively offset correction) EMS are $\delta = 1.58$ dB (respectively $\delta = 2.18$ dB) for $q = 64$, $\delta = 1.69$ dB (respectively $\delta = 2.43$ dB) for $q = 128$ and $\delta = 1.81$ dB (respectively $\delta = 2.69$ dB) for $q = 256$. This explains why, as q increases, it is necessary to increase also (n_m, n_c) in order to stay close to the BP threshold.
The optimum threshold value δ for the scaled and offset EMS algorithms with $(n_m, n_c) = (7, 2)$ and $(n_m, n_c) = (13, 3)$ has been represented as a function of q and compared to that of BP decoding in Figure 4. We observe that for fields of small order (say $q \leq 32$), the corrected EMS algorithms with $(n_m, n_c) = (7, 2)$ are sufficient to closely approach BP decoding and the performance improves as q increases. For $q = 64$ and $q = 128$, the corrected EMS algorithms with $(n_m, n_c) = (13, 3)$ are necessary to approach BP, with a performance still improving as q increases. However, for $q \geq 256$, larger values (n_m, n_c) are required to approach BP. However for these large field, the potential performance gain achieved by increasing q is very small. We observe that this trend is similar to that reported in [7] with respect to the weight profile of the random ensembles of GF(q)-LDPC codes.

4.2 Frame error rate performance for small codeword lengths

In this section, we present the simulation results of the different EMS algorithms for various LDPC codes and various field orders. Motivated by the asymptotical threshold analysis presented in the last section, we focus on codes over high order field, that is GF(64), GF(128) and GF(256). For those high orders, it has been shown that the best LDPC codes decoded with BP should be *ultra-sparse* [7, 8], that is, with the minimum variable node connection $d_v = 2$. We have therefore simulated LDPC codes for two different rates and with $d_v = 2$: rate $R = 1/2$ ($d_v = 2, d_c = 4$)-LDPC codes and rate $R = 3/4$ ($d_v = 2, d_c = 8$)-LDPC codes. These codes have been designed with the PEG method of [20]. In all cases, the correction values have been chosen as those reported in Table 1, assuming the code lengths are long enough to validate results obtained via density evolution. In our simulations, we have more extensively studied the impact of the field order and the codeword length on the $(2, 4)$-LDPC code.
Figures 5 and 6 show the frame error rate (FER) of the $(2, 4)$-LDPC code for a small codeword size corresponding to a typical ATM frame ($K = 53$ information bytes), for two fields, GF(256) and GF(64), respectively. We observe that for GF(256), with a significant complexity reduction, the EMS algorithm with $(n_m, n_c) = (13, 3)$ and offset correction performs within 0.15 dB of BP decoding. Hence the threshold value obtained via density evolution provides a good estimate in that case. The EMS algorithm with $(n_m, n_c) = (13, 3)$ and factor correction performs even slightly better at low SNR values, but suffers from an error floor. This behavior with scaling correction factor has also been observed for irregular binary LDPC codes and in that case, can be compensated [21]. The sphere packing bound (SPB) of [22] has also been represented in Figures 5. We observe that the EMS algorithm with correction performs within 1.0 dB of the SPB with relatively low complexity. Hence the code considered is quite powerful considering that the SPB does not take into account the losses due to BPSK modulation (roughly evaluated at 0.2 dB) and

to iterative decoding (especially for such a relatively short code length). In fact, such codes have lengths for which very few competitive codes close to the SPB and decodable in real time can be found (for short lengths, BCH codes are close to the SPB and for long lengths, turbo or LDPC codes are close to the SPB) [23]. These observations confirm that medium length non binary LDPC codes over large finite fields are good codes [6, 7] and our results provide efficient decoding algorithms for fast VLSI implementations of these codes.

The performance gaps between the same algorithms become much smaller for the GF(64) code represented in Figure 6, mostly due to the fact that the relative difference between the field order q and the algorithm complexity (n_m, n_c) is much smaller compared to the GF(256) code. We can also observe a poorer performance achieved by BP decoding in that case, with an error floor starting at the FER $10^{-4.5}$. This can be attributed to both the smaller field order which worsen the low weight profile [7] and the suboptimality of iterative decoding. For indoor applications with those packet sizes, a frame error rate of about $10^{-2} - 5 \cdot 10^{-3}$ (very high) is usually desired. For other applications, still with those packet sizes (such as satellite communications), a FER below $10^{-8} - 10^{-9}$ (very low) needs to be achieved. Consequently, the results reported in Figures 5 and 6 are relevant to either case.

In order to show that our approach is general and also works for large values of d_c (and especially with n_c much smaller than d_c), the results for the $(2, 8)$-LDPC code over GF(128) are depicted in Figure 7. The EMS still performs very well compared to BP while $n_c = 2$ or $n_c = 3$ is very small compared to $d_c = 8$. The same remarks made for the previous figures apply to this figure.

Figures 8 and 9 depict the performance of the $(2, 4)$-LDPC codes for a larger codeword length corresponding to a typical MPEG frame size of $K = 188$ information bytes. In both cases, a better error performance has been achieved due to the increase in code length, and the same remarks as for the shorter codes can be made. Moreover, we observe that the gap between the SPB and the offset correction EMS has not increased compared to Figures 5-6. This is important since it shows that our reduced complexity algorithm is robust to the length of the LDPC codes.

Finally, we observe a phenomenon that has been already pointed out for irregular binary LDPC codes; the MS (in our case the EMS) corrected by an offset value could perform better than the BP algorithm at low FER. This can be seen in Figures 6, 7 and 9. Although this could be surprising, we must recall that the BP algorithm is not optimal, and that the effect of pseudo-codewords due to the presence of cycles in finite length codes could be different on the decoding algorithms.

5 Conclusion

We have presented several decoding algorithms for nonbinary LDPC codes over GF(q) which aim at reducing the computational complexity compared to the regular BP decoder. In the probability domain, the complexity of BP can be greatly reduced by performing the check node updates in the frequency domain. Although the use of FFTs has already been proposed in the literature for nonbinary codes, we show in this paper that the use of a tensoral representation of the messages is useful for a comprehensive description of the decoder. The complexity of the BP-FFT decoder is reduced to $O(d_c q \log_2(q))$ per degree d_c check node but still suffers from the fact that multiplications and divisions (for message normalization) are required. In order to avoid these complex operations, we have also proposed reduced complexity algorithms in the log-domain using log-density-ratio as messages. The proposed EMS algorithm is a generalization of the MS algorithm used for binary codes, and has the advantage of performing additions only, while using only a limited number n_m of messages for the check node update. We stress the fact that we focus on a local simplification of the check node processing. This means in particular that the EMS algorithm apply both for regular and irregular nonbinary LDPC codes, although all the simulations in this paper were done with regular LDPC codes.

The complexity of the EMS algorithm is $O(d_c n_m q)$ per check node. For values of n_m providing near-BP error performance, this complexity is roughly the same as that of the BP-FFT decoder, but without multiplications or divisions. We have also shown that by correcting the messages along the decoding process, the EMS algorithm can approach the performance of the BP-FFT decoder and even in some cases slightly outperform the BP-FFT decoder. The correction techniques that we developed are based on an application of a factor or an offset to the messages at the input of a variable node, and the factor/offset has been optimally chosen with the help of simulated density evolution. The EMS algorithm then becomes a good candidate for hardware implementation of nonbinary LDPC decoders, since its complexity has been greatly reduced compared to that of other decoding algorithms for non binary LDPC codes and the performance degradation is small or negligible.

A EMS reduces to MS in GF(2)

In this appendix, it is shown that the proposed suboptimal algorithm is based on assumptions that are coherent with the binary MS algorithm. More precisely, when we apply EMS to the binary case, it reduces to the MS algorithm. Since one can compute the check node output message recursively, taking the input messages pairwise, we can limit our discussion to the case of only two incoming messages and one ouput message.

A.1 Min-Sum Algorithm

In the binary case, the messages in their tensoral representation are simple vectors :

$$U_0 = \begin{bmatrix} U_0[0] \\ U_0[1] \end{bmatrix} \qquad U_1 = \begin{bmatrix} U_1[0] \\ U_1[1] \end{bmatrix} \qquad V_2 = \begin{bmatrix} V_2[0] \\ V_2[1] \end{bmatrix} \tag{19}$$

Let $u_0 = \log(U_0[0]/U_0[1])$ and $u_1 = \log(U_1[0]/U_1[1])$ be the two input messages in their usual LDR form, and let $v_2 = \log(V_2[0]/V_2[1])$ be the output message. The MS algorithm is given by:

$$v_2 = \text{sign}(u_0)\ \text{sign}(u_1)\ \min\left(|u_0|, |u_1|\right) \tag{20}$$

In order to compare the two algorithms (MS, EMS), we consider four cases:

Case 1: $(u_0 > 0,\ u_1 > 0,\ |u_0| > |u_1|) \Rightarrow v_2 = |u_1| = u_1$

Case 2: $(u_0 < 0,\ u_1 < 0,\ |u_0| > |u_1|) \Rightarrow v_2 = |u_1| = -u_1$

Case 3: $(u_0 < 0,\ u_1 > 0,\ |u_0| < |u_0|) \Rightarrow v_2 = -|u_1| = u_0$

Case 4: $(u_0 > 0,\ u_1 < 0,\ |u_0| < |u_0|) \Rightarrow v_2 = -|u_1| = -u_0$

Note that these four cases are sufficient because the other four possible cases are symmetrical only by exchanging u_0 and u_1.

A.2 EMS algorithm in GF(2)

The messages used in the EMS algorithm are defined by (see (10)):

$$\overline{U}_0 = \begin{bmatrix} 0 \\ \overline{U}_0[1] \end{bmatrix} \qquad \overline{U}_1 = \begin{bmatrix} 0 \\ \overline{U}_1[1] \end{bmatrix} \qquad \overline{V}_2 = \begin{bmatrix} 0 \\ \overline{V}_2[1] \end{bmatrix} \tag{21}$$

where $\overline{U}_0[1] = -u_0$, $\overline{U}_1[1] = -u_1$ and $\overline{V}_2[1] = -v_2$.

We have performed step $(\mathcal{S}_3)_{EMS}$ of the EMS algorithm in each of the four cases:

Case 1: $\overline{V}_2 = \begin{bmatrix} 0 \\ \overline{U}_1[1] \end{bmatrix} \overset{after\ post-processing}{\Rightarrow} \overline{V}_2 = \begin{bmatrix} 0 \\ \overline{U}_1[1] \end{bmatrix}$

Case 2: $\overline{V}_2 = \begin{bmatrix} \overline{U}_0[1] + \overline{U}_1[1] \\ \overline{U}_0[1] \end{bmatrix} \overset{after\ post-processing}{\Rightarrow} \overline{V}_2 = \begin{bmatrix} 0 \\ -\overline{U}_1[1] \end{bmatrix}$

Case 3: $\overline{V}_2 = \begin{bmatrix} 0 \\ \overline{U}_0[1] \end{bmatrix} \overset{after\ post-processing}{\Rightarrow} \overline{V}_2 = \begin{bmatrix} 0 \\ \overline{U}_0[1] \end{bmatrix}$

Case 4: $\overline{V}_2 = \begin{bmatrix} \overline{U}_0[1] + \overline{U}_1[1] \\ \overline{U}_1[1] \end{bmatrix} \overset{after\ post-processing}{\Rightarrow} \overline{V}_2 = \begin{bmatrix} 0 \\ -\overline{U}_0[1] \end{bmatrix}$

The results obtained by the EMS algorithm are the same than for the MS algorithm. The two algorithms are therefore equivalent in GF(2).

References

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

[2] T.J. Richardson, M.A. Shokrollahi and R.L. Urbanke, "Design of Capacity-Approaching Low-Density Parity Check Codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619-637, Feb. 2001.

[3] T.J. Richardson and R.L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599-618, Feb. 2001.

[4] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman, "Improved Low-Density Parity-Check Codes Using Irregular Graphs," *IEEE Trans. Inform. Theory*, vol. 47, pp. 585-598, Feb. 2001.

[5] S.Y. Chung, G.D. Forney, T.J. Richardson and R.L. Urbanke, "On the Design of Low-Density Parity Check Codes within 0.0045 dB of the Shannon Limit," *IEEE Commun. Lett.*, vol. 5, pp. 58-60, Feb. 2001.

[6] M. Davey and D.J.C. MacKay, "Low Density Parity Check Codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, pp. 165-167, June 1998.

[7] X.-Y. Hu and E. Eleftheriou, "Binary Representation of Cycle Tanner-Graph GF(2^q) Codes," *The Proc. IEEE Intern. Conf. on Commun.*, Paris, France, pp. 528-532, June 2004.

[8] D.J.C. MacKay and M. Davey, "Evaluation of Gallager Codes for Short Block Length and High Rate Applications," *The Proc. IMA Workshop on Codes, Systems and Graphical Models*, 1999.

[9] H. Song and J.R. Cruz, "Reduced-Complexity Decoding of Q-ary LDPC Codes for Magnetic Recording," *IEEE Trans. Magn.*, vol. 39, pp. 1081-1087, Mar. 2003.

[10] L. Barnault and D. Declercq, "Fast Decoding Algorithm for LDPC over GF(2^q)," *The Proc. 2003 Inform. Theory Workshop*, Paris, France, pp. 70-73, Mar. 2003,

[11] J. Chen and M. Fossorier, "Density Evolution for Two Improved BP-Based Decoding Algorithms of LDPC Codes," *IEEE Commun. Lett.*, vol. 6, pp. 208-210, May 2002.

[12] H. Wymeersch, H. Steendam and M. Moeneclaey, "Log-Domain Decoding of LDPC Codes over GF(q)," *The Proc. IEEE Intern. Conf. on Commun.*, Paris, France, June 2004, pp. 772-776.

[13] A. P. Hekstra, "Reduced-Complexity Decoding of LDPC Codes over GF(q)," preprint, 2003.

[14] F. Kschischang, B. Frey and H-A. Loeliger "Factor Graphs and the Sum Product Algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498-519, Feb. 2001.

[15] R. M. Tanner "A Recursive Approach to Low Complexity Codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533-547, Sept. 1981.

[16] N. Wiberg "Codes and Decoding on General Graphs," PhD thesis, Linkoping univ. Sweden, 1996.

[17] X. Li and M.R. Soleymani, "A Proof of the Hadamard Transform Decoding of the Belief Propagation Decoding for LDPCC over GF(q)", *The Proc. Fall 2004 Vehic. Tech. Conf.*, Sept. 2004.

[18] M. Fossorier, M. Mihaljević and H. Imai, "Reduced Complexity Iterative Decoding of LDPC codes based on Belief Propagation," *IEEE Trans. Commun.*, vol. 47, pp. 673-680, May 1999.

[19] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier and X.-Y. Hu, "Reduced Complexity Decoding of LDPC codes," *IEEE Trans. Commun.*, to appear.

[20] X.-Y. Hu, E. Eleftheriou and D.M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs," *IEEE Trans. Inform. Theory*, vol. 51, pp. 386-398, Jan. 2005.

[21] J. Chen, R. M. Tanner, C. Jones and Y. Li, "Improved Min-Sum Decoding Algorithms for Irregular LDPC Codes," *submitted to IEEE Commun. Lett.*, Jan. 2005.

[22] C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel", *Bell System Tech. Jour.*, vol. 38, pp. 611-656, 1959.

[23] S. Dolinar, D. Dolinar and F. Pollara, "Code performance as a function of block size", TMO Progress Report 42-133, Jet Propulsion Laboratory, May 1998.

[24] H. Wymeersch, H. Steendam and M. Moeneclaey, "Computational complexity and quantization effects of decoding algorithms of LDPC codes over GF(q)", *The Proc. ICASSP-2004*, Montreal, Canada, pp. 772-776, May 2004,

[25] A. Goupil, M. Colas, G. Gelle and D. Declercq, "On BP Decoding of LDPC Codes over Groups", *The Proc. Int. Symp. on Turbo-Codes*, Munich, Germany, April 2006,

[26] J. Zhao, F. Zarkeshvari and A.H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding LDPC codes", *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549-554, April 2005.

Code	q	(n_m, n_c)	EMS α_{fac}	EMS α_{off}	BP
$(d_v = 2, d_c = 4)$	64	(7,2)	$\alpha = 1.2$ $\delta = 0.73$	$\alpha = 0.5$ $\delta = 0.78$	$\delta = 0.58$
		(13,3)	$\alpha = 1.15$ $\delta = 0.645$	$\alpha = 0.4$ $\delta = 0.66$	
	128	(7,2)	$\alpha = 1.35$ $\delta = 0.84$	$\alpha = 0.9$ $\delta = 1.0$	$\delta = 0.52$
		(13,3)	$\alpha = 1.2$ $\delta = 0.61$	$\alpha = 0.5$ $\delta = 0.64$	
	256	(7,2)	$\alpha = 1.6$ $\delta = 1.10$	$\alpha = 1.8$ $\delta = 1.50$	$\delta = 0.5$
		(13,3)	$\alpha = 1.3$ $\delta = 0.64$	$\alpha = 0.6$ $\delta = 0.70$	
$(d_v = 2, d_c = 8)$	64	(7,2)	$\alpha = 1.4$ $\delta = 2.09$	$\alpha = 0.9$ $\delta = 2.15$	$\delta = 1.94$
		(13,3)	$\alpha = 1.25$ $\delta = 1.99$	$\alpha = 0.55$ $\delta = 2.00$	
	128	(7,2)	$\alpha = 1.6$ $\delta = 2.15$	$\alpha = 1.3$ $\delta = 2.26$	$\delta = 1.92$
		(13,3)	$\alpha = 1.4$ $\delta = 1.98$	$\alpha = 0.7$ $\delta = 2.00$	
	256	(7,2)	$\alpha = 1.8$ $\delta = 2.27$	$\alpha = 2.0$ $\delta = 2.50$	$\delta = 1.91$
		(13,3)	$\alpha = 1.4$ $\delta = 1.99$	$\alpha = 0.7$ $\delta = 2.01$	

Table 1: Correction factors and offsets for the $(d_v = 2, d_c = 4)$-LDPC and the $(d_v = 2, d_c = 8)$-LDPC codes, for three different fields.

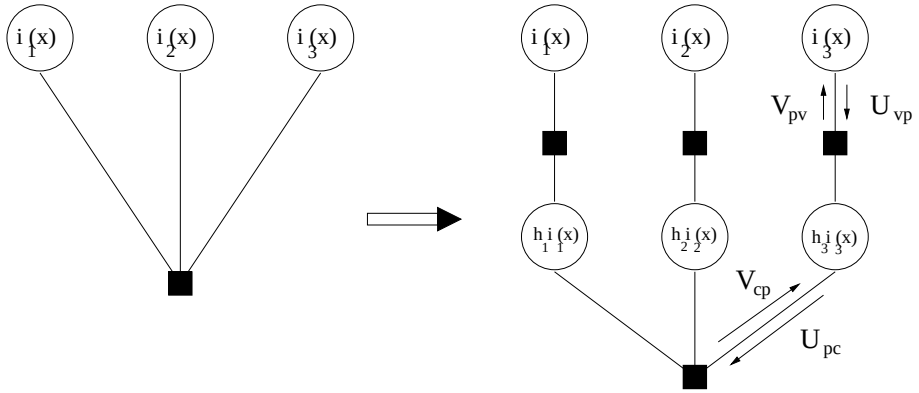

Figure 1: Transformation of the factor graph in order to explicitly represent the nonzero values in the matrix H.

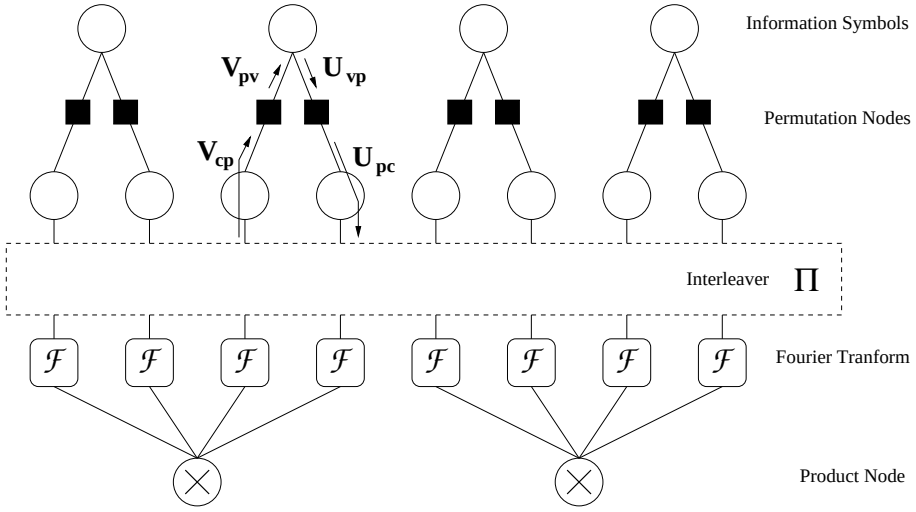

Figure 2: Factor graph of a LDPC code over GF(2^p).

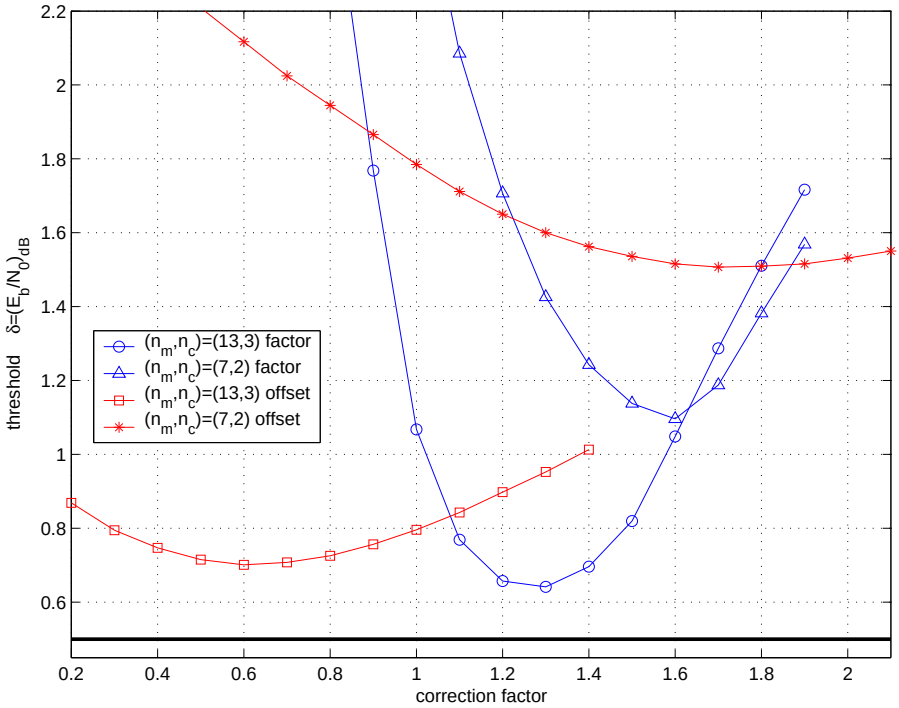

Figure 3: Decoding threshold of a regular $(2, 4)$ LDPC code over GF(256) vs. correction factors.

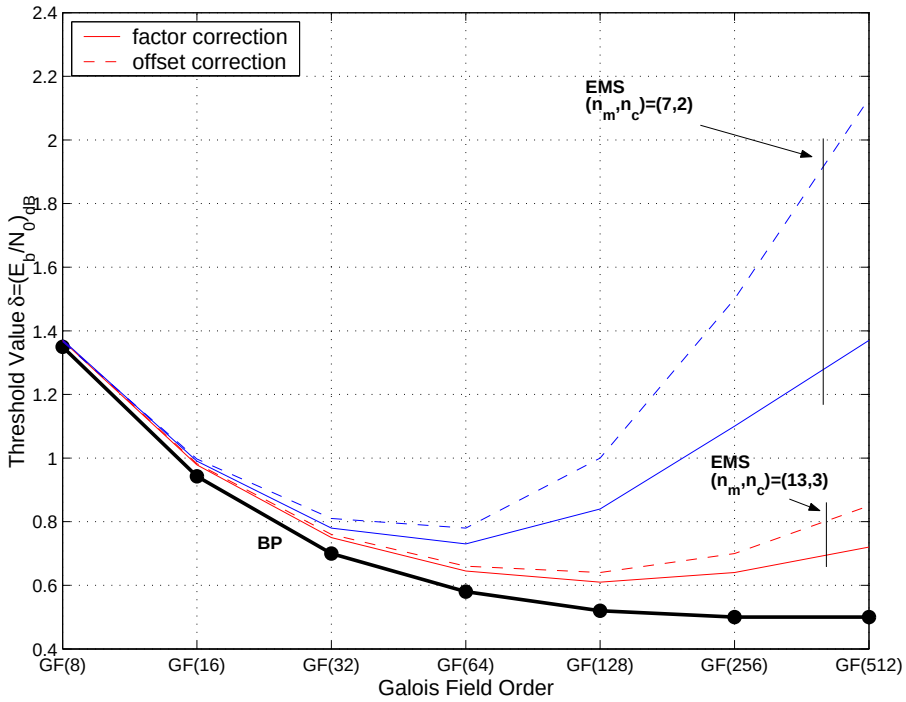

Figure 4: Threshold value $\delta = (E_b/N_0)_{dB}$ versus the field order for the $(d_v = 2, d_c = 4)$-LDPC code.

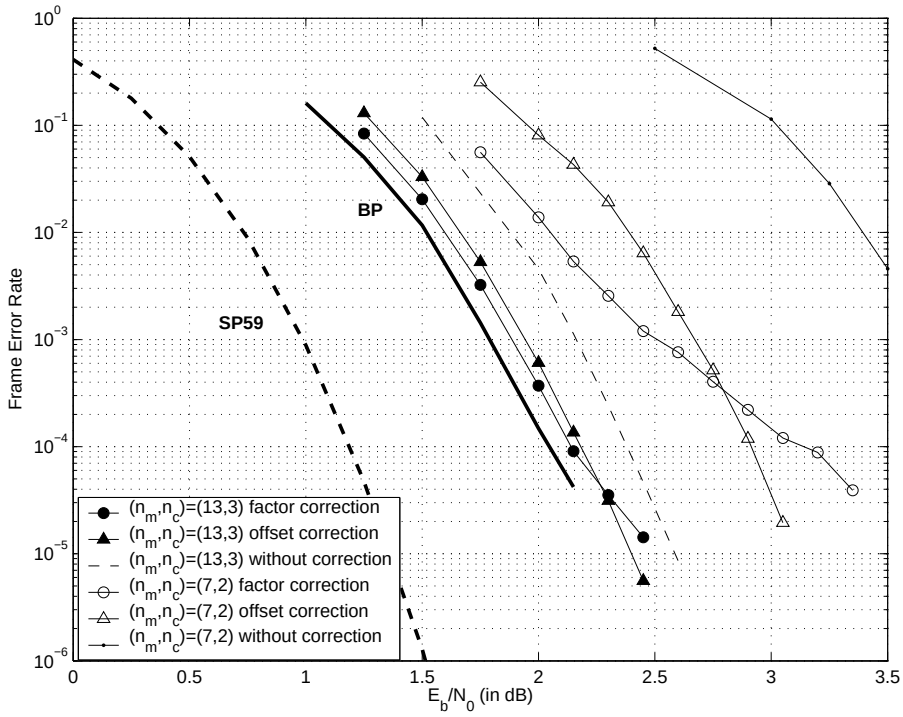

Figure 5: Comparison between BP and EMS decodings for a regular (2,4)-LDPC code over GF(256) ($R = 1/2$, $N = 106$ GF(256) symbols, $N_b = 848$ equivalent bits). 50 decoding iterations have been used for all decoders.

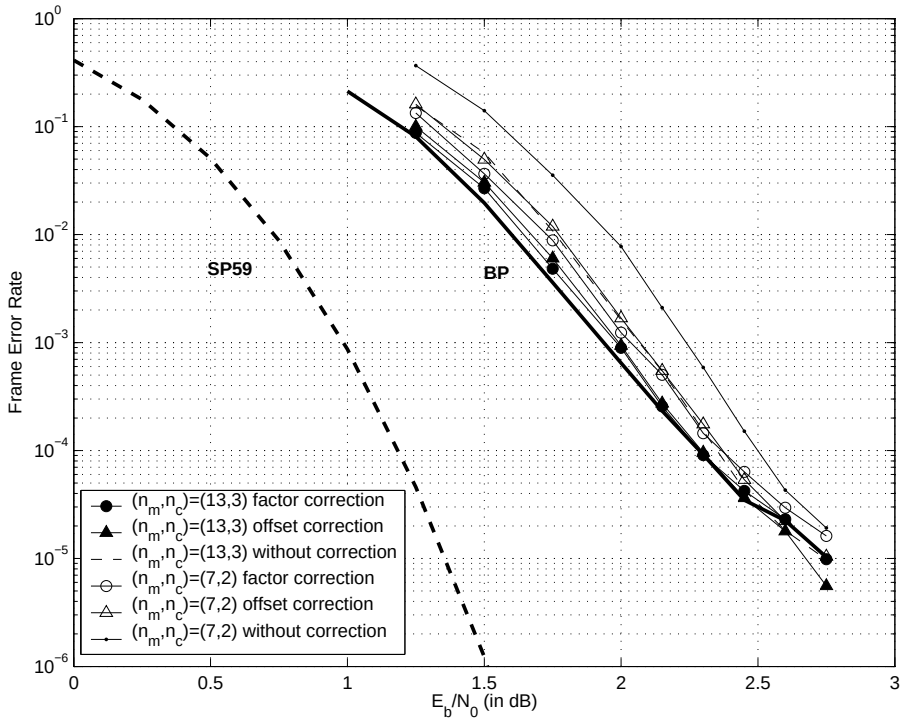

Figure 6: Comparison between BP and EMS decodings for a regular (2,4)-LDPC code over GF(64) ($R = 1/2$, $N = 142$ GF(64) symbols, $N_b = 852$ equivalent bits). 50 decoding iterations have been used for all decoders.

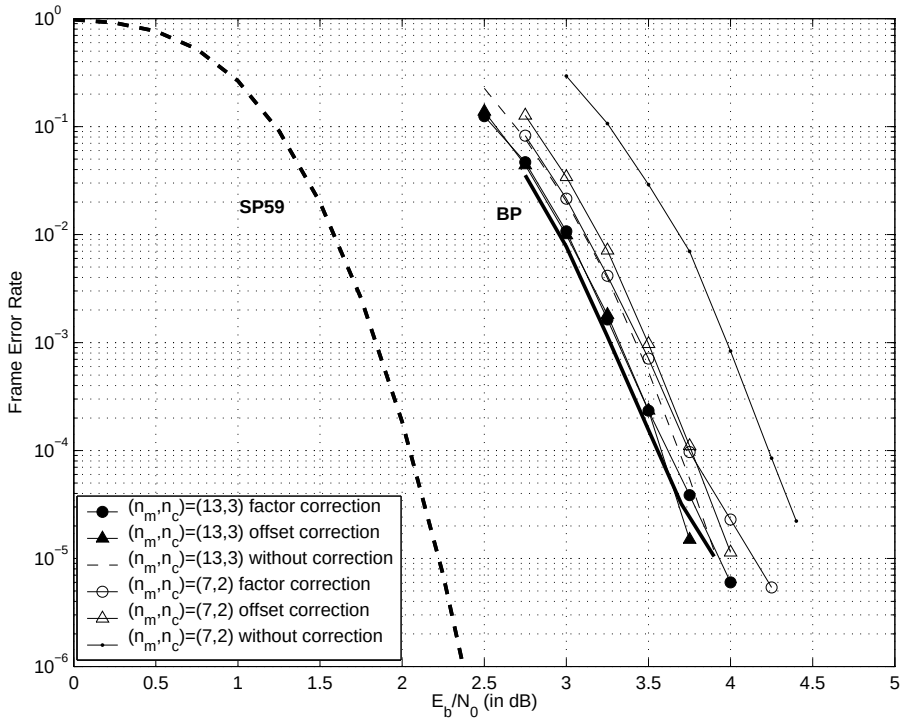

Figure 7: Comparison between BP and EMS decodings for a regular (2,8)-LDPC code over GF(128) ($R = 3/4$, $N = 84$ GF(128) symbols, $N_b = 588$ equivalent bits). 50 decoding iterations have been used for all decoders.

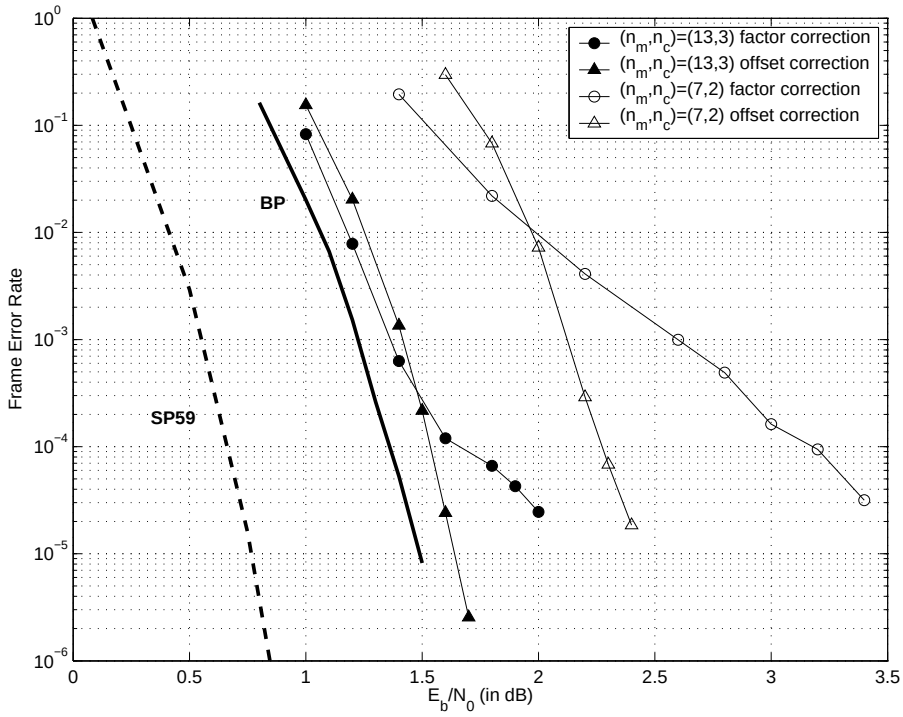

Figure 8: Comparison between BP and EMS decodings for a regular (2,4)-LDPC code over GF(256) ($R = 1/2$, $N = 376$ GF(256) symbols, $N_b = 3008$ equivalent bits). 50 decoding iterations have been used for all decoders.

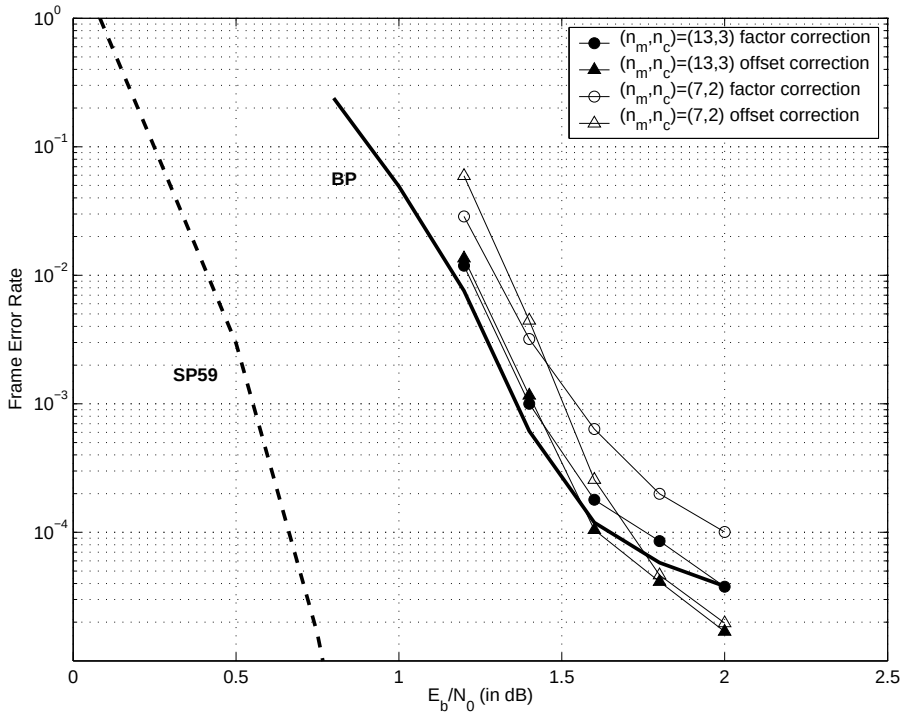

Figure 9: Comparison between BP and EMS decodings for a regular (2,4)-LDPC code over GF(64) ($R = 1/2$, $N = 500$ GF(64) symbols, $N_b = 3000$ equivalent bits). 50 decoding iterations have been used for all decoders.

List of Figures