An Efficient and Low-Complexity Iterative Reliability-Based Majority-Logic Decoding Algorithm for LDPC Codes

Qin Huang, Jingyu Kang, Li Zhang, Shu Lin and Khaled Abdel-Ghaffar
Department of Electrical and Computer Engineering
University of California, Davis
Davis, CA 95616
qinhuang, jykang, liszhang@ucdavis.edu; shulin, ghaffar@ece.ucdavis.edu

***Abstract*— This paper presents a reliability-based iterative majority-logic decoding algorithm for LDPC codes. This decoding algorithm is a binary message-passing algorithm and requires only binary logical operations and integer additions. Consequently, it can be implemented with simple combinational logic circuits. It either outperforms or performs just as well as the existing weighted bit-flipping or other reliability-based iterative decoding algorithms for LDPC codes in error performance with a faster rate of decoding convergence and less decoding complexity. Compared to the sum-product algorithm for LDPC codes, it offers effective trade-off between performance and decoding complexity. It is particularly effective for decoding LDPC codes constructed based on finite geometries and finite fields.**

I. Introduction

The ever-growing needs for cheaper, faster, and more reliable communication systems have forced many researchers to seek means to attain the ultimate limits on reliable communications. LDPC codes are currently the most promising coding technique to achieve Shannon capacity for a wide range of channels. These codes were first discovered by Gallager in 1962 [1] and then rediscovered in late 1990's [2]-[3]. Ever since their rediscovery, a great deal of research effort has been expended in design, construction, encoding, decoding, generalizations, performance analysis and applications of LDPC codes. Over the last 10 years, hundreds of papers have been published on these subjects, many classes of well performing LDPC codes have been constructed, and various algorithms for decoding these codes have been devised. The decoding algorithms that have been devised provide a *wide spectrum* of trade-offs between error performance, decoding complexity and decoding speed. Many LDPC codes have been chosen as the standard codes for various next generations of communication systems, such as wireless, optical, satellite, digital video broadcast, and 10G BASE-T Ethernet.

A *regular* LDPC code C [1] is given by the null space of a *sparse parity-check matrix* $\mathbf{H}$ over GF(2) that has the following structural properties: 1) each column has weight γ; and 2) each row has weight ρ. The code given by the null space of such a parity-check matrix $\mathbf{H}$ is called a (γ,ρ)-*regular* LDPC code. In almost all constructions of LDPC codes, the following additional structural property is further imposed on the parity-check matrix $\mathbf{H}$; 3) no two rows (or two columns) can have more than one place where they both have 1-components. Structural property-3 is a *constraint* on the rows and columns of $\mathbf{H}$ and is referred to as the *row-column (RC)-constraint*. This RC-constraint ensures that: 1) the Tanner graph [4] of the code given by the null space of $\mathbf{H}$ has a *girth* of at least 6; and 2) the minimum distance of the code is at least $\gamma + 1$ [5], [6]. If the columns and/or rows of $\mathbf{H}$ have *varying* weights, the null space of $\mathbf{H}$ gives an *irregular* LDPC code [7]. If $\mathbf{H}$ is an *array* of sparse *circulants* of the same size over GF(2), its null space gives a *quasi-cyclic (QC)*-LDPC code [5], [8]. If $\mathbf{H}$ is a single sparse circulant or a column of sparse circulants, the null space of $\mathbf{H}$ gives a *cyclic* LDPC code [5], [6].

Methods for decoding LDPC codes can be classified into three general categories: 1) *soft-decision decoding*; 2) *hard-decision decoding*; and 3) *reliability-based decoding*. Soft-decision decoding algorithms for LDPC codes are mostly *iterative* in nature and devised based on *belief propagation* [1], [9]. The most well known soft-decision iterative decoding algorithms based on belief propagation are the *sum-product algorithm (SPA)* [1], [3], [7], [10], [11] and its simplified versions. The SPA provides the *best* performance but requires the *largest* computational complexity among all the decoding algorithms devised for LDPC codes. The most well known simplified version of the SPA is the *min-sum* algorithm [10]. It requires less computational complexity than the SPA; however, it suffers some performance degradation compared to the SPA. Hard-decision decoding is the *simplest* in complexity among the three categories of decoding algorithms for LDPC codes; however, its simplicity results in a *significant* performance loss compared to the SPA. Well known hard-decision decoding algorithms for LDPC codes are the *one-step majority-logic-decoding (OSMLGD)* [5], [6] and the *bit-flipping (BF)* algorithms [1], [5], [6]. The BF-algorithm performs better than the OSMLGD-algorithm but requires a larger decoding complexity. A reliability-based decoding algorithm is devised by including some type of *reliability measures* of received symbols in a hard-decision decoding algorithm to improve

This research was supported by NASA under the Grant NNX07AK50G, NSF under the Grant CCF-0727478, and the gift grants from Intel and Northrop Grumman Space Technology.

its performance with some increase in decoding complexity. The most well known reliability-based decoding algorithms for LDPC codes are various *weighted-BF (WBF)* algorithms [5], [12]-[15]. WBF-algorithms provide efficient trade-offs between the high performance of the SPA and the simple decoding complexity of the hard-decision OSMLGD- and BF-algorithms. The best WBF-algorithm [15], in performance, performs *well within* 1 dB from the SPA.

This paper presents a *novel reliability-based iterative decoding algorithm* for LDPC codes that provides efficient trade-off between performance and decoding complexity. This algorithm is devised based on the simple concepts of the OSMLGD-algorithm. The algorithm includes a specific type of *soft reliability measure* of a received symbol in its *decoding function*. The reliability measure of each received symbol is improved through decoding iterations. This decoding algorithm either outperforms or performs just as well as the existing WBF-algorithms [12]-[15] and the newly proposed *differential binary message-passing decoding (DBMPD) algorithm* [16] in performance, but requires *less decoding complexity and has a faster rate of decoding convergence*. The proposed decoding algorithm is a binary message-passing algorithm and requires only logical operations and integer additions. As a result, they can be implemented with simple *combinational logic circuits*. Furthermore, the algorithm allows *parallel decoding* of received symbols to achieve a very high decoding speed.

II. A New View of the OSMLGD-Algorithm for LDPC Codes

Majority-logic decoding (MLGD) was first devised by Reed in 1954 [17] and later extended by Massey in 1963 [18]. Decoding of LDPC codes with the OSMLGD-algorithm was presented in [5]. In this section, we present the OSMLGD-algorithm for LDPC codes from a *new point of view*. We interpret the decoding of each received symbol in terms of its *intrinsic-information* and the *extrinsic-information* received from other received symbols participating in the syndrome-sums that are *orthogonal on* it. This interpretation leads us to develop a reliability-based iterative (RBI)-MLGD-algorithm to be presented in the next section. Even though the OSMLGD- and the RBI-MLGD-algorithms are devised for general LDPC codes, regular or irregular, we present them based on regular LDPC codes for simplicity.

Let C be a (γ,ρ)-LDPC code of length n given by an RC-constrained $m \times n$ matrix $\mathbf{H}$ over GF(2) with column and row weights γ and ρ, respectively. Let $\mathbf{v} = (v_0, v_1, ..., v_{n-1})$ be a codeword in C to be transmitted over the binary-input (BI) AWGN channel with two-sided power spectral density $N_0/2$. Assume transmission using BPSK signaling with unit energy per signal. Then the codeword $\mathbf{v}$ is mapped into a sequence of BPSK signals for transmission. This sequence of BPSK signals is commonly represented by a bipolar code sequence, $(2v_0 - 1, 2v_1 - 1, ..., 2v_{n-1} - 1)$, where the jth component $2v_j - 1 = +1$ for $v_j = 1$ and $2v_j - 1 = -1$ for $v_j = 0$. Suppose $\mathbf{v}$ is transmitted. Let $\mathbf{y} = (y_0, y_1, ..., y_{n-1})$ be the sequence of *samples* at the output of the channel receiver sampler. This sequence is commonly called a *soft-decision* received sequence. The samples of $\mathbf{y}$ are real numbers with $y_j = (2v_j - 1) + x_j$ for $0 \le j < n$, where x_j is a Gaussian random variable with zero-mean and variance $N_0/2$ [19].

For $0 \le j < n$, suppose each sample y_j of $\mathbf{y}$ is quantized into two levels and decoded *independently* based on the following hard-decision rule:

$$\begin{cases} z_j = 0, \text{for } y_j \le 0, \\ z_j = 1, \text{for } y_j > 0. \end{cases} \tag{1}$$

The above hard-decision of each received sample results in a binary sequence $\mathbf{z} = (z_0, z_1, ..., z_{n-1})$ which is referred to as the hard-decision received vector. The jth bit z_j of $\mathbf{z}$ is simply an estimate of the jth code bit v_j of the transmitted codeword $\mathbf{v}$. If $z_j = v_j$ for $0 \le j < n$, then $\mathbf{z} = \mathbf{v}$; otherwise, $\mathbf{z}$ contains one or more transmission errors. Decoding based on the hard-decision received vector $\mathbf{z}$ and the parity-check matrix $\mathbf{H}$ is referred to as hard-decision decoding.

Let $\mathbf{h}_0, \mathbf{h}_1, ..., \mathbf{h}_{m-1}$ denote the rows of $\mathbf{H}$, where the ith row $\mathbf{h}_i$ is an n-tuple over GF(2), $\mathbf{h}_i = (h_{i,0}, h_{i,1}, ..., h_{i,n-1})$, for $0 \le i < m$. An n-tuple $\mathbf{u}$ over GF(2) is a codeword in C *if and only if* $\mathbf{u}\mathbf{H}^T = \mathbf{0}$ (a zero m-tuple), i.e., the inner product $\mathbf{u} \cdot \mathbf{h}_i = 0$ for $0 \le i < m$. Since $\mathbf{v} = (v_0, v_1, ..., v_{n-1})$ is a codeword in C, $\mathbf{v}\mathbf{H}^T = \mathbf{0}$. The condition $\mathbf{v}\mathbf{H}^T = \mathbf{0}$ gives the following m *constraints* on the code bits of $\mathbf{v}$:

$$\mathbf{v} \cdot \mathbf{h}_i = v_0 h_{i,0} + v_1 h_{i,1} + \cdots + v_{n-1} h_{i,n-1} = 0, \tag{2}$$

for $0 \le i < m$. The above m linear sums of code bits are called *parity-check-sums*. The constraint given by (2) is called the *zero-parity-check-sum (ZPCS) constraint*. For $0 \le j < n$, if $h_{i,j} = 1$, then the jth code bit v_j participates (or is contained) in the ith parity-check-sum given by (2). In this case, we say that the ith row $\mathbf{h}_i$ of $\mathbf{H}$ *checks on* the jth code bit v_j of $\mathbf{v}$. Since $\mathbf{H}$ has constant column weight γ, there are γ rows of $\mathbf{H}$ that check on v_j for $0 \le j < n$. This implies that there are γ ZPCS's containing the code bit v_j.

To decode the hard-decision received vector $\mathbf{z}$, the first step is to compute its m-tuple syndrome, $\mathbf{s} = (s_0, s_1, ..., s_{m-1}) = \mathbf{z}\mathbf{H}^T$, where

$$s_i = \mathbf{z} \cdot \mathbf{h}_i = z_0 h_{i,0} + z_1 h_{i,1} + \cdots + z_{n-1} h_{i,n-1}, \tag{3}$$

for $0 \le i < m$, which is called a *syndrome-sum* of $\mathbf{z}$. If $\mathbf{s} = \mathbf{0}$, then the received bits in $\mathbf{z}$ satisfy all the m ZPCS-constraints given by (2) and hence $\mathbf{z}$ is a codeword. A syndrome-sum s_i of the received vector $\mathbf{z}$ that is not equal to zero is referred to as a *parity-check failure*. A syndrome-sum s_i that contains a received bit z_j is said to check on z_j.

For $0 \le i < m$ and $0 \le j < n$, we define the following two *index sets*:

$$N_i = \{j : 0 \le j < n, h_{i,j} = 1\}, \tag{4}$$

$$M_j = \{i : 0 \le i < m, h_{i,j} = 1\}. \tag{5}$$

Since $\mathbf{H}$ has constant column weight γ and constant row weight ρ, $|M_j| = \gamma$ for $0 \le j < n$ and $|N_i| = \rho$ for $0 \le i < m$. It follows from the definition of N_i that the ith syndrome-sum s_i of the syndrome $\mathbf{s}$ of a hard-decision received vector $\mathbf{z}$ given by (3) can be expressed in terms of the indices in N_i as follows:

$$s_i = \sum_{j \in N_i} z_j h_{i,j}, \tag{6}$$

for $0 \leq i < m$. For $0 \leq j < n$, define the following set of rows of $\mathbf{H}$:

$$\Omega_j = \{\mathbf{h}_i : i \in M_j\}. \tag{7}$$

It follows from the RC-constraint on the rows of $\mathbf{H}$ that Ω_j has the following structural properties: 1) every row in Ω_j checks on the code bit v_j; and 2) any code bit other than v_j is checked by at most one row in Ω_j. The rows in Ω_j are said to be *orthogonal on* the code bit v_j. For $0 \leq j < n$, define the following subset of syndrome-sums of $\mathbf{z}$,

$$S_j = \{s_i = \mathbf{z} \cdot \mathbf{h}_i : \mathbf{h}_i \in \Omega_j\}. \tag{8}$$

Due to the orthogonality property of Ω_j, the jth received bit z_j of $\mathbf{z}$ is contained in every syndrome-sum in S_j and any received bit other than z_j is contained in at most one syndrome-sum in S_j. Consequently, each syndrome-sum s_i in S_j can be expressed as follows:

$$s_i = z_j + \sum_{l \in N_i \setminus j} z_l h_{i,l}, \tag{9}$$

for $i \in M_j$. The syndrome-sum s_i consists of two parts, the received bit z_j itself and the contribution (the summation on the right side of (9)) from the other $\rho - 1$ received bits checked by $\mathbf{h}_i$. From the definitions of Ω_j and S_j and (9), it is clear that the received bit z_j is contained *only in* the syndrome-sums in S_j, *not in any other* syndrome-sum of $\mathbf{z}$ (or $\mathbf{s}$). The γ syndrome-sums in S_j are said to be *orthogonal on* z_j and are called *orthogonal syndrome-sums* on z_j. Therefore, the syndrome-sums in S_j contains all the information about the received bit z_j. This information will be used to decode z_j in the OSMLGD-algorithm and the RBI-MLGD-algorithm to be developed.

Let $\sigma_{i,j}$ denote the sum part of s_i given by (9), i.e.,

$$\sigma_{i,j} = \sum_{l \in N_i \setminus j} z_l h_{i,l}. \tag{10}$$

Then for $i \in M_j$,

$$s_i = z_j + \sigma_{i,j}. \tag{11}$$

For $0 \leq j < n$, define the following set:

$$\xi_j = \{\sigma_{i,j} : i \in M_j\}. \tag{12}$$

Decoding the received bit z_j based on the syndrome-sums in S_j orthogonal on it, the sums in ξ_j represent the extrinsic-information contributed by all the received bits, excluding z_j, that participate in the syndrome-sums in S_j. We call the sums in ξ_j the *extrinsic-information-sums orthogonal on* z_j.

An extrinsic-information-sum $\sigma_{i,j}$ in ξ_j is said to be *error-free* if it does not contain any erroneous received bits. From (11), we see that: 1) if $s_i = 0$ and $\sigma_{i,j}$ is error-free, then z_j must be error-free; 2) if $s_i = 1$ and $\sigma_{i,j}$ is error-free, then z_j is erroneous and its value must be changed from z_j to its *one's complement*, $1 + z_j$ (modulo-2 addition), to make the syndrome-sum s_i equal to zero. Intuitively, it is clear that the larger the number of zero syndrome-sums in S_j and the larger the number of error-free extrinsic-information-sums in ξ_j, the more reliably we can determine whether z_j is error-free or erroneous. Therefore, the number of error-free intrinsic-information-sums in ξ_j indicates to *what extent* the jth received bit z_j should remain the same or should be changed.

Now we develop the OSMLGD-algorithm for regular LDPC codes based on the concepts presented above. Consider a (γ,ρ)-regular LDPC code given by the null space of an $m \times n$ RC-constrained parity-check matrix $\mathbf{H}$ with column and row weights γ and ρ, respectively. Suppose the received vector $\mathbf{z}$ contains $\lfloor \gamma/2 \rfloor$ or fewer erroneous bits. There are two cases to be considered: 1) z_j is error-free and 2) z_j is erroneous. If z_j is error-free, then the erroneous bits in $\mathbf{z}$ can distribute among *at most* $\lfloor \gamma/2 \rfloor$ extrinsic-information-sums in ξ_j. In this case, there are at least $\gamma - \lfloor \gamma/2 \rfloor$ (*half or more than half of* γ) error-free extrinsic-information-sums orthogonal on z_j in ξ_j. Consequently, there are at least $\gamma - \lfloor \gamma/2 \rfloor$ syndrome-sums in S_j containing only error-free received bits and hence they must be equal to "0" (i.e., satisfying the ZPCS-constraint given by (2)). If z_j is an erroneously received bit, then the other erroneous bits in $\mathbf{z}$ can distribute among at most $\lfloor \gamma/2 \rfloor - 1$ extrinsic-information-sums in ξ_j. As a result, *at least* $\gamma - \lfloor \gamma/2 \rfloor + 1$ (*more than half* of γ) of extrinsic-information-sums in ξ_j contain only error-free received bits. This implies that at least $\gamma - \lfloor \gamma/2 \rfloor + 1$ syndrome-sums in S_j contain only one erroneously received bit which is z_j. Hence, these syndrome-sums fail the ZPCS-constraint given by (2) and must be equal to "1". In this case, z_j must be changed to its one's complement, $1 + z_j$, to make these failed syndrome-sums equal to zero. Based on the facts developed above, the rule of OSMLGD is given as follows: *For $0 \leq j < n$, the received bit z_j at the jth location of $\mathbf{z}$ is changed to $1 + z_j$ if a majority of the syndrome-sums orthogonal on z_j assumes the value "1"; otherwise, z_j remains unchanged.*

Based on the above decoding rule, correct decoding of a received vector $\mathbf{z}$ is *guaranteed* if the number of erroneous bits in $\mathbf{z}$ is $\lfloor \gamma/2 \rfloor$ or less. However, correct decoding is *not guaranteed* if $\mathbf{z}$ contains more than $\lfloor \gamma/2 \rfloor$ erroneous bits. This can be shown easily. Suppose there are $\lfloor \gamma/2 \rfloor + 1$ erroneous received bits in $\mathbf{z}$. If these erroneous bits distribute in $\lfloor \gamma/2 \rfloor + 1$ extrinsic-information-sums in ξ_j, one in each, then there are $\gamma - \lfloor \gamma/2 \rfloor - 1$ (less than half of γ) error-free extrinsic-information-sums in ξ_j. In this case, the above decoding rule will result in incorrect decoding of z_j. Therefore, an LDPC code C given by the null space of an RC-constrained parity-check matrix $\mathbf{H}$ with constant column weight γ is capable of correcting *any combination* of $\lfloor \gamma/2 \rfloor$ or fewer erroneously received bits with the OSMLGD-algorithm.

For $0 \leq j < n$, define

$$\Gamma_j = 2 \sum_{i \in M_j} s_i - \gamma = \sum_{i \in M_j} (2s_i - 1). \tag{13}$$

It is easy to prove that $\Gamma_j > 0$ *if and only if* a majority (more than $\lfloor \gamma/2 \rfloor$) of the syndrome-sums orthogonal on the received bit z_j assumes the value "1". Using Γ_j, the OSMLGD rule can be restated as follows: *for $0 \leq j < n$, the jth received bit*

z_j is changed to its complement $1 + z_j$ if $\Gamma_j > 0$; otherwise z_j remains unchanged.

With Γ_j as the *decoding function* for the jth received bit of a received vector $\mathbf{z}$, the OSMLGD-algorithm for an (γ,ρ)-regular LDPC code can be carried out with the following steps:

1. Compute the syndrome $\mathbf{s}$ of the received vector $\mathbf{z} = (z_0, z_1, ..., z_{n-1})$. If $\mathbf{s} = \mathbf{0}$, stop decoding and output $\mathbf{z}$ as the decoded codeword; otherwise, go to Step 2.
2. For $0 \leq j < n$, compute Γ_j and go to Step 3.
3. For $0 \leq j < n$, if $\Gamma_j > 0$, decode z_j into its one's complement $z_j^* = 1 + z_j$; otherwise z_j remains unchanged and set $z_j^* = z_j$. Form a new received vector $\mathbf{z}^* = (z_0^*, z_1^*, ..., z_{n-1}^*)$. Compute the syndrome $\mathbf{s}^* = \mathbf{z}^*\mathbf{H}^T$ of $\mathbf{z}^*$. If $\mathbf{s}^* = \mathbf{0}$, output $\mathbf{z}^*$ as the decoded codeword; otherwise, declare a *decoding failure*. Stop decoding.

The above decoding algorithm stops with *just one pass*. For this reason, it is referred to as the OSMLGD-algorithm. Decoding a received bit is solely based on the information provided by the syndrome-sums orthogonal on it. For $0 \leq j < n$, the value of the decoding function Γ_j of the jth received bit z_j is in the integer range $[-\gamma, +\gamma]$. The integer 0 is called the *decoding threshold*. The reliability of the decoding of z_j depends on both the range $[-\gamma, +\gamma]$ of Γ_j and the gap between the value of Γ_j and the decoding threshold 0. The larger the range of Γ_j and the larger the gap between the value of Γ_j and the decoding threshold 0, the more reliable the decoding of z_j is. Therefore, the OSMLGD-algorithm is effective only when the column weight γ of the RC-constrained parity-check matrix of an LDPC code is relatively large.

The OSMLGD-algorithm basically requires logical operations and simple integer additions which can be carried out in binary form using a binary adder. Therefore, an OSMLG-decoder can be implemented with simple combinational logic circuits and a buffer-register to store the received vector $\mathbf{z}$. All the received bits can be decoded in parallel at the same time to achieve a very fast decoding speed.

Even though the OSMLGD-algorithm can be applied to decode any LDPC code whose parity-check matrix satisfies the RC-constraint, it is particularly effective for decoding *finite-geometry* (FG)-LDPC codes [5], [6] and *finite-field* (FF)-LDPC codes [20]. For a FG- or a FF-LDPC code, a large number of syndrome-sums orthogonal on a received bit can be formed for decoding. For example, consider the cyclic LDPC code constructed based on the m-dimensional Euclidean geometry EG(m,q) over the field GF(q) [5], [6]. The parity-check matrix of this EG-LDPC code satisfies the RC-constraint and has column weight $\gamma = (q^m - 1)/(q - 1) - 1$.

III. A Reliability-Based Iterative MLGD-Algorithm for LDPC Codes

The OSMLGD-algorithm presented in the last section is a *one-pass* decoding algorithm. Decoding of a received bit is mainly based on the extrinsic-information-sums and the syndrome-sums orthogonal on it. There is basically no intrinsic-information (or reliability measure) provided to a received bit for decoding, except the hard-decision information of the received bit itself given by (1). The one-pass OSMLGD-algorithm can be improved by: 1) including a *certain type of intrinsic-information* in the decoding function of each received bit, 2) making the decoding algorithm iterative, and 3) using simple extrinsic-information to improve the reliability measure of a received bit through decoding iterations.

Again, consider an LDPC code C given by the null space of an RC-constrained $m \times n$ parity-check matrix $\mathbf{H}$ over GF(2) with column and row weights γ and ρ, respectively. Suppose a codeword $\mathbf{v} = (v_0, v_1, ..., v_{n-1})$ of C is transmitted over the BI-AWGN channel with BPSK signaling. Let $\mathbf{y} = (y_0, y_1, ..., y_{n-1})$ be the corresponding received sequence of unquantized samples at the output of the receiver sampler. With the hard-decision rule given by (1), the magnitude $|y_j|$ of the jth sample y_j of $\mathbf{y}$ can be used as a reliability measure of the hard-decision decoded bit z_j, since the magnitude of the log-likelihood ratio, $|\log Pr(y_j|v_j = 1)/\log Pr(y_j|v_j = 0)|$, associated with the hard-decision given by (1) is proportional to $|y_j|$. The larger the $|y_j|$, the more reliable the hard-decision z_j is.

Suppose the samples of $\mathbf{y}$ are symmetrically clipped and uniformly quantized into 2^b intervals, symmetric with respect to the origin [19]. Each interval has a length Δ and is represented by b bits. For $0 \leq j < n$, let q_j denote the quantized value of the sample y_j which is an integer representation of one of the $2^b - 1$ quantization intervals. Therefore, the range of q_j is $[-(2^{b-1} - 1), +(2^{b-1} - 1)]$. If the magnitude $|y_j|$ exceeds the quantization range $(2^{b-1} - 1)\Delta$, then set $|q_j| = 2^{b-1} - 1$. With this quantization, the magnitude $|q_j|$ of the quantized sample q_j gives a *soft measure* of the reliability of the hard-decision received bit z_j. In the following, we develop an RBI-MLGD-algorithm using q_j as the *initial* reliability measure of a received bit z_j at the beginning of decoding process. This reliability measure will be improved through the decoding iterations. Since a soft reliability measure is used in decoding a received bit, the decoding algorithm to be developed is referred to as a *soft RBI (SRBI)-MLGD-algorithm*.

To present the SRBI-MLGD-algorithm, we need to define some notations. Let k_{max} be the maximum number of iterations to be performed. For $0 \leq k \leq k_{max}$, let: 1) $\mathbf{z}^{(k)} = (z_0^{(k)}, z_1^{(k)}, ..., z_{n-1}^{(k)})$ be the received vector generated in the kth decoding iteration; 2) $\mathbf{s}^{(k)} = (s_0^{(k)}, s_1^{(k)}, ..., s_{m-1}^{(k)}) = \mathbf{z}^{(k)}\mathbf{H}^T$ be the syndrome of $\mathbf{z}^{(k)}$; 3) $S_j^{(k)}$ be the set of syndrome-sums orthogonal on the jth received bit $z_j^{(k)}$ of $\mathbf{z}^{(k)}$; 4) $\sigma_{i,j}^{(k)}$ be the extrinsic-information contributed to $z_j^{(k)}$ by the other received bits participating in the syndrome-sum $s_i^{(k)}$ in $S_j^{(k)}$ orthogonal on $z_j^{(k)}$; and (5) $R_j^{(k)}$ be the reliability measure of the jth received bit $z_j^{(k)}$ of $\mathbf{z}^{(k)}$.

From (11) and (13), we see that for $1 \leq j < n$, if we remove the jth bit $z_j^{(k)}$ of the received vector $\mathbf{z}^{(k)}$ from the syndrome-sum $s_i^{(k)}$ containing it, we obtain the following integer sum:

$$E_j^{(k)} = \sum_{i \in M_j} (2\sigma_{i,j}^{(k)} - 1), \tag{14}$$

where $2\sigma_{i,j}^{(k)} - 1 = -1$ for $\sigma_{i,j}^{(k)} = 0$ and $2\sigma_{i,j}^{(k)} - 1 = +1$

for $\sigma_{i,j}^{(k)} = 1$. For $0 \leq j < n$, $E_j^{(k)}$ is simply the *total extrinsic-information* contributed to the received bit $z_j^{(k)}$ by all the received bits, excluding $z_j^{(k)}$, that participate in the syndrome-sums in $S_j^{(k)}$ orthogonal on $z_j^{(k)}$. The value of $E_j^{(k)}$ is an integer ranging from $-\gamma$ to $+\gamma$. Then, for $0 \leq k < k_{max}$ and $0 \leq j < n$, the reliability of the jth received bit $\mathbf{z}^{(k+1)}$ at the $(k+1)$th iteration is given by

$$R_j^{(k+1)} = R_j^{(k)} + E_j^{(k)}. \tag{15}$$

For $k = 0$ and $0 \leq j < n$, the *initial reliability* $R_j^{(0)}$ of the received bit $z_j^{(0)} = z_j$ is set to q_j.

With the concepts and notations defined above, the SRBI-MGLD-algorithm for an LDPC code can be formulated as follows:

Initialization: Set $k = 0$, $\mathbf{z}^{(0)} = \mathbf{z}$ and the maximum number of iterations to k_{max}. For $0 \leq j < n$, set $R_j^{(0)} = q_j$.

1. Compute the syndrome $\mathbf{s}^{(k)} = \mathbf{z}^{(k)}\mathbf{H}^T$ of $\mathbf{z}^{(k)}$. If $\mathbf{s}^{(k)} = \mathbf{0}$, stop decoding and output $\mathbf{z}^{(k)}$ as the decoded codeword; otherwise, go to Step 2.
2. If $k = k_{max}$, stop decoding and declare a decoding failure; otherwise, go to Step 3.
3. Update the reliability measures of all the received bits of $\mathbf{z}^{(k)}$, $R_j^{(k+1)} = R_j^{(k)} + E_j^{(k)}$, for $0 \leq j < n$. Store the updated reliability measures of the received bits in $\mathbf{z}^{(k)}$. Go to Step 4.
4. $k \leftarrow k + 1$. For $0 \leq j < n$, make the following hard-decision: 1) $z_j^{(k)} = 0, \text{if } R_j^{(k)} \leq 0$; and 2) $z_j^{(k)} = 1, \text{if } R_j^{(k)} > 0$. Form a new received vector $\mathbf{z}^{(k)} = (z_0^{(k)}, z_1^{(k)}, ..., z_{n-1}^{(k)})$. Go to Step 1.

When updating the reliability measure of a received bit $\mathbf{z}^{(k)}$ at the Step 2 in the kth iteration, if the magnitude $|R_j^{(k)} + E_j^{(k)}|$ of the sum $R_j^{(k)} + E_j^{(k)}$ is greater than quantization range $(2^{(b-1)} - 1)\Delta$, we set the magnitude $|R_j^{(k+1)}|$ of the reliability measure $R_j^{(k+1)}$ of the new received bit $z_j^{(k+1)}$ to $(2^{(b-1)} - 1)$ (the maximum reliability measure set for a received bit by quantization).

Since the initial reliability measure of a received bit is an integer and its extrinsic-information in each iteration is an integer, the reliability measure of each modified received bit in each iteration is also an integer. Consequently, to carry out the above SRBI-MLGD-algorithm, only logical operations and integer additions are required.

We can interpret the SRBI-MLGD-algorithm in terms of the Tanner graph of an LDPC code given by the null space of an $m \times n$ parity-check matrix. The Tanner graph has n variable nodes and m check nodes. Syndrome of a received vector at Step 1 is computed at the check nodes. Reliability updating and hard-decisions of the received bits of a received vector at Steps 3 and 4 are carried out at the variable nodes. Also, the reliability measures of the received bits are stored at the variable nodes. Binary messages, received bits and syndrome-sums, are passed between variable and check nodes through the edges connecting the two types of nodes. Therefore, the SRBI-MLGD-algorithm is a binary message-passing algorithm.

The computational complexity of the above SRBI-MLGD-algorithm can be easily analyzed. From (6), we see that to compute the syndrome of a received vector at the first step, $(\rho - 1)m$ XOR (EXCLUSIVE-OR) operations (or modulo-2 additions) are required. From (11), (14) and (15), we see that to carry out Step 3 of the algorithm, we need: 1) γn XOR operations to form the extrinsic-information-sums $\sigma_{i,j}^{(k)} = s_i^{(k)} + z_j^{(k)}$ for $i \in M_j$ and $0 \leq j < n$; 2) $(\gamma - 1)n$ integer additions to compute extrinsic-information $E_j^{(k)}$ for $0 \leq j < n$; and 3) n integer additions to update the reliability measures of the n received bits. Step 4 requires n tests of the signs of the updated reliability measures of the n received bits of a received vector. Testing the sign of an integer is also a logical operation. Note that $\rho m = \gamma n$, which is the total number of 1-entries in the parity-check matrix $\mathbf{H}$. Let $\delta = \rho m = \gamma n$. Then to carry out one iteration of the SRBI-MLGD-algorithm, a total of $2\delta + n - m$ logical operations and a total of δ integer additions are required. Besides the computational complexity, the SRBI-MLGD-algorithm also requires memory to store the reliability measures of the n received bits of a received vector. Since each reliability measure q_j or its updated value is represented by b bits, a memory of bn units is needed to store the reliability measures of the n received bits of a received vector. The size of the memory required to store the reliability measures of the received bits depends on the number of quantization levels. Addition of two integers represented by two binary b-tuples can be accomplished with a b-bit binary adder using logic gates. Consequently, the SRBI-MLGD-algorithm can be implemented with combinational logic circuits to compute m syndrome-sums and n reliability measures of the received bits in each decoding iteration. To store the reliability measures of n received bits, bn flip-flops can be used.

The SRBI-MLGD-algorithm outperforms all the well known WBF-algorithms [12]-[15], and the newly proposed DBMPD-algorithm [16] in terms of error performance, rate of decoding convergence, computational complexity and size of memory all together as the measure of the effectiveness of a decoding algorithm. A significant feature of the proposed SRBI-MLGD-algorithm over the WBF-algorithms is that the received bits can be decoded in parallel which is important in high-speed communication systems. The DBMPD-algorithm also allows parallel decoding of the bits of a received vector. Among the various well-known WBF-algorithms for decoding LDPC codes, the best one in terms of error performance is the algorithm proposed by Jiang, Zhao, Shi and Chen in [15], called the *JZSC-WBF-algorithm*, and the one that performs close to the JZSC-BF-algorithm but requires less computational complexity is the algorithm proposed by Zhang and Fossorier in [12], called the *ZF-WBF-algorithm*. Both the JZSC-WBF- and the ZF-WBF-algorithm flip one bit at a time in decoding and they converge slowly. The DBMPD-algorithm performs as well as the ZF-WBF-algorithm and converges faster.

Tables I and II give the computational complexities and the memory requirements of the SRBI-MLGD-algorithm, the SPA, the JZSC-WBF-algorithm, the ZF-WBF-algorithm, the

DBMPD-algorithm and the simple BF-algorithm given in [5]. We see that the decoding complexity (computational complexity plus memory requirement) of the SRBI-MLGD-algorithm is significantly less than that of the JZSC-WBF-, ZF-WBF-, and DBMPD-algorithms and is much less than that of the SPA. It is about the same order as that of the simple BF-algorithm given in [5].

The performance of the SRBI-MLD-algorithm depends on the range of the reliability measure of a received bit. From (15), we see that range of the reliability measure of a received bit depends on the number of quantization levels, the length of quantization interval and the range of the extrinsic-information received from the other received bits. The larger the range of the reliability measure of a received bit, the better the performance of the algorithm can be achieved. The number of quantization levels and the length of quantization interval can be determined by simulation for a given code to achieve good error performance. The range $[-\gamma, +\gamma]$ of the extrinsic-information is determined by the column weight γ of the RC-constrained parity-check matrix of an LDPC code. The larger the range of the extrinsic-information, the more the extrinsic-intrinsic information a received bit can receive from other received bits. RC-constrained parity-check matrices with relatively large column weights can be constructed based on finite geometries [5] and finite fields [20]. However, for computer constructed random codes, it is quite difficult to construct RC-constrained parity-check matrices with large column weights.

The SRBI-MGLD-algorithm becomes the OSMLGD-algorithm if it is executed with just one iteration and the initial reliability measure $R_j^{(0)}$ of the jth received bit $z_j^{(0)} = z_j$ with $0 \leq j < n$ is replaced with $2\gamma z_j^{(0)}$.

IV. Examples and Experimental Results

In the following, we will use three LDPC codes to demonstrate the effectiveness of the SRBI-MLGD-algorithm. One code is constructed based on an Euclidean geometry and the other two codes are constructed based on finite fields.

Example 1: Consider the (4095,3367) cyclic EG-LDPC code C constructed based on the two-dimensional Euclidean geometry EG(2,2^6) over GF(2^6) [5], [6]. The parity-check matrix $\mathbf{H}$ of this code consists of a single 4095×4095 circulant with both column and row weights 64, which is formed by the incidence vectors of the lines of EG(2,2^6) not passing through the origin. Since the parity-check matrix of this code satisfies the RC-constraint, 64 syndrome-sums orthogonal on any received bit can be formed. The code has a minimum distance of exactly 65 [6]. With the OSMLGD-algorithm, the (4095,3367) EG-LDPC code is capable of correcting 32 or fewer errors. The bit-error performances of this code over the BI-AWGN channel decoded using SRBI-MLGD-algorithm and the SPA with 50 iterations are shown in Figure 1. For the SRBI-MLGD-algorithm, we use 8-bit uniform quantization with 255 levels and an interval length $\Delta = 0.015$, which is obtained by simulations. The range of the extrinsic-information for each received bit is $[-64, +64]$. At a BER of 10^{-6}, the SRBI-MLGD-algorithm performs only 0.65 dB away from the SPA with a tremendous reduction of computational complexity (see Table I).

We also decode the (4095,3367) EG-LDPC code with the JZSC-WBF-, the ZF-WBF- and the DBMPD-algorithms [16]. For decoding the code with the JZSC-WBF- and ZF-WBF-algorithms, we use both 50 and 100 iterations. For the DBMPD-algorithm, we also use 8-bit uniform quantization with 255 levels and an interval length $\Delta = 0.015$. From Figure 1, we see that the SRBI-MLGD-algorithm with 50 iterations significantly outperforms both the JZSC-WBF- and ZF-WBF-algorithms. With 50 iterations, the performance curves of the JZSC-WBF- and the ZF-WBF-algorithms overlap each other. The SRBI-MLGD-algorithm with 50 iterations performs equally as well as the JZSC-WBF- and ZF-WBF-algorithms with 100 iterations. Figure 1 also includes the performance of SRBI-MLGD-algorithm with 5 iterations. We see that the gap between 5 and 50 iterations is only about 0.2 dB. With only 5 iterations, the SRBI-MLGD still significantly outperforms the JZSC-WBF and the ZF-WBF-algorithms with 50 iterations. Figure 2 shows the average numbers of iterations required by the three decoding algorithms. We see that the SRBI-MLGD-algorithm converges much faster than both the JZSC-WBF- and ZF-WBF-algorithms. Furthermore, from Table I, we see that the SRBI-MLGD-algorithm requires less computational complexity than both the JZSC-WBF- and the ZF-WBF-algorithms. Also from Figure 1, we see that the SRBI-MLGD- and DBMPD-algorithms perform identically; however, the DBMPD-algorithm requires much larger computational complexity and memory than the SRBI-MLGD-algorithm as shown in Table I. Also included in Figure 1 is the bit-error performance of the (4095,3367) EG-LDPC code decoded with the OSMLGD-algorithm. We see that the SRBI-MLD-algorithm has 1.5 dB coding gain over the OSMLGD-algorithm. $\triangle\triangle$

Example 2: Based on GF(2^5), it is possible to construct an RC-constrained 31×31 array $\mathbf{H}$ of 31×31 circulant permutation and zero matrices over GF(2) with 31 zero matrices lying on the main diagonal of the array using the first method given in [20]. $\mathbf{H}$ is a 961×961 matrix with both column and row weights 30. The null space of $\mathbf{H}$ gives a (961,721) (30,30)-regular QC-LDPC code. The performances of this code decoded with the SPA, the SRBI-MLGD-, the JZSC-WBF-, the ZF-WBF- and the DBMPD-algorithms are shown in Figure 3. For the SRBI-MLGD- and DBMPD-algorithms, we use 8-bit uniform quantization with 255 levels and an interval length $\Delta = 0.025$. All the algorithms are carried out with 30 iterations. We see that at a BER of 10^{-6}, the SRBI-MLGD-algorithm performs only 0.5 dB away form the SPA. The SRBI-MLGD-algorithm outperforms both the JZSC-WBF- and ZF-WBF-algorithms. From Figure 3, we also see that the SRBI-MLGD-algorithm performs as well as the DBMPD-algorithm. However, decoding of the (961,721) QC-LDPC code with the DBMPD-algorithm requires larger decoding complexity than the SRBI-MLGD-algorithm (see Tables I and II). Figure 4 shows the average numbers of iterations required by the SRBI-MLGD-, the JZSC-WBF- and the ZF-WBF-algorithms. We see that the SRBI-MLGD-algorithm converges faster than the two WBF-algorithms.$\triangle\triangle$

The parity-check matrices of the two LDPC codes used in Examples 1-2 have large column weights. In the next example, we will consider a code whose parity-check matrix has a relatively small column weight.

Example 3: Using the prime field GF(73) and the third method given [20], we can construct an RC-constrained 73×73 array $\mathbf{H}$ of 73×73 circulant permutation matrices over GF(2). Take an 8×72 subarray $\mathbf{H}(8, 72)$ of $\mathbf{H}$, say taking the first 8 rows of $\mathbf{H}$ and removing the last column. This subarray is an RC-constrained 584×5256 matrix with column and row weights 8 and 72, respectively. The null space of this matrix gives a (8,72)-regular (5256,4679) QC-LDPC code. The bit-error performances of this code decoded with the SRBI-MLGD-, the JZSC-WBF-, the DBMPD-algorithms and the SPA are shown in Figure 5. Except for the JZSC-WBF-algorithm, all the algorithms are carried out with 50 iterations. The JZSC-WBF-algorithm is carried out with 100 iterations. For the SRBI-MLGD- and the DBMPD-algorithms, we again use 8-bit uniform quantization with 255 levels and an interval length $\Delta = 0.03$. From Figure 5, we see that at a BER of 10^{-6}, the SRBI-MLGD-algorithm performs 0.75 dB from the SPA, respectively. The SRBI-MLGD-algorithm with 50 iterations slightly outperforms the JZSC-WBF-algorithm with 100 iterations below the BER of 10^{-5}. The SRBI-MLGD- and the DBMPD-algorithms perform roughly the same; however, the SRBI-MLGD requires less computational complexity and memory than the DBMPD-algorithm.

Figure 6 shows the bit-error performance of the (5256,4679) QC-LDPC code decoded using the SRBI-MLGD-algorithm with 5, 10 and 50 iterations, respectively. We see that decoding converges very fast. Also we see that even with 10 iterations, the SRBI-MLGD-algorithm outperforms the JZSC-WBF-algorithm with 100 iterations below the BER of 10^{-7}. $\triangle\triangle$

V. Remarks and Conclusion

In this paper, we have presented a binary message-passing algorithm for LDPC codes. The algorithm requires only logical operations and integer additions and can be implemented with simple combinational logic circuits using flip-flops to store reliability measures of the received bits. Since the algorithm allows decoding all the received bits in parallel, it can be implemented to achieve very high speed. This is an advantage over the WBF-algorithms.

Besides the low decoding complexity of the proposed decoding algorithm for LDPC codes, experimental results show that it either outperforms or performs just as well as the existing WBF- or DBMPD-algorithms for LDPC codes in performance with a faster rate of decoding convergence. Compared to the SPA for LDPC codes, it offers superior trade-off between performance and decoding complexity. The proposed algorithm is particularly effective for decoding finite-geometry LDPC codes and LDPC codes constructed based on finite fields because the parity-check matrices of these codes not only satisfy the RC-constraint but they also have large row redundancy and column weights which provide large ranges of reliability measures (or extrinsic-information) of received bits.

The proposed algorithm can be applied to decode irregular LDPC codes. For an irregular LDPC code, its parity-check matrix has multiple column weights. As a result, the ranges of the decoding functions of the received bits are not uniform anymore, some with small ranges and some with large ranges. If there are columns with very small weights, say 1, 2 and 3, the proposed decoding algorithm becomes ineffective and may produce a high error-floor (this is also true for other reliability-based decoding algorithms, even for the SPA).

References

[1] R.G. Gallager, "Low density parity check codes," *IRE Trans. Inform. Theory*, IT-8, pp. 21-28, Jan. 1962.

[2] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity-check codes," *Electro. Lett.*, vol. 32, no. 18, pp.1645-1646, August 1996.

[3] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp.399-432, Mar. 1999.

[4] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, IT-27, no. 5, pp. 533-547, Sept. 1981.

[5] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low density parity check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, no. 7, pp.2711-2736, Nov. 2001.

[6] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, 2nd edition, Prentice-Hall, Upper Saddle River, NJ, 2004.

[7] T. J. Richardson, M. A. Shokrollahi. R.L. Urbamke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Therory*, vol. 47, no. 2, pp. 619-637, Feb. 2001.

[8] R. M. Tanner, "Spectral graphs for quasi-cyclic LDPC codes," *Proc. IEEE Int. Symp. Inform. Theory*, Washington, D.C., Jun. 2001, p. 226.

[9] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, CA: Morgan Kaufmann 1988.

[10] N. Wiberg, "Codes and Decoding on general graphs," *PhD. dissertation, Dept. Elec. Eng., University of Linkoping*, Linkoping, Sweden, 1996.

[11] F. R. Kschischang, B. J. Frey, and H. -A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47. no.2, pp. 498-519, Feb. 2001.

[12] J. Zhang and M. P. C. Fossorier, "A modified weighted bit-flipping decoding for low-density parity-check codes," *IEEE Commun. Letters*, vol. 8, pp. 165-167, Mar. 2004.

[13] Z. Liu and D. A. Pados, "A decoding algorithm for finite-geometry LDPC codes," *IEEE Trans. Commun.*, vol 53, no. 3, pp. 415-421, Mar. 2005.

[14] N. Miladinovic and M. P. Fossorier, "Improved bit-flipping decoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 4, pp. 1594-1606, Apr. 2005.

[15] M. Jiang, C. Zhao, Z. Shi, and Y. Chen, "An improvement on the modified weighted bit-flipping decoding algorithm for LDPC codes," *IEEE Commun. Letters*, vol. 9, pp. 814-816, Sept. 2005.

[16] N. Mobini, A. H. Banihashemi, and S. Hemati, "A differential binary-passing LDPC decoder," *Proc. IEEE Globecom.*, pp. 1561-1565, Washington D.C., Novermber, 2007.

[17] I. S. Reed, "A class of multiple-error-correcting codes and decoding scheme," *IRE Trans.* IT-4, pp. 38-49, Sept. 1954.

[18] J. L. Massey, *Threshold Decoding*, MIT Press, Cambridge, Mass. 1963.

[19] J. G. Proakis, *Digital Communications*, fifth edition, McGraw-Hill Higher Education, 2008.

[20] L. Lan, L. -Q. Zeng, Y. Y. Tai, L. Chen, S. Lin, and K. Abdel Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: a finite field approach," *IEEE Trans. Inform. Theory*, vol. 53, no. 7, pp. 2429-2458, Jul. 2007.

TABLE I
COMPUTATIONAL COMPLEXITIES REQUIRED PER ITERATION FOR DECODING AN LDPC CODE WITH VARIOUS ITERATIVE DECODING ALGORITHMS

Decoding Algorithm	Computation Cost per Iteration						
	BO	IA	IC	RA	RC	RM	Log
BF	$\delta + n - m$	$\delta - n$	n				
SRBI-MLGD	$2\delta + n - m$	δ					
DBMPD	$4\delta + n$	$4\delta + 2n$					
ZF-WBF	$\delta + m$			δ	δ	n	
JZSC-WBF	$\delta + m$			δ	2δ	n	
SPA						6δ	n

BO: Binary Operation; IA: Integer Addition; IC: Integer Comparison;
RA: Real Addition; RC: Real Comparison; RM: Real Multiplication
Log: Logarithm.

TABLE II
MEMORY REQUIREMENTS FOR DECODING AN LDPC CODE WITH VARIOUS ITERATIVE DECODING ALGORITHMS

Decoding Algorithm	Memory Requirement	
	Units (bits)	RN
BF	$m + n(b-1) + n\log n$	
SRBI-MLGD	$m + nb$	
DBMPD	$\delta(b+1) + n(2b+1)$	
ZF-WBF	m	$n + m$
JZSC-WBF	m	$n + 2m$
SPA		δ

RN: Real Number.
Remark: The number of memory units required to store a real number depends on the number of bits used to represent the real number.

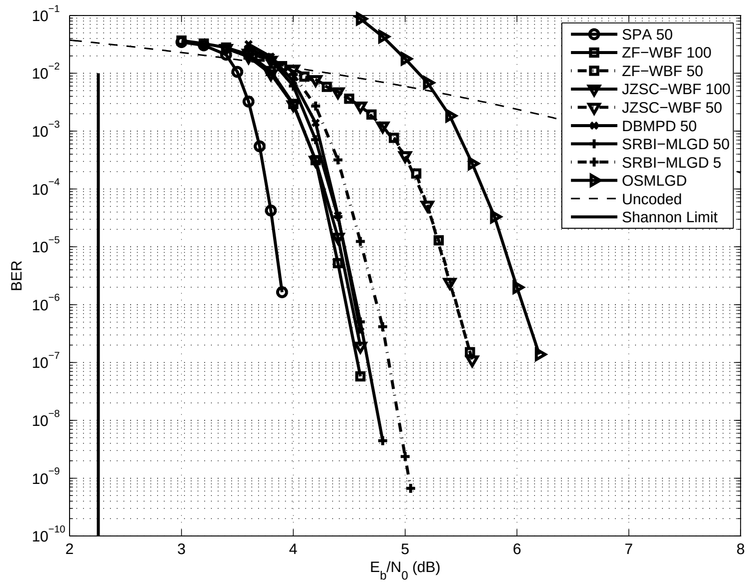

Fig. 1. Error performances of the (4095,3367) EG-LDPC code given in Example 1 decoded with the SRBI-MLGD- and other algorithms.

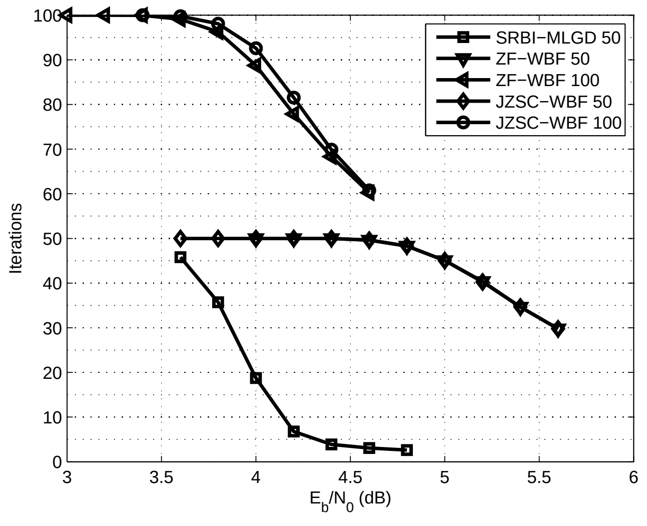

Fig. 2. Average numbers of iterations required for decoding the (4095,3367) EG-LDPC code with the SRBI-MGLD-, the JZSC-WBF- and the ZF-WBF-algorithms.

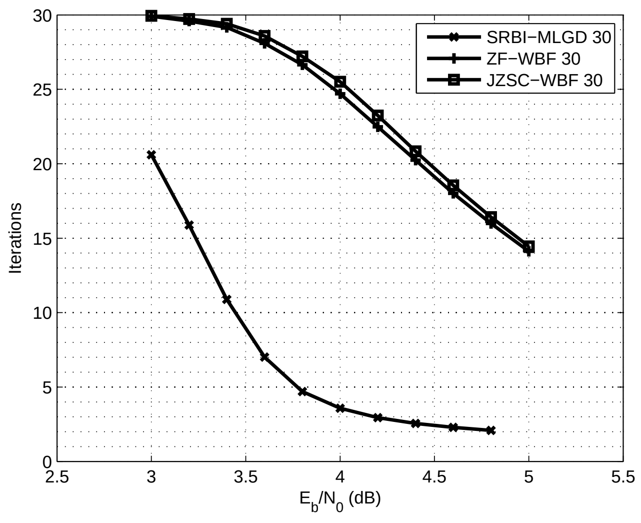

Fig. 4. Average numbers of iterations required for decoding the (961,721) QC-LDPC code with the SRBI-MLGD-, the JZSC-WBF- and the ZF-WBF-algorithms.

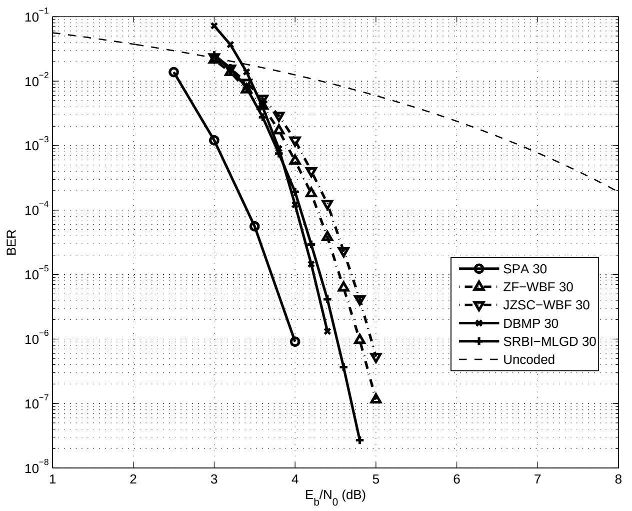

Fig. 3. Error performances of the (961,721) QC-LDPC code given in Example 2 decoded with the SRBI-MLGD- and other algorithms.

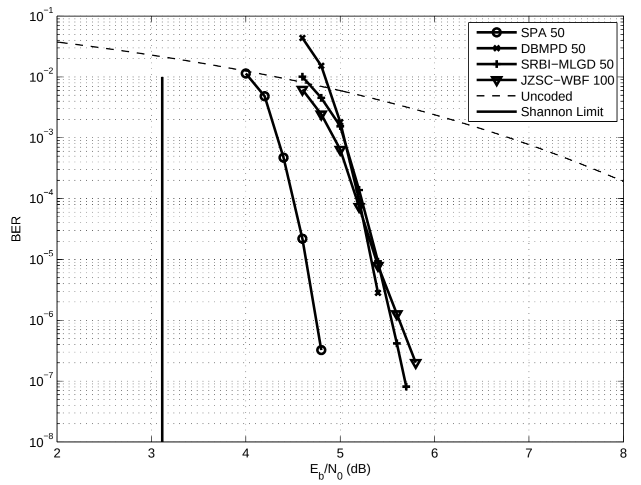

Fig. 5. Error performance of the (5256,4679) QC-LDPC code given in Example 3 decoded with the SRBI-MLGD- and other algorithms.

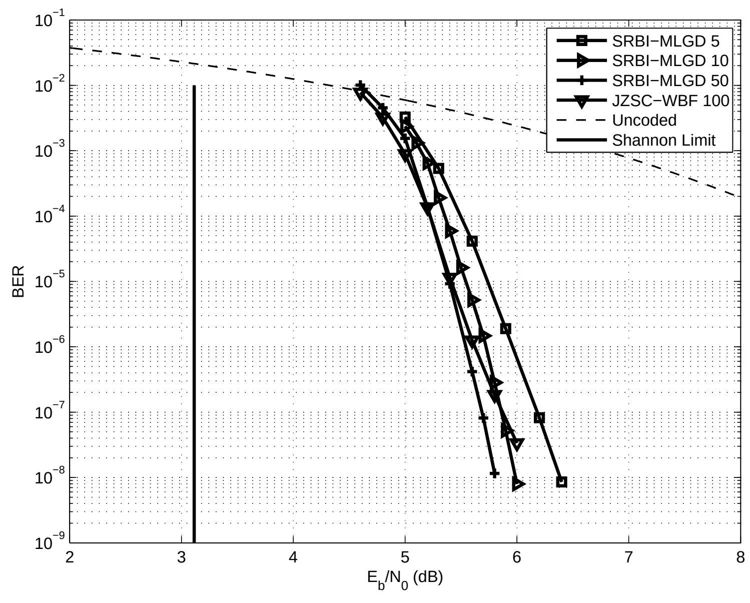

Fig. 6. Error performance of the (5256,4679) QC-LDPC code decoded with the SRBI-MLGD-algorithm with 5, 10 and 50 iterations.