MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Low Latency Decoding of EG LDPC Codes

Juntan Zhang, Jonathan S. Yedidia, Marc P.C. Fossorier

TR2005-036 June 2005

Abstract

We describe simple interactive decoders for low-density parity check codes based on Euclidean geometries, suitable for practical VLSI implementation in applications requiring very fast decoders. The decoders are based on shuffled and replica-shuffled versions of iterative bit-flipping and quantized weighted bit-flipping schemes. The proposed decoders converge faster and provide better ultimate performance than standard bit-flipping decoders. We present simulations that illustrate the performance versus complexity trade-offs for these decoders. We can show in some cases through importance sampling that no significant error-floor exists.

Publication History:–

1. First printing, TR-2005-036, June 2005

Low Latency Decoding of EG LDPC Codes

Juntan Zhang, Jonathan S. Yedidia and Marc P. C. Fossorier

Abstract

We describe simple iterative decoders for low-density parity check codes based on Euclidean geometries, suitable for practical VLSI implementation in applications requiring very fast decoders. The decoders are based on shuffled and replica-shuffled versions of iterative bit-flipping and quantized weighted bit-flipping schemes. The proposed decoders converge faster and provide better ultimate performance than standard bit-flipping decoders. We present simulations that illustrate the performance versus complexity tradeoffs for these decoders. We can show in some cases through importance sampling that no significant error-floor exists.

1 Introduction

Low density parity check (LDPC) codes were first discovered in 1960's [1] and have received significant attention recently because of their excellent performance when decoded using iterative decoders [2, 3]. LDPC codes can be constructed using random or deterministic approaches. In this report, we focus on a class of LDPC codes known as Euclidean Geometric (EG) LDPC codes, which are constructed deterministically using the points and lines of a Euclidean geometry [4, 5]. The EG LDPC codes that we consider are cyclic and consequently their encoding can be efficiently implemented with linear shift registers. Minimum distances for EG codes are also reasonably good and can be derived analytically. Iteratively decoded EG LDPC codes also seem to not have the serious error-floors that plague randomly-constructed LDPC codes; this fact can be explained by the observation made in [6] that EG LDPC codes do not have pseudo-codewords of weight smaller than their minimum distance. For these reasons, EG LDPC codes are good candidates for use in applications like optical communications that require very fast encoders and decoders and very low bit error-rates.

LDPC codes can be decoded using hard-decision, soft-decision and hybrid decoding methods. Soft decoding algorithms such as belief propagation (BP) provide good performance but require high decoding complexity, and are therefore not very suitable for high speed VLSI implementations. Instead, hard-decision and hybrid schemes such as bit flipping (BF) and quantized weighted bit-flipping (QWBF) offer a better trade-off between error performance, decoding complexity, and decoding speed.

Most standard iterative decoders of LDPC codes require at least several tens of iterations for the iterative decoding process to converge, which is not realistic for high-speed VLSI implementations. In [7, 8], a decoding scheme called "shuffled BP" was presented to reduce the required number of iterations of standard BP decoding. Related bit-flipping algorithms based on the shuffled BP idea are easy to construct. Recently, an improved but more complex iterative algorithm named "replica-shuffled iterative decoding" was developed to further decrease the required number of decoding iterations [9]. In this report, we study the performance of shuffled and replica-shuffled versions of bit-flipping (BF) and quantized weighted bit-flipping (QWBF) decoders for EG LDPC codes.

Another problem for VLSI implementations of LDPC decoders is related to the fact that to achieve a good performance, the LDPC code must have a large codeword length, and a correspondingly large parity check matrix. The large parity check matrix makes it difficult to implement the iterative decoder in a fully parallel manner. To deal with this problem, it makes sense to consider shuffled and replica-shuffled algorithms which divide the codeword bits into

groups, and update one group of bits at a time [7, 9].

This report is organized as follows. Section 2 describes the construction of EG LDPC codes. Section 3 briefly reviews standard BF, weighted bit-flipping, and QWBF decoders. Shuffled and group-shuffled versions of these decoders are presented in Section 4 and Section 5, respectively. In Section 6, replica group-shuffled schemes are discussed. Finally, in Sections 7 and 8, we provide simulation results for the various presented decoding methods.

2 Definition of EG LDPC codes

A binary LDPC code is specified by a parity-check matrix containing mostly zeros and only a small number of ones. A *regular* binary $(N, K)(d_v, d_c)$ LDPC code has a transmitted codeword block length N, a information block length K, and a parity check matrix with precisely d_v ones in each column and d_c ones in each row. We refer to the N elements of an LDPC codeword $\mathbf{w} = [w_n]$ as bits, and the M rows of the parity check matrix $\mathbf{H} = [H_{ml}]$ as checks. Accordingly, in a regular binary LDPC code, every code bit is checked by exactly d_v parity checks, and every parity check involves exactly d_c code bits. We denote the set of bits that participate in check m by $\mathcal{N}(m) = \{n : H_{mn} = 1\}$ and the set of checks in which bit n participates as $\mathcal{M}(n) = \{m : H_{mn} = 1\}$.

EG LDPC codes [4, 5] are regular LDPC codes characterized by a parity-check matrix which is constructed using a finite Euclidean geometry. Let α=(α_1,α_2,...,α_m) be an m-tuple whose component α_i is from the Galois field $GF(2^s)$. The set of all possible α's has cardinality 2^{ms} and forms an m-dimensional *Euclidean geometry* over $GF(2^s)$, which is denoted by $EG(m, 2^s)$. Each m-tuple α is called a *point* in $EG(m, 2^s)$. The all-zeros point is called the *origin*. Let α_1 and α_2 be two linearly independent points in $EG(m, 2^s)$. Then the collection of $\{\alpha_1 + \beta\alpha_2\}$, with $\beta \in GF(2^s)$, has 2^s points and forms a *line* (1-flat) in $EG(m, 2^s)$. There are $J_0 = \frac{(2^{(m-1)s}-1)(2^{ms}-1)}{2^s-1}$ lines in $EG(m, 2^s)$ that do not contain the origin.

To construct a cyclic EG LDPC code based on $EG(m, 2^s)$, we form the parity check matrix $\mathbf{H}_{EG}$ whose columns are all of the $2^{ms} - 1$ non-origin points in $EG(m, 2^s)$, and whose rows are the incidence vectors of all of the J_0 lines in $EG(m, 2^s)$ that do not contain the origin. The codes used in this paper all have $m = 2$, so $J_0 = N = 2^{2s} - 1$, which means that the parity check matrix that we use has an equal number of rows and columns (some of the rows are redundant).

3 BF and Quantized WBF decoding

Assume a codeword $\mathbf{w} = (w_1, w_2, \ldots, w_N)$ is transmitted over an AWGN channel with zero mean and variance $N_0/2$ using BPSK signaling and let $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ be the corresponding received sequence.

3.1 Standard BF Decoding

The standard BF decoding is a hard decision algorithm. Let $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ be the binary hard decision sequence obtained from $\mathbf{y}$ as follow:

$$z_n = \begin{cases} 1 & \text{if } y_n \leq 0 \\ 0 & \text{if } y_n > 0 \end{cases}$$

Let $\mathbf{s}$ be the syndrome of $\mathbf{z}$:

$$\mathbf{s} = (s_1, s_2, \ldots, s_M) = \mathbf{z} \cdot \mathbf{H}^T \tag{1}$$

where

$$s_m = \sum_{n=1}^{N} z_n h_{mn}. \tag{2}$$

The received vector $\mathbf{z}$ is a codeword if and only if $\mathbf{s} = \mathbf{0}$. If $\mathbf{s} \neq \mathbf{0}$, errors are detected, and any nonzero syndrome s_m indicates a parity failure. Let F_n be the set of nonzero syndromes checking on bit n, i.e, $F_n = \{s_m : s_m = 1 \text{ and } m \in \mathcal{M}(n)\}$. In standard BF decoding, the decoder computes all the syndromes and then flips any bit which is involved in more than a fixed number δ of parity failures. Based on these new values, the syndromes are recomputed and the process is repeated until a codeword is found or the maximum number of iterations is reached.

Thus, the standard BF decoding is carried out as follows:

Step 1 Compute $\mathbf{s} = (s_1, s_2, \ldots, s_M) = \mathbf{z} \cdot \mathbf{H}^T$

Step 2 For $n = 1, 2, \ldots, N$, flip z_n with $|F_n| \geq \delta$.

Step 3 Repeat step 1 and 2 until $\mathbf{s} = \mathbf{0}$ or the maximum number of iterations I_{max} is reached.

It should be noted that this version of BF decoding can be viewed as a simplified version of the conventional Gallager algorithm B [1]. This simplification can be justified by the large values of d_c and d_v for EG LDPC codes.

3.2 WBF Decoding

The performance of standard BF decoding can be improved upon by using a soft-valued reliability measure for the received symbols. The standard WBF algorithm [4] first computes for $m = 1, 2, \ldots, M$,

$$|y|_{min-m} \quad = \quad \min_{n:n\in\mathcal{N}(m)} |y_n|. \tag{3}$$

As explained below, $|y|_{min-m}|$ is a measure of the reliability of the mth check.

Next, WBF decoding is carried out as follows:

Step 1 For $m = 1, 2, \ldots, M$, compute the syndrome $s_m = \sum_{n=1}^{N} z_n H_{mn}$ from $\mathbf{z}$.

Step 2 For $n = 1, 2, \ldots, N$, compute:

$$E_n \quad = \quad \sum_{m\in\mathcal{M}(n)} (2s_m - 1)|y|_{min-m} \tag{4}$$

Step 3 Flip the bit z_n for $n = \arg\max_{1\le n\le N} E_n$.

Step 1 to 3 are repeated until all the parity check equations are satisfied or until the maximum number of iterations I_{max} is reached.

WBF decoding achieves better performance than BF decoding by making more accurate decisions for each bit based on a flipping criteria that considers soft reliability information. For the AWGN channel, a simple measure of the reliability of a received symbol y_n is its magnitude, $|y_n|$. The larger the magnitude $|y_n|$ is, the larger the reliability of the corresponding hard-decision digit z_n is. For $m = 1, 2, \ldots, M$, $|y|_{min-m}$ given in equation (3) can be viewed as a measure of the reliability of the syndrome s_m computed with z_n's, $n \in \mathcal{N}(m)$. For $n = 1, 2, \ldots, N$, the larger E_n is, the less likely the hard decision z_n is, which is why the bits with the largest values for E_n are flipped first.

3.3 Quantized WBF decoding

From the practical point of view, WBF decoding is problematic for VLSI implementations, because a real number $|y|_{min-m}$ must be stored for each check, these real numbers must be added to determine the reliability E_n of each bit, and the bits must then be sorted according to their values of E_n. In a more practical version of WBF decoding, which we call "quantized WBF" (QWBF) decoding, each bit is assigned a "high" or "low" reliability based on whether $|y_n|$

is greater or less than a pre-assigned threshold Δ_1. Then, for each check, if all the bits involved in that check have "high" reliability, the check is also considered to have high reliability, but if even a single bit involved in the check has "low" reliability, the check is considered to have low reliability. High-reliability checks are assigned a value of $|y|_{min-m} = 2$, while low-reliability checks are assigned a value of $|y|_{min-m} = 1$. Next, QWBF decoding proceeds as follows:

Step 1 For $m = 1, 2, \ldots, M$, compute the syndrome $s_m = \sum_{n=1}^{N} z_n H_{mn}$ from $\mathbf{z}$.

Step 2 For $n = 1, 2, \ldots, N$, compute:

$$E_n = \sum_{m \in \mathcal{M}(n)} (2s_m - 1)|y|_{min-m} \tag{5}$$

Step 3 Flip all bits z_n for which E_n exceeds a pre-defined threshold Δ_2.

Step 1 to 3 are repeated until all the parity check equations are satisfied or until the maximum number of iterations I_{max} is reached.

Note that in QWBF decoding, two pre-determined thresholds Δ_1 and Δ_2 have to be specified. In the simulations described below, these thresholds were chosen by empirical testing. Our chosen thresholds should be reasonably good, although we cannot be certain of how far they are from the optimal thresholds; a theoretical analysis would certainly be desirable.

Compared to WBF decoding, QWBF decoding has the advantages that only one bit needs to be stored to record the reliability of each bit and check, that all the addition is simple integer arithmetic, and that no sorting of the bits by reliability needs to be done. For these reasons, we consider QWBF decoding to be much more realistic than WBF decoding for our target applications.

4 Shuffled BF and QWBF decoding

As mentioned in Section 1, standard BF decoders require an undesirably large number of iterations to converge. *Shuffled* BF (or QWBF) decoding is designed to accelerate the decoding process. Let us first assume, for the sake of argument, that the bits are processed serially; one bit is processed in each unit time. During a given iteration of decoding, we assume that the n-th bit is processed in the n-th unit of time. If the flipping condition is satisfied, this bit is flipped. Generally, the new value of z_n is more likely to be correct than the old one. Consequently, the syndromes $\{s_m : m \in \mathcal{M}(n)\}$ based on the new value of z_n are more reliable than the corresponding old ones. In shuffled BF (or QWBF) decoding, at each iteration, once a bit is flipped,

all the syndromes involving this bit are flipped too and the processing of the remaining bits is based on these new syndrome values. Because the more reliable bit values are taken advantage of as soon as available, the shuffled version of BF (or WBF or QWBF) decoding is expected to converge faster than the standard one.

Shuffled BF is carried out as follows.

Initialization Compute $\mathbf{s} = (s_1, s_2, \ldots, s_M) = \mathbf{z} \cdot \mathbf{H}^T$

Step 1 For $n = 1, 2, \ldots, N$, if $|F_n| \geq \delta$, flip z_n and flip $\{s_m : m \in \mathcal{M}(n)\}$.

Step 2 Repeat step 1 until $\mathbf{s} = \mathbf{0}$ or I_{max} is reached.

Similarly, the shuffled WBF is carried out as follows.

Initialization For $n = 1, 2, \ldots, N$, assign each bit a value z_n, and a one-bit reliability $|y_n|$. For $m = 1, 2, \ldots, M$, compute the reliability value $|y|_{min-m}$ using the one-bit reliabilities $|y_n|$ and the threshold Δ_1.

Step 1 For $n = 1, 2, \ldots, N$, compute:

$$E_n = \sum_{m \in \mathcal{M}(n)} (2s_m - 1)|y|_{min-m} \tag{6}$$

Step 2 For $n = 1, 2, \ldots, N$, if $|E_n| \geq \Delta_2$, flip z_n and flip $\{s_m : m \in \mathcal{M}(n)\}$.

Step 1 to 2 are repeated until all the parity check equations are satisfied or I_{max} is reached.

5 Group-shuffled schemes

An entirely serial implementation of shuffled BF decoding would be desirable from the point of view of performance, but is not very realistic in terms of VLSI implementation. On the other hand, an entirely parallel implementation is not so desirable, nor even realistic given the large block-lengths that one would expect to use, as well as the large values d_c and d_v for EG LDPC codes.

Thus, a more realistic scenario is for the bits in an EG LDPC code to be processed in *groups* of bits, where the groups are processed serially, but the bits within a group are processed in parallel. We call such decoding schemes "group-shuffled" decoding.

Assume the N bits of a codeword are divided into G groups and each group contains $\frac{N}{G} = N_G$ bits (assuming $N \bmod G = 0$ for simplicity). Grouped shuffled BF decoding is carried out as follows:

Initialization Compute $\mathbf{s} = (s_1, s_2, \ldots, s_M) = \mathbf{z} \cdot \mathbf{H}^T$

Step 1 For $g = 1, 2, \ldots, G$,

i) process the following step in parallel: For $g \cdot N_G + 1 \leq n \leq (g+1) \cdot N_G + 1$, if $|F_n| \geq \delta$, flip z_n.

ii) process the following step in parallel: For $g \cdot N_G + 1 \leq n \leq (g+1) \cdot N_G + 1$, if $|F_n| \geq \delta$, flip $\{s_m : m \in \mathcal{M}(n)\}$.

Step 2 Repeat step 1 until $\mathbf{s} = \mathbf{0}$ or I_{max} is reached.

Group-shuffled QWBF decoders operate in an analogous way; the full details are omitted here.

6 Replica Group-shuffled BF and QWBF decoding

In group-shuffled BF (or QWBF) decoding, as presented in the previous section, the groups of bits are processed sequentially. After one iteration is complete, the group of bits that is processed last will be most reliable, because it has used values for all the previously processed bits that have been updated. On the other hand, one can imagine creating a second decoder, which processed the groups in a different order; in this decoder, the reliability ordering of the groups would be different, depending on the ordering.

To take advantage of this property of group-shuffled BF (or QWBF) decoders, in a *replica group-shuffled* BF (or QWBF) decoder [9], two or more group-shuffled subdecoders ("replicas") based on different updating orders operate simultaneously and cooperatively. After each group of bits is processed, the replica sub-decoders exchange bit values with each other (according to the rule that each replica transmits those bit values that it just processed, and the other replicas copy the transmitted bit values) and then the next group of bits is processed in each replica based on these new values.

In general, replica-shuffled decoders obtain faster decoding at the price of duplicating sub-decoders. Note, however, that replica-shuffled decoders are only really faster if the cost of

transmitting and copying bit values between decoders is much less than the cost of updating bit values.

To demonstrate how a replica group-shuffled BF decoder operates, we consider the case when there are just two sub-decoders, each of which divides the bits into G groups of N/G bits. We assume that the first sub-decoder (which we call $\overrightarrow{D}$) processes the groups in the order the $1, 2, \ldots, G$, while the second sub-decoder ($\overleftarrow{D}$) processes the groups in the order $G, G-1, \ldots, 1$.

Let $\overrightarrow{\mathbf{s}}$, $\overrightarrow{\mathbf{z}}$, and $\overrightarrow{F_n}$ represent the syndromes, bit values, and number of violated constraints in the first sub-decoder $\overrightarrow{D}$, and define notations associated with decoder $\overleftarrow{D}$ in a similar way. Replica group-shuffled BF decoding with these two sub-decoders is then carried out as follows:

Initialization Compute $\overrightarrow{\mathbf{s}} = \overleftarrow{\mathbf{s}} = \mathbf{z} \cdot \mathbf{H}^T$. Let $\overrightarrow{\mathbf{z}} = \overleftarrow{\mathbf{z}} = \mathbf{z}$.

Step 1 For $\overrightarrow{g} = 1, 2, \ldots, G$ in sub-decoder $\overrightarrow{D}$, and for $\overleftarrow{g} = G, G-1, \ldots, 1$ in sub-decoder $\overleftarrow{D}$, simultaneously process the two sub-decoders a according to the following rules (the arrows over g and F_n should be understood):

- **i)** (Update bits) Process the following step in parallel: For $g \cdot N_G + 1 \leq n \leq (g+1) \cdot N_G + 1$, if $|F_n| \geq \delta$, flip z_n.
- **ii)** (Update syndromes) Process the following step in parallel: For $g \cdot N_G + 1 \leq n \leq (g+1) \cdot N_G + 1$, if $|F_n| \geq \delta$, flip $\{s_m : m \in \mathcal{M}(n)\}$.
- **iii** (Exchange Information) All the bits associated with the group that was just updated in the first sub-decoder are copied in the second sub-decoder, and vice-versa. Then all the syndromes associated with the copied bits are updated.

Step 2 Repeat step 1 until $\overrightarrow{\mathbf{s}} = \mathbf{0}$ (or $\overleftarrow{\mathbf{s}} = \mathbf{0}$) or I_{max} is reached.

Replica group-shuffled QWBF decoding is carried out in an analogous way; the full description is omitted here.

7 Simulation results of shuffled schemes

We present simulation results for two-dimensional ($m = 2$) cyclic EG LDPC codes with parameters ($N = 4095, K = 3367$), and ($N = 16373, K = 14179$). The codes were represented using square N by N parity check matrices.

Figure 1 depicts the error rates for iterative decoding of the $(4095, 3367)$ EG-LDPC code using standard BF decoding and group-shuffled BF decoding. The word error rate and bit error

rates are simultaneously shown in many of our plots; obviously the bit error rates are the lower curves. The number of groups used in the group-shuffled decoder was $G = 2$, $G = 16$ and $G = 4095$. The maximum number of iterations I_{max} for the group-shuffled decoder was only two. Note that using 16 groups is nearly as good as 4095 (fully serial operation). In fact, for all the simulation results shown, using four groups was also nearly as good as using 16. Note also that the word error rate performance using 16 groups and two iterations in group-shuffled decoding is nearly as good as as that using 10 iterations in standard BF decoding. Figure 2 shows similar results for the $(16383, 14179)$ EG-LDPC code.

Figure 3 depicts the error rates of iterative decoding of the $(4095, 3367)$ EG-LDPC code using standard QWBF decoding and the group-shuffled QWBF algorithm, for $G = 2, 16, 4095$. The maximum number of iterations for group shuffled BF was set to be 5. The threshold parameters used had values $\Delta_1 = 0.09$ and $\Delta_2 = 8.0$. There is clearly a performance gain from using QWBF instead of standard BF (about .5 dB at a BER of 10^{-5}). Again we see that a group-shuffled QWBF decoder with 16 groups using fewer iterations can perform roughly as well as a standard QWBF decoder that uses more iterations.

8 Simulation results of replica group-shuffled schemes

Figure 4 depicts the error rates for iterative decoders of the $(4095, 3367)$ EG-LDPC code using standard BF and replica group-shuffled BF decoding with four subdecoders, for $G = 2, 16, 4095$. The maximum number of iterations for the replica group-shuffled BF was set to be 2. We observe some small improvement in performance compared to using ordinary group-shuffled BF decoding (about .1-.2 dB at a BER of 10^{-5}). Note that the WER performance of replica group-shuffled BF decoding with four subdecoders and $I_{max} = 2$, and $G = 2$ (in fact, for $G \geq 4$), is now approximately the same as that of the standard BF with maximum number of 10, and the BER performance is even better.

Figure 5 depicts the error rates of the same code decoded by the standard and replica group-shuffled BF methods, with $I_{max} = 20$ and $I_{max} = 10$, respectively. The point of this figure is more theoretical than practical–to see whether replica group-shuffled BF decoding clearly outperforms standard BF decoding if I_{max} is large enough. We see that it does indeed outperform standard BF decoding for both WER and BER. This can partly explained by the fact that less error propagation occurs in the replica group-shuffled method.

Figure 6 shows the error rates using replica group-shuffled QWBF decoding using four replicas and $G = 16$. This plot should be considered a preliminary one, because the thresholds used

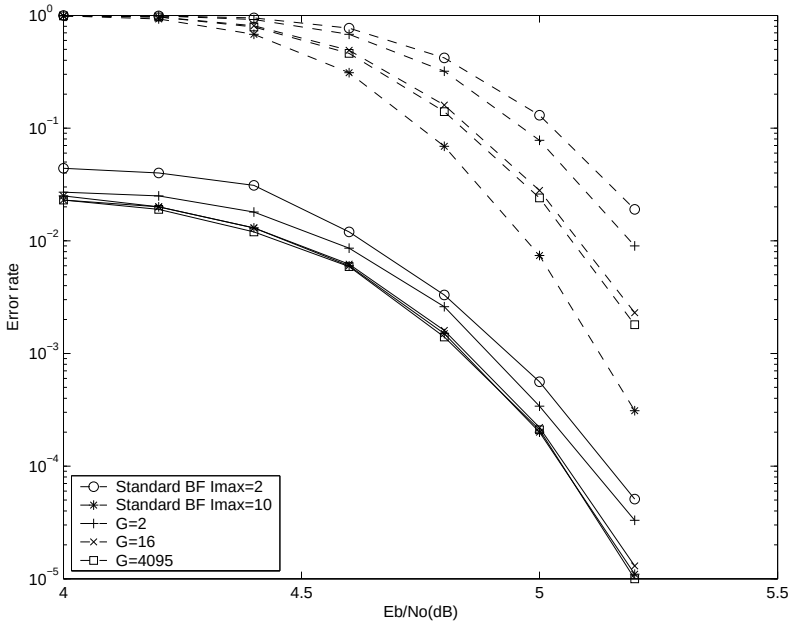

Figure 1: Error rates of the $(4095, 3367)$ EG-LDPC code with the standard BF and group shuffled BF algorithm, for $G = 2, 16, 4095$ and at most 2 iterations.

($\Delta_1 = 0.09$ and $\Delta_2 = 8.0$) were not optimized in any way (we simply used the same thresholds as those chosen for the decoder that did not use replicas). The performance is noticeably better than that of replica group-shuffled BF decoding, but only slightly better than group shuffled QWBF decoding (without using replicas). On the other hand, it is possible that when the thresholds are further optimized, there will be a greater performance improvement.

8.1 Performance at very low error rates

An important issue in optical communication systems and in storage systems is the performance at very low error-rates. The question that must be answered is whether there is a hidden error floor in the high SNR regime. This question is hard to answer through simulations, but we can make some progress, and set worse-case bounds on the error floor, by using *importance sampling* for the case of BF decoding.

To obtain performance curves for very low error rates, we generated received blocks with a fixed number n of bit flips. Of course, although the number n of bit flips was fixed, the positions of the bit flips were generated randomly, and changed for each received block. For each n, we then performed simulations to find the corresponding error performance $P(n)$. We know that no errors at all will occur if $n \leq t$, where t is the bounded error correcting capability of the code $t = \lfloor (d_{min} - 1)/2 \rfloor$ (this corresponds to the reasonable assumption that the decoder has a

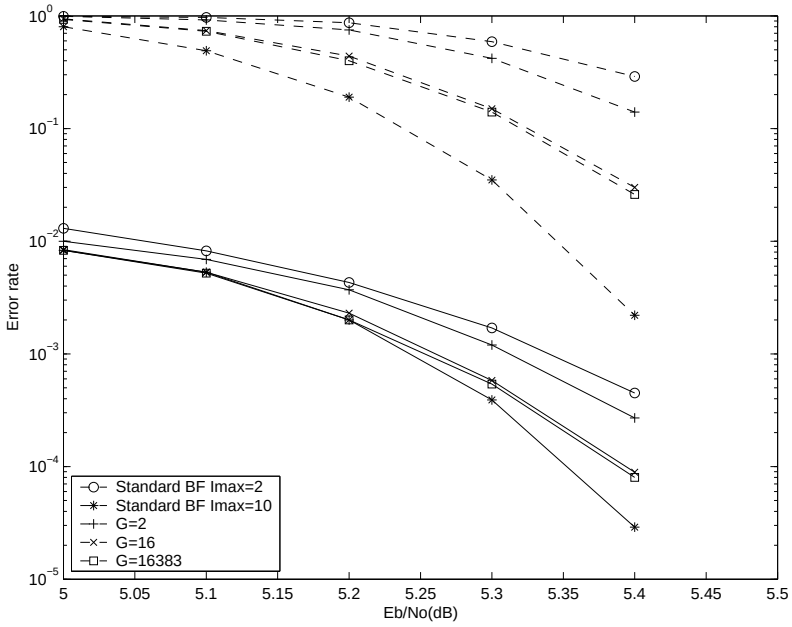

Figure 2: Error rates of the (16383.14179) EG-LDPC code with the standard BF and group shuffled BF algorithm, for $G = 2, 16, 4095$ and at most 2 iterations.

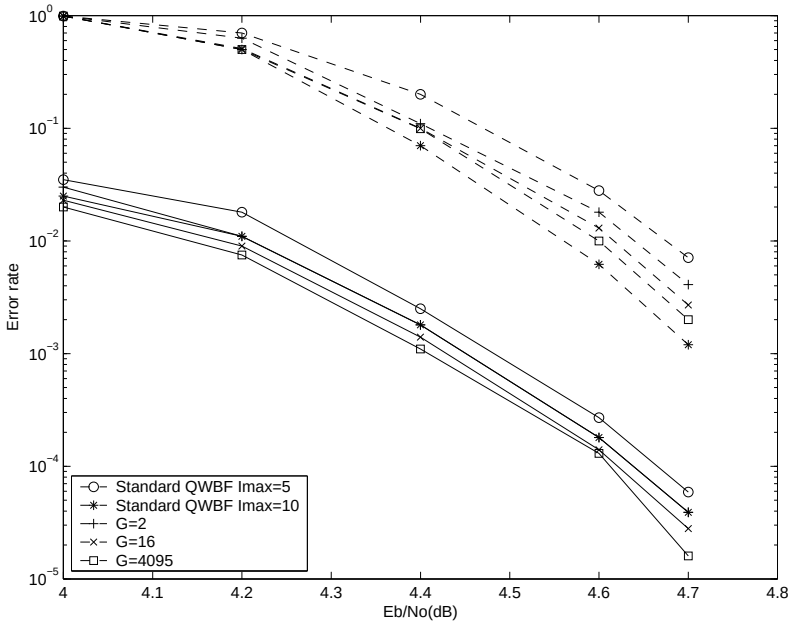

Figure 3: Error rates of the $(4095, 3367)$ EG-LDPC code using standard QWBF decoding and the group shuffled QWBF algorithm, for $G = 2, 16, 4095$ and at most 5 iterations.

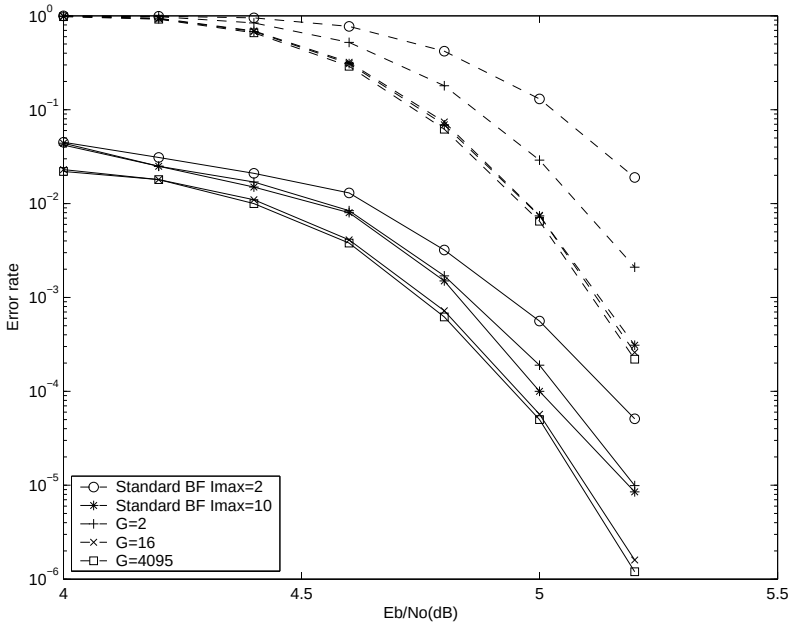

Figure 4: Error rates of the $(4095, 3367)$ EG-LDPC code using standard BF decoding and the replica group shuffled BF algorithm with four subdecoders, for $G = 2, 16, 4095$ and at most 2 iterations.

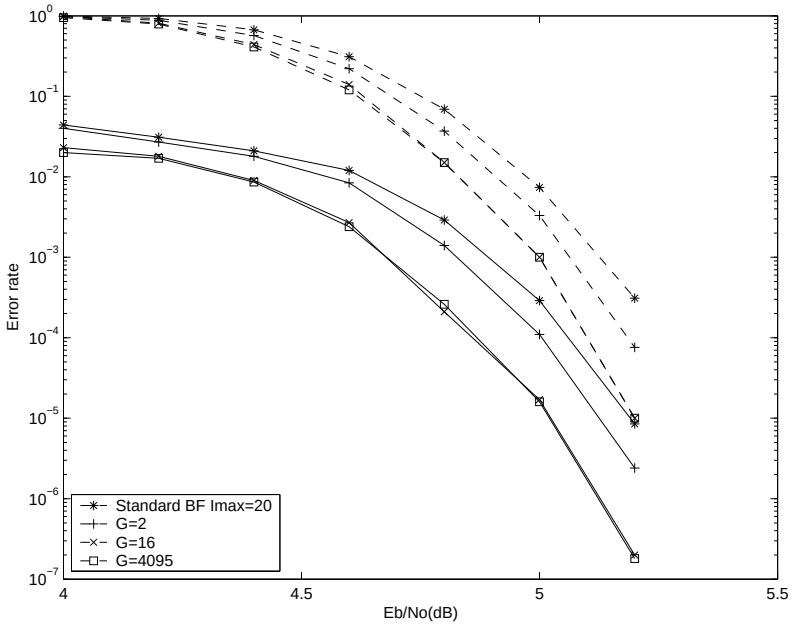

Figure 5: Error rates of the $(4095, 3367)$ EG-LDPC code using standard BF decoding and the replica group shuffled BF algorithm with four subdecoders, for $G = 2, 16, 4095$ and at most 10 iterations.

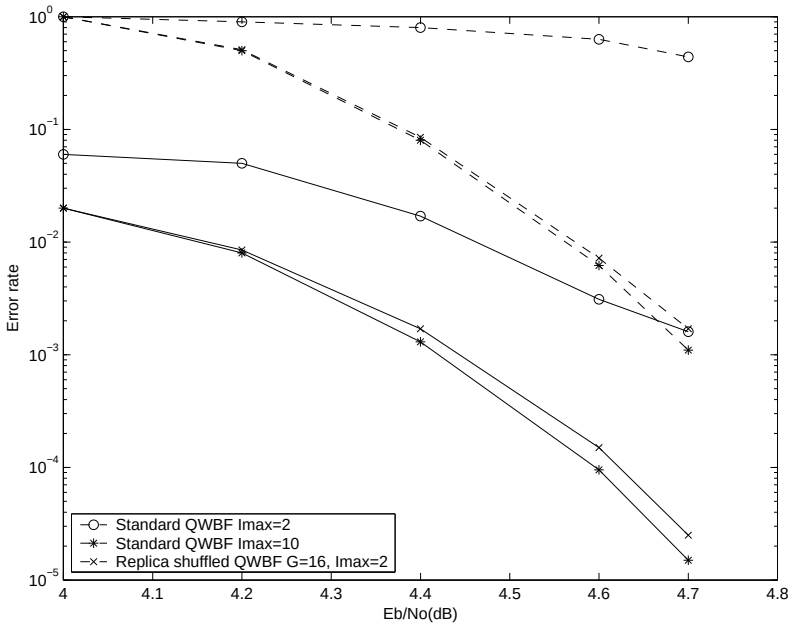

Figure 6: Error rates of the $(4095, 3367)$ EG-LDPC code using standard QWBF decoding and the replica group shuffled BF algorithm with four subdecoders, for $G = 16$ and at most 2 iterations.

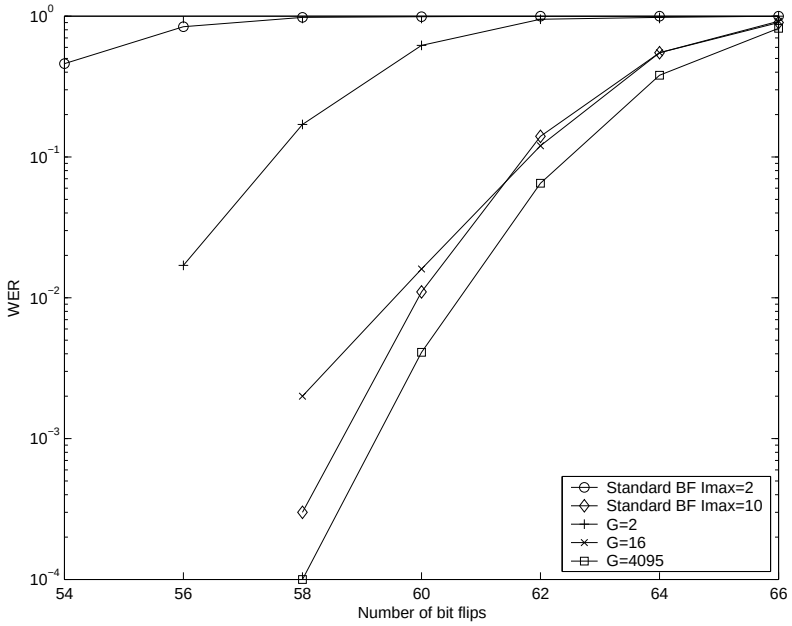

Figure 7: WER of the $(4095, 3367)$ EG-LDPC code using standard BF decoding and the replica group shuffled BF algorithm with four subdecoders, for $G = 2, 16, 4095$, for a fixed number of errors and $I_{max} = 2$.

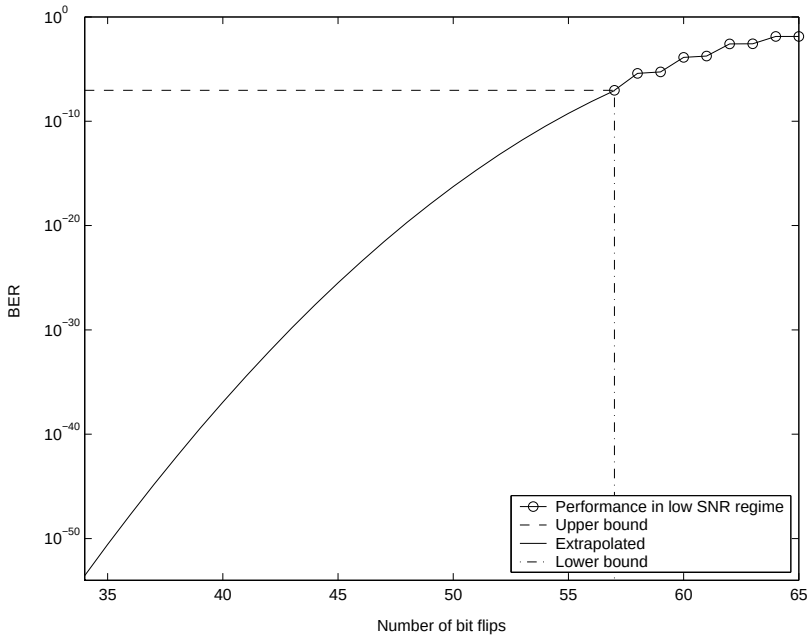

Figure 8: Replica group-shuffled BF decoding of the (4095,3367) EG LDPC code for fixed number of errors with 4 subdecoders and $G = 16$, $I_{max} = 2$. The point of this figure is to show the extrapolations used in constructing the worst-case (upper bound), best-case (lower bound) and likely-case (extrapolated) performance curves at low error rates.

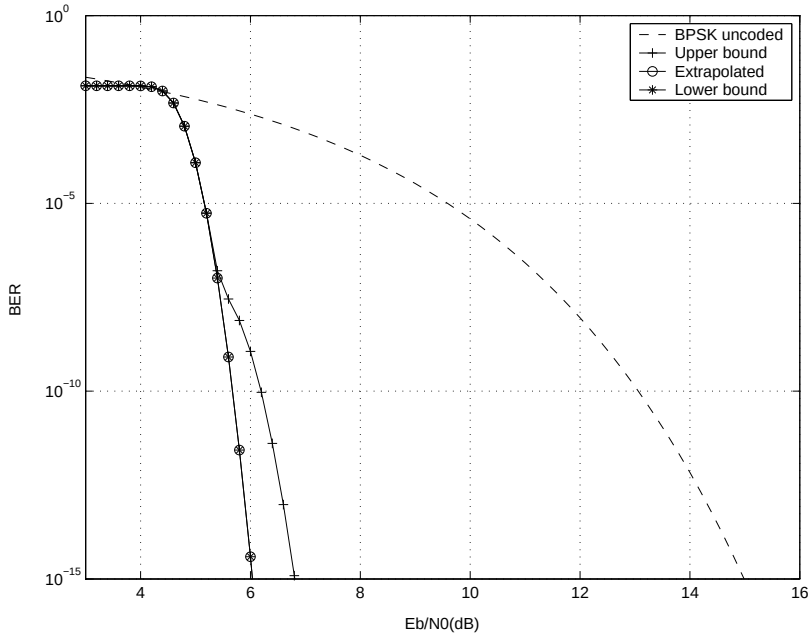

Figure 9: Replica group shuffled BF decoding of the (4095,3367) EG LDPC code in high SNR regime with 4 subdecoders and $G = 16$, $I_{max} = 2$. Note that the worst-case (upper bound) performance is never more than 1 dB worse than the likely-case (extrapolated) performance, indicating that there is no error floor.

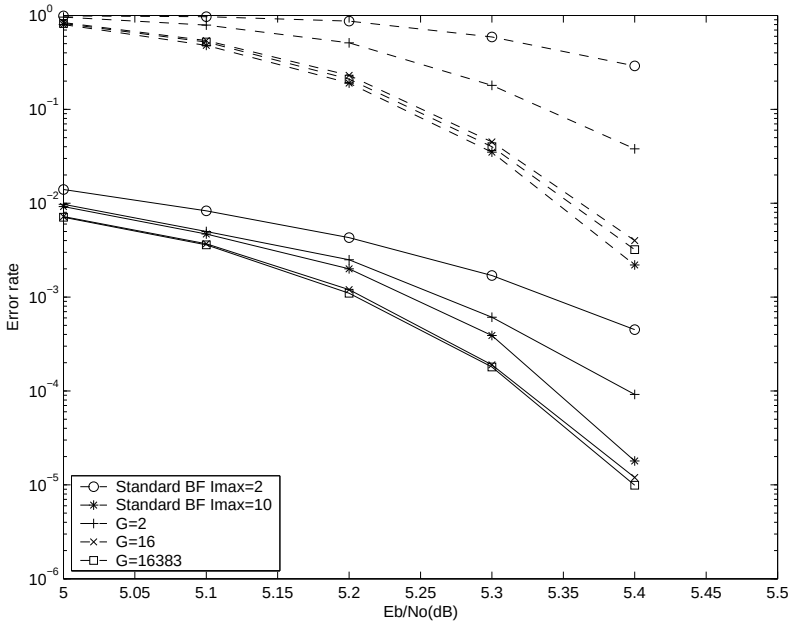

Figure 10: Error rate of the $(16383, 14179)$ EG-LDPC code with the standard BF and group replica shuffled BF algorithm with four subdecoders, for $G = 2, 4, 8, 16, 4095$ and at most 2 iterations.

built-in bounded-error decoder and/or that the possibility of a decoding failure when $n \leq t$ is completely negligible).

The overall error performance P was then obtained by the average

$$P_s = \sum_{n=t+1}^{N} P(n) \binom{N}{n} p_e^n (1 - p_e)^{N-n} \tag{7}$$

For BPSK signaling over AWGN channel, the transition probability $p_e = Q(\sqrt{RE_b/N_0})$, where R is the code rate and E_b/N_0 is the signal to noise ratio per information bit.

Figure 7 depicts the error performance of the standard BF and replica group-shuffled BF decoding methods with 4 subdecoders for decoding the (4095,3367) EG LDPC code with a fixed number of errors. The maximum number of iterations for replica shuffled BF was set to 2.

Using the results from figure 7, we can determine worst-case, best-case, and extrapolated performances of this decoder down to very low error rates. For WER's smaller than 10^{-4}, no reliable evaluation of $P(n)$ was possible, we computed: (a) a worst-case upper bound on (7) by assuming the same $P(n_{min})$ as the smallest simulated for weights n', $t < n' < n_{min}$; (b) a best-case lower bound on (7) by assuming $P(n) = 0$ for weights n', $t < n' < n_{min}$; and (c) a likely-case approximation by extrapolating $P(n')$ for weights n', $t < n' < n_{min}$. Figure 8 depicts these extrapolations. The worst-case upper bound is derived using the horizontal line

in this figure, the best-case lower bound is derived using the vertical line, and the likely-case extrapolation is derived using the curved line.

Figure 9 depicts the performance of the replica group-shuffled BF decoder for the (4095,3367) EG LDPC codes in high SNR regime based on Figure 8. Note that the best-case and likely-case scenarios are nearly identical. More importantly, the worst-case scenario is only slightly worse (less than 1 dB) than the likely-case scenario for BER's between 10^{-10} and 10^{-15}. This means that we can *guarantee* that there will be no significant error floor using this decoding method.

Finally, Figure 10 depicts the error rate of iterative decoding of the $(16383, 14179)$ EG-LDPC code with standard BF and group replica shuffled BF algorithm with four subdecoders, for $G = 2, 16, 4095$. The maximum number of iterations for group replica shuffled BF was set to be 2. We observe that the WER performance of group replica shuffled BF decoding with four subdecoders and maximum number of 2, and group number larger or equal to four, are approximately the same as that of the standard BF with maximum number of 10. These results are similar to those for the $(4095, 3367)$ code.

9 Conclusions

We have described many different decoders and it may be worthwhile to give some final pointers to orient the reader. The major conclusion is that group-shuffled BF decoders and replica group-shuffled BF or QWBF decoders give good performance using a very small number (two) of iterations and four or more groups, comparable or better that of standard decoders with ten or more iterations. The gain obtained by upgrading from BF to QWBF decoders is rather large (probably around .5 dB), but one must quantize the channel output using one bit ("high" versus "low" reliability). Replica group-shuffled decoders using four replicas also have a further performance advantage compared to ordinary group-shuffled decoders, but the gain is not very large. Finally, we have demonstrated that group-shuffled and replica group-shuffled BF decoders will not have a significant error floor. Although it would be harder to demonstrate using importance sampling, there is no reason to expect error floors for QWBF decoders either.

References

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

[2] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399-431, Mar. 1999.

[3] T. Richardson and R. 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] Y. Kou, S. Lin and M. Fossorier, "Low density parity check codes based on finite geometries: a rediscovery and more," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711-2736, Nov. 2001.

[5] S. Lin and D.J. Costello, Jr., "Error Control Coding," Second Edition, 2004.

[6] P.O. Vontobel and R. Koetter, "Lower Bounds on the Minimum Pseudo-weight of Linear Codes," Proc. IEEE International Symposium on Information Theory, 2004.

[7] J. Zhang and M. Fossorier, "Shuffled Belief Propagation Decoding," *Proceedings of 36th Annual Asilomar Conference on Signals, Systems and Computers*, pp. 8-15, Nov. 2002.

[8] H. Kfir and I. Kanter, "Parallel Versus Sequential Updating for Belief Propagation Decoding," submitted to *Physical Review E*, July 2002.

[9] J. Zhang, Y. Wang, M.P.C. Fossorier, and J.S. Yedidia, "Replica Shuffled Iterative Decoding," MERL Technical Report, Dec. 2004.