IMPROVING BER PERFORMANCE OF LDPC CODES BASED ON INTERMEDIATE DECODING RESULTS

Esa Alghonaim, Mohamed Adnan Landolsi, and Aiman El-Maleh
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia

ABSTRACT

The paper presents a novel approach to reduce the bit error rate (BER) in iterative belief propagation (BP) decoding of low density parity check (LDPC) codes. The behavior of the BP algorithm is first investigated as a function of number of decoder iterations, and it is shown that typical uncorrected error patterns can be classified into 3 categories: oscillating, nearly-constant, or random-like, with a predominance of oscillating patterns at high Signal-to-Noise (SNR) values.
A proposed decoder modification is then introduced based on tracking the number of failed parity check equations in the intermediate decoding iterations, rather than relying on the final decoder output (after reaching the maximum number of iterations). Simulation results with a rate ½ (1024,512) progressive edge-growth (PEG) LDPC code show that the proposed modification can decrease the BER by as much as 10-to-40%, particularly for high SNR values.

Index Terms— LDPC codes, BER performance, Belief Propagation iterative decoding, Error correction coding.

1. INTRODUCTION

One of the leading families of error-correcting codes is known as Low Density Parity Check (LDPC) codes, which were first introduced by Gallager in [1]. LDPC codes have been found to rival other state-of-the-art coding families (such as Turbo Codes), and demonstrate performance that can asymptotically achieve the information-theoretic limits, while at the same time having the distinct advantage of low-complexity, near-optimal soft message-passing iterative decoding based on the Belief Propagation (BP) algorithm (also known as the Sum-Product algorithm) [2,3].

One of the advantageous features of LDPC codes is that error detection is always achieved as a by-product of the BP decoding, without the need of extra cyclic redundancy check-CRC codes (unlike other FEC families, like turbo, for example). This is because decoder failures, after reaching a maximum pre-set number of iterations, are always known. From an error correcting perspective, decoder failures clearly lead to a frame (or packet) error, which also implies a number of bit errors. Typically, both frame error rate (FER) and bit error rate (BER) curves are used to characterize LDPC and other coding families. It should be pointed out that in the case of LDPC codes, the number of bit errors for a given frame error occurrence (after exhausting the maximum number of allowable iterations) can vary considerably with the intermediate iterations. Although most previous works consider performance results for a given number of maximum iterations (usually in the range of 100), a few researchers have addressed this bit error variation aspects, as in [4] where the bit error transitions probabilities are considered. Also, in [5], a modified BP decoding algorithm based on error oscillation is presented.

In this paper, a more thorough categorization of the evolution of bit error patterns as a function of decoder iterations is first presented, particularly highlighting the importance of using intermediate decoder results to minimize BER. Following that, a simple approach for deciding at which stage to output decoded bits is presented. This is based on tracking the number of unsatisfied check node equations, which correlate strongly with increased bit error occurrences.

The rest of this paper is organized as follows. First, in Section 2, a brief review of LDPC codes and their iterative decoding are introduced. In Section 3, we investigate the characteristics of error patterns resulting from BP decoding failures. Then in Section 5, the proposed method for improving BER is presented. Experimental results are given in Section 6, and final conclusions in Section 7.

2. OVERVIEW OF LDPC CODES

LDPC codes are a class of linear block codes that use a sparse, random-like parity-check matrix [1,2]. LDPC codes can also be represented by bi-partite factor graphs having two types of nodes: *variable bit nodes* and *check nodes*, interconnected by edges whenever a given check bit appears in the parity check equation of the corresponding information bit. The iterative message-passing belief propagation algorithm [2,3] can used for decoding LDPC codes, and is shown to achieve optimum performance when the underlying code graph is cycle-free In the following, a brief description of this algorithm is given based on the notation in [2], and this will be

subsequently used in discussing the codes' BER performance characterization.

The elements of a codeword **c** are referred to as bits, and the rows of the parity matrix **H** are designated as checks. The set of bits *l* that participate in check *m* is denoted by $L(m) = \{l : H_{ml} = 1\}$, and the set of checks involving bit *l* is denoted by $M(l) = \{m : H_{ml} = 1\}$. A set $L(m)$ in which bit *l* is excluded is denoted by $L(m) \setminus l$, and likewise $M(l) \setminus m$ refers to $M(l)$ without check *m*. The decoding algorithm has two alternating parts in which soft information q_{ml} and r_{ml} associated with each nonzero element of **H** are iteratively updated. The quantity q_{ml}^i can be thought of as the probability that bit node *l* sends to check node *m* indicating $\Pr(x_l = i)$ with *i=0, 1*. On the other hand, the quantity r_{ml}^i is meant to be the probability that the *m*th check node gathers about the *l*th bit of **x** being $x_l = i$ when the other bits have probabilities given by $\{q_{ml'} : l' \in L(m) \setminus l\}$. The iterative steps of the BP algorithm are summarized next.

1. Initialization: The variables q_{ml}^0 and q_{ml}^1 are initialized to the probabilities p_l^0 and p_l^1 that code bit x_l is 0 or 1, respectively. Then, for every non-zero entry in the matrix **H**, we have the following: $q_{ml}^i = p_l^i$.

2. Horizontal Step: Each check node *m* gathers all information q_{ml}^i, and updates the belief on the *l*th bit based on other bits that participate in check equation *m*.

$$r_{ml}^i = \sum_{x_{l'}: l' \in L(m) \setminus l} P(\{x_{l'}\} \mid x_l = i, \mathbf{H}) \times \prod_{l' \in L(m) \setminus l} q_{ml'}^{x_{l'}} \qquad (1)$$

for *i=0,1*. The exclusion of the term q_{ml} in the computation of r_{ml} is necessary in order to only keep "extrinsic" information about the *l*th bit from the *m* parity check equation. The function $P(\{x_{l'}\} \mid x_l = i, \mathbf{H})$ is an indicator function given by:

$$P(\{x_{l'}\} \mid x_l = i, \mathbf{H}) = \begin{cases} 1, if \sum \oplus \{x_{l'}, i\} = 0 \\ 0, otherwise \end{cases} \qquad (2)$$

where $\sum \oplus$ represents modulo-2 summation.

3. Vertical Step: The computed values q_{ml}^i are used to update information that each bit node *l* propagates back to check node *m*. In doing so, the information coming from check node *m* itself is not be included, giving

$$q_{ml}^i = \alpha_{ml} p_l^i \prod_{m' \in M(l) \setminus m} r_{m'l}^i \qquad (3)$$

for *i=0,1*. The term α_{ml} is used as a normalization factor to ensure $q_{ml}^0 + q_{ml}^1 = 1$. Each bit *l* collects probability information from the check nodes that connect to it, and updates its a posteriori probabilities (APP):

$$q_l^i = \alpha_l p_l^i \prod_{m \in M(l)} r_{ml}^i \qquad (4)$$

for *i=0,1*. The term α_l is chosen such that $q_l^0 + q_l^1 = 1$. It is noted that two parts contribute to the APP: intrinsic information from p_l, and an extrinsic one from the r_{ml} terms. After each vertical-horizontal iteration, a hard decision is made on each bit APP q_l, resulting in a tentative decoded vector $\hat{c}$. This is then used to decide whether to stop the decoding algorithm as outlined next.

4. Stopping Criterion: The decoded vector $\hat{c}$ is used to check if $H\hat{c} = 0$, in which case it is declared as a valid codeword. If not, iterative decoding continues until it is eventually successful, or a preset number of maximum iterations is reached.

An illustration of the iterative LDPC decoding algorithm is shown in Figure 1. A log-domain implementation (omitted here for brevity) of this algorithm can also used to further reduce computational requirements.

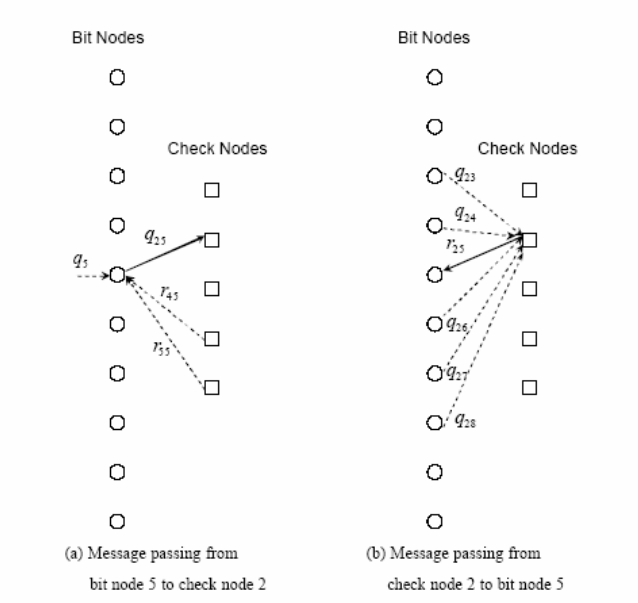

Figure 1: Illustration of LDPC Decoding by BP algorithm.

3. ERROR PATTERNS OF BP-LDPC DECODING

As a first step to characterize the BER performance of LDPC BP decoding, we performed extensive simulations with a rate ½ (1024,512) progressive-edge-growth (PEG) LDPC code [6] over AWGN channels for various SNR values. We point out here that we also considered other random-like LDPC code constructions as well, and

similar conclusions were observed. Hence, for brevity, we only present results with the PEG LDPC code example. A block is considered to be in error if the maximum decoding iteration is reached without satisfying the check equations, i.e. the syndrome $\hat{\bar{c}} \cdot H$ is non-zero. With the help of simulation results, frame errors could be classified into three patterns, described as follows:

I- *Oscillating error pattern*: with a nearly periodic change between maximum and minimum values. An important feature of this error pattern is the high variation in bit error count as a function of decoding iteration number.

II- *Nearly-constant error pattern*: where the bit error count becomes quasi-constant after only a few decoding iterations.

III-*Random-like error pattern:* where the error count evolution follows a random shape, characterized by low variation range.

Figure 2 shows one example for each of the three error patterns obtained from simulation results of a (1024, 512) LDPC code.

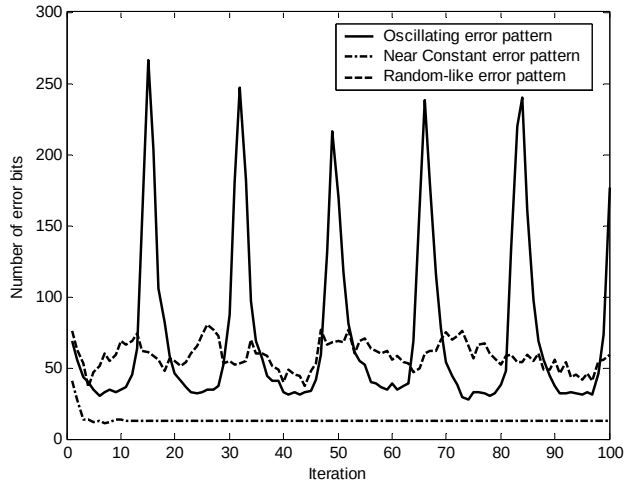

Figure 2. Illustration of the three error patterns

It is of interest to investigate the relative fraction of these error patterns, and this is illustrated in Table 1 for the (1024,512) PEG code that we consider in this paper. It is seen that that the oscillating error pattern increases with SNR, and becomes clearly dominant around 3dB. Since the oscillating error pattern has maximum variation in the number of uncorrected codeword bits (i.e. maximum peak to peak difference), it is therefore expected that this error pattern that will be most critical for optimizing overall BER performance. For example, as shown in Figure 2, if we set the number of maximum iterations to 60, then decoding failure gives around 30 bits in error. On the other hand, setting the maximum iterations to 50 results in nearly 210 bit errors. This observation motivates the search for a way to get the best intermediate error bits for a given frame decoding error. In the following section, we further discuss this idea in detail.

Table 1. Percentage of error patterns among total frame errors

	Error pattern type		
SNR (dB)	Oscillating	Near-constant	Random-like
2.25	4 %	0 %	96 %
2.50	12 %	0 %	88 %
2.75	24 %	1 %	75 %
3.00	63 %	3 %	34 %

4. PROPOSED IMPROVEMENT

The proposed improvement to the BP decoding algorithm is based on the use of intermediate decoding results. For each decoder iteration, we compute the number of bits in error and we store the estimated bit values corresponding to the minimum number of bits in error. As an example, consider the decoding of a block that gave the oscillating pattern shown in Figure 1. Iteration 74 corresponds to minimum number of bits in error. In this example, the proposed algorithm returns estimated bits ($\hat{c}_i$) resulting from iteration 74 (not from the last iteration, as with common decoding). However, from a practical perspective, a normal question arises, namely: since we do not know the correct values of bits, how we decide when to pick the intermediate results (in order to achieve minimum bit error count)?. In this work, our extensive numerical results show that, by tracking the evolution of the failed check equations (check nodes in the bipartite graph), we can get a very good indication on how to optimally produce bit error estimates. Indeed, it is found that there is a strong correlation between the non-zero entries in the syndrome $H\hat{c}$ and the number of bit errors. Figure 3 illustrates this fact for the case of oscillating error patterns (as discussed previously), and similar observations also applied for the other cases as well.

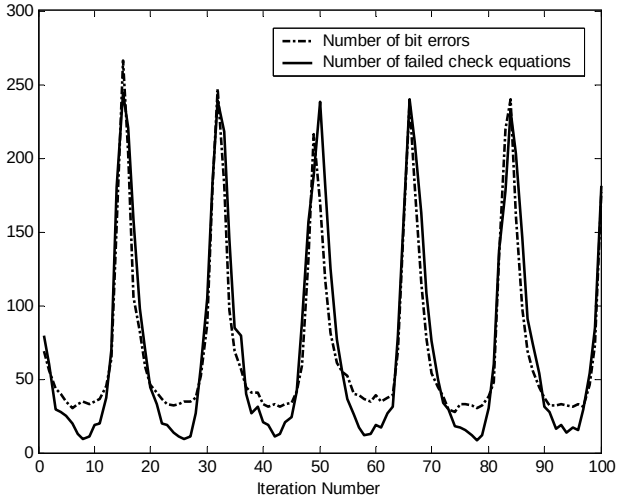

Figure 3. Correlation between uncorrected codeword bits and number of failed parity check equations

To summarize, a formal description of the proposed method to improve the BP decoding based on intermediate decoding iteration results is given below:

BP-LDPC (conventional) decoding
Initialize variable nodes
Loop
Update check and variable nodes
Compute estimated variable nodes vector $\hat{c}$
Compute syndrome vector: $H\hat{c}$
Until $H\hat{c}$ =0 or maximum iterations reached
Return $\hat{c}$

BP-LDPC decoding with the proposed improvement
Initialize variable nodes
Set Minimum = number of check nodes
Loop
Update check and variable nodes
Compute estimated variable nodes vector $\hat{c}$
Compute syndrome vector: $H\hat{c}$
Check Errors = number of non-zero elements in $H\hat{c}$
If Check Errors < Minimum then
Minimum = Check Errors
$\hat{m} = \hat{c}$
Until Check Errors = 0 or maximum iterations reached
Return $\hat{m}$

5. NUMERICAL RESULTS

We performed extensive simulations to verify the effectiveness of the proposed idea. Since simulation of LDPC is time consuming, especially at high SNR, a parallel computing simulation platform was developed to run the LDPC simulations on 130 nodes on a departmental LAN network.

Figure 4 shows the simulation results for a PEG code of block size 1024 and a rate of ½. This code, free of 4 and 6 cycles and with minimized 8 cycles [6], was chosen as a representative example, but similar observations applied with other constructions, as will be reported in future work. From Figure 4, it is seen that the difference in BER achieved from the proposed method increases with SNR. This is because the percentage of the oscillating patterns increases as SNR increases, as shown in Table 1.

6. CONCLUSION

The paper presented a method to improve the residual BER level in belief propagation decoding of LDPC codes. A thorough categorization of the evolution of bit errors as a function of decoder iterations was presented, and results showed that uncorrected error patterns can be classified into 3 categories: oscillating, nearly-constant, or random-like, with a predominance of oscillating patterns at high SNR.

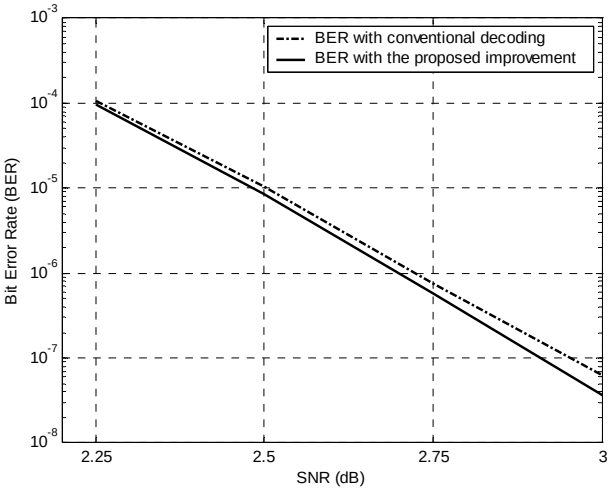

Figure 4. BER results for the (1024,512) PEG LDPC code, with and without the proposed improvement.

These observations highlighted the importance of using intermediate decoder results to minimize BER. A simple approach for deciding at which stage to output decoded bits was introduced based on tracking the number of unsatisfied check node equations, which correlate well with bit error occurrences. Numerical results with rate ½, block size 1024 PEG code were used to demonstrate the effectiveness of the proposed method, and it was found that BER reduction of to 40% can be achieved, for SNR levels around 3dB.

ACKNOWLEDGMENT

The authors thank King Fahd University of Petroleum & Minerals for support of this work under project no. EE/DENSITY/387.

REFERENCES

[1] R. G. Gallager, "Low Density Parity-Check Codes". MIT Press, Cambridge, MA, 1963.

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

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

[4] S.H. Lee et al, "Bit probability transition characteristics of LDPC code," *in Proc. 10th Int'l. Conf. on Telecommunications* 2003, *ICT'03*, vol.1, pp.553–557, 2003.

[5] S. Gounai, T. Ohtsuki, and T. Kaneko, "Modified Belief Propagation Decoding Algorithm for Low-Density Parity Check Code Based on Oscillation," *in Proc. IEEE Vehic. Tech. Conf. Spring 2006*. Vol. 3, pp. 1467-1471.

[6] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE GLOBECOM 2001*, San Antonio, TX, Nov. 2001, pp. 995–1001.