

# A List Decoding Algorithm for Short Low-Density Parity-Check Codes

T.M.N. Ngatched and F. Takawira

School of Electrical, Electronic and Computer Engineering

University of KwaZulu-Natal, Durban, 4041, South Africa

Tel: +27 (031) 260 2736

Fax: +27 (031) 260 2740

Email: [ngatchedt@ukzn.ac.za](mailto:ngatchedt@ukzn.ac.za), [ftakaw@ukzn.ac.za](mailto:ftakaw@ukzn.ac.za)

**Abstract—** In this paper, a list decoding algorithm for low-density parity-check (LDPC) codes is presented. The algorithm uses a modification of the simple Gallager bit-flipping algorithm to generate a sequence of candidate codewords iteratively one at a time using a set of test error patterns based on the reliability information of the received symbols. It is particularly efficient for short block LDPC codes, both regular and irregular. Computer simulation results are used to compare the performance of the proposed algorithm with other known decoding algorithms for LDPC codes, with the result that the presented algorithm offers excellent performances. Performances comparable to those obtained with iterative decoding based on belief propagation can be achieved at a much smaller complexity.<sup>1</sup>

## I. INTRODUCTION

Low-density parity-check (LDPC) codes, originally invented by Gallager in 1962 [1], forgotten for decades, rediscovered by many [2, 3], and their performances have been the subject of much recent experimentation and analysis. The interest in these codes stems from their near Shannon limit performance, their simple descriptions and implementations, and their amenability to rigorous theoretical analysis. LDPC codes have become strong competitors to turbo codes [4 – 7] for error control in many communication and digital storage systems where high reliability is required.

LDPC codes can be decoded using various decoding methods, ranging from low to high complexity and from reasonably good to very good performances. These decoding methods include: One-step majority-logic (MLG) decoding [8], Gallager's bit flipping (BF) decoding [1], weighted MLG decoding (WMLG) [9], weighted BF decoding (WBF) [10], Bootstrap decoding (BD) [11], Modified WBF (MWBF) [12] the algorithm proposed in [13] which we call improved weighted bit flipping (IWBF), the algorithms in [24], a posteriori probability (APP) decoding [1], and iterative decoding based on belief propagation (IDBP) (commonly known as sum-product algorithm (SPA)) [14 -16]. IDBP offers the best performance, but has the largest complexity. Each decoding iteration requires many real number addition, subtraction, multiplication, division, exponential and logarithm operations. Some approximations of IDBP can be used which can reduce the complexity with small loss in performance. In [17], a reduced complexity IDBP has been proposed for LDPC codes and is referred to as *uniformly most powerful* (UMP) BP-based algorithm. Unfortunately,

the UMP BP-based algorithm does not work well for codes with check sums of larger weight. A normalized IDBP algorithm, which can improve the UMP BP-based algorithm by normalization, is proposed in [18]. However, this algorithm also requires real division operations.

The above decoding methods provide a wide range of trade-offs among decoding complexity, decoding speed, and error performance. MLG and BF decoding belong to hard-decision decoding methods. APP and IDBP/SPA decoding are soft-decision schemes. They require extensive decoding computation but they provide the best error performance. WMLG, WBF, and IWBF decoding are “between” hard- and soft-decision decoding methods. They improve the error performance of the MLG and BF decoding with some additional computational complexity, and offer a good trade-off between error performance and decoding complexity. In particular, simulation results on finite geometry LDPC codes in [12] show that, with IWBF, performances less than 1 dB away from IDBP can be achieved.

In this paper, a new decoding algorithm for LDPC codes is presented. The algorithm is similar to the decoding strategies presented in [20 – 22]. It is based on the observation that, at high signal-to-noise ratio (SNR), the most likely codeword is located, with a very high probability, in a sphere around the received vector. Therefore, the idea is to generate a list of codewords containing the closest codewords to the received vector. In essence, error patterns are formed based on the received vector. Test sequences are then created by adding the errors patterns to the hard decision sequence of the received vector (using modulo-2 addition). Each test sequence is then decoded using a modification of the simple Gallager BF algorithm [1] as a hard-decision (HD) decoder to produce, if possible, a candidate codeword. When a candidate codeword is generated, it is tested based on an optimality condition. If it satisfies the optimality condition, then it is the most likely (ML) codeword and decoding stops. If it fails the optimality test, a new candidate is generated. The process continues until either the ML codeword is found or a stopping criterion is met. The complexity of the proposed algorithm is probabilistic and depends on the property of the received sequence. It generates a larger set of candidates when a noisy sequence is received and a smaller set of candidates when a clean sequence is received.

The remainder of the paper is organized as follows: In the next section, notations, definitions, and the proposed algorithm is presented. In Section III, we evaluate the complexity of the algorithm. Some simulation results are provided in Section IV to illustrate the performance of the

<sup>1</sup> This work was partially supported by Alcatel and Telkom South Africa as part of the centers of Excellence Programme.

proposed algorithm and, finally, conclusions are drawn in Section V.

## II. THE DECODING ALGORITHM

### A. Preliminaries

Let  $C$  be a binary block code of length  $n$ , dimension  $k$  and minimum distance  $d$ . Suppose  $C$  is used for error control over a binary-input additive white Gaussian noise (BIAWGN) channel using binary phase-shift-keying (BPSK) signaling with unit energy. Let  $\mathbf{c} = (c_0, c_1, \dots, c_{n-1})$  be the transmitted codeword. For BPSK transmission,  $\mathbf{c}$  is mapped into a bipolar sequence  $\mathbf{x} = (x_0, x_1, \dots, x_{n-1})$  by  $\mathbf{x} = S(\mathbf{c})$ , where  $S$  is a function defined as

$$S(c_i) = \begin{cases} -1 & \text{if } c_i = 0 \\ +1 & \text{if } c_i = 1 \end{cases}. \quad (1)$$

Let  $\mathbf{y} = (y_0, y_1, \dots, y_{n-1})$  be the soft-decision received sequence at the output of the receiver matched filter. For  $0 \leq i \leq n-1$ ,  $y_i = x_i + n_i$  where  $n_i$  is a Gaussian random variable with zero mean and variance  $N_o/2$ . An initial binary hard decision of the received sequence,  $\mathbf{y}^H = (y_0^H, y_1^H, \dots, y_{n-1}^H)$ , is determined as follows

$$y_i^H = \begin{cases} 1, & \text{if } y_i \geq 0 \\ 0, & \text{if } y_i < 0 \end{cases}. \quad (2)$$

We define a reliability vector  $\boldsymbol{\alpha}$  of length  $n$ . The components  $\alpha_i$  of  $\boldsymbol{\alpha}$  are defined as the bit-log-likelihood ratio

$$\alpha_i \triangleq \ln \frac{P(c_i = 1 | y_i)}{P(c_i = 0 | y_i)}. \quad (3)$$

The absolute value of  $\alpha_i$ ,  $|\alpha_i|$ , is called the reliability of the initial decision  $y_i^H$ . For the AWGN channel, a simple measure of the reliability,  $|\alpha_i|$ , of a received symbol  $y_i$  is its magnitude,  $|y_i|$ . The larger the magnitude  $|y_i|$  is, the larger the reliability of the hard-decision digit  $y_i^H$  is. It is known that the maximal-likelihood decoder selects  $\mathbf{c}_m$  as an estimate if  $S(\mathbf{c}_m) \cdot \boldsymbol{\alpha} \geq S(\mathbf{c}_{m'}) \cdot \boldsymbol{\alpha}$  for all  $m' \neq m$  [19]. The objective becomes to find a codeword maximizing the dot product  $S(\mathbf{c}_m) \cdot \boldsymbol{\alpha}$ .

Define the following index sets

$$D_0(\mathbf{c}) \triangleq \{i | c_i = y_i^H, 0 \leq i \leq n-1\}, \quad (4)$$

$$D_1(\mathbf{c}) \triangleq \{i | c_i \neq y_i^H, 0 \leq i \leq n-1\} \\ = \{0, 1, \dots, n-1\} \setminus D_0(\mathbf{c}). \quad (5)$$

Let  $m(\mathbf{c}) = \|D_1(\mathbf{c})\|$ , where  $\|D\|$  denotes the cardinal number of  $D$ . For convenience, we assume that the components of all vectors are numbered in the order of increasing reliability, so that if  $i \leq j$ ,  $|\alpha_i| \leq |\alpha_j|$ . It is shown in [20, 21] that if a codeword  $\mathbf{c}$  satisfies

$$l(\mathbf{y}, \mathbf{c}) \leq \sum_{i=0}^{d-m(\mathbf{c})} |\alpha_{D_0(\mathbf{c})}|, \quad (6)$$

then there is no codeword which is more likely than  $\mathbf{c}$ . In (6),  $l(\mathbf{y}, \mathbf{c})$  is defined by

$$l(\mathbf{y}, \mathbf{c}) = \sum_{i \in D_1(\mathbf{c})} |\alpha_i|, \quad (7)$$

and is called the correlation discrepancy of  $\mathbf{c}$ . Equation (6) is referred to as optimality test criterion (OTC).

### B. Algorithm

A binary LDPC code is completely described by its sparse binary parity-check matrix  $\mathbf{H}$ . For an  $(n, k)$  LDPC code,  $\mathbf{H}$  has  $n$  columns and  $m \geq n-k$  rows. For a regular LDPC code,  $\mathbf{H}$  has a constant column weight  $\gamma$  and a constant row weight  $\rho$ , where  $\rho$  and  $\gamma$  are small compared with the code length. A LDPC code is said to be irregular if its parity-check matrix  $\mathbf{H}$  has varying column weights and varying row weights. Suppose a binary  $(n, k)$  LDPC code is used for error control over a BIAWGN channel using BPSK signaling.

Let  $\mathbf{e}^{(r)} = (e_0^{(r)}, e_1^{(r)}, \dots, e_{n-1}^{(r)})$  be a binary vector of length  $n$ . The set of vectors  $\{\mathbf{e}^{(r)}\}$ ,  $r = 0, 1, 2, \dots$ , contains all the binary combinations in the least reliable positions of the received vector  $\mathbf{y}$ . Since the components of the vectors are numbered in the order of increasing reliability,

$$\sum_{j=0}^{n-1} e_j^{(r)} 2^j = r, \quad (8)$$

where  $r$  is the value of the binary vector  $\mathbf{e}^{(r)}$  in its normal representation form. The algorithm decodes  $\mathbf{y}^H \oplus \mathbf{e}^{(r)}$  in the order  $r = 0, 1, 2, \dots$ , where  $\oplus$  denotes a modulo-2 addition. The decoding stops when: (1) a codeword satisfying the OTC is found; (2) no better codeword can be generated; (3) a predefined maximum number of candidate codewords,  $N_{c_{\max}}$ , has been generated.

To reduce the complexity of the algorithm, a simple test is performed before decoding of each test sequence to prevent the HD decoding to result in a codeword that has already been generated. If the test fails, the test sequence is not decoded. Furthermore, the sequence  $\{\mathbf{e}^{(r)}\}$ ,  $r = 0, 1, 2, \dots$ , is generated by two functions  $f_1$  and  $f_2$  which transfer  $\mathbf{e}^{(r)}$  to  $\mathbf{e}^{(r+1)}$  as follows

$$f_1(\mathbf{e}^{(r)}) = \mathbf{e}^{(r+1)}, e_i^{(r+1)} = \begin{cases} 0, & i < \lambda_1 \\ 1, & i = \lambda_1 \\ e_i^{(r)}, & i > \lambda_1 \end{cases}, \quad (9)$$

$$f_2(\mathbf{e}^{(r)}) = \mathbf{e}^{(r+1)}, e_i^{(r+1)} = \begin{cases} 0, & i < \lambda_2 \\ 1, & i = \lambda_2 \\ e_i^{(r)}, & i > \lambda_2 \end{cases} \quad (10)$$

where,

$$\lambda_1 = \min(i | e_i^{(r)} = 0, 0 \leq i \leq n-1), \quad (11)$$

$$\lambda_2 = \min(i | e_i^{(r)} = 0 \text{ and } e_{i-1}^{(r)} = 1, 0 \leq i \leq n-1). \quad (12)$$

Note that  $f_1$  is a simple binary counting function. The function  $f_2$ , introduced in [22], avoids trials by which the decoding never outputs a codeword maximizing the likelihood function.

For each  $\mathbf{e}^{(r)}$ , define a vector

$$\tilde{\mathbf{e}}^{(r)} = (\tilde{e}_0^{(r)}, \tilde{e}_1^{(r)}, \dots, \tilde{e}_{n-1}^{(r)}), \quad (13)$$

such that

$$\tilde{e}_i^{(r)} = \begin{cases} e_i^{(r)}, & i < \mu \\ 1, & \mu < i < \mu + \delta \\ 0, & i \geq \mu + \delta \end{cases}, \quad (14)$$

where

$$\mu = \max\{i \mid e_i^{(r)} = 1, 0 \leq i \leq n-1\} \quad (15)$$

for  $r \geq 1$  and  $\mu = 0$  for  $r = 0$ . Then we define  $\Delta_r$  by

$$\Delta_r = \max_{0 \leq i \leq r} S(\mathbf{x}^{(i)}) \cdot \boldsymbol{\alpha}. \quad (16)$$

The steps of the algorithm are described in detail below, where  $w_H(\mathbf{a})$  denotes the Hamming weight of  $\mathbf{a}$  and  $\hat{\mathbf{c}}$  the estimation of the transmitted codeword, respectively.

- Step 1: Initialization: Set the number of decoded codewords  $N_c = 0$ . Calculate  $\mathbf{y}^H$ . Set  $r = 0$ ,  $\Delta_{-1} = -\infty$ , and the list containing the decoded codewords to  $\Omega = \emptyset$ .
- Step 2: Compute  $\mathbf{z}^{(r)} = \mathbf{y}^H \oplus \mathbf{e}^{(r)}$ . Compute  $S(\mathbf{y}^H \oplus \tilde{\mathbf{e}}^{(r)}) \cdot \boldsymbol{\alpha}$ . If  $\Delta_{r-1} < S(\mathbf{y}^H \oplus \tilde{\mathbf{e}}^{(r)}) \cdot \boldsymbol{\alpha}$ , then continue to Step 3), otherwise go to Step 9).
- Step 3: Check the Hamming distances between  $\mathbf{z}^{(r)}$  and all the codewords already generated. If all the Hamming distances are greater than  $\delta$ , then continue to Step 4). Otherwise set  $\mathbf{e}^{(r+1)} = f_1(\mathbf{e}^{(r)})$ , and go to Step 2).
- Step 4: Find the number of unsatisfied parity-check sums for each bit, denoted  $p_i$ ,  $0 \leq i < n$ . Find the set  $B$  of bits for which  $p_i$  is the largest.
- Step 5: Flip the bits of  $\mathbf{z}^{(r)}$  belonging to the set  $B$ . Let  $\mathbf{v}^{(r)}$  be the resulting vector. If  $\mathbf{v}^{(r)}$  is a codeword (i.e. if  $\mathbf{v}^{(r)} \cdot \mathbf{H}^T = \mathbf{0}$ ), continue to Step 6). Otherwise, set  $\mathbf{e}^{(r+1)} = f_1(\mathbf{e}^{(r)})$  and go to Step 2).
- Step 6:  $\Omega \leftarrow \Omega \cup \mathbf{v}^{(r)}$ ,  $N_c \leftarrow N_c + 1$ . Compute  $\Delta_r$ . If  $\Delta_r > \Delta_{r-1}$ , continue to Step 7). Otherwise go to Step 8).
- Step 7:  $\hat{\mathbf{c}} = \mathbf{v}^{(r)}$ . If  $\mathbf{v}^{(r)}$  satisfied the OTC (i.e. equation (6)), go to Step 10), otherwise continue to Step 8).
- Step 8: If  $N_c = N_{c_{\max}}$ , go to Step 10). Otherwise set  $\mathbf{e}^{(r+1)} = f_1(\mathbf{e}^{(r)})$  and go to Step 2).
- Step 9: If  $w_H(\mathbf{e}^{(r)}) = 1$ , go to Step 10). Otherwise set  $\mathbf{e}^{(r+1)} = f_2(\mathbf{e}^{(r)})$  and go to Step 2).
- Step 10: Stop the decoding. An estimate of the codeword is  $\hat{\mathbf{c}}$ .

The test in Step 3) is to prevent the HD decoding in Steps 4) and 5) to result in a codeword that has already been generated. The parameter  $\delta$  here is a design parameter

which should be chosen to optimize the error performance while minimizing the decoding delay and complexity. It is an indication of the error correcting capability of the code using a HD decoder. It is, unfortunately, difficult to quantify the optimum value of  $\delta$ . This value depends on the code parameters and can be determined through simulation. Once  $\delta$  is found,  $d$  is set to  $2\delta + 1$ .

### III. COMPUTATIONAL COMPLEXITY

The computational complexity of the proposed algorithm depends on the number of  $\mathbf{e}^{(r)}$ 's generated and the number of times one applies hard-decision decoding during the decoding of  $\mathbf{y}$ . Denote the number of times to apply hard-decision decoding as  $N_{HD}$ , the number of  $\mathbf{e}^{(r)}$ 's generated during the decoding of  $\mathbf{y}$  as  $N_r$ , and their averages as  $\bar{N}_{HD}$  and  $\bar{N}_r$  respectively. It is important to note that, for each  $r$ , the HD decoding is applied only when the tests in Steps 2) and 3) are satisfied. The complexity of these steps is  $O(n)$ . All the operations involved in the HD decoding are associated with modulo-2 arithmetic and comparisons. The complexity of the HD decoding can be estimated as  $O(n)$ . Before decoding  $\mathbf{y}$ , we must calculate the reliability vector  $\boldsymbol{\alpha}$  and permute components of vectors in order of increasing reliability. The complexity of doing this is  $O(n \log n)$ . Thus the total complexity to decode  $\mathbf{y}$  is

$$(N_{HD} + N_r)O(n) + O(n \log n), \quad (17)$$

and the average complexity is

$$(\bar{N}_{HD} + \bar{N}_r)O(n) + O(n \log n). \quad (18)$$

For a particular code,  $\bar{N}_{HD}$  and  $\bar{N}_r$  can be evaluated through computer simulations.

### IV. SIMULATION RESULTS

The error performance, in terms of bit-error rate (BER) and frame-error rate (FER) as a function of the normalized signal-to-noise ratio (SNR) per information bit, of the algorithm presented in Section II is compared with some existing algorithms on a number of short LDPC codes. However, the results are only presented for one LDPC code. The code is an irregular LDPC code of length  $n = 64$ , dimension  $k = 44$ , and code-node degree distribution  $\lambda(x) = 0.621814x + 0.270771x^2 + 0.107415x^4$ . The rate of the code is 0.6875. The code is constructed using the progressive edge growth (PEG) algorithm [23]. The PEG construction creates matrices with very large girth and, to the best of our knowledge, yields the best binary LDPC codes at short block length. The following algorithms are simulated: WBF, MWBF, IWBF, IDBP, and the algorithm proposed herein. The maximum number of iterations is set equal to 100 for IDBP, and 20 for the other algorithms. For the proposed algorithm the value of the parameter  $\delta$  is 2. At each SNR value, at least 5000 codewords errors are detected.

The BER and FER performances of the code are depicted in Figs. 1 and 2 respectively for different values of  $N_{c_{\max}}$ . It can be observed that, for the code simulated, no significant improvement in performance is achieved for values of  $N_{c_{\max}}$

greater than 8. In Fig. 3 the average number of codewords generated during the decoding process using the proposed algorithm is shown for different values of  $N_{c_{\max}}$ . It can be observed that at medium to high SNR, the average number of codewords generated is independent of the value of  $N_{c_{\max}}$ . Fig. 4 shows the average complexity of the algorithm in terms of the average number of times the HD decoding is used and the number of error patterns generated during the decoding as a function of the SNR. It can be observed that the decoding complexity required by the algorithm depends on the property of the received sequence. These results show that the algorithm generates a larger set of candidates when a noisy sequence is received and a smaller set of candidates when a clean sequence is received. In Figs. 5 and 6, the performance of the proposed algorithm (with  $N_{c_{\max}} = 8$ ) is compared with the other decoding algorithms. It can be observed that performances similar to those obtained with IDBP can be achieved. It is important to note that the complexity of the proposed algorithm is much less than that of IDBP.

## V. CONCLUSION

In this paper, a new decoding algorithm for LDPC codes is presented. The algorithm generates a list of codewords containing the closest codewords to the received vector by decoding test sequences using a modification of the Gallager BF algorithm. Simulation results show that, performances comparable to those obtained using iterative decoding based on belief propagation can be achieved at a much smaller complexity.

## REFERENCES

- [1] R. G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, MA: MIT Press, 1963.
- [2] M. Sisper and D. A. Spielman, "Expander codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710-1722, Nov. 1996.
- [3] D. J. C. Mackay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electron. Lett.*, vol. 32, pp. 1645-1646, Aug. 1996.
- [4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," *Proc. 1993 IEEE International Conference on Communications*, Geneva, Switzerland, pp. 1064-1070, May 1993.
- [5] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, pp. 1261-1271, Oct. 1996.
- [6] S. Benedetto and G. Montorsi, "Unveiling Turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409-428, March 1996.
- [7] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429-445, March 1996.
- [8] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice Hall, Englewood, New Jersey, 1983.
- [9] V. D. Kolesnik, "Probability decoding of majority codes," *Prob. Peredachi Inform.*, vol. 7, pp. 3-12, July 1971.
- [10] 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, pp. 2711-2736, Nov. 2001.
- [11] A. Noh and A. H. Banihashemi, "Bootstrap decoding of low-density parity-check codes," *IEEE Commun. Lett.*, vol. 6, pp. 391-393, Sept. 2002.
- [12] J. Zhang and M. P. C. Fossorier, "A modified weighted bit-flipping decoding of low-density parity-check codes," *IEEE Commun. Lett.*, 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, pp. 415-421, Mar. 2005.
- [14] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, IT-45, pp. 399-432, March 1999.
- [15] F. R. Kschischang, B. J. Frey and H. -A. Loeliger, "Factor Graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498-519, Feb. 2001.
- [16] R. Lucas, M. Fossorier, Y. Kou, and S. Lin, "Iterative decoding of one-step majority logic decodable codes based on belief propagation," *IEEE Trans. Commun.*, vol. 48, pp. 931-937, June 2000.
- [17] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity-check codes," *IEEE Trans. Commun.*, vol. 47, pp. 673-680, May 1999.
- [18] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 50, pp. 406-414, March 2002.
- [19] T. Y. Hwang "Decoding linear block codes for minimizing word error rate," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 733-737, Nov. 1979.
- [20] T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa, "An efficient maximum-likelihood decoding algorithm for linear block codes with algebraic decoder," *IEEE Trans. Inform. Theory*, vol. 40, pp. 320-327, Mar. 1994.
- [21] D. J. Taipale, M. B. Pursley, "An improvement to generalized minimum distance decoding," *IEEE Trans. Inform. Theory*, vol. 31, pp. 167-172, Jan. 1991.
- [22] T. Kaneko, T. Nishijima, and S. Hirasawa, "An improvement of soft-decision maximum likelihood decoding algorithm using hard-decision bounded-distance," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1314-1319, Jul. 1997.
- [23] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Progressive edge-growth Tanner graph," *IBM Research, Zurich Research Laboratory*, 8803, Rüschlikon, Switzerland.
- [24] T. M. N. Ngatched, M. Bossert, and A. Fahrner, "Two decoding algorithms for low-density parity-check codes" in *Proc. ICC'05*, Seoul, South Korea, vol. 1, pp. 673 - 677, May 2005.

## BIOGRAPHIES

**Telex Magloire Ngatched** holds a MScEng (Cum Laude) from the university of Natal and a PhD degree from the University of KwaZulu-Natal. At present he is a Research Fellow at the University KwaZulu-Natal. His research interests include algebraic coding theory, iterative decoding algorithms, and construction of optimal codes.

**Fambirai Takawira** holds a BScEng with first class honours in Electrical Engineering from Manchester University and a PhD degree from Cambridge, United Kingdom. At present he is a professor of Digital Communications and head of the School of Electrical, Electronic, and computer Engineering, University of Natal.

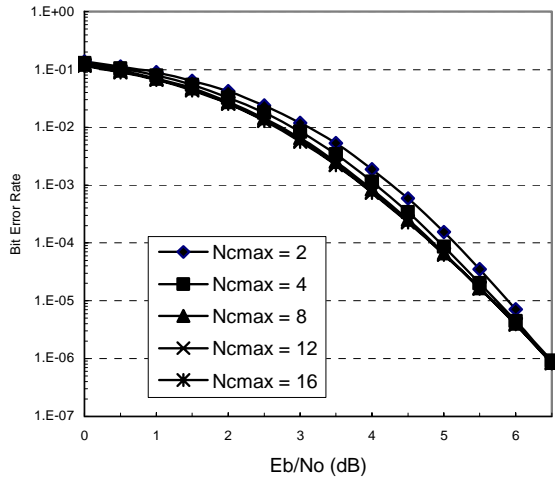


Fig. 1: Bit-error-rate of the proposed decoding algorithm for different values of  $N_{c_{max}}$ .

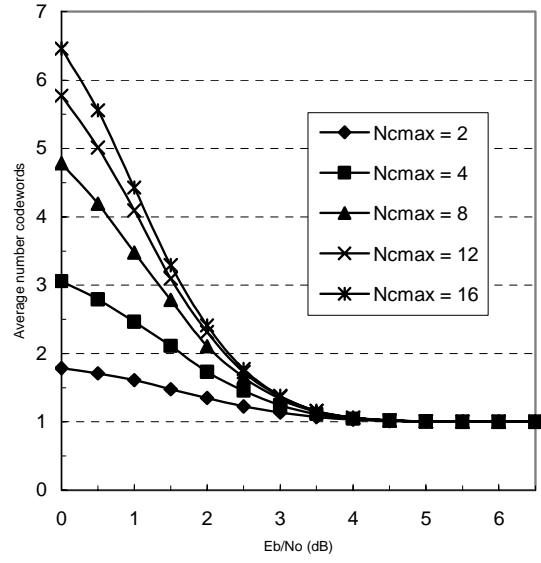


Fig. 3: Average number of codewords generated during the decoding process using the proposed decoding algorithm for different values of  $N_{c_{max}}$ .

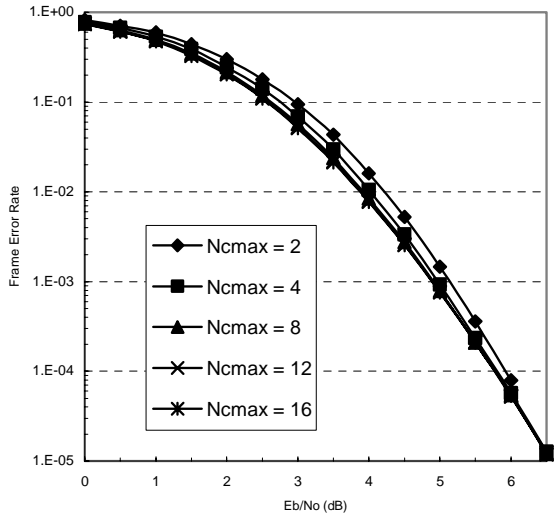


Fig. 2: Frame-error-rate of the proposed decoding algorithm for different values of  $N_{c_{max}}$ .

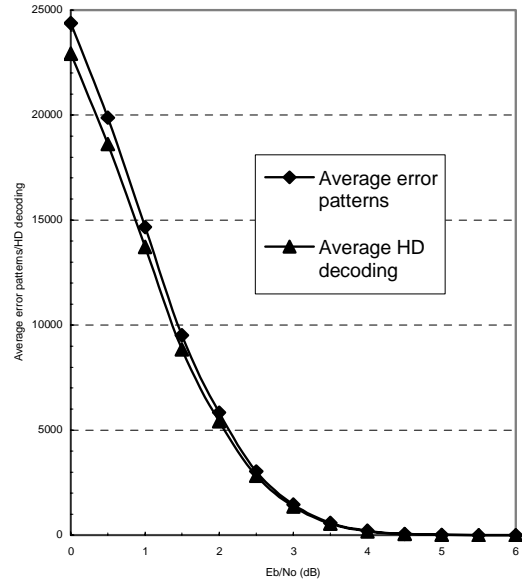


Fig. 4: Average complexity of the decoding algorithm.  $N_{c_{max}} = 8$ .

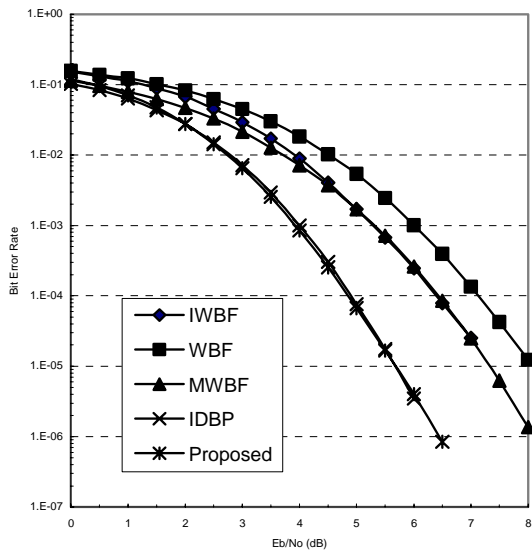


Fig. 5: Bit-error-rate comparison between the proposed algorithm and different decoding algorithms. For the proposed algorithm  $N_{c_{\max}} = 8$ .

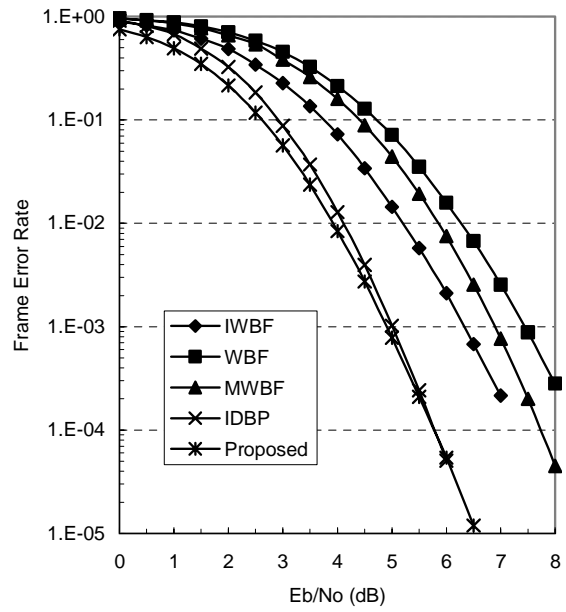


Fig. 6: Frame-error-rate comparison between the proposed algorithm and different decoding algorithms. For the proposed algorithm  $N_{c_{\max}} = 8$ .