

Multi-Threshold Bit Flipping Algorithm for Decoding Structured LDPC Codes

Yuan-hua Liu, Xin-liang Niu, and Mei-ling Zhang

Abstract—A novel bit-flipping (BF) decoding algorithm is proposed for structured low-density parity-check (LDPC) codes. The proposed BF algorithm which is called multi-threshold bit-flipping (MTBF) algorithm, flips multiple bits by using multiple thresholds at each iteration. It only requires some real operations before iterations and requires only logical operations during the iterations. The MTBF algorithm has lower complexity than the weighted bit-flipping algorithm and its modifications. Simulations show that the proposed algorithm has fast convergence speed and good performance.

Index Terms—Bit flipping, iterative decoding, low-density parity-check codes.

I. INTRODUCTION

LOW-DENSITY parity-check (LDPC) codes have recently attracted much attention due to their capacity-approaching performance and low decoding complexity. As an attractive class of LDPC codes, structured LDPC codes have recently attracted much attention due to their easy implementations. Structured LDPC codes can be decoded using hard-decision, soft decision, and hybrid decoding methods. Soft decoding algorithms, such as belief propagation (BP), sum product (SP) algorithm and min-sum (MS) algorithm provide good performance, but require real operations, and are therefore not very suitable for hardware implementations. Instead, hard-decision and hybrid schemes such as bit flipping (BF) [1], weighted BF (WBF) [2] and its variants [3]–[10] have low complexity, and various WBF algorithms offer a good tradeoff between performance, complexity, and decoding speed.

The first BF algorithm was developed by Gallager [1]. Although the bit error rate (BER) performance of the BF algorithm is inferior to that of the BP algorithm, in general, the BF algorithm enables the design of a much simpler decoder, which is easier to implement. To reduce the performance gap between BF decoding and BP decoding, weighted bit flipping (WBF) [2], modified WBF (MWBF) [3], improved MWBF (IMWBF) [4], adaptive weighted multibit-flipping (AWMBF) [5], parallel weighted bit-flipping (PWBF) [6], gradient descent bit flipping (GDBF) [9] and many other algorithms have been proposed.

Manuscript received June 12, 2014; revised November 11, 2014; accepted November 11, 2014. Date of publication November 24, 2014; date of current version February 6, 2015. This work was supported by the National Natural Science Foundation of China under Grants 61201194, 61471294, 61301091, the Natural Science Basic Research Plan in Shaanxi Province of China (2014JQ8318), the Program for young teachers in Xi'an University of Posts and Telecommunications (ZL2014-03), and the National Basic Research Program of China (973 Program) (2012CB328300). The associate editor coordinating the review of this paper and approving it for publication was E. Paolini.

Y. Liu and M. Zhang are with the School of Telecommunication and Information Engineering of Xi'an University of Posts and Telecommunications, Xi'an 710121, China (e-mail: yuanyhliu@163.com).

X. Niu is with Xi'an Branch of China Academy of Space Technology, Xi'an 710000, China.

Digital Object Identifier 10.1109/LCOMM.2014.2373352

© 2014 IEEE.

Although these variants of BF algorithms achieved better performance than the standard BF algorithm, real operations were introduced to the iterative process and the decoding complexity was increased. To offer a better tradeoff between performance, complexity, and decoding speed, a multi-threshold bit-flipping (MTBF) algorithm is proposed in this letter. It only requires some real operations before iterations, and only logical operations during the iterations. It flips multiple bits by using multiple thresholds at each iteration and has low complexity, fast convergence speed and good performance.

The rest of this paper is organized as follows. Section II presents some preliminaries and definitions for the bit-flipping algorithms. The proposed MTBF decoding algorithm is introduced and analyzed in Section III. Simulation results of several BF decoding algorithms are compared in Section IV. Finally, Section V concludes the paper.

II. PRELIMINARIES

A. Notation and Definitions

A binary LDPC code is specified by a sparse parity-check matrix containing mostly zeros and only a small number of ones. A regular binary $(N, K)(\gamma, \rho)$ LDPC code of length N and dimension K has a sparse $M \times N$ parity-check matrix $H = [h_{m,n}]$. The parity-check matrix H has exactly γ '1's in each column and ρ '1's in each row. We denote the set of column indices such that the m -th row is '1' in the parity-check matrix as $\mathcal{N}(m) = \{n : H_{mn} = 1\}$, and the set of row indices such that the n -th column is '1' in the parity-check matrix by $\mathcal{M}(n) = \{m : H_{mn} = 1\}$.

Suppose the code bits are binary phase-shift keying (BPSK) modulated, and transmitted over the the binary-input additive white Gaussian noise (AWGN) channel. A codeword $c = [c_0, c_1, \dots, c_{N-1}]$ is mapped to $x = [x_0, x_1, \dots, x_{N-1}]$ by $x_n = 1 - 2c_n$ before transmission, where $c_n \in \{0, 1\}$. At the receiver, we get the soft-decision received vector $r = [r_0, r_1, \dots, r_{N-1}]$, where $r_n = x_n + v_n$, and v_n are i.i.d. Gaussian random variables with zero mean and variance $\sigma^2 = N_0/2$. Let $z = [z_0, z_1, \dots, z_{N-1}]$ be the binary hard-decision vector obtained from r , where $z_n = 0$, if $r_n \geq 0$, otherwise, $z_n = 1$. Let $s = [s_0, s_1, \dots, s_{M-1}]$ be the vector of syndromes of z , where $s = zH^T$ is computed by the check-sums $s_m = \sum_{n=0}^{N-1} z_n H_{mn}$.

B. Brief Review of Standard BF Algorithm

The standard BF decoding algorithm is a hard decision algorithm based on a very simple concept. In the decoding process of BF algorithm, some unreliable bits which are contained in more than a fixed number T of unsatisfied check-sums are flipped repeatedly until all check-sums are satisfied or the maximum number of iterations is reached. The value

of T should be chosen to optimize the error performance. The standard BF decoding is given as follows.

- Step 1) For $m = 0, 1, \dots, M-1$, compute the syndrome $s_m = \sum_{n=0}^{N-1} z_n H_{mn}$. If all the syndromes are zero, stop the decoding.
- Step 2) For $n = 0, 1, \dots, N-1$, compute the number of nonzero syndromes f_n checking on bit z_n and flip z_n if $f_n > T$.
- Step 3) Repeat Step 1 and 2 until all the syndromes are zero or the maximum number of iterations is reached.

It can be seen that the standard BF decoding requires only logical operations.

III. MULTI-THRESHOLD BIT-FLIPPING ALGORITHM

The standard BF decoding has the lowest complexity, however, hard decision results in performance loss. To improve the performance, WBF decoding and its variants utilize the soft-valued reliability information for the received symbols. However, real operations are introduced in the iterative process and the complexity is increased. To offer a better tradeoff between performance and complexity, [8] proposed an improved BF (IBF) with two decoding thresholds, which improved the performance while maintaining low complexity. It can be seen that the performance can be further improved by using multiple thresholds without additional complexity. In this section, we will introduce a low complexity MTBF algorithm with better performance than the standard BF, IBF and WBF algorithms.

For the AWGN channel, a simple measure of the reliability of a received symbol r_n is its magnitude $|r_n|$. The larger the magnitude $|r_n|$ is, the larger the reliability of the corresponding hard-decision bit z_n is, and a larger decoding threshold should be set to reduce the possibility to be flipped. On the contrary, smaller thresholds should be set for channel observations with smaller magnitudes to increase the possibility to be flipped. Based on this analysis, we propose a multi-threshold bit-flipping decoding to improve the performance of BF algorithm. In the MTBF algorithm, the hard-decision bits are divided into multiple groups based on the magnitudes of the corresponding observations before iterations, and each group has its own threshold. During each iteration, we use multiple thresholds to determine the hard-decision bits whether to be flipped or not.

Based on the principle of majority-logic decoding, if a majority of check equations related to one bit has a value of one, the current bit will be voted down by large majority, and it will be flipped. Thus, the minimum threshold is set as $\lfloor \gamma/2 \rfloor$, where γ is the column weight of H , i.e., the number of check equations corresponding to each bit, the maximum threshold is set as $(\lfloor \gamma/2 \rfloor + \lfloor \gamma/2 \rfloor - 1)$, and the difference between the thresholds of the adjacent groups is set as 1. Thus, the hard-decision bits are divided into $\lfloor \gamma/2 \rfloor$ groups. To reduce the implementation complexity, we group the hard-decision bits uniformly based on their magnitudes in this letter, although the non-uniform division can be used to improve the performance.

In addition, if the bit z_n after flipping is different from the original hard-decision bit, the reliability of the current bit z_n will be reduced and its corresponding threshold should be decreased. If the flipping time of bit z_n is odd, then z_n is dif-

ferent from the original hard-decision bit, and its corresponding threshold T_n should be reduced to $T_n - 1$. Otherwise, threshold T_n should be increased by 1, i.e., $T_n = T_n + 1$.

The procedure of the MTBF decoding is given as follows.

- Step 1) Divide the code bits into multiple groups with different thresholds according to the magnitudes of received symbols and a predetermined parameter α . For $n = 0, 1, \dots, N-1$, if $k\alpha \leq |r_n| < (k+1)\alpha$, $k = 0, 1, \dots, \lfloor \gamma/2 \rfloor - 2$, the threshold is set to be $T_n = \lfloor \gamma/2 \rfloor + k$, else if $|r_n| \geq (\lfloor \gamma/2 \rfloor - 1)\alpha$, the threshold is set to be $T_n = \lfloor \gamma/2 \rfloor + \lfloor \gamma/2 \rfloor - 1$. The parameter α is a positive real number, which can be chosen by empirical testing to optimize the performance.
- Step 2) For $m = 0, 1, \dots, M-1$, compute the check-sum $s_m = \sum_{n=0}^{N-1} z_n H_{mn}$. If all the check-sums are zero, stop the decoding.
- Step 3) For each bit z_n ($n = 0, 1, \dots, N-1$), compute the number of unsatisfied check-sums f_n , and flip z_n if f_n is larger than its relative threshold. If the bit z_n has been flipped an odd number of times, then T_n is reduced by 1, i.e., $T_n = T_n - 1$. Moreover if the flipping time is even, then $T_n = T_n + 1$.
- Step 4) Repeat Step 2) and 3) until all the check-sums are satisfied or the maximum number of iterations is reached.

It is easy to see that the MTBF algorithm requires real comparisons to divide the code bits before iterations and it requires only logical operations during each iteration, i.e., MTBF decoding has the same complexity as the standard BF algorithm. Since multiple bits will be flipped at each iteration, the MTBF algorithm can converge very fast. It can be seen that the MTBF decoding requires a sufficiently number of threshold values to improve the performance, and it is more suitable for the LDPC codes whose column weight is relatively large. Generally, it is much easier for the structured construction methods to design LDPC codes with a relatively large variable node degree than the random constructions. It follows that the proposed MTBF algorithm is more suitable for the structured LDPC codes whose parity check matrix has relatively large column weight.

IV. PERFORMANCE RESULTS

In this section, performance of several EG-LDPC codes under different BF decodings is compared over an AWGN channel with zero mean and variance $\sigma^2 = N_0/2$ via BPSK modulation.

The BER performance of the (1023, 781) EG-LDPC code with the BF, WBF, IBF, IMWBF, GDBF, AWMBF, PWBF, BP and MTBF decodings is shown in Fig. 1. The parameter α for MTBF decoding is set to be 0.1. The maximum numbers of iterations are set to 10, 20, and 50 for different decodings. The MTBF algorithm achieves about 0.25 dB coding gain over WBF and 0.15 dB coding gain over GDBF at $\text{BER} = 10^{-6}$. Although MTBF performs worse than IMWBF, AWMBF and PWBF, it has lower complexity, which will be seen later. In addition, the MTBF decoding with 10 iterations and 20 iterations give almost the same performance, which implies that it converges fast and has short decoding delay. Fig. 2 shows

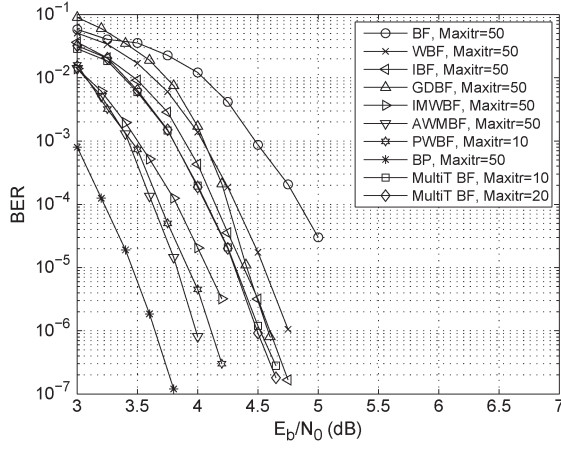


Fig. 1. BER performance of the (1023,781) EG-LDPC code decoded by using various BF algorithms after different iterations.

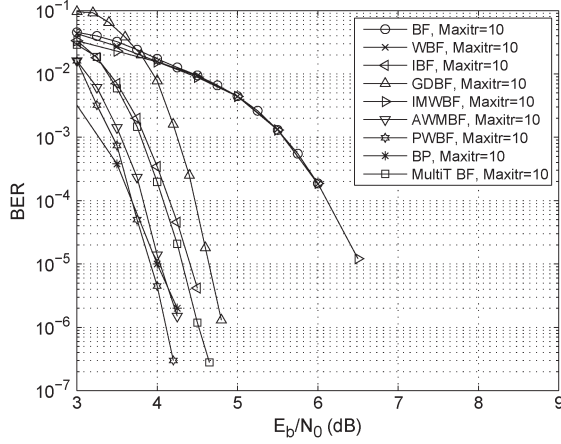


Fig. 2. BER performance of the (1023,781) EG-LDPC code decoded by using various BF algorithms at most 10 iterations.

a comparison of the (1023, 781) EG-LDPC code between the BER of MTBF and the BER of the competing decoders when 10 iterations are used for all of them. From Fig. 1 and Fig. 2, we can see that the BF, WBF and IMWBF algorithms need more iterations to converge and MTBF performs better than IMWBF after 10 iterations. Although MTBF with 10 iterations performs worse than AWMBF with 10 iterations, the gap between them is smaller, which implies that MTBF converge faster than AWMBF.

The optimal value of α in Step 1) for a specific LDPC code can be determined through simulations. Fig. 3 shows the BER performance of the (1023,781) EG-LDPC code decoded by the MTBF algorithm for different choices of α . From Fig. 3, we can see that the error rate is not very sensitive to the value of α , and the optimal value of α is not dependent on the signal noise ratio (SNR).

Fig. 4 shows the BER performance of the (4095,3367) EG-LDPC code decoded by using various BF algorithms and BP decoding after different iterations. The parameter α for MTBF decoding is set to be 0.05. The MTBF algorithm achieves about 0.25 dB coding gain over WBF and 0.18 dB coding gain over IBF at $\text{BER} = 10^{-5}$. Although MTBF performs worse than the IMWBF, AWMBF and PWBF algorithms, it has lower complexity, which can be seen later.

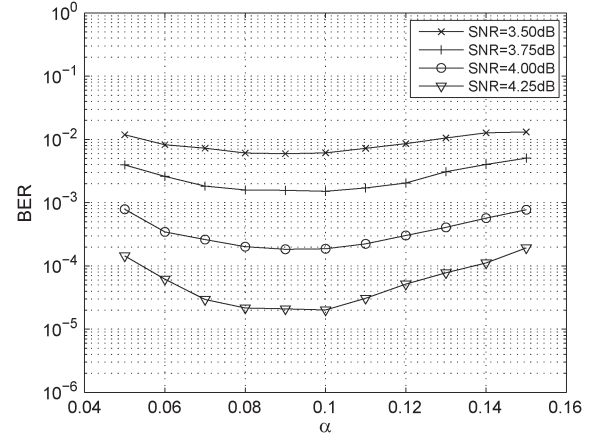


Fig. 3. BER performance of the (1023,781) EG-LDPC code decoded by the MTBF algorithm for different choices of α .

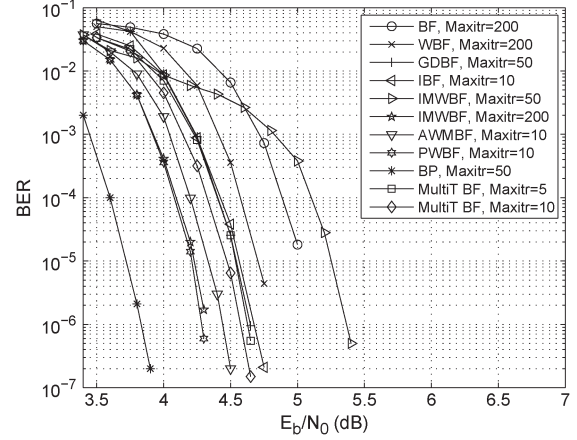


Fig. 4. BER performance of the (4095,3367) EG-LDPC code decoded by using various BF algorithms after different iterations.

Since real addition operation is much more complex than logical operation, the computational complexity is dominated by the number of real operations. It is known that WBF require $M(p-1)$ real operations for initialization, IMWBF and AWMBF require $2M(p-1)$ real operations for initialization. At each iteration, WBF and IMWBF need about $(N-1+p\gamma)$ real operations, and AWMBF needs $(N+L^{(k)}p\gamma)$ real operations, where $L^{(k)}$ is the number of bits flipped at k th iteration. The MTBF algorithm requires at most $\lfloor \gamma/2 \rfloor N$ real comparisons for initialization and no real operation is required during the iterations. In addition, the parameter α does not need to be very accurate and the real comparisons of the MTBF algorithm for initialization can be quantized much less accurately than the real operations in various WBF algorithms, thus the MTBF algorithm needs less quantization bits to implement a real operation.

Fig. 5 plots the average number of iterations (dashed line) and average number of real operations (solid line) needed for various algorithms to decode the (1023,781) EG-LDPC code. The maximum number of iterations is set to 50. It can be observed that the MTBF algorithm requires considerably less real operations than the other algorithms, and it converges faster than AWMBF for $E_b/N_0 \geq 3.8$ dB. Since a large amount of real operations will result in long decoding delay, the MTBF algorithm has higher decoding speed.

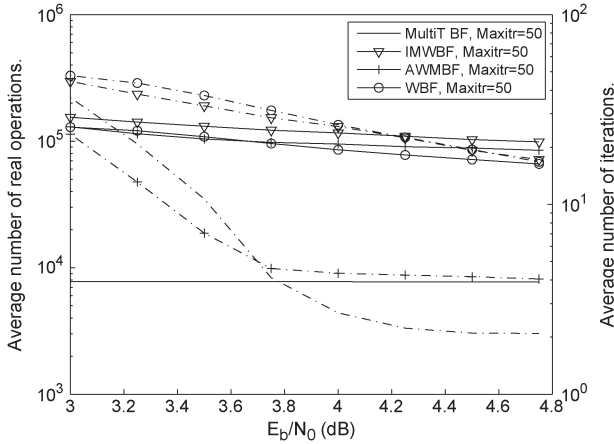


Fig. 5. Average number of real operations (solid) and average number of iterations (dashed) of the (1023,781) EG-LDPC code with various BF algorithms, at most 50 iterations.

In this section, the EG-LDPC codes are chosen to prove the efficiency of the proposed algorithm. It should be noted that there are many other LDPC codes which are suitable for our decoder, if the column weight is large enough.

V. CONCLUSION

A multi-threshold bit-flipping decoding algorithm for structured LDPC codes has been presented. Only logical operations are required during the decoding iterations, and the complexity is similar to the standard BF decoding. In the proposed decoding, bits are determined whether to be flipped or not by multiple thresholds. Bits with larger magnitude are checked

based on larger thresholds, which can reduce the probability of error flipping. If the flipping time is odd, then the corresponding threshold is decreased, otherwise the corresponding threshold is increased, which can improve the performance. Compared to the WBF algorithm, the proposed algorithm reduces computational complexity and decoding time while achieves better performance. It is very competitive for design of high throughput decoders.

REFERENCES

- [1] R. G. Gallager, "Low density parity check codes," *IEEE Trans. Inf. Theory*, vol. IT-8, no. 1, pp. 21–28, Jan. 1962.
- [2] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.
- [3] J. Zhang and M. P. C. Fossorier, "A modified weighted bit-flipping decoding of low-density parity-check codes," *IEEE Commun. Lett.*, vol. 8, no. 3, pp. 165–167, Mar. 2004.
- [4] M. Jiang, C. Zhao, Z. Shi, and Y. Chen, "An improvement on the modified weighted bit flipping decoding algorithm for LDPC codes," *IEEE Commun. Lett.*, vol. 9, no. 9, pp. 814–816, Sep. 2005.
- [5] T.-C. Chen, "Adaptive weighted multibit-flipping decoding of LDPC codes based on ordered statistics," *IET Commun.*, vol. 7, no. 14, pp. 1517–1521, Sep. 2013.
- [6] X. Wu, C. Zhao, and X. You, "Parallel weighted bit-flipping decoding," *IEEE Commun. Lett.*, vol. 11, no. 8, pp. 671–673, 2007.
- [7] T. M. N. Ngatched, M. Bossert, A. Fahrner, and F. Takawira, "Two bit-flipping decoding algorithms for low-density parity-check codes," *IEEE Trans. Commun.*, vol. 57, no. 3, pp. 591–596, Mar. 2009.
- [8] Y. Liu and M. Zhang, "An improved bit-flipping decoding of structured low-density parity-check codes," *J. Beijing Univ. Posts Telecommun.*, vol. 35, no. 4, pp. 116–119, 2012.
- [9] T. Wadayama *et al.*, "Gradient descent bit flipping algorithms for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 6, pp. 1610–1614, Jun. 2010.
- [10] J. Cho and W. Sung, "Adaptive threshold technique for bit-flipping decoding of low-density parity-check codes," *IEEE Commun. Lett.*, vol. 14, no. 9, pp. 857–859, Sep. 2010.