Chapter 101
Improved Min-Sum Decoding Algorithm for Moderate Length Low Density Parity Check Codes

Waheed Ullah, Jiangtao and Yang FengFan

Abstract In this chapter, a new technique to improve the min-sum decoding algorithm for the low density parity check (LDPC) code has been proposed. This technique is based on the magnitude overestimation correction of the variable message by using two normalized factors in all iterations. The variable message is modified with a normalized factor when there is a sign change and with another normalized factor when there is no sign change during any two consecutive iterations. In this way, the algorithm gives a more optimum approximation to the min-sum decoding algorithm. This new technique outperforms for medium and short length codes and for small number of iterations, which make it suitable for practical applications and hardware implementation.

Keywords LDPC codes · Sum product algorithm · Min-sum · Belief propagation · Parity check matrix · Tanner graph

101.1 Introduction

Low density parity check (LDPC) codes, also known as Gallager codes, are a type of linear block codes, first proposed by Gallagar [1] and were scarcely considered in the three decades that followed due to its computational complexity and the limited computational ability of receivers at that time. LDPC was reinvented by

W. Ullah (✉) · Jiangtao · Y. FengFan
College of Electronics and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China
e-mail: uet_waheed@yahoo.com

X. He et al. (eds.), *Computer, Informatics, Cybernetics and Applications*, Lecture Notes in Electrical Engineering 107, DOI: 10.1007/978-94-007-1839-5_101,

Mackay and Neal [2] and have taken considerable attention recently due to its powerful error correcting capabilities and their Shannon limit performance [3–5], with belief propagation decoding algorithm. The name comes from the characteristic of their parity check matrix [6] which contains only a few ones (1s) as compared to the number of zeros (0s). The Belief Propagation (BP) or the sum-product decoding algorithm (SPA) is the best performing algorithm with very high computation complexity and also has dependence on the noise variance. The other popular iterative decoding algorithms which offer extremely low hardware complexity with little performance degradation, is the min-sum algorithm (MSA) [7]. The min-sum algorithm reduces the decoding complexity and is free of noise variance as well. Based on these properties of MSA, several approaches have been made to keep the performance close to SPA while still the decoding complexity is less. Different methods are used to bring the simplified form of the algorithm close in performance to the original BP or sum product algorithm. Some well know approaches are the normalized min-sum (Normalized MSA) and the offset min-sum (Offset MSA) [8]. Density evolution [9, 10], is used to analyze the performance of these decoding algorithms for determining the optimum values of the key parameters either as normalized or offset values. Check message is modified during the iteration process to avoid it from over estimation which brings the min-sum algorithm close in performance to the standard SPA [11] and make them suitable for practical applications and hardware implementation [12]. Due to min-sum, which is a reduced complexity algorithm, LDPC has gained popularity in a wide area of practical applications like local area networks, satellite and inter-satellite communications, deep sea communications etc. Some practical considerations and implementations have been have been proposed in [13, 14].

The choice of the scaling factor in the normalized and offset types of min-sum algorithm is not fixed. The suitable parameters can only be chosen by simulation prior to the implementation. This chapter is focused on the performance improvement to the min-sum decoding algorithm [15] by using two hardware friendly scaling factors. The performance of min-sum decoding algorithm is improved further by adding one additional scaling factor. As in normalized min-sum [9], there is only one scaling factor used which is found by exhaustive search algorithm for better performance.

The proposed improved min-sum algorithm (IMSA) modifies the variable message during the two consecutive iterations. When the signs of the present and previous messages are different then it is modified with one scaling factor, otherwise the message is altered by another scaling factor. The results show that the proposed IMSA is better than normalized MSA and even novel modified min-sum algorithm both in performance and complexity for the normalization factors. The results are also compared with SPA to further validate the significant improvement in performance.

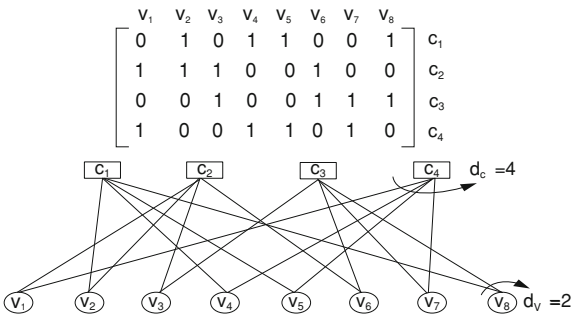

Fig. 101.1 Tanner graph representation of parity check matrix

101.2 Representation of LDPC Codes

101.2.1 Algebraic Representation

LDCP code can be denoted in general as (N, d_v, d_c), where N is the length of the code equal to the number of columns in the parity check matrix, d_c is the number of ones (1s) in a column of a parity check matrix, d_v is the number of ones (1s) in a row of a parity check matrix. LDPC codes can be regular and irregular. If the number of ones (1s) in each row and column of a parity check matrix are the same, it is called regular; and if the number of ones (1s) in each row and column are not the same, it is called irregular. For a regular code, following condition applies

$$M.d_c = N.d_v \tag{101.1}$$

where M and N are the rows and columns of a parity check matrix respectively. The code is valid only if $H.\text{code}^{\text{T}} = 0$, where H is the sparse parity check matrix.

101.2.2 Tanner Graph Representation

The sparse parity check matrix [6] is best represented by a bipartite graphs know as Tanner graphs [14]. Each row of the parity check matrix represents the variable node and each column represents the check node. The one in each row or column shows the connectivity between variable and check nodes. The set of bit nodes connecting to check node m is denoted by $N(m) = \{n|h_{mn} = 1\}$ and the set of check nodes connecting to bit node n is by $M(m) = \{m|h_{mn} = 1\}$. A typical Tanner graph is shown in the Fig. 101.1. This graph is for (6, 2, 4) regular LDPC code.

101.2.3 LDPC Min-Sum Decoding Algorithm

Let $X = \{x1, x2 \ldots..x_n\}$ be the transmitted code after binary phase shift keying (BPSK). It is transmitted over an additive white Gaussian noisy (AWGN) channel.

$$\mathrm{Y} = \mathrm{X} + \mathrm{n} \tag{101.2}$$

where n is an AWGN and $Y = \{y_1, y_2 \ldots..y_n\}$.

Now LDPC min-sum decoding [7, 10, 13], can be stated in the following steps for a parity check matrix $\mathrm{H_{mn}}$, where m is the number of rows and n is the number of columns.

Step 1 Initialization: Set $Ln = Y$ as initial log likelihood ratio (LLR) and for each $(m, n) \in \{(m, n) | h_{mn} = 1\}$

$$V_{mn}^{0} = L_n \tag{101.3}$$

Set the maximum number of iterations (I_{max}) as $i = 0$ to I_{max}.

Step 2 Row processing: bit nodes to check nodes

For $m = 0$ to $M-1$, update C^{i}_{mn} for each $n \in N(m)$

$$C_{mn}^{i} = \prod_{\substack{n' \in N(m) \\ n' \neq n}} \operatorname{sign}\left(V_{mn'}^{i-1}\right) . \min_{\substack{n' \in N(m) \\ n' \neq n}} |V_{mn'}^{i-1}| \tag{101.4}$$

Step 3 Column processing: check nodes to bit nodes

For $n = 0$ to $N-1$, update

$$\tilde{L}_n^i = L_n + \sum_{m \in M(n)} C_{mn}^{i} \tag{101.5}$$

Now updating V^{i}_{mn} for each $m \in M(n)$

$$V_{mn}^{i} = L_n^i - C_{mn}^{i} \tag{101.6}$$

Step 4 Hard Decision:

$$\hat{x}_n = \begin{cases} 0, & \text{for} \quad \tilde{L}_{n>0} \\ 1, & \text{for} \quad \tilde{L}_{n\leq 0} \end{cases} \tag{101.7}$$

Step 5 Stop condition: If the parity check equation is satisfied i.e.

$$H.(\hat{x} .\hat{x}_2 \ldots\ldots\ldots .\hat{x}_n)^T = 0 \tag{101.8}$$

Or maximum iteration (I_{max}) is reached then terminate the decoding or otherwise $i = i + 1$ and go back to step 2.

The message passing between check nodes and variable nodes in Steps 2 and 3 can also be represented in a graphical way as shown in Figs. 101.2 and 101.3.

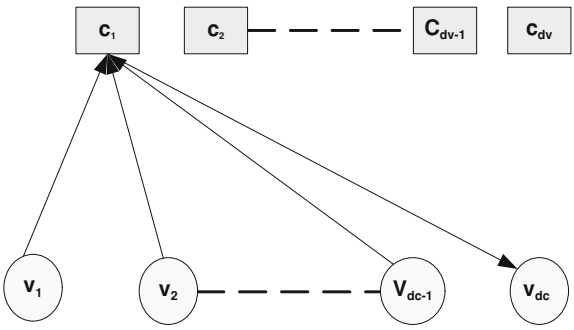

Fig. 101.2 Step 2. Check node update

Row processing: bit nodes to check nodes

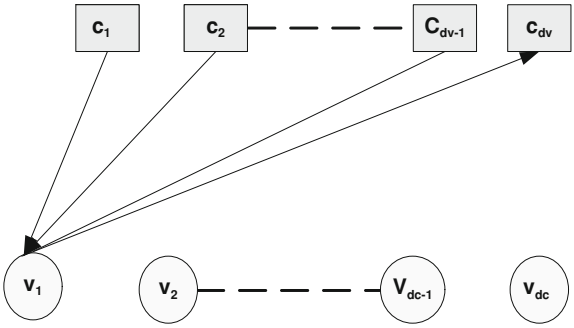

Fig. 101.3 Step 3. Variable node update

Column Processing: check nodes to bit nodes

101.3 Modified Min-Sum Decoding Algorithm

101.3.1 Variable Message Update Conditions

Min sum algorithm (MSA) greatly reduces the complexity of SPA at the cost of performance degradation and the bit error ratio is significantly higher than SPA. All efforts are made to make the MSA close to SPA in performance while keep it simple. The Offset and Normalized MSA alter the inaccurate magnitude for the check node update calculated in step 2. The offset and normalized min-sum algorithms increase the hardware complexity very less but improve the performance significantly for MSA and bring it close to SPA.

The modified min-sum algorithm (MMSA) [15] takes into account the following two conditions for two consecutive iterations:

(i) The sign of the present and previous variable messages are the same then the magnitude increase is comparatively small.
(ii) The sign of the present variable message and the previous variable messages are different. In this case, the magnitude increase is large and the variable message needs to be corrected to avoid overestimation.

Based on the these two fundamental facts, the newly proposed technique modifies the variable message at the check node processing with two different scaling factors either at a sign change or at no sign change. The choice of the scaling factors is obviously dependent on the magnitude increase and its hardware implementation complexity. The range for both the scaling factors s is such that; $0 < s < 1$. For the first condition, as the magnitude increase is less, the scaling factor is chosen in the range 0.5–0.9, and for the second, as the magnitude increase is large so it is modified by a factor in the range 0.1–0.5.

101.3.2 Method for Variable Message Correction

In the step 3, Eq. 101.6, the variable message is calculated at the ith iteration but before using for update, it is stored temporarily as $V_{mn}^{i,\text{tmp}}$.

$$V_{mn}^{i,\text{tmp}} = L_n^i - C_{mn}^i \tag{101.9}$$

Next the signs of the present message $V_{mn}^{i,\text{tmp}}$ and previous message V_{mn}^{i-1} are compared as; If

$$\text{sign}(V_{mn}^{i,\text{tmp}}) == \text{sign}(V_{mn}^{i-1}) \tag{101.10}$$

Then update the message as;

$$V_{mn}^{i} = sf_1(V_{mn}^{i,\text{tmp}}) \tag{101.11}$$

Else if

$$\text{sign}(V_{mn}^{i,\text{tmp}}) \neq \text{sign}(V_{mn}^{i-1}) \tag{101.12}$$

Then update the message as;

$$V_{mn}^{i} = sf_2(V_{mn}^{i,\text{tmp}}) \tag{101.13}$$

The scaling factors sf_1 and sf_2 are chosen in such a way that these could be conveniently implemented in hardware, and at the same time provide good approximation to the error performance. Now if the signs are different then the change in magnitude is large and is modified with small factor to reduce the overestimation effect. The scaling factors set for the simulation are $sf_1 = 0.5$ and $sf_2 = 0.25$. Now Eqs. 101.11 and 101.13 can be re-written as;

$$V_{mn}^{i} = 0.5(V_{mn}^{i,\text{tmp}}) \tag{101.11a}$$

$$V_{mn}^{i} = 0.25(V_{mn}^{i,\text{tmp}}) \tag{101.13a}$$

This brings further improvement to the MSA in both lower and upper region of SNR by using two scaling factors. The complexity is very less both in hardware and software but the performance achieved is far better.

101.3.3 Hardware Implementation and Complexity Analysis

We see that both scaling factors are easily implemented in hardware as shift registers. The scaling factors chosen are the division by 2 and 4 which are simply implemented in hardware as shift registers as data is shifted by one and two respectively. The hardware complexity is less than the min-sum algorithm in Haiyang et al. [15] as there is no adder needed. The sign comparator decides which input to select for assigning to the current message V_{mn}^{i} through multiplexer (Mux) unit. The shift register is fast and easy to implement. So the hardware complexity does not increase reasonably while the performance achievement is better. If we compare the hardware complexity with normalized min sum and offset min-sum decoding algorithms, then the complexity is increased just by a sign comparator which is comparatively very less but the BER advantage is significant

Also this is a reliable way for updating the variable message. Instead of uniform modification to all the variable messages, it gives the flexibility to update the messages in two ways which gives the advantage of better performance.

101.4 Simulation Results

Two types of codes are selected for the validation of the performance results for the proposed improved MSA through computer simulations. Regular medium and short length LDPC codes (1024, 512) and (648, 324) are chosen for all the decoding algorithms to evaluate the performance improvement. After encoding process and binary phase shift keying (BPSK) modulation, the desired code is passed through AWGN for a range of signal to noise ratio (SNR) values. The maximum allowable number of iteration is kept as 10. In the Fig. 101.4, we see clearly that the improved min-sum decoding algorithm outperform than MMSA, and even from SPA for the selected length of codes. The outperformance of improved min sum decoding algorithm than standard sum product algorithms is due to the fact that SPA depends on large sparse parity check matrix while the parity check matrix selected here is moderate length.

101.5 Comparison and Analysis

Simulation is performed for validating the performance of the proposed technique for medium and short length codes which best suits for most of the practical applications. The performance is tested at 10 maximum number of iteration for all types of the decoding algorithms. The results obtained are compared with MMSA and SPA. Also, simulations are carried out for two different code lengths. We see that for the (1024, 512) LPDC code at 4.5db SNR in Fig. 101.5, the BER for IMSA, MMSA and SPA are 0.000606, 0.001553 and 0.002029 respectively.

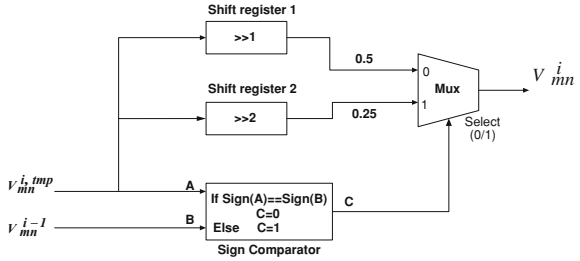

Fig. 101.4 Implementation for variable message correction

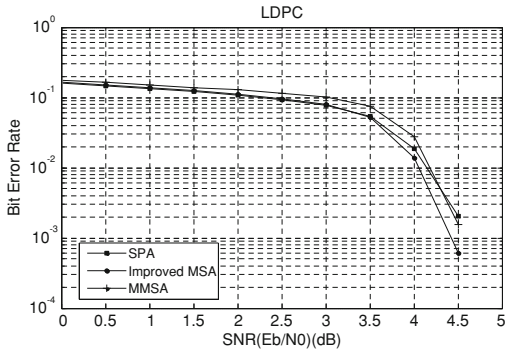

Fig. 101.5 BER performance comparison (1024, 512) LDPC code, maximum number of decoding iterations = 10

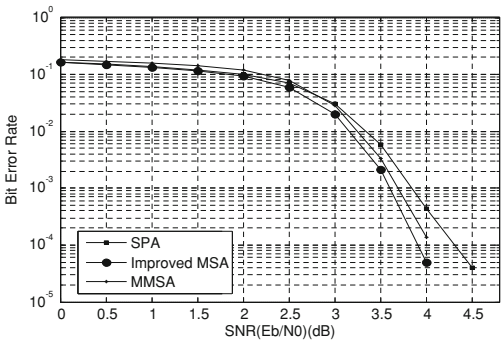

Fig. 101.6 BER performance comparison (648, 324) LDPC code, maximum number of decoding iteration = 10

Similarly for the (648, 324) LDPC code at SNR 4 in Fig. 101.6, the BER for IMSA, MMSA, SPA is 0.000050, 0.000137 and 0.000437, respectively. In both types of the codes, it has been observed that the proposed technique yields better results than other min-sum decoding algorithms.

101.6 Conclusion

In this chapter a totally new approach is used with some of the existing methods to improve the performance of the min-sum decoding algorithm for medium and short length codes which can be easily applied to practical systems. The proposed

method is an efficient way for modifying the variable message during the vertical process in all iterations and hence the overestimation is corrected optimally. Due to its two way normalization to correct the variable message magnitude, this algorithm has an inherit capability of improved performance. Furthermore, it is simple to implement, and the hardware cost is less.

Acknowledgments Supported by Science and Technology on Avionics Integration Laboratory and National Foundation of Aeronautical Science and Research under contract No. 20105552031.

References

1. Gallager RG (1963) Low-density parity-check code. MIT Press, Cambridge
2. Mackay D, Neal R (1996) Near Shannon limit performance of low density parity check codes. Electron Lett 32(18):1645–1646
3. Chung S, Forney GD, Richardson JJ, Urbanke R (2001) On the design of low-density parity-check codes within 0.0045 db of the Shannon limit. IEEE Commun Lett 5:58–60
4. Richardson T, Shokrollahi MA, Urbanke RL (2001) Design of capacity-approaching irregular low-density parity-check codes. IEEE Trans Inf Theory 47:619–637
5. Richardson T, Urbanke R (2000) The capacity of low-density parity check codes under message-passing decoding. IEEE Trans Inf Theory 47:599–618
6. Mackay D (1999) Good error correcting codes on very sparse matrices. IEEE Trans Inf Theory 45(2):399–431
7. Fossorier MPC, Mihaljevic M, Imai H (1999) Reduced complexity iterative decoding of low density parity check codes based on belief propagation. IEEE Trans Commun 47(5):673–680
8. Chen J, Dholakia A, Eleftheriou E et al (2005) Reduced-complexity decoding of LDPC codes. IEEE Trans Commun 53(8):1288–1299
9. Chen J, Fossorier MPC (2002) Density evolution for two improved BP-based decoding algorithms of LDPC codes. IEEE Commun Lett 6(5):208–210
10. Heo J (2003) Analysis of scaling soft information on low density parity check code. Electron Lett 55:219–221
11. Chen J, Fossorier MPC (2002) Near optimum universal belief propagation based decoding of low-density parity check codes. IEEE Trans Commun 50(3):406–414
12. Zhao J, Zarkeshvari F, Banihashemi AH (2005) On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes. IEEE Trans Commun 53:549
13. Chandrasetty VA, Aziz SM (2010) FPGA implementation of high performance LDPC decoder using modified 2-bit min-sum algorithm. Proceedings of the 2nd international conference on computer research and development, Kuala Lumpur, 7–10 May 2010, pp 881–885
14. Aziz SM, Pham MD (2010) Implementation of low density parity check decoders using a new high level design methodology. J Comput 5(1):0234–0237
15. Hai-yang L, Wen-ze QU, Bin L, Jiang-peng L, Shi-dong L, Jie C (2010) Novel modified min-sum decoding algorithm for low-density parity-check codes, www.sciencedirect.com. J China Univ Posts Telecommun 17:1–5