Journal of Southeast University (English Edition) Vol. 25, No. 4, pp. 423 – 426 Dec. 2009 ISSN 1003—7985

Improved parallel weighted bit-flipping algorithm

Liu Xiaojian Zhao Chunming Wu Xiaofu

(National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China)

Abstract: An improved parallel weighted bit-flipping (PWBF) algorithm is presented. To accelerate the information exchanges between check nodes and variable nodes, the bit-flipping step and the check node updating step of the original algorithm are parallelized. The simulation experiments demonstrate that the improved PWBF algorithm provides about 0.1 to 0.3 dB coding gain over the original PWBF algorithm. And the improved algorithm achieves a higher convergence rate. The choice of the threshold is also discussed, which is used to determine whether a bit should be flipped during each iteration. The appropriate threshold can ensure that most error bits be flipped, and keep the right ones untouched at the same time. The improvement is particularly effective for decoding quasi-cyclic low-density parity-check (QC-LDPC) codes.

Key words: low-density parity-check (LDPC); parallel weighted bit-flipping (PWBF); improved modified weighted bit-flipping (IMWBF) algorithm; weighted-sum weighted bit-flipping (WSWBF) algorithm

Low-density parity-check (LDPC) codes, discovered by Gallager in 1962, have attracted much research interest recently. The belief propagation (BP) and its suboptimal approximation, the min-sum algorithm[1-4], can achieve excellent performance. However, they are usually too complex for implementation. The bit-flipping (BF) decoding algorithm originally devised by Gallager is very simple, but often results in serious performance loss. The weighted BF (WBF) algorithm provides an effective trade-off between error performance and decoding complexity[5]. The main shortcoming of this algorithm is its slow convergence rate, since it flips only one bit after each iteration.

The parallel weighted bit-flipping (PWBF) decoding algorithm[6-7] provides a parallel implementation framework for various WBF algorithms[8-9]. Compared with the serial form of the WBF algorithm, the PWBF decoding algorithm flips multiple bits after each iteration. Simulations prove that it only needs five iterations to approach the performance of various improved WBF decoding algorithms with 50 or more iterations. However, a performance gap still exists between the PWBF decoding algorithm and the min-sum algorithm, with a similar decoding rate.

In this paper, we discuss the improvement of the PWBF decoding algorithm. To accelerate the information exchanges between check nodes and variable nodes, we propose a scheme, which carries out several steps of neighboring iterations in parallel. This improvement, while requiring very low additional computational complexity, results in a performance gain over the PWBF algorithm and a higher decoding rate at the same time. The proposed scheme is effective, especially when applied to QC-LDPC codes[10-15]. Simulations prove that the improved PWBF algorithm achieves better performance and converges faster.

Received 2009-04-22.

Biographies: Liu Xiaojian (1982—), female, graduate; Zhao Chunming (corresponding author), male, doctor, professor, cmzhao2@seu.edu.cn.

Foundation items: The National High Technology Research and Development Program of China (863 Program) (No. 2009AA01Z235, 2006AA01Z263), the Research Fund of the National Mobile Communications Research Laboratory of Southeast University (No. 2008A10).

Citation: Liu Xiaojian, Zhao Chunming, Wu Xiaofu. Improved parallel weighted bit-flipping algorithm[J]. Journal of Southeast University (English Edition), 2009, 25(4): 423 – 426.

1 PWBF Algorithm

Let $\boldsymbol{C}$ be a regular LDPC code with the block length N and the dimension K, which can be represented by a parity-check matrix $\boldsymbol{H} = (h_{m,n})$ with M rows and N columns. Let ρ and γ denote the column-weight and the row-weight of the matrix $\boldsymbol{H}$, respectively.

We assume information transmission using BPSK signaling over an AWGN channel. Let $\boldsymbol{c} = \{c_1, c_2, \ldots, c_N\}$ denote a codeword, which is mapped to $\boldsymbol{x} = \{x_1, x_2, \ldots, x_N\}$ by $x_n = 2c_n - 1$ before transmission. At the receiver, we obtain the received vector $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$, where $y_n = x_n + v_n$, $n = 1, 2, \ldots, N$, and v_n is the zero-mean additive Gaussian noise with a variance of $\sigma^2 = N_0/2$.

Let $\boldsymbol{z} = \{z_1, z_2, \ldots, z_N\}$ be the hard-decision sequence obtained from $\boldsymbol{y}$ with $z_n = \mathrm{sgn}(y_n)$, where $\mathrm{sgn}(y_n) = 1$ if $y_n \geqslant 0$, and $\mathrm{sgn}(y_n) = 0$ if $y_n < 0$. We denote $\boldsymbol{N}(m) = \{n: h_{m,n} = 1, 1 \leqslant n \leqslant N\}$ and $\boldsymbol{M}(n) = \{m: h_{m,n} = 1, 1 \leqslant m \leqslant M\}$.

The PWBF algorithm is proposed in Ref. [5]. Some parameters are defined as follows: k is the iteration counter; ε_M is the set of unsatisfied check nodes; $F_{\mathrm{th}}(k)$ is the threshold to be optimized; F_k is the set of bits to be flipped at the k-th iteration.

When the PWBF algorithm uses the functions of the IMWBF algorithm, it is carried out as follows.

Algorithm 1 PWBF algorithm

1) Initialization

Set $k = 1$; $z = \mathrm{sgn}(y)$.

2) Iteration

① Compute the syndrome component $s_m = \sum_{n \in N(m)} z_n$ for each $m \in [1, M]$, and collect all unsatisfied check nodes to form $\varepsilon_M = \{m \mid s_m = 1, m \in [1, M]\}$.

② Compute flipping function E_n for each $n \in [1, N]$,

$$E_n = \sum_{m \in M(n)} (2s_m - 1) \mid \omega_{n,m} \mid - \alpha \mid y_n \mid \qquad (1)$$

where $\omega_{n,m} = \min_{i \in N(m) \backslash n} \mid y_i \mid$, and α is a positive constant less than 1.

③ Locate $n^* = \arg\max_{n \in N(m)} \{E_n\}$ and send a "flip" signal to variable node n^* for each $m \in \varepsilon_M$.

④ Accumulate the "flip" signal to $F_k(n)$ for each $n \in [1, N]$.

⑤ Identify the bits whose $F_k(n)$ are larger than $F_{\text{th}}(k)$, and flip them in parallel.

⑥ Increase k by 1, and repeat steps ① to ⑤ until all the parity equations are satisfied or the maximum number of iterations is reached.

The PWBF algorithm can be adopted to various WBF algorithms with different flipping functions E_n. In this study, we choose the parallel IMWBF (PIMWBF) algorithm and the parallel WS-BF (PWSWBF) algorithm to illustrate the effect of improvement given in the next section. The flipping function of the PIMWBF algorithm is shown in Eq. (1). For the PWSWBF algorithm, the flipping function is computed as follows.

First, the reliability of check nodes is defined as

$$w_m = K - \underset{i \in R(m) \setminus n}{\text{num}} \{ |y_n| \leqslant T \} \tag{2}$$

where K is a positive integer no greater than the row weight of the matrix $\boldsymbol{H}$; $R(m)$ is the set of reliable symbols that participate in the m-th check equation; T is the threshold. A symbol's reliability is judged by its magnitude $|y_n|$. If $|y_n| \geqslant T$, we call it "reliable", and "unreliable" otherwise. If $w_m < 0$, we set it to 0. The newly defined E_n is as follows:

$$E_n = \sum_{m \in M(n)} w_m E_n^* \tag{3}$$

where

$$E_n^* = \begin{cases} |y_n| - \min\limits_{n \in N(m)} |y_n|/2 & s_m = 0 \\ |y_n| - \min\limits_{n \in N(m)} |y_n|/2 - \max\limits_{n \in N(m)} |y_n| & s_m \neq 0 \end{cases} \tag{4}$$

Besides, we set $n^* = \arg \min\limits_{n \in N(m)} \{E_n(\alpha)\}$ in step ③.

2 Improved PWBF algorithm

In the PWBF algorithm, when a bit is flipped in step ⑤, its information will not be used to update the syndrome components or the flipping functions until the current iteration is finished. If most flipped bits are contaminated by noise, completing these updates immediately can help the algorithm to obtain a correct codeword. To achieve this goal, we propose an improved PWBF algorithm, which is carried out as follows.

Algorithm 2 Improved PWBF algorithm

1) Initialization

Set $z = \text{sgn}(y)$.

For $m = 1, 2, \ldots, M$,

① Compute the syndrome component $s_m = \sum\limits_{n \in N(m)} z_n$. Collect the unsatisfied check node to form $\varepsilon_M = \{m \mid s_m = 1, m \in [1, M]\}$.

② Compute flipping function E_n for each $n \in [1, N]$.

2) Iteration

For $m = 1, 2, \ldots, M$ and $i \in \varepsilon_M$,

①Locate a variable node n^*, and send it a "flip" signal.

②Accumulate the "flip" signal to $F_k(n^*)$, if $F_k(n^*) \geqslant F_{\text{th}}(k)$.

③ Flip the bit n^*.

④ Recompute the syndrome component s_m for each $m \in M(n^*)$.

⑤ Modify the flipping functions of variable nodes that participate in the checks, whose indices are contained in $M(n^*)$.

During the process of steps ③-⑤, steps ①-② should be suspended to make sure that they can use the newly updated information. Such a stop-and-wait scheme is easy to be applied to various LDPC codes; however, it may result in a slow convergence rate. We have two methods to solve the problem. The first one is parallelism. Only one clock cycle is needed for step ③ and step ④, respectively, when a fully parallel scheme is applied. For a regular LDPC code, every variable node connects to ρ check nodes. With $\rho \ll M$, the increase in processing elements is small. The other way is to use the LDPC codes whose matrices $\boldsymbol{H}$ can avoid the case that several successive check nodes contain common components. When steps ①-② and steps ③-⑤ are carried out as in Fig. 1(b), once a bit n^* is flipped in step ③, the syndrome components s_m, whose indices are contained in $M(n^*)$, can be recomputed. If the check node, which step ① processes in the next phase, is not contained in $M(n^*)$, the information updated in successive steps ④-⑤ can be fully passed to the remaining steps ①-②.

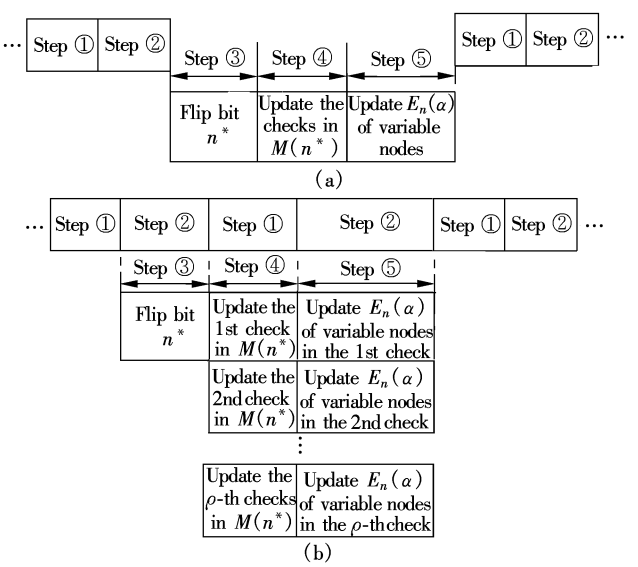

Fig. 1 The arrangement of the improved PWBF algorithm. (a) Stop-and-wait scheme with parallelism; (b) Fully parallel scheme with QC-LDPC codes

The matrix $\boldsymbol{H}$ of a QC-LDPC code is given as an array of sparse circulants with the same size[15]. Consider the array of $b \times b$ circulants over GF(2), which has the following structural properties: 1) The weight of each circulant is 1; 2) No two rows have more than one 1-component in common. If we split the matrix $\boldsymbol{H}$ row-wise into several $b \times N$ row-blocks as in Fig. 2, the two properties ensure that within each row-block the bits joining in the same check do not participate in other checks. So, within each row-block, we can process steps ①-② and steps ③-⑤ in parallel. The comparison of the stop-and-wait scheme and the fully parallel scheme is shown in Fig. 1. Between each row-block, however, steps ①-② should be suspended for three phases, since different row-blocks may have several common 1-components. In the improved PWBF algorithm, steps ④-⑤

correspond to steps ①-② of the PWBF algorithm. From the view of the PWBF algorithm, the neighboring iterations are partly performed in parallel, as shown in Fig. 3. With $\rho \ll b$, the time to finish an iteration is approximately half that of the original PWBF algorithm. So the improved PWBF algorithm can achieve a higher decoding rate.

Let ω_i, $i = 1, 2, \ldots, K$, where K is the upper-bound of iteration numbers, be the size of s_m at the beginning of each iteration. Supposing that each step of the improved PWBF algorithm can be finished in a clock cycle, the total cycles that are needed to decode a frame are about $M + 2\sum_{i=0}^{K} \omega_i +$

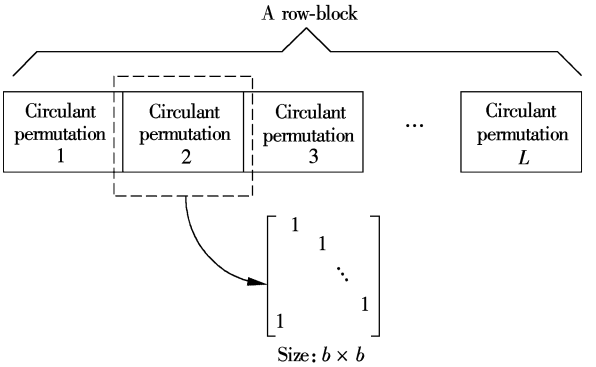

Fig. 2 An example of a row-block ($L = N/b$)

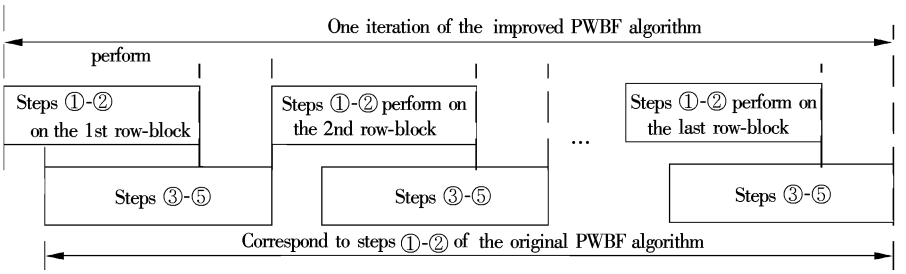

Fig. 3 One iteration of the improved PWBF algorithm with QC-LDPC codes

$o(M)$, where M denotes the time spent on the initialization with the pipelined scheme. As for the PWBF algorithm, it consumes about $M + 2\sum_{i=0}^{5} \omega_i + o(M)$ cycles for each frame. At the BER of 10^{-5}, $\sum_{i=0}^{5} \omega_i$ is close to $2M$, which is proved by simulations. Thus, the throughput gain through the proposed decoding compared with the original decoding is $3M/(1.8M) \approx 1.7$.

The threshold $F_{\text{th}}(k)$ ($k = 1, 2, \ldots, I_{\max}$) has a great impact on the performance of the PWBF algorithm. A high threshold can help correct symbols avoid being flipped; however, sometimes it also causes error symbols to be ignored. The effect of a low threshold is the opposite. For the turbo time-scheduling, we usually choose small $F_{\text{th}}(k)$ as the threshold. The low thresholds accelerate the exchange of information between check nodes and variable nodes. Although some correct symbols may be flipped due to a low threshold, the influence on flipping functions of variable nodes is slight, because each time only up to ρ syndrome components are refreshed, where ρ is quite small compared with M. On the other hand, they are easy to correct in the posterior iterations, since they usually have relatively high reliability. Besides, as most error symbols are easy to locate, the correct information can be utilized as quickly as possible.

3 Simulation Results

In this section, we consider two QC-LDPC codes based on finite fields[13] in our simulation: one is the (961, 721) QC-LDPC with the block length $N = 961$ and a rate of approximately 0.75; the other one is the (8 148, 7 571) QC-LDPC with $N = 8\,148$ and a rate of approximately 0.93.

We evaluate the error performances of these codes decoded with the following algorithms: 1) The belief propagation algorithm; 2) The offset min-sum algorithm; 3) The PWSWBF algorithm; 4) The improved PWSWBF algorithm; 5) The PIMWBF algorithm; 6) The improved PIMWBF algorithm; 7) The WBF algorithm; 8) The BF algorithm. The improved PIMWBF algorithm is the PIMWBF algorithm with the turbo scheme as shown in Fig. 1(b). For algorithms 1) and 6), the maximum number of iterations is set to 5, and for the remaining it is set to 100.

Fig. 4 shows the performances of the (961, 721) QC-LDPC code with various decoding algorithms. For the offset min-sum algorithm, the normalized factor is set to 0.3. $F_{\text{th}}(k)$ for the PWBF and the PWSWBF algorithms are set to [10, 7, 7, 6, 5], and α is set to 1.8. For the improved PIMWBF algorithm and the improved PWSWBF algorithm, $F_{\text{th}}(k)$ are set to [5, 2, 2, 2, 2], and α is set to 1.8. For the improved PWSWBF algorithm and the PWSWBF algorithm, K, T are set to 0.3 and 5, respectively.

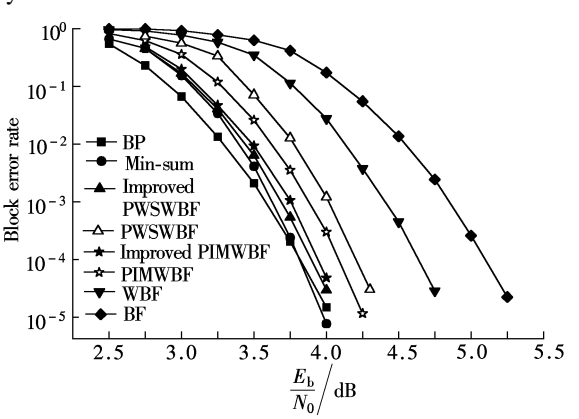

Fig. 4 Block-error performances of (961, 721) QC-LDPC code with various decoding algorithms

According to Fig. 4, with the turbo scheme, both the PWSWBF algorithm and the PIMWBF algorithm obtain better error performance compared with other algorithms. At the block error rate of 10^{-5}, their block error rate(BER) performances are approximate to the min-sum algorithm.

Fig. 5 shows the performances of the (8 148, 7 571) QC-LDPC code with various decoding algorithms. For the offset min-sum algorithm, the normalized parameter is set to 0.5. $F_{th}(k)$ for the PIMWBF algorithm and the PWSWBF algorithm are set to [4, 3, 3, 2, 2], and α is set to 0.6. For the improved PIMWBF algorithm and the improved PWSWBF algorithm, $F_{th}(k)$ are set to [5, 1, 1, 1, 1], and α is set to 0.6. K, T are set to 0.5 and 18 for the improved PIMWBF algorithm, and to 0.3, 9 for the PWSWBF algorithm.

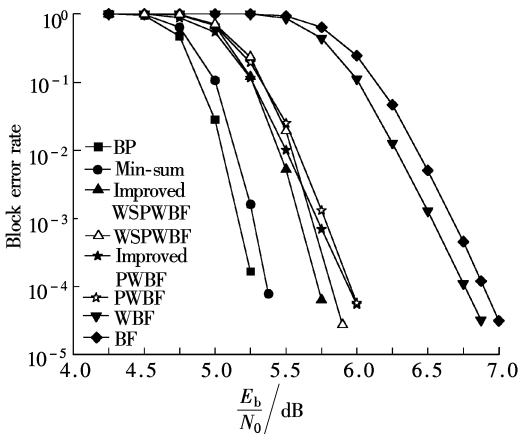

Fig. 5 Block-error performances of (8 148, 7 571) QC-LDPC code with various decoding algorithms

4 Conclusion

In this paper, the improvement of the PWBF algorithm is presented. Simulation results show that the improved PWBF algorithm performs better than the original PWBF algorithm. Furthermore, a higher convergence rate can be achieved with the QC-LDPC codes and the parallel scheme in the updating process. The cost is an increase in the processing units. It offers an effective trade-off between the performance of the min-sum algorithm and the complexity of the WBF algorithm.

References

[1] Richardson T, Urbanke R. The capacity of low-density parity-check nodes under message-passing decoding [J]. *IRE Trans Inform Theory*, 2001, **47**(2): 599 - 618.

[2] Wiberg N. Codes and decoding on general graphs [D]. Linköping, Sweden: Department of Electrical Engineering of Linköping University, 1996.

[3] Kschischang F R, Frey B J, Loeliger H A. Factor graphs and the sum-product algorithm [J]. *IRE Trans Inform Theory*, 2001, **47**(2): 498 - 519.

[4] Shan M, Zhao C M, Jiang M. Improved weighted bit-flipping algorithm for decoding LDPC codes [J]. *IEE Proc Commun*, 2005, **152**(6): 919 - 922.

[5] Wu Xiaofu, Zhao Chunming, You Xiaohu. Parallel weighted bit-flipping decoding [J]. *IEEE Commun Lett*, 2007, **11**(8): 671 - 673.

[6] Wu Xiaofu, Ling Cong, Jiang Ming, et al. Towards understanding weighted bit-flipping decoding [C]//*IEEE Int Symp Inform Theory* 07. Nice, France, 2007: 1666 - 1670.

[7] Zhang Juntan, Fossorier Marc P C. A modified weighted bit-flipping decoding of low-density parity-check codes [J]. *IEEE Commun Lett*, 2004, **8**(3): 165 - 167.

[8] Liu Zhenyu, Pados Dimitris A. A decoding algorithm for finite-geometry LDPC codes [J]. *IEEE Trans Commun*, 2005, **53**(3): 415 - 421.

[9] Jiang Ming, Zhao Chunming, Shi Zhihua, et al. An improvement on the modified weighted bit flipping decoding algorithm for LDPC codes [J]. *IEEE Commun Lett*, 2005, **9**(9): 814 - 816.

[10] Kou Y, Lin S, Fossorier M P C. Low-density parity-check codes based on finite geometries: a rediscovery and new results [J]. *IEEE Trans Info Theory*, 2001, **47**(7): 2711 - 2736.

[11] Song S, Lan L, Lin S, et al. Construction of quasi-cyclic LDPC codes based on the primitive elements of finite fields [C]//*IEEE Information Sciences and Systems*. Princeton, NJ, USA, 2006: 22 - 24.

[12] Kim Sunghwan, No Jong-Seon, Chung Habong, et al. Quasi-cyclic low-density parity-check codes with girth larger than 12 [J]. *IEEE Trans Inform Theory*, 2007, **53**(8): 2885 - 2891.

[13] Li Zongwang, Chen Lei, Zeng Lingqi, et al. Efficient encoding of quasi-cyclic low-density parity-check codes [C]// *IEEE Global Telecommunications Conference*. St Louis, MO, USA, 2005: 71 - 81.

[14] Liva G, Ryan W E, Chiani M. Quasi-cyclic generalized LDPC codes with low error floors [J]. *IEEE Trans Commun*, 2008, **56**(1): 49 - 57.

[15] Sha Jin, Gao Minglun, Zhang Zhongjin, et al. Efficient decoder implementation for QC-LDPC codes [C]//*Proceedings of International Conference on Communications, Circuits and Systems*. Guangzhou, China, 2006: 2498 - 2502.

改进型并行比特翻转算法

刘晓健　赵春明　吴晓富

(东南大学移动通信国家重点实验室, 南京 210096)

摘要: 提出一种改进的并行比特翻转算法. 为了加快校验节点和变量节点之间的信息传递速率, 算法中的比特翻转及校验和更新2个步骤采用并行化处理. 仿真结果表明, 改进后的算法相对于原有的并行比特翻转算法在误帧率性能上能够取得0.1 ~0.3 dB 的增益. 同时, 改进算法在译码吞吐率的性能上也有所改善. 此外, 还讨论了翻转门限的选择方法, 这些门限决定了每次迭代中哪些比特需要被翻转. 通过选择合适的翻转门限, 可使错误的比特尽量多地被翻转, 并避免翻转正确的比特. 该改进算法比较适用于对具有准循环结构的 LDPC 码进行译码.

关键词: 低密度奇偶校验(LDPC); 并行比特翻转; 改进的权重型比特翻转算法; 校验和加权的权重型比特翻转算法

中图分类号: TN911.22