Research Article
Weighted-Bit-Flipping-Based Sequential Scheduling Decoding Algorithms for LDPC Codes

Qing Zhu and Le-nan Wu

School of Information Science and Engineering, Southeast University, Nanjing 210096, China

Correspondence should be addressed to Qing Zhu; dreamathstat@163.com

Received 24 April 2013; Accepted 22 June 2013

1. Introduction

Low-density parity-check (LDPC) codes were first invented by Gallager [1] but had been neglected for decades until Mackay brought them back to light in 1996 [2]. Since then, much attention had been attracted for their excellent Shannon limit approaching error-correcting performance through belief propagation (BP) [3] decoding algorithm. This iterative decoding algorithm, sometimes also called sum-product algorithm (SPA) [4], is a powerful algorithm to approximately solve many NP hard problems such as statistical inference in physics [5–7], hypothesis testing, cooperative localization, and channel coding.

There are lots of researches with various decoding algorithms. Among existing LDPC decoding algorithms, bit-flipping (BF) algorithms are the simplest. The operations of check-nodes in BF are modulo-two additions while the variable-nodes only need simple comparison operations. BF decoding algorithms are easy to implement, but they usually perform not so well when compared to BP decoding algorithms, so various weighted BF (WBF) [8–12] decoding algorithms were proposed.

To improve the standard BP decoding performance, several sequential scheduling strategies in BP have been invented. In sequential scheduling strategies, the messages are computed in a serial manner using the newest updated information. Sequential strategies were introduced as a sequence updates based on check-node (CSBP) [13–15] or variable-node (VSBP) [16, 17]. Simulations demonstrate that sequential strategies converge about twice as fast as the standard parallel BP decoding algorithms (Flood) without any extra computing burden. The kernel steps of sequential updating algorithms focus on finding the order of message updating which converges fastest. To our knowledge, the best decoding algorithms in the sense of performance is informed dynamic scheduling (IDS) [18–24] algorithms, which update messages dynamically. A metric called residual [18] is used in IDS which decides the updating order of propagated messages. Metric computing and selecting operations can cause significant increase in computational complexity in IDS. In order to achieve the tradeoff between decoding performance and computational complexity, a low-complexity sequential WBF-based scheduling algorithm is proposed, in which the priority used in WBF determines the order of scheduling. Simulation results show that it can provide a good tradeoff.
between block errors performance and complexity for short-length LDPC codes.

This paper is organized as follows. Section 2 reviews BP (Flood and CSBP) and WBF. Section 3 introduces our WBF-based serial BP (WBFSBP) strategies. Section 4 analyzes the computing complexity of WBFSBP. Section 5 reveals the simulation results. Section 6 draws the conclusions.

2. WBF and BP

Let $N$ and $K$ be the block length and the information length of a binary LDPC code; thus the rate of the code is $r = K/N$. Let $M = N - K$; an LDPC code is described by an $M \times N$ parity-check matrix $H$. $H_{mn}$ presents the entry of row $m$ and column $n$ in $H$. The set $\mathcal{N}(m)$ denotes the nodes that adjoin $m$ by $\mathcal{N}(m) = \{ n : H_{mn} = 1 \}$, similarly for the set $\mathcal{N}(n) = \{ m : H_{mn} = 1 \}$. $\mathcal{N}(m)/n$ denotes the set $\mathcal{N}(m)$ excluding $n$. Binary phase shift keying (BPSK) modulation over an additive white Gauss noise (AWGN) channel is assumed for information transmission, which maps a codeword vector $c = (c_1, c_2, \ldots, c_n)$ into a vector $x = (x_1, x_2, \ldots, x_n)$, where $c_i = 2x_i - 1$, $i \in [1, N]$. The received vector is $y = (y_1, y_2, \ldots, y_n)$, where $y_i = x_i + n_i$, $i \in [1, N]$, and $n_i$ is the AWGN. $z = (z_1, z_2, \ldots, z_n)$ denotes the hard-decision vector acquired from $y : z_i = 1$ if $y_i > 0$ and $z_i = 0$ if $y_i \leq 0$. 

### Table 1: Computation complexity from variable-nodes to check-nodes.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Updating</th>
<th>Reliability computation</th>
<th>Sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flood</td>
<td>$e$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>CSBP</td>
<td>$e$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>IDS</td>
<td>$Md(d_i - 1)$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>WBFSBP</td>
<td>$e$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

### Table 2: Computation complexity from check-nodes to variable-nodes.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Updating</th>
<th>Reliability computation</th>
<th>Sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flood</td>
<td>$e$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>CSBP</td>
<td>$e$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>IDS</td>
<td>$Md(d_i - 1)$</td>
<td>$O(N^2)$</td>
<td></td>
</tr>
<tr>
<td>WBFSBP</td>
<td>$e$</td>
<td>$O(Md_d_c)$</td>
<td>$O(N \log N)$</td>
</tr>
</tbody>
</table>

Figure 3: FER versus $E_b/N_0$ performance of BP (Flood), CSBP, and WBFSBP decoding algorithms with maximal number of iterations of 20 with LDPC code of blocklength of 256 and rate of 0.5. BP algorithm with maximal number of iterations of 20.
2.2. Standard BP (Flood) Decoding Algorithm. Commonly, the communication of the nodes in the corresponding Tanner graph comprises the Flood algorithm. For every variable-node \( v_j \) and check-node \( c_i \), the corresponding message generation functions \( m_{v_j \rightarrow c_i} = f_{c_i \rightarrow v_j}(m) \) and \( m_{c_i \rightarrow v_j} = f_{v_j \rightarrow c_i}(m) \) are defined as follows [2]:

\[
m_{v_j \rightarrow c_i} = \sum_{c_i \in \mathcal{X}(v_j) \setminus c_i} m_{v_j \rightarrow c_i} + C_{v_j},
\]

\[
m_{c_i \rightarrow v_j} = 2 \arctanh \left( \prod_{v_j \in \mathcal{X}(c_i) \setminus v_j} \tanh \left( \frac{m_{v_j \rightarrow c_i}}{2} \right) \right),
\]

where \( C_{v_j} = \log (p(y_j | v_j = 0) / p(y_j | v_j = 1)) \) is the channel information of \( v_j \). \( L(v_j) \) denotes the LLR of \( v_j \).

As shown in Figure 1, in time slot 1, messages are passed to all of these check-nodes by all variable-nodes that connect to them, and in time slot 2, all of the processed messages are sent back to all variable-nodes. The Flood decoding algorithm is formally described in Pseudocode 1.

2.3. CSBP. The messages calculated in CSBP [15] are the same as those in the Flood decoding algorithms. The only difference between them is the way of messages updating. In every iteration of CSBP, the messages are updated in a sequential way: update every check-node one by one in an ascending order (or descending order) arranged in the corresponding Tanner graph. The passing process of messages in CSBP is shown in Figure 2. Check-nodes are represented by \( (c_1, c_2, c_3) \), and \( (b_1, b_2, \ldots, b_6) \) for variable-nodes. In time slot 1, messages are passed to \( c_1 \) by all variable-nodes that connect to \( c_1 \). In time slot 2, the processed messages are sent back to those variable-nodes. The updating rules of \( c_2 \) and \( c_3 \) are identical with those of \( c_1 \). The realization of CSBP is described in Pseudocode 2.

3. WBF-Based Serial BP

We will introduce our WBFSBP algorithm using a new message updating schedule. In general, sequential decoding algorithms are composed of two major steps.

Step 1. Determine the order of nodes to be updated.

Step 2. Compute the messages.

In Flood and CSBP, Step 1 is skipped. In IDS, these two steps are operated alternately for every node. It means that after updating each single node, “residual” computing and sorting operation is needed, which causes much extra computing complexity. For the sake of getting the tradeoff between performance and computing complexity, a simple sequential decoding algorithm called WBF-based serial BP is proposed. First, let us review the flipping function \( e_{\text{WBF,n}}^{(k)} \)
1: Set every $m_{c \rightarrow v} = 0$
2: Set every $m_{v_j \rightarrow c_i} = C_{V_j}$
3: while stop rules unsatisfied do
4: for all $c_j (a = 1, 2, \ldots, M)$
5: for each $v_k \in N(c_j)$, compute and generate $m_{v_k \rightarrow c_j} (a = 1, 2, \ldots, M)$ simultaneously
6: end for
7: end while

<table>
<thead>
<tr>
<th>Pseudocode 1: Pseudocodes of Flood.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Set every $m_{c \rightarrow v} = 0$</td>
</tr>
<tr>
<td>2: Set every $m_{v_j \rightarrow c_i} = C_{V_j}$</td>
</tr>
<tr>
<td>3: while stop rules unsatisfied do</td>
</tr>
<tr>
<td>4: for all $c_j (a = 1, 2, \ldots, M)$</td>
</tr>
<tr>
<td>5: for each $v_k \in N(c_j)$ do</td>
</tr>
<tr>
<td>6: compute and generate $m_{v_k \rightarrow c_j} (a = 1, 2, \ldots, M)$</td>
</tr>
<tr>
<td>7: end for</td>
</tr>
<tr>
<td>8: end for</td>
</tr>
<tr>
<td>9: end while</td>
</tr>
</tbody>
</table>

The decoding performance of the Flood, CSBP, and WBFSBP over AWGN channels is presented in this part. The LDPC codes are constructed based on Gallager’s random method without any 4-cycle. Figures 3 and 4 present the FER performance of the (128, 256) Gallager LDPC code with BP (Flood), CSBP, and WBFSBP decoding algorithms. The $k_{max}$ is set to 20 and 50, respectively, and $\alpha$ is set to 1.3 in the computation of $E_{WBF,m}^k$. Figure 3 shows that, at the FER of $1e - 4$, WBFSBP acquires about 0.2 dB and 0.35 dB promotions over the CSBP and BP in the case of $k_{max} = 20$, while in Figure 4, the coding gains are 0.25 dB and 0.45 dB in the case of $k_{max} = 50$. Figure 4 also

of a variable-node in WBF decoding algorithm described in Section 2. The larger the $E_{WBF,m}^k$ is, the more unreliable the corresponding variable-node is. For each check-node, a new metric is defined as follows:

$$E_{WBF,m}^k = \max_{n \in N(m)} \sum \max_{e \in E(n)} E_{WBF,n}^k$$  \hspace{1cm} (5)$$

where submax ($\cdot$) denotes the second maximal value of ($\cdot$). If there is more than one unreliable variable-node connected to a check-node, the message passing within this check-node should be very low. We concentrate on the check-node which has only one unreliable neighbor variable-node. The check-nodes with larger value of $E_{WBF,m}^k$ are more likely containing just one unreliable neighbor variable-node. So updating these check-nodes firstly can correct errors in time and speed up convergence, thus reducing the iterations. In contrast to IDS, the sorting operation is not dynamically decided by node but is simply obtained from sorting the set of $E_{WBF,m}^k$ in each iteration. When the sorting is finished, the rest of updating computation is the same as that of CSBP. The detailed steps of WBFSBP are shown in Pseudocode 3.

## 4. Complexity

Let $d_v$ and $d_c$ represent the degree of any variable-node and check-node, respectively; $e$ is the total number of all edges in the corresponding Tanner graph, so $e = d_v \cdot N = d_c \cdot M$, where $N$ is the number of variable-nodes and $M$ is the number of check-nodes. In Tables 1 and 2, we have listed the computation complexity of check-to-variable phase and variable-to-check phase, respectively. In an iteration of Flood, every edge of the Tanner graph should be updated once for each direction, so the number of updated messages via check-to-variable and variable-to-check is $e$, respectively. Sequential updating strategies surpass the Flood without any extra cost. We note that the check-to-variable message updating computations in one IDS iteration is the same as reverse message updating computations in one WBFSBP iteration, both equal $e$. In each iteration, sorting of $\{E_{WBF,m}^k\}$ requires $\mathcal{O}(N \log N)$ operations while dynamic scheduling strategies in IDS need $\mathcal{O}(N^2)$.

## 5. Simulation Results

The decoding performance of the Flood, CSBP, and WBFSBP over AWGN channels is presented in this part. The LDPC codes are constructed based on Gallager’s random method without any 4-cycle. Figures 3 and 4 present the FER performance of the (128, 256) Gallager LDPC code with BP (Flood), CSBP, and WBFSBP decoding algorithms. The $k_{max}$ is set to 20 and 50, respectively, and $\alpha$ is set to 1.3 in the computation of $E_{WBF,m}^k$. Figure 3 shows that, at the FER of $1e - 4$, WBFSBP acquires about 0.2 dB and 0.35 dB promotions over the CSBP and BP in the case of $k_{max} = 20$, while in Figure 4, the coding gains are 0.25 dB and 0.45 dB in the case of $k_{max} = 50$. Figure 4 also
0: Set iteration number $k = 0$, take $k_{\text{max}}$ as the maximum number of iteration, pick up $w_{nm}$.
Set every $m_{i\rightarrow j} = 0$, every $m_{ij\rightarrow i} = C_{ij}$.
1: Compute $s^k = z^k H^T$, if $s^k = 0$, stop decoding and output $z^k$.
2: For $n = 1, 2, \ldots, N$, compute
$$E^k_W = \sum_{m \in \mathcal{J}(n)} (2s^k_m - 1)w_{nm} - \alpha |y_n|$$
3: sort $E^k_{WUBF}$ in descending order and put the corresponding check-nodes to $U$.
4: while stop rules unsatisfied do
5: for $i$ do
6: for $i$th check-node $c_{ij}$ in $U$
7: for every $v_b \in \mathcal{N}(c_{ij})$ do
8: compute and generate $m_{v_b \rightarrow a_i}$
9: end for
10: for every $v_b \in \mathcal{N}(c_{ij})$ do
11: compute and generate $m_{a_i \rightarrow v_b}$
12: end for
13: end for
14: end while

Pseudocode 3: Pseudocodes of WBFSBP.

shows that at $E_b/N_0 = 4$, WBFSBP-20 can achieve the same FER performance as what BP-50 and CSBP-50 do. In other words, WBFSBP algorithm can reduce more than half of the iterations of that BP algorithm needed at some $E_b/N_0$.

Figure 5 presents the FER performance of the $(102, 204)$ Gallager LDPC code with above-mentioned decoding algorithms. The $k_{\text{max}}$ is set to 20 and $\alpha$ is set to 1.3. We see that WBFSBP-20 can beat BP-50 for all $E_b/N_0$. Compared to CSBP-20, WBFSBP-20 can acquire about 0.15 dB coding gain.

6. Conclusion

In this paper, a new sequential scheduling decoding algorithm is proposed, in which the order of message passing is based on the weight factor computed in WBF algorithm. We focus on the check-nodes which have just one unreliable neighbor variable-node. Updating these check-nodes preferentially can avoid errors of propagation, thus speeding up the convergence of decoding algorithm. Compared to IDS, our decoding algorithm gives a good tradeoff between error performance and decoding complexity. As a future work, the scheduling algorithm can be generalized to LDPC codes over GF($q$).

Acknowledgments

The authors would like to thank all anonymous reviewers whose constructive suggestions were very helpful. The work of this paper is sponsored by the State 863 Project (2008AA01Z227), the National Natural Science Foundation of China (NSFC), under Grant 61271204.

References


