Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 371206, 6 pages
http://dx.doi.org/10.1155/2013/371206

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

Academic Editor: Yudong Zhang

Low-density parity-check (LDPC) codes can be applied in a lot of different scenarios such as video broadcasting and satellite communications. LDPC codes are commonly decoded by an iterative algorithm called belief propagation (BP) over the corresponding Tanner graph. The original BP updates all the variable-nodes simultaneously, followed by all the check-nodes simultaneously as well. We propose a sequential scheduling algorithm based on weighted bit-flipping (WBF) algorithm for the sake of improving the convergence speed. Notoriously, WBF is a low-complexity and simple algorithm. We combine it with BP to obtain advantages of these two algorithms. Flipping function used in WBF is borrowed to determine the priority of scheduling. Simulation results show that it can provide a good tradeoff between FER performance and computation complexity for short-length LDPC codes.

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

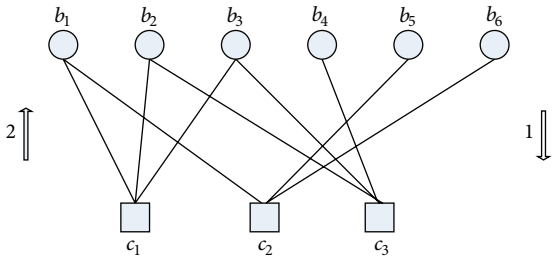

Figure 1: Standard BP algorithm (Flood).

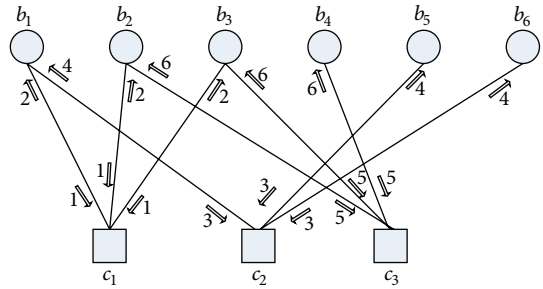

Figure 2: CSBP algorithm.

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

Algorithms	Updating	Reliability computation	Sorting
Flood	e	0	0
CSBP	e	0	0
IDS	$Md_c(d_v - 1)$	0	0
WBFSBP	e	0	0

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

Algorithms	Updating	Reliability computation	Sorting
Flood	e	0	0
CSBP	e	0	0
IDS	e	$Md_c(d_v - 1)(d_c - 1)$	$\mathcal{O}(N^2)$
WBFSBP	e	Md_cd_v	$\mathcal{O}(N \log N)$

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 $\mathbf{H}$. H_{mn} presents the entry of row m and column n in $\mathbf{H}$. The set $\mathcal{N}(m)$ denotes the nodes that adjoin

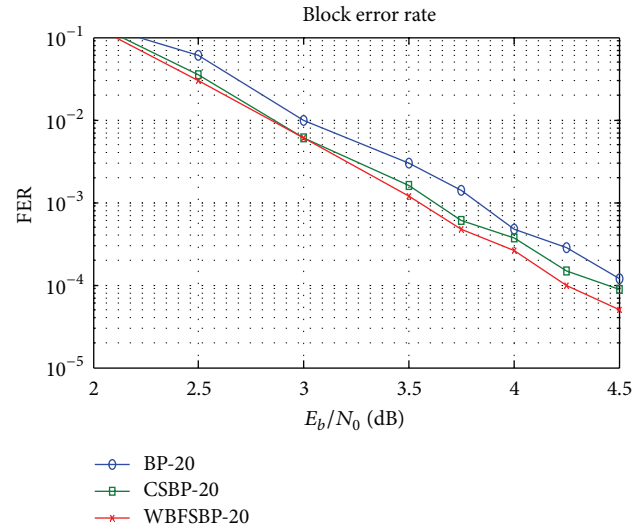

Figure 3: FER versus E_b/N_0 performance of BP (Flood), CSBP, and WBFSBP decoding algorithms with maximal number of iterations of 20 with LDPC code of blocklength of 256 and rate of 0.5. BP algorithm with maximal number of iterations of 20.

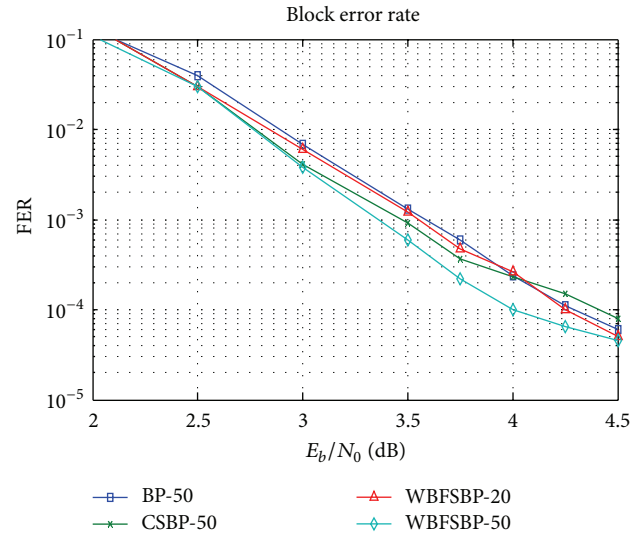

Figure 4: FER versus E_b/N_0 performance of BP (Flood), CSBP and WBFSBP decoding algorithms with maximal number of iterations of 50 with LDPC code of blocklength of 256 and rate of 0.5. As compared, performance of max iteration 20 of WBFSBP is given.

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 $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ into a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where $c_i = 2x_i - 1$, $i \in [1, N]$. The received vector is $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, where $y_i = x_i + n_i$, $i \in [1, N]$, and n_i is the AWGN. $\mathbf{Z} = (z_1, z_2, \ldots, z_n)$ denotes the hard-decision vector acquired from $\mathbf{y}$: $z_i = 1$ if $y_i > 0$ and $z_i = 0$ if $y_i \leq 0$.

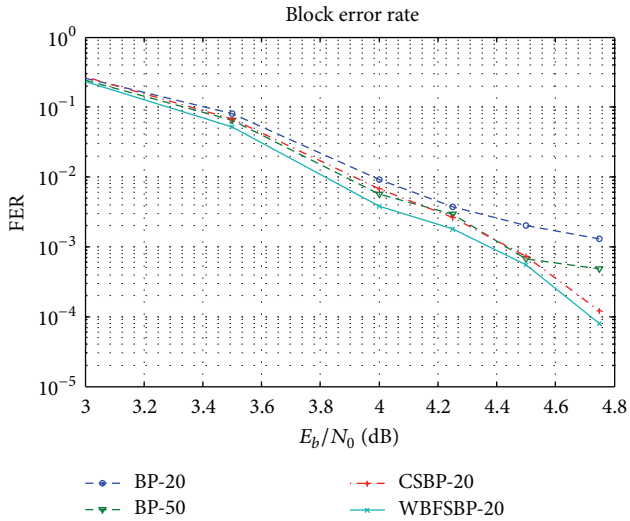

FIGURE 5: FER versus E_b/N_0 performance of BP (Flood), CSBP and WBFSBP decoding algorithms with maximal number of iterations of 20 with LDPC code of blocklength of 204 and rate of 0.5. BP-20 denotes BP algorithm with maximal number of iterations of 20. As compared, performance of max iteration 50 of BP is given.

$\mathbf{S} = (s_1, s_2, \ldots, s_M) = \mathbf{z} \cdot H^T$ denotes the syndrome of hard-decision vector $\mathbf{z}$.

2.1. WBF. Firstly, a metric called "reliability" and a function-associated flipping probability of each check-node are calculated via [9]

$$w_{n,m} = \min_{i \in \mathcal{N}(m) \backslash n} |y_i|, \quad m \in [1, M], \; n \in \mathcal{N}(m),$$
$$e_{\mathrm{WBF},n} = \sum_{m \in \mathcal{N}(n)} (2s_m - 1) w_{n,m} - \alpha |y_n|. \tag{1}$$

The WBF algorithm is described as follows [9]:

(1) set iteration number $k = 0$, take $k_{\max}$ as the maximum number of iterations, and pick up $w_{n,m}$,

(2) calculate $s^k = z^k H^T$; if $s^k = 0$, stop decoding and output z^k,

(3) for $n = 1, 2, \ldots, N$, calculate $e^k_{\mathrm{WBF},n}$ as follows:

$$e^k_{\mathrm{WBF},n} = \sum_{m \in \mathcal{N}(n)} \left(2s^k_m - 1\right) w_{n,m} - \alpha |y_n|, \tag{2}$$

(4) $z^{n^*} = z^{n^*} + 1$ based on

$$n^* = \arg \max_{n \in [1,N]} e^k_{\mathrm{WBF},n}, \tag{3}$$

(5) $k = k + 1$ and go to step (3), until stopping rules are satisfied.

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 $m_{c_i \to v_j} = f_{c_i \to v_j}(m)$ and $m_{v_j \to c_i} = f_{v_j \to c_i}(m)$ are defined as follows [2]:

$$m_{v_j \to c_i} = \sum_{c_a \in \mathcal{N}(v_j) \backslash c_i} m_{c_a \to v_j} + C_{v_j},$$
$$m_{c_i \to v_j} = 2 \operatorname{arctanh} \left(\prod_{v_b \in \mathcal{N}(c_i) \backslash v_j} \tanh \left(\frac{m_{v_b \to c_i}}{2} \right) \right), \tag{4}$$
$$L(v_j) = \sum_{c_a \in \mathcal{N}(v_j)} m_{c_a \to v_j} + C_{v_j},$$

where $C_{v_j} = \log(p(y_j \mid v_j = 0)/p(y_j \mid v_j = 1))$ 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 ($e^k_{\mathrm{WBF},n}$)

1: Set every $m_{c\to v} = 0$
2: Set every $m_{v_j\to c_i} = C_{v_j}$
3: **while** stop rules unsatisfied **do**
4: for all c_a $(a = 1, 2, \ldots, M)$
5: for each $v_b \in \mathcal{N}(c_a)$, compute and generate $m_{v_b\to c_a}$ $(a = 1, 2, \ldots, M)$ simultaneously
6: for each $v_b \in \mathcal{N}(c_a)$, compute and generate $m_{c_a\to v_b}$ $(a = 1, 2, \ldots, M)$ simultaneously
7: **end while**

Pseudocode 1: Pseudocodes of Flood.

1: Set every $m_{c\to v} = 0$
2: Set every $m_{v_j\to c_i} = C_{v_j}$
3: **while** stop rules unsatisfied **do**
4: **for** i **do**
5: for check-node c_i
6: **for** every $v_b \in \mathcal{N}(c_i)$ **do**
7: compute and generate $m_{v_b\to c_i}$
8: **end for**
9: **for** every $v_b \in \mathcal{N}(c_i)$ **do**
10: compute and generate $m_{c_i\to v_b}$
11: **end for**
12: **end for**
13: **end while**

Pseudocode 2: Pseudocodes of CSBP.

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

$$E^k_{\mathrm{WBF},m} = \max_{n\in\mathcal{N}(m)} e^k_{\mathrm{WBF},n} - \mathrm{sub}\max_{n\in\mathcal{N}(m)} e^k_{\mathrm{WBF},n}, \tag{5}$$

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 is almost nonsense. If all neighbor variable-nodes of a check-node are reliable, the priority of updating this check-node should be very low. So we concentrate on the check-node which has only one unreliable neighbor variable-node. The check-nodes with larger value of $E^k_{\mathrm{WBF},m}$ 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 node by node but is simply obtained from sorting the set of $E^k_{\mathrm{WBF},m}$ 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 $\{E^k_{\mathrm{WBF},m}\}$ 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 α is set to 1.3 in the computation of $E^k_{\mathrm{WBF},m}$. 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 $k_{\max}$ as the maximum number of iteration, pick up $w_{n,m}$,
Set every $m_{c\to v} = 0$, every $m_{v_j\to c_i} = C_{v_j}$.
1: Compute $s^k = z^k H^T$, if $s^k = \mathbf{0}$, stop decoding and output z^k.
2: For $n = 1, 2, \ldots N$, compute

$$e^k_{\mathrm{WBF},n} = \sum_{m\in\mathcal{N}(n)} (2s^k_m - 1)w_{n,m} - \alpha\left|y_n\right|$$

3: sort $E^k_{\mathrm{WBF},m}$ in descending order and put the corresponding check-nodes to U
4: **while** stop rules unsatisfied **do**
5: **for** i **do**
6: for ith check-node c_{U_i} in U
7: **for** every $v_b \in \mathcal{N}(c_{U_i})$ **do**
8: compute and generate $m_{v_b\to c_{U_i}}$
9: **end for**
10: **for** every $v_b \in \mathcal{N}(c_{U_i})$ **do**
11: compute and generate $m_{c_{U_i}\to v_b}$
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 $k_{\max}$ is set to 20 and α 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

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[2] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, 1996.

[3] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's "belief propagation" algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, 1998.

[4] F. R. Kschischang, B. J. R. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[5] Y. Zhang, Y. Jun, G. Wei, and L. Wu, "Find multi-objective paths in stochastic networks via chaotic immune PSO," *Expert Systems with Applications*, vol. 37, no. 3, pp. 1911–1919, 2010.

[6] Y. Zhang, L. Wu, G. Wei, and S. Wang, "A novel algorithm for all pairs shortest path problem based on matrix multiplication and pulse coupled neural network," *Digital Signal Processing*, vol. 21, no. 4, pp. 517–521, 2011.

[7] Y. Zhang and L. Wu, "Optimal multi-level thresholding based on maximum Tsallis entropy via an artificial bee colony approach," *Entropy*, vol. 13, no. 4, pp. 841–859, 2011.

[8] D. Burshtein, "On the errror correction of regular LDPC codes using the flipping algorithm," *IEEE Transactions on Information Theory*, vol. 54, no. 2, pp. 517–530, 2008.

[9] M. Jiang, C. Zhao, Z. Shi, and Y. Chen, "An improvement on the modified weighted bit flipping decoding algorithm for LDPC codes," *IEEE Communications Letters*, vol. 9, no. 9, pp. 814–816, 2005.

[10] X. Wu, C. Zhao, and X. You, "Parallel weighted bit-flipping decoding," *IEEE Communications Letters*, vol. 11, no. 8, pp. 671–673, 2007.

[11] X. Wu, C. Ling, M. Jiang, E. Xu, C. Zhao, and X. You, "New insights into weighted bit-flipping decoding," *IEEE Transactions on Communications*, vol. 57, no. 8, pp. 2177–2180, 2009.

[12] N. Miladinovic and M. P. C. Fossorier, "Improved bit-flipping decoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1594–1606, 2005.

[13] H. Kfir and I. Kanter, "Parallel versus sequential updating for belief propagation decoding," *Physica A*, vol. 330, no. 1-2, pp. 259–270, 2003.

[14] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 976–996, 2003.

[15] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS '04)*, pp. 107–112, October 2004.

[16] J. Zhang, Y. Wang, M. Fossorier, and J. S. Yedidia, "Replica shuffled iterative decoding," in *Proceedings of the IEEE International Symposium on Information Theory Proceedings (ISIT '05)*, pp. 454–458, 2005.

[17] J. Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209–213, 2005.

[18] A. I. V. Casado, M. Griot, and R. D. Wesel, "Improving LDPC decoders via informed dynamic scheduling," in *Proceedings of the IEEE Information Theory Workshop (ITW '07)*, pp. 208–213, Tahoe City, CA, USA, September 2007.

[19] A. I. V. Casado, M. Griot, and R. D. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 932–937, Glasgow, UK, June 2007.

[20] D. Levin, S. Litsyn, and E. Sharon, "Lazy scheduling for LDPC decoding," *IEEE Communications Letters*, vol. 11, no. 1, pp. 70–72, 2007.

[21] A. I. V. Casado, M. Griot, and R. D. Wesel, "LDPC decoders with informed dynamic scheduling," *IEEE Transactions on Communications*, vol. 58, no. 12, pp. 3470–3479, 2010.

[22] J.-H. Kim, M.-Y. Nam, and H.-Y. Song, "Variable-to-check residual belief propagation for LDPC codes," *Electronics Letters*, vol. 45, no. 2, pp. 117–118, 2009.

[23] G. J. Han and X. Liu, "An efficient dynamic schedule for layered belief-propagation decoding of LDPC codes," *IEEE Communications Letters*, vol. 13, no. 12, pp. 950–952, 2009.

[24] Y. Gong, X. Liu, W. Ye, and G. J. Han, "Effective informed dynamic scheduling for belief propagation decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 59, no. 10, pp. 2683–2691, 2011.