Dynamic Layered Decoding Scheduling for LDPC Codes Aided by Check Node Error Probabilities

Chenyuan Jia, Dongxu Chang, Ruiyuan Wang, Guanghui Wang, Guiying Yan, Cunquan Qu

***Abstract*—In this study, a new scheduling strategies for low-density parity-check (LDPC) codes under layered belief propagation (LBP) is designed. Based on the criteria of prioritizing the update of check nodes with lower error probabilities, we propose two dynamic scheduling methods: dynamic error belief propagation (Dyn-EBP) and dynamic penalty error belief propagation (Dyn-PEBP). In Dyn-EBP, each check node is restricted from being updated the same number of times, whereas Dyn-PEBP removes this restriction and instead introduces a penalty term to balance the number of updates. Simulation results show that, for 5G new radio (NR) LDPC codes, our proposed scheduling methods can outperform existing dynamic and offline scheduling strategies under various blocklengths and code rates. This demonstrates that prioritizing the update of check nodes with lower error probabilities can lead to higher decoding efficiency and validates the effectiveness of our algorithms.**

***Index Terms*—Low-density parity-check code, scheduling scheme, decoding efficiency**

I. Introduction

LOW-DENSITY parity-check (LDPC) code [1] has been widely used in communication fields, such as 5G new radio (NR) [2], IEEE 802.11 (WiFi) [3]. It was demonstrated by D. MacKay and M. Neal in 1996 [4] that LDPC codes can approach the Shannon limit. The belief propagation (BP) decoding algorithm [5] of LDPC code is most commonly used due to its low complexity and good performance in practice.

The conventional BP algorithm employs a flooding scheduling strategy to conduct message-passing between variable nodes and check nodes. This strategy updates all variable-to-check (V2C) messages or check-to-variable (C2V) messages simultaneously in one iteration, achieving a high degree of parallelism. However, the flooding BP algorithm is known for its slow convergence speed. This is because the latest information available in the current iteration can only be utilized in the subsequent iteration. The delayed information may hinder the benefits of information transmission, thus affecting the algorithm's convergence rate [6].

In 2004, D. Hocevar introduced the concept of layered belief propagation (LBP) decoding [6], which updates information sequentially. This decoding strategy can accelerate the decoding process as it ensures that the most recent information is disseminated. In comparison to the flooding scheduling strategy, LBP can reduce decoding complexity by nearly 50% while achieving the same performance. Despite its benefits, LBP does not have a specific design for the decoding sequence order. To further reduce decoding complexity, scheduling the order of decoding sequences is a key direction for improving LBP. In [7]–[10], several dynamic scheduling have been proposed. These policies determine the scheduling sequences based on the latest information from the previous updates, meaning the decoding order is established during the decoding process.

However, existing dynamic scheduling approaches often rely on heuristic criteria, such as residual magnitudes or syndrome-based checks, which tend to yield limited performance gains. In [11], the authors demonstrate—over both binary erasure channels (BEC) and binary-input additive white Gaussian noise (AWGN) channels—that an efficient decoding schedule should prioritize the update of check nodes with lower error probabilities. Based on this insight, a simple scheduling algorithm utilizing only channel information to characterize error probabilities was proposed in [11].

To further improve decoding performance, this study analyzes the evolution of check node error probabilities and proposes the dynamic error belief propagation (Dyn-EBP) algorithm, in which each check node is strictly constrained to be updated only once per iteration. In addition, a more flexible version—dynamic penalty error belief propagation (Dyn-PEBP)—is proposed, which introduces elastic constraints on the number of check node updates per iteration.

Simulation results over 5G NR LDPC codes demonstrate that both Dyn-EBP and Dyn-PEBP outperform existing scheduling strategies, including but not limited to LBP, least-punctured and highest-degree (LPHD) scheduling [12], residual-decaying-based residual belief propagation (RD-RBP) [10], and the method in [11]. These results confirm that prioritizing the update of low-error-probability check nodes significantly improves decoding efficiency and validate the effectiveness of the proposed algorithms.

This paper is organized as follows. Section II provides an introduction to LDPC decoding, LBP, and scheduling sequences. Section III analyzes decoding errors. Section IV presents the implementation and complexity analysis of Dyn-EBP and Dyn-PEBP. Simulation results are compared and discussed in Section V. Finally, Section VI concludes the study and outlines future work.

This work is partially supported by the National Key R&D Program of China, (2023YFA1009600).

C. Jia, D. Chang, G. Wang and C. Qu are with the School of Mathematics, Shandong University, Shandong, China (e-mail: chenyuanjia@mail.sdu.edu.cn; dongxuchang@mail.sdu.edu.cn; ghwang@sdu.edu.cn; cqqu@sdu.edu.cn).

G. Yan is with the Academy of Mathematics and Systems Science, CAS, University of Chinese Academy of Sciences, Beijing, 100190 China (e-mail: yangy@amss.ac.cn).

R. Wang is with the Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China (e-mail: rywang24@mails.tsinghua.edu.cn).

II. Preliminaries

A. Belief Propagation for LDPC Codes

A brief review of the iterative BP decoding algorithm [5] is provided to facilitate the analysis of scheduling sequences. A bipartite graph, named Tanner graph, consisting of N variable nodes v_j, $j = 1, 2, \cdots, N$, and M check nodes c_i, $i = 1, 2, \cdots, M$, is used to facilitate the description of LDPC decoding.

The message-passing process of BP decoding consists of V2C message updates and C2V message updates. The message update rule of V2C is

$$m_{v_j \to c_i}^{(l)} = C_{v_j} + \sum_{k \in \mathcal{N}(v_j) \backslash c_i} m_{c_k \to v_j}^{(l-1)}, \tag{1}$$

and the message update rule of C2V is

$$m_{c_i \to v_j}^{(l)} = 2 \tanh^{-1} \left[\prod_{h \in \mathcal{N}(c_i) \backslash v_j} \tanh \left(m_{v_h \to c_i}^{(l-1)} / 2 \right) \right], \tag{2}$$

where C_{v_j} is the channel message of variable node v_j in log-likelihood ratio (LLR) form, l is the number of iterations, $N(v)$ represents the index set of nodes connected directly to node v, $m_{i \to \alpha}^{(l)}$ means the message from variable node i to check node α in iteration l and $m_{\alpha \to i}^{(l)}$ means the message from check node α to variable node i in iteration l. The initial message $m_{i \to \alpha}^{(0)}$ and $m_{\alpha \to i}^{(0)}$ is 0.

The above procedure iterates until the predefined stopping criterion is met, i.e., either all of the parity-check constraints are satisfied or the predefined maximum number of iterations is reached. At this point let l_{end} denote the final number of iterations. The LLR of variable node v_j after the decoding can be calculated as

$$L_j^{(l)} = C_{v_j} + \sum_{i \in \mathcal{N}(v_j)} m_{c_i \to v_j}^{(l-1)}, \tag{3}$$

then the transmitted bits x_j, $j = 1, 2, \cdots, N$, are estimated from the hard decision according to

$$x_j = \begin{cases} 0, & \text{if } L_{v_j} \geq 0 \\ 1, & \text{otherwise} \end{cases}. \tag{4}$$

B. Layered Belief Propagation and Scheduling Sequences

In each decoding iteration, LBP [6] sequentially selects each check node in the graph and updates the information on all edges connected to the selected check node. Specifically, it performs V2C message updates along all edges connected to the selected check node, followed by C2V message updates. The order in which check nodes are selected for updating in LBP is referred to as the scheduling sequence, defined as follows:

Definition 1 (scheduling sequences)**.** *The scheduling sequence is a sequence composed of the indices of check nodes. It represents the order in which check nodes are selected for decoding in LBP, with the decoder selecting check nodes for decoding according to the order in which they appear in the scheduling sequences.*

When the same message-passing order is used in each iteration, the scheduling sequence can consist of only the check node update order within one iteration.

Different scheduling sequences can have a great influence on the message-passing efficiency in layered decoding, consequently affecting the decoding performance and latency [13].

III. Error Probability of Check Node

It has been shown in [11] that efficient scheduling sequences should prioritize updating check nodes with lower error probabilities, as such nodes are more likely to correct erroneous information. This section examines the error probability of a check node conditioned on the current LLR messages received from its neighboring variable nodes.

Definition 2 (check node error)**.** *An error occurs in a check node if an odd number of errors occur in its neighboring variable nodes.*

Lemma 1. *For an LDPC code with N variable nodes, denote the LLR value of the j-th variable node as L_j $j = 1, 2, \cdots, N$. Then, the error probability of the j-th variable node can be expressed as follows.*

$$p_j^{v_\epsilon} = \frac{1}{1 + \left[\frac{1}{\lambda} I(L_j < 0) + \lambda I(L_j \geq 0) \right] e^{|L_j|}}, \tag{5}$$

where $\lambda = \frac{P(x_j=0)}{P(x_j=1)}$ is the prior ratio, and $I(\cdot)$ denotes the indicator function.

Proof. Considering first the case $L_j > 0$, by a hard decision, the transmitted bits x_j will be estimated as $x_j = 0$. Thus the error probability of the j-th variable node can be defined as $p_j^{v_\epsilon} = \mathbb{P}(x_j = 1 | y_j)$. Its equational relationship with LLR is derived below.

$$\begin{aligned} L_j &= \log \left(\frac{\mathbb{P}(y_j | x_j = 0)}{\mathbb{P}(y_j | x_j = 1)} \right) \\ e^{L_j} &= \frac{\mathbb{P}(y_j | x_j = 0)}{\mathbb{P}(y_j | x_j = 1)} \\ e^{L_j} &= \frac{\mathbb{P}(x_j = 0 | y_j)}{\mathbb{P}(x_j = 1 | y_j)} \frac{\mathbb{P}(x_j = 1)}{\mathbb{P}(x_j = 0)} \\ 1 + \lambda e^{L_j} &= \frac{1}{\mathbb{P}(x_j = 1 | y_j)} \\ \mathbb{P}(x_j = 1 | y_j) &= \frac{1}{1 + \lambda e^{L_j}}. \end{aligned}$$

Similarly, where $L_j < 0$, the error probability of the j-th variable node can be defined as $p_j^{v_\epsilon} = \mathbb{P}(x_j = 0 | y_j) = \frac{1}{1 + \frac{1}{\lambda} e^{-L_j}}$. Considering the above situations, Eq.(5) holds. □

For $\lambda = 1$, the error probability simplifies to [14] [15]:

$$p_j^{v_\epsilon} = \frac{1}{1 + e^{|L_j|}}. \tag{6}$$

Lemma 2 (Gallager [1])**.** *For an LDPC code with M check nodes, the LLR L_j is given for the j-th variable node. If the*

prior ratio $\lambda = 1$, the error probability of the i-th check node will be expressed as follows.

$$p_i^{c_\epsilon} = \frac{1}{2}\left[1 - \prod_{j \in N(c_i)} \left(1 - \frac{2}{1 + e^{|L_j|}}\right)\right], \tag{7}$$

where $i = 1, 2, \cdots, M$.

IV. The proposed methods

A. Dyn-EBP decoding

In [11], decoding scheduling was analyzed for the first time from the perspective of error probability direction. The author proposed a dynamic scheduling strategy based solely on channel information. However, since additional useful information is generated during the decoding process, we aim to combine both the channel information and the messages generated from the decoding process to design a more adaptive and effective scheduling strategy.

Many dynamic scheduling algorithms have been designed from the perspective of maximizing decoding residuals [7]–[10]. However, such approaches often lead to imbalanced updates, where certain check nodes are repeatedly updated while others are rarely, if ever, selected. This uneven update distribution can adversely affect the overall decoding performance [7]. To address this issue, we propose Dyn-EBP, a dynamic scheduling algorithm that ensures all check nodes are updated once in each iteration.

To simplify the decoding process, the Dyn-EBP algorithm performs dynamic scheduling of check nodes based directly on their error probabilities. Initially, all messages $m_{v_j \to c_i}$ are set to the corresponding channel message C_{v_j}. The algorithm maintains a dynamically updated set P of check nodes, ordered in ascending order of their error probabilities $p_i^{c_\epsilon}$. After a check node completes its message update, it is immediately removed from P, ensuring that each check node is updated only once per iteration. If necessary, it will be reinserted into the set in the next iteration.

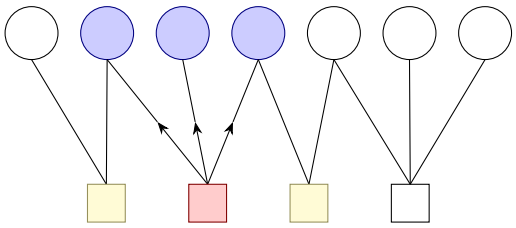


Fig. 1: An example of Dyn-EBP performing a C2V where the red check node c_k propagates messages to the blue variable nodes v_a and influences the error probability $p_b^{c_\epsilon}$ of the neighbors of these variable nodes - the yellow check nodes.

Then, as shown in Fig. 1, without loss of generality, we assume that the check node c_k has the error probability $p^{c_\epsilon *}$, which is the smallest of all check nodes. After $m_{c_k \to v_a}$ is propagated, only $p_b^{c_\epsilon}$ need to be recomputed with $b \in \mathcal{N}(v_a) \backslash c_k$. After updating the error probabilities of the check nodes, a reordering is required to select the check node with the smallest error probability for the next iteration. Specific algorithmic details can be found in Algorithm 1.

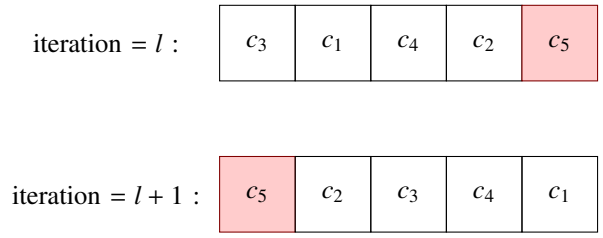


Fig. 2: An example of two neighboring scheduling sequences where the red block indicates the check node for successive repeated updates.

There exists a possibility that the tail of the current iteration's schedule overlaps with the head of the next iteration. For example, in Fig. 2, the last check node in the scheduling sequence of the l-th iteration is c_5, which also appears as the first check node in the scheduling sequence of the $(l+1)$-th iteration. As a result, c_5 performs message propagation consecutively without receiving any new incoming information, leading to redundant updates. This redundancy is objectively detrimental to decoding performance. Therefore, it is necessary to minimize the likelihood of such overlap as much as possible.

Algorithm 1 Dyn-EBP decoding for LDPC codes

Initialize all $m_{c_i \to v_j} = 0$, $m_{v_j \to c_i} = C_{v_j}$, $L_j = C_{v_j}$
Compute all $p_i^{c_\epsilon}$ and generate P
while $P \neq \emptyset$ **do**
    Let $p_k^{c_\epsilon}$ be the first message in P
    Let $P = P \backslash \{p_k^{c_\epsilon}\}$
    for every $a \in \mathcal{N}(c_k)$ **do**
        Generate and propagate $m_{v_a \to c_k}$
    end for
    for every $a \in \mathcal{N}(c_k)$ **do**
        Generate and propagate $m_{c_k \to v_a}$
        Compute $L_a = L_a + m_{c_k \to v_a}$
        for every $b \in \mathcal{N}(v_a) \backslash c_k$ **do**
            Compute $p_b^{c_\epsilon}$
        end for
    end for
    Reorder P
end while
if Stopping rule is not satisfied **then**
    Position=2
end if

B. Dyn-PEBP decoding for LDPC codes

To better address the issues discussed in Subsection IV-A and enhance algorithmic flexibility—specifically, relaxing the constraint that each check node can be updated only once per iteration—this study introduces a penalty term on the number of updates into the error probability calculation. This penalty limits the scheduling intervals of repeated updates to the same check nodes across neighboring iterations, thereby improving

decoding efficiency. The improved iterative penalized error probability for check node i is defined as

$$\tilde{p}_i^{c_\epsilon} = p_i^{c_\epsilon} + f(l_i), \tag{8}$$

where l_i denotes the number of times the i-th check node has been updated, and $f(l_i)$ is a penalty function. For simplicity, we set $f(l_i) = \gamma \cdot l_i$, with $\gamma \in [0, 1]$ denoting the penalty coefficient.

Unlike Dyn-EBP, this algorithm no longer requires removing the smallest element from the set P; instead, it employs a greedy update strategy guided by (8). Detailed steps are provided in Algorithm 2.

Regarding the penalty factor γ, when γ approaches 0, the algorithm behaves greedily by focusing primarily on minimizing error probability. Conversely, as γ approaches 1, the penalty term becomes more dominant (since error probabilities lie within the range [0, 1]), which effectively discourages repeated updates of check nodes within each iteration. The special case of $\gamma = 1$ corresponds exactly to the Dyn-EBP mode.

Algorithm 2 Dyn-PEBP decoding for LDPC codes

Initialize all $m_{c_i \to v_j} = 0$, $m_{v_j \to c_i} = C_{v_j}$, $L_j = C_{v_j}$, $l_i = 0$
Compute all $\tilde{p}_i^{c_\epsilon}$ and generate P
Let $\tilde{p}_k^{c_\epsilon}$ be the first message in P
for every $a \in \mathcal{N}(c_k)$ **do**
    Generate and propagate $m_{v_a \to c_k}$
end for
for every $a \in \mathcal{N}(c_k)$ **do**
    Generate and propagate $m_{c_k \to v_a}$
    Compute $L_a = L_a + m_{c_k \to v_a}$
    for every $b \in \mathcal{N}(v_a)$ **do**
        Compute $\tilde{p}_b^{c_\epsilon}$
    end for
end for
Let $l_k = l_k + 1$
Compute $\tilde{p}_k^{c_\epsilon}$
Reorder P
if Stopping rule is not satisfied **then**
    Position=3
end if

C. Algorithmic complexity analysis

We analyze the algorithmic complexity of the proposed dynamic scheduling algorithms, namely Dyn-EBP and Dyn-PEBP. Let E denote the total number of edges in the Tanner graph with M check nodes and N variable nodes, such that $E = \bar{d}_v N = \bar{d}_c M$, where $\bar{d}_v$ and $\bar{d}_c$ represent the average degrees of variable and check nodes, respectively.

The main computational steps per iteration are as follows:

- **V2C Update**: Each variable node sends messages to its connected check nodes. As each variable node has an average degree of $\bar{d}_v$, this step requires $O(E)$ operations per iteration.
- **C2V Update**: Each check node receives messages from neighboring variable nodes and sends updated messages. With an average degree of $\bar{d}_c$, this step also requires $O(E)$ operations.
- **Error Probability Calculation**: The error probability of each check node is computed from its incoming messages. According to Fig. 1, each message affects $O(\bar{d}_v)$ check nodes. Since each error probability involves a product of $O(\bar{d}_c)$ terms, this step incurs $O(E\bar{d}_v\bar{d}_c)$ operations per iteration.
- **Dynamic Reordering**: To prioritize check node updates based on current error probabilities, a priority queue P is maintained. A direct update of all M nodes results in a complexity of $O(M^2 \log M)$. To reduce this cost, we employ a Fibonacci heap [16], which supports `INSERT`, `MINIMUM`, and `DECREASE-KEY` operations in amortized $\Theta(1)$ time. An empty heap is initialized and filled with M check nodes using their error probabilities at a cost of $\Theta(M)$. During message updates, the check node with the lowest error probability is selected, and affected check nodes are updated. Each such operation costs at most $O(\bar{d}_c\bar{d}_v)$, assuming updates do not increase error probabilities. Therefore, the worst-case complexity of this step per iteration is $O(M\bar{d}_c\bar{d}_v) = O(E\bar{d}_v)$.

Combining all components, the overall computational complexity per iteration of Dyn-EBP and Dyn-PEBP is $O(E\bar{d}_v\bar{d}_c)$. This is lower than that of conventional residual-based scheduling algorithms [7]–[10], particularly due to the efficient reordering enabled by Fibonacci heap optimization.

V. Simulation Results

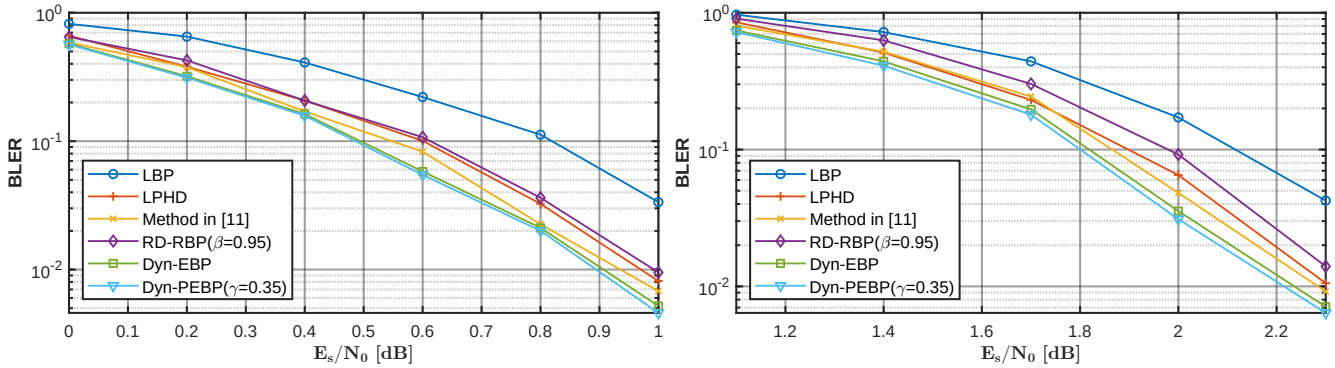
In this section, we evaluate the performance of the proposed algorithms through numerical simulations under various coding rates, blocklengths, and signal-to-noise ratios (SNRs). The benchmark algorithms include LBP, LPHD scheduling [12], RD-RBP [10] and the method in [11]. The experiments are conducted on the 5G NR LDPC base graph 1 (BG1). To accommodate high-throughput decoding requirements, the number of decoding iterations is fixed at five. Considering that puncturing sets the initial error probabilities of neighboring check nodes to 0.5, the lowest-degree check node connected to exactly one punctured variable node is selected to be the first update check node.

We evaluate QC layer-based scheduling by averaging the error probabilities of corresponding check nodes in the lifted graph. The parameter γ in Dyn-PEBP is empirically tuned and found to perform best around 0.35. Similarly, the parameter β in RD-RBP is set to 0.95, which aligns with the optimal value reported in [10].

Fig. 3 illustrates the BLER performance across various SNRs, blocklengths, and code rates. Error-probability-based scheduling consistently delivers superior performance over the baseline methods.

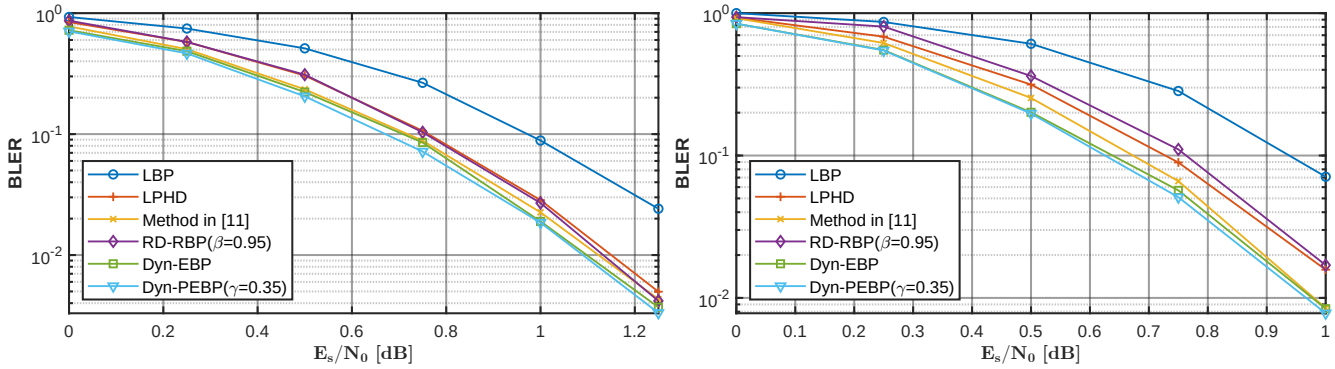
VI. Conclusions

This work proposes error-probability-based scheduling strategies for LDPC decoding—Dyn-EBP and its penalty-enhanced variant Dyn-PEBP—evaluated under the 5G NR



(a) Decoding of a blocklength-1482 rate-0.386 LDPC code.

(b) Decoding of a blocklength-1170 rate-0.4681 LDPC code.



(c) Decoding of a blocklength-1320 rate-0.4 LDPC code.

(d) Decoding of a blocklength-2640 rate-0.4 LDPC code.

Fig. 3: BLER performance of LBP, LPHD scheduling, method in [11], RD-RBP, Dyn-EBP, and Dyn-PEBP under QC layer-based scheduling versus E_s/N_0 at iteration 5.

LDPC BG1 framework. These methods efficiently prioritize the updating of reliable check nodes and are well suited for high-throughput decoding scenarios. Future directions include optimizing the trade-off parameter γ, analyzing the role of $f(l_i)$ across iterations, and exploring parallel implementations to meet high-throughput demands. The proposed framework may also extend to other BP-based systems like turbo codes, iterative detection, and graphical model inference.

References

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

[2] 3GPP, "5G NR; multiplexing and channel coding (release 15)," *3GPP TS 38.212 v15. 2.0*, 2018.

[3] I. C. S. L. S. Committee *et al.*, "Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications," *IEEE Std 802.11ˆ*, 2007.

[4] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, 1997.

[5] D. J. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[6] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *IEEE Workshop on Signal Processing Systems, 2004. SIPS 2004.*, pp. 107–112, IEEE, 2004.

[7] A. I. V. Casado, M. Griot, and R. D. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *2007 IEEE International Conference on Communications*, pp. 932–937, IEEE, 2007.

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

[9] X. Liu, Y. Zhang, and R. Cui, "Variable-node-based dynamic scheduling strategy for belief-propagation decoding of LDPC codes," *IEEE Communications Letters*, vol. 19, no. 2, pp. 147–150, 2014.

[10] H. Zhang and S. Chen, "Residual-decaying-based informed dynamic scheduling for belief-propagation decoding of LDPC codes," *IEEE Access*, vol. 7, pp. 23656–23666, 2019.

[11] D. Chang, G. Wang, G. Yan, and Z. Ma, "Analysis of efficient scheduling in layered decoding of LDPC codes," in *2024 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2024. Officially accepted and presented; indexing pending.

[12] B. Wang, Y. Zhu, and J. Kang, "Two effective scheduling schemes for layered belief propagation of 5G LDPC codes," *IEEE Communications Letters*, vol. 24, no. 8, pp. 1683–1686, 2020.

[13] D. Chang, G. Wang, G. Yan, and D. Yin, "An optimization model for offline scheduling policy of low-density parity-check codes," *IEEE Communications Letters*, vol. 28, no. 8, pp. 1740–1744, 2024.

[14] P. Hoeher, I. L, and U. Sorger, "Log-likelihood values and monte carlo simulation – some fundamental results," 08 2000.

[15] N. ul Hassan, M. Lentmaier, and G. P. Fettweis, "Comparison of LDPC block and LDPC convolutional codes based on their decoding latency," in *2012 7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pp. 225–229, 2012.

[16] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM*, vol. 34, pp. 596–615, July 1987.