Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 9517840, 16 pages
https://doi.org/10.1155/2020/9517840

Research Article

Design of Improved BP Decoders and Corresponding LT Code Degree Distribution for AWGN Channels

Lei Zhang and Li Su

Information Engineering College, Capital Normal University, Beijing 100048, China

Correspondence should be addressed to Lei Zhang; 6576@cnu.edu.cn

Received 28 January 2020; Revised 16 September 2020; Accepted 3 October 2020; Published 20 October 2020

Academic Editor: Chaoyun Song

This paper presents the performance of a hard decision belief propagation (HDBP) decoder used for Luby transform (LT) codes over additive white Gaussian noise channels; subsequently, three improved HDBP decoders are proposed. We first analyze the performance improvement of the sorted ripple and delayed decoding process in a HDBP decoder; subsequently, we propose ripple-sorted belief propagation (RSBP) as well as ripple-sorted and delayed belief propagation (RSDBP) decoders to improve the bit error rate (BER). Based on the analysis of the distribution of error encoded symbols, we propose a ripple-sorted and threshold-based belief propagation (RSTBP) decoder, which deletes low-reliability encoded symbols, to further improve the BER. Degree distribution significantly affects the performance of LT codes. Therefore, we propose a method for designing optimal degree distributions for the proposed decoders. Through simulation results, we demonstrate that the proposed RSBP and RSDBP decoders provide significantly better BER performances than the HDBP decoder. RSDBP and RSTDP combined with the proposed degree distributions outperformed state-of-the-art degree distributions in terms of the number of encoded symbols required to recover an input symbol correctly (NERRIC) and the frame error rate (FER). For a hybrid decoder formulated by combining RSDBP with a soft decision belief propagation decoder, the proposed degree distribution outperforms the other degree distributions in terms of decoding complexity.

1. Introduction

The Luby transform (LT) codes proposed in [1] are the first practical fountain code that performs well on reliable communications over a binary erasure channel (BEC). Successful hard decision belief propagation (HDBP) decoding is possible when$(1+\varepsilon)k$ encoded symbols are available, where ε is the overhead of decoding. With the advantage of being rateless, LT codes have been introduced in broadcast services and noisy channels [2]. The performance of LT codes over additive white Gaussian noise (AWGN) channels has been investigated in [3]. To improve decoding performance, soft information is used in a soft decision belief propagation (SDBP) decoder, which is used as the decoding algorithm over noisy channels [4].

Different strategies have been proposed to improve the performance of LT codes over AWGN channels. A Gauss–Jordan-elimination-assisted belief propagation (BP) decoder was proposed to address the premature termination of BP decoding [5]. However, it is only practical for short LT codes. Generally, an SDBP decoder begins when all encoded symbols are available. Therefore, in greedy spreading serial decoding, encoded symbols are processed at once, and messages propagated greedily to improve the convergence speed [6]. However, the increase in decoding complexity was demonstrated in [5, 6]. A cross-level decoding scheme that combines LT codes with low-density parity check (LDPC) codes was proposed [7]. Although this method provided an effective decoding scheme, it required additional bit decoding from the LDPC, thereby increasing the decoding complexity. The piggybacking BP decoding algorithm, which decreases the decoding overhead and decoding delay, was proposed for repeated accumulated (RA) rateless codes [8]. However, it is only useful for RA rateless codes. A parallel soft iterative decoding algorithm was proposed for satellite systems [9]. Similar to the study in [7], it is only effective when combining LDPC codes with LT codes in the physical layer. A low-complexity BP decoder was proposed to improve

Pseudocode of LT encoding
Input: input symbols $X = (x_1, x_2, \cdots, x_k)$, degree distribution $\Omega(d)$
Output: an encoded symbol c
initialize an encoded symbol c=0
select a degree d from [1, k] according to $\Omega(d)$
select d different input symbols from X and add to a neighbor set V
for input symbol v in V**do**
    c=c XOR v
end for
return$c$

Algorithm 1.

performance by deleting low-reliability symbols at the cost of a slight transmission efficiency loss [10]. The BP-based algorithm is combined with the log likelihood ratio- (LLR-) based adaptive demodulation (ADM) algorithm to further reduce the decoding complexity [11]. The maximum a posteriori probability-based ADM algorithm was proposed to improve performance by discarding incorrect bits [12]. An adaptive decoding algorithm was proposed to reduce the decoding complexity by reducing the number of active check nodes [13], which degraded the performance of LT codes. In [10–13], the decoding complexity was reduced at the expense of increasing overhead because unreliable symbols were deleted. The trade-off between performance and decoding complexity was analyzed in [14]. Reducing the decoding complexity is important for the practicability of LT codes over noisy channels. However, the decoding complexity of the SDBP decoder remains high.

Several degree distributions have been proposed for LT codes over AWGN channels. An optimization process is formulated to design a new degree distribution, which improves the performance of LT codes over AWGN channels [15]. Three types of check-node degree distributions are proposed to improve the performance of systematic LT codes over AWGN channels [16]. A novel optimization model was proposed to design degree distributions over AWGN multiple access channels [17]. A ripple-based design of the degree distribution for AWGN channels was proposed in [18]. However, designing a good degree distribution and improving the performance in HDBP decoding over noisy channels remain an open problem.

Compared with SDBP decoding, HDBP decoding significantly reduced decoding complexity, which is extremely important for battery-powered equipment. The use of HDBP decoding can effectively reduce the decoding complexity of the hybrid decoding scheme, in which SDBP decoding will be invoked when HDBP decoding fails. Herein, the performance of HDBP decoding is analyzed, and improved HDBP decoders and their corresponding degree distributions are proposed. First, we investigate the ripple size throughout the decoding process and argue that sorting encoded symbols in ripple improves decoding performance; subsequently, we propose a ripple-sorted BP (RSBP) decoder. Based on the RSBP decoder, we discovered that with more encoded symbols available before decoding started, the decoding performance improved. Hence, we

Pseudocode of hard decision BP decoding
Input: encoded symbols received from channels
Output: recovered input symbols $\widehat{X}$
initialize ripple R as an empty queue
initialize recovered input symbols $\widehat{X}$ as an array
initialize waited encoded symbols Y as an array
whilesizeof$(\widehat{X}) < k$**do**
    receive an encoded symbol y from channels
    XORs$(\widehat{X}, y)$
    degree$(y) == 1$?push(R, y): push(Y, y)
    whilesizeof$(R) > 0$**do**
        dequeue an input symbol $\widehat{x}$ from R
        push$(\widehat{X}, x)$
        for encoded symbol y in Y**do**
            XOR$(y, \widehat{x})$
            degree$(y) == 1$?push(R, y): push(Y, y)
        end for
    end while
end while
return$\widehat{X}$

Algorithm 2.

propose an improved BP decoder known as a ripple-sorted and delayed BP (RSDBP) decoder. Based on the analysis of the distribution of error encoded symbols, we argue that low-reliability encoded symbols should be deleted to improve decoding performance and propose a ripple-sorted and threshold-based BP (RSTBP) decoder. Second, by analyzing the random walk model, we propose a method to generate a set of candidate ripple-size evaluations. A ripple-based design of degree distribution known as the generalised degree distribution algorithm (GDDA) is used to generate the degree distribution [19]. Based on the Monte Carlo method, the optimal degree distribution for a specific BP decoder is achieved. Simulation results demonstrated that our proposed RSBP and RSDBP decoders outperformed the BP decoder in terms of the bit error rate (BER) performance. Additionally, RSDBP and RSTDP combined with the proposed degree distributions outperformed state-of-the-art degree distributions in terms of the number of encoded symbols required to recover an input symbol correctly (NERRIC) and the frame error rate (FER). For the hybrid decoder formulated by combining

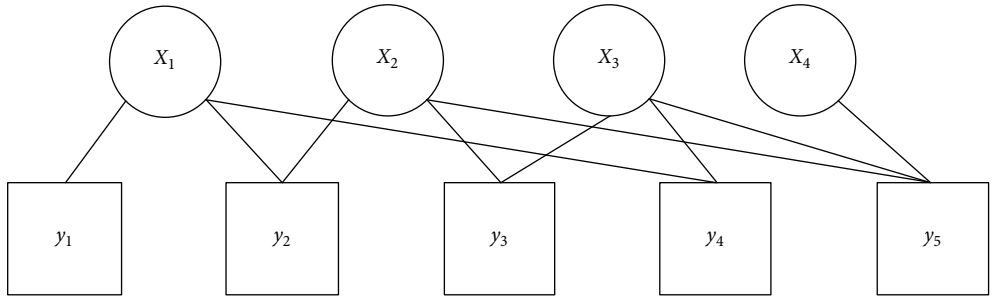

Figure 1: An example of tanner graph.

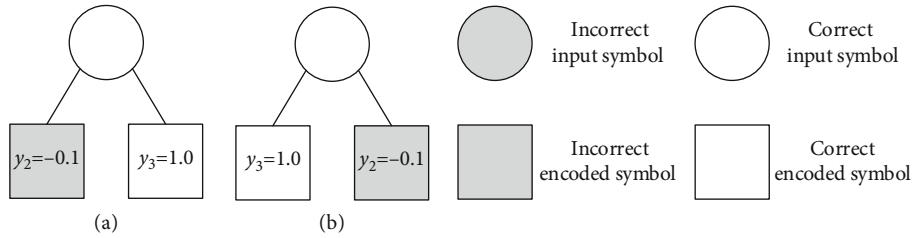

Figure 2: (a) lower-reliable encoded symbol arrived first. (b) higher-reliable encoded symbol arrived first.

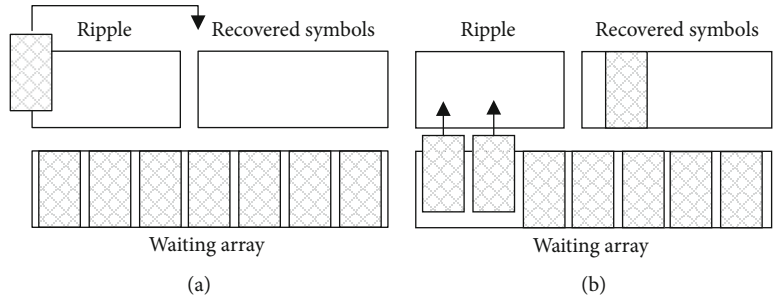

Figure 3: (a) Get a symbol from ripple and XOR with encoded symbols in waiting array. (b) Add the degree-one encoded symbols to the ripple.

RSBP with an SDBP decoder, the proposed degree distribution outperformed state-of-the-art degree distributions in terms of decoding complexity.

The remainder of this paper is organised as follows. In Section 2, a review of the system model and the encoding and decoding of LT codes are provided. In Section 3, the performance of HDBP decoding is analyzed. In Section 4, our RSBP, RSDBP, and RSTBP decoders are presented. In Section 5, the performance of the proposed decoders is analyzed. In Section 6, a method to generate the optimal degree distribution for a specific BP decoder is proposed. In Section 7, our experimental design is outlined, and the efficiency of the proposed decoders and the proposed degree distribution are demonstrated by experimental results. Finally, our study is summarised as follows.

2. Background

2.1. System Model. Information messages must be transmitted from the source to the destination over AWGN channels. Messages are partitioned into blocks, and each block is partitioned into symbols. The input symbols of the LT codes are denoted as $X = (x_1, x_2, \cdots, x_k)$, which is a combination of original symbols and a cyclic redundancy check (CRC). Typically, a single input symbol can be one bit or even a packet. For simplicity, one bit is regarded as an input symbol in this study. At the source, a stream of encoded symbols $C = (c_1, c_2, \cdots, c_N, \cdots)$ is generated from k input symbols. The encoded symbol c_j is modulated by the binary phase shift keying and transmitted to the destination independently as s_i. At the destination, the output of the AWGN channels for each symbol s_j is as follows:

$$y_j = s_j + n_j, \tag{1}$$

where $n_i \sim N(0, N_0/2)$, with $N(\cdot)$ being the normal distribution. At the destination, $(1+\varepsilon)k$encoded symbols are received to recover the k input symbol. Generally, a soft

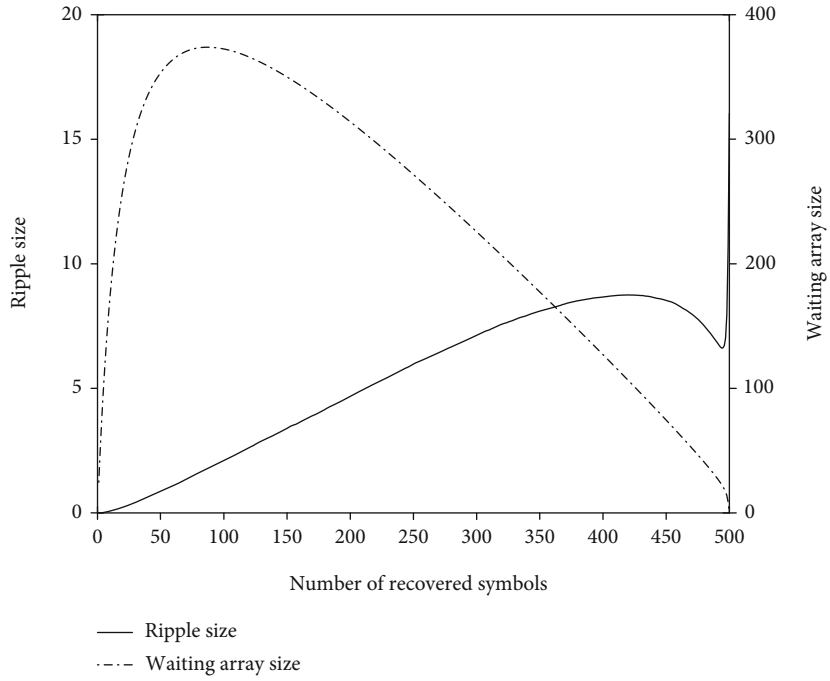

Figure 4: Ripple size and waiting array size as a function of number of recovered symbols for $k = 500$.

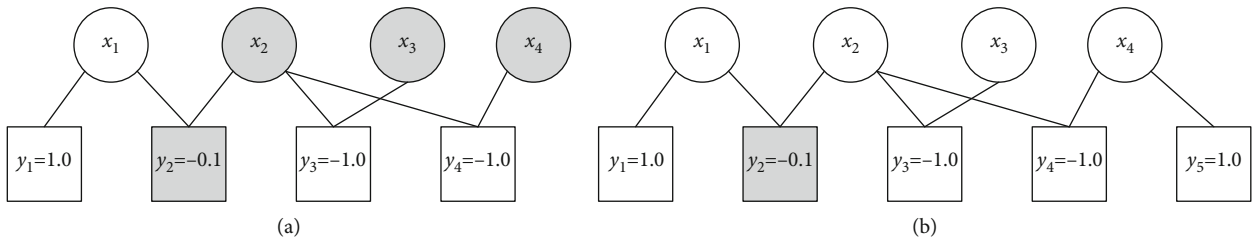

Figure 5: (a) BP decoder. (b) Delayed BP decoder.

demodulation is performed on encoded symbols. The log-likelihood ratio (LLR) of the encoded symbol is defined as

$$L\left(y_j\right) = \ln \frac{\Pr\left(s_j = 1 \middle| y_j\right)}{\Pr\left(s_j = 0 \middle| y_j\right)}. \quad (2)$$

In this study, HDBP decoding was concatenated with SDBP decoding, which can reduce decoding complexity. The encoded symbols with LLR were passed to HDBP decoding, and the output of decoding was verified by a CRC. Decoding is successful if it passes; otherwise, SDBP decoding is invoked.

2.2. BP Encoding. Given k input symbols $X = (x_1, x_2, \cdots, x_k)$ and a degree distribution$\Omega(d)$, $d = 1, 2, \cdots, k$, subsequently, an infinite number of encoded symbols are generated according to Algorithm 1.

2.3. Hard Decision BP Decoding. BP decoding is widely used for LT codes, which are implemented in different variants for different channels. HDBP decoding is used in BEC, whereas SDBP decoding is used in noisy channels. We discovered that HDBP decoding concatenated with SDBP decoding can be used in noisy channels, which will be analyzed herein. In HDBP decoding, encoded symbols participating in the decoding process are considered as correct symbols. Hence, simple inversed XOR operations are performed. The pseudocode of HDBP decoding is shown in Algorithm 2, where decoding is performed at once. Decoding is completed when sufficient encoded symbols are received.

3. Analysis of Hard Decision BP Decoding

For HDBP decoding, suppose n encoded symbols $y_1, y_2, \cdots, y_n$ are sufficient to recover k input symbols $x_1, x_2, \cdots, x_k$. The relationship between the input symbols and encoded symbols can be expressed by a Tanner graph. For example, the Tanner graph of four input symbols and five encoded symbols is shown in Figure 1.

3.1. Error Probability of Hard Decision BP Decoding. Let $\rho(y_j)$ denotes the error probability of the encoded symbol y_j.

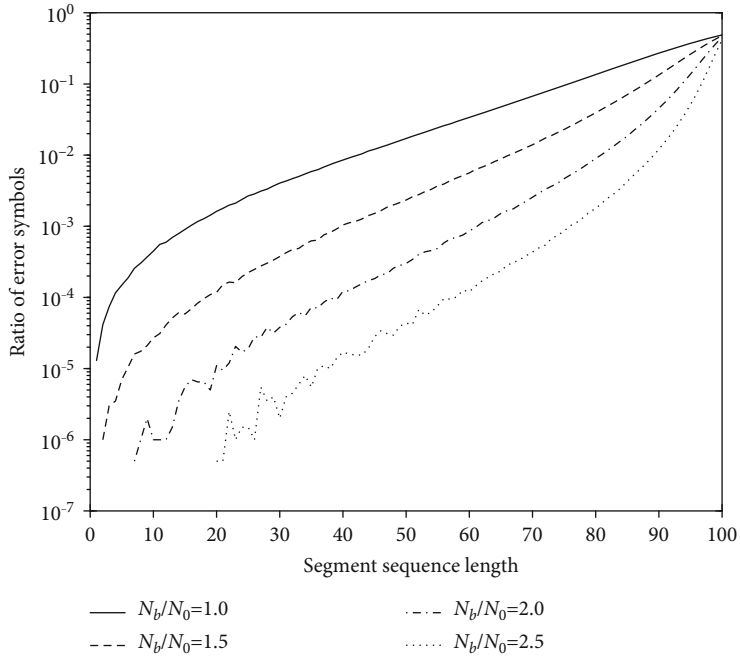

Figure 6: Ratio of error symbols as a function of segment sequence length.

```
Pseudocode of calculating threshold
Input: Probability δ
Output: Threshold t
initialize an ordered array S
initialize n=0
whilei<Mdo
    generate a packet p and add to S
    ifp is error then
        n=n+1
    end if
    i++9:  end while
e = n × δ
i=M
whilei ≥ 0do
    p=S[i]
    ifp is error then
        e=e − 1
    end if
    ife<= 0 do
        t = Abs(LLR(S[i]))
        break
    end if
    i−
end while
returnt
```

Algorithm 3.

Generally, $\rho(y_1) < \rho(y_2)$, if $|L(y_1)| > |L(y_2)|$ and vice versa. A decreasing function exists such that

$$\rho\left(y_j\right) = f\left(\left|L\left(y_j\right)\right|\right). \tag{3}$$

Let $\rho(x_i)$ denotes the error probability of the input symbol x_i if it is recovered by the encoded symbol y_j, which is shown in formula (4).

$$\rho(x_i) = \begin{cases} \rho\left(y_j\right) & d\left(y_j\right) = 1 \\ \max\left(\rho\left(y_j\right), \rho\left(N\left(y_j, x_i\right)\right)\right) & d\left(y_j\right) > 1 \end{cases}, \tag{4}$$

where $N(y_j, x_i)$ denotes the neighbors of y_j except x_i. In HDBP decoding, input symbols are recovered individually in sequence. In Figure 1, (x_1, x_2, x_3, x_4) is a reasonable sequence of input symbols recovered in decoding, and it is not the only one. For a sequence, we define $Q(x_i)$ as the set of encoded symbols that are the only neighbors of the input symbol x_i at the end of decoding. For the sequence (x_1, x_2, x_3, x_4), we have $Q(x_1) = \{y_1\}$,$Q(x_2) = \{y_2, y_3\}$, $Q(x_3) = \{y_4\}$, and $Q(x_4) = \{y_5\}$. We discovered that x_2 can be recovered by both y_2 and y_3. Let$Q'(x_i)$ denotes the set of encoded symbols supported to decode x_i. We have $Q'(x_2) = \{y_1, y_2\}$ if it is recovered by y_2; otherwise, $Q'(x_2) = \{y_1, y_4, y_3\}$. Therefore, we have

$$\rho(x_i) = \max\left(\rho\left(Q'(x_i)\right)\right). \tag{5}$$

Let $\mathbb{Q} = \{Q'(x_1), Q'(x_2), \cdots, Q'(x_k)\}$; the error probability of HDBP decoding is shown in formula (6).

$$\rho(\mathbb{Q}) = 1 - \prod_{i=1}^{k} 1 - \rho(x_i). \tag{6}$$

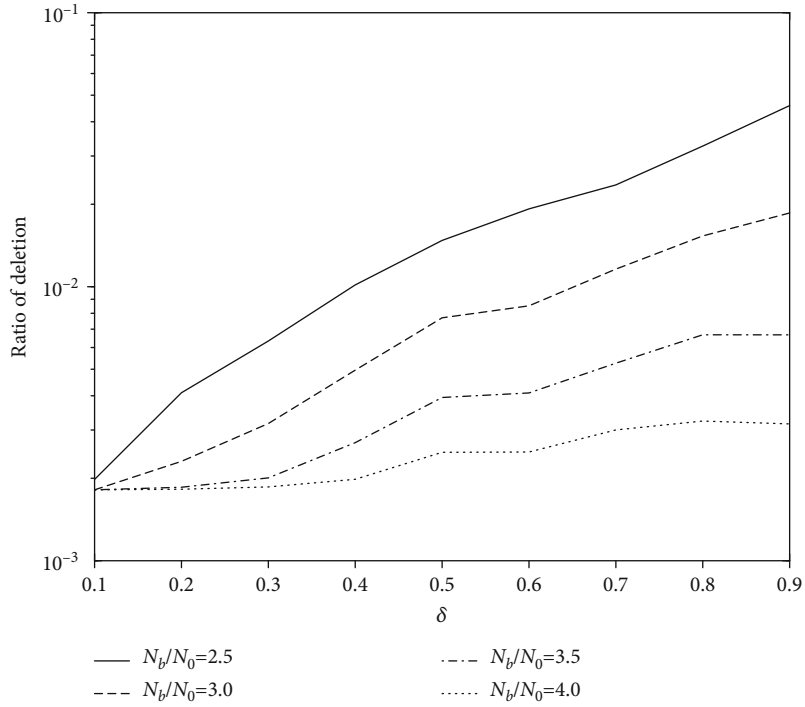

Figure 7: Ratio of deletion as a function of δ.

Table 1: Computational complexities of four decoders.

	BP	RSBP	RSDBP	RSTBP
XOR	$O(\bar{d}k)$	$O(\bar{d}k)$	$O(\bar{d}k)$	$O(\bar{d}k)$
SORT	–	$O(k \log R_{\max})$	$O(k \log R_{\max})$	$O(k \log R_{\max})$

Table 2: Numerical results of computational complexities with k=256.

	BP	RSBP	RSDBP	RSTBP
XOR	2156.27	2156.60	2157.47	2157.52
SORT	–	560.89	571.29	571.30

For a Tanner graph, several different $\mathbb{Q}$ exist. Our aim is to optimize the supported set of each input symbol to reduce the error probability of decoding. For example, x_2 should be recovered by a supported set with a lower error probability. For example, the Tanner graph with the LLR value is shown in Figure 2. The LLRs of y_2 and y_3 were set as -0.1 and 0.2, respectively. As shown in Figure 2(a), the input symbol is incorrect if it is recovered by y_2. Otherwise, it is correct if it is recovered by y_3, which is shown in Figure 2(b).

3.2. Improvement in Error Probability. In HDBP decoding, each input symbol is recovered by $1+\varepsilon$ encoded symbols on average. In other words, ε input symbols will be recovered by two encoded symbols. This is a valid assumption because the probability of an input symbol recovered by more than two encoded symbols is small. Consider the case in which both encoded symbols y_1 and y_2 have only the neighbor of the input symbol $x_i$at the end of decoding. The error probability of x_i is shown in formula (7) if it is recovered by y_1 or y_2 at random.

$$\rho(x_i) = \frac{1}{2}\sum_{j=1}^{2} \max\left(\rho\left(y_j\right), \rho\left(N\left(y_j, x_i\right)\right)\right). \tag{7}$$

Otherwise, the error probability of x_i is as shown in formula (8) if it is recovered by the encoded symbol with a lower error probability.

$$\rho(x_i) = \min\left(\max\left(\rho(y_1), \rho(N(y_1, x_i))\right), \max\left(\rho(y_2), \rho(N(y_2, x_i))\right)\right). \tag{8}$$

Therefore, the error probability of decoding is reduced if the encoded symbol with a lower error probability is selected to recover the corresponding input symbol.

4. Improved Hard Decision BP Decoders

As shown, the error probability of the input symbol can be reduced by selecting the support set with a lower error probability. In this section, we propose three improved HDBP decoders to reduce the probability of decoding.

4.1. Ripple-Sorted BP Decoder. The structure of the HDBP decoder is shown in Figure 3. First, degree-one encoded symbols are added to the ripple to start the decoding. The symbols in the ripple are processed individually until the ripple is empty. Two methods can be used to reduce the error probability of recovered symbols in the decoding process.

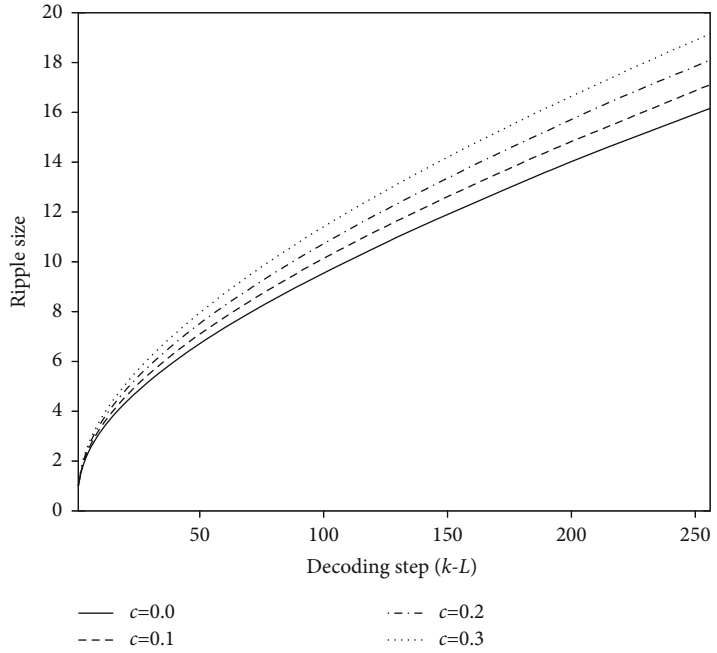

Figure 8: Ripple size as a function of decoding step for different c.

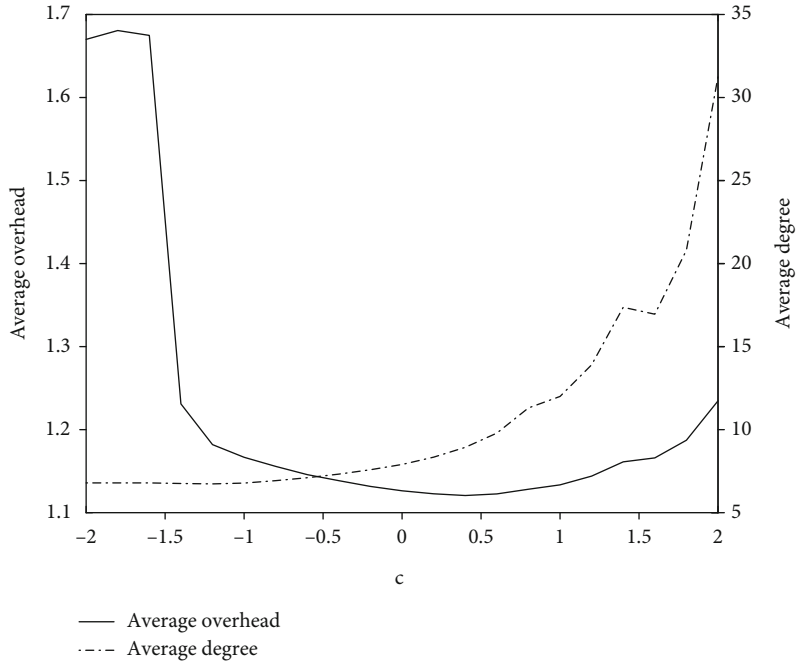

Figure 9: Average overhead and average degree as a function of c.

The first one is to sort symbols in the ripple. The second one is to sort symbols in the waiting array.

Lemma 1. *Sorting the symbols in the waiting array can be replaced by sorting the symbols in the ripple, and both the RSBP decoder and waiting-array-sorted BP (WSBP) decoder can reduce the error probability of decoding.*

Proof. The symbols released in each step depend only on the symbol being processed and the symbols in the waiting array; they are irrelevant to the order of the waiting array. The released symbols are sorted in the WSBP decoder, whereas the released symbols are sorted after being added to the ripple in the RSBP decoder. We assume that two symbols y_1 and y_2 are released simultaneously and $|L(y_1)| > |L(y_2)|$ without loss of generality. If the remaining neighbor of both y_1 and y_2 is x_1,

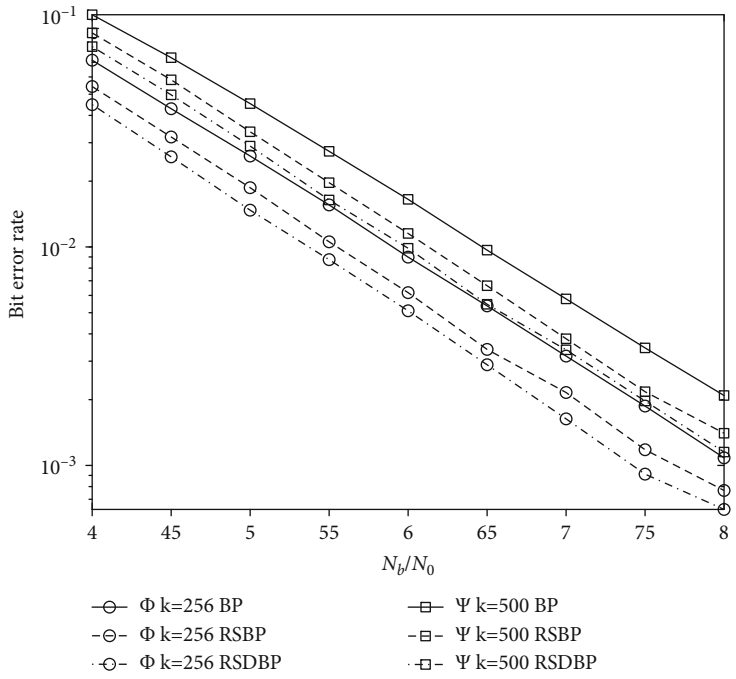

Figure 10: BER as a function of N_b/N_0.

Table 3: Computational time of three decoders.

k	BP (ms)	RSBP (ms)	RSDBP (ms)
256	0.766	0.768	0.772
500	3.069	3.081	3.088

Table 4: Optimal parameter values and average degree of the proposed degree distribution Ω.

k	$m_{\max}$	c	$\bar{d}$	ε
256	3	0.3	7.68	0.1614
500	4	0.4	9.08	0.1196

Table 5: Average degree of compared degree distributions.

	k	$\bar{d}$	ε
Φ	256	8.11	0.1636
Ψ	500	16.10	0.1381

the error probability of x_1 is reduced in both the RSBP and WSBP decoders. Hence, Lemma 1 is proven.

Lemma 2. *For HDBP decoding, sorting symbols in ripple is better than sorting symbols in the waiting array.*

Proof. We assume that two symbols y_1 and y_2 exist in the ripple and $|L(y_1)| > |L(y_2)|$. The remaining neighbors of y_1 and y_2 are x_1 and x_2, respectively. It is clear that the error probability of the symbol released in this step is equal to or greater than the error probability of y_1. Therefore, the symbol with minimal error probability should recover the corresponding input symbol in each decoding step. However, the error probability of the symbol released in the next steps may be less than the error probability of y_2. If y_3, $|L(y_3)| > |L(y_2)|$ is released and added to the ripple when y_1 is processed, the remaining neighbor of y_3 will be x_2. The input symbol x_2 should be recovered by y_3. In this case, the performance of the RSBP decoder is better than that of the WSBP decoder. Hence, Lemma 2 is proven.

he design of the ripple size evolution assumes that the ripple size should be remain more than one throughout the decoding process. Therefore, in theory, the performance of the RSBP decoder is better than that of the HDBP decoder. To analyze the performance improvement, the ripple size and waiting array size are analyzed by Monte Carlo simulations. The result is shown in Figure 4 with $k = 500$ and the degree distribution in [18], where the average ripple size and average waiting array size in each decoding step are calculated by 100000 simulations. The percentage of ripple sizes greater than one exceeds 80%, which means that symbols in the ripple can be sorted based on the absolute LLR value. As shown in Figure 4, the waiting array size is large at the beginning of decoding, which means that the probability of y_1 and y_2 released in the same decoding step is high. Additionally, we discovered that the number of symbols in the waiting array is larger than that in the ripple exception of $n_r \geq 499$. Therefore, sorting symbols in ripple is more efficient than sorting symbols in a waiting array. The proposed RSBP decoder can be implemented by replacing push (R, y) with pushAndSort (R, y) in Algorithm 2.

4.2. Ripple-Sorted and Delayed BP Decoder. For HDBP decoding, the number of symbols released in each step

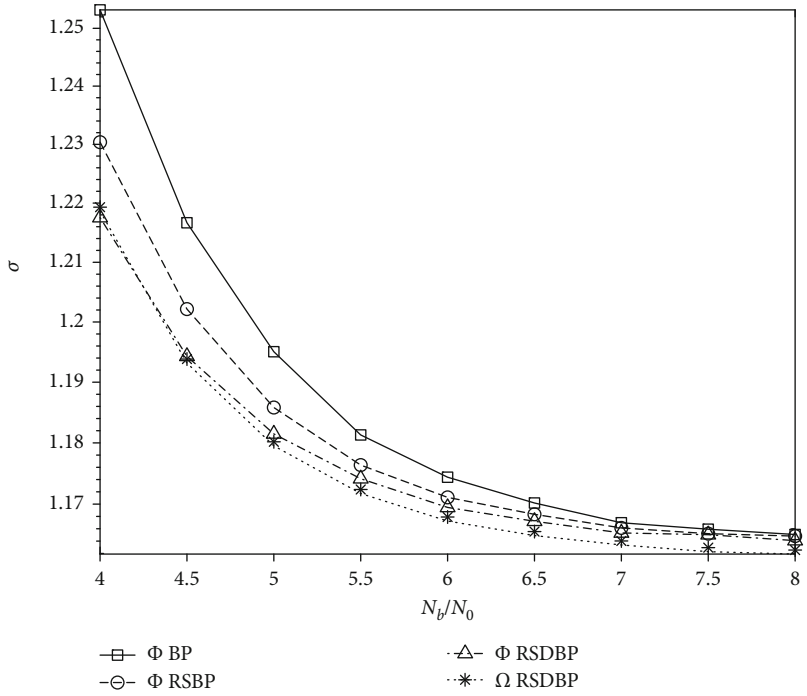

Figure 11: σ as a function of SNR for $k = 256$.

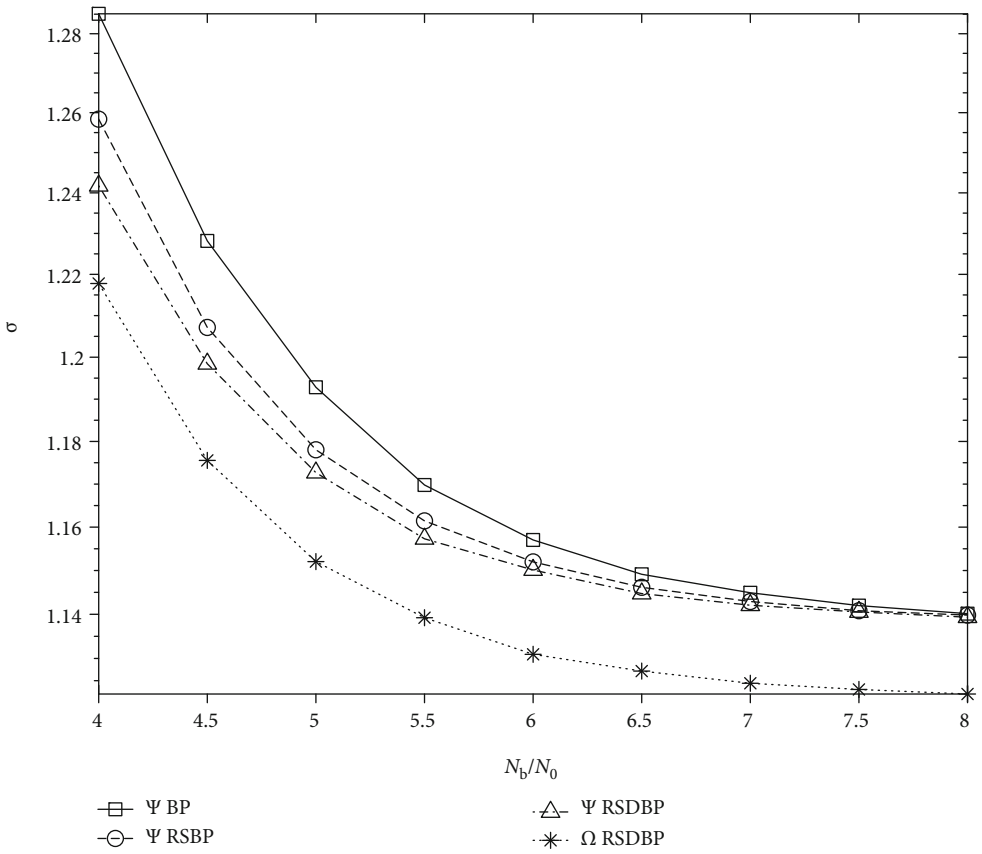

Figure 12: σ as a function of SNR for $k = 500$.

increased with the size of the waiting array. Therefore, the performance increased with the size of the waiting array. For example, as shown in Figure 5(a), the input symbol x_2 recovered by y_2 is incorrect because y_2 is incorrect. As a result of error propagation, x_3, x_4 are also incorrect. In Figure 5(b), the decoding process is delayed until sufficient encoded symbols are available. The input symbol x_4 recovered by y_5 is correct because y_5 is correct; therefore, x_3, x_4 are correct as well. Consequently, the encoded symbol y_2 with a high error probability is redundant.

Lemma 3. *The more encoded symbols are available before decoding starts, the better is the BER performance of decoding.*

Proof. We assume that the input symbol x can be recovered by one of y_1, y_2 with $|L(y_1)| < |L(y_2)|$. If y_1 is processed before y_2 is available, then the error probability of x is reduced if decoding is delayed until y_2 is available. If more encoded symbols are available before decoding starts, the error probability of more input symbols will be reduced. Hence, Lemma 3 is proven.

Based on Lemma 3, we propose our RSDBP decoder, which delays the start of the decoding until $k(1+\varepsilon)$ encoded symbols are received. The parameter ε depends on k and the degree distribution. For example, ε is set as 0.16 for $k = 256$ with a degree distribution in [20]. The proposed RSDBP decoder can be implemented by starting the RSBP decoding process until sufficient encoded symbols are added to the waiting array.

4.3. Ripple-Sorted and Threshold-Based BP Decoder. Let P denotes the ratio of error symbols, which increases as the SNR decreases. The BER performance of the RSDBP decoder decreased as P increased. To reduce the probability of incorrect encoded symbols participating in decoding, the encoded symbols with a high error probability should be deleted. The distribution of error symbols can be analyzed using Monte Carlo simulations. For example, $2k$ ($k = 500$) encoded symbols are generated and sorted by the error probability, denoted as $y_1, y_2, \cdots, y_{2k}, |L(y_1)| \geq |L(y_2)| \geq \cdots \geq |L(y_{2k})|$. We segmented $y_1, y_2, \cdots, y_{2k}$ into 100 segments. The ratio of error encoded symbols in each segment is shown in Figure 6. As shown, only a small number of error encoded symbols exist, and the ratio of error encoded symbols increased with the segment sequence. Therefore, most error encoded symbols can be deleted from decoding if the tails of the sorted encoded symbols are deleted.

For HDBP decoding, the received encoded symbol y_j will be deleted if $|L(y_j)| < t$, where t denotes the threshold. Otherwise, it will participate in decoding. Let δ denotes the probability that an error symbol will be deleted. For deletion probability δ, the threshold t can be calculated by Monte Carlo simulations, as shown in Algorithm 3.

Let ω denotes the ratio of encoded symbols deleted by decoding, which depends on δ. Figure 7 shows the ratio of deletion as a function of δ. As shown, the ratio of deletion decreased as the SNR increased, whereas it increased with δ.

Table 6: Optimal parameter values for proposed degree distribution.

k	δ	$m_{\max}$	c	$\bar{d}$
256	0.01	3	0.22	7.36
	0.90	3	0.24	7.40
500	0.01	4	0.25	8.62
	0.90	4	0.40	8.84

Therefore, the trade-off between the BER performance and overhead can be adjusted by δ.

Based on the analysis of symbol deletion, we propose a new decoder named the RSTBP decoder, in which encoded symbols with higher error probabilities are deleted from decoding. The proposed RSTBP decoder can be implemented by deleting encoded symbols that exceed the threshold.

5. Analysis of the Improved BP Decoder

Lemma 4. *The decoding complexity of the sorting ripple satisfies the constraint*

$$C \leq O((1+\varepsilon)klogR(k)). \tag{9}$$

Proof. The ripple size decreases as the decoding process proceeds. Initially, $R(k)$ symbols are released and sorted, and the decoding complexity is $O(R(k)\log R(k))$. The remaining $(1+\varepsilon)k - R(k)$ symbols will be inserted into the ripple, and the decoding complexity is less than $O(((1+\varepsilon)k - R(k))\log R(k))$. Hence, Lemma 4 is proven.

The computational complexities of the four decoders are shown in Table 1, where $\bar{d} = 1/k\sum_{d=1}^{k} d\Omega(d)$ and $R_{\max} = \max(R_k, R_{k-1}, \cdots, R_1)$, and the numerical results of computational complexities obtained by simulations are shown in Table 2. It is noteworthy that the number of XOR operations depends only on the average degree, and the number of SORT operations in RSDBP is the same as that in RSTBP. The number of SORT operations in RSBP is slightly less than that in RSDBP because small input symbols have been recovered before $(1+\varepsilon)k$ encoded symbols are available.

Lemma 5. *For a P and δ, the number of error encoded symbols participating in decoding is*

$$N_e = \frac{(1-\delta)P}{1-\delta P}(1+\varepsilon)k. \tag{10}$$

Proof. Let N denotes the number of encoded symbols received, and we have $N(1-\delta P) = (1+\varepsilon)k$. The error encoded symbols participating in decoding are $N_e = NP(1-\delta)$. Hence, Lemma 5 is proved.

Lemma 6. *The number of input symbols recovered by error encoded symbols directly satisfies the constraint*

$$N_e{}' \leq N_e - \frac{\varepsilon N_e{}^2}{2(1+\varepsilon)k}. \tag{11}$$

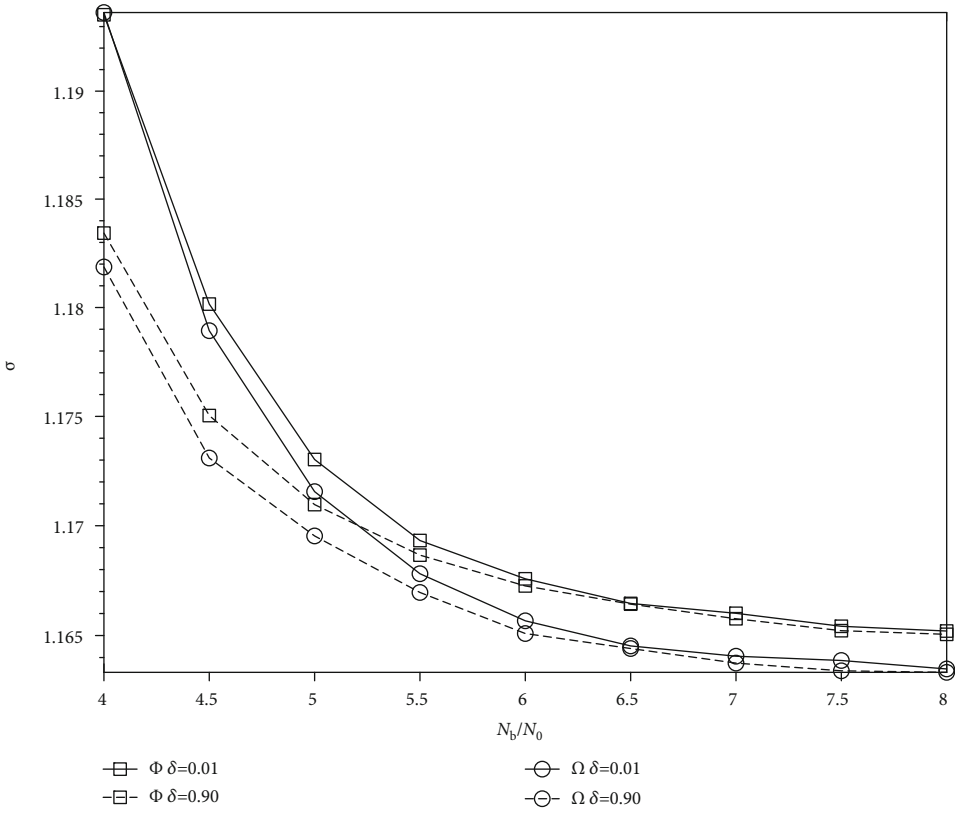

Figure 13: σ as a function of SNR for $k = 256$.

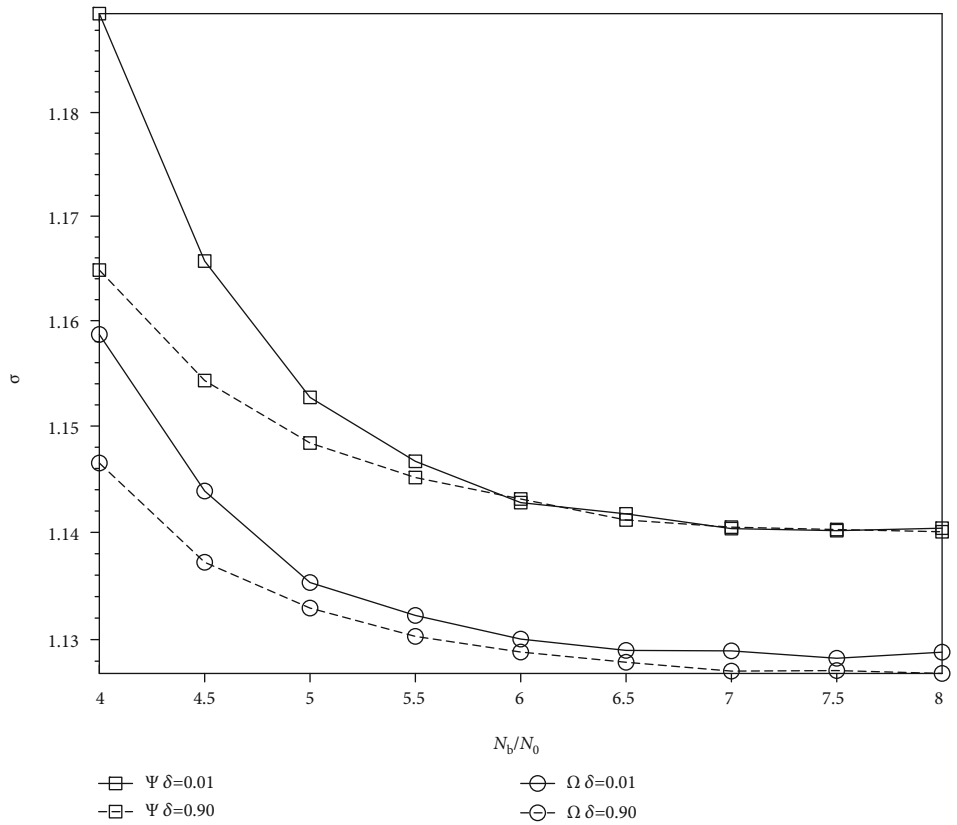

Figure 14: σ as a function of SNR for $k = 500$.

Proof. There exist ε pairs of encoded symbols. Since N_e is small compared with $(1+\varepsilon)k$, the probability of an error encoded symbol pairing with a correct encoded symbol is $\varepsilon N_e/(1+\varepsilon)k$. Let $(y_1, y_2), |y_1| \le |y_2|$ denotes a pair of encoded symbols without loss of generality. If one of y_1 and y_2 is an error encoded symbol, the probability that y_1 is the error encoded symbol exceeds 0.5. Therefore, more than $\varepsilon {N_e}^2/2(1+\varepsilon)k$ error encoded symbols will be considered as redundant symbols. Hence, Lemma 6 is proven.

Definition 7. (error propagation probability). the neighbors of the encoded symbol are selected randomly. Therefore, the probability that an encoded symbol with degree d is affected by an error input symbol that satisfies the constraint

$$\rho(d) = \frac{C_1^1 C_k^{d-1}}{C_k^d}. \tag{12}$$

We observed that the error propagation probability decreased with the average degree. For example, no error propagation was observed when the average degree was one. Hence, a trade-off occurred between the error propagation probability and overhead.

Definition 8. (number of affected encoded symbols). let $d_1, d_2, \cdots, d_L$ denote the degrees of L encoding symbols that will recover L input symbols. The number of encoded symbols affected by an error symbol in step L directly satisfies the constraint

$$N(L) = \sum_{i=1}^{L} \rho(d_i). \tag{13}$$

Lemma 9. (total number of affected encoded symbols). *let $l_1, l_2, \cdots, l_{N(L)}$ denotes the steps affected by the error symbol in steps k-L. The total number of encoded symbols affected by an error symbol satisfies the constraint*

$$\mathbb{N}(L) \approx \begin{cases} N(L) + \sum_{l=1}^{N(L)} \mathbb{N}(l) & L > 1 \\ N(L) & L = 1 \end{cases}. \tag{14}$$

Proof. Compared with k, the average degree of encoded symbols is small. Hence, $\rho(d)$ is relatively small. The number of encoded symbols that are affected by an error symbol directly and indirectly is small. Therefore, the double counting problem is disregarded; hence, the lemma is proven.

To validate the performance of LT codes in AWGN channels, we propose a new indicator known as NERRIC, which is defined as follows:

$$\sigma = (1+\varepsilon)/\tau, \tag{15}$$

where τ denotes the BER of decoding.

Table 7: Optimal parameters of the proposed degree distribution with fixed overhead.

k	ε	$m_{\max}$	c
256	0.20	3	-0.4
	0.25	3	0.4
500	0.15	6	0.4
	0.20	6	0.5

6. The Optimal Degree Distribution for a Specific BP Decoder

Studies regarding the design of an optimal degree distribution for a specific BP decoder over AWGN channels are limited, as previously discussed. Herein, a method for designing a degree distribution for a specific goal is proposed. The ripple size evolution is important for the design of a degree distribution. Random walk was used to model the number of encoded symbols released in each step. We assumed that the number of encoded symbols released in each step was a Poisson distribution.

Lemma 10. (symbol release). *let $\varphi(m)$ denotes the probability that m encoded symbols will be released in each step. It satisfies the constraint*

$$\varphi(m) = e^{-1}/m. \tag{16}$$

Proof. The number of encoded symbols released is a Poisson distribution. The expectation of this distribution is one. Therefore, Lemma 10 is proven.

Let $m_{\max}$ denotes the maximum number of encoded symbols released in a single decoding step. For a fixed $m_{\max}$, Monte Carlo simulations can be used to generate plenty of ripple size evolutions. Each ripple addition is modeled as a random walk with a probability distribution $\varphi(m)$. The ripple size evolution is modeled as follows:

$$R_L = R_L + c\sigma_L, \tag{17}$$

where $\bar{R}_L$ and σ_L denote the average ripple size and variance of the simulation results in decoding step L, respectively; c denotes a parameter to adjust the ripple size evolution. For $m_{\max} = 3$, the ripple size as a function of decoding step for different c is shown in Figure 8. It is clear that the expected ripple size evolution can be generated by carefully adjusting parameter c. Given the ripple size evolution, the degree distribution can be calculated using the GDDA. The degree distribution is obtained based on formula (18).

$$\Omega = GDDA(RSE(m_{\max}, c)), \tag{18}$$

where $RSE(m_{\max}, c)$ denotes the ripple size evaluation determined by parameters $m_{\max}, c$.

Let Ω denotes the degree distribution designed to minimize average overhead. Let Ω' denotes another well-

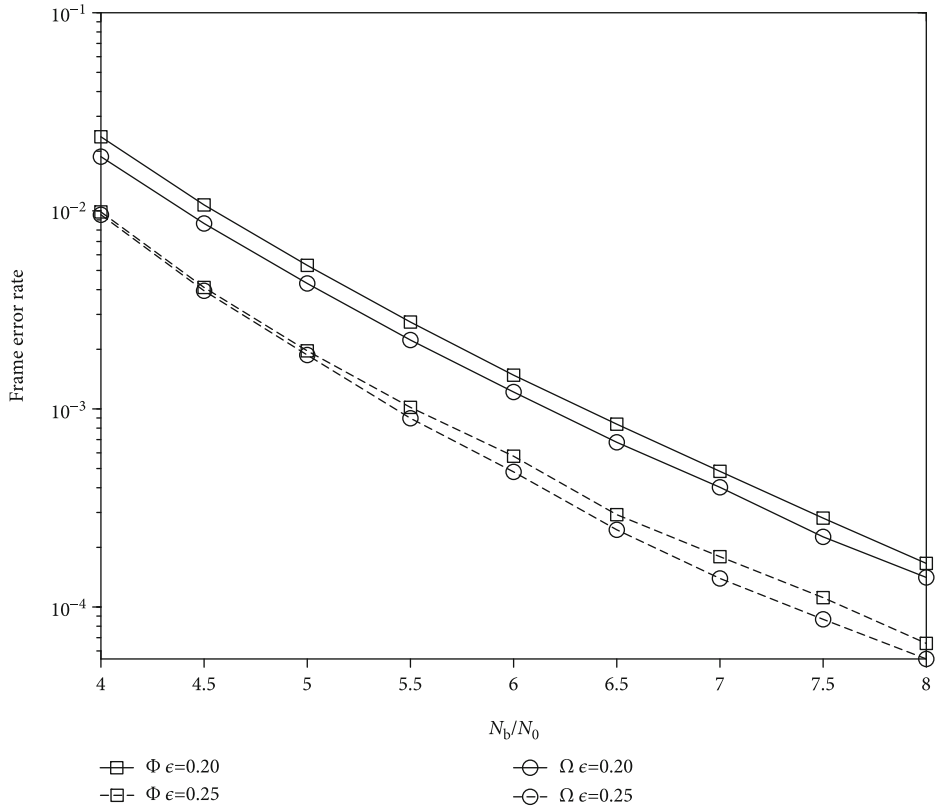

Figure 15: Frame error rate as a function of SNR for $k = 256$.

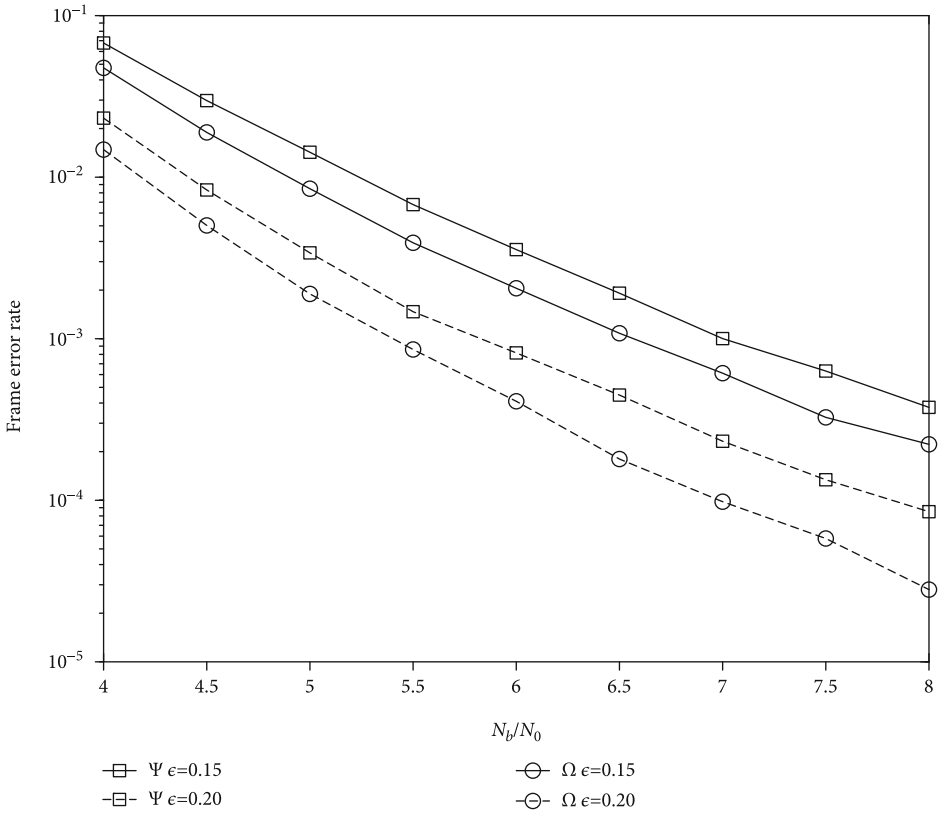

Figure 16: Frame error rate as a function of SNR for $k = 500$.

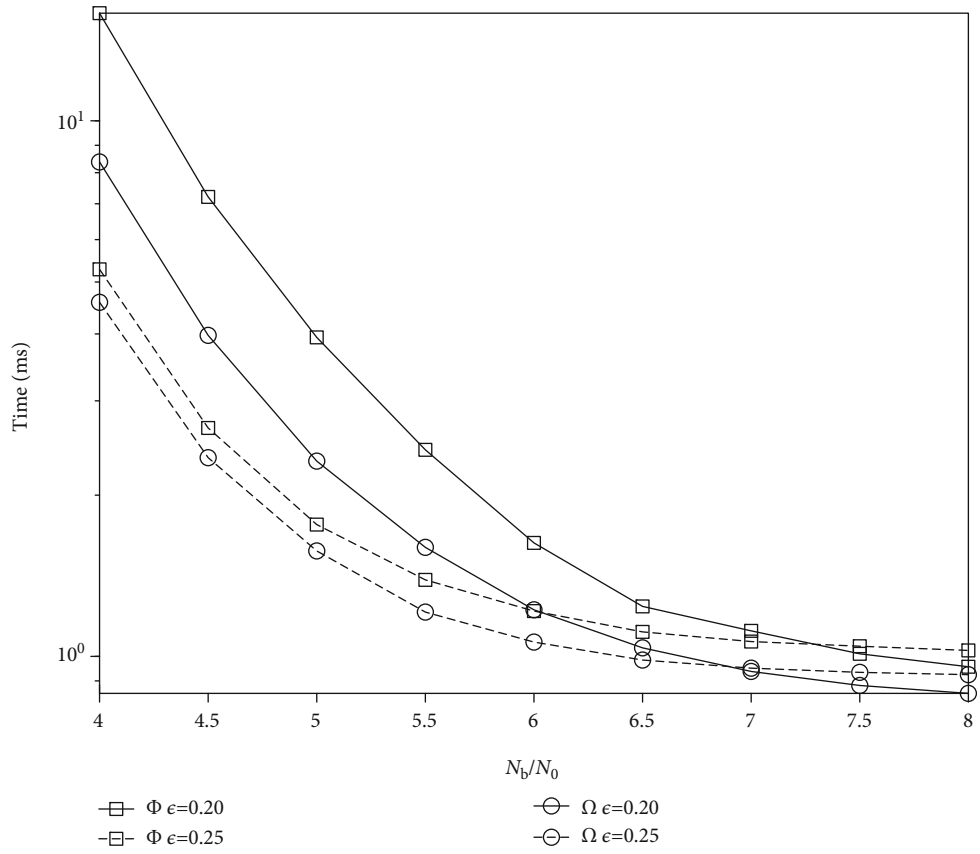

Figure 17: Decoding time as a function of SNR for $k = 256$.

designed degree distribution to decrease the average degree at the expense of increasing the average overhead. The average overhead and average degree as a function of parameter c are shown in Figure 9. The BER decreased as the average degree decreased because of two reasons. First, the more encoded symbols participated in decoding, the more encoded symbols recovered the same input symbol, resulting in a decrease error probability of decoding. Subsequently, the error propagation decreased with the average degree. Therefore, the BER is in conflict with the average overhead. Additionally, the average degree directly determines the number of operations during the encoding and decoding processes.

Let $G(\Omega)$ denotes the objective function of the degree distribution Ω. Additionally, the optimal parameters $(m_{\max}, c)$ can be converted to a pure optimization problem as follows:

$$m_{\max}, c = \arg \min_{m_{\max}, c} G(GDDA(RSE(m_{\max}, c))) \quad (19)$$

The variable c is used for the range [-1,1] and $m_{\max}$ for the range $[3, \sqrt{k}]$. Generally, for a fixed $m_{\max}$, it might appear that a lower value of c would be desirable for decreasing both the average degree and BER at the expense of increasing the average overhead.

7. Numerical Results

In this section, some simulation results are provided to validate our study. The decoding algorithms were implemented in C++ and executed on a computer with a Xeon E3-1505 M CPU and 16 GB of RAM under Windows10. The degree distributions proposed in [18, 20] were used in our simulations, which are denoted as Φ and Ψ, respectively, and our proposed degree distribution is denoted as Ω. The BER as a function of N_b/N_0 is shown in Figure 10. The BERs of the RSBP and RSDBP decoders were better than that of the BP decoder, consistent with our analyses. For example, with $k = 500$ and $N_b/N_0 = 4.0$, the BER of the BP decoder was 0.115, whereas the BER of the RSDBP decoder was 0.082. The computational times of BP, RSBP, and RSDBP are shown in Table 3. As shown, the computational times of the three decoders were similar.

The RSDBP decoder combined with the proposed degree distribution Ω was compared with the other decoders. The degree distribution Ω was designed to optimize σ by selecting the appropriate $m_{\max}$ and c; the optimal parameters and average degree of Ω are shown in Table 4, whereas the average degrees of Φ and Ψ are shown in Table 5. As shown, the average degree of Ω is smaller than those of the others.

Figures 11 and 12 illustrate the NERRIC σ achieved by different decoders and different degree distributions with $k = 256$ and $k = 500$, respectively. It is clear that the RSBP

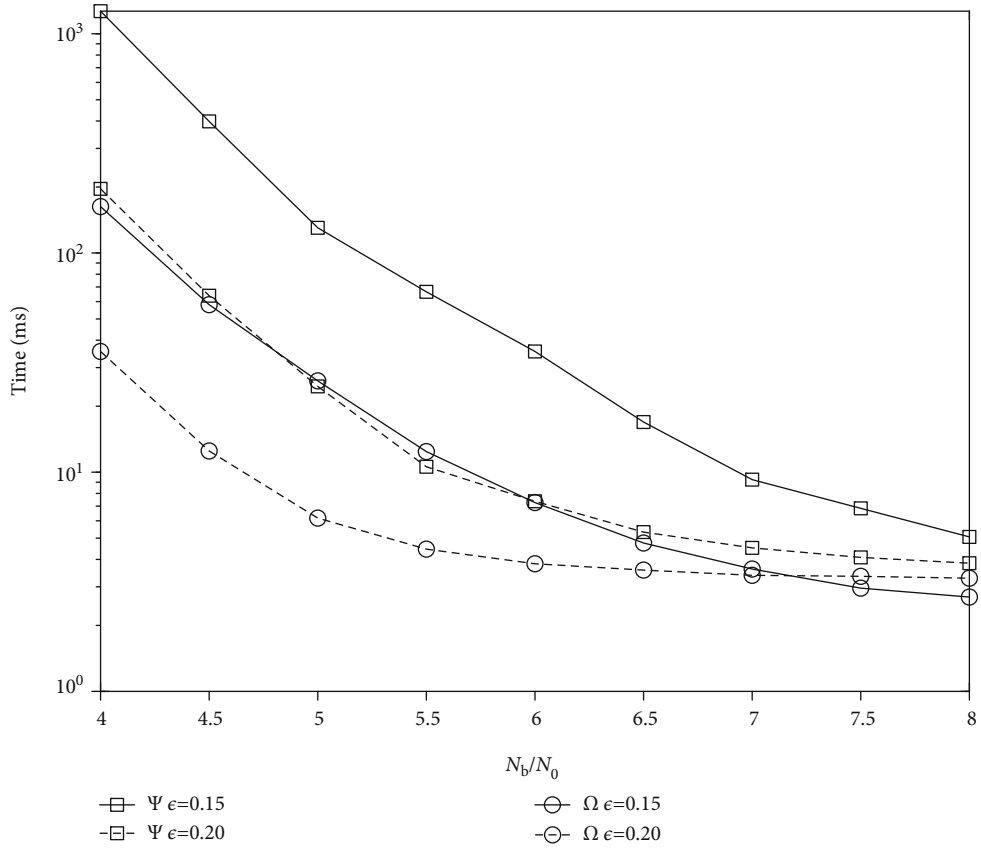

Figure 18: Decoding time as a function of SNR for $k = 500$.

and RSDBP decoders outperformed the BP decoder, which is consistent with the theoretical analysis. The improvement decreased as the SNR increased because barely any error encoded symbols were discovered in channels with higher SNRs. Furthermore, RSDBP combined with the proposed degree distribution outperformed the other methods, and the improvement increased with the SNR. For example, with $k = 500$ and $N_b/N_0 = 4.0$, the σ of the RSDBP decoder combined with Ψ was 1.241, whereas the σ of RSDBP combined with the proposed degree distribution was 1.217. This is because the optimization goal was to minimize σ, and the probability of error propagation decreased with the average degree.

The RSTBP decoder combined with the degree distribution Ω was compared with the other decoders, and the optimal parameters of Ω with different $\delta$and k are listed in Table 6.

Figures 13 and 14 show σ as a function of SNR for $k = 256$ and $k = 500$, respectively. As shown from the figures, the proposed degree distribution Ω yielded better results than the others for both $\delta = 0.01$ and $\delta = 0.90$to minimize σ for optimization. As the SNR increased, the performance of Ω was better than that of the others. This is because the average degree of Ω was smaller. Furthermore, as δincreased, σ decreased more slowly. This is because the number of error encoded symbols decreased as the SNR increased, and the number of encoded symbols deleted at $\delta = 0.01$approached that at $\delta = 0.90$.

In hybrid decoding, the decoding complexity decreased as the FER increased. For RSDBP decoder, the degree distributionΩcan be tuned to achieve a lower FER in a fixed overhead. The optimal parameters of the degree distribution Ω with different ε and k values are shown in Table 7.

Figures 15 and 16 show the FER as a function of the SNR with $k = 256$ and $k = 500$, respectively. It was observed that the proposed optimal degree distribution outperformed the others for different fixed overheads. For instance, in the case of $k = 500$, $\varepsilon = 0.2$, and $N_b/N_0 = 4.0$, the FERs of Ψ and Ω were 0.0232 and 0.0138, respectively. This is because a better trade-off between the average overhead and average degree was achieved to reduce the effect of error propagation.

A hybrid decoder can be formulated by combining the RSDBP and SDBP decoders. Figures 17 and 18 show the decoding time as a function of the SNR for $k = 256$ and $k = 500$, respectively. It was observed that Ω outperformed the others in terms of the decoding complexity, as Ω was better than the others in the HDBP decoding stage.

8. Conclusions

Herein, we first analyzed the improvement of BP decoding by introducing a sorting ripple, delaying the decoding process, and deleting low-reliable symbols. Subsequently, we proposed three improved HDBP decoders, namely, RSBP, RSDBP, and RSTBP decoders. We demonstrated that both RSBP and RSDBP outperformed BP decoding in terms of

NERRIC although the decoding complexity increased slightly. Compared with the RSDBP decoder, the RSTBP decoder further increased the NERRIC but the average overhead increased. Furthermore, a ripple size evolution-based design of the optimal degree distribution was proposed. Numerical simulations demonstrated that the proposed degree distribution outperformed the others in terms of both the NERRIC and FER. The proposed scheme was not limited to AWGN channels and LT codes. It can be readily extended to noisy channels and Raptor codes. In future work, the energy consumption of LT codes will be investigated to identify a balance among the FER, average overhead, and average degree.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Beijing Natural Science Foundation (4194073).

References

[1] M. Luby, "LT codes," in *Proc. IEEE Symp. Found. Comp. Sci*, pp. 271–280, Vancouver, Canada, 2002.

[2] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, "Reliable multimedia download delivery in cellular broadcast networks," *IEEE Trans. Broadcast*, vol. 53, no. 1, pp. 235–246, 2007.

[3] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings*, p. 37, Chicago, USA, 2004.

[4] H. Jenkač, T. Mayer, T. Stockhammer, and W. Xu, "Soft decoding of LT-codes for wireless broadcast," in *Proc. IST Mobile Summit 2005*, 2005.

[5] A. Kharel and L. Cao, "Decoding of short LT codes over BIAWGN channels with Gauss-Jordan elimination-assisted belief propagation method," in *2015 Wireless Telecommunications Symposium*, pp. 1–6, 2015.

[6] L. He, J. Lei, and Y. Huang, "A greedy spreading serial decoding of LT Codes," *IEEE Access*, vol. 7, pp. 31186–31196, 2019.

[7] C. Cao, H. Li, and Z. Hu, "A new cross-level decoding scheme for LT Codes," *IEEE Communications Letters*, vol. 19, no. 6, pp. 893–896, 2015.

[8] R. Sun, M. Zhao, J. Liu, J. Sun, H. Guan, and Z. Zhao, "Piggybacking belief propagation decoding for rateless codes based on RA structure," in *2018 IEEE/CIC international conference on communications in China (ICCC)*, pp. 237–241, IEEE, 2018.

[9] M. Zhang, S. Chan, and S. Kim, "Soft iterative decoding algorithms for rateless codes in satellite systems," *Algorithms*, vol. 12, no. 8, p. 151, 2019.

[10] L. Weijia and C. Shengnan, "Performance analysis of complexity reduction BP decoding of rateless codes by deleting low reliable symbols," *The Journal of China Universities of Posts and Telecommunications*, vol. 23, no. 5, pp. 26–31, 2016.

[11] C. Albayrak and K. Turk, "Reduced-complexity decoding of LT Codes," *Wireless Personal Communications*, vol. 94, no. 3, pp. 969–975, 2017.

[12] K. Turk and P. Fan, "Adaptive demodulation using rateless codes based on maximum a posteriori Probability," *IEEE Communications Letters*, vol. 16, no. 8, pp. 1284–1287, 2012.

[13] I. Hussain, M. Xiao, and L. K. Rasmussen, "Reduced-complexity decoding of LT codes over noisy channels," in *2013 IEEE wireless communications and networking conference (WCNC)*pp. 3856–3860, IEEE, 2013.

[14] D. Park and S. Y. Chung, "Performance—complexity tradeoffs of rateless codes," in *2008 IEEE international symposium on information theory*, pp. 2056–2060, IEEE, 2008.

[15] A. Kharel and L. Cao, "Improved fountain codes for BI-AWGN channels," in *2017 IEEE wireless communications and networking conference (WCNC)*, pp. 1–6, IEEE, 2017.

[16] L. Wang and W. Tang, "Performance analysis and improvement of LT codes over AWGN channels," *Journal of Computers*, vol. 9, no. 4, pp. 974–982, 2014.

[17] D. Deng, D. Xu, and S. Xu, "Optimisation design of systematic LT codes over AWGN multiple access channel," *IET Communications*, vol. 12, no. 11, pp. 1351–1358, 2018.

[18] J. H. Sorensen, T. Koike-Akino, P. Orlik, J. Ostergaard, and P. Popovski, "Ripple design of LT codes for BIAWGN Channels," *IEEE Transactions on Communications*, vol. 62, no. 2, pp. 434–441, 2014.

[19] L. Zhang, T. Li, J. Liao, Q. Qi, and J. Wang, "Design of improved Luby transform codes with decreasing ripple size and feedback," *IET Communications*, vol. 8, no. 8, pp. 1409–1416, 2014.

[20] S. Xu and D. Xu, "Design of degree distributions for finite length LT codes," *Wireless Personal Communications*, vol. 98, no. 2, pp. 2251–2260, 2018.