IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 31, 2019, accepted February 19, 2019, date of publication March 7, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2903523

A Greedy Spreading Serial Decoding of LT Codes

LIANG HE, JING LEI, AND YING HUANG
Department of Communication Engineering, College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China

Corresponding author: Jing Lei (leijing@nudt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702536.

ABSTRACT In this paper, we propose new serial decoding of Luby transform (LT) codes over additive white Gaussian noise channels. LT encoder generates a potentially limitless number of encoded packets, and the decoder incrementally collects the packets to ensure successful recovery of the information. In the proposed algorithm, the newly coming code nodes are the first to pass the messages, and it is their neighboring source nodes that will receive these input messages and update their output messages; then, the next neighboring code nodes that have not been covered are to be included in the updating group; in this greedy way, the message propagation is conducted from neighbors to neighbors. The analysis demonstrates that the proposed algorithm has a faster convergence speed than the conventional ones, and simulation shows that it has an effective bit error rate performance.

INDEX TERMS AWGN channel, fountain codes, LT codes, soft decoding.

I. INTRODUCTION

To guarantee reliable data transmissions over noisy channels, forward error correction (FEC), or channel code, is a promising way for receivers to restore the information. Along with FEC, automatic repeat-request (ARQ) is applied when receivers can not recover the data correctly. With regard to a varied noisy channel, it is challenging to set the rate of the code if one utilizes channel code with a constant rate. That is, when a channel is varying from time to time, a low rate code promises a good transmission if the channel is in bad condition but it may cause a waste of channel use when it is in good condition, and vice versa if one chooses a code with a relatively high rate. It is also difficult to ensure a good quality of service when it comes to multicast and broadcast communication with such coding schemes. In point to multiple environment, receivers send requests of retransmission when their errors can not be corrected, and too many demands can lead to a feedback implosion. Even if the transmitter is able to handle these requests, the retransmission can cause delays among other receivers. Fountain codes [1]–[5] can be a solution. With such codes, one can produce a potentially limitless number of code symbols, which is why it is also known as rateless codes. Such codes can be applied in massive MIMO system [6] and relay channels [7]. And they have been widely used in practice, such as 3GPP Multimedia Broadcast Multicast Services (MBMS) and and Digital Video Broadcasting (DVB) standards.

The associate editor coordinating the review of this manuscript and approving it for publication was Zesong Fei.

Fountain codes are originally invented for data distribution on the Internet which can be regarded as erasure channels. Packets transmitted on such channels are treated as corrupted or error free at the receivers' end. A set of k information symbols are encoded into an arbitrary number of packets, each receiver picks up them and join them into the decoding procedure if they are correct, and the decoding will succeed when enough symbols are received. It is like collecting drops of water from a fountain into a cup until it is filled, where the name "fountain" comes from. For the transmitter, the encoding is ended by setting a maximum number of packets or indications from each receiver that it has successful recovered the information. Therefore, the feedback implosion that conventional channel codes and ARQ techniques incur can be avoided by fountain codes. Moreover, it is natural that this "rateless" property makes fountain codes adapted to systems where the channel state information (CSI) is not available.

LT codes are the first practical realization of fountain codes introduced by Luby [8]. The encoded packets are linear combination of information symbols on binary field. In the following sections, we refer to the packets as code symbols or code nodes, and information symbols as source symbols or source nodes, alternatively. Shokrollahi [9] developed LT codes by introducing precoding, which is called "Raptor Codes". Raptor codes are also a class of rateless codes, its performance depend on the inner LT codes to a large extent, and some decoding algorithm of Raptor codes can be drawn to apply to that of the latter.

LT codes were originally designed over binary erasure channels (BEC) with two decoding methods, namely, belief propagation (BP) and Gaussian elimination (GE). BP is a fast algorithm with complexity of $O(klog(k))$, and its performance of successful decoding is exploited in [10]. GE is a maximum likelihood decoding with complexity of $O(k^2)$, which was further developed by Kim *et al.* [11], [12] and Bioglio *et al.* [13]. Lazaro *et al.* [14] analyzed the distribution of inactivation in GE, and redesigned the output degree distribution with simulated annealing. On noisy channels, there are coding schemes combine LT codes on erasure level with other FEC codes on noise level, and the decoding is conducted on both levels. Hybrid hard and soft decoding algorithms under such scheme was proposed in [15] and [16], later Chen *et al.* [17] developed a new cross-level decoding that will not bring computation in real field to hard decoding. Essentially, their decoding of LT codes is still hard BP algorithm.

In AWGN channels or fading channels, LT codes are usually concatenated with a linear code forming so-called Raptor codes. Decoding of single LT codes over these noisy channels can get intuition of that of the latter ones. In [18] and [19], Log-BP soft decoding of LT codes were reviewed, which stemmed from the decoding of Gallager's LDPC [20]. Decoding is carried out iteratively, and if it fails, additional packets are needed in order to redo the process. Such algorithm keeps collecting residual output symbols and redoes the decoding round by round, which seems to regard the rateless codes as ones with fixed rate but in an incremental way. Numerous attempts of decoding are needed to restore the source symbols, and plus the original Log-BP algorithm itself, the decoding of LT codes on noisy channels have pretty high complexity.

Fortunately, since the decoding is incrementally deployed, the message left from the last iteration can be used for the next attempt. Hu *et al.* [21] proposed a parallel storage (PS) BP algorithm, which initialized the current decoder with the results obtained by the previous iterations. PSBP significantly advances the decoding process and decreases the complexity comparing to the conventional one. But its remaining decoding details stays the same as the original one. Wu *et al.* [22] noticed that the structure was always varying with the joining of the newly incoming packets, which gave inspirations to elaborately devise the message passing rules. Wu *et al.* [22] developed a serial storage (SS) BP decoding, which collects packets in groups and serially propagates messages from the latest groups to the earliest ones. SSBP makes the message convergence much faster, however, it leaves a huge delay of decoding due to the grouping techniques.

This paper proposes a greedy serial BP (GSBP) decoding of LT codes over AWGN channels. The receiver starts decoding when a required number of packets have been collected. When it fails, the decoder restarts a new attempt with a set of newly incoming packets, and the previous results are set as the initialization. Then the messages pass from the newly received ones to their neighbors, after which the updated messages are transmitted to the next neighbors. Thus, the messages are propagated from neighbors to neighbors, the latest information is spread greedily at a fast speed.

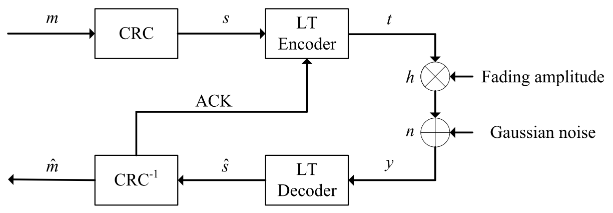

FIGURE 1. System model with LT coding over fading channel.

The paper discusses how fast the message propagation will cover the whole decoding graph. By analyzing the degree distribution of merged nodes, it is found that it only takes several rounds to go through the graph. Also, the speed of decoding convergence is studies, with the Gaussian approximation, it is observed that the proposed scheme has faster convergence speed over the parallel ones. Regarding the decoding complexity, since each node transmits messages only once during each iteration, both the proposed algorithm and the parallel ones have the same level of decoding complexity in one iteration.

This paper is organized as follows. Section II reviews the encoding and conventional Log-BP decoding of LT codes. Then the proposed greedily spreading serial decoding algorithm is detailed in Section III. In section IV, asymptotic analysis of the decoding, including the speed of greedy spread and convergence, and the decoding complexity, is demonstrated. After that, simulation results of the BER are presented in section V. Finally, we draw conclusions in section VI.

II. REVIEW OF LT CODES

A. SYSTEM MODEL

We have adopted a similar system in [22]. It is a simple point-to-point communication model, which can be expressed by

$$y_i = h_i \cdot t_i + n_i, \tag{1}$$

where y_i, t_i, and n_i represent received symbol, transmitted symbol, and added white Gaussian noise, respectively, h_i signifies the channel gain, and i stands for the time slot. Channel state information is available at the receiver but not at the encoder. Castura and Mao [23] have shown that the distribution of h_i can be neglected in such system. Thus, this paper simplifies this model into AWGN channel with parameter SNR.

In each decoding attempt, the decoder employs an iterative method to restore the information. At the end of each iteration, hard decision is made to judge whether the results are coincided with the information bits. A cyclic redundancy check (CRC) should be appended to justify the correctness of decoding results. It is assumed that the CRC is embedded into the information sequence. Once the decoding succeeds, the receiver sends acknowledgment (ACK) to inform the sender to transmit the next information sequence, and the current decoding is terminated. Otherwise, when the source symbols are not correctly recovered within maximum iterations, the receivers needs more packets to start the next decoding attempt.

B. ENCODING

An LT encoder generates a sequence of information symbols into a potentially infinite number of encoded packets. Here one symbol or packet indicate as many as binary bits, and without loss of generality, we refer to a single symbol or packet as one bit in this paper. To obtain packets, an output degree distribution should be designed, where the term degree indicates the number of neighbors of a symbol. The distribution can be concluded by the polynomial

$$\Omega(x) = \sum_{i=1}^{Dc_{max}} \Omega_i \cdot x^i, \tag{2}$$

where Ω_i denotes the probability that a degree i should be selected, and Dc_{max} is the maximum degree. Now we encode k information bits $s = [s_1, s_2, \ldots, s_k]$ into a stream of LT code symbols $e = [e_1, e_2, \ldots]$. In order to generate e_i, its degree d is chosen randomly based on the designed distribution $\Omega(x)$, and its d neighbors are usually selected uniformly random from the k source symbols. Thus, e_i is obtained by xoring its neighbors such that $e_i = s_{p_1} \oplus s_{p_2} \oplus \cdots \oplus s_{p_d}$, where "$\oplus$" indicates exclusive OR. Accordingly, the equation $e_i = g_i \cdot s$ denotes the connection between one encoded bits and the information bits, in which $g_i = [g_{i1}, g_{i2}, \ldots, g_{ik}]$, i.e., $g_{ij} = 1$ indicates that s_j participates the xor operation to generate e_i, otherwise not.

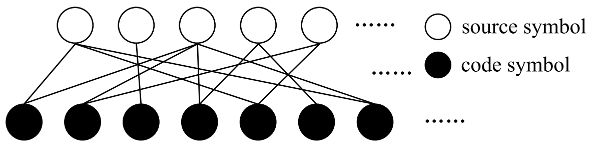

FIGURE 2. Encoding of LT codes.

Fig. 2 gives a simple illustration of LT encoding process. The transparent circles in the bipartite graph indicate the source/information symbols, the ones filled in black are encoded packets, and their links have clearly shown the generation constructions. The connections can be expressed by generation G matrix as well, which satisfies

$$G \cdot s^T = e^T. \tag{3}$$

It should be noted that the multiplication and addition are both conducted in binary field. Since LT encoder generates limitless packets, G has a potentially limitless number of rows, while the number of its column keeps the same of the information dimension.

C. DECODING

Decoding algorithm of LT codes over AWGN channel draws inspiration from that of LDPC codes [20], since they are both graph based. The message pass decoding of an LDPC code runs in terms of its $(N-K) \times N$ check matrix H. Variable nodes/symbols represent the code symbols, corresponding to the columns of H, and check nodes/symbols are imagined in order for decoding, corresponding to the rows of H. Log-BP algorithm is the conventional decoding method of LDPC codes, in which decoding messages, log-likelihood ratio (LLR), are propagated among variable nodes and check nodes. LT code is treated as a fixed-rate code temporarily when a decoding attempt is conducted, during which the code construction is also fixed. Once the current attempt is failed, more packets are needed to enter the construction joining the next decoding attempt. Thus, we say the bipartite graph of LT codes is fixed during each decoding attempt, but it varies incrementally during the whole decoding process.

We use BPSK mapping code bits 1s and 0s into transmitted bits -1s and 1s, respectively. We take advantage of log-likelihood ratio to indicate the probability that a binary random variable takes value from $\{0, 1\}$ or $\{1, -1\}$, that is

$$L(t_i) = \ln \frac{\Pr(t_i = +1)}{\Pr(t_i = -1)} = \ln \frac{\Pr(e_i = 0)}{\Pr(e_i = 1)}, \tag{4}$$

where Pr denotes the probability. Over AWGN channels, received bit y_i is the sum of transmitted bit t_i and added white Gaussian noise n_i with variance σ^2 and mean value 0, namely, $y_i = t_i + n_i$. It is assumed that the code bit takes value on $\{0, 1\}$ with equal probability. By Bayes' theorem, the observed LLR from AWGN channel of received symbol y_i can be derived from the following equations. We have

$$\begin{aligned} L(y_i) &= \ln \frac{\Pr(t_i = +1|y_i)}{\Pr(t_i = -1|y_i)} \\ &= \ln \frac{\Pr(y_i|t_i = +1) \cdot \Pr(t_i = +1)/\Pr(y_i)}{\Pr(y_i|t_i = -1) \cdot \Pr(t_i = -1)/\Pr(y_i)} \\ &= \ln \frac{\Pr(y_i|t_i = +1)}{\Pr(y_i|t_i = -1)}, \end{aligned} \tag{5}$$

with

$$\Pr(y_i|t_i = \pm 1) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp(-\frac{y_i \mp 1}{2\sigma^2}), \tag{6}$$

we obtain

$$L(y_i) = \frac{2y_i}{\sigma^2}. \tag{7}$$

To establish the bipartite graph on which the decoding process is based, check matrix of the LT code should be conducted. We have $G \cdot s^T = e^T$, thus $G \cdot s^T + e^T = 0$. When the current length of code bits is N, the corresponding check matrix becomes

$$H = [G, I_N], \tag{8}$$

where I_N is identity matrix of dimension N. Here the source symbols and code symbols are concatenated forming variable symbols v, namely, $v = [s, e]$, which satisfies $H \cdot v^T = 0$. Fig. 3 presents a simple decoding bipartite graph.

Assume decoding of the LT code starts when N_0 code symbols are received. Among variable nodes, source nodes are regarded as punctured ones which are not transmitted, so the channel input LLR of them are set to be 0s, and only code nodes have useful channel messages. According to (7), the channel input LLR of each code node is $\frac{2y_i}{\sigma^2}$. Let $V(i)$, $C(i)$ denote the index set of variable/check nodes which are connected to the i-th check/variable node, respectively. The maximum iteration times is set to be L_{max}. The current attempt carries out as follows.

1) Initialization. Let $v_{o,i}$ denotes channel information of the i-th variable node, and the initialization becomes

$$v_{o,i} = \begin{cases} 0, & i \leq K \\ 2y_{i-K}/\sigma^2, & i > K. \end{cases} \tag{9}$$

The current iteration l is set to be 1, and all the other messages are set to be 0.

2) Update the messages from check nodes to variable nodes. In the l-th iteration, for each of the N_0 check nodes, the message from the i-th check node to its j-th neighboring variable node is

$$c^l_{i,j} = 2\tanh^{-1}[\prod_{j' \in V(i), j' \neq j} \tanh(v^{l-1}_{j',i}/2)], \tag{10}$$

where $i = 1, 2, \ldots, N_0$, and for each $i, j \in V(i)$.

3) Update the messages from variable nodes to check nodes. In the l-th iteration, for each of the $K + N_0$ variable nodes, the message from the j-th variable node to its j-th neighboring check node is

$$v^l_{j,i} = v_{o,j} + \sum_{i' \in C(j), i' \neq i} c^l_{i',j}, \tag{11}$$

where $j = 1, 2, \ldots, K + N_0$, and for each $j, i \in C(j)$.

4) Update the final LLR messages of source nodes.

$$v^l_j = v_{o,j} + \sum_{i' \in C(j)} c^l_{i',j}, \tag{12}$$

where $j = 1, 2, \ldots, K$, corresponding the indexes of source nodes.

5) Hard decision.

$$\hat{s}^l_j = \begin{cases} 0, & v^l_j \geq 0 \\ 1, & v^l_j < 0, \end{cases} \tag{13}$$

where $j = 1, 2, \ldots, K$. Assign $l = l + 1$. If $\hat{s}^l$ passes CRC, which means $\hat{s}^l = s$, the decoding succeeds. If not and $l \leq L_{max}$, go back to step 2). The current decoding attempt fails when CRC is not satisfied and $l > L_{max}$.

In Fig. 3, it is observed that each of the code nodes has only one single link to its only neighbor. According to (11), the LLR message that a code node passes to its neighboring check node is always the same, namely, its initial channel information. Further, the LLR message that a check node delivers to its only neighboring code node is unnecessary, since the code bits are not required to be recovered. Thus, it is equivalent that the channel information of each code

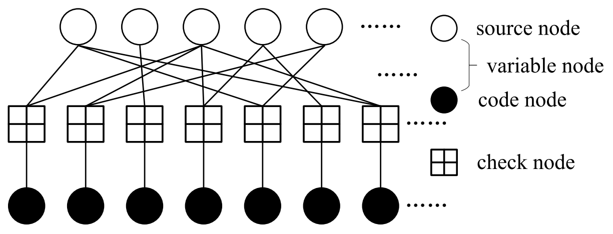

FIGURE 3. Decoding of LT codes.

node should be treated as the inherent message of each corresponding check node, and the message propagation should be constrained only among the source nodes and code nodes. Let $c_{o,i}$ denotes inherent message of the i-th check node, then $c_{o,i} = 2y_i/\sigma^2$. Thus, formula (10) is modified as

$$c^l_{i,j} = 2\tanh^{-1}[\tanh(c_{o,i}/2) \cdot \prod_{j' \in V(i), j' \neq j} \tanh(v^{l-1}_{j',i}/2)]. \tag{14}$$

And the index j in (11) should take values in $1, 2, \ldots, K$.

Additional code symbols are needed when the decoding fails, and we assume the number is N_s. At the same time, the dimension of the check matrix increases and the construction of the decoding bipartite graph varies. Standard parallel (SP) BP decoder restarts the decoding as shown before, but H becomes an $(N_0+N_s-K)\times(N_0+N_s)$ one, and the number of code nodes and check nodes both increase by N_s, whereas parallel storage (PS) BP decoder differs from the SPBP one only in one way. PSBP keeps the LLR information updated in the previous decoding attempt unchanged, and the other steps are the same as those in SPBP.

III. GREEDILY SPREADING SERIAL DECODING

In order for better demonstration of the decoding, we use the concept "step rate", parameterized by R_s, to denote the ratio of number of additional packets to the information dimension. Specifically, the number of additional packets in each decoding attempt is $N_s = K \cdot R_s$.

GSBP propagates the messages greedily, which can facilitate the convergence speed. The first decoding attempt starts when $N_0 + N_s$ encoded packets have been collected. The soft information is first passed from the newly coming N_s code nodes, after they have updated the messages, the neighboring source nodes will transmit the messages to their neighbors. Then, it is these neighboring check nodes' turn to update the messages, noticing that among these check nodes, some have already done the updates, and these will not update twice. Now it is again the source nodes that are connected to the previous check nodes, to update their messages to their neighboring check nodes, and those that have updated will not be included in the current operation as well. A "round" begins when a set of check nodes update the messages, and ends precisely before the very next set of check nodes update the messages. The decoding of the current iteration goes like this way round by round, until the message propagation tracks cover the whole decoding bipartite graph.

Specifically, the current iteration halts when the number of updated code nodes have accounted for 95 percent of the received code nodes.The reason why we set this percentage is that, the occasion that some source code has none links and that the decoding bipartite graph is not fully connected may occurs, and the percentage allows the messages to propagate adequately in the graph.

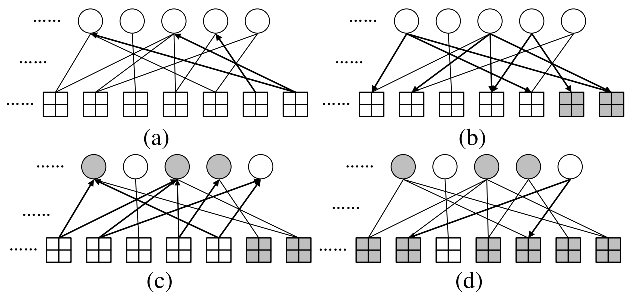

FIGURE 4. Illustration of the proposed decoding.

Fig. 4 gives an example of how the messages are propagated. Fig. 4 (a) and (b) make up one round, (c) and (d) do the next round. The two rightmost check nodes represent newly coming packets. In Fig. 4 (a), LLR messages begin to pass from the two check nodes to their three neighbors, after that, the two are shadowed to indicate that they have been updated. Then the three neighboring source nodes transmit messages to their neighbors, after which they are filled with shadow as well. In Fig. 4 (c), neighboring check nodes except the shadowing ones propagate the messages. The process proceeds in this way until most connections have been covered. In Fig. 4 (d), it is observed that there is still a pair of nodes that can not be reached by the propagation. As mentioned above, similar situation will occur when the bipartite graph is not fully connected. We can employ a parallel decoding for a single iteration at first before the serial decoding, to make the LLR messages totally spread. Moreover, as the increase of the dimension of check nodes, the chance that the graph can not be connected will vanish, and that is where additional packets are required when decoding fails.

Hard decision is performed at the end of each iteration. Within the maximum iterations, if the results passes CRC, the decoding succeeds, otherwise, decoding moves forward to the next iteration. Similarly, additional packets are needed to carry out the next decoding attempt, and our algorithm keeps the LLR messages left by the last attempt unchanged. Not like SPBP or PSBP, the proposed algorithm employ the decoding serially, and the advantage of serial decoding lies in its faster convergence speed.

We treat the channel information as the inherent feature of check nodes and set it in the initialization. And the source nodes are regarded as punctured, so their initial LLR values are set to be 0s and the values are not added to their final LLR messages. Let $V(i)$, $C(i)$ denote the index set of source/check nodes which are connected to the i-th check/source node, respectively. Let C denote the index set of the whole received check nodes. It is assumed the first attempt begins at receiving (N_0+N_s) code symbols. The details of the serial update rules are described as follows.

1) Initialization: Let $c_{o,i}$ denotes inherent message of the i-th check node, and its initialization is

$$c_{o,i} = \frac{2y_i}{\sigma^2}, \tag{15}$$

where $i = 1, 2, \ldots, N_0 + N_s$. The current iteration l is set to be 1, and all the other messages are set to be 0.

2) Initialize the relevant index set. The index set of check/source nodes that are going to be updated in the next round is C_n/V_n, and that have already been updated is C_d/V_d. They are initialized as

$$C_n = \{i|N_0 + 1 \le i \le N_0 + N_s\}, \tag{16}$$
$$C_d = \emptyset, \tag{17}$$
$$V_n = \emptyset, \tag{18}$$
$$V_d = \emptyset. \tag{19}$$

3) Update the messages from check nodes to source nodes.

$$\begin{aligned} c^l_{i,j} = {} & 2\tanh^{-1}[\tanh(c_{o,i}/2) \\ & \cdot \prod_{j'\in V(i), j'\neq j, j'\notin V_d} \tanh(v^{l-1}_{j',i}/2) \\ & \cdot \prod_{j'\in V(i), j'\neq j, j'\in V_d} \tanh(v^{l}_{j',i}/2)], \end{aligned} \tag{20}$$

where $i \in C_n$, and for each $i, j \in V(i)$.

4) Update the relevant sets. After step 3), the index set of source nodes that are to be updated is changed, and that of check nodes which have been updated is changed as well.

$$V_n = \bigcup_{i\in C_n} V(i) - V_d, \tag{21}$$
$$C_d = C_d \bigcup C_n. \tag{22}$$

5) Update the messages from source nodes to check nodes.

$$v^l_{j,i} = \sum_{i'\in C(j), i'\neq i, i\notin C_d} c^{l-1}_{i',j} + \sum_{i'\in C(j), i'\neq i, i\in C_d} c^{l}_{i',j}, \tag{23}$$

where $j \in V_n$, and for each $j, i \in C(j)$.

6) Update the relevant sets. After step 5), the index set of check nodes that are to be updated is changed, and that of source nodes which have been updated is changed as well.

$$C_n = \bigcup_{i\in V_n} C(i) - C_d, \tag{24}$$
$$V_d = V_d \bigcup V_n \tag{25}$$

7) Determine the end of the current iteration. If $|C_d| < 0.95 \cdot |C|$, go back to step 3), otherwise, go to the next step.

8) Update the final LLR messages of source nodes.

$$v_j^l = \sum_{i' \in C(j)} c_{i',j}^l, \tag{26}$$

where $j = 1, 2, \ldots, K$, corresponding the indexes of source nodes.

9) Hard decision.

$$\hat{s}_j^l = \begin{cases} 0, & v_j^l \geq 0 \\ 1, & v_j^l < 0, \end{cases} \tag{27}$$

where $j = 1, 2, \ldots, K$. Assign $l = l + 1$. Three cases may occurs. **Case 1**: If $\hat{s}^l = s$, the decoding succeeds. **Case 2**: If not succeed and $l \leq L_{max}$, go back to step 2). **Case 3**: If $l > L_{max}$ and decoding attempt fails, the decoder waits additional N_s encoded packets to be received to conduct the next attempt. Assign $N_0 = N_0 + N_s$, and go back to step 1), but instead of reset other messages to 0, the next attempt keeps the LLR messages left by the last one unchanged.

IV. ASYMPTOTIC ANALYSIS

A. ANALYSIS OF GREEDY SPREAD

As mentioned in section II, LT encoder selects information symbols uniformly at random to generate code symbols. As the dimension of information symbols and code symbol tend to infinite, the degree of source nodes tend to be Poisson distribution [8]. The polynomial of the distribution is

$$\Lambda(x) = \sum_{i=1}^{Dv_{\max}} \Lambda_i \cdot x^i, \tag{28}$$

where Λ_i denotes the probability that a source node with degree i should be selected, and Dv_{max} is the maximum degree. Λ_i is approximated by $e^{-\theta} \cdot \theta^i / i!$, which makes the degree distribution become

$$\Lambda(x) = e^{\theta(x-1)} = \sum_{i=0}^{\infty} \frac{e^{-\theta} \cdot \theta^i}{i!} \cdot x^i \tag{29}$$

where θ is the average degree, defined by $\theta = Dc_{avg} \cdot N/K$, and Dc_{avg} is the average degree of check nodes, given by

$$Dc_{avg} = \sum_{i=1}^{Dc_{\max}} \Omega_i \cdot i. \tag{30}$$

The degree distributions of source nodes and check nodes determine how fast the propagation covers the graph in the proposed decoding algorithm. At the beginning of decoding, messages are first propagated from the latest N_s check nodes, and the total number of the neighboring source nodes that will receive these messages depends on the degree of these check nodes. Then at the next propagation, it is these source nodes' turn to spread the messages, and the total number of the neighboring check nodes relies on the degree of these source nodes. By this greedy propagation, most of the nodes will be covered at last. In order to research on the speed of the propagation, the degree distribution of a group of nodes should be derived.

We focus on the total number of receiving nodes when a set of transmitting nodes are considered. Here transmitting node refers to the one that sends out messages, which could be check/source node, and receiving node refers to the one that takes in messages, which could be source/check node accordingly. A bunch of transmitting nodes are merged into one single node in order to investigate the combined degree distribution. We use the term "rank" to denote the extent of combination. Rank 1 node is the one with the original degree distribution, namely, the distribution of source node or check node. Specifically, if a node has a rank of n, then it is merged by n node of which each has a rank of 1. The degree distribution of transmitting node with rank 1 is precisely that of a source node or a check node, which is known at the beginning.

When two nodes are merged into one, the degree of the new node is not simply the sum of the degrees of the two old nodes, since there is a chance that the two nodes share the same neighboring nodes. An example of two check nodes merging into one node is shown in Fig. 5. The check nodes in the figure both have 3 neighboring source nodes, and they have one common source node, so when the two nodes merge into one, the new node has a degree of 5 ($3 + 3 - 1$) instead of 6 ($3 + 3$). Thus, the distribution of the merged node can not be directly obtained by the convolution of the two old distributions.

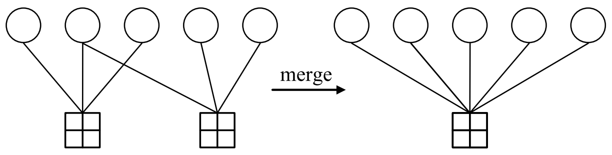

FIGURE 5. An example of merging two check nodes.

Let $D_n(i)$ denotes the probability that a transmitting node with rank n has a degree of i, $Dr_n(i, r)$ denotes the probability of degree i while the old two nodes to be merged share r same neighbors, and M denotes the total number of receiving nodes, which limits the maximum degree of the merged node ($0 \leq i \leq M$). Considering two nodes may have common neighbors, the degree distribution of transmitting node with rank n could be derived from those of two nodes with rank $n - 1$ and rank 1 respectively. The recursive rule is derived as

$$Dr_n(i, r) = \sum_{m=1}^{i} [D_{n-1}(m) \cdot D_1(i + r - m) \cdot \frac{\binom{m}{r} \cdot \binom{M-m}{i-m}}{\binom{M}{i+r-m}}] \tag{31}$$

and

$$D_n(i) = \sum_{r=0}^{i} Dr_n(i, r). \tag{32}$$

It is not hard to understand the above equation. The two transmitting nodes to be merged may share at least zero or at most i neighbors when the merged new node has a degree of i, thus, we obtain (32). A new merged node of degree i where r neighbors have been shared must be combined by two nodes whose degrees add to $i + r$, and that is why the "$\sum$" comes in (31). The fraction item in (31) denotes the probability that a rank $n - 1$ node with degree m and a rank 1 node with degree $i + r - m$ share r same neighbors, and it should be noted that the numerator and denominator have a common factor "$\binom{M}{m}$", which has been removed. Multiplied by the removed factor, the denominator would denote the number of possible combinations of transmitting node with m neighbors and the other with $i + r - m$ neighbors, where the neighbors are among the M receiving nodes, and the numerator would denote the number of possible combinations of the two nodes sharing r same receiving nodes.

As an example, we will investigate the combined degree distribution when the number of source/code node is $K = 1000/N = 2000$. The widely used output degree distribution in [9] is applied, which is

$$\begin{aligned}\Omega(x) &= 0.007969x + 0.493570x^2 + 0.166220x^3 \\ &\quad + 0.072646x^4 + 0.082558x^5 + 0.056058x^8 \\ &\quad + 0.037229x^9 + 0.055590x^{19} + 0.025023x^{65} \\ &\quad + 0.003135x^{66}.\end{aligned} \tag{33}$$

According to (28)~(30), the degree distribution of source node is approximated by

$$\Lambda(x) = \sum_{i=1} \frac{e^{-11.74} \cdot 11.74^i}{i!} \cdot x^i. \tag{34}$$

For convenience of computation, those 'Λ_i's less than 10^{-4} is neglected, and the remaining fractions are normalized so that they sum to 1.

Let vector D_n^c denotes the degree distribution of the check node of rank n, and D_n^s denotes that of the source node. Then D_1^c and D_1^s is defined by (33) and (34), respectively. The combined degree distribution of merged check/source node with any rank can be deduced based on (31) and (32). Fig. 6 illustrates the degree distributions of merged check node (CN) with rank 1, 400, 800, 1200, 1600 and 2000, and those of merged source node (SN) is presented in Fig. 7. It is observed that the degree of the two are both like Gaussian distribution when the nodes are with a high rank, regardless of what the original distribution is. In Fig. 6, the degree concentrates near the maximum degree when the rank tends to K, which means that, as the number of check nodes become larger, the probability of covering the whole source nodes tend to be 1, and the situation is true of Fig. 7.

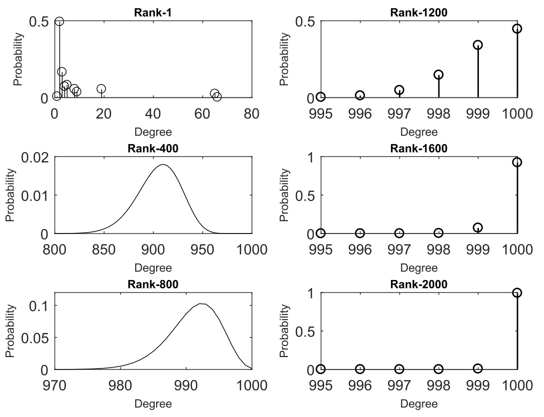

FIGURE 6. Degree distributions of merged CN under different ranks.

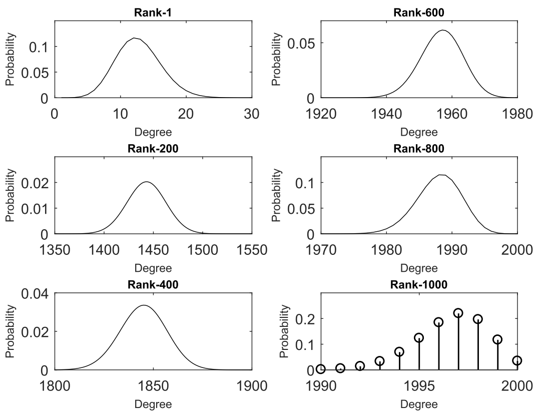

FIGURE 7. Degree distributions of merged SN under different ranks.

The degree distributions of the merged node are utilized to research on the speed of the message propagation. Since the propagation starts from the latest N_s code nodes, the average degree of the merged code node with rank N_s is treated as the expected number of the neighboring source nodes. Then making use of the degree distribution of the merged source node, the expected number of neighboring check nodes can be derived. Thus, the average degree of merged transmitting node is regarded as the rank of the merged receiving node. By averaging the degree of merged CN/SN in the whole range of rank, the process of message propagation can be depicted in Fig. 8, where $N_s = 100$.

The solid line in Fig. 8 is the average degree of merged check node, with its rank on the horizontal axis and the related degree on the vertical axis. The dash dot line corresponds to that of merged source node, with its rank on the vertical axis and the degree on the horizontal axis. By swapping the axes of the curve for source node like this, the evolution of the message propagation can be shown with arrows in one figure. LLR messages start from the latest N_s code nodes, and arriving at neighboring source nodes of an expected number of 444. Then messages are updated and transmitted to neighboring check nodes of an average number of 1882. Tracking the arrows in Fig. 8, it is observed that it takes 3

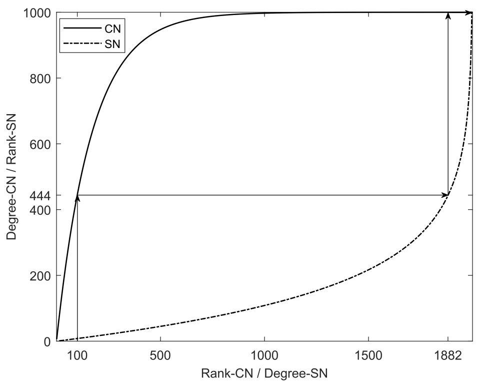

FIGURE 8. **Speed of message propagation.**

rounds of propagation on average to cover all the connections in the decoding graph.

Base on (2), (28), (32) and (31), the speed of the message propagation can be derived like the form in Fig. 8, for any other degree distributions. And it is observed that the proposed decoding exchanges messages in a fast speed, allowing the updated LLR messages to be immediately made advantage of in the current iteration, and most of the nodes will be covered in several rounds.

B. ANALYSIS OF CONVERGENCE SPEED

On AWGN channels, Gaussian approximation [24] is applied to study the speed of convergence during the iterations. The theory in [24] approximates the LLR messages among the two kinds of nodes to variables conformed to Gaussian distribution. The mean and variance of the distribution vary during the iterations, and error probability can be deduced based on the two parameters.

The channel input LLR message is regarded as the inherent message of check node, rewritten as $c_{o,i} = 2y_i/\sigma^2$. Since the error probability is independent of the specific values of information bits, all zero bits are encoded in order to analyze the decoding performance, without loss of generality. Then 0s are mapped into 1s by BPSK, and $y_i = 1 + n_i$, where n_i is Gaussian noise with variance σ^2 and mean 0. We get $c_{o,i} = (1+n)\cdot 2/\sigma^2$, and

$$c_{o,i} \sim N(2/\sigma^2, 4/\sigma^2). \tag{35}$$

It is observed that the variance of the Gaussian variable is twice as large as the mean, and this condition is enforced to all LLR messages during the analysis. Thus, only means of Gaussian variables are needed to investigate the rate of convergence.

Edge degree distribution should be obtained to start Gaussian approximation. Here edge refers to the connection between SN and CN. The degree of an edge at the SN's/CN's side is defined as the degree of its connected SN/CN, which partly features the propagation depth of the massages passed on this edge. Degree distribution of the edge on the side of SN can be calculated by $\Lambda(x)'/\Lambda(1)'$, which is actually the same of $\Lambda(x)$ in (29), given by

$$\lambda(x) = e^{\theta(x-1)} = \sum_{i=0}^{\infty} \frac{e^{-\theta}\cdot\theta^i}{i!}\cdot x^i = \sum_{i=1}^{\infty} \lambda_i \cdot x^{i-1}, \tag{36}$$

where λ_i denotes the probability that an edge is linked to an SN with degree i. And accordingly, the edge degree distribution on the CN's side is

$$\omega(x) = \Omega(x)'/\Omega(1)' = \sum_{i=1}^{Dc_{\max}} \omega_i \cdot x^{i-1}. \tag{37}$$

Let m_v denote the mean value of LLR messages sending from source node, and m_c denote that from check node. Except for the channel input information, all the LLR messages are initialized to 0. We will first analyze the decoding convergence of conventional parallel algorithm.

In (11), the initial value of SN is 0, and input messages of SN are considered independent. Since all the variables on the LHS and RHS of (11) are Gaussian, the evolution of $m_{v,i}$ for the i-th SN could be obtained by

$$m_{v,i}^l = (|C(i)|-1)m_c^l, \tag{38}$$

where $|C(i)|$ denotes the degree of the i-th SN. m_v is the average of $m_{v,i}$ of the whole SN. Since LLR messages are transmitted along the edges, m_v could be derived based on the edge distribution of SN. Thus,

$$m_v^l = \sum_{i=1} (i-1)\cdot\lambda_i\cdot m_c^l = (\theta-1)m_c^l. \tag{39}$$

Assuming the input messages of CN are independent, from (14) we have

$$E\left[\tanh(c_{i,j}^l/2)\right] = E\left[\tanh(c_{o,i}/2)\right] \cdot \prod_{j'\in V(i), j'\neq j} E\left[\tanh(v_{j',i}^{l-1}/2)\right] \tag{40}$$

By Gaussian approximation,

$$E\left[\tanh(c_{i,j}^l/2)\right] = \frac{1}{\sqrt{4\pi m_c^l}} \int_R \tanh(\frac{u}{2})\cdot e^{-\frac{(u-m_c^l)^2}{4m_c^l}} du. \tag{41}$$

Define $\phi(x)$ as

$$\phi(x) = \begin{cases} 1-\dfrac{1}{\sqrt{4\pi x}}\displaystyle\int_R \tanh(\frac{u}{2})\cdot e^{-\frac{(u-x)^2}{4x}} du, & x>0 \\ 1, & x=0, \end{cases} \tag{42}$$

where $\phi(\infty)=0$. $\phi(x)$ is continuous, convex and monotonically decreasing on $[0,\infty)$. To calculate $\phi(x)$, [24] gives an approximated form as

$$\phi(x) = \begin{cases} \sqrt{\dfrac{\pi}{x}}(1-\dfrac{10}{7x})\cdot e^{-x/4}, & x>10 \\ e^{\alpha x^{\beta}+\gamma}, & 0<x\le 10, \end{cases} \tag{43}$$

where $\alpha = -0.4527$, $\beta = 0.86$ and $\gamma = 0.0218$. Then, from (40), we have

$$m_{c,i}^l = \phi^{-1}\{1-[1-\phi(m_0)][1-\phi(m_v^{l-1})]^{|V(i)|-1}\}, \tag{44}$$

where m_0 is the mean of $c_{o,i}$, namely, $2/\sigma^2$. Similarly, m_c^l is obtained from $m_{c,i}^l$ based on the edge distribution of CN, that is

$$m_c^l = \sum_{i=1}^{Dc_{\max}} \omega_i \cdot \phi^{-1}\{1 - [1 - \phi(m_0)][1 - \phi(m_v^{l-1})]^{i-1}\}. \tag{45}$$

With (45) and (39), the evolution of the LLR messages for parallel decoding could be figured out.

In the next, we will investigate the decoding convergence of the proposed algorithm. Let r_{max} denote the maximum round in one iteration. $V_{n,r}/C_{n,r}$ refers to the index set of SN/CN that is going to update the output messages in the r-th round. $V_{n,0}$ and $C_{n,0}$ are both initialized to $\emptyset$. Let $m_{v,r}^l/m_{c,r}^l$ indicate the mean of output messages of SN/CN in the r-th round, l-th iteration. $m_{v,0}^0/m_{c,0}^0$ are both initialized to 0.

In (20), except for the inherent message, the input messages of CN consist of two part. One part comes from the SN in the last round, which is newly updated, the other comes from the SN in the current round, which have not updated in the current iteration but have done in the precious one. Thus, the evolution rule of output messages of CN is

$$m_{c,r}^l = \sum_{i=1}^{Dc_{\max}} \omega_i \phi^{-1}\{1 - [1 - \phi(m_0)] \cdot [1 - \phi(\mu_1 m_{v,r}^{l-1} + \mu_2 m_{v,r-1}^l)]^{i-1}\}, \tag{46}$$

with

$$\mu_1 = \frac{|V_{n,r}|}{|V_{n,r}| + |V_{n,r-1}|}, \quad \mu_2 = \frac{|V_{n,r-1}|}{|V_{n,r}| + |V_{n,r-1}|}. \tag{47}$$

The mean of output LLR messages of CN in the l-th iteration is

$$m_c^l = \frac{\sum_{r=1}^{r_{\max}} |C_{n,r}| \cdot m_{c,r}^l}{\sum_{r=1}^{r_{\max}} |C_{n,r}|}. \tag{48}$$

Similarly, according to (23), the evolution rule of output messages of SN is given by

$$m_{v,r}^l = (\theta - 1) \cdot (\eta_1 m_{c,r+1}^{l-1} + \eta_2 m_{c,r}^l), \tag{49}$$

with

$$\eta_1 = \frac{|C_{n,r+1}|}{|C_{n,r+1}| + |C_{n,r}|}, \quad \eta_2 = \frac{|C_{n,r}|}{|C_{n,r+1}| + |C_{n,r}|}. \tag{50}$$

The mean of output LLR messages of SN in the l-th iteration is

$$m_v^l = \frac{\sum_{r=1}^{r_{\max}} |V_{n,r}| \cdot m_{v,r}^l}{\sum_{r=1}^{r_{\max}} |V_{n,r}|}. \tag{51}$$

With (48) and (51), the evolution of the LLR messages for the proposed decoding could be worked out.

The values of the mean are related to the corresponding $|V_{n,r}|$ or $|C_{n,r}|$. To compare the evolution of the parallel and the proposed serial decoding, we will work out by simulation. Degree distribution of check node in (33) is applied. The dimension of the source nodes is set as $K = 1000$, and that of the check nodes is set as $N = 1000, 1200, 1400, 1600, 1800, 2000$, and σ^2 is set to 1. $|V_{n,r}|$ and $|C_{n,r}|$ for the proposed decoding are obtained based on the previous subsection.

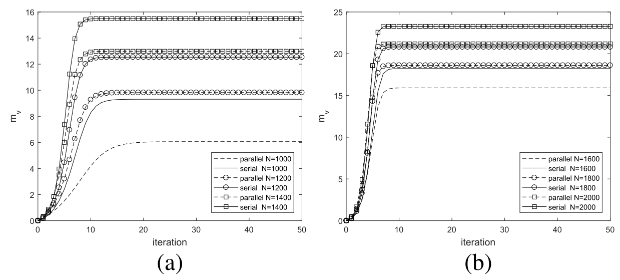

FIGURE 9. **Evolution of m_v under the parallel and serial decoding.**

Fig. 9 shows the evolution of m_v, where the dash-dot curve is that of the parallel decoding and the solid line is that of the proposed serial decoding. At the beginning, serial m_v is slightly smaller than parallel m_v, after that, the former increases faster than the latter, and the gap between them grows wider. Though the gap between the parallel decoding and the serial one tend to be smaller as N increases, the evolution of m_v of the latter is still faster than the former. Both curves stay at a specific iteration, which is caused by fixed point introduced in [24]. As m_v becomes larger, the decoding error probability gets smaller, and it will tend to 0 as m_v tend to infinity. When the evolution curve hits a fixed point, the error probability stops decreasing. However, the occurrence of fixed point is skipped here, since we focus on the speed of convergence. Fig. 9 illustrates that the proposed serial decoding is more effective than the parallel one when the same number of iterations are employed. Further, in order to achieve a specific error probability, the proposed algorithm needs fewer iterations.

C. COMPARISON OF COMPLEXITY

Complexity of decoding depends on the structure of the code and the decoding algorithm. Since the two algorithms are performed based on the same decoding graph, here the discussion of the structure is skipped. Reviewing (14) and (20), the numbers of items that involved in the update of check nodes are the same, though the messages may come from the last iteration or the current iteration. For parallel decoding and serial decoding, output messages of CN are both updated only once in one iteration, thus, the complexity of the two algorithm is the same at the CN's end. The same situation is true for the SN. Reviewing (11) and (23), since all the initial messages of SN are 0, $v_{o,j}$ in (11) could be omitted. Number of addition items to update SN for parallel decoding equals to that for the serial one, and all the SN are updated only once in one iteration, thus, the complexity of the two algorithm is also the same at the SN's end.

For the proposed algorithm, there are additional set operations. In the analysis of greedy spread, it is observed that only several rounds should be taken to cover one iteration. Also, since decoding complexity concentrates on the update of the nodes, the complexity contributed by set operations could be neglected. Thus, the complexity of the conventional parallel algorithm and the proposed one is nearly the same, with respect to one iteration. The whole complexity is directly related with the decoding iterations. The last subsection shows that decoding convergence of the proposed algorithm is faster than the conventional one, therefore, the overall complexity of the former is potentially better than the latter.

V. SIMULATION RESULTS

In this section, we present simulation results of SPBP, PSBP, and GSBP decoding results for LT codes over AWGN. Results are obtained by averaging 1000 times of relevant simulations, and the generator matrix of LT codes keeps the same for these algorithms in each simulation. Degree distribution in (33) is used, and the SNR of the AWGN channel is set to $-2.83\ dB$, where the channel capacity is $0.5\ bit/symbol$. Different maximum iterations are considered in order to observe the BER performance. The decoding starts when $N = 2000$, and the step rate is set to 0.1.

Fig. 10 and Fig. 11 give the bit error performance under the three algorithms. It is observed that the GSBP performs the best at different code rates under the same maximum iterations. While the BER of SPBP is the highest, since it discards the messages of the last decoding attempt. When l_{max} is no larger than 40, GSBP obtains rather lower BER than the other two at the same code rate. As the increasing of l_{max}, the gaps between the three curves becomes narrower. In Fig. 11, it is witnessed that the curves start from nearly the same point, differ among code rate $2.1^{-1} \sim 2.4^{-1}$. And GSBP along with PSBP arrives at around $5 * 10^{-4}$ when $R = 2.5^{-1}$. Actually, increasing of l_{max} after 80 or so does not give much improvement in terms of BER, especially for PSBP and GSBP. In order to reach a BER of 10^{-3}, GSBP needs a maximum iteration of 40 and a code rate of 2.4^{-1}, whereas PSBP requires l_{max} of 40 but a code rate less of 2.5^{-1}, or 200 for l_{max} when $R = 2.4^{-1}$.

As mentioned in section IV, the complexity of these algorithms is the same within one iteration. Since the conventional algorithms need more iterations or more additional packets to achieve the same BER, the proposed GSBP has the smallest complexity among them all.

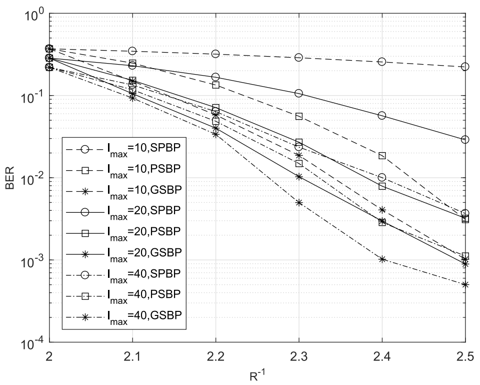

FIGURE 10. BER under different algorithms with $l_{max} = 10, 20, 40$.

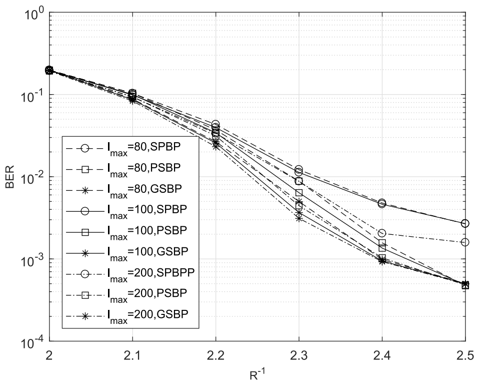

FIGURE 11. BER under different algorithms with $l_{max} = 80, 100, 200$.

VI. CONCLUSIONS

In this paper we proposed a greedy spreading serial BP decoding of LT codes over AWGN channel. Traditional soft decoding of LT codes was reviewed, and we explained why check nodes hold the channel input information as the inherent values. In GSBP, LLR messages are firstly propagated from the newly coming check nodes to their neighbors, then the propagation is conducted from ones to all their neighbors. By analyzing the spreading speed, it was found that the nodes would be covered just in several rounds. Then we showed that the decoding convergence of GSBP was faster than conventional algorithm by Gaussian approximation, and the former had less decoding complexity. Simulations demonstrate that the proposed algorithm has better BER performance than traditional algorithms over AWGN channel. Future work will consider combining precoding with LT codes to further improve the BER performance.

REFERENCES

[1] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 56–67, Oct. 1998.

[2] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1528–1540, Oct. 2002.

[3] D. J. C. MacKay, "Fountain codes," *IEE Proc. Commun.*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[4] T. Nozaki, "Fountain codes based on zigzag decodable coding," in *Proc. Int. Symp. Inf. Theory Its Appl.*, Oct. 2014, pp. 274–278.

[5] B. Jun, P. Yang, J. S. No, and H. Park, "New fountain codes with improved intermediate recovery based on batched zigzag coding," *IEEE Trans. Commun.*, vol. 65, no. 1, pp. 23–36, Jan. 2017.

[6] B. Chen, C. Zhu, W. Li, J. Wei, V. C. M. Leung, and L. T. Yang, "Original symbol phase rotated secure transmission against powerful massive MIMO eavesdropper," *IEEE Access*, vol. 4, pp. 3016–3025, Jun. 2016.

[7] J. Xiong, L. Cheng, D. Ma, and J. Wei, "Destination-aided cooperative jamming for dual-hop amplify-and-forward MIMO untrusted relay systems," *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 7274–7284, Sep. 2016.

[8] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci.*, Nov. 2002, pp. 271–280.

[9] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[10] L. Suo, G. Zhang, L. Jing, and T. Xiang, "Performance analysis for finite length LT codes via classical probability evaluation," *IEEE Commun. Lett.*, vol. 21, no. 9, pp. 1957–1960, Sep. 2017.

[11] S. Kim, K. Ko, and S. Y. Chung, "Incremental Gaussian elimination decoding of raptor codes over BEC," *IEEE Commun. Lett.*, vol. 12, no. 4, pp. 307–309, Apr. 2008.

[12] S. Kim, S. Lee, and S. Y. Chung, "An efficient algorithm for ML decoding of raptor codes over the binary erasure channel," *IEEE Commun. Lett.*, vol. 12, no. 8, pp. 578–580, Aug. 2008.

[13] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, "On the fly gaussian elimination for LT codes," *IEEE Commun. Lett.*, vol. 13, no. 12, pp. 953–955, Dec. 2009.

[14] F. Lazaro, G. Liva, and G. Bauch, "Inactivation decoding of LT and Raptor codes: Analysis and code design," *IEEE Trans. Commun.*, vol. 65, no. 10, pp. 4114–4127, Oct. 2017.

[15] Y. Cao and S. D. Blostein, "Cross-layer raptor coding for broadcasting over wireless channels with memory," in *Proc. 11th Can. Workshop Inf. Theory*, May 2009, pp. 130–135

[16] K. Wang, Z. Chen, and H. Liu, "A novel decoding scheme for LT-codes in wireless broadcasting systems," *IEEE Commun. Lett.*, vol. 17, no. 5, pp. 972–975, May 2013.

[17] C. Cao, H. Li, and Z. Hu, "A new cross-level decoding scheme for LT codes," *IEEE Commun. Lett.*, vol. 19, no. 6, pp. 893–896, Jun. 2015.

[18] H. Jenkac, T. Mayer, T. Stockhammer, and W. Xu, "Soft decoding of LT codes for wireless broadcast," in *Proc. IST Mobile Summit*, 2005, pp. 1–5.

[19] K. Zhang, X. Huang, and C. Shen, "Soft decoder architecture of LT codes," in *Proc. IEEE Workshop Signal Process. Syst.*, Oct. 2008, pp. 210–215.

[20] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[21] K. Hu, J. Castura, and Y. Mao, "Reduced-complexity decoding of Raptor codes over fading channels," in *Proc. IEEE Globecom*, Nov. 2006, pp. 1–5.

[22] K. Wu, Z. Zhang, S. Chen, S. Yang, and P. Qiu, "Serial decoding of rateless code over noisy channels," *J. Zhejiang Univ. Sci. C*, vol. 12, no. 10, pp. 855–866, 2011.

[23] J. Castura and Y. Mao, "Rateless coding over fading channels," *IEEE Commun. Lett.*, vol. 10, no. 1, pp. 46–48, Jan. 2006.

[24] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

LIANG HE received the B.S. degree in information and communication engineering from the National University of Defense Technology, Changsha, China, in 2017, where he is currently pursuing the M.S. degree with the Department of Communication Engineering, School of Electronic Science. His research interests include channel coding, iterative decoding, and data mining.

JING LEI received the B.Sc., M.Sc., and Ph.D. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 1990, 1994, and 2009, respectively. She was a Visiting Scholar with the School of Electronics and Computer Science, The University of Southampton, U.K. She is currently a Distinguished Professor with the Department of Communication Engineering, College of Electronic Science, NUDT, and also the Leader of the Communication Coding Group. She has published many papers in various journals and conference proceedings and five books. Her research interests include information theory, LDPC, space–time coding, advanced multiple access technology, physical-layer security, and wireless communication technology.

YING HUANG received the B.E. degree in electrical engineering from Xiangtan University, Xiangtan, China, in 2000, and the M.S. and Ph.D. degrees in information and communication engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2002 and 2014, respectively. She was a Visiting Scholar with the Department of Electronics and Computer Science, The University of Southampton, U.K. She is currently an Associate Professor with the Department of Communication Engineering, NUDT. She has published in excess of 20 journal and international conference papers. Her research interests include physical-layer security, channel codes, modulation recognition, and coded cooperation.