A New Iterative LT Decoding Algorithm for Binary and Nonbinary Galois Fields

Yuexin Mao, Jie Huang, Bing Wang, Jianzhong Huang, Wei Zhou, and Shengli Zhou

Abstract: **Digital fountain codes are record-breaking codes for erasure channels. They have many potential applications in both wired and wireless communications. Most existing digital fountain codes operate over binary fields using an iterative belief-propagation (BP) decoding algorithm. In this paper, we propose a new iterative decoding algorithm for both binary and nonbinary fields. The basic form of our proposed algorithm considers both degree-1 and degree-2 check nodes (instead of only degree-1 check nodes as in the original BP decoding scheme), and has linear complexity. Extensive simulation demonstrates that it outperforms the original BP decoding scheme, especially for a small number of source packets. The enhanced form of the proposed algorithm combines the basic form of the algorithm and a guess-based algorithm to further improve the decoding performance. Simulation results demonstrate that it can provide better decoding performance than the guess-based algorithm with fewer guesses, and can achieve decoding performance close to that of the maximum likelihood decoder at a much lower decoding complexity. Last, we show that our nonbinary scheme has the potential to outperform the binary scheme when choosing suitable degree distributions, and furthermore it is insensitive to the size of the Galois field.**

Index Terms: **Binary erasure channel (BEC), decoding algorithm, digital fountain codes, Luby transform (LT) codes, nonbinary.**

I. INTRODUCTION

Digital fountain codes [1]–[4] are record-breaking codes for erasure channels [5]. In such codes, an encoder is a fountain that produces an endless supply of water drops (encoded packets). For k original packets, a receiver only needs to receive k' (which is about 5% larger than k in practice) packets to recover the original k packets [6]. Fountain codes are rateless in the sense that the number of encoded packets that can be generated from a source is potentially limitless. Furthermore, the number of encoded packets generated can be determined on the fly. The first practical realization of digital fountain codes is Luby transform (LT) codes invented by Luby [7]–[9]. LT codes are very similar to Gallager's low-density parity-check (LDPC) codes [10], but they use a highly irregular weight distribution for the underlying graphs designed over the binary erasure channel (BEC) [11]–[14]. Later, an improved version, Raptor codes [15], was invented by Shokrollahi. Raptor codes introduce a precoding phase to relax the condition of LT codes: When a constant fraction of source packets is decoded through LT codes, all the source packets can be recovered through an erasure correcting block code (while LT codes need to decode all source packets successfully).

The properties of digital fountain codes have made them attractive choices for both wired and wireless communications [4]. Firstly, since they do not need to feedback the intermediate information to the transmitter immediately (only after recovering all the source packets, the receiver sends the feedback information to the transmitter), and the amount of redundancy can be determined on the fly, they are applicable to highly asymmetric wireless links with unknown and/or time-varying loss rates. Secondly, their efficient encoding and decoding algorithms allow them to be executed on low-power embedded devices, and these codes are *universal* in the sense that the decoder is capable of recovering the original symbols from any set of output symbols whose size is close to the optimal size with high probability. Last, they are very suitable for multicast/broadcast applications: With fountain codes, repair packets required by a receiver can be useful to all receivers in a multicast group since each receiver only needs to receive any subset of k' packets to recover the original data. Indeed, recently, a version of Raptor codes has been selected as the global standard for multimedia broadcast/muticast services (MBMSs) to 3G mobile devices [16].

For decoding, an iterative belief-propagation (BP) decoding algorithm for LT codes is similar to that of LDPC codes over the BEC. This decoding algorithm is near optimal for very large k, while its performance may not be good for small k. For small to moderate k, [17] has proposed near-linear decoding algorithms for LDPC codes over BECs. Especially, when the number of guesses is unlimited, the guess based algorithm (Algorithm C in [17]) is an efficient implementation of the optimal maximum likelihood (ML) decoder. The same idea has also been proposed in [18].

In this paper, we propose a new hard iterative decoding algorithm over the BEC for small to moderate k. The novelty of our proposed algorithm lies in that it proceeds from check nodes instead of source nodes as in [17] and [18]. Specifically, the basic form of the algorithm considers both degree-1 and degree-2 check nodes: When there is no degree-1 check node, it introduces a *merge function* that merges the source nodes that are

Manuscript received August 14, 2012; approved for publication by Emanuele Viterbo, Division I Editor, May 2, 2013.

The work was partially supported by NSF CAREER award 0746841 and NSF Grant No. 1128581. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Y. Mao and B. Wang are with the Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269, USA, email:{yuexin.mao, bing}@engr.uconn.edu.

J. Huang and W. Zhou are with Marvell Semiconductor, Santa Clara, CA 95054, USA, email: {jiehuang, wei}@marvell.com.

J.-Z. Huang is with LSI Corporation, San Jose, CA, 95131, USA, email: hjzhong.xidian@gmail.com.

S. Zhou is with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269, USA, email: shengli@engr.uconn.edu.

Digital Object Identifier 10.1109/JCN.2013.000072

connected to degree-2 check nodes, and continues the decoding process. This basic algorithm has linear complexity, and extensive simulation demonstrates that it significantly outperforms the original LT decoding scheme (that only considers degree-1 check nodes), especially for small k. This algorithm is similar to the algorithm in [19] that is proposed for LDPC codes. However, our work is done independently and concurrently with the work in [19]. We further propose an enhanced form of the algorithm that combines the basic algorithm and a guess-based algorithm that is inspired by [17]. The enhanced algorithm runs the basic algorithm first, and then applies the guess-based algorithm when the basic algorithm stops (i.e., no degree-1 or degree-2 check node exists). Running the basic algorithm first reduces the decoding complexity, while running the guess-based algorithm afterwards further improves the decoding performance. Our simulation results demonstrate that the enhanced algorithm can provide better decoding performance than the guess-based algorithm with fewer guesses (and hence has lower decoding complexity). Furthermore, the enhanced algorithm achieves decoding performance close to that of the ML decoder with much lower decoding complexity.

Our second main contribution is that we extend the proposed iterative decoding scheme to nonbinary fields while most existing digital fountain codes operate over binary fields. With the proposed decoder, we find that our nonbinary decoding scheme has the potential to outperform its binary counterpart when the degree distribution is chosen properly. Furthermore, our nonbinary scheme is insensitive to the size of the Galois field, and a small size, such as 4, is sufficient to realize most of the performance gains.

The rest of the paper is organized as follows. Section II reviews the encoding and decoding algorithms of LT codes and Raptor codes. Section III presents our proposed decoding algorithm in the binary field and evaluates its performance through simulation. Section IV presents our proposed decoding algorithms in nonbinary Galois fields. Last, Section V concludes the paper and presents future work.

II. LT CODES AND RAPTOR CODES

In this section, we briefly describe LT codes and Raptor codes. Consider k source packets, $s_1, \cdots, s_k$, and n encoded packets , $c_1, \cdots, c_n$. We slightly abuse notation to use $c_1, \cdots, c_n$ to represent both check node $c_1, \cdots, c_n$ and its related value, respectively. Similarly, we use $s_1, \cdots, s_k$ to represent source node $s_1, \cdots, s_k$ and its related value, respectively. In general, n is slightly larger than k (about 5% larger for large k) [6]. In practice, LT encoding is applied on symbols at the same position across different packets. For example, assume each packet has p symbols, then p parallel LT codes are used to encode k packets into n packets. Each packet has its own cyclic redundancy check (CRC) bits to verify the correctness. Denote $\mathbf{s} = [s_1, \cdots, s_k]^T$ and $\mathbf{c} = [c_1, \cdots, c_n]^T$. A LT code can be described as $\mathbf{c} = \mathbf{G}\mathbf{s}$ where the matrix $\mathbf{G}$ defines a Tanner graph that specifies the connections between source and encoded packets. Note that the source and encoded packets in the Tanner graph are represented by source nodes and check nodes, respectively. The iterative LT decoding process is a hard decoding algorithm as described in [6].

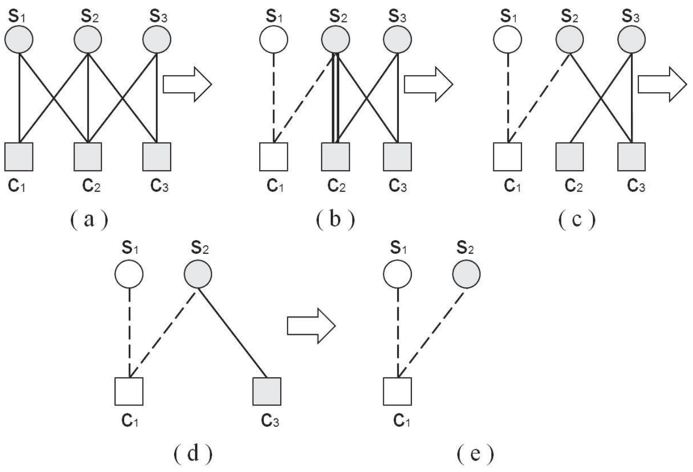


Fig. 1. Illustration of the basic LT decoding algorithm in the binary field ($k = 3, n = 3$).

Raptor codes improve upon LT codes by introducing a precoding phase that relaxes the condition of LT codes. In precoding phase, m source packets encode to k intermediate packets with an (k, m) erasure correcting block code that ensures to recover all the m source packets from l intermediate packets, $k \geq l \geq m$. Then, k intermediate packets are encoded into n packets using LT codes. Thus, LT codes only need to decode a fraction of intermediate packets (l packets or more), then all source packets can be decoded. Raptor codes are the first known class of fountain codes with near-linear time encoding and decoding.

III. PROPOSED BINARY LT DECODING ALGORITHM

In this section, we describe the proposed LT decoding algorithm in the binary field. The algorithm has two forms: Basic form and enhanced form. The basic form improves upon the original BP decoding algorithm by considering both degree-1 and degree-2 check nodes. In particular, when there is no degree-1 check node, it introduces a merge function that merges the source nodes that are connected to degree-2 check nodes, and continues the decoding process. To further improve the decoding performance, the enhanced form improves the basic form by introducing a guess-based algorithm when neither degree-1 nor degree-2 check nodes exist. The decoding complexity is defined as the number of addition and multiplication operations per source packet. We may assign larger weights to multiplication operations since multiplications tend to be more complex than additions. In this paper, for simplicity, we do not differentiate addition and multiplication operations when calculating the decoding complexity. Let N denote the number of addition and multiplication operations in the decoding process. Then, the decoding complexity $\mathcal{O}$=N/k. In the following, we first describe the basic and enhanced algorithms along with their decoding complexities, and then evaluate their performance through simulation.

A. Basic Algorithm

We first use the example in Fig. 1 to illustrate the basic algorithm, and then describe the algorithm formally. In this example, three source nodes, s_1, s_2, and s_3, connect to three check nodes, c_1, c_2, and c_3, satisfying

$$s_1 \oplus s_2 = c_1, \tag{1}$$

$$s_1 \oplus s_2 \oplus s_3 = c_2, \tag{2}$$

$$s_2 \oplus s_3 = c_3. \tag{3}$$

The original BP decoding scheme fails to decode any of the source packets because there is no degree-1 check node, while it is easy to see that all the three source nodes can be decoded successfully: We can decode s_3 by adding (1) and (2) as $s_3 = c_1 \oplus c_2$, and after that, we can decode s_2 through (3), and then, decode s_1 through (1). Our algorithm decodes all the three source nodes successfully through message passing in the Tanner graph. For ease of illustration, initially, we mark all the check nodes and source nodes in the Tanner graph as grey, and all the edges between check nodes and source nodes as solid (see Fig. 1(a)). The basic algorithm proceeds as follows. First, we find a degree-2 check node, c_1. The two source nodes, s_1 and s_2, that are connected to c_1 satisfy (1), and hence, if one of them is decoded, the other can be decoded through (1). Therefore, we can merge these two source nodes into a single node. To reduce the number of operations, we merge the node with a smaller degree to the one with a larger degree. In this example, since s_1 has a smaller degree, we merge s_1 into s_2 by representing s_1 using s_2 as $s_1 = c_1 \oplus s_2$, referred to as a *merge function*, and store the merge function (since it will be used in later stages to decode the node that is being merged). The above is represented as a sequence of operations in the Tanner graph: We remove c_1, and two edges, (s_1, c_1) and (s_2, c_2), that are connected to c_1 (for ease of illustration, we mark c_1 and s_1 as white and the two edges, (s_1, c_1) and (s_2, c_1), as dashed in the Tanner graph), and shift all the other edges that are connected to s_1 to s_2 (in this example, edge (s_1, c_2) is shifted to s_2, leading to an additional edge from s_2 to c_2). In addition, the values of the check nodes that are originally connected to s_1 need to be updated by adding c_1 (since $s_1 = c_1 \oplus s_2$ and we use s_2 to represent s_1), which is represented as s_1 passing a value c_1 to these check nodes (before merging s_1 into s_2). In this example, s_1 is connected to c_1, and hence, s_1 passes a value c_1 to c_2 so that c_2 can update its value to $c_2' = c_1 \oplus c_2$. Summarizing the above, we have the graph in Fig. 1(b). In Fig. 1(b), c_2 represents check node c_2, not the value of c_2. Therefore, although the value of c_2 has changed, we still represent it as c_2 in the figure. The same notation convention holds for all the figures in this paper. Then, the two edges from s_2 to c_2 both have coefficients of 1, and hence, cancel each other. This canceling is represented by removing the two edges from s_2 to c_2, leading to Fig. 1(c). At this point of time, we have a degree-1 check node, c_2, that is connected to source node s_3, and hence s_3 is successfully decoded as $s_3 = c_2' = c_1 \oplus c_2$. We remove c_2, pass the value of s_3 to all the other check nodes that are connected to s_3, i.e., c_3 in this case, and c_3 updates its value to $c_3' = c_3 \oplus c_2' = c_1 \oplus c_2 \oplus c_3$. We then remove all the edges from s_3, i.e., (s_3, c_2) and (s_3, c_3), and remove s_3, leading to Fig. 1(d). Afterwards, s_2 is connected to a degree-1 check node, c_3, and hence s_2 can be decoded. This is represented as removing c_3 and deleting edge (s_2, c_3) in the Tanner graph, leading to Fig. 1(e). After decoding s_2, we can decode s_1 with the stored merge function $s_1 = c_1 \oplus s_2$. The basic decoding algorithm is described more formally as follows.

Basic Binary Decoding Algorithm

(1) Find a degree-1 check node c_i that is connected to a single source node s_j.

(a) Set the value of c_i to s_j. Recover all the source nodes that have been merged into s_j (the merge operations are described in step (2)). Increase the number of operations, N, by the number of source nodes thus recovered (since recovering each such source node requires one addition).

(b) Pass the value of s_j to all the other check nodes that are connected to s_j. For each such check node c_l, update its value to $c_l' = c_l \oplus c_i$. Increase the number of operations, N, by the number of such check nodes (updating each check node requires one addition).

(c) Delete all the edges connected to s_j.

(2) When step (1) stops (i.e., no degree-1 check node exists), find a degree-2 check node. If no degree-2 check node can be found, stop the algorithm. Otherwise, suppose we find a degree-2 check node, c_l, that is connected to two source nodes s_i and s_j. Store the merge function $c_l = s_i \oplus s_j$, and merge the source node with a smaller degree into the one with a larger degree. In the following, suppose we merge s_i into s_j for illustration.

(a) Remove c_l and the two edges (s_i, c_l) and (s_j, c_l).

(b) Pass the value of c_l to all the other check nodes that are connected to s_i. Suppose there are m such check nodes. For each such check node c_r, update its value to $c_r' = c_r \oplus c_l$. Shift edge (s_i, c_r) to be an edge (s_j, c_r). If there exist two edges between s_j and c_r after edge shifting, cancel these two edges. Increase the number of operations N by $2m$ (since the above procedure incurs two additions, one for updating the value and one for shifting the edge, for each of the m check nodes).

(3) Go to step (1) until all source nodes are determined, or no degree-1 or degree-2 check node can be found.

It is clear that the above decoding algorithm has linear complexity.

B. Enhanced Algorithm

The enhanced algorithm improves upon the basic algorithm by continuing the decoding process through a guess-based algorithm once the basic algorithm stops (i.e., no degree-1 or degree-2 check node can be found). The guess-based algorithm is inspired by Algorithm C in [17] whose complexity is related to the size of the graph. In this paper, we use a simpler rule to determine the source nodes to guess. Furthermore, to reduce the decoding complexity, we first start with the basic algorithm to reduce the size of the graph, and then apply the guess-based algorithm to improve the performance. As the number of guesses increases, the performance of the guess-based algorithm can approach the optimal ML decoder with increased decoding complexity. In the paper, we allow up to $g_{\max}$ guesses.

The guess-based algorithm chooses source nodes to guess and follows a check node labeling procedure as follows. Recall that the basic algorithm stores the merge functions, removes all source nodes that have been decoded successfully, and removes

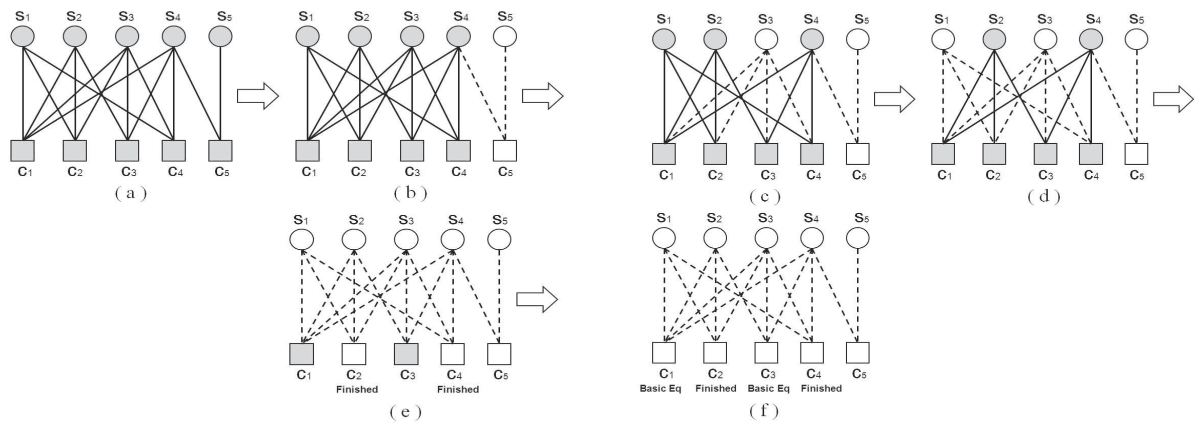


Fig. 2. Illustration of the enhanced LT decoding algorithm ($k = 5, n = 5$).

all check nodes that have been used to decode source nodes or used in the merge functions. When the guess-based algorithm starts, it marks the source nodes that have not been decoded or merged as *unknown*, and marks the remaining check nodes from the basic algorithm as *unlabeled*. It then sequentially guesses the values of up to $g_{\max}$ unknown source nodes. After guessing a source node, it marks the source node as known. It then sequentially checks all the unlabeled check nodes. Specifically, consider an unlabeled check node, c_i. If the values of all but one of the source nodes that are connected to c_i are known, then it sets the value of the unknown source node to the XOR of the known source nodes and c_i, and then labels c_i as *finished*; if all the source nodes that are connected to c_i are known, then it labels c_i as a *basic equation*. If all the check nodes are labeled or the number of guesses reaches $g_{\max}$, it then solves the set of basic equations to obtain values of guessed source nodes, and then recovers the involved source nodes.

We next use the example in Fig. 2 to illustrate the decoding process of the enhanced algorithm, and then describe the algorithm in detail. In the example, five source nodes, s_1, s_2, s_3, s_4, and s_5, connect to five check nodes, c_1, c_2, c_3, c_4, and c_5, satisfying

$$s_1 \oplus s_2 \oplus s_3 \oplus s_4 = c_1, \tag{4}$$

$$s_1 \oplus s_2 \oplus s_3 = c_2, \tag{5}$$

$$s_2 \oplus s_3 \oplus s_4 = c_3, \tag{6}$$

$$s_1 \oplus s_3 \oplus s_4 = c_4, \tag{7}$$

$$s_4 \oplus s_5 = c_5. \tag{8}$$

For ease of illustration, initially, all source nodes and check nodes are marked as grey, and the connections between these source nodes and check nodes are marked as solid (see Fig. 2(a)). We first run the basic decoding algorithm, and find a degree-2 check node, c_5. The two source nodes, s_4 and s_5, that are connected to c_5 satisfy (8). So, we merge the node with a smaller degree to the one with a larger degree. In this case, we merge s_5 into s_4 by representing s_5 using s_4 as $s_5 = s_4 \oplus c_5$, and store this merge function. We then, remove c_5 and the two edges (s_4, c_5) and (s_5, c_5) (correspondingly, we mark s_5 and c_5 as white, and the two edges (s_4, c_5) and (s_5, c_5) as dashed in the Tanner graph, see Fig. 2(b)). Then, the basic decoding algorithm fails to go further since there is no degree-1 or degree-2 check node. So, we continue with the guess-based algorithm, which starts with marking all grey source nodes as unknown and all grey check nodes as unlabeled. We then find an unknown source node with the largest degree (when there are multiple such nodes, we select one of them arbitrarily), mark it as known, and let x_1 be its value. In this example, s_3 has a degree of 4, all other three nodes have the degree of 3. We, therefore, choose s_3, and mark it as known with the value of x_1. This is represented in the Tanner graph in Fig. 2(c), where we mark s_3 in white, and mark the edges incident to s_3 as dashed. After that, we examine all the unlabeled check nodes, and find none of them can be labeled. So, we make the second guess. The remaining unknown source nodes has the same degree. We arbitrarily choose s_1, assume it is known, and let x_2 be its value (in the Tanner graph, we mark s_1 as white, and the edges incident to s_1 as dashed, see Fig. 2(d)). After that, for check node c_2, all but one source node that are connected to c_2 are known, we hence label c_2 as finished, and set the value of the unknown source node, s_2, to $x_1 \oplus x_2 \oplus c_2$. Similarly, we label c_4 as finished, and set the only unknown source node that c_4 connects to, i.e., s_4, as $x_1 \oplus x_2 \oplus c_4$. The above operations are illustrated by marking c_2, s_2, c_4, and s_4 as white, and marking the edges incident to s_2 and s_4 as dashed in the Tanner graph (see Fig. 2(e)). Now, since all the source nodes are known, we label c_1 and c_3 as *basic equations*, and mark them as white in the Tanner graph (see Fig. 2(f)). Last, we solve the two basic equations of c_1 and c_3 based on (4) and (6). Substituting the values of s_1, s_2, s_3, and s_4 into these two equations yields

$$x_2 \oplus x_1 = c_1 \oplus c_2 \oplus c_4, \tag{9}$$

$$x_1 = c_2 \oplus c_3 \oplus c_4. \tag{10}$$

From (9) and (10), we solve for x_1 and x_2 to obtain $x_1 = c_2 \oplus c_3 \oplus c_4$ and $x_2 = c_1 \oplus c_3$, and hence decode s_3 and s_1 successfully. Because s_2 and s_4 are represented by x_1 and x_2, we can also decode s_2 and s_4. After we decode s_4, we can decode s_5 based on the stored merge function as $s_5 = s_4 \oplus c_5$. Hence, all the five source nodes are decoded successfully using the enhanced algorithm. Our enhanced algorithm is described more formally as follows.

Enhanced Binary Decoding Algorithm

(1) Run the basic binary decoding algorithm.

(2) When step (1) stops, start the guess-based algorithm. Let g represent the number of guesses. Initialize it to be 1. Do the following:

(a) Find the source node that has the largest degree among all the unknown source nodes. Suppose it is s_i. Assume it is known, and set its value to x_g. Check all the unlabeled check nodes. For a check node c_j, suppose m source nodes are connected to it. Consider the following two scenarios:

(a1) If all but one of the m source nodes are known, then label c_j as finished. Suppose the value of the lth known source node is

$$a_{0,l} \oplus a_{1,l}x_1 \oplus \cdots \oplus a_{g,l}x_g,\ \{a_{0,l}, \ldots, a_{g,l}\} \in \{0,1\}$$

where $a_{g,l}$ represents the coefficient of x_g in the expression of the lth source node. Then, the unknown source node is computed as

$$\underbrace{(\mathrm{a}_{0,1} \oplus \cdots \oplus \mathrm{a}_{0,\mathrm{m}-1} \oplus \mathrm{c_j})}_{\mathrm{b}_0} \oplus \underbrace{(\mathrm{a}_{1,1} \oplus \cdots \oplus \mathrm{a}_{1,\mathrm{m}-1})}_{\mathrm{b}_1} x_1 \oplus \cdots \oplus \underbrace{(\mathrm{a}_{\mathrm{g},1} \oplus \cdots \oplus \mathrm{a}_{\mathrm{g},\mathrm{m}-1})}_{\mathrm{b_g}} x_g$$

where $\{b_0, b_1, \cdots, b_g\}$ represent the numbers of non-zero values in the summations. Increase the number of operations N by $\sum_{l=0}^{g} \max(b_l - 1, 0)$. The value of the unknown source node can be represented as

$$a_{0,m} \oplus a_{1,m}x_1 \oplus \cdots \oplus a_{g,m}x_g,\\ \{a_{0,m}, a_{1,m}, \cdots, a_{g,m}\} \in \{0,1\}.$$

(a2) If all the m source nodes are known, then label c_j as a basic equation. The basic equation can be represented as

$$\underbrace{(\mathrm{a}_{1,1} \oplus \cdots \oplus \mathrm{a}_{1,\mathrm{m}})}_{\mathrm{b}_1} x_1 \oplus \cdots \oplus \underbrace{(\mathrm{a}_{\mathrm{g},1} \oplus \cdots \oplus \mathrm{a}_{\mathrm{g},\mathrm{m}})}_{\mathrm{b_g}} x_g = \underbrace{(\mathrm{a}_{0,1} \oplus \cdots \oplus \mathrm{a}_{0,\mathrm{m}} \oplus \mathrm{c_j})}_{\mathrm{b}_0}.$$

Increase the number of operations N by $\sum_{l=0}^{g} \max(b_l - 1, 0)$.

(b) If all the check nodes are labeled or the number of guesses g reaches $g_{\max}$, go to step (c). Otherwise, increase g by 1, and go back to step (a).

(c) Solve the set of basic equations using Gaussian elimination to obtain solutions to $x_1, \cdots, x_g$. If there exists a unique solution, all the source nodes that have been assigned $x_1, \cdots, x_g$ are decoded successfully; otherwise, a subset of the source nodes are decoded successfully.

(d) For each source node decoded in step (c), recover all the source nodes that have been merged into it (during the basic algorithm). Increase the number of operations N by the number of such recovered source nodes.

It is clear that the decoding complexity of the enhanced algorithm increases with $g_{\max}$. Since $g_{\max}$ is usually a small number, solving the basic equations using Gaussian elimination can be done quickly. Based on [17], the decoding complexity of the enhanced algorithm is $O(g_{\max}^2 k_{\mathrm{enh}})$, where k_{enh} is the number of unknown source nodes when the enhanced algorithm starts.

C. Performance Evaluation

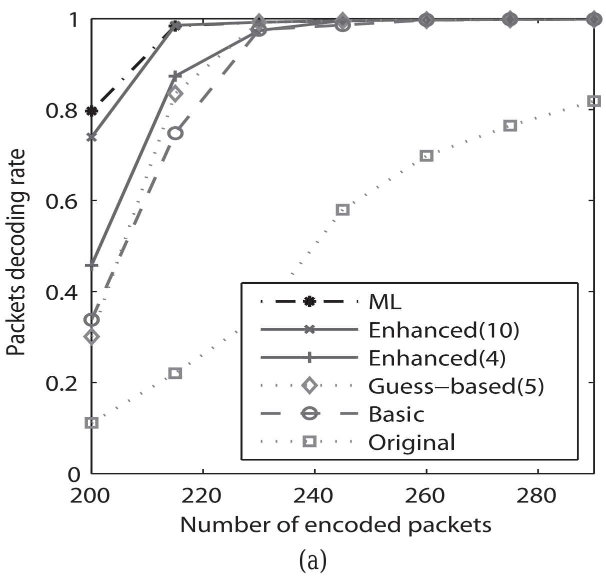
Assume that a sender uses Raptor codes to encode k source packets into n encoded packets, and transmits these encoded packets to a receiver. We use the degree distribution that has been adopted as the standard for MBMS in 3GPP for encoding [16]. More specifically, the degree distribution is

$$\Omega(x) = 0.01563x^{40} + 0.07986x^{11} + 0.11134x^{10} + 0.11339x^4 + 0.21096x^3 + 0.45905x^2 + 0.00977x. \tag{11}$$

For convenience, we refer it to the *MBMS degree distribution* henceforth. Our performance metrics are *packet decoding rate*, i.e., the total number of packets that are decoded successfully over the number of source packets, k, and *decoding complexity*. Our results below are obtained from 200 simulation runs, using randomly generated seeds. For each setting, we set k to 100, 200, 500, or 1000. The 95% confidence intervals are tight and hence omitted from the figures.

In the following, we only report the results for $k = 200$ (the results for other values of k have similar trends). Fig. 3(a) plots the packets decoding rates of five decoding algorithms: (i) ML algorithm (implemented using Gaussian elimination), (ii) the enhanced algorithm with $g_{\max} = 10$ and $g_{\max} = 4$, (iii) the guess-based algorithm with $g_{\max} = 5$ (which is similar to Algorithm C in [17]), (iv) the basic algorithm, and (v) the original BP algorithm (step (1) in the basic algorithm). We can see that the basic algorithm can decode up to 55% more packets than the original BP decoder. The packet decoding rate of the enhanced algorithm improves as the number of allowed guesses increases, and approaches that of the ML decoder when $g_{\max} = 10$. Fig. 3(b) plots the decoding complexity for the considered algorithms. As expected, the decoding complexity of the original BP decoder is the lowest, followed by the basic algorithm, the enhanced algorithm, and the ML decoder (the ML decoder is implemented using Gaussian elimination and its decoding complexity is much higher than all the others. Hence, it is omitted from the figure for clarity). The enhanced algorithm with $g_{\max} = 10$ is up to 90% lower than that of the ML decoder. When $g_{\max}$ increases, the decoding performance approaches that of the ML decoder, while the decoding complexity is also increasing. So, the optimal $g_{\max}$, which is a tradeoff between the decoding complexity and the decoding performance should be chosen depending on the requirements from the particular applications.

The enhanced algorithm runs the basic algorithm first, and then starts the guess-based algorithm when no degree-1 or degree-2 check node exists. For comparison, we also obtain the results when running the guess-based decoding algorithm alone. In Figs. 3(a) and (b), the enhanced algorithm with $g_{\max} = 4$ achieves higher packet decoding rates for all values of n and achieves lower decoding complexity for $n > 215$ than the guess-based algorithm with $g_{\max} = 5$. This demonstrates the importance of running the basic algorithm first in the enhanced algorithm to reduce graph size, and hence the decoding complexities. When $n \le 215$, the enhanced algorithm with $g_{\max} = 4$ has higher decoding complexity than the guess-based algorithm with $g_{\max} = 5$. This can be explained as that the



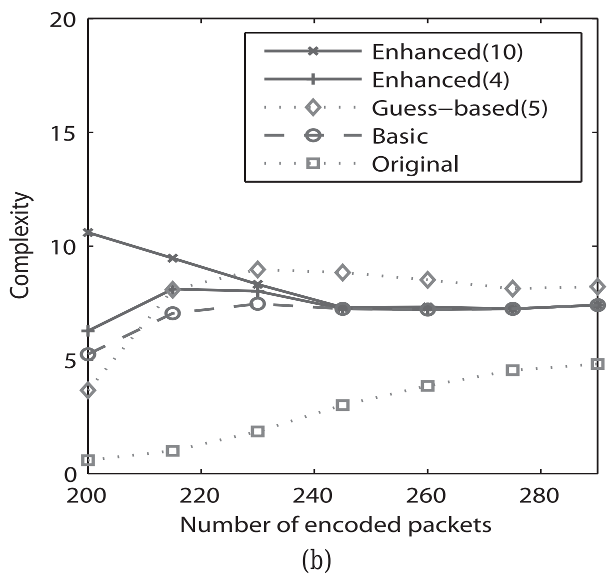


Fig. 3. Performance of the proposed LT decoding algorithm and the guess-based algorithm in the binary field under the MBMS degree distribution, $k = 200$ and n is varied from 200 to 295: (a) Packet decoding rate and (b) decoding complexity.

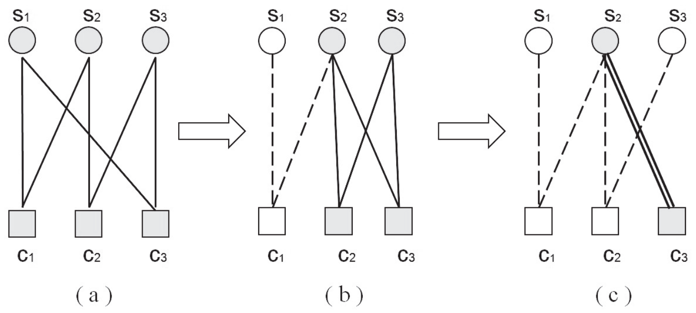


Fig. 4. Illustration of the basic LT decoding algorithm in nonbinary fields ($k = 3, n = 3$).

decoding rate of the basic algorithm is low, a large number of unknown source packets are left when the enhanced algorithm starts. Last, we briefly compare the performance of the basic algorithm and the guess-based algorithm. We find that the basic algorithm can decode more packets than the guess-based algorithm with $g_{\max} = 4$ (not plotted), while decode less packets than the guess-based algorithm with $g_{\max} = 5$. On the other hand, the basic algorithm has lower decoding complexity than the guess-based algorithm ($g_{\max} = 4$ or 5) for $n > 200$.

IV. PROPOSED NONBINARY LT DECODING ALGORITHM

Most of existing digital fountain codes are defined over the binary field [18]. In this section, we extend the proposed binary LT decoding algorithm to nonbinary fields. The motivations are two-fold: (i) Operating in nonbinary fields, the ML decoding for LDPC codes can potentially have better performance than its binary counterparts given the same sparsity of the encoding matrix [20]. We expect that when choosing the degree distributions properly, nonbinary LT decoding algorithms can outperform their binary counterparts. (ii) One unique advantage of nonbinary codes over binary codes is that nonbinary codes can match very well the underlying modulation scheme at the physical layer, and bypass the need for a symbol-to-bit conversion at the receiver.

A. Basic Nonbinary Decoding Algorithm

Suppose the decoding is over a nonbinary field, $\mathrm{GF}(q), q > 2$. We first use the example in Fig. 4 to illustrate this algorithm, and then describe the algorithm formally. In this example, three source nodes, s_1, s_2, and s_3, connect to three check nodes, c_1, c_2, and c_3. Each edge is associated with a coefficient. Let G_{ji} be the coefficient on edge (c_j, s_i), $G_{ji} \in \mathrm{GF}^*(q)$, where $\mathrm{GF}^*(q) = \mathrm{GF}(q) \setminus \{0\}$. We have

$$G_{11}s_1 \oplus G_{12}s_2 = c_1, \tag{12}$$
$$G_{22}s_2 \oplus G_{23}s_3 = c_2, \tag{13}$$
$$G_{31}s_1 \oplus G_{33}s_3 = c_3. \tag{14}$$

We first find a degree-2 check node, c_1. The two source nodes, s_1 and s_2, that are connected to c_1 satisfy (12). Again, we merge the node with a smaller degree to the one with a larger degree (breaking tie arbitrarily). In this case, since s_1 and s_2 have the same degree, without loss of generality, we merge s_1 into s_2 by representing s_1 using s_2 as $s_1 = G_{11}^{-1}(c_1 \oplus G_{12}s_2) = G_{11}^{-1}c_1 \oplus G_{11}^{-1}G_{12}s_2$, and store this merge function. We then remove c_1 and the two edges (s_1, c_1) and (s_2, c_1). Using s_2 to represent s_1, (14) is transformed to

$$G_{31}G_{12}G_{11}^{-1}s_2 \oplus G_{33}s_3 = c_3', \tag{15}$$

with $c_3' = c_3 \oplus G_{31}G_{11}^{-1}c_1$. Therefore, s_1 passes a value, $G_{31}G_{11}^{-1}c_1$, to check node c_3, and c_3 updates its value to c_3'. When merging s_1 into s_2, edge (s_1, c_3) is shifted to s_2, leading to an edge from s_2 to c_3 with the coefficient of $G_{31}G_{12}G_{11}^{-1}$, as shown in Fig. 4(b). We then find another degree-2 check node, c_2. The two source nodes, s_2 and s_3, that are connected to c_2 satisfy (13). Since s_2 and s_3 have the same degree, without loss of generality, we use s_2 to represent s_3 as

$s_3 = G_{23}^{-1}(c_2 \oplus G_{22}s_2) = G_{23}^{-1}c_2 \oplus G_{23}^{-1}G_{22}s_2$, and remove c_2 and the two edges (c_2, s_2) and (c_2, s_3). Using s_2 to represent s_3, (15) is transformed to

$$(G_{31}G_{12}G_{11}^{-1} \oplus G_{33}G_{22}G_{23}^{-1})s_2 = c_3'', \tag{16}$$

with $c_3'' = c_3' \oplus G_{33}G_{23}^{-1}c_2$. Therefore, s_3 passes a value, $G_{33}G_{23}^{-1}c_2$, to check node c_3, and c_3 updates its value to c_3''. When merging s_3 into s_2, edge (s_3, c_3) is shifted to s_2, leading to an edge from s_2 to c_3 with the coefficient of $G_{33}G_{22}G_{23}^{-1}$, as shown in Fig. 4(c). The two edges from s_2 to c_3 can then be merged into a single edge by adding their coefficients together on the Galois field, which only cancel out when the addition is zero which differs from the binary case where the two edges cancel out each other. If the combined coefficient is not equal to zero, then s_2 can be recovered by (16). Furthermore, given s_2, we can decode s_1 and s_3 using the stored merge functions. Hence, all the source nodes can be decoded successfully.

Remark: Note that if the above example were in the binary field (i.e., $G_{ji} = 1$), none of the source nodes can be decoded successfully. In general, if the connections among r source nodes and r degree-2 check nodes are cyclic, with appropriate ordering, the corresponding equations can be represented as $\mathbf{c} = \mathbf{G}\mathbf{s}$ with $\mathbf{c} = [\mathbf{c_1}, \mathbf{c_2}, \cdots, \mathbf{c_r}]^{\mathbf{T}}$, $\mathbf{s} = [\mathbf{s_1}, \mathbf{s_2}, \cdots, \mathbf{s_r}]^{\mathbf{T}}$, and $\mathbf{G}$ given by [21]

$$\mathbf{G} = \begin{bmatrix} G_{11} & G_{12} & 0 & \dots & 0 \\ 0 & G_{22} & G_{23} & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ G_{r1} & \dots & 0 & 0 & G_{rr} \end{bmatrix} \tag{17}$$

where G_{ij}s are nonzero elements from a Galois field. In binary field, all G_{ij}s take value of 1, and the rank of $\mathbf{G}$ is $r-1$. That is, it loses rank, and thus none of the source nodes can be decoded. But, in the nonbinary field, it is possible that the rank of $\mathbf{G}$ equals r. In that case, these equations are independent, and all the source packets involved can be decoded by the proposed decoder with linear complexity. In fact, for this particular example, the basic nonbinary decoding algorithm is similar to the algorithm presented in Lemma 4 of [21]. The basic nonbinary decoding algorithm can be represented more formally as follows.

Basic Nonbinary Decoding Algorithm

(1) Find a degree-1 check node c_i that is connected to a single source node s_j.

(a) Set the value of $G_{ij}^{-1}c_i$ to s_j. Recover all the source nodes that have been merged into s_j (the merge operations are described in step (2)). Increase the number of operations, N, by two times the number of source nodes thus recovered as recovering each such source node requires two operations, one multiplication and one addition.

(b) Pass the value of s_j to all the other check nodes that are connected to s_j. For each such check node c_l, update its value to $c_l' = c_l \oplus G_{lj}s_j = c_l \oplus G_{lj}G_{ij}^{-1}c_i$. Increase the number of operations, N, by two times the number of such check nodes (one multiplication and one addition for each such check node).

(c) Delete all the edges connected to s_j.

(2) When step (1) stops (i.e., no degree-1 check node exists), find a degree-2 check node. If no degree-2 check node can be found, stop the algorithm. Otherwise, suppose we find a degree-2 check node, c_l, that is connected to two source nodes s_i and s_j. Store the merge function $G_{li}s_i \oplus G_{lj}s_j = c_l$, and merge the source node with a smaller degree into the one with a larger degree. In the following, suppose we merge s_i into s_j for illustration. That is, we represent s_i as $s_i = G_{li}^{-1}c_l \oplus G_{li}^{-1}G_{lj}s_j$. Increase the number of operations, N, by 2 (two multiplications to get $G_{li}^{-1}c_l$ and $G_{li}^{-1}G_{lj}$, respectively).

(a) Remove c_l and the two edges (c_l, s_i) and (c_l, s_j).

(b) Pass the value of $G_{li}^{-1}c_l$ to all the other check nodes that are connected to s_i. Suppose there are m such check nodes. For each such check node c_r, update its value to $c_r' = c_r \oplus G_{ri}G_{li}^{-1}c_l$. Increase the number of operations N by $2m$ (one addition and one multiplication for each of the m check nodes). Shift edge (s_i, c_r) to be an edge (s_j, c_r). The coefficient of the new edge is $G_{li}^{-1}G_{lj}G_{ri}$. If after shifting the edge, there exist two edges between s_j and c_r, merge two edges to one and update the coefficient on this edge to $G_{li}^{-1}G_{lj}G_{ri} \oplus G_{rj}$. Increase the number of operations N by $2m$ (since for each of the m check nodes, obtaining the coefficient of the new edge and updating the coefficient requires one multiplication and one addition).

(3) Go to step (1) until all source nodes are determined, or no degree-1 and degree-2 check node can be found.

It is clear that the above decoding algorithm has linear complexity.

B. Enhanced Nonbinary Decoding Algorithm

The enhanced nonbinary decoding algorithm is similar to its binary counterpart. The only differences are in steps (2)(a1) and (a2), which are

(a1) If all but one of the m source nodes are known, then label c_j as finished. Suppose the value of the lth known source node is

$$a_{0,l} \oplus a_{1,l}x_1 \oplus \cdots \oplus a_{g,l}x_g,\ \{a_{0,l}, \cdots, a_{g,l}\} \in \mathrm{GF}(q)$$

where $a_{g,l}$ represents the coefficient of x_g in the expression of the lth source node. Then, the unknown source node is computed as

$$\begin{aligned} G_m'^{-1}[&\underbrace{(\mathrm{G_1'a_{0,1}} \oplus \cdots \oplus \mathrm{G_{m-1}'a_{0,m-1}} \oplus \mathrm{c_j})}_{\mathrm{b_0}} \\ &\oplus \underbrace{(\mathrm{G_1'a_{1,1}} \oplus \cdots \oplus \mathrm{G_{m-1}'a_{1,m-1}})}_{\mathrm{b_1}} x_1 \\ &\oplus \underbrace{(\mathrm{G_1'a_{2,1}} \oplus \cdots \oplus \mathrm{G_{m-1}'a_{2,m-1}})}_{\mathrm{b_2}} x_2 \oplus \cdots \\ &\oplus \underbrace{(\mathrm{G_1'a_{g,1}} \oplus \cdots \oplus \mathrm{G_{m-1}'a_{g,m-1}})}_{\mathrm{b_g}} x_g] \end{aligned}$$

where $\{b_0, b_1, \cdots, b_g\}$ represent the numbers of non-zero values in the summations, and $\{G_1', \cdots, G_m'\}$ represent the coefficients on the edges between the m source nodes and check node c_j. Increase the number of operations N by $\sum_{l=0}^{g} 2b_l$. The value of the unknown source node can be represented as

$$a_{0,m} \oplus a_{1,m}x_1 \oplus \cdots \oplus a_{g,m}x_g,\ \{a_{0,m}, a_{1,m}, \cdots, a_{g,m}\} \in \mathrm{GF}(q).$$

(a2) If all the m source nodes are known, then label c_j as a basic equation. The basic equation can be represented as

$$\underbrace{(\mathrm{G}'_1\mathrm{a}_{1,1}\oplus\cdots\oplus\mathrm{G}'_\mathrm{m}\mathrm{a}_{1,\mathrm{m}})}_{\mathrm{b}_1}x_1\oplus\underbrace{(\mathrm{G}'_1\mathrm{a}_{2,1}\oplus\cdots\oplus\mathrm{G}'_\mathrm{m}\mathrm{a}_{2,\mathrm{m}})}_{\mathrm{b}_2}x_2\oplus\cdots$$
$$\oplus\underbrace{(\mathrm{G}'_1\mathrm{a}_{\mathrm{g},1}\oplus\cdots\oplus\mathrm{G}'_\mathrm{m}\mathrm{a}_{\mathrm{g},\mathrm{m}})}_{\mathrm{b}_\mathrm{g}}x_g=\underbrace{(\mathrm{G}'_1\mathrm{a}_{0,1}\oplus\cdots\oplus\mathrm{G}'_\mathrm{m}\mathrm{a}_{0,\mathrm{m}}\oplus\mathrm{c}_\mathrm{j})}_{\mathrm{b}_0}$$

Increase the number of operations N by $\sum_{l=0}^{g}\max(2b_l-1,0)$.

The decoding complexity of the nonbinary enhanced algorithm is $O(2g^2_{\max}k_{\mathrm{enh}})$, where k_{enh} is the number of unknown source nodes when the enhanced algorithm starts.

Remark: Our nonbinary decoding algorithm operates on symbol-level. In [22] and [23], the proposed nonbinary peeling algorithm operates on bit-level. Let us take an example to illustrate the difference between symbol-level and bit-level nonbinary decoding algorithms. Assume a receiver receives a symbol x which has two bits over $\mathrm{GF}(4)$ over the BEC. The two bits are represented as x_1 and x_2, respectively. If x_1 is received correctly and x_2 is erased, x is discarded in symbol-level algorithms. While in bit-level algorithms, x is still kept, and if bit x_2 can be recovered in the further decoding process, then symbol x can be reconstructed. If the nonbinary peeling algorithm proposed in [22], [23] operates on symbol-level, the decoding process corresponds to a check node of degree-1 in the setting of step (1) in our basic nonbinary LT decoding algorithm. The reason why we choose symbol-level algorithms is that symbol-level algorithms can match very well with the underlying processing at the physical layer. Indeed, as discussed in Section II, a packet in a practical system is completely erased if it cannot pass the CRC. Since our nonbinary enhanced algorithm can achieve decoding performance close to that of the ML decoder (see subsection IV-C), we hence omit the performance comparison between symbol-level and bit-level nonbinary algorithms in this paper.

C. Performance Evaluation

We first evaluate the performance of our proposed nonbinary LT decoder, and then compare the performance of binary and nonbinary decoders. Last, we investigate the impact of the Galois field size on performance. The simulation setting is the same as that in subsection III-C.

C.1 Nonbinary Decoders

We only report the results for $k = 200$ with n varies from 200 to 295 (the results for other values of k have similar trends). Fig. 5(a) plots packet decoding rates with nonbinary ML decoder, binary ML decoder, the enhanced nonbinary LT decoder, the nonbinary guess-based algorithm and the basic nonbinary LT decoder. We can see that nonbinary ML decoder outperforms binary ML decoder under the same degree distribution, which has motivated our work to implement the nonbinary decoding scheme. The performance of the enhanced algorithm improves with g increases. In particular, when $g_{\max} = 15$, its performance is close to that of nonbinary ML decoder, and furthermore it can decode up to 4% more packets than binary ML decoder with $n = 200$. The basic algorithm dramatically outperforms the original LT decoding scheme (see the packet decoding rates of the original scheme in Fig. 3(a)). Fig. 5(b) plots the decoding complexity of the enhanced and basic decoding algorithms. As expected, the decoding complexities of the nonbinary algorithms are larger than their binary counterparts (see Fig. 3(b) on the decoding complexities of the binary schemes). The decoding complexity of the enhanced nonbinary algorithm increases with $g_{\max}$, while is nearly 90% lower than that of the nonbinary ML decoder. The decoding complexity of nonbinary ML decoder is obtained through an implementation using Gaussian elimination; it is not plotted in the figure for clarity.

For comparison, we also obtain the results with the nonbinary guess-based decoding algorithm. Our nonbinary enhanced algorithm with $g_{\max} = 3$ achieves higher packets decoding rates for all values of n while has lower decoding complexity for $n > 200$ than the guess-based algorithm with $g_{\max} = 5$, which demonstrates the importance of running the basic algorithm before initiating the guess-based decoding algorithm in the enhanced decoder.

C.2 Binary versus Nonbinary Decoders

The performance of nonbinary decoders relative to binary decoders depends on the degree distribution. We have shown examples that are decodable in nonbinary fields while they are not in the binary field (see subsection IV-A). We can also come up with examples where the binary decoder outperforms the nonbinary decoder. The binary decoder can perform better because in the binary field, if two source packets s_i and s_j are connected to a check node c_l that has degree of 2, we can merge s_i into s_j. Afterwards, if there are two edges from s_i to a check node c_r, these two edges cancel each other, and hence the degree of c_r is reduced. But, for nonbinary case, they may not cancel each other. So, binary merge function can decrease the degree of check node c_r, and there is a chance that the degree of c_r decreases to 1 (an example is in Fig. 1), which is helpful for continuing the decoding process.

We now compare the performance of our binary and nonbinary decoders through simulation. Under the MBMS degree distribution, the performance of the basic algorithm in nonbinary fields is similar to that in the binary field (figures omitted). On the other hand, based on our prior experience on nonbinary LDPC codes for channel coding [24], we expect the optimal degree distribution in nonbinary fields to have a much smaller average degree and higher density for degree of 2 than its counterpart in the binary field. We, therefore, design another degree distribution as

$$\Omega(x) = 0.0649x^{15} + 0.9253x^2 + 0.0098x. \tag{18}$$

In this new degree distribution, the average node degree is 2.834, much lower than 4.621 in the MBMS distribution; the highest node degree is 15, much lower than 40 in MBMS degree distribution. The much lower average node degree can lead to significantly lower encoding and decoding complexities. On the other hand, due to the much lower highest-node-degree, many source nodes may have no connection to check nodes, leading to lower block decoding rate than that under MBMS degree distribution. This, however, is not much a concern since we use Raptor codes (where packet decoding rate is a more important metric).

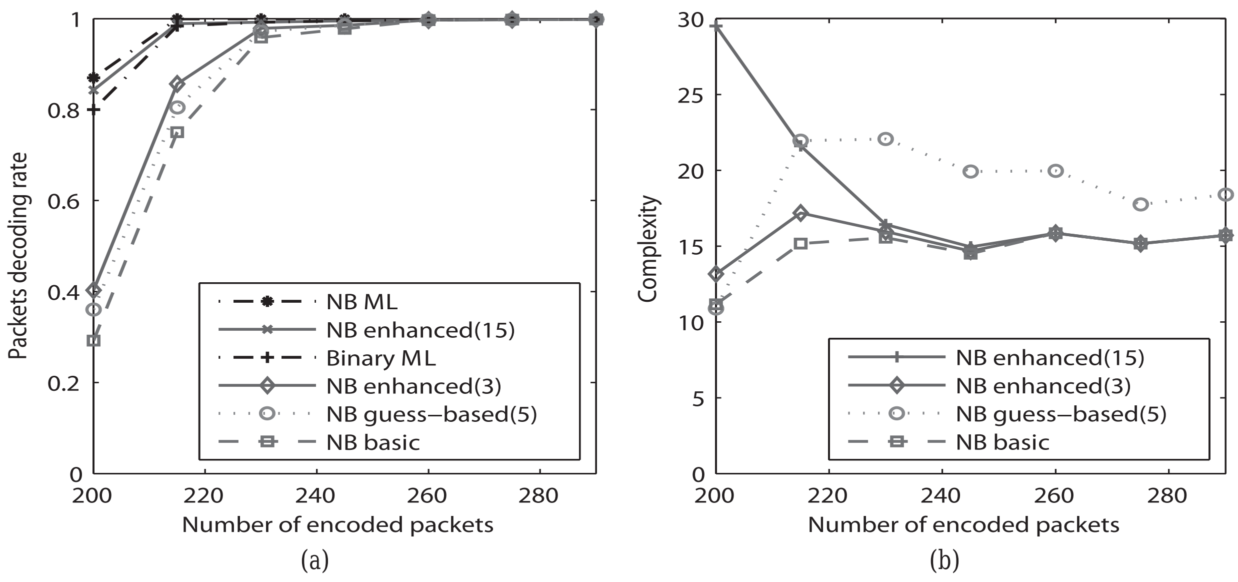


Fig. 5. Performance of the proposed LT decoding algorithm and the guess-based algorithm in nonbinary fields under the MBMS degree distribution, $k = 200$ and n is varied from 200 to 295: (a) Packets decoding rate and (b) decoding complexity.

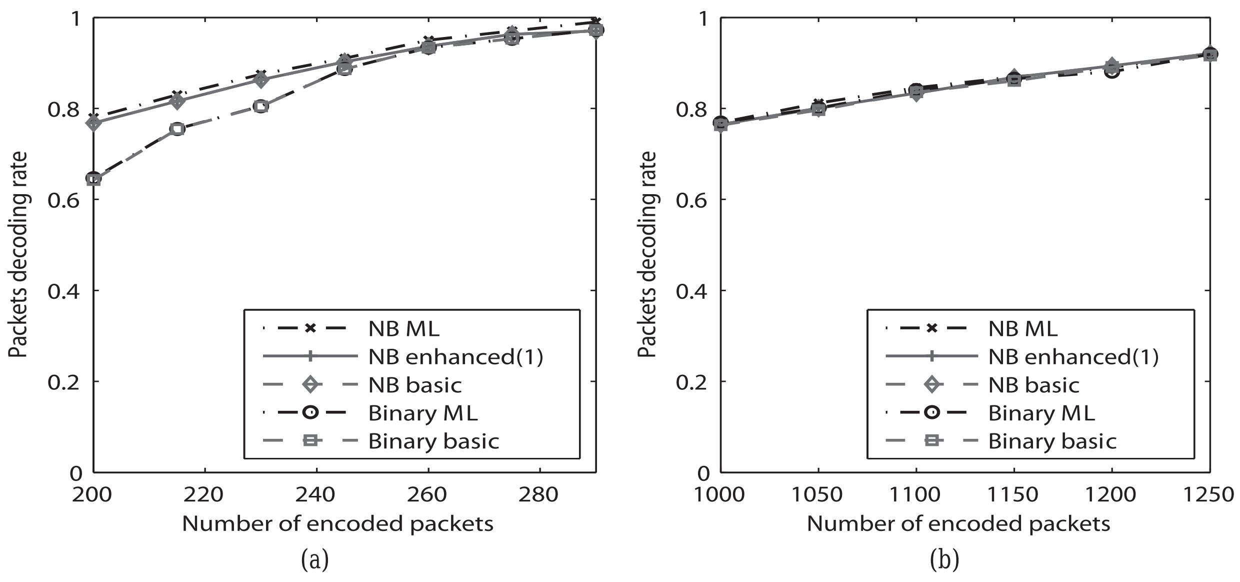


Fig. 6. Packet decoding rates of nonbinary and binary decoders under degree distribution (18): (a) $k = 200$ and (b) $k = 1000$.

Fig. 6(a) plots the packets decoding rate of our proposed algorithms in both binary and nonbinary fields for $k = 200$ under degree distribution (18). For comparison, we also plot the results of binary and nonbinary ML decoders. We observe that the nonbinary enhanced algorithm with up to one guess slightly outperforms the nonbinary basic algorithm, and both achieve performance close to that of the nonbinary ML decoder. The binary basic algorithm and binary ML decoder have similar performance, which is worse than that of the nonbinary decoders (the packet decoding rate is approximately 15%). The different relative performance of nonbinary algorithms compared to their binary counterparts in the two degree distributions (i.e., distributions (18) and (11)) confirms that the relative performance depends on the degree distribution, motivating our future work to develop respective optimal degree distributions for our nonbinary and binary decoders. Fig. 6(b) plots the packet decoding rates when $k = 1000$. We observe that the nonbinary ML decoder performs similar to binary ML decoder, the nonbinary basic algorithm only slightly outperforms the binary algorithm, indicating that the benefits of decoding in nonbinary fields is less dramatic for large k.

C.3 Impact of Galois Field Size

We now investigate the performance of the ML decoder when varying the size of the Galois field from 2 to 1024 (it is the binary decoder when the size is 2). Fig. 7 plots the results using the degree distribution in (18) for $k = 200$. As we can see, the nonbinary decoders outperform the binary decoder, while the performances of the various nonbinary decoders are similar, indicating that our nonbinary scheme is insensitive to the size of the Galois field, and a small size, such as 4, is sufficient to realize most of the performance gains. However, these observations

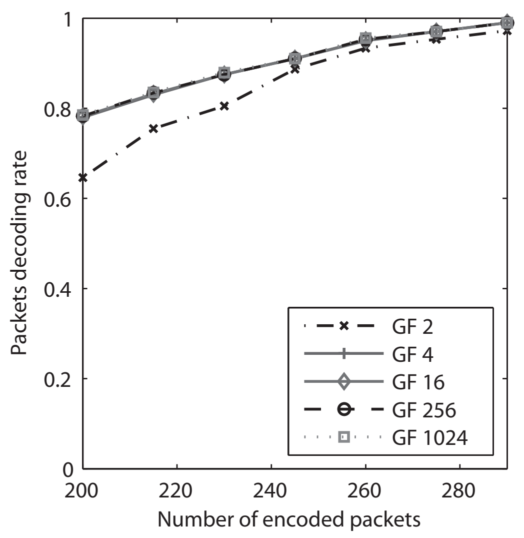


Fig. 7. ML packets decoding rate of GF(2), GF(4), GF(16), GF(256), GF(1024) under degree distribution (18), $k = 200$ and n is varied from 200 to 295.

may change as the degree distribution varies, which is left as further research.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduced a new iterative LT decoding algorithm for both binary and nonbinary fields. Our proposed scheme differs from existing decoding schemes [17], [18] in that it proceeds from check nodes instead of source nodes. The basic form of the algorithm considers both degree-1 and degree-2 check node and has linear complexity. Simulation results demonstrate that it significantly outperforms the original LT decoding algorithm, especially for small k. To further improve the decoding performance, we developed an enhanced form of the algorithm that combines the basic form of the algorithm and a guess-based algorithm. Simulation results demonstrate that it can provide better decoding performance than the guess-based algorithm with fewer guesses. Furthermore, it can achieve the performance of ML decoder as the number of guesses increases, while its decoding complexity is much lower than of the ML decoder. In addition, our nonbinary decoding algorithms have potential to outperform the binary algorithms when choosing suitable degree distributions. Last, our nonbinary scheme is insensitive to the size of the Galois field. Our future research topics include: (i) An analytical framework on performance analysis for the proposed decoders and (ii) optimization of the degree distributions especially for the nonbinary codes.

ACKNOWLEDGEMENTS

The authors would also like to thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proc. ACM SIGCOMM*, Vancouver, BC, Canada, 1998.

[2] J. 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] J. W. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads," in *Proc. IEEE INFOCOM*, 1999.

[4] M. Mitzenmacher, "Digital fountains: A survey and look forward," in *Proc. ITW*, San Antonio, TX, 2004.

[5] P. Elias, "Coding for two noisy channels," in *Proc. Symp. Inf. Theory*, London, UK, 1956.

[6] D. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[7] M. Luby, "LT codes," in *Proc. Annu. IEEE Symp. FOCS*, Vancouver, BC, Canada, Nov. 2002.

[8] ——, "Information additive code generator and decoder for communication systems," in *U.S. Patent No. 6307487*, 2001.

[9] ——, "Information additive code generator and decoder for communication systems," in *U.S. Patent No. 6373406*, 2002.

[10] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.

[11] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 569–584, Feb. 2001.

[12] P. Oswald and M. A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 12, pp. 3017–3028, Dec. 2002.

[13] M. A. Shokrollahi, "New sequences of linear time erasure codes approaching the channel capacity," in *Proc. AAECC*, 1999, pp. 65–76.

[14] ——, "Capacity-achieving sequences," *IMA Volumes in Mathematics and its Applications*, vol. 123, pp. 153–166, 2000.

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

[16] "Technical specification group services and system aspects; multimedia broadcast/multicast services (MBMS); protocols and codecs (Release 6)," 3rd generation partnership project (3GPP), Tech. Rep. 3GPP TS 26.346 V6.3.0, 3GPP, 2005.

[17] H. Pishro-Nik and F. Fekri, "On decoding of low-density parity-check codes over the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 50, no. 3, pp. 439–454, Mar. 2004.

[18] A. Shokrollahi, S. Lassen, and R. Karp, "Systems and processes for decoding chain reaction codes through tnactivation," U.S. Patent 6,856,263, 2005.

[19] P. M. Olmos, J. J. Murillo-Fuentes, and F. Perez-Cruz, "Tree-structured expectation propagation for decoding finite-length LDPC codes," *IEEE Commun. Lett.*, vol. 15, no. 2, pp. 235–237, Feb. 2011.

[20] M. C. Davey and D. Mackay, "Low-density parity-check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, June, 1998.

[21] J. Huang and J.-K. Zhu, "Linear time encoding of cycle GF(2^p) codes through graph analysis," *IEEE Commun. Lett.*, vol. 10, no. 5, pp. 369–371, May, 2006.

[22] V. Rathi, "Conditional entropy of non-binary LDPC codes over BEC," in *Proc. Int. Symp. Inf. Theory*, Toronto, Canada, July 2008.

[23] V. Rathi and I. Andriyanova, "Some results on MAP decoding of nNon-binary LDPC codes over the BEC," *IEEE Trans. Inf. Theory*, vol. 57, no. 4, pp. 2225–2242, Apr. 2011.

[24] J. Huang, S. Zhou, and P. Willett, "Nonbinary LDPC coding for multicarrier underwater acoustic communication," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 9, pp. 1684–1696, Dec. 2008.

Yuexin Mao received his B.S. degree in Electrical Engineering from University of Electronic Science and Technology of China, China in 2007, and M.S. degree in Electrical Engineering from the University of Bridgeport, Bridgeport, in 2009. He is currently a Ph.D. candidate in Computer Science & Engineering Department at the University of Connecticut. His research interests are in the areas of computer networks, network coding, data mining, and social network analysis.

Jie Huang received his Ph.D. degree from University of Science and Technology of China in 2006, majored in Electrical Engineering. He is now with Marvell Semi-conductor at Santa Clara, CA.

Bing Wang is currently an Associate Professor of the Computer Science & Engineering Department at the University of Connecticut. She received her B.S. degree in Computer Science from Nanjing University of Science & Technology, China in 1994, and M.S. degree in Computer Engineering from Institute of Computing Technology, Chinese Academy of Sciences in 1997. She then received M.S. degrees in Computer Science and Applied Mathematics, and a Ph.D. degree in Computer Science from the University of Massachusetts, Amherst in 2000, 2004, and 2005, respectively. Her research interests are in computer networks, multimedia, and distributed systems. She received NSF CAREER Award in 2008.

Jianzhong Huang (M'13) received the B.S. and M.Sc. degrees in Communication Engineering from Xidian University, Xian, Shaanxi, China, in 2003 and 2006, respectively, and the Ph.D. degree in Electrical Engineering from the University of Connecticut, Storrs, in 2012. He is now a Senior Engineer in the LSI Corporation, San Jose, CA. His research interests lie in the areas of signal processing for wireless and underwater acoustic communications, specifically implementation of multicarrier systems such as orthogonal frequency division multiplexing (OFDM), with focus on channel estimation and equalization, channel coding, iterative receiver design, and implementation complexity.

Wei Zhou received her M.S. degree in Electrical Engineering from the University of Connecticut, Storrs, in 2012. She is now with Marvell Semi-conductor at Santa Clara, CA.

Shengli Zhou (SM'11) received the B.S. degree in 1995 and the M.Sc. degree in 1998, from the University of Science and Technology of China (USTC), Hefei, both in Electrical Engineering and Information Science. He received his Ph.D. degree in Electrical Engineering from the University of Minnesota (UMN), Minneapolis, in 2002. He has been an Assistant Professor with the Department of Electrical and Computer Engineering at the University of Connecticut (UCONN), Storrs, 2003-2009, and now is the Charles H. Knapp Associate Professor in Electrical Engineering.

His general research interests lie in the areas of wireless communications and signal processing. His recent focus is on underwater acoustic communications and networking. He served as an Associate Editor for IEEE Transactions on Wireless Communications, from Feb. 2005 to Jan. 2007, and IEEE Transactions on Signal Processing, from Oct. 2008 to Oct. 2010. He is now an Associate Editor for IEEE Journal of Oceanic Engineering. He received the 2007 ONR Young Investigator Award and the 2007 Presidential Early Career Award for Scientists and Engineers (PECASE).