

On Systematic LT Codes

Xiaojun Yuan, *Student Member, IEEE*, and Li Ping, *Senior Member, IEEE*

Abstract—We propose a family of systematic rateless codes that are universally capacity-approaching on a binary erasure channel (BEC) regardless of the channel erasure rate. A significant property of the proposed codes is their linear encoding and decoding complexity, which is considerably lower than existing alternatives.

Index Terms—Systematic LT codes, Raptor codes.

I. INTRODUCTION

RATELESS codes can provide reliable and efficient information delivery without the *a priori* knowledge of channel state information at the transmitter, and have potential applications, e.g., in Automatic Repeat-reQuest (ARQ) and digital fountain systems [1]. LT codes [2] are a well-known family of rateless codes in binary erasure channels (BEC), and Raptor codes [3] are an enhanced version of LT codes with linear encoding and decoding complexity.

In many practical situations, systematic codes are preferred. For example, in a relatively good channel with rare erasure, both encoding and decoding processes for systematic codes can be avoided if no erasure occurs in a transmission block, which leads to a reduced cost. LT codes and raptor codes are, in their straightforward forms, non-systematic. A technique to design systematic Raptor codes was proposed in [3] with encoding complexity roughly αk^2 per bit, where k is the information length and α is a constant. With the technique described in [5], α can be made relatively small, but it is still burdensome when k is large.

In this letter, we propose a low-cost approach to the design of systematic LT codes by slightly modifying the LT code structure without seriously affecting performance, as well as by using the doping technique [4] to alleviate the error-floor problem without affecting the complexity order of the codes. We show that our proposed systematic LT codes have linear complexity, and that they are universally efficient regardless of the channel condition.

II. AN OUTLINE OF LT CODES

We begin with an outline of LT codes. Denote by G the graphic representation of an LT code. An example of G is shown in Fig. 1(a), where each black node represents a state bit s_i and each white node represents an output bit v_j . The encoding rule is given by

$$v_j = \sum_{s_i \text{ is connected to } v_j \text{ via an edge}} s_i \quad (1)$$

Manuscript received May 28, 2008. The associate editor coordinating the review of this letter and approving it for publication was G.-H. Im. This work was fully supported by a strategic research grant from City University of Hong Kong, China (Project No. 7002099).

The authors are with the Department of Electrical Engineering, City University of Hong Kong, HK SAR (e-mail: {xjyuan, eeliping}@cityu.edu.hk). Digital Object Identifier 10.1109/LCOMM.2008.080825.

where the summation is binary. In a conventional LT code, state bits simply represent input bits (that carry information); but they may be formed differently, as detailed later.

The degree of a node in G is defined as the number of edges connected to this node. The edges are randomly connected to the black nodes, and thus the left degrees (for black nodes) follow a Poisson distribution, denoted by $\Lambda(x) = \sum_i A_i x^i$, where i is the proportion of the black nodes with degree i . The right degree distribution (for white nodes) is specified by an optimized polynomial $P(x) = \sum_i P_i x^i$, where P_i is the proportion of the white nodes with degree i .

The white nodes in G are transmitted over an erasure channel. The receiver aims at recovering the black nodes from a partially reconstructed graph with some white nodes lost. The belief propagation (BP) algorithm can be used for this purpose as follows.

The BP Algorithm: Find a degree-1 white node v_j . Recover the unique black node s_i connected to v_j . Then delete v_j , s_i and the edges connected to s_i . Repeat the above process to the residual graph until all black nodes are recovered (decoding success) or until no degree-1 white nodes can be found in the non-empty residual graph (decoding failure).

A coding graph G is BP-decodable if the above process ends at "decoding success". The *overhead* is defined as the normalized difference between the number of the received bits and that of the input bits (normalized by the latter). The *complexity* is proportional to the number of edges in the coding graph, since each edge corresponds to a binary addition in encoding and decoding. Let n be the number of input bits. It is possible to design LT codes with complexity $O(n \log n)$ that can reliably recover all of the input bits at a small overhead [1]. However, LT codes with complexity $O(n)$ (referred to as *linear complexity*) suffer from an error-floor problem [3], i.e., some proportion of input bits cannot be recovered by BP decoding. For example, for LT codes with

$$P(x) = 0.008x + 0.494x^2 + 0.166x^3 + 0.073x^4 + 0.083x^5 + 0.056x^8 + 0.037x^9 + 0.056x^{19} + 0.025x^{65} + 0.003x^{66} \quad (2)$$

$n = 65536$ and overhead = 0.03, the proportion of unrecovered input bits after BP decoding ranges from 0.3% to 0.8% with high probability. We henceforth always focus on linear-complexity LT codes.

III. SYSTEMATIC LT CODES

A. The Basic Idea

The systematic LT coding structure is illustrated in Fig. 1(b) that has the same structure as the LT code in Fig. 1(a). The only difference is that in Fig. 1(b) the input bits form part of

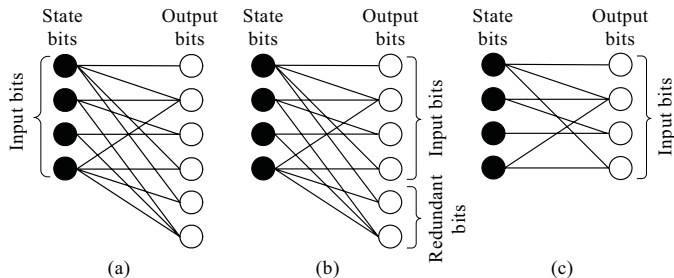


Fig. 1. (a) An LT coding graph G ; (b) the corresponding systematic LT coding graph G ; and (c) the sub-graph G^{ENC} .

the output bits. The other output bits are called redundant bits. We denote by G^{ENC} the sub-graph induced by the input bits (see Fig. 1(c)) and still by G the overall graph. The encoding process of systematic LT codes consists of two steps:

- 1) Determine state bits from input bits via LT decoding;
 - 2) Determine redundant bits from state bits via LT encoding.
- To ensure linear complexity, G^{ENC} should be BP-decodable, which is our focus below.

B. Number of State Bits

Let k be the number of input bits. We show that the optimum number of state bits (denoted by n) is $n = k$. To see this, first suppose that $k > n$. From (1), each output bit can be viewed as a linear equation. Step 1 of the systematic LT encoding process corresponds to solving n unknown variables from k linear equations. Since the values of input bits are arbitrary, the solution may not exist when $k > n$.

On the other hand, suppose that $k < n$. The codes in Figs. 1(a) and 1(b) have exactly the same decoding performance in terms of frame error rate (FER) since they share the same structure. Let m be the number of the received output bits to achieve a certain FER performance for the LT code in Fig. 1(a). The overhead of the LT code is $m/n - 1$. To achieve the same performance for the code in Fig. 1(b), the required overhead is $m/k - 1 (> m/n - 1)$. Thus, $k < n$ leads to inefficiency.

Therefore, the optimum choice is $n = k$, i.e., the numbers of the black and white nodes in G^{ENC} (see Fig. 1(c)) should be the same. This is always assumed in the following.

C. Construction of G^{ENC}

We now discuss the edge connections on G^{ENC} . We need to ensure that G^{ENC} is BP-decodable, as well as that the nodal degree distributions on G^{ENC} are the same as a typical LT code, i.e., $\Lambda(x)$ on the left and $P(x)$ on the right. A brute-force approach to constructing G^{ENC} is to repeatedly generate G^{ENC} using LT encoding and check whether it is BP-decodable. Such an attempt will fail with high probability, since the checking process is equivalent to LT decoding without overhead, i.e., decoding k state bits from k received output bits. Even if it succeeds, the left degree distribution may not match $P(x)$ well since we are selecting G^{ENC} from an event with a very small probability.

In [3], the authors suggested to construct G^{ENC} by identifying k input bits from a larger LT coding graph G' with

$k(1 + \epsilon)$ output bits, where ϵ is non-negative and referred to as encoding margin. Note that ϵ here is equivalent to the overhead of the LT code defined by G' . If BP decoding on G' is successful, the sub-graph induced by the k deleted white nodes forms a BP-decodable G^{ENC} . However, the error-floor problem for linear-complexity LT codes may cause difficulty in this approach. A quite large ϵ is required to ensure that G^{ENC} can be found after a reasonable number of trials, but the realized right degree distribution of G^{ENC} will severely deviate from $P(x)$ if ϵ is large. To see this, consider the coefficient P_1 in $P(x)$. Because of the nature of BP decoding, the degree-1 output bits of G' have a high probability of being selected in constructing G^{ENC} . The total number of degree-1 output bits in G' is $(1 + \epsilon)P_1k$, and so the proportion of degree-1 output bits in G^{ENC} is close to $(1 + \epsilon)P_1$. This can be quite different from the desired P_1 . Considerable changes also occur for other coefficients in $P(x)$ when ϵ is large.

Besides the above difficulty, the error-floor problem also prevents full recovery of input bits in decoding. The authors in [3] proposed a precoding technique to overcome the error-floor problem. However, the systematic encoding process may become much more complicated, since the precoding constraints must be taken into account in determining the positions of the input bits in the output.

IV. THE PROPOSED SOLUTION

We propose the following solution to overcome the error-floor problem while keeping the linear-complexity property.

A. Modifying the Structure of G'

We first follow the approach in Section III.C to generate G' with a small ϵ , and then modify G' to be BP-decodable as follows. Suppose that BP decoding on a copy of G' stops at a non-empty residual graph R (otherwise G' is already BP-decodable). We produce a degree-1 white node by modifying R , so that BP decoding can continue on R . This can be achieved by first randomly identifying a white node v_j (with residue degree d on R), and then deleting $d-1$ edges of v_j . G' is modified accordingly as follows: identify v_j on G' , delete the $d-1$ edges of v_j (corresponding to those deleted in R), and regenerate $d-1$ new edges by connecting v_j randomly to the recovered black nodes (i.e., those black nodes in G' but not in R). The reason for regenerating new edges is to ensure that the right degree distribution of G' is not affected in modification. The reason for connecting new edges to the recovered black nodes is that these edges can be deleted in BP decoding (as a result of recovering their connected black nodes), which guarantees that R is still a valid residual graph of G' after modification. The above process repeats until BP decoding on G' yields an empty R . Afterwards, we can obtain G^{ENC} from G' , as discussed in Section III.C.

The above graph modification does not affect $P(x)$ of G' . The deviation of $P(x)$ in G^{ENC} is also small when ϵ is small. Moreover, the deviation of $\Lambda(x)$ in G^{ENC} due to graph modification is determined by the error floor (since graph modification takes place only when BP decoding on G' fails). Properly designed LT codes can ensure a reasonably low error floor at a small overhead [3]. Thus, the distribution deviations in G^{ENC} resulted above are small.

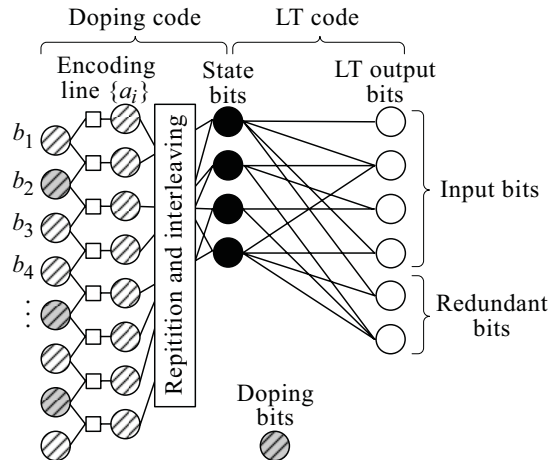


Fig. 2. An example of systematic LT codes with doping. $r = 2$, $k = 4$, and $p = 1/3$. Each small square represents a parity check.

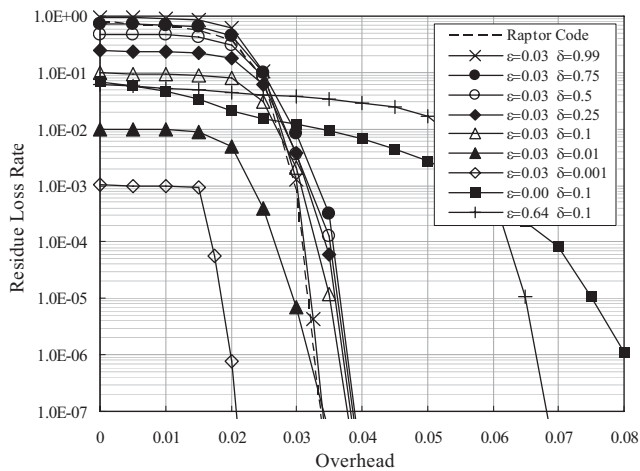


Fig. 3. The performance of the proposed codes at various δ and ϵ .

B. The Doping Technique

The proposed graph modification can solve the error-floor problem in encoding. We now employ the doping technique [4] to solve the error-floor problem in decoding. The overall coding structure is illustrated in Fig. 2. The codeword includes both LT output and doping bits that are uniformly mixed in transmission. Let p be the doping ratio, i.e., the fraction of doping bits in the codeword. We use a small p so that the doping code has a minor impact on the waterfall performance, while it can significantly relieve the error-floor problem.

The structure of the doping code is illustrated in the left-half of Fig. 2. A doping encoder is initialized as follows: repeat each of the k state bits by r times; randomly scramble the resulting rk bits to form an encoding line (with a_i denoting the i -th bit); apply accumulate addition to $\{a_i\}$, i.e., set $b_1 = a_1$ and calculate $b_i = a_i + b_{i-1}$ for $i = 2, \dots, rk$. In encoding, the doping encoder randomly selects bits in $\{b_j\}$ as doping bits. The doping code is actually a left-regular version of the SR-LDPC codes proposed in [6]. Due to this similarity, the decoding for the doping code is standard and thus omitted. SR-LDPC codes have linear encoding and decoding complexity, and so is the doping code. Also note that the selection of a

same b_j is allowed in generating the doping bits, and so the doping bits can be generated independently and limitlessly, i.e., the doping code is rateless.

The overall encoding process is scheduled as follows. The state bits are first determined using BP decoding on G^{ENC} ; then, the doping bits and the redundant LT output bits are generated (while maintaining a certain doping ratio p). In decoding, BP decoding is applied to the received LT output bits to recover the state bits; then, the residue errors are further reduced by decoding the doping code; finally, the lost input bits are recovered based on the state bits. Note that all of the above steps can be done in linear complexity.

V. SIMULATION RESULTS AND CONCLUSIONS

We demonstrate the performance of the proposed systematic LT codes using the distribution in (2) with $p = 0.015$, $r = 4$ and $k = 65536$. Each performance point is calculated based on the simulation of at least 2000 coding blocks.

Fig. 3 shows the performance of the proposed codes with various encoding margins ϵ and channel erasure rates δ . The performance of the raptor code is also included for comparison. We make the following observations.

- Fix $\delta = 0.1$ and consider the code performance as a function of ϵ . When ϵ is large (such as $\epsilon = 0.64$), the realized degree distribution of the input bits may significantly deviate from the designated $P(x)$. When ϵ is too small (such as $\epsilon = 0$), the modification of G' involves a large proportion of input bits. In both cases, the performance is relatively poor.
- The best performance is observed at around $\epsilon = 0.03$. In this case, the modification of G' involves no more than 0.8% input bits with high probability; and it leads to a performance degradation of no more than 0.005 in overhead (compared with the raptor code).
- The performance of the proposed code at $\delta = 0.99$ is very close to that of the raptor code, since after heavy loss in the channel the received code is almost “non-systematic”.
- For a very small ϵ , the required overhead for the proposed code reduces together with δ . This is as expected due to the systematic nature of the code.

To conclude, we have proposed a family of linear-complexity systematic LT codes, and shown by simulation that they are universally efficient regardless of channel erasure rate.

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 '98*, Vancouver, BC, Canada, Sept. 1998, pp. 56–67.
- [2] M. Luby, “LT-codes,” in *Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS)*, Vancouver, BC, Canada, Nov. 2002.
- [3] A. Shokrollahi, “Raptor codes,” *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.
- [4] X. Yuan and Li Ping, “Doped accumulate LT codes,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT) 2007*, Nice, France, June 2007, pp. 2001–2005.
- [5] 3GPP TSG-SA WG4 S4-AHP238, *Specification Text for Systematic Raptor Forward Error Correction*, PSM SWG, Sophia Antipolis, France, Apr. 2005.
- [6] L. Ping and R. Sun, “Simple erasure correcting codes with capacity approaching performance,” in *Proc. Global Telecommunications Conference, GLOBECOM '02*, Taiwan, China, Nov. 2002, pp. 1046–1050.