Traditional, Lengthened and Shortened LT Codes Comparison

Guilherme R. Colen, Weiler A. Finamore and Moisés V. Ribeiro

***Abstract*— Shortened and lengthened systematic Luby Transform Codes (LT-codes) have been shown, in this paper, to perform better than their equivalent mother codes by simulating transmission of LT-coded information over a binary erasures channel. The performance is improved as the shortening (lengthening) factor is increased while reducing (keeping) its complexity, when compared to the complexity of the mother code. The reported performance of both shortened and lengthened systematic LT-codes were obtained by designing the codes by using a truncated robust soliton degree distribution. Our results shows also that systematic LT-codes (despite the degree distribution) performs better than non-systematic LT-codes.**

***Keywords*— Shortened LT-codes, Lengthened LT-codes, Systematic LT-codes, Fountain Codes.**

I. Introduction

Luby-Transform Codes (LT-codes) have being proposed to protect data transmitted over channels that can be treated as a Binary Erasure Channel (BEC). A binary symbol transmitted over a BEC is delivered, at the channel output, either the same as the transmitted symbol or declared to be erased if not recognized as a 0 nor a 1. Many practical communication systems are well modeled by scenarios that incorporate the BEC.

As praised in many previous papers [1] LT-codes are low complexity codes which are very efficient. They are very efficient in that they transmit, with high probability of success, over a BEC with capacity C bits/channel-use, a block of k binary input symbols, by using binary-codewords with length n not much larger than $\frac{k}{C}$, the minimum number of bits required by Shannon's channel coding theorem. The longer the value of k, the higher the efficiency.

LT-codes can be viewed as linear block codes with codewords $\mathbf{c}_i$ of length n obtained by the multiplication of the corresponding information-word $\mathbf{u}_i$, of length k, by a $k \times n$ matrix $\mathbf{G}$ (code generating matrix). The LT-codes are designed by constructing a degree distribution which (as will be later explained) plays a fundamental role in the system performance. A procedure widely used to build new block codes is to introduce simple modifications to a good mother code [2] rendering a modified code with improved characteristics.

The analysis made, by simulating the transmission of encoded blocks trough a BEC channel, has shown that shortened and lengthened LT-codes performs much better than plain LT-codes. Our study has focused on systematic LT-codes.

The paper is organized as follows: Section II presents a description of both the conventional LT-codes. The shortened and lengthened LT-codes are presented in Section III. Simulation results are given in Section IV and the concluding remarks are in Section V.

The authors are with the Universidade Federal de Juiz de Fora. Financial support of FAPEMIG and CAPES is gratefully acknowledged.

II. LT-codes

We describe in this section the LT-encoder-decoder pair, together with some useful degree distributions. [1], [2], [4]. Some notations which will ease the discussion of the lengthened and shortened LT-codes are also introduced.

A. LT-encoder

Let us consider that a vector $\mathbf{u} = (u_1, u_2, \ldots, u_k)$ with k symbols is to be encoded. Both vector $\mathbf{u}$ and the vector $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ that appears at the encoder output have components belonging to the set $\{0,1\}$. The components c_j, $j = 1, \ldots, n$ are related to the input vector by

$$c_j = \mathbf{u} \cdot \mathbf{g}_j = \sum_{\ell=1}^{k} u_\ell \cdot g_{j,\ell} \tag{1}$$

where $\mathbf{g}_j = (g_{j,1}, g_{j,2}, \ldots, g_{j,k})$, are the column vectors of the $k \times n$ code generating matrix. Constructing the code generator matrix involves using the theory developed and explained in the paper by Luby [1] and extended in other papers [4]. This amounts, in the end, to establishing the Hamming weights $d_j = \omega(\mathbf{g}_j)$ of the column vectors and, to deciding which vector components (the matrix elements on column j) will be set to 1. The vector of weights of the column vectors, $\mathbf{d} = (d_1, d_2, \ldots, d_n)$ are to be chosen at random and, in this sense, can be viewed as the values observed when a sequence of equally distributed random variables $(D_1, D_2, \ldots, D_n)$, is sampled according to a finite support probability distribution specified by $P(D_j = i) = p_i,\ (j = 1, \ldots, n;\ i = 1, \ldots, k)$. Once the weight d_j of column vector $\mathbf{g}_j$ has been established, it remains to be decided which components will be set to one. This can be done by selecting, at random and with equal probability, the d_j input symbols which will be summed up to form the codeword component c_j.

A correspondence between the encoder described by 1 and a bipartite graph can now be easily introduced by associating every input symbol u_i with the input node α_i and, in the same vein, the codeword symbol c_j can be associated to an encoder output node β_j. This graph will have an edge e_{ij} connecting nodes α_i and β_j if the generating matrix element $g_{ij} = 1$.

A systematic LT-code will have generating matrix $\mathbf{G}_s = \left[\mathbf{I}_k \mid \mathbf{P}\right]$, where $\mathbf{I}_k$ is an identity matrix of size k and $\mathbf{P}$ is a $k \times (n-k)$ generator matrix, constructed by tacking the proper degree distribution similarly to what is done for matrix construction of non-systematic LT-codes — the choice of degree distributions for systematic codes will be fully explained in section II-C.

The encoding procedure for a fixed rate encoder with k input symbols and n output symbols is summarized by the following algorithm — this *LT*-encoder is said to operate at rate $R_{LT} = \frac{k}{n}$ or, in other words, with an expansion ratio $\rho_{LT} = \frac{n}{k}$.

LT-encoding Algorithm

1) *Initialize the encoder:*
   a) Specify the encoder degree sequence $\mathbf{d} = (d_1, d_2, \ldots, d_n)$.
   b) Specify the $\mathbf{d}$-matrix $\mathbf{G_d}$ by constructing the matrix column-vectors $\mathbf{g}_j = (g_{j,1}, g_{j,2}, \ldots, g_{j,k})$ according do the degree sequence $\mathbf{d}$.
   c) Set the encoder input to be the sequence $\mathbf{u} = (u_1, u_2, \ldots, u_k)$.
2) *Generate the transmitted-codeword:* find the encoder output $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ according to equation 1.

B. LT*-decoder*

We next describe the decoding procedure. This procedure works for non-systematics and systematics codes. Let us suppose that the codeword $\mathbf{c}$ has be transmitted and that $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, the BEC channel output, has been delivered to the receiver. Since the erased symbols are of no use the first decoding step is to identify the position of the erased symbols and, after purging these symbols, to adjust, accordingly, the degree sequence to be used by the decoder. We are considering that synchronism issues have being handled and that the decoder degree vector $\widetilde{\mathbf{d}}_1 = (\widetilde{d}_{1,1}, \widetilde{d}_{1,2}, \ldots, \widetilde{d}_{1,n'})$ are the degrees of the $n' \leq n$ non-erased received symbols $\widetilde{\mathbf{c}}_1 = (\widetilde{c}_{1,1}, \widetilde{c}_{1,2}, \ldots, \widetilde{c}_{1,n'})$ (it should be noticed that $\widetilde{\mathbf{c}}_1$ corresponds to the codeword that would have being transmitted were $\widetilde{\mathbf{d}}_1$ the encoder degree vector).

The decoder task is to estimate the transmitted information $\mathbf{u}$. To begin with, at Step 1, the decoder input is the vector, of dimension n', $\widetilde{\mathbf{c}}_1$ with its associated degree vector $\widetilde{\mathbf{d}}_1$. To begin with, the estimated transmitted information (decoder output), $\widetilde{\mathbf{u}}_1 = (\widetilde{u}_{1,1}, \widetilde{u}_{1,2}, \ldots, \widetilde{u}_{1,k})$, is set to be a sequence of erasures, i.e., for all $\ell = 1, \ldots, k$ the estimated symbol values are set to erasures — $\widetilde{u}_{1,\ell} = E$.

The decoding procedure starts with vectors $\left(\widetilde{\mathbf{c}}_1, \widetilde{\mathbf{d}}_1, \widetilde{\mathbf{u}}_1\right)$ and produces a sequence of vectors $\{\left(\widetilde{\mathbf{c}}_1, \widetilde{\mathbf{d}}_1, \widetilde{\mathbf{u}}_1\right), \left(\widetilde{\mathbf{c}}_2, \widetilde{\mathbf{d}}_2, \widetilde{\mathbf{u}}_2\right), \ldots, \left(\widetilde{\mathbf{c}}_m, \widetilde{\mathbf{d}}_m, \widetilde{\mathbf{u}}_m\right)\}$ by successively transforming the triple of vectors $\left(\widetilde{\mathbf{c}}_m, \widetilde{\mathbf{d}}_m, \widetilde{\mathbf{u}}_m\right)$ and associated $\mathbf{d}_m$-matrix $\mathbf{G}_{\mathbf{d}_m}$ into the triple $\left(\widetilde{\mathbf{c}}_{m+1}, \widetilde{\mathbf{d}}_{m+1}, \widetilde{\mathbf{u}}_{m+1}\right)$ with associated $\mathbf{d}_{m+1}$-matrix $\mathbf{G}_{\mathbf{d}_{m+1}}$. To define this transformation which will be named *graph reduction* let us recall that to every triple $\left(\widetilde{\mathbf{c}}_m, \widetilde{\mathbf{d}}_m, \widetilde{\mathbf{u}}_m\right)$ there is graph $\mathcal{G}_m$ (uniquely) associated with it.

Let the triple $\left(\widetilde{\mathbf{c}}_m, \widetilde{\mathbf{d}}_m, \widetilde{\mathbf{u}}_m\right)$ and associated graph be $\mathcal{G}_m = (\mathcal{A}_m, \mathcal{B}_m, \mathcal{E}_m)$. The set $\mathcal{B}_m = \{\beta_{m,j_1}, \beta_{m,j_2}, \ldots, \beta_{m,J_m}\}$, with cardinality J_m, is the set of vertices associated to vector $\widetilde{\mathbf{c}}_m$ — these will be called β-nodes or β-vertices. The α-nodes set, $\mathcal{A}_m = \{\alpha_{m,i_1}, \alpha_{m,i_2}, \ldots, \alpha_{m,I_m}\}$, with cardinality I_m, is the set of vertices, associated to vector $\widetilde{\mathbf{u}}_m$. $\mathcal{E}_m$, with cardinality $\sum_{x=1}^{J_m} \widetilde{d}_{m,x}$, is the set with edges $(\beta_{m,j}, \alpha_{m,i})$ where (j, i) are positions in matrix $\mathbf{G}_{\widetilde{\mathbf{d}}_m}$ corresponding to $g_{m,j,i} = 1$.

Definition (Graph reduction).
We say that a graph $\mathcal{G}_m$ is reducible if there is one component of vector $\widetilde{\mathbf{d}}_m$, which is equal to 1. The reduction takes place by first identifying the degree 1 vertex $\beta_{m,j'}$, with associated value $\widetilde{c}_{m,j'}$, with smallest index j'. Let the edge connected to this node be $(\beta_{m,j'}, \alpha_{m,i'})$. The reduction transformation proceeds by constructing the vector $\widetilde{\mathbf{u}}_{m+1}$ with the same values attached to vector $\widetilde{\mathbf{u}}_m$ except for the value attributed to vertex $\widetilde{u}_{m+1,i'}$ (an erasure) which is replaced by the value $\widehat{c}_{m,j'}$. The subset of the β-nodes which are neighbors to $\alpha_{m,i'}$, the newly valued α-vertices, namely, $\mathcal{N}_{m,i'} = \{\beta_{m,j'_1}, \beta_{m,j'_2}, \ldots, \beta_{m,J'_m}\}$, are next identified and a vector $\widetilde{\mathbf{c}}_{m+1}$ is created which has the same values attached to vector $\widetilde{\mathbf{c}}_m$ except for the components $\{c_{m,j'_1}, c_{m,j'_2}, \ldots, c_{m,J'_m}\}$ which are to be replaced by $\{c_{m,j'_1} \oplus \widehat{c}_{m,j'}, c_{m,j'_2} \oplus \widehat{c}_{m,j'}, \ldots, c_{m,J'_m} \oplus \widehat{c}_{m,j'}\}$. Finally, the reduced graph $\mathcal{G}_{m+1}$ is the graph that remains when suppressing the α-vertices $\alpha_{m,i'}$ and the β-vertices belonging to $\mathcal{N}_{m,i'}$ (and, of course, the corresponding edges). ◊

The decoding procedure is summarized next.

LT-decoding Algorithm

1) *Initial recursion decoding step*
   a) Initialize the the recursion step counter: $m = 1$;
   b) Set the initial recursion decoder degree sequence to $\widetilde{\mathbf{d}}_m = (\widetilde{d}_{1,1}, \widetilde{d}_{1,2}, \ldots, \widetilde{d}_{1,n'})$;
   c) Specify the $\widetilde{\mathbf{d}}_1$-matrix $\mathbf{G}_{\widetilde{\mathbf{d}}_1}$ by constructing the matrix column-vectors $\mathbf{g}_{m,j} = (g_{1,j,1}, g_{1,j,2}, \ldots, g_{1,j,k})$ according do the degree sequence $\widetilde{\mathbf{d}}_1$;
   d) Set the initial recursion decoder input to be the sequence $\widetilde{\mathbf{c}}_m = (\widetilde{c}_{1,1}, \widetilde{c}_{1,2}, \ldots, \widetilde{c}_{1,n'})$;
   e) Set the initial recursion decoder output to be the sequence $\widetilde{\mathbf{u}}_m = (\widetilde{u}_{1,1}, \widetilde{u}_{1,2}, \ldots, \widetilde{u}_{1,k})$ with $\widetilde{u}_{1,\ell} = E$ for all $\ell = 1, 2, \ldots, k$;
   f) Set $\mathcal{G}_1 = (\mathcal{A}_m, \mathcal{B}_m, \mathcal{E}_m)$ — $\mathcal{G}_1$ is the initial decoding-graph where $\mathcal{B}_m = \{\beta_{m,j_1}, \beta_{m,j_2}, \ldots, \beta_{m,J_m}\}$ is a cardinality J_m set of vertices associated to vector $\widetilde{\mathbf{c}}_1$, $\mathcal{A}_m = \{\alpha_{m,i_1}, \alpha_{m,i_2}, \ldots, \alpha_{m,I_m}\}$ is a cardinality I_m set of vertices, associated to vector $\widetilde{\mathbf{u}}_1$, and $\mathcal{E}_1$ is a cardinality $\sum_{x=1}^{j_1} \widetilde{d}_{1,x}$ set with edges $(\beta_{m,j}, \alpha_{m,i})$ where (j, i) are positions of matrix $\mathbf{G}_{\mathbf{d}_1}$ corresponding to $g_{m,j,i} = 1$.
2) If graph $\mathcal{G}_m$ is irreducible, go to final step (Step 5) — a decoding failure has occurred;
3) Make a graph reduction by setting $\mathcal{G}_{m+1} = (\mathcal{A}_{m+1}, \mathcal{B}_{m+1}, \mathcal{E}_{m+1})$ — $\mathcal{G}_{m+1}$ is the reduced graph with associated the triple $\left(\widetilde{\mathbf{d}}_{m+1}, \widetilde{\mathbf{c}}_{m+1}, \widetilde{\mathbf{u}}_{m+1}\right)$;
4) If there is a symbol $\widetilde{u}_{m+1,\ell} = E$ for some $\ell = 1, 2, \ldots, k$, increment the recursion counter ($m = m+1$) and return to Step 2;
5) Stop — the estimated transmitted sequence is given by

$$\widehat{\mathbf{u}} = \widetilde{\mathbf{u}}_{m+1}.$$

C. Degree Distributions

LT codes do work properly only if a good degree distribution is designed. This section describes thus some relevant degree distributions that can be found in the literature. The first degree distribution, the Ideal Soliton Degree Distribution (ISdd), introduced in [1], is expressed by

$$\rho(d) = \begin{cases} 1/k, & \text{for } d = 1, \\ \frac{1}{d(d-1)}, & \text{for } d = 2,\, 3,\, ...,\, k, \end{cases} \tag{2}$$

where $\rho(d)$ is the probability that a given node has degree d. The ISdd distribution has however some drawbacks, as pointed out by Luby who proposed, in [1], a more practical distribution, namely the Robust Soliton Degree Distribution (RSdd). In order to specicy the RSdd, let us first introduce the function

$$\tau(d) = \begin{cases} \frac{R}{d \cdot k}, & \text{for } d = 1,\, 2,\, ...,\, \frac{k}{R} - 1, \\ \frac{R \cdot \ln(R/\delta)}{k}, & \text{for } d = \frac{k}{R}, \\ 0, & \text{for } d = \frac{k}{R} + 1,\, ...,\, k, \end{cases} \tag{3}$$

in which δ is the probability of LT decoding failure and, for some suitable constant $c > 0$, $R = c \ln(k/\delta)\sqrt{k}$. The RSdd is then defined as

$$\mu(d) = \frac{\rho(d) + \tau(d)}{\sum_{d=1}^{k} \left(\rho(d) + \tau(d)\right)}. \tag{4}$$

Further improvement were obtained by Tee at all [3] who proposed the Improved Robust Soliton Degree Distribution (IRSdd). To describe the IRSdd let us introduce the set $\mathcal{D} = \{d_i \mid k \cdot \mu(d_i) < 1\}$. We thus have

$$\mu'(d) = \begin{cases} \mu(d) + \nu & \text{for } d = 1, \\ \mu(d) & \text{for } d \notin \mathcal{D}, \\ 0 & \text{for } d \in \mathcal{D}, \end{cases} \tag{5}$$

where $\nu = \sum_{d \in \mathcal{D}} \mu(d)$.

The distributions previously described are devised to design non-systematic codes. Systematic LT codes can be built by using the Truncated Robust Soliton Degree Distribution (TRSdd) proposed in [4] defined by

$$\Omega(d) = \begin{cases} \frac{1}{\beta'}\left[1 + \frac{R}{k} + \nu\right], & \text{for } d = \gamma, \\ \frac{\gamma}{\beta'}\left[\frac{1}{d(\frac{d}{\gamma} - 1)} + \frac{R}{d \cdot k}\right], & \text{for } d = 2\gamma,\, ...,\, \frac{k \cdot \gamma}{R} - \gamma, \\ \frac{R}{\beta' \cdot k}\left[\ln\left(\frac{R}{\delta}\right) + \frac{1}{\frac{k}{R} - 1}\right], & \text{for } d = \frac{k \cdot \gamma}{R}, \\ 0, & \text{for otherwise}, \end{cases} \tag{6}$$

where γ is an integer number such that $1 \leq \gamma \leq R$ and $\beta' = \sum_d \left[\rho(d) + \tau(d)\right] + \nu$.

III. Modified LT Codes

In this section, we describe two systematic LT-codes modifications named the Lengthened LT-Codes and Shortened LT-Codes. These are well known code modifications which consists in either augmenting the length of the information sequence by concatenating to it a block of known symbols or substituting a block of symbols from the the information sequence by a known block of symbols. The effect of these procedures is to improve the LT-codes performance.

A. Lengthened LT Code (LLT-code)

This modification is achieved by pre-appending a block of ℓ_L known bits to the original information sequence $\mathbf{u}$. Considering that $\mathbf{u}_0$ is the length ℓ_L block of known symbols the resulting block is $\mathbf{u}_L = (\mathbf{u}_0,\, \mathbf{u})$. Our simulation results show no noticeable change of performance if the vector of known symbols is a random vector or a vector of all ones or all zeros. The systematic LT-code is now designed to encode a length $\ell_L + k$ augmented input vector i.e., has generating matrix $\mathbf{G}_{s,L} = [\,\mathbf{I}_{k+\ell_L} \mid \mathbf{P}_L\,]$, of size $\ell_L + k \times n + \ell_L$. It should be noticed that, at the encoder output, the codeword is $\mathbf{c}_L = (\mathbf{u}_0,\, \mathbf{c})$, yet only the part $\mathbf{c}$ is to be transmitted (since $\mathbf{u}_0$ is meant to be known by the both the encoder and decoder). At the receiver, the concatenation of the received information $\mathbf{v}$ and the block $\mathbf{u}_0$ of known symbols is fed to the decoder.

The LLT-code, has rate $R_{\text{LLT}} = \frac{k}{n}$, equal to the rate of the mother code (the original code from which the LLT-code has been derived). We will refer to the ratio of ℓ_L and k as the lengthening factor $\lambda_L = \frac{\ell_L}{k}$. Notice that complexity of the LLT-codes can remain the same as that of the plain systematic LT-codes if one takes $\mathbf{u}_0 = \mathbf{0}$.

B. Shortened LT-Code (sLT-code)

Shortened LT-Codes represent a code modification with smaller the encoding and decoding complexity as compared to the complexity of an equivalent LT-code of same rate, if $\mathbf{u}_0 = \mathbf{0}$. It forces ℓ_s of the k input symbols to have known values and doesn't increase the number of input symbols. We will indicate the new input vector by the notation $\mathbf{u}_s = (\mathbf{u}_0,\, \mathbf{u})$ and let $\ell_s = |\mathbf{u}_0|$ be its length. The information vector $\mathbf{u}$ thus has length $k_s = k - \ell_s$ and the transmitted vector $\mathbf{c}$ has length $n_s = n - \ell_s$. The same matrix $\mathbf{G}_s$ will be used to construct the encoder and decoder. One thus have

$$\begin{aligned} R_{sLT} &= \frac{k_s}{n_s} \\ &= \frac{k - \ell_s}{n - \ell_s}. \end{aligned} \tag{7}$$

and $R_{sLT} < R_{LT}$.

When comparing sLT-codes and LT-codes on an equal rate basis the encoder has to output vectors of size $n_s = \frac{k_s \cdot n}{k}$. The mother-code generating matrix will be $\mathbf{G}_s$ and the new generator matrix to encode the $k_s = k - \ell_s$ input block is $\mathbf{G}_{s,s} = [\,\mathbf{I}_{k_s} \mid \mathbf{P}_s\,]$, where, it should be noticed, $\mathbf{G}_{s,s}$ is a $k_s \times n_s$ systematic generator matrix. The sLT-code, with k

input symbols and n output symbols, operates thus with a shortening factor $\lambda_s = \frac{\ell_s}{k}$. We thus have $n_s = n(1 - \lambda_s)$.

IV. Results

In this section results obtained via computer simulations are presented and discussed. Simulations were run by transmitting the LT-encoder output trough a BEC. To design the encoders we have used, in all cases, the parameters c and δ equal to 0.1 and 0.02, respectively.

We start by examining the influence on the performance of the LT-code code when the degree distributions discussed on Section II-C are selected to implement the code. Systematic as well as non-systematic codes were examined. We next compare the performance of LT-codes with the performance of modified codes, namely, LLT-codes and sLT-codes — the degree distribution rendering the best LT-code performance was selected in all cases.

The performance of LT-codes for several degree distributions are displayed in Fig. 1-3. In Fig. 1, where the performance parameter is the Bit Erase Rate (BErR) the results obtained for fixed-rate, systematic and non-systematic, LT-codes with $k = 500$ and $n = 743$ are presented — four degree distributions, namely ISdd, RSdd, IRSdd and TRSdd were examined. For the TRSdd we have chosen the value $\gamma = \lfloor R \rfloor = \min\{x \in \mathbb{Z} | x \leq R\}$. Observing Fig. 1, one realize that IRSdd is the best degree distribution for non-systematic codes. But, the best Degree Distribution for systematic codes is TRSdd. We also see, moreover, considering all the examined degree distributions, that systematic LT-codes have a better BErR performance than non-systematics LT-codes. Another interesting observation is that the LT-codes designed under the TRSdd are worthless. In Fig. 2 the Failure Rate (FR) of the examined LT-codes, again with $k = 500$ and $n = 743$, is exhibited. Similarly to what has been observed under the BErR-performance, Systematic LT-codes, constructed with a TRSdd, yields the best FR-performance when compared to its counterpart and performs better then the best non-systematic LT-code (constructed with the IRSdd). Comparisons of LT-codes operating in rateless mode — i.e., codeword symbols are transmitted until all information symbols are correctly recovered — can be made by observing the plots on Fig. 3. These plots illustrates the redundancy (versus the BEC quality) required in order to recover all k symbols correctly. It can be seeing that the systematic LT-codes constructed under the TRSdd will require the smaller redundancy on a good quality channel (P_e smaller 0.15). When non-systematic LT-codes are considered the IRSdd is the best independently of the BEC quality. Important to highlight, in this figure not appear the TRSdd for non-systematics code because this distribution does't work for non-systematics codes.

The main contribution of this work is the analysis of the performance of lengthened and shortened systematic LT-codes. This comparison is showed in figures 4 to 6. To every simulation shown in these figures we have use $\gamma = \lfloor R \rfloor$. Fig. 4 presents the BErR versus Channel Quality for conventional, lengthened and shortened systematic LT-codes built under TRSdd.

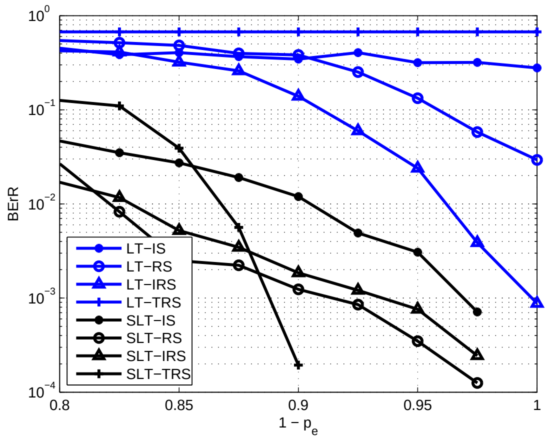

Fig. 1. BErR versus Channel Quality for systematic and non-systematic LT-codes with ISdd, RSdd, IRSdd and TRSdd degree distributions.

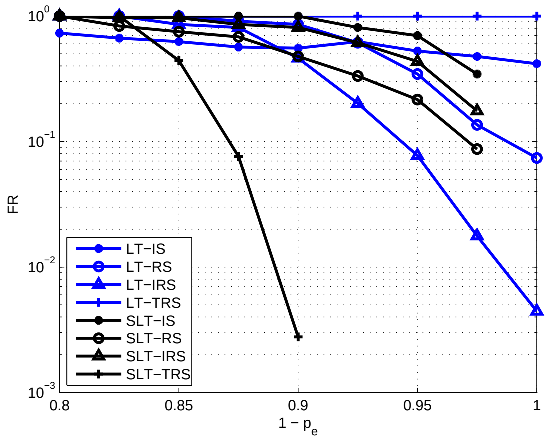

Fig. 2. FR versus Channel Quality for systematic and non-systematic LT-codes with ISdd, RSdd, IRSdd and TRSdd degree distributions.

Fig. 4 shows that when ℓ_L or ℓ_s increase the code's performance improves. It also shows that lengthened or shortened codes is better than conventional LT-codes. Moreover, shortening proves to be better than lengthening, in this situation.

Fig. 5 displays FR versus Channel quality for the same scenarios as that on Fig. 4. Again, at equal rates, shortened codes performs better than lengthened codes and improved their performances as ℓ_L or ℓ_s increase.

Finally, Fig. 6 shows the redundancy, versus channel quality, required to ensure that all k symbols are correctly recovered. In this figure, we see that for channels with $p_e \geq 0.9$ all codes performance are about the same. Otherwise, he best performance is obtained with shortened codes. For example, at $p_e = 0.8$ LT-codes require that $2.55k$ symbols be received before all symbols are recovered. Lengthened LT-codes, with $\lambda_L = 0.3$, requires $2.075k$ while shortened LT-codes, with $\lambda_s = 0.3$, requires only $1.631k$ symbols to be received.

V. Conclusions

Shortened and lengthened LT-codes have been investigated in this paper. The most relevant observation, is that the shortened codes performs better than their counterparts either

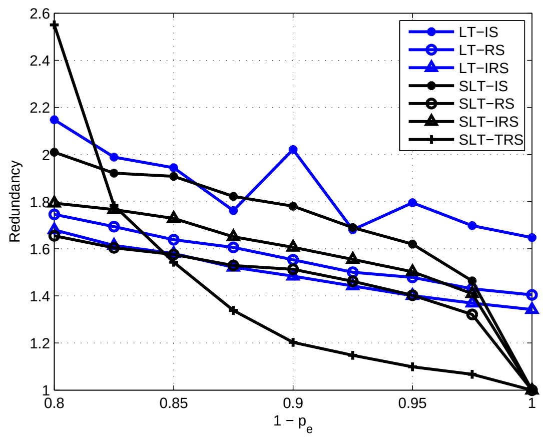

Fig. 3. Redundancy versus Channel Quality for systematic and non-systematic LT-codes with ISdd, RSdd, IRSdd and TRSdd degree distributions.

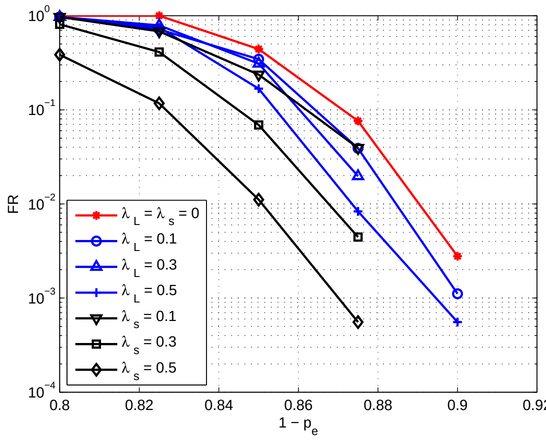

Fig. 5. FR versus Channel Quality for tradicional, lengthened and shortened systematic LT-codes with TRSdd degree distributions.

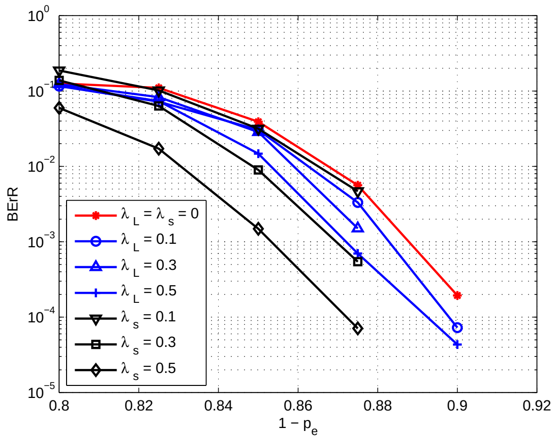

Fig. 4. BErR versus Channel Quality for tradicional, lengthened and shortened systematic LT-codes with TRSdd degree distributions.

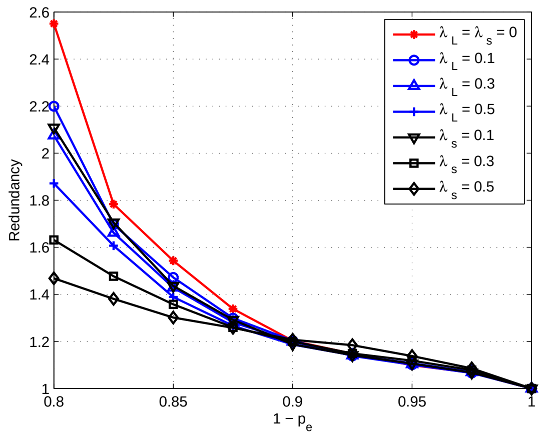

Fig. 6. Redundancy versus Channel Quality for tradicional, lengthened and shortened systematic LT-codes with TRSdd degree distributions.

in terms of redundancy, failure rate or bit erasure rate. This performance is much better when operating on a poor channel ($p_e \leq 0.9$) — which may be advantageous when the channel quality has a wide range variation. Shortened and lengthened LT-codes have their performance improved as both the shortening rate and the lengthening rate are, respectively, increased. It is worth mentioning that the encoding and decoding of a shortened LT-code requires a smaller complexity when compared to the complexity of the mother code if the information bits forced to be know are all zeros (otherwise the complexity is kept the same). Similarly the encoding and decoding of a lengthened LT-code keeps the complexity the same as that of the mother code if the information bits introduced for the elongation are all zeros (otherwise the complexity will increase). The reported performance of both shortened and lengthened systematic LT-codes were obtained by designing the codes with the TRSdd since this has been the degree distribution that resulted on the best performance for the mother codes. Non-systematic codes performs better when the IRSdd is selected. It should be further emphasized that systematic LT-codes (despite the degree distribution) performs better than non-systematic LT-codes. We conclude thus that modifying LT-code are a good implementation alternative when transmitting information over a binary erasures channel if the code is to be used in unicast mode. If multicast transmission is needed further investigation is required. We are currently investigating the use of modified LT-Codes when used on channels other than the BEC.

Acknowledgements

This work was supported in part by CNPq, CAPES, FAPEMIG, FINEP, P&D ANEEL-CEMIG, INERGE, and Smarti9 Ltda.

References

[1] M. Luby, *LT-codes*, in Proceedings of 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), Vancouver, Canada, 2002.

[2] R. E. Blahut, *Algebraic Codes for Data Transmission*, Cambridge, England: Cambridge University Press, 2003.

[3] R. Tee, T. Nguyen, L. Yang, and L. Hanzo *Serially Concatenated Luby Transform Coding and Bit-Interleaved Coded Modulation Using Iterative Decoding for the Wireless Internet* Proceedings of the VTC 2006 Spring, Melbourne, Australia, vol. 138, no.4, pp. 177-182, May 2006.

[4] T. Nguyen, L. Yang, S. X. Ng and L. Hanzo *An Optimal Degree Distribution Desin mand a Conditional Random Integer Generator for the Systematic Luby Transform Coded Wireless Internet* International IEEE Wireleess Communication & Network Conference(WCNC), Las Vegas, USA, March 31 - April 03, 2008.