OPTIMIZATION OF GENERALIZED LT CODES FOR PROGRESSIVE IMAGE TRANSFER

Suayb S. Arslan, Pamela C. Cosman and Laurence B. Milstein

Dept. Electrical and Computer Engineering, University of California, San Diego

ABSTRACT

Rateless codes allow a user to incrementally send additional redundancy, so they can be useful for heterogeneous and time-varying networks for which the choice of redundancy level in advance is difficult. Rateless codes are an attractive application layer forward error correction solution due to their flexibility and capacity-approaching performance. The original rateless codes were developed for the delivery of equally important information. In many multimedia applications, some data symbols are more important than others. Unequal error protection (UEP) designs are attractive solutions for such transmissions. However previous UEP rateless code designs were aimed for coarsely layered sources and might exhibit poor performance for fine-grained progressive coding. The main objective of this paper is to introduce a more generalized coding scheme, parameters of which can be tailored for progressive multimedia transmission. We present the optimization of a generalized rateless code using two different progressive source transmission protocols. Proposed coding scheme is shown to exhibit better unequal protection and recovery time properties than other published results.

Index Terms— Progressive sources, iterative decoding, rateless codes, unequal error protection, unequal recovery and iteration time.

1. INTRODUCTION

In internet communications, packets are either received reliably or lost completely due to channel impairments, network congestion and excessive delays. Although some types of multimedia data can tolerate channel residual errors to some extent, data downloads that include executable files generally require error free transmission. Data files sent over the internet are often chopped into fixed or variable size packets, and each packet is either received without error or corrupted and therefore erased during the transmission. For example, a cyclic redundancy check (CRC) code is typically used to detect packet errors. When there is an error in the packet, the receiver discards the whole packet [1].

This work was supported in part by LG Electronics Inc., the National Science Foundation under Grant CCF-0915727, the Center for Wireless Communications at the University of California, San Diego, and the UC Discovery Grant Program of the State of California.

One way of solving the image transmission problem for erasure channels is to use forward error correction (FEC) with a judiciously chosen code rate during the transmission. Yet, this results in an inefficient use of network resources, as the receivers with good channel conditions will experience unnecessary overhead, and/or the receivers with bad channel conditions will not be able to receive any meaningful information. Alternatively, systems can employ a feedback mechanism from the receiver to the sender to manage the retransmission of erased packets. Such protocols might be burdened with an excessive number of feedback messages and retransmissions. Rateless codes, such as Luby Tranform (LT) codes [2], do not assume any information about the channel and therefore are suitable matches for transmitting data over the internet where the channels have varying or unknown parameters. Similarly, such codes can be good candidates for multicast transmissions, because there is no prior assumption about the channels.

In multimedia applications, a portion of the data is generally more important than the rest. In particular, initial parts of a progressive bit stream are more important for the reconstruction of the image than are later parts [3]. In other internet browsing applications, timely recovery of the more important sections of source files can save us from unnecessary transmissions. This is because it might be enough for the end user to see the low resolution image or video before requesting further transmission. Such examples show that codes having unequal error protection (UEP) and unequal recovery time (URT) properties might be useful.

In the original study of LT codes, equal error protection (EEP) of the information symbols is assumed. The degree of a coded symbol d is defined to be the number of information symbols used to compute the value of the coded symbol. It is assigned according to a degree distribution (DD). After choosing the degree for each coded symbol, a d-element subset of the information block is chosen randomly according to a selection distribution (SD). It is shown in a series of studies [4]–[6] that UEP LT codes can be produced simply by allowing coded symbols to make more edge connections with more important parts of the information bit stream with high probability by modifying either one of the above two distributions. This way, the unrecovered symbol probability becomes lower for the high priority content of the original source. The ideas presented in those studies are successfully applied to various transmission scenarios [7].

In this paper, we present a generalized approach for designing rateless codes based on a systematic degree-dependent selection idea. The proposed design leads to a joint optimization of degree and selection distributions with an increased–size parameter set. However, for practical reasons, we reduce the number of parameters of the system subject to optimization. In addition, we present two different progressive source transmission protocols using the generalized code. We tailor the parameters of the proposed design in order to minimize the expected distortion. We show that, although the previous UEP schemes described in [4]–[6] can be used in conjunction with proposed protocols to provide unequal protection for progressive transmission, the proposed generalization of the rateless codes and its configuration for progressive transmissions give increased improvements in terms of end-to-end expected distortion. Furthermore, we evaluate the system performance as a function of the iteration index of the decoding algorithm. We argue that an unequal iteration time (UIT) property might be particularly important for portable devices which are constrained by low-complexity receiver architectures.

2. PRELIMINARIES

2.1. Progressive Source Coding

In progressive image coding, all encodings of the source at lower rates are embedded in the beginning of the source bit stream at higher rates. Thus, the importance of coded bits decreases with each successive bit. Moreover, bits later in the bit stream are of no use unless the bits that precede them are reliably received. Progressive transmission is useful in multimedia streaming applications over the internet, particularly for fast browsing of high definition multimedia in non-homogeneous networks. However, progressive encoders are often highly susceptible to noisy channel effects [3]. Therefore, a good protection mechanism is needed for the reliable transfer of the compressed data.

2.2. LT Codes

In LT encoding [2], coded symbols are generated from a set of k information symbols. Using a binary information block $\mathbf{x}^T = (x_1, x_2, \ldots, x_k) \in \mathbb{F}_2^k$, the degree of the mth coded symbol y_m, denoted d_m, is chosen according to a suitable DD $\Upsilon(x) = \sum_{\ell=1}^{k} \Upsilon_\ell x^\ell$, where Υ_ℓ is the probability of choosing degree $\ell \in \{1, \ldots, k\}$. After choosing the degree $d_m \in \{1, \ldots, k\}$, a d_m-element subset of $\mathbf{x}$ is chosen randomly according to a SD. In standard LT encoding, this corresponds to generating a random vector $\mathbf{w}_m$ of length k, and $weight(\mathbf{w}_m) = d_m$ positions are selected from a uniform distribution to be logical 1, without replacement. The coded symbol is given by $y_m = \mathbf{w}_m^T \mathbf{x}$ (mod 2). At the receiver, the belief propagation algorithm (BPA) is used to reduce decoding complexity, which is important with increasing block length.

Luby [2] proposed the *Robust Soliton* distribution (RSD) which performs satisfactorily in practice. Using RSD, at each iteration of the decoding algorithm, the number of expected decoded symbols is $R = c \cdot \ln(k/\delta)\sqrt{k} \geq 1$ for some suitable constant $c > 0$ and an allowable failure probability δ of the decoder.

3. GENERALIZATION OF UEP LT CODES

In this section, we apply the degree-dependent selection idea to the original LT coding in order to provide increased UEP, URT and UIT properties.

3.1. UEP Generalized LT (UEP GLT) Coding

We partition the information block into r variable size disjoint sets $s_1, s_2, \ldots, s_r$ (s_j has size $\alpha_j k, j = 1, \ldots, r$, such that $\sum_{j=1}^{r} \alpha_j = 1$, where $\alpha_j k$ are integers). In the encoding process, after choosing the degree number for each coded symbol, we select the edge connections according to a *Generalized* SD given by

Definition 1: (*Generalized* SD) For $i = 1, \ldots, k$: $P_i(x) = \sum_{j=1}^{r} p_{j,i} x^j$, where $p_{j,i} \geq 0$ is the conditional probability of choosing the information set s_j, given that the degree of the coded symbol is i, and $\sum_{j=1}^{r} p_{j,i} = 1$.

Note that $p_{j,i}$ are design parameters of the system, subject to optimization. Since $\sum_{j=1}^{r} p_{j,i} = 1$, the number of design parameters subject to optimization is $(r-1) \times k$. The design steps taken to generate each coded symbol are summarized in **Algorithm 1**. It is easy to see that [4] is a special case of the proposed UEP GLT coding. Since encoding-decoding is done according to two interrelated distributions, the design criterion in our case is to select both distributions judiciously to minimize the average distortion. To reduce the number of optimization parameters, we choose $p_{j,i}$ to be an exponential function of the degree number i for $j = 1, 2, \ldots, r-1$ as follows:

Definition 2: (*Exponential* SD) $p_{j,i} = A_j + B_j \times \exp\{-(i-1)/C_j\}$ for $i = 1, 2, \ldots, k$ where $\{A_j \geq 0, B_j \geq 0, C_j \geq 0\}_{j=1}^{r-1}$ are design parameters satisfying $\sum_{j=1}^{r} p_{j,i} = 1$ for all i.

Since the low degree check nodes are expected to make, on average, more edge connections with the more important information sets, the exponential SD is selected. Note that we reduce the parameter space size from $(r-1)k+k-1$ to $3(r-1)+k-1$. If we use standard DDs (for example, the RSD) and a predetermined partitioning set $\{\alpha_1, \ldots, \alpha_r\}$ in conjunction with an exponential SD, we will have only $3(r-1)$ parameters subject to optimization. Note that an additional optimization can be run over the partitioning set $\{\alpha_1, \dots, \alpha_r\}$. In that case, the parameter space size increases to $3(r-1)+r-1 = 4(r-1)$ since $\sum_{j=1}^{r} \alpha_j = 1$. As will be shown in the numerical results section, this will lead to a slightly better performance at the expense of increased complexity.

```
Algorithm 1: UEP GLT Encoding
for $m = 1, \dots, n$,
    1. Choose a degree $d_m \in \{1, \dots, k\}$ according to $\Upsilon_w(x)$
    2. Initialize c_deg = 1, c_edge[r] = 0[a].
    3. while c_deg $\leq d_m$
        • Choose $j \in \{1, \dots, r\}$ according to the Generalized SD $\{p_{1,d_m}, p_{2,d_m}, \dots, p_{r,d_m}\}$[b]
        • if count_edge[j] $< \alpha_j k$
            • Choose a symbol from $s_j$ uniform randomly [c].
            • c_edge[j] = c_edge[j] + 1.
        • else
            • Choose a symbol $\phi \in \bigcup_{i=1}^{r} s_i/s_j$ uniform randomly.
            • if $\phi \in s_t, t \neq j$
                ○ c_edge[t] = c_edge[t] + 1.
        • c_deg = c_deg + 1.
    4. XOR all the selected information symbols to find the value of $y_m$.
```

[a]Here, *c_edge[r]* denotes a vector of length r. Also, *c_edge[r]* = **0** denotes that each entry of the vector is initialized to 0.

[b]This means that a set index j (s_j) is selected with probability p_{j,d_m}

[c]*count_edge[j]* previously chosen information symbols are excluded from this selection process (all selections are without replacement)

In expanding window fountain (EWF) codes [5], one of the expanding windows is first selected by a coded symbol before the selection of its edge connections. After choosing a specific window W_j with probability Γ_j, all the edge connections of that coded symbol are constrained to be chosen from the selected window according to a window-specific DD given by for $j = 1, \dots, r$, $\Phi^{(j)}(x) = \sum_{i=1}^{|W_j|} \Phi_i^{(j)} x^i$. When we compare the generalized code with EWF codes, we use a DD called the *compound degree distribution* $\Lambda^c(x)$, given as follows,

Definition 3: (*Compound* DD) $\Lambda^c(x) = \sum_{i=1}^{k} \Lambda_i^c x^i$ where $\Lambda_i^c \triangleq \sum_{j=1}^{r} \rho_j \Phi_i^{(j)}$ and Λ_i^c is the probability of choosing degree i, $\Phi_i^{(j)} \triangleq 0$ if $i > |W_j|$ and $0 \leq \{\rho_j\}_{j=1}^{r} \leq 1$ such that $\sum_{j=1}^{r} \rho_j = 1$.

3.2. Unequal Iteration Time Property

The URT definition given in [4] or [5] is with respect to the reception overhead ($\epsilon = n/k - 1$). Given a target bit rate, increasing portions of information bits can be decoded after receiving increasing numbers of encoded bits, so that information bits can be recovered in a progressive manner. So URT is concerned with performance as a function of the number of received symbols. In contrast to this definition, UIT is concerned with the performance as a function of the number of iterations of the decoding algorithm. Usually, it is assumed that the decoding algorithm iterates as much as needed. Since rateless codes are most effective with increasing source block sizes, this requires more iterations of the BPA. However, in many portable wireless applications, low-complexity designs are desired for less battery consumption. In that scenario, UIT is relevant and provides insights about the performance of the various schemes.

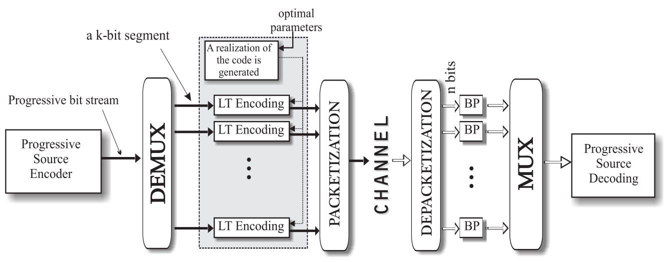

Fig. 1: The block diagram of the progressive transmission scheme.

4. PROGRESSIVE TRANSMISSION PROTOCOLS AND DESIGN PARAMETER SELECTION

4.1. Progressive source transmission system description

In internet communications, the basic unit of multimedia information is a fixed or variable length packet or a bit-segment [8]. In this section, we describe the way we transmit a progressive source using the proposed extension of rateless codes. The system block diagram of our proposed setup is shown in Fig. 1. A bit stream is produced using an $L_x \times L_y$ grayscale image coded with SPIHT and arithmetic coding. The progressive bit stream is assumed to have a total of B bits, i.e., a source rate of $B/(L_x \times L_y)$ bits per pixel (bpp). We describe two different demultiplexing and packetizations for the rateless transmission of the source.

4.1.1. Equal size packetization (Protocol 1)

The first protocol is based on equal size packets which are created as follows: The progressive bit stream of B bits is written horizontally from left to write in a source block of size $k \times \lfloor B/k \rfloor$ as shown in Fig. 2. The reason for using

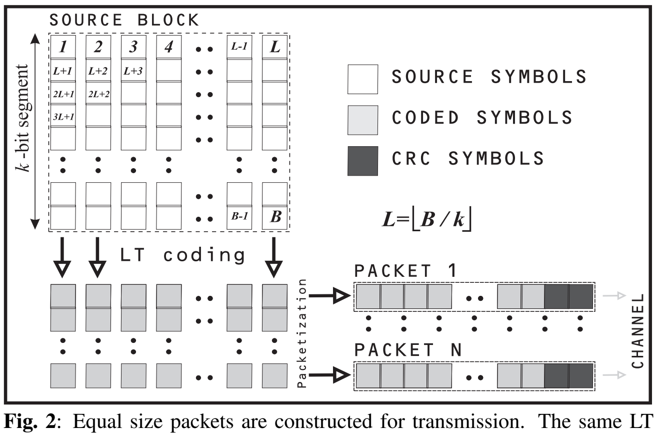

Fig. 2: Equal size packets are constructed for transmission. The same LT code is applied to each column to produce coded symbols.

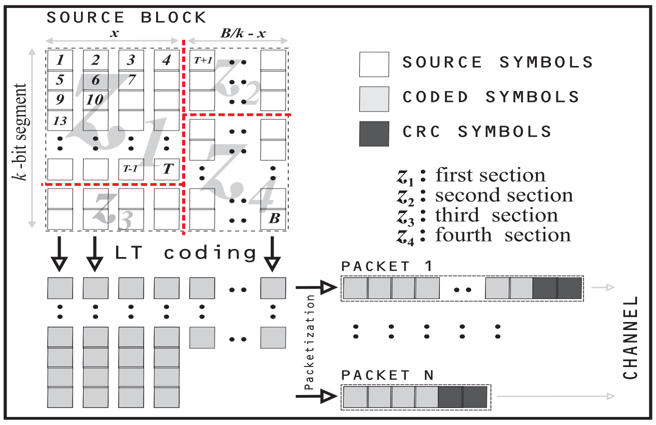

Fig. 3: Variable size packets are constructed for transmission. Different LT codes are applied to two different sets of columns to produce coded symbols.

such a demultiplexing methodology is that the proposed coding scheme is most powerful when the source bits within each column (k-bit segment) have unequal importance. In contrast, we could have written the source bits vertically before the LT encoding takes place. However, in that case, each segment would include almost equally-significant content.

After the rearrangement of source bits, we generate a particular realization of the proposed LT code and apply it to each segment to produce coded symbol streams. After LT encoding, horizontally aligned output symbols are concatenated with some CRC bits to form a packet before transmission. Because of the way the packets are formed, each coded symbol stream is exposed to the same erasure pattern. We collect n packets i.e., n coded symbols (i.e., an overhead of ϵ is assumed) at the receiver for each segment's decoding. Note that since we apply the same realization of the code to each column and use independent decoding, if m information bits are useful in each column after each BPA, because of the rearrangement process, we will have $\lfloor B/k \rfloor m$ total number of useful bits in the progressive bit stream for source decoding.

4.1.2. Variable size packetization (Protocol 2)

In the second protocol, the source block is partitioned into four disjoint sections. Next, the source bits are written horizontally in these sections named z_1, z_2, z_3 and z_4, as shown in Fig. 3. Coded symbols are generated by the proposed generalization of LT codes with two different set of parameters. The first x columns of the source block are encoded using $(A_j^{(1)}, B_j^{(1)}, C_j^{(1)}, \epsilon_1)$ and the rest of the columns are encoded using $(A_j^{(2)}, B_j^{(2)}, C_j^{(2)}, \epsilon_2)$, where ϵ_1 and ϵ_2 are the overheads the first x columns use and the rest of the columns use, respectively. Note that overheads ϵ_1 and ϵ_2 should satisfy the overhead constraint given by $x\epsilon_1 + (\lfloor B/k \rfloor - x)\epsilon_2 = \lfloor B/k \rfloor \epsilon$. Since ϵ_1 is not necessarily equal to ϵ_2, we will have two different packet sizes. Packets are produced by combining coded and CRC symbols before transmission.

Although protocol 2 is a more flexible scheme than protocol 1, the parameter set subject to optimization is larger, e.g., x is another parameter now subject to optimization. In addition, the second protocol has to communicate more header information with the receiver, such as packet sizes and optimal parameters of the rateless codes used to transmit the image.

4.2. Optimization Problem

The maximum number of iterations of the BPA (M_{max}) is selected so that for the test cases run, the BPA never required more than M_{max} iterations. In our case, we chose $M_{max} = 70$ because the algorithm always terminated in all the simulations (either by correct decoding or by having a decoding failure) prior to 70 iterations being reached. The general optimization problem is given by:

$$\min_{\substack{\Upsilon(x) \\ P_1(x),\ldots,P_k(x)}} \overline{\mathfrak{D}}_M \text{ s.t. } n \text{ coded symbols are collected,} \quad (1)$$

where $n \geq k$, and $\overline{\mathfrak{D}}_M$ is the average mean square distortion at the Mth iteration of the BPA. Note that the minimization can be done at any specific iteration M, $1 \leq M \leq M_{max}$. This could be useful if different UEP, URT and UIT characteristics are desired for a specific application. The proposed code is specified by the coefficients of $\Upsilon(x), P_1(x), \ldots, P_k(x)$, and the total number of coefficients is $(r+1)k$. However, since probabilities in each distribution must sum to one, we have $rk - 1$ parameters subject to optimization. For large k, it is infeasible to jointly optimize all design parameters. Exponential SD is a way of reducing the size of the parameter set subject to optimization. We use numerical experimentation and heuristics to find the optimum solution. More specifically, we discretized a possible range first, then we searched over all possibilities before reporting the set of parameters that minimize the mean distortion.

5. NUMERICAL RESULTS

Quality assessment of the decoded images is presented in terms of the average peak signal to noise ratio (PSNR), expressed in dB, a performance measure inversely related to the average mean square distortion by the Mth iteration of the BPA, $\overline{\mathfrak{D}}_M$. We use a standard 512×512 *Lena* image. We initially set $r = 2$, $B = 50000$ bits (a source rate of ≈ 0.19 bpp) and run all realizations 10^4 times. Since a specific image with a particular source encoder determines the rate-distortion (R-D) characteristics of the source, we can find the distortion[1] corresponding to each of the 10^4 different realizations of the code. Then, we take the average of these distortion values. We optimize the parameters of each system to give the minimum average distortion. Lastly, minimum average distortion is converted to average PSNR. Note that the reported values are optimal within the range of values considered, and to the accuracy of the step size of the discretization.

[1]We use mean square error (MSE) as our distortion metric throughout the paper.

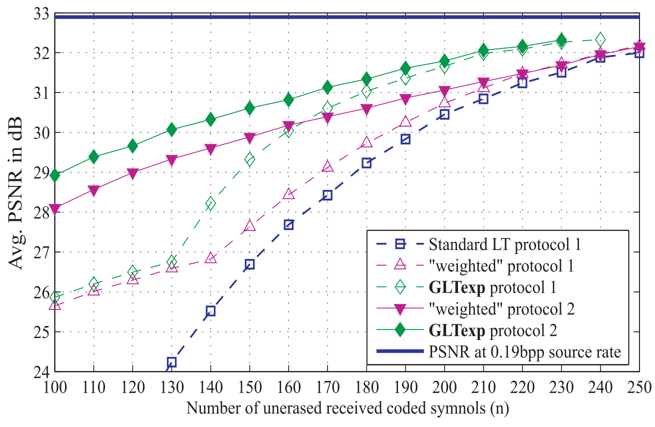

Fig. 4: Performance comparisons using *Lena*, $k = 100$, $\alpha_1 = 0.3$ and different protocols.

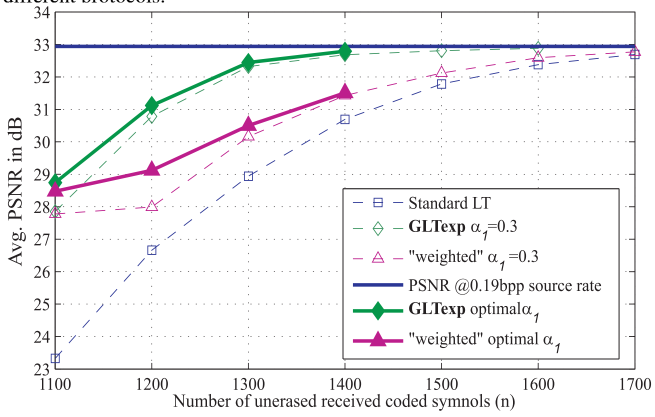

Fig. 5: Performance comparisons using *Lena*, $k = 1000$ and protocol 1.

In our first simulation, we compare our UEP GLT scheme with the weighted approach [4]. Both schemes use the RSD with $\gamma = c = 0.01$. We define a version of the proposed scheme named **GLTexp**, which uses the Exponential SD with $A_1^{(1)} + B_1^{(1)} = 1$ and $A_1^{(2)} + B_1^{(2)} = 1$. For a given overhead ϵ, the first protocol optimizes the set $\{A_1^{(1)}, C_1^{(1)}\}$ so that the proposed scheme achieves minimum distortion. The second protocol optimizes $\{A_1^{(1)}, C_1^{(1)}, A_1^{(2)}, C_1^{(2)}, \epsilon_1, \epsilon_2, x\}$ for minimum distortion such that the overhead constraint of protocol 1 (ϵ) is met.

For a fair comparison, we also optimize the weighting parameter of the weighted approach [4] for minimum distortion. Average PSNR versus the number of reliably received coded symbols (n) is plotted in Fig. 4 for both protocols using a fixed $\alpha_1 = 0.3$ for each system. We also included the performance of standard LT coding (EEP scheme) in the same figure for comparison. For protocol 2, x is also subject to optimization. Performance results are shown for $k = 100$ as a function of n, and are observed to be increasing with growing n. Although the proposed scheme does not show a major performance improvement over the weighted approach up to $n = 130$ using protocol 1, it provides over a 1dB improvement for a substantial range of n (from 140 to 210), and a huge improvement over standard LT coding. Using protocol 2, an almost 1dB gain is possible for any n of interest. As can

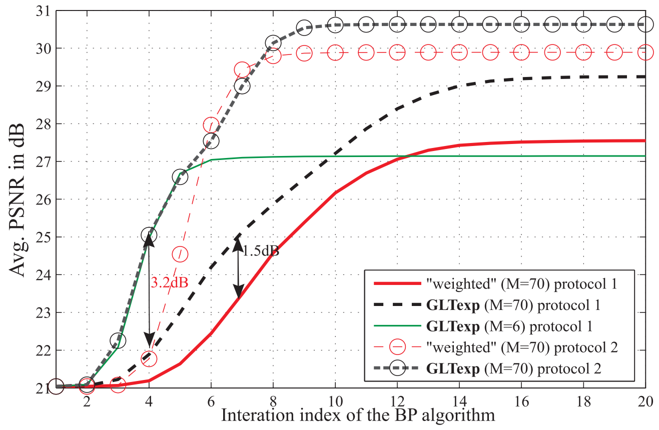

Fig. 6: UIT Performance comparisons using *Lena* for $k = 100$, $n = 150$, $\alpha_1 = 0.3$ and $B = 50000$ bits using different protocols.

be seen in Fig. 4, after collecting $n > 240$ coded symbols, all the systems perform almost a complete decoding of the whole source block and hence they exhibit a similar performance. As a perspective, the PSNR performance of a source with a rate of ≈ 0.19bpp, and which is operating under error-free conditions, is included in the figures for reference. Similar performance results for $k = 1000$ are shown in Fig. 5. It can be observed that using protocol 1, the proposed scheme provides over a 1dB improvement for a range of n from 1150 to 1500 for both fixed and optimal α_1, and more improvement over standard LT coding. These results also advocate the asymptotically optimal behavior of LT codes. Using the proposed scheme for example, an overhead of $\epsilon = 0.3$ when $k = 1000$ gives around 32.5dB average PSNR, whereas the same overhead with $k = 100$ gives around 27.6dB average PSNR.

Note that Figs. 4 and 5 can be considered to be depictions of both the URT and UEP performances of the various schemes. These figures all show the quality as a function of the number of received symbols. Therefore, for a fixed quality, the horizontal distance between two curves is a measure of how much earlier in the received bit stream one system can recover that fixed quality compared to another system (URT property). For a fixed number of received symbols, the vertical distance between two curves is a measure of the PSNR gain (UEP property).

Next, we show the UIT performance of the proposed scheme, that is, how the PSNR varies if the BP algorithm is not allowed to iterate until its natural termination[2], but is instead cut off at some early iteration M, by design. The proposed scheme and the weighted approach are compared using parameter values optimized for $M = 70$ as a function of the number of iterations in the BP algorithm using different protocols. Fig. 6 also shows a performance curve that results by solving the minimization problem in Eqn. (1) for $M = 6$ us-

[2]In case of natural termination, the algorithm either decodes all the information bits, or declares failure, i.e., there remains no degree-one coded symbol after edge eliminations and node updates, even if the decoding of the whole source block is not complete.

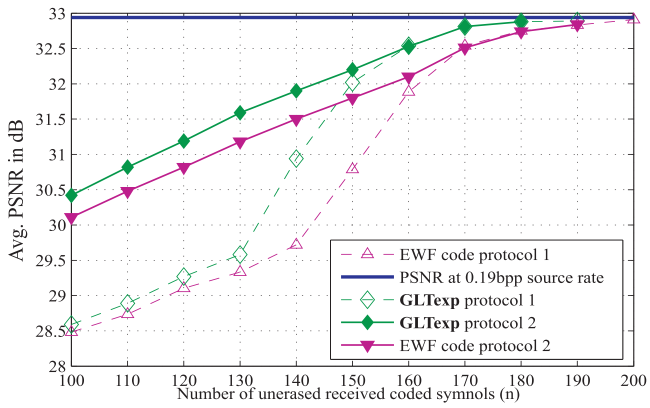

Fig. 7: Performance comparisons with EWF codes using *Lena*, $k = 100$ and protocol 1.

ing protocol 1. This curve gives a large gain over the weighted approach at iteration 6 at the expense of some performance loss (compared to the weighted approach) at later iterations. Similarly, using protocol 2, although the gain is reduced for iteration indexes larger than 10, a huge improvement can be observed at iteration index 4. This result shows that we can tailor the parameters of the proposed scheme to achieve better UIT properties at the expense of some loss in performance at later iterations.

Finally, Fig. 7 compares EWF codes with the proposed generalization with $k = 100$ using both protocols. The EWF code has the parameters α_1 and Γ_1 subject to optimization [5]. The DDs for the UEP EWF code, $\{\Phi^{(1)}(x), \Phi^{(2)}(x)\}$ are the *Truncated Robust Soliton distribution* [5] $\Omega_{rs}(k_{rs}, \gamma, c)$ using $\gamma = c = 0.01$, in which k_{rs} is the maximum degree of each DD and is constrained not to exceed the size of the corresponding window i.e., the size of the first window $W_1 = \alpha_1 k$ or the size of the second window $W_2 = k$ [5]. The **GLTexp** now uses the compound DD, i.e., for any coded symbol, the probability of choosing degree i is given by

$$\Lambda_i^c = \rho_1 \Phi_i^{(1)} + (1 - \rho_1)\Phi_i^{(2)} \tag{2}$$

where $\Phi_i^{(1)} = 0$ for $i > \alpha_1 k$. **GLTexp** uses the Exponential SD with $A_1^{(1)} + B_1^{(1)} = 1$. In this case, it optimizes the set $\{\alpha_1, \rho_1, A_1^{(1)}, C_1^{(1)}\}$ so that the proposed scheme achieves minimum distortion. As can be observed using protocol 1, **GLTexp** gives similar gains for the range of n from 130 to 170. Since EWF codes can use different degree distributions for each window (and window sizes are optimized in our transmission scenario) and protocol 2 allows more flexibility, the performance of EWF codes is greatly improved when they are used with protocol 2. However, around a 0.5dB gain over the EWF codes is still possible using the proposed scheme in conjunction with protocol 2.

6. CONCLUSION

Rateless codes are a type of erasure-correcting code with simple encoding and decoding structures that are used both for point-to-point communications and for multicasting information. The initial design of such codes is aimed at recovering the entire information block, and therefore might not be the best choice when different parts of the data have different levels of importance, such as image or video files compressed in a progressive or scalable fashion. In this study, we introduced a generalized version of UEP LT codes having a larger set of parameters, and hence a more flexible rateless coding scheme. We also introduced two different progressive transmission protocols using this generalized version of UEP LT codes. The proposed scheme was shown to provide an efficient progressive coding property. We compared the proposed scheme with two other major UEP LT codes for a progressive transmission scenario. Simulations showed that the proposed coding scheme outperformed the previous schemes by providing improved UEP, URT and UIT properties.

7. REFERENCES

[1] D. J. C. MacKay, “Fountain codes,”, *IEE Proc.-Commun.*, vol. 152, pp. 1062–1068, 2005.

[2] M. Luby, “LT-Codes”, *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 271–280, 2002.

[3] A. Said and W. A. Pearlman, “A new, fast, and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 6, pp. 243–250, June 1996.

[4] N. Rahnavard, B. N. Vellambi and F. Fekri, “Rateless Codes with Unequal Protection Property”, *IEEE Trans. Inf. Theory,* Vol. 53, No. 4, pp. 1521–1532, April 2007.

[5] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk and R. Piechocki, “Expanding window Fountain codes for Unequal Error Protection”, *IEEE Trans. Commun.,* Vol. 57, No. 9, pp. 2510–2516, Sep. 2007.

[6] S. S. Woo and M. K. Cheng, ”Prioritized LT codes,” *in Proc. of 42nd Annual Conf. on Information Sciences and Systems, CISS 2008,* Princeton, NJ, USA, pp. 568–573, 2008.

[7] J. Wagner, J. Chakareski, and P. Frossard, “Streaming of scalable video from multiple server using rateless codes,” *in Proc. of the IEEE ICME,* Toronto, Canada, 2006.

[8] J. Kurose and K. Ross, “Computer Networking: A Top-Down Approach Featuring the Internet”, Addison Wesley, Second Edition, 2002.