Generalized Unequal Error Protection LT Codes for Progressive Data Transmission

Suayb S. Arslan, *Student Member, IEEE*, Pamela C. Cosman, *Fellow, IEEE*, and Laurence B. Milstein, *Fellow, IEEE*

***Abstract*— The original design of standard digital fountain codes assumes that the coded information symbols are equally important. In many applications, some source symbols are more important than others, and they must be recovered prior to the rest. Unequal error protection (UEP) designs are attractive solutions for such source transmissions. In this paper, we introduce a more generalized design for the first universal fountain code design, Luby transform (LT) codes, that make it particularly suited for progressive bit stream transmissions. We apply the generalized LT codes to a progressive source and show that it has better UEP properties than other published results in the literature. For example, using the proposed generalization, we obtained up to 1.7-dB peak signal to noise ratio gain in a progressive image transmission scenario over the two major UEP fountain code designs.**

***Index Terms*— Fountain codes, iterative decoding, lossy channels, progressive sources, unequal error protection (UEP), unequal iteration time (UIT), unequal recovery time (URT).**

I. Introduction

ERASURE channels can be used to model various data networks, where packets are either received reliably or lost completely due to channel impairments, buffer overflows or excessive delays. For example, data files sent over the internet are chopped into fixed or variable size packets, and each packet is either received without error or corrupted due to the channel and therefore considered erased during the transmission. A cyclic redundancy check (CRC) code is typically used to detect packet errors. When such errors are detected, and the CRC code reports failure, the receiver discards the whole packet [1].

For communicating over erasure channels, it is customary to employ a feedback mechanism from the receiver to the sender to manage the retransmission of erased packets. Such retransmission protocols might be burdened with an excessive number of feedback messages [2]. One interest of the coding community was to make erasure-correcting codes require no feedback (or almost no feedback) with reduced encoding and decoding complexities. Digital fountain (DF) codes are a class of erasure-correcting codes that require almost no feedback with simple encoding and decoding structures. Fountain codes are also known as *rateless* codes i.e., we can independently generate as many coded symbols as desired from the original content [3]. The first known rateless code is the Luby transform (LT) code [3].

Because of the attractive features of fountain codes, and recently introduced unequal error protection (UEP) fountain code designs, they have become increasingly popular for multimedia communications needing UEP [4]. The original design of fountain codes assumes that the encoded data bears equally important information about the source. For example, executable computer programs are such data files that can be transmitted using equal error protection (EEP) mechanisms. However, in some multimedia applications, a portion of the data might be more important than the rest. In particular, initial parts of the progressive bit stream are more important for the reconstruction of the image than are later portions [5]. When the coding allows for a low quality or low resolution version of the source to be recovered from an initial portion of the bit stream, and hence recovered earlier in time than a high quality version of the source, this is referred to as the unequal recovery time (URT) property. In 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 UEP and URT properties might be useful [6].

A. Related Work

Unequal protection for progressive sources is usually achieved by joint source channel coding (JSCC), in which more important contents are encoded with stronger codes (see [7]–[10]). Those studies assume that channel state information is available at the transmitter. When the channel erasure rate or the channel fading coefficients are unknown and non-uniform, one JSCC mechanism targets the worst-case error rate as the primary design criterion in choosing the optimal code rate/s [11]. An optimal fixed code rate for a good state is typically unable to perform well when the channel is bad. At the other end, when the code rate is optimized for the bad

Manuscript received August 31, 2011; revised December 16, 2011; accepted April 1, 2012. Date of publication April 26, 2012; date of current version July 18, 2012. This work was supported in part by LG Electronics Inc., Intel 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. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Anthony Vetro.

The authors are with the Department of Electrical and Computer Engineering, University of California, San Diego, CA 92093-0407 USA (e-mail: sarslan@ucsd.edu; pcosman@ucsd.edu; lmilstein@ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2012.2195668

state, there will typically be many unnecessary redundant bits. Rateless codes, such as LT codes [3], Raptor codes [12], and Online codes [13] do not assume any information about the channel and, therefore, are good matches for transmitting data over time varying channels with unknown parameters. Similarly, such codes can be very good candidates for multicast transmissions because there is no prior assumption about the channels.

In the original study of LT codes, EEP of all the information symbols is assumed. The degree of a coded symbol d is assigned according to a suitable distribution called the degree distribution (DD). After choosing the degree for each coded symbol, a d-element subset of the information block is chosen randomly according to another distribution called the selection distribution (SD). It is shown in a series of studies [14]–[17] 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 the DD and SD. 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 [18], [19].

In [16], a structured LT code graph is constructed. Then the proposed code is randomized using a degree assigner and a random selector. Although the terminology in [16] is different from that in [14], the unequal protection is achieved essentially the same way using a fixed DD and a non-uniform SD. In other studies such as [20], the authors introduced the use of overlapped windows to allow one to have a virtually extended block for superior performance compared to fixed-window encoding. Although the main objective was to increase the virtual block size of the LT codes for a better performance, it is recognized that such block size enhancement can be used to provide UEP. Later, a similar idea is used in [21], called block duplication, for the EEP of a two-layer source. It is shown that under certain assumptions, the method of [21] provides slightly better performance than [14]. The gains reported in [17] over [14] are greater due to using different DDs for each window. This shows the combined choice of degree and SD had a great impact on the final performance of the fountain code. Last, the algorithm presented in [22] chooses degree-one coded symbols from the high priority class of source symbols, and the edge connections of degree-two coded symbols are selected non-uniformly similar to the weighted approach of [14]. Unequal protection is achieved by this slight modification and a fixed DD, and the rest of the encoding is exactly the same as in original LT coding. Although all of these studies ([14]–[22]) consider fountain codes with a UEP property for a given application scenario, a joint optimization of degree and SD remains untouched.

B. Contribution and Organization

In this paper, we propose a generalization for two major UEP LT codes. Specifically, inspired by [22], we introduce a systematic degree-dependent selection concept that can be applied to previous unequal protection rateless code designs. In addition, we present a progressive source transmission scheme using rateless codes. Since the beginning part of progressive bit streams is more important than the succeeding parts, previous UEP designs can directly be used in the transmission of progressive sources. However, progressive sources do not consist of only a few layers as traditionally assumed in previous scalable video transmission scenarios [23]. We tailor the parameters of the proposed design for this particular scenario to minimize the expected distortion. We show that although the previous UEP schemes can be used to provide unequal protection for progressive bit streams, the proposed generalization of the rateless codes, and its configuration for progressive bit streams, give dramatic improvements in terms of end-to-end expected distortion over the UEP rateless code designs described in [14]–[17]. Furthermore, we introduce a new property called unequal iteration time (UIT), where we evaluate system performance as a function of the iteration index of the decoding algorithm. This might be particularly important for portable devices that are constrained by low-complexity receiver architectures.

The rest of this paper is organized as follows. Section II gives necessary background for progressive source coding, LT codes, and previous UEP designs. Section III introduces the proposed UEP generalized LT (GLT) coding design, the encoding algorithm, the optimization problem, and the UIT property. Section IV presents the proposed progressive transmission set-up in detail. It also describes parameter selections of the UEP GLT coding for comparisons to some of the major UEP designs. Section V presents numerical results to show the effectiveness of the proposed idea. Finally, Section VI concludes this paper.

II. Background

In this section, we first review progressive source coding. Then, we review fountain codes and review the original encoding and decoding structures for LT codes. Last, we briefly discuss the previous UEP designs.

A. Progressive Source Coding

In progressive source coding, prefixes of a single bitstream allow the decoder to progressively reconstruct the source. In other words, all encodings of the source at lower bit rates are embedded in the beginning of the encoded source at higher bit rates. Thus, progressive source bitstreams have the property that the beginning part of the bit stream is more important than the succeeding parts of the bit stream. Progressive source coders also have the property that bits later in the bit stream are of no use unless the bits that precede them are reliably received. In this respect, it is, therefore, convenient to define *useful bits* to be the set of consecutive bits that are recovered from the beginning of the bit stream up to the first unrecovered bit location. In progressive source transmission, it is of more concern to consider the decoded useful bits rather than the decoded total bits.

Progressive transmission might be very useful in multimedia communication scenarios, in particular fast browsing of high definition media in non-homogeneous networks. However, such bit streams use various forms of variable length codes, making them extremely susceptible to noisy channel effects, i.e., any bit error might render the rest of the bit stream useless. Therefore, a good protection mechanism is needed for reliable transfer of such compressed data.

B. DF Codes

Fountain codes are random bipartite graph codes in which the source data is recoverable from any subset of coded symbols, when this subset contains enough coded symbols in a lossy packet transmission scenario. These codes have significantly less complex encoding and decoding structures than more traditional Reed-Solomon codes, and they also exhibit high erasure correction performance for large block lengths. A fountain encoder generates potentially an unlimited number of coded symbols. Once enough coded symbols are collected at the receiver, an iterative decoder using a belief propagation (BP) algorithm recovers the information symbols through *graph pruning* [24].

1) Encoding: An LT encoder takes a set of k symbols of information to generate coded symbols of the same alphabet. We consider a binary alphabet and let a binary information block $\mathbf{x}^T = (x_1, x_2, \ldots, x_k) \in \mathbb{F}_2^k$ consist of k bits. The m-th coded symbol y_m is generated in the following way: First, the degree of 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 $\ell \in \{1, \ldots, k\}$. Then, after choosing the degree $d_m \in \{1, \ldots, k\}$, a d_m-element subset of $\mathbf{x}$ is chosen randomly according to the SD. For standard LT coding, the SD is the uniform distribution. 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. More specifically, this means that any possible binary vector of weight d_m is selected with probability $1/\binom{k}{d_m}$. Finally, the coded symbol is given by $y_m = \mathbf{w}_m^T \mathbf{x}$ (mod 2). Note that all these operations are in modulo 2.

Some of the coded symbols are erased by the channel, and for decoding purposes, we concern ourselves only with those n-coded symbols which arrive unerased at the decoder. Hence, the subscript m on y_m, d_m, and $\mathbf{w}_m$ runs only from 1 to n, and we ignore at the decoder those quantities associated with erased symbols. Note that the way we generate each coded symbol is independent of the way we generate other coded symbols. LT codes have been shown to be asymptotically optimal, i.e., as k tends to infinity, then $n \to k$ bits will be enough to recover all the information content [3], [13].

2) Decoding: The maximum likelihood decoding of LT codes over the binary erasure channel (BEC) is the problem of recovering k information symbols from the n reliably received coded symbols. Although maximum likelihood decoding is optimal, it is computationally prohibitive for long block lengths (k large). In order to allow linear decoding complexity with increasing block length, the BP algorithm is used [1].

Let us denote information symbols as *variable nodes*, and coded symbols as *check nodes* in the bipartite graph representation of LT coding, as shown in Fig. 1. Assuming there is at least one degree-one coded symbol received, the BP starts decoding from degree-one coded symbols. The content

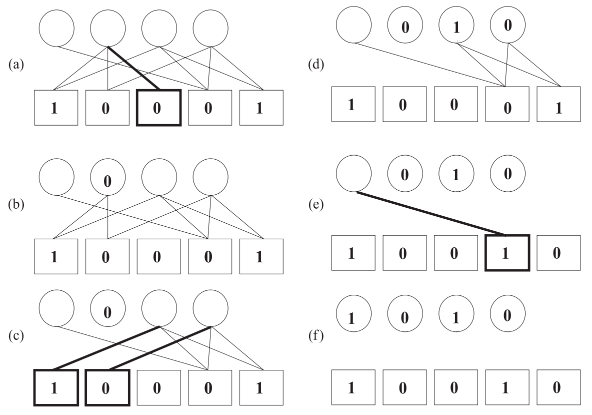


Fig. 1. BP algorithm for LT decoding. $k = 4$, $n = 5$. Squares (□) are check nodes (coded symbols), and circles (○) represent variable nodes (information symbols). (a) BP starts from degree-one coded symbol and (b)–(f) variable node updates alternate with coded symbol updates.

is immediately sent to their one neighbor (variable nodes) to decode the information bits. An example of the decoding algorithm operation with $k = 4$ and $n = 5$ is shown in Fig. 1. As can be seen in Fig. 1(a), BP starts decoding from the degree-one coded symbol. In Fig. 1(b), the degree-one coded symbol has transferred its content to its one neighboring information symbol, and the corresponding edge is eliminated. This is called the variable node update step. In the next decoding step shown in Fig. 1(c), the recovered information symbol is added modulo 2 to each connected coded symbol to update its content. Then, the corresponding edges are eliminated. This step is called the check node update step. In the later stages, as depicted in Fig. 1(d)–(f), update steps for variable nodes and check nodes are performed alternately to recover the information symbols. Finally, after all node updates are made, as shown in Fig. 1(f), we recover all the information symbols. When there is no degree-one check node at any stage of BP, the algorithm stops and declares a decoding error. Details of BP can be found in [3].

For the decoding of a whole information block to be successful, we need at least one degree-one check node at each iteration. The set of degree-one nodes is called the *ripple*. If the ripple does not have any elements, the decoder stops iterating. Luby [3] proposed the *ideal-soliton* DD so that at each decoding iteration, the expected ripple size is one. This means in expectation, the algorithm never stops and decodes the whole information block.

Definition 1—Ideal-Soliton Distribution:

1) $\Omega_1 = 1/k$;
2) for $i = 2, \ldots, k$: $\Omega_i = 1/i(i-1)$.

However, the performance of the ideal-soliton distribution is poor in practice because the ripple size can very likely be zero. Later, Luby [3] proposed the *robust soliton* distribution (RSD), which overcomes this problem. The expected ripple size of RSD 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. RSD is shown to give good performance using practical information block sizes, such as $k = 100$ or $k = 1000$.

Definition 2—Robust-Soliton Distribution:

1) for $i = 1, \ldots, k$: probability of choosing degree i is given by $\mu_i = (\Omega_i + \tau_i)/\beta$, where

a) $\tau_i = \begin{cases} R/ik, & \text{for } i = 1, \ldots, k/R - 1 \\ R\ln(R/\delta)/k, & \text{for } i = k/R \\ 0, & \text{for } i = k/R + 1, \ldots, k; \end{cases}$

b) $\beta = \sum_{i=1}^{k} (\Omega_i + \tau_i)$.

C. UEP DF Code Designs

Previous LT codes with UEP and URT properties can be classified into two main categories.

1) Weighted Approach: In studies, such as [14], [15], and [21], the authors proposed various intuitive modifications to the SD of LT codes. These unequal protection schemes will be referred to as a *weighted approach* in this paper. A generic description of the weighted approach is as follows. A block of k source symbols is divided into r disjoint sets $s_1, \ldots, s_r$, having sizes $|s_j| = \alpha_j k$ symbols, where $0 < \alpha_j < 1$ are design parameters satisfying $\sum_{j=1}^{r} \alpha_j = 1$, $\alpha_j k$ is an integer, and $|.|$ denotes the cardinality of the argument. The encoding process is the same as for LT codes, except the check nodes select their adjacent variable nodes non-uniformly. More specifically, after choosing a degree d_m according to some DD, d_m information symbols are selected one by one: first the set s_j is chosen from $\{s_1, \ldots, s_r\}$ with probability ω_j. After choosing the set index, the input symbol is selected uniformly from the set s_j only. This selection process is repeated d_m times, without replacement, to determine d_m distinct connections to the information symbols. Finally, the value of the coded symbol is given by the sum of the selected d_m information symbols. The probability of selecting a particular set is designed such that the probability of recovery for more important classes of bits is higher than that of the less important classes of bits. Therefore, this approach, in which the neighbors of a coded symbol are selected non-uniformly, is a generalization of LT codes.

2) Expanding Window Fountain (EWF) Codes: Another approach, called EWF codes, was developed in [17]. An information block of k bits was divided into r successively larger windows. This can be thought of as first dividing the information block into r disjoint sets $s_1, \ldots, s_r$, and then defining r embedded windows $\{W_j\}_{j=1}^{r}$, such that $W_j = \bigcup_{l=1}^{j} s_l$. To generate a new EWF coded symbol, one of the windows is randomly selected by the coded symbol according to a window SD given as follows.

Definition 3—Window SD:

1) $\Gamma(x) = \sum_{j=1}^{r} \Gamma_j x^i$

where Γ_j is the probability that window W_j is chosen.

Upon selection of the window, standard LT coding is applied only to the bits contained in that window using a suitably chosen DD given as follows.

Definition 4—jth Window DD:

1) for $j = 1, \ldots, r$: $\Phi^{(j)}(x) = \sum_{i=1}^{|W_j|} \Phi_i^{(j)} x^i$

where $\Phi_i^{(j)}$ is the conditional probability of choosing degree i, given that W_j is selected by the coded symbol.

The same procedure is invoked for each encoded symbol. Thus, a UEP EWF code is a rateless code, which provides unequal protection by choosing the appropriate set $\{\Gamma(x), \Phi^{(1)}(x), \Phi^{(2)}(x), \ldots, \Phi^{(r)}(x)\}$. UEP EWF coding modifies not only the SD but also the DD, which makes the code a more flexible unequal protection scheme compared to the weighted approach. However, the superiority of this scheme over the weighted approach depends on whether the DD for each window is judiciously selected.

III. UEP GLT CODING

In this section, first we apply the degree-dependent selection idea to the weighted approach to provide increased UEP, URT, and UIT properties. Note that the same idea can be applied to UEP EWF codes through the use of a degree-dependent window SD. More specifically, we propose to use a (window) SD that depends on the degree number of the particular check node to give priority decoding to earlier bits of a progressive bit stream.

A. Generalization of "Weighted Approach"

Similar to previous studies, let us 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 $\sum_{j=1}^{r} \alpha_j = 1$ and the $\alpha_j k$ values 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 as follows.

Definition 5—Generalized SD:

1) 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. For convenience, we denote the proposed SD in a matrix form as follows:

$$\mathbf{P}_{r\times k} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,k} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,k} \\ \vdots & \vdots & \ldots & \vdots \\ p_{r-1,1} & p_{r-1,2} & \cdots & p_{r-1,k} \\ p_{r,1} & p_{r,2} & \cdots & p_{r,k} \end{bmatrix}.$$

Since the set of probabilities in each column sums to unity, the number of design parameters of $\mathbf{P}_{r\times k}$ is $(r-1) \times k$. Similarly, for the proposed GLT, the DD can be expressed in a vector form as $\boldsymbol{\lambda}_k$, where the ith vector entry is the probability that a coded symbol chooses degree i. Note that $\boldsymbol{\lambda}_k$ and $\mathbf{P}_{r\times k}$ completely determine the performance of the proposed GLT code. More specifically, the design steps taken to generate each coded symbol are summarized in Algorithm 1.

In the BP algorithm, we observe that not all the check nodes decode information symbols at each iteration. For example, degree-one check nodes immediately decode neighboring information symbols at the very first iteration [3]. Then, degree-two and degree-three check nodes recover some of the information bits later in the sequence of iterations. In general, at the later update iterations, low degree check nodes will

Algorithm 1 UEP GLT encoding (for "weighted approach")

```
for $m = 1, \ldots, n$,
  1) Choose a degree $d_m \in \{1, \ldots, k\}$ according to some appropriate DD $\Upsilon_w(x)$.
  2) Initialize count_deg = 1, count_edge[r] = 0.[a]
  3) while count_deg $\leq d_m$
       a) Choose a set index $j \in \{1, \ldots, r\}$ according to the Generalized SD $\{p_{1,d_m}, p_{2,d_m}, \ldots, p_{r,d_m}\}$[b]
       b) if count_edge[j] $< \alpha_j k$
            i) Choose an information symbol from $s_j$ uniform randomly. count_edge[j] previously chosen information symbols are excluded from this selection process (selection without replacement).
            ii) count_edge[j] = count_edge[j] + 1.
       c) else
            i) Choose an information symbol from all the sets except $s_j$ uniform randomly. Again, the selection uses only information symbols not previously chosen.
            ii) if selected information symbol belongs to $s_t, t \neq j$
                  A) count_edge[t] = count_edge[t] + 1.
            iii) end if
       d) end if
       e) count_deg = count_deg + 1.
  4) end while
  5) XOR all the selected information symbols to find the value of $y_m$.
end for
```

[a]Here, *count_edge[r]* denotes a vector of length r. Also, *count_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}.

Algorithm 2 UEP GLT encoding (for EWF codes)

```
for $m = 1, \ldots, n$,
  1) Choose a degree $d_m \in \{1, \ldots, k\}$ according to some appropriate DD $\Upsilon_{ewf}(x)$.
  2) Initialize count_deg = 1, count_edge[r] = 0.[a]
  3) while count_deg $\leq d_m$
       a) Choose a window index $j \in \{1, \ldots, r\}$ i.e., $W_j$ according to the Generalized window SD $\{\gamma_{1,d_m}, \gamma_{2,d_m}, \ldots, \gamma_{r,d_m}\}$[b]
       b) if count_edge[j] $< k \sum_{i=1}^{j} \alpha_i$
            i) Choose an information symbol from $W_j$ uniform randomly. count_edge[j] previously chosen information symbols are excluded from this selection process (selection without replacement).
            ii) count_edge[j] = count_edge[j] + 1.
       c) else
            i) Choose an information symbol from the set $\bigcup_{i=j+1}^{r} s_i$ uniform randomly. Again, the selection uses only information symbols not previously chosen by the same coded symbol.
       d) end if
       e) count_deg = count_deg + 1.
  4) end while
  5) XOR all the selected information symbols to find the value of $y_m$.
end for
```

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

[b]This means that the window index j ($W_j = \bigcup_{i=1}^{j} s_i$) is selected with probability γ_{j,d_m}.

already be released from the decoding process, and higher degree check nodes start decoding the information symbols (due to edge eliminations). So, the coded symbols take part in different stages of the BP decoding process, depending on their degree numbers.

It is clear from the previous designs that unequal protection is achieved by allowing coded symbols to make more edge connections with more important information sets. This increases the probability of decoding the more important symbols. However, coded symbols are able to decode information symbols in different iterations of the BP depending on their degree numbers. For example, at the second iteration of the BP algorithm, the probability that degree-2 coded symbols decode information symbols is higher than that of coded symbols with degrees > 2. If the BP algorithm stops unexpectedly at an early iteration, it is essential that the more important information symbols are recovered. This suggests that it is beneficial to have low degree check nodes generally make edge connections with important information sets. This can be achieved by the proposed scheme. An example realization for $r = 2$ is shown in Fig. 2. A check node with degree 6 makes $t_2 = 1$ connection with s_1, whereas a check node with degree 4 makes $t_1 = 3$

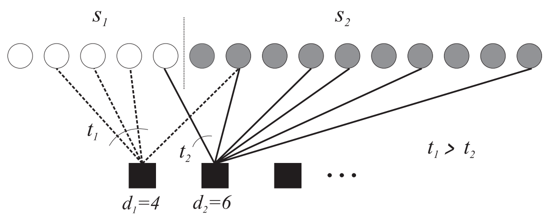


Fig. 2. Non-uniform selection of variable nodes is made based on the degree of the check node. Black squares represent coded symbols (check nodes), and circles represent information symbols (variable nodes). White and gray circles represent two different priority classes. Only the edge connections of the first two coded symbols are shown for clarity.

connections with s_1 satisfying $t_1 > t_2$. The main advantage of the proposed scheme over the previous designs is to allow a greater flexibility in the LT encoding process. We will see later that this flexibility will allow us to tailor the parameters of the system to a progressive source transmission scenario.

B. Generalization of EWF Codes

In the encoding process of this generalization, after choosing the degree number for each coded symbol, we select the edge connections according to a *generalized window* SD, given as follows.

Definition 6—Generalized Window SD:

1) for $i = 1, \ldots, k$: $L_i(x) = \sum_{j=1}^{r} \gamma_{j,i} x^j$

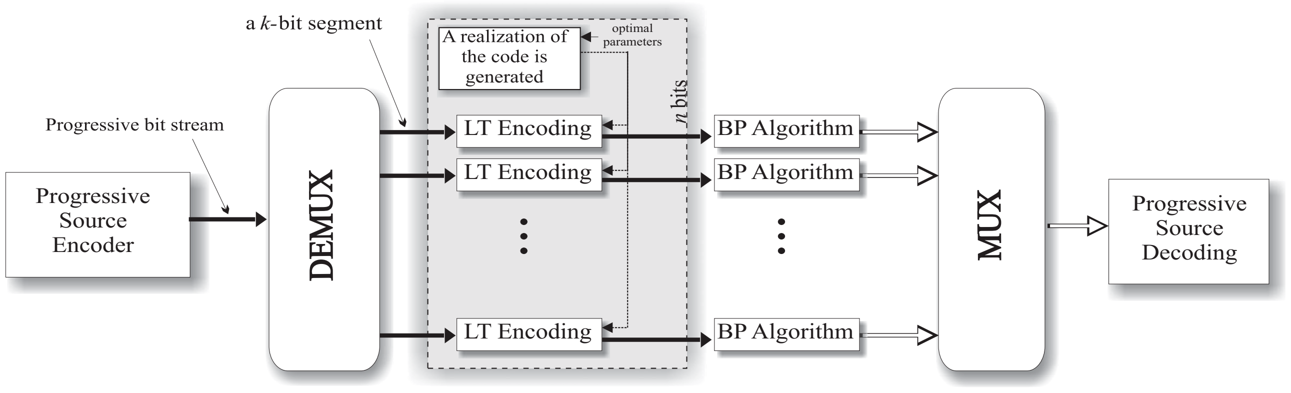


Fig. 3. Block diagram of the progressive transmission scheme.

where $\gamma_{j,i} \geq 0$ is the conditional probability of choosing the information set s_j, given that the degree of the coded symbol is i and $\sum_{j=1}^{r} \gamma_{j,i} = 1$.

Similar to the previous generalization, $\gamma_{j,i}$ are design parameters of the system, subject to optimization. For convenience, we denote the proposed window SD in a matrix form as follows:

$$\mathbf{L}_{r\times k} = \begin{bmatrix} \gamma_{1,1} & \gamma_{1,2} & \cdots & \gamma_{1,k} \\ \gamma_{2,1} & \gamma_{2,2} & \cdots & \gamma_{2,k} \\ \vdots & \vdots & \ldots & \vdots \\ \gamma_{r-1,1} & \gamma_{r-1,2} & \cdots & \gamma_{r-1,k} \\ \gamma_{r,1} & \gamma_{r,2} & \cdots & \gamma_{r,k} \end{bmatrix}.$$

The set of probabilities in each column sums to unity, and the number of design parameters of $\mathbf{L}_{r\times k}$ is again $(r-1)\times k$. Similarly, we observe that $\boldsymbol{\lambda}_k$ and $\mathbf{L}_{r\times k}$ completely determine the performance of the proposed generalization of EWF codes. More specifically, the design steps taken to generate each coded symbol are summarized in Algorithm 2.

C. UIT Property

The URT definition, given in [15] or [17], is with respect to the reception overhead (ϵ), i.e., it examines the fraction of the source message that can be recovered for different overhead values. This means that, given a target bit-error 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.

Thus, 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 in the BP algorithm. However, in many portable wireless applications, low-complexity designs are desired for less battery consumption. In that scenario, UIT is relevant and provides insight about the performance of the compared schemes.

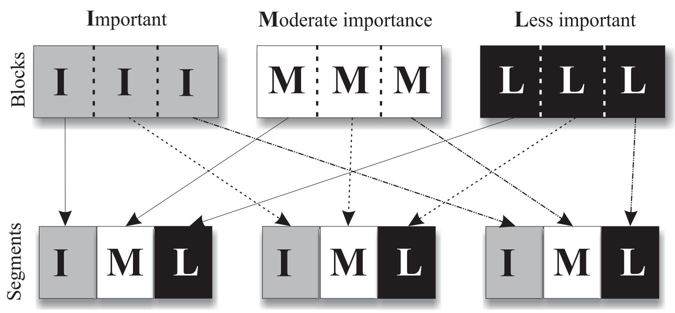


Fig. 4. Demultiplexing of the source bit stream.

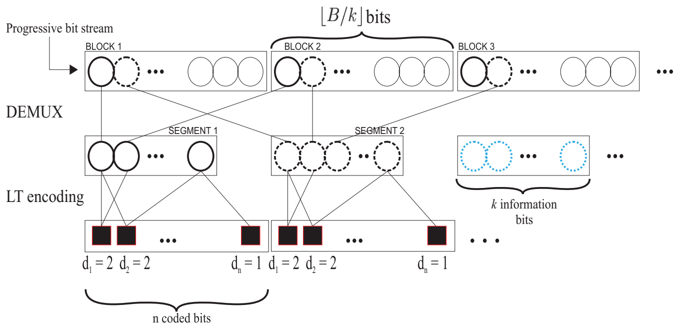


Fig. 5. Each information block is of size $\lfloor B/k \rfloor$ information bits. The zth k-bit segment is generated as shown i.e., the zth bit of each information block is collected. After forming each k-bit segment as described, a realization of the LT coding is generated and the same realization is applied to each k-bit segment. An example realization is shown in this figure.

IV. Progressive Source Transmission System and Design Parameter Selection

A. Progressive Source Transmission System Description

Our previous descriptions are based on nodes containing bits. In most practical multimedia transmission systems, the basic unit of information is usually a fixed or variable length packet [25]. In this section, we describe the way we transmit a progressive bit stream using rateless codes. The system block diagram of our proposed setup is shown in Fig. 3.

A bit stream is produced using a $L_x \times L_y$ grayscale image coded with set partitioning in hierarchical trees 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 (b/pixel). First, the source bit stream is divided into equal size blocks of bits. Then, the bits in those blocks are rearranged as shown in Fig. 4 to produce equal size segments. Using such a configuration, for instance, the initial part of each block constitutes the contents of the first segment. Thus, the total source bit stream is demultiplexed to produce a set of equal size segments. Those segments are the output of the DEMUX block in Fig. 3. To be more specific, the progressive bit stream of B bits is divided into k equal size blocks (each with $\lfloor B/k \rfloor$ information bits) as shown in Fig. 5. Those information blocks are demultiplexed into $\lfloor B/k \rfloor$ equal size segments (each with k bits) in the following way. The zth k-bit segment is generated by collecting the zth bit of each information block. The reason for using such a demultiplexing methodology is that the proposed coding scheme is most powerful when the source bits within each segment have unequal importance. Using demultiplexing, for example, the information bits in the first block, the most important information block, are equally shared by the segments. In contrast, we could have skipped the demultiplexing, i.e., we could have treated the source blocks directly as our segments (without the rearrangement shown in Fig. 4) before LT encoding takes place. However, in that case, each k-bit information segment would include almost equally-significant content.

After demultiplexing, we generate a particular realization of the proposed DF code and apply it to each k-bit segment to produce coded symbols. Assuming that each coded symbol stream goes through the same erasure channel, we collect n coded symbols at the receiver for each k-bit segment decoding. Note that since we apply the same realization of the random code to each k-bit segment and use independent decoding, if m information bits are useful in each segment after each BP decoding (as shown in Fig. 3), because of the rearrangement/demultiplexing operation, we will have $\lfloor B/k \rfloor m$ total number of useful bits in the progressive bit stream for source decoding.

B. Comparison With the "Weighted Approach"

It is easy to see that the "weighted approach" is a special case of the proposed UEP GLT coding given in Algorithm 1. Since encoding–decoding is done according to two interrelated distributions (SD and DD), the design criterion in our case is to select both distributions judiciously to minimize the average distortion, as given by (3). To reduce the number of optimization parameters, let us choose $p_{j,i}$ to be an exponential function of the degree number i, for $j = 1, 2, \ldots, r-1$ as follows.

Definition 7—Exponential SD:

$$1)\quad p_{j,i} = A_j + B_j \times \exp\left\{-\frac{i-1}{C_j}\right\} \text{ for } i = 1, 2, \ldots, k \quad (1)$$

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.

The exponential SD is an intuitive choice, because the low degree check nodes make, on average, more edge connections with the more important information sets. Note that using the exponential SD, we reduce the parameter space size from $(r-1)k+k-1$ to $3(r-1)+k-1$. Let us use standard DDs (for example, the robust soliton distribution) and a predetermined partitioning set $\{\alpha_1, \ldots, \alpha_r\}$ in conjunction with an exponential SD, so that we will have only $3(r-1)$ parameters subject to optimization. Note that an additional optimization can be run over the partitioning set $\{\alpha_1, \ldots, \alpha_r\}$. In that case, the parameter space size increases to $3(r-1)+r-1 = 4(r-1)$. As will be shown in the numerical results section, this will lead to a slightly better performance at the expense of increased complexity.

C. Comparison With UEP EWF Codes

In EWF codes [17], one of the expanding windows is first selected by a coded symbol before the selection of its edge connections. After choosing a specific window, all the edge connections of that coded symbol are constrained to be chosen from the selected window. In our UEP GLT coding process (Algorithm 1), the edge connections are not constrained in that way. Thus, although the EWF code is not a special case of the proposed UEP GLT code given in Algorithm 1, it is a special case of Algorithm 2.

Let us use a DD called the *compound DD* $\Lambda^c(x)$, given as follows.

Definition 8—Compound DD:

$$1)\quad \Lambda^c(x) = \sum_{i=1}^{k} \Lambda_i^c x^i, \quad \Lambda_i^c \triangleq \sum_{j=1}^{r} \rho_j \Phi_i^{(j)}$$

where 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 $\sum_{j=1}^{r} \rho_j = 1$.

It can be shown that the compound DD is a valid probability mass function by realizing that it is a convex combination of probability mass functions.

The main motivation behind using such a distribution is the following. In a progressive transmission, recovery of the whole source block is not usually the key objective. The goal is to maximize the number of useful source symbols. Thus, the source block size that can be communicated to the receiver with small error probability is the outcome of the optimization. Using a compound DD enables us to tailor a set of DDs (specifically good for a set of source block sizes) for our progressive source transmission scenario. Using appropriate weights ρ_j, we can make the compound DD a good DD for a specific information block size and a source whose rate-distortion (R-D) characteristic is known. Finally, the parameters of the compound DD are fed to the optimization to minimize the distortion.

Similar to previous reasoning, in order to reduce the number of optimization parameters, we choose $\gamma_{j,i}$ to be an exponential function of the degree number i for $j = 1, 2, \ldots, r-1$ as follows.

Definition 9—Exponential Window SD:

For $i = 1, 2, \ldots, k$

$$1)\quad \gamma_{j,i} = \begin{cases} \overline{A}_j + \overline{B}_j \times \exp\left\{-\frac{i-1}{\overline{C}_j}\right\}, & \text{if } i \leq k \sum_{t=1}^{j} \alpha_t \\ 0, & \text{if } i > k \sum_{t=1}^{j} \alpha_t \end{cases} \quad (2)$$

where $\{\overline{A}_j, \overline{B}_j, \overline{C}_j\}_{j=1}^{r-1}$ are design parameters satisfying $\sum_{j=1}^{r} \gamma_{j,i} = 1$ for all i.

D. Optimization Problem

The BP algorithm can terminate at any iteration with some non-zero probability. Let $Pr(Y = s)$ be the probability that the BP algorithm terminates at iteration s. In this type of algorithm, one typically chooses the maximum number of iterations of the BP algorithm ($M_{\max}$) such that $Pr(Y > M_{\max})$ is negligible. $M_{\max}$ is usually chosen based on the value of k, the number of information symbols being encoded in all the simulations. In our case, we chose $M_{\max} = 70$ because the algorithm always terminated (either by correct decoding or by having a decoding failure) prior to 70 iterations being reached. The optimization problem is given by

$$\min_{\boldsymbol{\lambda},\mathbf{P}(\text{or }\mathbf{L})} \overline{\mathfrak{D}}_M \text{ s.t. } n \text{ coded symbols are received unerased} \quad (3)$$

where $n \geq k$, and $\overline{\mathfrak{D}}_M$ is the average mean square distortion at the Mth iteration of the BP algorithm. Note that the optimization problem for Algorithm 2 is exactly the same, except that we replace $\mathbf{P}$ with $\mathbf{L}$. 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 minimization is over the entries of $\boldsymbol{\lambda}$ and $\mathbf{P}$ (or $\mathbf{L}$). Note that the proposed code is specified by the entries of two matrices, $\boldsymbol{\lambda}$ and $\mathbf{P}$ (or $\mathbf{L}$), and the total number of entries is $(r+1)k$. However, the columns of $\mathbf{P}$ (or $\mathbf{L}$) as well as the entries of $\boldsymbol{\lambda}$ should sum up to one. Therefore, we have $(r-1)k + (k-1) = rk - 1$ parameters subject to optimization. For large k, as a practical matter, it is infeasible to jointly optimize all design parameters. Previous sections discussed how we reduce the number of parameters subject to optimization. We use numerical experimentation and exhaustive search to find the optimum solution.

V. Numerical Results

In this section, we will compare the performance of our proposed scheme with that of two major classes of unequal protection rateless codes: "weighted approach" and UEP EWF codes. We use the minimum distortion criterion throughout our simulations. Quality assessment of the decoded images is given 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 BP algorithm, $\overline{\mathfrak{D}}_M$. We use standard 512×512 *Lena* and 512×512 *Goldhill* images.

We initially set $B = 50\,000$ bits ($\approx$0.19 b/pixel source rate) and run all realizations (encoding and decoding using $M_{\max} = 70$) 10^4 times, and compute the total number of useful bits (number of recovered consecutive information symbols) in each realization. We considered two different values for k: $k = 100$ and $k = 1000$. Since, a specific image with a particular source encoder determines the R-D characteristics of the source, we can find the distortion[1] corresponding to the useful number of bits. Then, we take the average of these distortion values computed for each of the 10^4 different realizations. We optimize the parameters of each system to give the minimum average distortion (the solution to the optimization problem). Last, minimum average distortion is converted to average PSNR.

[1]We use mean square error as our distortion metric throughout this paper.

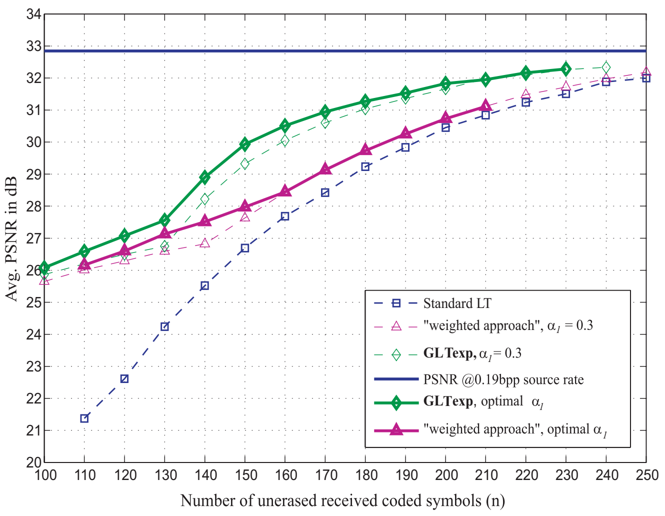


Fig. 6. Performance comparisons using *Lena* and $k = 100$.

A. Comparisons With the "Weighted Approach"

In our first simulation, we compare our UEP GLT scheme using Algorithm 1 with the weighted approach. We set $r = 2$ and $B = 5 \times 10^4$. Both schemes use the RSD with $\gamma = c = 0.01$ i.e., $\Upsilon_w(x)$ is an RSD in Algorithm 1. We define the following version of the proposed scheme.

1) *GLTexp:* This scheme uses the exponential SD with $A_1 + B_1 = 1$ and optimizes the set $\{A_1, C_1\}$ so that the proposed scheme achieves minimum distortion.

For a fair comparison, we also optimize the weighting parameter ω_1 for the weighted approach for minimum distortion. Average PSNR versus the number of reliably received coded symbols (n) is plotted for both systems using a fixed $\alpha_1 = 0.3$ and the optimal α_1 value for each system. For the weighted approach, the optimal α_1 (α_1^*) turns out to be 0.3 for $n \geq 160$. We also included the performance of standard LT coding (EEP scheme) in the same figure for comparison. In Figs. 6 and 7, 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 $n = 130$ coded symbols, it provides over a 1-dB improvement for a substantial range of n (from 140 to 210) for both fixed or optimal α_1, and a huge improvement over standard LT coding. 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 $\approx$0.19 b/pixel, and which is operating under error-free conditions, is included in the figures for reference.

In Fig. 8, we show the performance of these systems for $k = 1000$. As can be seen, when k gets large, the performance of each system increases, as does the gain of the proposed scheme over the weighted approach. It is quite common to define the coding overhead of a rateless code as $\epsilon \triangleq (n/k) - 1$. Using the

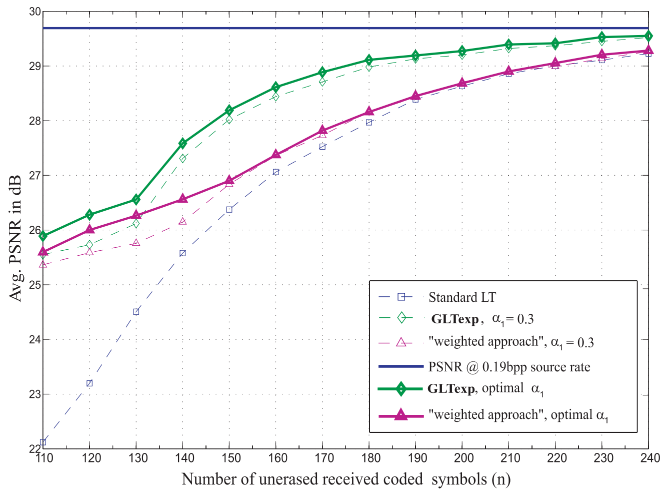


Fig. 7. Performance comparisons using *Goldhill* and $k = 100$.

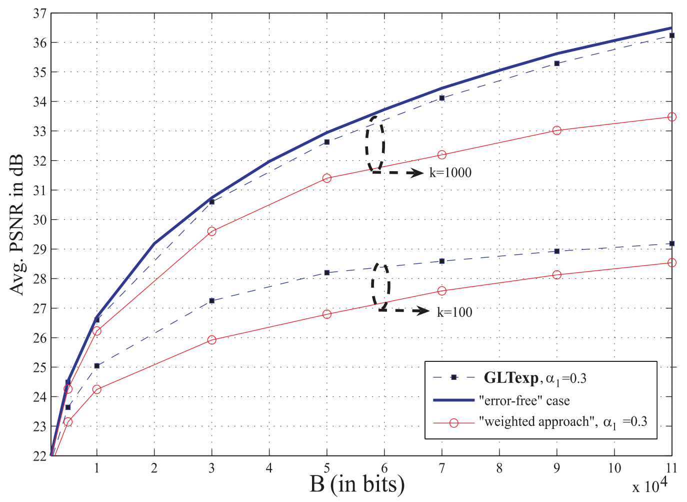


Fig. 9. Performance comparisons using *Lena* and a overhead of $\epsilon = 0.4$ for a range of B.

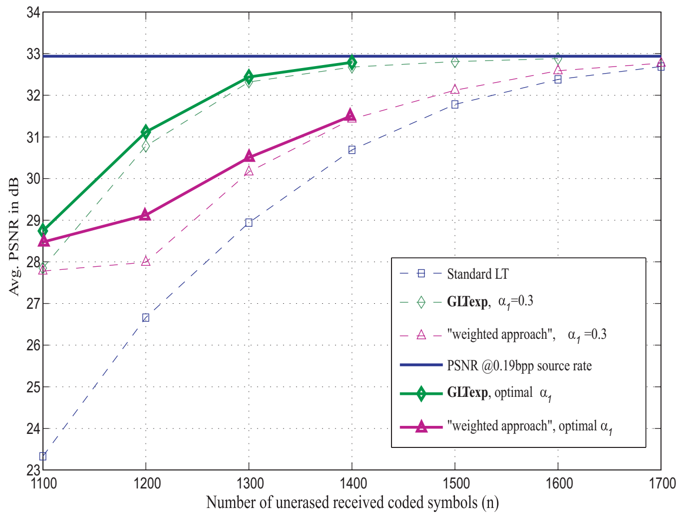


Fig. 8. Performance comparisons using *Lena* and $k = 1000$.

TABLE I
NUMBER OF CODED SYMBOLS, WHICH NEED TO BE RECEIVED RELIABLY FOR A PSNR OF 31 dB USING *LENA*

	$k = 100$		$k = 1000$	
	n	Savings	n	Savings
GLTexp	171	43	1196	229
weighted approach	209	5	1350	75
standard LT	214	–	1425	–

TABLE II
OPTIMAL PARAMETERS OF VARIOUS UEP DESIGNS SHOWN IN FIGS. 6 AND 8

$k = 100$, $\epsilon = 0.5$	A_1	B_1	C_1	α_1	PSNR (dB)
"weighted approach"	0.55	0.0	N/A	0.3	27.61
"weighted approach" (α_1^*)	0.95	0.0	N/A	0.5	28.05
GLTexp	0.19	0.81	2.0	0.3	29.19
GLTexp (α_1^*)	0.06	0.94	0.9	0.1	29.82
$k = 1000$, $\epsilon = 0.3$	A_1	B_1	C_1	α_1	PSNR (dB)
"weighted approach"	0.45	0.0	N/A	0.3	29.63
"weighted approach" (α_1^*)	0.76	0.0	N/A	0.6	30.39
GLTexp	0.17	0.83	1.9	0.3	32.23
GLTexp (α_1^*)	0.25	0.75	1.2	0.25	32.46

proposed scheme with *Lena*, for example, an overhead of 0.3 when $k = 1000$ gives around 32.5-dB average PSNR, whereas the same overhead with $k = 100$ gives around 27.6-dB average PSNR. Another way of looking at these performance curves is to consider the percentage of savings of the coded symbols reliably received for a given image quality. Table I shows the number of coded symbols which need to be received to obtain a PSNR of 31 dB using various rateless code schemes. As can be seen, a substantial savings (relative to standard LT coding) in terms of the received unerased coded symbols can be obtained. Also, Table II shows the optimal parameters used to obtain some of the data points shown in Figs. 6 and 8.

Note that Figs. 6–8 can all 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).

Fig. 9 shows performance comparisons for a range of B and $\epsilon = 0.4$. As a perspective on PSNR, assume that all the source bits are recovered. We call this idealized scenario the "error-free" case. We observe that the proposed scheme not only improves the performance over the weighted approach, but also gives results that are close to the "error-free" case. One other observation is that for $k = 100$, an improvement of almost 1 dB is possible for a range of B (from $B \approx 2 \times 10^4$ to $B \approx 6 \times 10^4$). For the range of values shown on the abscissa, increasing B yields larger gains when $k = 1000$, but not when

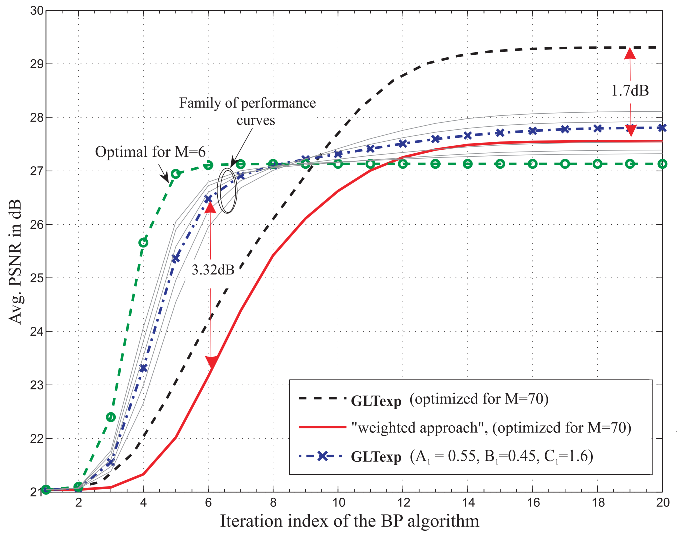


Fig. 10. UIT performance comparisons using *Lena* for $n = 150$, $\alpha_1 = 0.3$, and $B = 50\,000$ bits.

$k = 100$. The reason is related to the system description and the assumption made in Section A. A closer look at Fig. 5 reveals that k also equals the number of blocks, and for a fixed k, the size of the information blocks increases with growing B. For large B and small k, the contents of each segment might be quite different. For example, if $k = 2$, the first segment will have one bit from the beginning of the total bit stream and one bit from the middle. The last segment will have one bit from the middle of the bit stream and one bit from the end. Since B is large and the progressive coder has a non-increasing and convex R-D characteristic, these two segments are substantially different from each other. Considering the progressive source transmission system shown in Fig. 3, we note that the same code is applied for each segment. Therefore, a set of optimal values tailored to a particular segment might not be the best set of parameters for other segments. This degrades the performance of the system. If k gets large, the R-D characteristics of each segment become similar, and thus the parameters of the proposed code fit a larger set of segments. We note that the progressive source transmission system shown in Fig. 3 assumes that one segment contains almost equally important content compared to another segment, although the importance varies considerably within each segment. In Fig. 9, for the parameters of this scenario, $k = 100$ turns out to be too small for this assumption to hold; the k-bit segments differ from each other in the importance of their information content. However, if we increase k from 100 to 1000, the assumption of Fig. 3 will approximately hold and segments become almost equally important and have similar R-D characteristics. Thus, the proposed scheme gives increased gains at larger B values, as shown in Fig. 9.

Finally, 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. We note that the previous results are based on solving the minimization problem in (3) for $M = 70$. However, the parameters that give the best performance for $M = 70$ do not necessarily give the best performance at early iterations. A simulation result is shown in Fig. 10, in which 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. A 1.7-dB PSNR gain when the number of iterations is greater than 16 is shown over the weighted approach. Fig. 10 also shows a performance curve that results by solving the minimization problem in (3) for $M = 6$. 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. The performance curves optimized for $M = 6$ and $M = 70$ suggest that there is a family of performance curves that give better PSNR gains at early iterations, if we allow some performance loss at later iterations. For example, in Fig. 9 we show a family of curves obtained by varying parameter C_1 from 1.2 to 2.2 with fixed $\{A_1 = 0.55, B_1 = 0.45\}$. From this family of curves, we show one of them with $C_1 = 1.6$. It can be observed that, although this sample system performs worse than the system optimized for $M = 70$, it performs better than the weighted approach for all iterations, but especially for some early iterations. For example, at iteration 6, it provides a 3.32-dB gain over the weighted approach. 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.

[2]By natural termination, we mean 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 though the decoding of the whole source block is not complete.

B. Comparisons With UEP EWF Codes

We now compare the proposed generalization i.e., Algorithm 2, with UEP EWF codes. We also compare the proposed scheme with increased parameter sizes. The EWF code has the parameters α_1 and Γ_1 subject to optimization. The DDs for the UEP EWF code, $\{\Phi^{(1)}(x), \Phi^{(2)}(x)\}$ are the *truncated RSD* [17] $\Omega_{rs}(k_{rs}, \gamma, c)$ 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 $W_2 = k$ [19]. Details of the design parameters of the comparison systems are summarized in Table III. The UEP GLT scheme given in Algorithm 2 uses the compound DD i.e., $\Upsilon_{\text{ewf}}(x)$ is $\Lambda^c(x)$. In other words, for any coded symbol, the probability of choosing degree i is given by

$$\Lambda_i^c = \rho_1 \Phi_i^{(1)} + (1 - \rho_1)\Phi_i^{(2)} \tag{4}$$

where $\Phi_i^{(1)} = 0$ for $i > \alpha_1 k$. Let $\overline{\alpha}_1^{\text{ewf}}$ and $\overline{\Gamma}_1^{\text{ewf}}$ be the optimum parameters of the EWF code in the minimum-distortion sense for our progressive transmission scenario. For GLTexp, for all n, we set $\alpha_1 = \overline{\alpha}_1^{\text{ewf}}$ and $\rho_1 = \overline{\Gamma}_1^{\text{ewf}}$. In addition to GLTexp, we define two other versions of the proposed UEP GLT with a larger size of the parameter set subject to optimization.

1) *GLTexpOpt:* This scheme uses the exponential SD with $\overline{A}_1 + \overline{B}_1 = 1$. It optimizes the set $\{\alpha_1, \rho_1, \overline{A}_1, \overline{C}_1\}$ so that

TABLE III
PARAMETERS OF THE UEP GLT AND COMPARISON OF UEP DF DESIGNS

UEP schemes	r	Optimized parameters	DD	SD
Standard LT	1	N/A	$\Omega_{rs}(k, \gamma, c)$	Uniform
UEP EWF	2	α_1, Γ_1	$(\Phi^{(1)}(x), \Phi^{(2)}(x)) = (\Omega_{rs}(\alpha_1 k, \gamma, c), \Omega_{rs}(k, \gamma, c))$	Uniform within the window
UEP GLT	2	$\alpha_1, \rho_1, \overline{A}_1, \overline{B}_1, \overline{C}_1$	$\Lambda^{(c)}(x)$	Exponential window

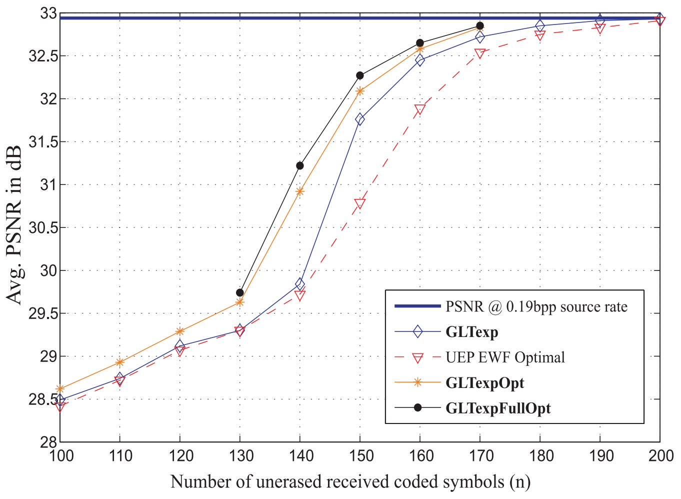


Fig. 11. Performance comparisons with EWF codes using *Lena* with $k = 100$.

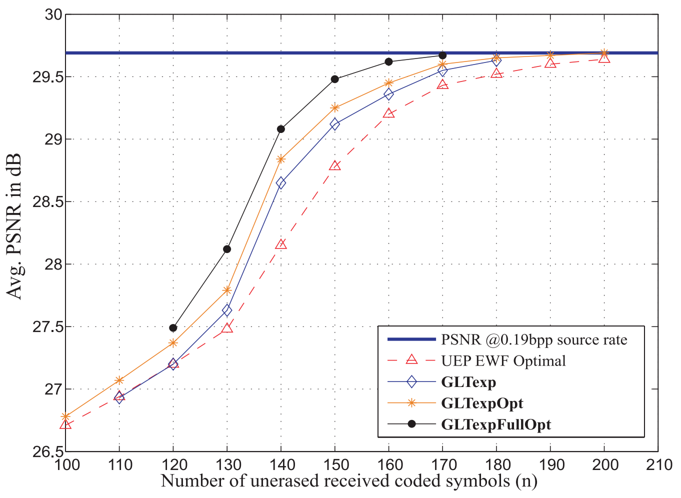


Fig. 12. Performance comparisons with EWF codes using *Goldhill* with $k = 100$.

the proposed scheme achieves the minimum distortion.

2) *GLTexpFullOpt:* This scheme uses the exponential SD and optimizes the whole set of parameters, i.e., (α_1, ρ_1, $\overline{A}_1$, $\overline{B}_1$, $\overline{C}_1$) so that the proposed scheme achieves the minimum distortion.

Figs. 11 and 12 compare these unequal protection schemes with $k = 100$ in terms of the average PSNR as a function of n, the number of unerased symbols. As we increase the parameter space of the proposed UEP GLT scheme as described, we observe more sizable improvements in a progressive transmission scenario. For example, GLTexp optimizes only two parameters of the exponential SD and gives

TABLE IV
OPTIMAL PARAMETERS OF VARIOUS UEP DESIGNS FOR VARIOUS k AND ϵ

k = 100, $\epsilon = 0.4$	$\overline{A}_1$	$\overline{B}_1$	$\overline{C}_1$	α_1	Γ_1	PSNR(dB)
UEPEWF	N/A	N/A	N/A	0.5	0.97	29.63
GLTexpOpt	−0.62	1.62	1.1	0.24	0.55	30.94

k = 1000	ϵ	$\overline{A}_1$	$\overline{B}_1$	$\overline{C}_1$	α_1	Γ_1	PSNR(dB)
UEPEWF	0.1	N/A	N/A	N/A	0.65	0.95	30.27
	0.2	N/A	N/A	N/A	0.68	0.92	30.78
	0.3	N/A	N/A	N/A	0.2	0.15	31.63
GLTexpOpt	0.1	−0.94	1.94	1.1	0.36	0.52	31.23
	0.2	−0.7	1.7	0.8	0.16	0.55	31.71
	0.3	−0.7	1.7	0.9	0.15	0.55	32.39

some improvement over the UEP EWF code. Note that both GLTexp and the UEP EWF code optimize two parameters. If we increase the parameter space subject to optimization, the relative gains over the UEP EWF code performance increase. For example, in Fig. 11, GLTexpOpt uses four parameters, and GLTexpFullOpt uses five parameters to minimize the source reconstruction distortion, giving 1.3- and 1.53-dB average PSNR gains, respectively, compared to the UEP EWF code when $\epsilon = 0.4$. Table IV shows some of the optimal parameters used to obtain the performance curves in Fig. 11.

VI. CONCLUSION

Fountain codes are a type of erasure-correcting code with simple encoding and decoding structures used both for point-to-point communications and for multicasting information. The initial design of such codes was 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 paper, 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 a progressive transmission scheme using this generalized version of UEP LT codes. We compared the proposed scheme with two other major UEP LT codes in the literature for the progressive transmission scenario. Simulations show that the proposed coding scheme outperforms these previous schemes by providing improved UEP, URT, and UIT properties. As a future work, we plan to investigate the asymptotic analysis of the proposed generalization.

REFERENCES

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

[2] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

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

[4] *Universal Mobile Telecommunications System (UMTS): Multimedia Broadcast/Multicast Service (MBMS): Protocols and Codecs*, Standard ETSI TS 126 346 V8.1.0, 2005.

[5] 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, no. 3, pp. 243–250, Jun. 1996.

[6] L. Xu, "Resource-efficient delivery of on-demand streaming data using UEP codes," *IEEE Trans. Commun.,* vol. 51, no. 1, pp. 63–71, Jan. 2003.

[7] P. G. Sherwood and K. Zeger "Progressive image coding for noisy channels," *IEEE Signal Process. Lett.,* vol. 4, no. 7, pp. 189–191, Jul. 1999.

[8] T. Thomos, N. V. Boulgouris, and M. G. Strintzis, "Wireless image transmission using turbo codes and optimal unequal error protection," *IEEE Trans. Image Process.,* vol. 14, no. 11, pp. 1890–1901, Nov. 2005.

[9] X. Pan, A. H. Banihashemi, and A. Cuhadar, "Combined source and channel coding with JPEG2000 and rate-compatible low-density parity-check codes," *IEEE Trans. Signal Process.,* vol. 54, no. 3, pp. 1160–1164, Mar. 2006.

[10] S. S. Arslan, P. C. Cosman, and L. B. Milstein, "Concatenated block codes for unequal error protection of embedded bit streams," *IEEE Trans. Image Process.,* vol. 21, no. 3, pp. 1111–1122, Mar. 2012.

[11] A. Nosratinia, J. Lu, and B. Aazhang, "Source-channel rate allocation for progressive transmission of images," *IEEE Trans. Commun.*, vol. 51, no. 2, pp. 186–196, Feb. 2003.

[12] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory,* vol. 52, no. 6, pp. 2410–2423, Jun. 2006.

[13] P. Maymounkov, "Online codes," Secure Computer Systems Group, New York Univ., New York, Tech. Rep. TR2002-833, 2002.

[14] N. Rahnavard and F. Fekri, "Finite-length unequal error protection rateless codes: Design and analysis," in *Proc. IEEE Global Telecommun. Conf.*, Nov.–Dec. 2005, pp. 1–5.

[15] 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, Apr. 2007.

[16] S. K. Chang, K. C. Yang, and J. S. Wang, "Unequal-protected LT code for layered video streaming," in *Proc. IEEE Int. Conf. Commun.*, Beijing, China, Jun. 2008, pp. 500–504.

[17] 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.

[18] J. Wagner, J. Chakareski, and P. Frossard, "Streaming of scalable video from multiple server using rateless codes," in *Proc. IEEE Int. Conf. Multimedia Exposit.*, Toronto, ON, Canada, Jul. 2006, pp. 1501–1504.

[19] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Scalable video multicast using expanding window fountain codes," *IEEE Trans. Multimedia*, vol. 11, no. 6, pp. 1094–1104, Oct. 2009.

[20] M. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, "Sliding window digital fountain codes for streaming of multimedia applications," in *Proc. IEEE Int. Symp. Circuits Syst.*, New Orleans, LA, May 2007, pp. 3467–3470.

[21] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Unequal error protection using LT codes and block duplication," in *Proc. Middle Eastern Multi-conf. Simul. Model.*, Aug. 2008, pp. 1–5.

[22] S. S. Woo and M. K. Cheng, "Prioritized LT codes," in *Proc. 42nd Annu. Conf. Inf. Sci. Syst.*, Princeton, NJ, 2008, pp. 568–573.

[23] T. Schierl, T. Stockhammer, and T. Wiegand, "Mobile video transmission using scalable video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1204–1217, Sep. 2007.

[24] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[25] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, 2nd ed, Reading, MA: Addison-Wesley, 2002.

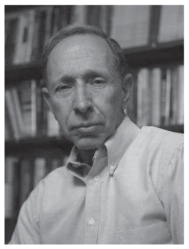
Suayb S. Arslan (S'06) received the B.S. degree in electrical and electronics engineering from Bogazici University, Istanbul, Turkey, in 2006, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, San Diego, in 2009 and 2012, respectively.

He was with Mitsubishi Electric Research Laboratory, Boston, MA, in 2009, where he was involved in research and development of image and video processing algorithms for biomedical applications. In 2011, he joined Quantum Corp., Irvine, CA, where he conducted research on advanced detection algorithms and post processing for increased capacity tape drives. His current research interests include wireless/wireline digital multimedia communication and storage, joint source-channel coding, information theory, image/video processing, and cross layer design optimizations.

Pamela C. Cosman (S'88–M'93–SM'00–F'08) received the B.S. degree (Hons.) in electrical engineering from the California Institute of Technology, Pasadena, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1989 and 1993, respectively.

She was an NSF Post-Doctoral Fellow with Stanford University and a Visiting Professor with the University of Minnesota, Minneapolis, from 1993 to 1995. In 1995, she joined the Faculty of the Department of Electrical and Computer Engineering, University of California at San Diego (UCSD), San Diego, where she is currently a Professor. She was the Director of the Center for Wireless Communications, UCSD, from 2006 to 2008. Her current research interests include image and video compression and processing and wireless communication.

Dr. Cosman was a Guest Editor of the June 2000 special issue of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS on error-resilient image and video coding and the Technical Program Chair of the Information Theory Workshop in San Diego in 1998. She was an Associate Editor of the IEEE COMMUNICATIONS LETTERS from 1998 to 2001 and the IEEE SIGNAL PROCESSING LETTERS from 2001 to 2005. She was the Editor-in-Chief from 2006 to 2009, as well as a Senior Editor from 2003 to 2005 and from 2010 to 2012, of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. She is a member of Tau Beta Pi and Sigma Xi. She was a recipient of the ECE Departmental Graduate Teaching Award, a Career Award from the National Science Foundation, a Powell Faculty Fellowship, and a Globecom Best Paper Award in 2008.



Laurence B. Milstein (S'66–M'68–SM'77–F'85) received the B.E.E. degree from the City College of New York, New York, in 1964, and the M.S. and Ph.D. degrees in electrical engineering from the Polytechnic Institute of Brooklyn, Brooklyn, NY, in 1966 and 1968, respectively.

He was with the Space and Communications Group, Hughes Aircraft Company, Culver City, CA, from 1968 to 1974. From 1974 to 1976, he was a member of the Department of Electrical and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. Since 1976, he has been with the Department of Electrical and Computer Engineering, University of California at San Diego, San Diego, where he is currently the Ericsson Professor of Wireless Communications Access Techniques and a former Department Chairman, working in the area of digital communication theory with special emphasis on spread-spectrum communication systems. He has also been a consultant to both government and industry in radar and communications.

Dr. Milstein was an Associate Editor for Communication Theory for the IEEE TRANSACTIONS ON COMMUNICATIONS and Book Reviews for the IEEE TRANSACTIONS ON INFORMATION THEORY, an Associate Technical Editor for the IEEE *Communications Magazine*, and the Editor-in-Chief of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He was the Vice President for Technical Affairs of the IEEE Communications Society in 1990 and 1991, and a former Chair of the IEEE Fellows Selection Committee. He was a recipient of the Military Communications Conference Long Term Technical Achievement Award in 1998, an Academic Senate UCSD Distinguished Teaching Award in 1999, an IEEE Third Millennium Medal in 2000, the IEEE Communication Society Armstrong Technical Achievement Award in 2000, and various prize paper awards, including the MILCOM Fred Ellersick Award in 2002.