Multi-Edge Framework for Unequal Error Protecting LT Codes

H. V. Beltrão Neto, W. Henkel
Jacobs University Bremen
Campus Ring 1
D-28759 Bremen, Germany
Email: {h.beltrao, w.henkel}@jacobs-university.de

V. C. da Rocha Jr.
Department of Electronics and Systems
Federal University of Pernambuco
P.O. Box 7800 Recife, Brazil, 50711-970
Email: vcr@ufpe.br

***Abstract*—A multi-edge framework for unequal error protecting (UEP) LT codes is derived by distinguishing between the edges connected to each protection class. Under the framework introduced, two existing techniques for the design of unequal error protecting LT codes can be evaluated and explained in a unified way. Furthermore, a simple and flexible design technique is proposed for UEP LT codes with good performance.**

I. Introduction

First introduced by Luby in [1], LT codes form together with Raptor [2] and Online codes [3] the class of the so-called rateless codes. Rateless codes are particularly suitable for transmitting through channels that can be represented by the binary erasure channel (BEC) with unknown erasure probabilities, a situation where traditional codes turn out to be suboptimal. Rateless codes are also very interesting for multicast transmission since they eliminate the requirement for retransmission.

The encoding algorithm of LT codes can be described as follows. Suppose one intends to transmit a message composed of k input symbols. Each output symbol is formed by first determining its degree i according to a probability distribution $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$, where Ω_i denotes the probability of i being chosen. The output symbol is then formed choosing i input symbols uniformly and at random and performing an XOR operation on them. The process is repeated until a sufficient number of output symbols $n = \gamma k$ arrives at the receiver. The quantity γ is called the overhead. We can depict the encoding procedure as a bipartite graph with k variable nodes and n check nodes. By means of this representation, the decoding algorithm of LT codes can be easily described as an instance of the belief propagation (BP) algorithm [4] on the graph induced by the encoding. In this article, we focus on LT codes developed for systems where the source bits being transmitted have different sensitivities to errors. In such systems, it is often wasteful or even infeasible to provide uniform protection for the whole data stream raising the interest in codes with unequal error protection capabilities.

The development of unequal error protecting (UEP) rateless codes was first presented by Rahnavard et al. in [5], where the authors propose the partitioning of the block to be transmitted into protection classes with symbols on distinct classes having different probabilities of being chosen after the degree i is determined. Another UEP scheme was presented in [6] where the authors achieve UEP properties by means of a windowing technique. In this paper, we show that these two schemes can be interpreted as particular cases of multi-edge type unequal error protecting LT codes, what provides a common framework for comparison of both schemes. Furthermore, we propose a simple and flexible technique for generating UEP LT codes with good performance.

The paper is organized as follows. In Section II we present the multi-edge description of UEP LT codes and derive its multi-edge degree distributions. Section III contains a short description of existing construction techniques for UEP LT codes and its multi-edge type analysis. Further in the same section, we propose a novel technique for reaching unequal error protection with LT codes. The asymptotic analysis of multi-edge UEP LT codes is presented in Section IV and some simulation results in Section V. Section VI finalizes the paper with some concluding remarks.

II. Multi-edge unequal error protecting LT codes

The multi-edge framework was originally derived for low-density parity-check codes (LDPC) in [7]. In the multi-edge setting, several edge classes can be defined within the bipartite graph induced by the encoding and thus, every node is characterized by the number of connections to edges of each class.

A. Multi-edge description of bipartite graphs

A multi-edge type analysis allows a bipartite graph to be specified through two multinomials associated to variable and check nodes. Let the two multinomials be defined by

$$L(\mathbf{x}) = \sum_{\mathbf{d}} L_{\mathbf{d}} \mathbf{x}^{\mathbf{d}} \quad \text{and} \quad R(\mathbf{x}) = \sum_{\mathbf{d}} R_{\mathbf{d}} \mathbf{x}^{\mathbf{d}} \,, \tag{1}$$

where $\mathbf{d}$ and $\mathbf{x}$ are vectors which are explained as follows. First, let m_e denote the number of edge types used to represent the graph ensemble. Each node in the ensemble graph has associated to it a vector $\mathbf{x} = (x_1, ..., x_{m_e})$ that indicates the different types of edges connected to it, and a vector $\mathbf{d} = (d_1, ..., d_{m_e})$ referred to as *edge degree vector* which denotes the number of connections of a node to edges of type i, where

978-1-4577-0436-9/11/$26.00 ©2011 IEEE

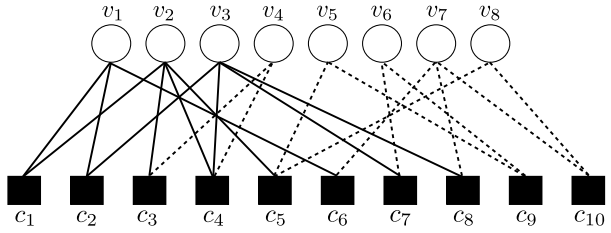

Fig. 1. Multi-edge graph with two different edge types for an LT code with $k = 8$ and $\gamma = 10/8$.

$i \in (1, \ldots, m_e)$. We use $\mathbf{x}^{\mathbf{d}}$ to denote $\prod_{i=1}^{m_e} x_i^{d_i}$. Finally, the coefficients $L_{\mathbf{d}}$ and $R_{\mathbf{d}}$ are non-negative reals such that $L_{\mathbf{d}}k$ and $R_{\mathbf{d}}k$ represent the number of variable and check nodes of type[1] $\mathbf{d}$, respectively. Furthermore, we have the additional notations defined in [8]

$$L_{x_j}(\mathbf{x}) = \frac{\partial L(\mathbf{x})}{\partial x_j} \quad \text{and} \quad R_{x_j}(\mathbf{x}) = \frac{\partial R(\mathbf{x})}{\partial x_j} \, . \tag{2}$$

Note that in a valid multi-edge ensemble, the number of connections of all edge types should be the same at both variable and check nodes sides. This gives rise to the *socket count equality* constraint which can be written as

$$L_{x_j}(\mathbf{1}) = R_{x_j}(\mathbf{1}), \quad j = 1, ..., m_e \, , \tag{3}$$

where $\mathbf{1}$ denotes a vector with all entries equal to 1, with length being clear from the context.

B. Multi-edge UEP LT codes: graph representation and degree distributions

Unequal error protecting LT codes can be included in a multi-edge framework in a straightforward way. This can be done by distinguishing between the edges connected to different protection classes within a codeword. According to this strategy, the edges connected to variable nodes within a protection class are all of the same type. For example, in Fig. 1, we divide the variable nodes into two classes. The first three variable nodes belong to the first class and are connected only to type-1 edges (depicted by solid lines), and the last five variable nodes compound another protection class, connected to only type-2 edges (depicted by dashed lines). It is worth noting that in contrast to the variable nodes (input symbols), the check nodes (output symbols) admit connections with edges of different types simultaneously as can be inferred from Fig. 1. The multi-edge degree distributions for the code depicted in Fig. 1 are:

$$L(\mathbf{x}) = \frac{1}{8}x_1^3 + \frac{2}{8}x_1^4 + \frac{4}{8}x_2^2 + \frac{1}{8}x_2^3 \, , \tag{4}$$

$$R(\mathbf{x}) = \frac{2}{8}x_1^2 + \frac{4}{8}x_1x_2 + \frac{1}{8}x_1^2x_2 + \frac{1}{8}x_1x_2^2 + \frac{2}{8}x_2^2 \, . \tag{5}$$

In the following, we will divide the variable nodes into m_e protection classes $(C_1, C_2, \ldots, C_{m_e})$ with monotonically decreasing levels of protection.

[1]We will frequently refer to nodes with edge degree vector $\mathbf{d}$ as "type $\mathbf{d}$" nodes.

1) Node perspective degree distributions: We now determine the variable and check node degree distributions $L(\mathbf{x})$ and $R(\mathbf{x})$ for UEP LT codes.

In order to determine $R(\mathbf{x}) = \sum_{\mathbf{d}} R_{\mathbf{d}}\mathbf{x}^{\mathbf{d}}$, we need to compute the fraction of check nodes of type $\mathbf{d}$, i.e., the coefficients $R_{\mathbf{d}}$. Recall that according to the encoding algorithm of LT codes, the probability of an output symbol having degree i is Ω_i (in the UEP context, we call $\Omega(x)$ the overall output symbol degree distribution). Given that the degree of an output symbol corresponds to the number of edges connected to it, i.e., $i = \sum_{j=1}^{m_e} d_j$, where d_j denotes the number of edges of type j connected to a check node, we have

$$R_{\mathbf{d}} = \Omega_i \cdot \frac{\binom{k}{i}^{-1}}{d_1! \cdots d_{m_e}!} \frac{k_1!}{(k_1 - d_1)!} \cdots \frac{k_{m_e}!}{(k_{m_e} - d_{m_e})!} \, . \tag{6}$$

Asymptotically (as k increases to infinity), we can approximate Eq. (6) by[2]

$$R_{\mathbf{d}} = \Omega_i \cdot \frac{i!}{d_1!d_2! \cdots d_{m_e}!} \omega_1^{d_1}\omega_2^{d_2} \cdots \omega_{m_e}{}^{d_{m_e}} \, , \tag{7}$$

where ω_j is the probability of an input symbol of the class C_j being chosen among the k input symbols and $i = \sum_{j=1}^{m_e} d_j$. The check node degree distribution is then given by

$$R(\mathbf{x}) = \sum_{\mathbf{d}} \Omega_i \cdot \frac{i!}{d_1!d_2! \cdots d_{m_e}!} \omega_1^{d_1}\omega_2^{d_2} \cdots \omega_{m_e}{}^{d_{m_e}} \mathbf{x}^{\mathbf{d}} \, . \tag{8}$$

For computing the variable node degree distribution $L(\mathbf{x}) = \sum_{\mathbf{d}} L_{\mathbf{d}}\mathbf{x}^{\mathbf{d}}$, first recall that according to our previous definitions, the variable nodes belonging to C_j are only connected to edges of type j. This means that the edge degree vectors $\mathbf{d}$ have only one non-zero component, e.g., for the two-class case $\mathbf{d} = (d_1, 0)$ or $(0, d_2)$.

Let L_{d_j} represent $L_{\mathbf{d}}$ when d_j is the only non-zero component of $\mathbf{d}$. By simple combinatorial arguments, the probability of a variable node of class C_j having degree d_j is given by

$$L_{d_j} = \binom{\mu_j\gamma k}{d_j} p_j^{d_j}(1 - p_j)^{\mu_j\gamma k - d_j} \, , \tag{9}$$

where $\mu_j = R_{x_j}(\mathbf{1})$ is the average number of type-j edges and, p_j is the probability of an input symbol being chosen among the k_j symbols of C_j. The variable node degree distribution can then be written as

$$L(\mathbf{x}) = \sum_{\mathbf{d},j} \binom{\mu_j\gamma k}{d_j} p_j^{d_j}(1 - p_j)^{\mu_j\gamma k - d_j} \mathbf{x}^{\mathbf{d}} \, . \tag{10}$$

Equations (8) and (10) are quite general and apply for any unequal error protecting LT code with m_e protection classes. In order to clarify the exposed concepts in a simple manner, we will from now on consider codes with only two protection classes, i.e., codes with $m_e = 2$. In this particular case, Eqs. (8) and (10) are reduced to

$$R(\mathbf{x}) = \sum_{d_1,d_2} \Omega_{d_1+d_2} \cdot \binom{d_1 + d_2}{d_1} \omega_1^{d_1}(1-\omega_1)^{d_2} \cdot x_1^{d_1}x_2^{d_2} \, , \tag{11}$$

[2]This approximation corresponds to approximating the encoding without replacement of LT codes as an encoding with replacement. This approximation becomes more precise as $k \to \infty$.

$$L(\mathbf{x}) = \sum_{j=1}^{2} \sum_{d_j=1}^{\mu_j \gamma k} \binom{\mu_j \gamma k}{d_j} p_j^{d_j} (1-p_j)^{\mu_j \gamma k - d_j} \cdot x_j^{d_j} \,. \quad (12)$$

III. Construction algorithms for unequal error protecting LT codes

Herein, we proceed to a multi-edge type analysis of the unequal error protecting LT codes presented in [5] and [6] and propose a novel construction strategy for such a class of codes. For simplicity, we will consider codes with two protection classes but the derivation for $m_e > 2$ is straightforward.

A. Weighted approach

The first strategy (to which we will refer as the weighted approach) for constructing UEP LT codes was proposed in [5]. In that work, the authors proposed a partition of the k variable nodes into m_e sets of sizes $\alpha_1 k, \alpha_2 k, \ldots, \alpha_{m_e} k$ such that $\sum_{j=1}^{m_e} \alpha_j = 1$ and the probability of an edge being connected to a particular variable node within the set j being q_j. By introducing a bias on the probabilities[3] q_j, some sets of symbols become more likely to be selected during the encoding which makes them more protected.

Considering the two-class case, the selection probabilities are defined as $\omega_1 = \alpha k_M$ and $\omega_2 = (1-\alpha)k_L$, where α is the fraction of input symbols that belong to the first class and, $k_M > 1$ and $0 < k_L < 1$ are assigned to the set of *more important bits* (MIB) and *less important bits* (LIB), respectively. Replacing these values in Eq. (11), we obtain the coefficients of the multi-edge check node degree distribution of the weighted scheme.

$$R_{\mathbf{d}} = \Omega_{d_1+d_2} \cdot \binom{d_1+d_2}{d_1} (\alpha k_M)^{d_1} \left((1-\alpha)k_L\right)^{d_2} . \quad (13)$$

By its turn, the variable node degree distribution is obtained by making $p_j = \frac{1}{k_j}$ in Eq. (11), where k_j is the number of variable nodes belonging to the protection class C_j.

For the two-class case, encoding is implemented by first defining the degree value i from a degree distribution $\Omega(x)$ and making $d_1 = \min([\alpha i k_m], \alpha k)$ and $d_2 = i - d_1$. Then d_1 and d_2 symbols are chosen among the MIB and the LIB, respectively. The encoding block is generated performing a bitwise XOR operation over the $i = d_1 + d_2$ selected symbols.

B. Windowed approach

The second UEP LT code construction strategy (from now on referred to as the windowed approach) was introduced in [6]. Similar to [5], the windowed approach partitions the input symbols into protection classes of $k_1, k_2, \ldots, k_{m_e}$ symbols such that $k_1 + k_2 + \cdots + k_{m_e} = k$. Then, the i-th window is defined as the set of the first $w_i = \sum_{j=1}^{i} k_j$ input symbols and thus the most important symbols form the first window while the whole block comprises the final m_eth window.

Each output symbol is encoded first selecting a window i, with each window having associated to it a probability Γ_i of being chosen. Then, during the encoding, each output symbol is formed according to the regular LT encoding algorithm considering only the symbols inside the selected window. This posed, the coefficients of the check node degree distribution of the windowed scheme for two-class case is given by

$$R_{\mathbf{d}} = \Omega_{d_1+d_2} \cdot [(1-\Gamma_1)\binom{d_1+d_2}{d_1}\alpha^{d_1}(1-\alpha)^{d_2} + \Gamma_1(1-\text{sgn}(d_2)]. \quad (14)$$

Note that the term $\Gamma_1(1-\text{sgn}(d_2))$ indicates the generation of output symbols with $\mathbf{d} = (d_1, 0)$ when the window w_1 is selected (what occurs with probability Γ_1). As for the weighted case, the variable node degree distribution can be obtained from Eq. (11) with $p_j = 1/k_j$.

C. Flexible UEP LT codes construction algorithm

We showed in the previous subsections that both the weighted and the windowed schemes rely on the modification of the probability of occurrence of an output symbol of type $\mathbf{d}$, i.e., they modify the coefficients R_d of a non-UEP LT code (Eq. (7)) in order to favor the selection of some class of input symbols during encoding.

The scheme we propose here works by biasing the coefficients $R_{\mathbf{d}}$ in order to increase the average number of edges of the most protected classes. In fact, we transfer edges from one less important class to another more important one.

Consider an LT code with two protection classes. Its check node degree distribution can be written as

$$R(\mathbf{x}) = R_{(1,0)}x_1 + R_{(0,1)}x_2 + R_{(2,0)}x_1^2 + R_{(1,1)}x_1x_2 + R_{(0,2)}x_2^2 + R_{(3,0)}x_1^3 + \cdots + R_{(0,i_{max})}x_2^{i_{max}}, \quad (15)$$

where $R_{(d_1,d_2)} = R_{\mathbf{d}}$ for $\mathbf{d} = (d_1, d_2)$ and $i_{max} = \max(i|\Omega_i > 0)$. In order to keep the original overall output symbol degree distribution $\Omega(x)$, the coefficients $R_{\mathbf{d}}$ must satisfy the following condition

$$\sum_{\mathbf{d}} R_{\mathbf{d}} = \Omega_i, \text{ for all } \mathbf{d} : \sum_{j=1}^{m_e} d_j = i \,. \quad (16)$$

In the two-class case for example, $R_{(1,0)} + R_{(0,1)} = \Omega_1$, $R_{(2,0)} + R_{(1,1)} + R_{(0,2)} = \Omega_2$, and so on. The idea of our proposed scheme is to increase the probability of selection of the most important input symbols by increasing the occurrence probability of output symbols which are more connected to input symbols of the most sensitive class. For example, in the two-class case we increase the values of the coefficients $R_{(d_1,d_2)}$ with $d_1 > d_2$ while observing the condition given in Eq. (16).

The encoding of the flexible UEP LT codes is similar to the traditional LT codes except that instead of selecting the output symbol degree i according to $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$, one must select an edge degree vector $\mathbf{d}$ according to $R(\mathbf{x}) = \sum_{\mathbf{d}} R_{\mathbf{d}} \mathbf{x}^{\mathbf{d}}$, where $R_{\mathbf{d}}$ denotes the probability of the edge degree vector $\mathbf{d}$ being chosen. After that, an output symbol with edge degree vector $\mathbf{d} = (d_1, d_2)$ is formed by selecting d_1 input symbols from C_1, d_2 input symbols from C_2 uniformly and at random, and performing a bitwise XOR operation between them. Note

[3]For the regular equal error protecting LT codes $q_1 = \ldots = q_{m_e} = \frac{1}{k}$.

that in contrast to the weighted approach, there is no need to determine the degrees d_i individually prior to encoding, but only the edge degree vector $\mathbf{d}$. This makes the extension of the encoding algorithm to the case $m_e > 2$ (not trivial for the weighted approach) straightforward.

The idea of biasing the coefficients $R_{\mathbf{d}}$ in order to increase the average number of edges of the most protected classes will become clearer with the following toy example.

1) UEP LT construction example: Consider an LT code with degree distribution $\Omega(x) = 0.15x + 0.55x^2 + 0.30x^3$. We like to construct a two-class unequal error protecting LT code where 10 % of the input symbols belong to the most protected class, i.e., $\alpha = 0.1$. We can compute the coefficients $R_{\mathbf{d}}$ for the non-UEP case by means of Eq. (11) with $\omega_1 = \alpha$ and $\omega_2 = 1 - \alpha$. In this example, we will refer to the equal error protecting LT codes check node coefficients as $R_{\mathbf{d}}$ and the ones of UEP LT codes as $R_{\mathbf{d}}^{UEP}$

In order to favor the selection of the most important input symbols during encoding, we increase the values of the coefficients $R_{(d_1,d_2)}$ with $d_1 > d_2$ by raising the value of such a coefficient while observing the condition (16). A simple way of realizing this is to transfer a fraction f of a coefficient $R_{(a,b)}$ where $a < b$ to the coefficient $R_{(b,a)}$, e.g., if $f = 0.1$ we make $R_{(2,1)}^{UEP} = R_{(2,1)} + 0.1R_{(1,2)}$ and $R_{(1,2)}^{UEP} = R_{(1,2)} - 0.1R_{(1,2)}$. In order to further refine the performance of the UEP LT codes, we can define different values of f, e.g., $f = f_1$ for symbols with $d_2 = 0$ and $f = f_2$ otherwise. The impact of this strategy on the asymptotic performance of UEP LT codes is presented in the next section.

IV. Asymptotic analysis of multi-edge UEP LT codes

The asymptotic analysis of multi-edge LT codes can be done by means of density evolution. Note though that in a multi-edge type analysis, we require the computation of one density for each edge type. Recall that the decoding of LT codes is analogous to the belief-propagation decoding of a code being transmitted through an erasure channel. This analogy allows us to use the results derived for the BEC channel for computing the probability of an LT code input symbol not being recovered.

Theorem 1: The erasure probability y_l^j of an input symbol of class j of a multi-edge LT code with node perspective degree distribution pair $(L(\mathbf{x}), R(\mathbf{x}))$ at iteration $l \geq 0$ is given by

$$y_l^j = L_{x_j}(1 - \rho_j(1 - y_{l-1}^1, \ldots, 1 - y_{l-1}^{m_e})) \,, \qquad (17)$$

where $y_{-1} = 1$, $\rho_j(\mathbf{x}) = \frac{R_{x_j}(\mathbf{x})}{R_{x_j}(\mathbf{1})} = \sum \rho_{\mathbf{d}}^{(j)}(\mathbf{x})$, and $\rho_{\mathbf{d}}^{(j)}$ denotes the fraction of type j ($j = 1, \ldots, m_e$) edges connected to check nodes of type $\mathbf{d}$.

Proof. Let G denote the bipartite graph corresponding to an LT encoding. Consider a subgraph G_l^j of G formed by a variable node v^j, chosen uniformly and at random among the ones of class j and all its neighbors within distance $2l$. Asymptotically, the subgraph G_l^j is a tree [9] of depth $2l$ that represents the dependency between the value assumed by v^j and the messages sent by the other variable nodes after l message passing decoding iterations. Let the variable node v^j be the root (depth 0) of G_l^j and assume that y_l^j is its erasure probability. Consider now, that the nodes at depth 2 in G_l^j are the roots of independent G_{l-1}^j trees. Consider the variable-to-check messages in the lth iteration. By assumption, each such message is an erasure with probability y_{l-1}^j and all messages are independent. Recall that in a multi-edge framework, each check node of type $\mathbf{d} = (d_1, d_2, ..., d_{m_e})$ is connected to d_1 edges of type 1, d_2 edges of type 2, and so on. Since we are considering a BP decoding, by definition, a check-to-variable message emitted by a check node of degree i along a particular edge is an erasure iff any of the $i - 1$ incoming messages is an erasure. Thus, the erasure probability of an outgoing message of a check node at depth l with edge degree vector $\mathbf{d}$ sent through an edge of type j is equal to $1 - (1 - y_{l-1}^1)^{d_1} \cdots (1 - y_{l-1}^j)^{d_j - 1} \cdots (1 - y_{l-1}^{m_e})^{m_e}$. Since the outgoing edge has probability $\rho_{\mathbf{d}}^j$ to be connected to a check node of type $\mathbf{d}$, it follows that the expected erasure probability of a check-to-variable message in the lth iteration is equal to $1 - \rho_j(1 - y_{l-1}^1, \ldots, 1 - y_{l-1}^{m_e})$ where $\rho_j(\mathbf{x}) = \sum \rho_{\mathbf{d}}^{(j)}(\mathbf{x})$. Now consider the erasure probability of the root node at iteration l. By definition, a variable node will be considered erased if all its incoming messages are erasures. Since a variable node of class j has only connections to edges of type j, the probability of a variable node of degree d_j being erased is $(1 - \rho_j(1 - y_{l-1}^1, \ldots, 1 - y_{l-1}^{m_e}))^{d_j}$. Averaging over the variable node degree distribution L_{x_j} we obtain that the erasure probability of an input symbol of class j at iteration l is given by $L_{x_j}(1 - \rho_j(1 - y_{l-1}^1, \ldots, 1 - y_{l-1}^{m_e}))$ as claimed.

□

Equation (17) together with the degree distributions $(L(\mathbf{x}), R(\mathbf{x}))$ allows us to compute the asymptotic ($k \to \infty$) performance of a multi-edge UEP LT code with overall output symbol degree distribution $\Omega(x)$. Herein, we consider multi-edge UEP LT codes with the overall output symbol degree distribution proposed in [2]

$$\begin{aligned}\Omega(x) = {} & 0.007969x + 0.493570x^2 + 0.166220x^3 \\ & + 0.0726464x^4 + 0.082558x^5 + 0.056058x^8 \\ & + 0.037229x^9 + 0.055590x^{19} + 0.025023x^{64} \\ & + 0.003135x^{66}. \end{aligned} \qquad (18)$$

Figure 2 shows the asymptotic performance of the three different unequal error protecting LT codes construction strategies presented in this paper. The parameters for the weighted ($k_m = 2.077$) and the windowed ($\Gamma_1 = 0.084$) approaches are optimized for an overhead $\gamma = 1.05$ according to [6]. The flexible UEP LT performance was obtained for $f_1 = 0.09$ and $f_2 = 0.13$.

V. Simulation results

In this section we present the simulation results for both weighted and our proposed scheme for generating UEP LT codes. We only compare the schemes that bias the check node

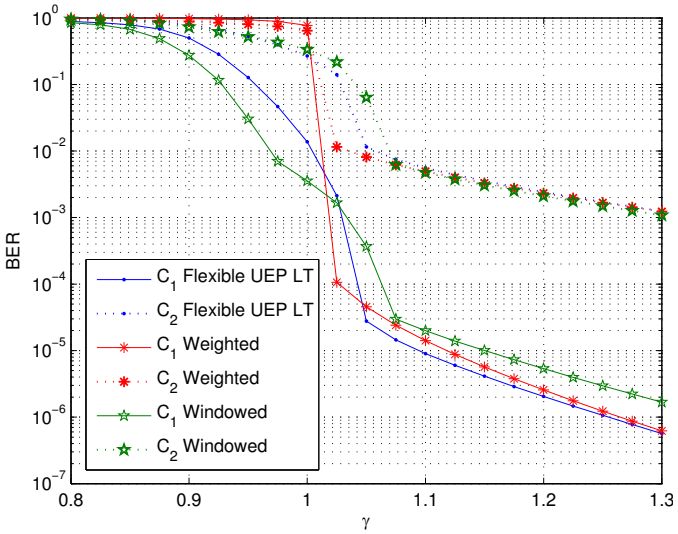

Fig. 2. Asymptotic performance of the weighted, windowed and the proposed flexible UEP LT construction strategies. The parameters of the three approaches were optimized for an overhead $\gamma = 1.05$.

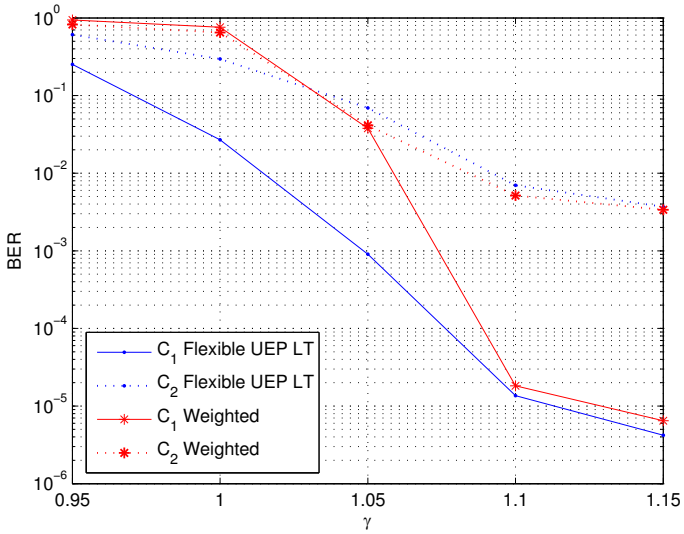

Fig. 3. Simulation results of the weighted and flexible schemes for $k = 5000$.

degree distribution. Since the windowed scheme performs a pre-coding[4] on the regular LT-encoding and not a bias on the selection probability of the bits within the most protected class, its simulation results are not depicted.

Figure 3 shows the bit-error rates after performing LT decoding. The parameters for both the weighted and the flexible UEP LT approaches are the same applied to the asymptotic analysis described in the previous section and depicted in Fig. 2. We assume the transmission of $k = 5000$ input symbols divided into two different levels of protection. The first protection class is composed of 10 % of the input symbols ($k_1 = 0.1k$), and the second protection class contains the other $k_2 = k - k_1$ input symbols.

Note that for the finite-length simulation, the flexible UEP LT strategy reaches a lower BER for the most protected class than the weighted approach. This means that for reaching a BER = 10^{-3} for the most important input symbols, we need to collect $1.05k = 5250$ output symbols using the proposed flexible UEP LT approach while for reaching the same BER for the weighted approach we would need to collect approximately $1.075k = 5375$ output symbols. This can also be interpreted as a smaller recovery time, since the flexible approach reaches a target BER slightly faster than the weighted one.

VI. Concluding Remarks

We introduced a multi-edge type analysis of unequal error protecting LT codes. Within the multi-edge framework, we showed that both the weighted and the windowed schemes rely on a modification of the probability of occurrence of an output symbol with edge degree vector $\mathbf{d}$. Furthermore, we derived the density evolution equations for UEP LT codes, analyzed two of the existing techniques for generating UEP LT codes, and proposed one third which we called flexible UEP LT approach. Finally, we showed by means of simulation that our proposed scheme performed better than the weighted scheme for both low and high overheads.

[4]For example, in the two-class case the windowed scheme is equivalent to first generating $\Gamma_1 \gamma k$ symbols of class one (a pre-coding) and then proceeding to the regular LT-encoding.

Acknowledgment

This work is funded by the German Research Foundation (DFG).

References

[1] M. Luby, "LT codes," in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, Nov. 2002, pp. 271–282.
[2] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.
[3] P. Maymounkov, "Online codes," NYU, Tech. Rep. TR2003-883, Nov. 2002.
[4] F. R. Kschischang, B. J.Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.
[5] N. Rahnavard, B. N. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *IEEE Transactions on Information Theory*, vol. 53, no. 4, pp. 1521–1532, April 2007.
[6] D. Sejdinović, D. Vukobratović, A. Doufexi, V. Šenk, and R. Piechocki, "Expanding window fountain codes for unequal error protection," *IEEE Transactions on Communications*, vol. 57, no. 9, pp. 2510–2516, Sep. 2009.
[7] T. Richardson and R. Urbanke, "Multi-Edge Type LDPC Codes," Tech. Rep., 2004, submitted to IEEE Transaction on Information Theory.
[8] ——, *Modern Coding Theory*. Cambridge University Press, 2008.
[9] M. Luby, M. Mitzenmacher, and A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proceedings of the 9th SIAM Symposium on Discrete Algorithms*, Jan. 1998, pp. 364–373.