Transactions Letters

On Raptor Code Design for Inactivation Decoding

Kaveh Mahdaviani, *Student Member, IEEE,* Masoud Ardakani, *Senior Member, IEEE,* and Chintha Tellambura, *Fellow, IEEE*

***Abstract*—Based on a new vision of the inactivation decoding process, we set a new degree distribution design criterion for the LT part of Raptor codes. Under an infinite block length assumption, a family of degree distributions that satisfy the new design criterion is analytically derived. The finite length performance of this family is investigated by using computer simulations and is shown to outperform the conventional design.**

***Index Terms*—Finite length Raptor codes, inactivation decoding, Soliton distribution.**

I. Introduction

LUBY transform (LT) codes were originally introduced as the first practical fountain codes in [1]. As such, LT codes are designed to transmit a theoretically endless stream of symbols until the receiver has enough symbols to decode all the information bits. Raptor codes [2], an extension of LT codes, employ an outer code to enable the receiver to recover the whole information stream from any sufficiently large subset of recovered intermediate symbols. This idea significantly improves the performance of LT codes, as the recovery of the last few percentages of the information bits, which could be very slow, is now done by using the outer code.

Raptor codes are able to asymptotically achieve the channel capacity on any binary erasure channel (BEC) without any channel state information at the transmitter or the receiver. This universal capacity-achieving property enables optimal performance even in time-varying channels. Accordingly, these codes are the natural choice for broadcasting/multicasting to a group of receivers with different and even unknown channel qualities. As a result Raptor codes have been adopted by the 3rd Generation Group Partnership Project (3GPP) to be used in multimedia broadcast/multicast services (MBMS) for forward error correction [3] and digital video broadcast-handheld (DVB-H) [4]. The desirable properties of Raptor codes have motivated many researchers to study their performance and design for other channels [5], [6]. Decoder design for Raptor codes has also been an active research area [7]–[9].

In this work, we study the design of Raptor codes for the BEC when inactivation decoding (ID) [9] is used. An ID decoder is essentially a maximum likelihood decoder with controlled complexity, which can accomplish the decoding with a smaller number of received symbols than any other decoder requires. Hence, ID is incorporated in 3GPP as a practical decoder [3]. Despite the rich literature on code design for the conventional edge deletion decoding (e.g., [2], [10], [11]), code design for ID has not yet received much attention.

In the remainder of this article, we first briefly review the encoding and decoding of Raptor codes, focusing on ID. In Section III, we introduce our code design, by proposing a new design criterion, and then we use this criterion for an analytical design. The numerical comparisons between the code used by the 3GPP and our proposed code are presented in Section IV.

II. Encoding and Decoding of Raptor Codes

Encoding of the Raptor codes is done in two separate steps. In the first step, k information bits are coded to $n = k/R$ intermediate bits by using an outer code of rate R. In the second step, the LT encoder first uses a probability distribution to choose an integer $m \in \{1, \cdots, D\}$, $D \leq n$, and then uniformly at random chooses m intermediate bits whose XOR forms an output symbol for transmission. The probability distribution of m is characterized by a generating polynomial $\Omega(x) = \sum_{i=1}^{D} \Omega_i x^i$. Here, $m = i$ occurs with probability Ω_i.

Decoding is similarly performed in two separate steps. First the LT code is decoded, and then the outer code is decoded in the second step. Assuming that the outer code can recover the whole information block from any subset of $n(R+\sigma)$, $\sigma > 0$ recovered intermediate bits, we focus our discussion on the LT decoder.

For LT decoding, a decoding graph [2] is formed based on the set of received symbols. The decoding graph is a bipartite graph with one vertex set corresponding to the set of all intermediate bits, and the other set corresponding to the output bits (output nodes). Initially, each output node is adjacent to the group of intermediate nodes forming the corresponding received bits.

Various decoding solutions can be used. Gaussian elimination, although optimal, is typically too complex. A modified version of the belief propagation algorithm, called edge deletion decoding (EDD) [1], is an efficient alternative when an appropriate design of $\Omega(x)$ is performed. EDD requires a small overhead in the number of received symbols for successful decoding [12]. This algorithm uses degree-one output nodes in the decoding graph to deduce the value of their neighbouring intermediate nodes, and then removes

Paper approved by T. M. Duman, the Editor for Coding Theory and Applications of the IEEE Communications Society. Manuscript received July 5, 2011; revised January 17 and April 18, 2012.

The authors are with the Dept. of Elec. and Comp. Eng., University of Alberta (e-mail: {kmahdavi, ardakani, chintha}@ece.ualberta.ca).

Digital Object Identifier 10.1109/TCOMM.2012.072612.110143

the recovered intermediate nodes to achieve new degree-one output nodes iteratively.

Inactivation Decoding

For moderate block lengths (1024 to 8192 bits), which are of interest in applications supported by the 3GPP standard, a modified version of EDD, called inactivation decoding (ID), was introduced in [9]. The main difference between ID and EDD occurs when the set of degree one output nodes, called the ripple, becomes empty. In this case, the EDD stops until the ripple is refilled by receiving more symbols from the channel. ID, however, instead of waiting for more symbols, selects some unrecovered intermediate nodes in the remaining decoding graph and temporarily excludes them from the graph. This process is called inactivation. By inactivating some of the intermediate nodes, their edges will also be excluded temporarily, reducing the degree of some of the remaining output nodes. Thus, the decoder extracts some *reduced* degree-one output nodes, whose values can be found in terms of the inactivated bits. The decoder can now recover more intermediate bits (albeit, in terms of the inactivated bits) until the ripple is empty again, and another inactivation can be performed. Finally, the decoder uses Gaussian elimination for the inactivated bits and finishes the decoding by using a back filling process, which evaluates all the intermediate bits which have been recovered in terms of the inactivated bits.

In order to solve the subsystem of linear equations formed by the inactivated bits, this subsystem should be full rank. The subsystem has a high probability of being full rank because it is dense. However, if it is not full rank, the receiver receives more symbols to obtain more equations and remove the rank deficiency. This process results in a very small average reception overhead, which has been shown to be less than one percent in practice [8], [13].

For selecting a node to be inactivated, many different strategies can be used [9]. One trivial choice is to randomly select an unrecovered intermediate node connected to a reduced degree-two output node. We will refer to this strategy as "Random ID". Another strategy, introduced in [9], is to inactivate one of the nodes in the maximum connected component of G, where G is the *degree-two induced subgraph* [14] of the remaining decoding graph (see Fig. 1). Inactivating any of the bits in a connected component of G causes the immediate recovery of all the other bits in that component. Hence, the second strategy, which we refer to as "Max-Component ID," performs better than Random ID. Since G is subject to change during the decoding process, the search for the largest connected component must be repeated during each inactivation step. Thus, Max-Component ID is considerably more complex than Random ID.

Degree distribution design for ID is considered in [14], where Max-Component ID is assumed, and the design criterion is to statistically guarantee the existence of a giant connected component in G at each inactivation step. Having a giant connected component guarantees the recovery of a large portion of nodes at each inactivation step.

Accordingly, [14] introduced a procedure that takes an $\Omega(x)$ and determines on average for how many inactivations a giant component will almost surely be present in G. The design problem is then to find a generating polynomial $\Omega(x)$, which will guarantee the existence of a giant connected component until the desired portion of the intermediate bits is recovered. This design criterion, in addition to a mixture of optimization methods, has been used to design a degree distribution which the 3GPP has adopted [3].

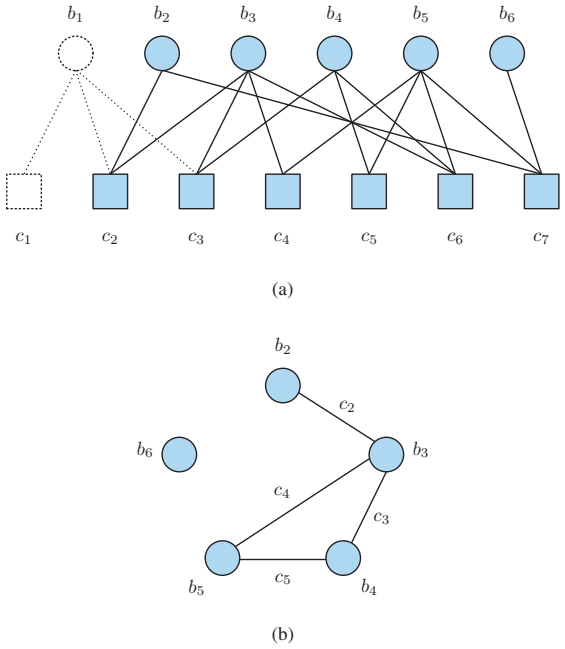

Fig. 1. (a) The decoding graph with output nodes c_1 to c_7 and intermediate bits b_1 to b_6. The ripple is initiated with c_1, which recovers b_1 and then becomes empty. (b) The reduced degree-two induced subgraph G based on the remaining effective decoding graph. In G, every reduced degree-two output node will represent an edge. Here, G contains two connected components. The maximum component contains four nodes. Max-Component ID may, for example, choose b_4 as a node in the maximum component of G for inactivation. This choice will refill the ripple with c_3 and c_5, which, in turn, recover b_3 and b_5 in terms of b_4.

III. Degree Distribution Design

In this section, a new design criterion is proposed from which a more efficient $\Omega(x)$ for ID is designed. The basic difference between our design and that of [14] is that [14] aims to increase the portion of bits that are *guaranteed* to be recovered after each inactivation, whereas our design aims to increase the *average* portion of recovered bits after each inactivation. Notice that the actual number of recovered bits is usually more than the *guaranteed* portion. Thus, it appears reasonable to aim at increasing the average recovery.

From the discussion in Section II, it is obvious that for a fixed decoder structure and with a constant performance for the outer code, all the properties of a Raptor code are characterized by the generating polynomial $\Omega(x)$. Similar to the case of design for the EDD, an infinite block length assumption is made for the analytical design of $\Omega(x)$. However, the performance of the finite length case is evaluated through simulations.

For a Raptor code under EDD, the main performance measure is the overhead. Under ID, however, the overhead may not be as meaningful because ID performs an inactivation

instead of receiving extra symbols. As a result, a good measure of performance appears to be the number of required inactivations [8], [13], which directly affects the decoding complexity. Therefore, our design goal is to reduce the number of required inactivations.

A. Evolution of $\Omega(x)$ During ID

In ID, after each inactivation, the remaining degree distribution changes. As a result, to study the average performance analytically, we need the remaining degree distribution, based on the original $\Omega(x)$ and the portion of the recovered intermediate bits δ. Denoting the new degree distribution as $\Omega_\delta(x)$, we have $\Omega_0(x) = \Omega(x)$, and $\Omega_1(x) = 1$. Also, since the selection of the intermediate bits in the encoding is uniformly random, recovering a δ portion of intermediate bits is equivalent to deleting a randomly chosen δ portion of intermediate nodes in the decoding graph. Hence, we can assume that a random δ portion of the edges of the decoding graph is also deleted. Therefore, a randomly chosen output node of initial degree j will be of degree $i = 1, \cdots, j$ with probability $\binom{j}{i}(1-\delta)^i\delta^{j-i}$. Then, the average degree distribution of the output symbols in the remaining graph will be

$$\Omega_\delta(x) = \sum_{i=1}^{D} \Omega_{\delta,i} x^i = \sum_{i=1}^{D} \left(\sum_{j=0}^{D-i} \Omega_{i+j} \binom{i+j}{j} (1-\delta)^i \delta^j \right) x^i = \Omega((1-\delta)x + \delta).$$

As a result,

$$\Omega_{\delta,i} = \sum_{j=0}^{D-i} \Omega_{i+j} \binom{i+j}{j} (1-\delta)^i \delta^j. \tag{1}$$

B. A New Design Criterion

The new design is based on a new insight into the ID process. As mentioned in Section II, ID starts with an EDD phase and works until the ripple is empty. At this point, inactivation is performed, and another phase of EDD is started. Thus, one can think of ID as a series of EDDs, each applied to a portion of the unrecovered bits. According to this view, we need a degree distribution that will perform well under a series of EDDs despite the recovery of any portion of bits. Designing such a distribution is a challenging task because the degree distribution for each EDD step may be different. Thus, the performance may differ in each step. A degree distribution which remains close to optimal in all EDD steps is, therefore, desired. Accordingly, we first investigate another effect of recovering a δ portion of the intermediate bits on the degree distribution of the output nodes.

An intermediate bit b_i is recovered when the degree of an output node $c_{i'}$, connected to b_i, is reduced to one. In fact, the last edge of $c_{i'}$ connects it to b_i. After recovering b_i, the output node $c_{i'}$ can no longer be effective in the decoding process. Now, assume that the receiver has originally received $(1+\varepsilon)k$ symbols for a very small $\varepsilon > 0$ (in practice $\varepsilon < 0.01$). Therefore, after recovering a δ portion of intermediate bits, a δ portion of the output nodes will not be effective for the rest of the decoding process (see Fig. 2). In other words, decoding continues by performing EDD on the remaining decoding subgraph containing $(1-\delta)$ portion of output nodes and the unrecovered intermediate nodes.

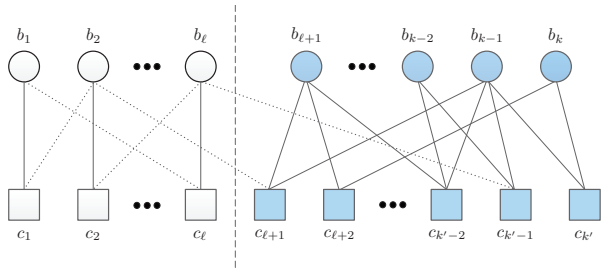

Fig. 2. The decoding graph at some intermediate step of decoding. The receiver started the decoding after receiving $k' = (1+\varepsilon)k$ output symbols, where ε is a very small positive number. Up to this point, a $\delta = \frac{\ell}{k}$ of the intermediate bits have been recovered. The left part shows the non-effective part of the decoding graph at this moment, which could not participate in the decoding of the remaining bits. The right part is still effective and contains $(1-\delta)k$ intermediate bits and approximately $(1-\delta)k'$ output nodes as well.

Now, recall that $\Omega_{\delta,j}$ represents the fraction of reduced-degree j, $j \geq 2$ output nodes after recovering a δ portion of intermediate nodes. Accordingly, starting with a close-to-optimal $\Omega(x)$, if for any $0 < \delta < 1$, $\Omega(x)$ satisfies

$$\forall j \geq 2, \quad \Omega_{\delta,j} = (1-\delta)\Omega_j \tag{2}$$

then the close-to-optimal performance for the next EDD step is preserved. This way, the code recovers a large portion of bits in each EDD step. Thus, we use (2) as a design criterion.

C. The Proposed Code Design

According to (1),

$$\Omega_{\delta,i} = \frac{(1-\delta)^i}{i!} \sum_{j=0}^{D-i} \frac{\Omega_{i+j}(i+j)!\delta^j}{j!}.$$

Now, let us define

$$f_{\delta,i} \triangleq \sum_{j=0}^{D-i} \frac{\Omega_{i+j}(i+j)!(\delta)^j}{j!}. \tag{3}$$

Then we obtain $\Omega_{\delta,i} = \frac{(1-\delta)^i}{i!} f_{\delta,i}$. In addition, according to (2), for all $i \geq 2$ it is desired to have $\Omega_{\delta,i} = (1-\delta)\Omega_i$. Thus, we can formulate part of the design criterion as $f_{\delta,i} = i!(1-\delta)^{-(i-1)}\Omega_i$, or equivalently,

$$\forall i \geq 2, \quad \sum_{j=0}^{D-i} \frac{\Omega_{i+j}(i+j)!\delta^j}{j!} = i!(1-\delta)^{-(i-1)}\Omega_i = i(i-1)\Omega_i \sum_{j=0}^{\infty} \frac{(i+j-2)!\delta^j}{j!}. \tag{4}$$

In (4), the summation on the right-hand side is derived by evaluating the Taylor series expansion of the function $(i-2)!x^{-(i-1)}$ centred at $x_0 = 1$ for $x = 1-\delta$. Assuming the maximum degree D could be infinite, equation (4) suggests the following solution:

$$\Omega(x) = \sum_{i=2}^{\infty} \frac{1}{i(i-1)} x^i. \tag{5}$$

Surprisingly, this solution is the well known ideal Soliton distribution [1]. This, however, is an infinite degree distribution which cannot be used in a practical setup. With a finite allowed maximum degree D, (5) must be modified. In the next subsection, we provide a finite approximation of (5) that remains close to optimal throughout the decoding process by satisfying (2) with a good approximation.

D. Finite Maximum Degree Design

Recall that for the outer code to finish the decoding successfully, the LT code requires a recovery rate greater than $R+\sigma$. Also, for a finite maximum degree design, to satisfy (2) and motivated by (5), we seek an $\Omega(x)$ approximately in the form of

$$\Omega(x) = \sum_{i=2}^{D} \frac{c}{i(i-1)} x^i.$$

As was first mentioned in [15], the hypergraph collapse process studied in [16] is identical to the EDD process. Now, let r be a positive real number less than or equal to the smallest positive root of $(1+\varepsilon)\Omega'(x) + \ln(1-x) = 0$. Then, as $k \to \infty$, under EDD, rk intermediate bits are recoverable with a high probability from any set of $(1+\varepsilon)k$ received bits [16]. Similar results were also obtained in [2], based on the And-Or tree analysis [17]. This result was used in [11] to study the performance of EDD for recovery of a less-than-one portion of the message bits as needed in Raptor codes. The average recoverable portion of bits for a given degree distribution, therefore, is equal to the smallest positive root of $(1+\varepsilon)\Omega'(x) + \ln(1-x) = 0$.

Thus, to achieve a recovery rate of $R+\sigma$, we need $\Omega'(x) > \frac{-\ln(1-x)}{(1+\varepsilon)}$ for all $x \in (0, R+\sigma)$. By using the Taylor series expansion $-\ln(1-x) = \sum_{i=1}^{\infty} \frac{x^i}{i}$, a necessary condition for all $x \in (0, R+\sigma)$ can be derived as

$$\begin{aligned}\Omega(x) = \int_0^x \Omega'(t)dt &\geq \int_0^x \frac{-\ln(1-t)}{(1+\varepsilon)}dt \\ &= \frac{x(1-\ln(1-x)) + \ln(1-x)}{(1+\varepsilon)} \\ &= \sum_{i=2}^{\infty} \frac{x^i}{(1+\varepsilon)i(i-1)}.\end{aligned}$$

Among all the terms of the form $\omega_i(x) \triangleq \frac{x^i}{i(i-1)}$, the term $\omega_j(x)$ has the maximum derivative in the interval $I_j \triangleq (\frac{j-1}{j}, \frac{j}{j+1})$. Also, $\forall i > j \geq \ell \geq 2$, $\frac{d}{dx}\omega_j(x) > \frac{d}{dx}\omega_i(x)$ for any $x \in I_\ell$. Therefore, to have

$$\forall x \in (0, R+\sigma) \quad \Omega(x) > \frac{1}{(1+\varepsilon)} \sum_{i=2}^{\infty} \omega_i(x) \tag{6}$$

for a given ε, it is enough to set

$$\Omega(x) = \sum_{i=2}^{m} \frac{c}{i(i-1)} x^i + \left(1 - \frac{c(m-1)}{m}\right) x^{m+1}, \tag{7}$$

where m is an integer such that $m \geq m^*$ and $\frac{m^*-1}{m^*} \leq (R+\sigma) \leq \frac{m^*}{m^*+1}$, and $c \geq \frac{1}{1+\varepsilon}$. Clearly, choosing a larger m results in a better approximation to (5).

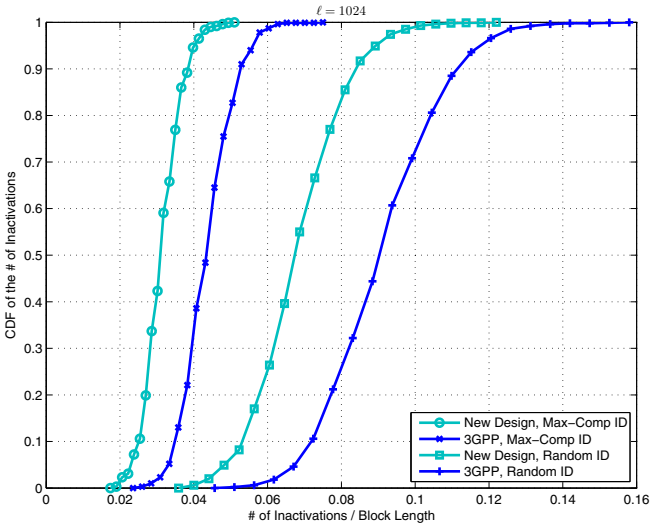

Fig. 3. CDF of the normalized number of inactivations required for successful decoding, $\ell = 1024$.

Using (6) and (7), we obtain

$$\forall x \in (0, (R+\sigma)),$$
$$c \geq \frac{x(1-\ln(1-x)) + \ln(1-x) - x^{m+1}(1+\varepsilon)}{(1+\varepsilon)(\sum_{i=2}^{m} \frac{x^i}{i(i-1)} - \frac{(m-1)}{m}x^{m+1})}. \tag{8}$$

The right-hand side of (8) is a strictly increasing function of x. Thus, we can finish the design by choosing c equal to the value of the right-hand side evaluated at $x = R+\sigma$.

In order for the decoding to start and recover a portion of intermediate bits before the first inactivation, we provide a very small positive Ω_1, as do the existing approach.

Setting $m = m^*$ provides the lowest computational complexity since doing so obtains the smallest average degree of the distribution. However, setting $m = m^*$ also reduces the probability of covering a randomly selected intermediate bit in an output symbol and therefore slightly increases the reception overhead. This slight increase is a side-effect of a decreasing outer code rate in the 3GPP for a smaller block length. At $\ell = 1024$, the outer code rate is reduced to $R = 0.9381$, and setting $m = m^* = 16$ makes the average degree of our $\Omega(x)$ slightly smaller than that of the 3GPP. Moreover, the probability of leaving an intermediate bit uncovered (not involved in any of the equations corresponding to the received bits) is approximately $e^{-\Omega'(1)(1+\varepsilon)}$ [18], where $\Omega'(1)$ represents the average degree. Therefore, for successful decoding, a slightly higher overhead will be needed. As reported in Table I, this increase is around 0.33% for $\ell = 1024$ and only 0.04% for $\ell = 8192$.

The choice of the outer code rate R in the 3GPP is based on the performance of the adopted $\Omega(x)$. Appropriate outer code selection for our proposed $\Omega(x)$ can be considered. Among other solutions for this slight increase in the overhead, one can either allow m to be larger than m^* or add a term of higher order to prevent the loss of coverage. In Section IV, we will compare our design numerically with that of the 3GPP.

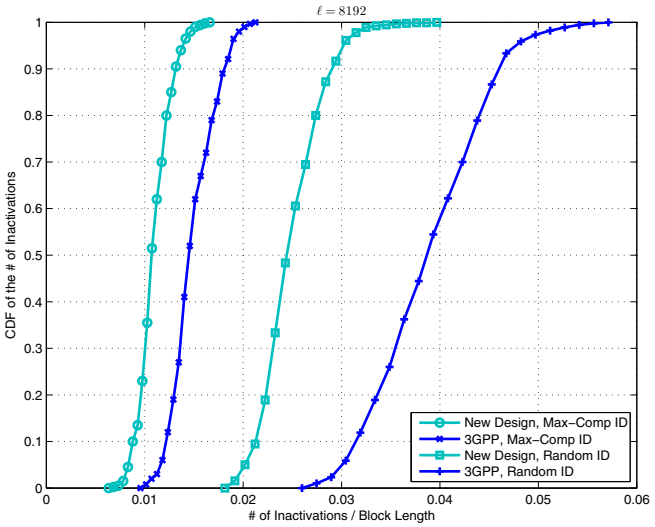

Fig. 4. CDF of the normalized number of inactivations required for successful decoding, $\ell = 8192$.

IV. Numerical Results

In order to verify the performance of our proposed codes, we compare the performance of the degree distribution adopted in 3GPP with the proposed degree distribution introduced in (7), where m is chosen to be equal to m^*. In each case, similar to [8], [13], [14] we assume that the receiver receives enough overhead to form a full rank equation system in terms of the intermediate variables. As ID is a version of ML decoding, having a full rank equation system is sufficient for successful decoding. Hence, the decoding success rate is always one.

Figures 3 and 4 depict the average performance of both degree distributions for different block lengths ℓ, and the different strategies used for selecting a node for inactivation. As discussed in Section III, the basis of our comparison is the number of required inactivations. Thus, Figures 3 and 4 provide the cumulative distribution function (CDF) for a normalized number of inactivations (i.e., the number of inactivated nodes required for successful decoding divided by the block length). Figure 3 compares the performance of our proposed degree distribution with that of the 3GPP codes under two different selection strategies for a block length of $\ell = 1024$. Figure 4 repeats the same comparison for $\ell = 8192$. In both figures, the rate of the outer code is chosen according to 3GPP guidelines. This rate for $\ell = 8192$ is equal to $R = 0.9834$ and for $\ell = 1024$ is $R = 0.9381$. As Figures 3 and 4 reveal, for all cases, the performance of our proposed degree distribution is superior to the degree distribution of the 3GPP.

Table V shows the average performance measures for our simulations, which are again based on $m = m^*$. In this table, $\bar{d} = \Omega'(1)$ is the average degree of the distributions, $\bar{\varepsilon}$ represents the average reception overhead. Also, $\bar{I}_{Max-Comp}$, and $\bar{I}_{Rand}$ denote the average normalized number of inactivations when the selection of nodes for inactivation has been performed based on using the Max-Component ID and Random ID strategies, respectively. Table V indicates that our codes significantly reduced the number of inactivations at the cost of a slightly higher overhead.

TABLE I
AVERAGE PERFORMANCE MEASURES FOR THE 3GPP CODE AND THE PROPOSED DESIGN WITH $m = m^*$.

Code	ℓ	$\bar{d} = \Omega'(1)$	$\bar{\varepsilon}$	$\bar{I}_{Max-Comp}$	$\bar{I}_{Rand}$
3GPP	1024	4.6184	0.38%	4.49%	9.40%
	8192	4.6184	0.44%	1.54%	4.03%
New Design	1024	3.9739	0.66%	3.20%	7.04%
	8192	5.0854	0.48%	1.13%	2.59%

V. Conclusion

A new criterion for the design of degree distributions for inactivation decoding was presented. Based on this criterion, a family of degree distributions was found analytically. The suggested family was modified for the practical case of finite maximum degree. The simulation results confirmed the superiority of the proposed codes over existing designs.

References

[1] M. Luby, "LT codes," in *Proc. 2002 IEEE Symp. on Foundations Comput. Science*, pp. 271–280.
[2] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.
[3] "Technical specification group services and system aspects; multimedia broadcast/multicast services (MBMS); protocols and codecs (release 6)," 3GPP, Tech. Rep. 3GPP TS 26.346 V6.3.0, 2005.
[4] ETSI, "DVB BlueBook A101 digital video broadcasting (DVB); IP datacast over DVB-H: content delivery protocols," Tech. Rep. TS 102 472 V1.2.1, Dec. 2006.
[5] Z. Chen, J. Castura, and Y. Mao, "On the design of Raptor codes for binary-input Gaussian channels," *IEEE Trans. Commun.*, vol. 57, no. 11, pp. 3269–3277, Nov. 2009.
[6] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Trans. Inf. Theory*, vol. 52, no. 5, pp. 2033–2051, May 2006.
[7] A. AbdulHussein, A. Oka, and L. Lampe, "Decoding with early termination for Raptor codes," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 444–446, June 2008.
[8] S. Kim, S. Lee, and S.-Y. Chung, "An efficient algorithm for ML decoding of Raptor codes over the binary erasure channel," *IEEE Commun. Lett.*, vol. 12, no. 8, pp. 578–580, Aug. 2008.
[9] A. Shokrollahi, S. Lassen, and R. Karp, "Systems and processes for decoding chain reaction codes through inactivation," United States Patent Serial Number 6856263, Feb. 2005.
[10] P. Pakzad and A. Shokrollahi, "Design principles for Raptor codes," in *Proc. 2006 IEEE Inf. Theory Workshop*, pp. 165–169.
[11] S. Sanghavi, "Intermediate performance of rateless codes," in *Proc. 2007 IEEE Inf. Theory Workshop*, pp. 478–482.
[12] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 569–584, Feb. 2001.
[13] A. Shokrollahi and M. Luby, "Raptor codes," *ser. Foundations Trends Commun. Inf. Theory*, vol. 6, no. 3–4, pp. 213–322, 2009.
[14] A. Shokrollahi, "Fountain codes," *2nd Tutorial, 2009 IEEE International Symp. on Inf. Theory*.
[15] E. Maneva and A. Shokrollahi, "New model for rigorous analysis of LT-codes," in *Proc. 2006 IEEE International Symp. on Inf. Theory*, pp. 2677–2679.
[16] R. W. R. Darling and J. R. Norris, "Structure of large random hypergraphs," *Annals Applied Probability*, vol. 15, no. 1A, pp. 125–152, Feb. 2005.
[17] M. Luby, M. Mitzenmacher, and M. A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proc. 1998 ACM-SIAM Symp. Discrete Algorithms*, pp. 364-373.
[18] D. J. C. Mackay, "Fountain codes," *IEE Commun.*, vol. 152, pp. 1062–1068, 2005.