Systematic Luby Transform Codes as Incremental Redundancy Scheme

T. L. Grobler*†, E. R. Ackermann*†, J. C. Olivier*† and A. J. van Zyl‡
*Department of Electrical, Electronic and Computer Engineering
University of Pretoria, Pretoria 0002, South Africa
Email: trienkog@gmail.com, etienne.ackermann@ieee.org
†Defence, Peace, Safety and Security (DPSS)
Council for Scientific and Industrial Research (CSIR), Pretoria 0001, South Africa
‡Department of Mathematics and Applied Mathematics
University of Pretoria, Pretoria 0002, South Africa

***Abstract*—Systematic Luby Transform (fountain) codes are investigated as a possible incremental redundancy scheme for EDGE. The convolutional incremental redundancy scheme currently used by EDGE is replaced by the fountain approach. The results of the simulations performed for each incremental redundancy scheme show that the fountain approach outperforms the convolutional approach on the second retransmission when implemented on the EDGE platform. The results also indicate that if the packet sizes used by a specific platform is large enough the fountain approach will always outperform the convolutional approach.**

I. Introduction

Enhanced Data Rates for GSM Evolution (EDGE) [1], i.e. 2.75G, is a digital mobile phone technology that allows increased data transmission rates and improved data transmission reliability when compared to older technologies like General Packet Radio Service (GPRS) available to users of Global System for Mobile communications (GSM). EDGE uses nine modulation and coding schemes (MCS) to vary its data rates. Gaussian minimum-shift keying (GMSK) is used by the lower four coding schemes (similar to GPRS) and 8 phase shift keying (8PSK) for the upper five of its nine coding schemes. The fact that 8PSK can modulate 3 bits per symbol is the reason for the increase in the data rate of EDGE (when compared with GMSK that can modulate only 1 bit per symbol).

One frame block (1392 bits) in EDGE is transmitted over 4 bursts in a total time of 20 ms making the bit rate 278.4 kbit/s and yielding a bit rate of 69.6 kbit/s per time slot (this is true for uncoded using no headers). Due to the use of 8PSK modulation the receiver is now more complex and employs sophisticated equalization techniques. The actual maximum bit rate of EDGE is achieved by MCS9 and equals 59.2 kbit/s, the difference is due to EDGE implementation issues like headers and parity bits [1]. Since only MCS5 to 9 use 8PSK modulation these schemes were used for simulation purposes (only these schemes achieve a higher throughput when compared to GPRS).

EDGE also uses link adaptation (LA) [2] [3] and incremental redundancy (IR) [3] [4]. Link adaptation is the process of selecting the best MCS for the current channel conditions. Otherwise a too strong or weak code is used for the channel, hampering throughput. EDGE also uses incremental redundancy, which sends additional redundancy information to the receiver after a decoding failure. Both transmissions can now be combined for better decoding. EDGE combines these two techniques to maximize throughput. Edge uses the same convolutional code (CC) for each MCS, but varies the amount of punctures for each MCS. To implement incremental redundancy the puncturing patterns for each MCS is changed after each transmission. In short each MCS has a different amount of punctures per transmission and different puncturing patterns to enable retransmissions.

The convolutional approach [3] used for incremental redundancy can be replaced by a fountain approach [5] [6]. A digital fountain can be compared to a running tap. When a cup is filled it is not important which droplets land in the cup, but only that enough water is obtained to fill the cup. Where the droplets represent the encoded packets and the glass represent the receiver. So the receiver only require enough encoded packets to decode (same size as original message). The above metaphor also highlights another important property of digital fountain codes namely its ability to generate an infinite amount of unique encoded packets from the original source. So incremental redundancy can be implemented through a fountain code by only opening the same tap for the second transmission.

The article starts by giving an overview of a specific fountain code used for the incremental redundancy scheme implemented in this article namely Luby Transform (LT) codes [7] [8]. That same section will also discuss the noisy decoding [9] [10] [11] [12] of this code since fountain codes were originally developed for the erasure channel [13]. The LT section will be followed by a section discussing the two incremental redundancy approaches used in this article. The article will finish with computer simulation results and a conclusion.

II. LT Codes

The LT code is the first rate less code used to approximate a fountain and is discussed below [7] [8].

A. Encoder

An LT code is actually a dynamic low-density generator matrix (LDGM) code. The encoder can be described by using the following [7] [8]:
Each encoded parity bit p_n can be generated from the message bits $m_1, m_2, \cdots, m_k$ through the following two steps:

1) Choose the degree d_n of p_n randomly from a degree distribution $\kappa(d)$.
2) Choose at random d_n message bits, and set p_n equal to the bit wise sum, modulo 2, of the chosen message bits.

This procedure is exactly the same as multiplying a message with a dynamic random $\mathbf{G}$ matrix. An LT code is rate less, for it can keep on deriving parity bits without end through its dynamic $\mathbf{G}$ matrix. It is important to note here that a standard LT code is non-systematic [7], since only the parity bits that were generated is sent through the erasure channel. To create a systematic LT code [9] is quite simple; send the message bits before the parity bits. This code comes in handy for decoding noisy codewords. The result of the encoding procedure can also be presented by using a factor graph (graphical representation) as shown in Fig. 1. The left side nodes represent the original message bits $m_1, m_2, \cdots, m_k$ and the right side nodes represent the parity bits $p_1, p_2, \cdots, p_n$.

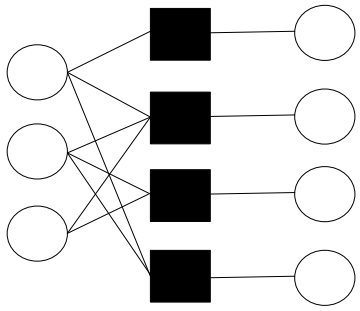

Fig. 1. A simple factor graph of an LT code

B. Degree Distribution

As seen from Section II-A the most important element of an LT code is the degree distribution $\kappa(d)$ it uses for encoding. The design of a good degree distribution of an LT code (for an erasure channel) is closely related to the classical problem of throwing k balls into K bins [7] [8]. From probability theory it is known that $k = K \cdot \ln(K/\delta)$ balls are required so that each of the K bins contains at least one ball with probability $1-\delta$. When performing the same analysis on a LT code the balls are analogous to the edges (in the factor graph) and the bins are the same as the message bits. The amount of edges required makes sense, all of the message bits must at least be connected to the graph to obtain successful decoding. Another important concept to keep in mind is that all the message bits needs to be covered (meaning that the erasure decoding process from [7] [8] may not halt). The decoding algorithm from [7] [8] will not halt if at least one parity bit has degree one after each iteration of the erasure decoding algorithm. Using the above it is easy to see that a good LT code will have $O(K \cdot \ln(K/\delta))$ edges, average left degree of $O(\ln(K/\delta))$ and encoding and decoding times of $O(K \cdot \ln(K/\delta))$. Take note that in this section k represents the amount of edges in a graph and K the amount of message nodes, this is done since in classic LT notation the amount of message bits is always referred to with a capital letter K. The ideal soliton distribution [7],

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

conforms to the above requirements. One of its primary characteristics is that after every iteration of the decoding process exactly one parity bit has degree one. This however works poorly in practice since minor fluctuations in decoding will lead to termination of decoding prematurely [7]. The robust soliton distribution has two extra parameters c and δ; it is designed to keep the expected number of degree-one parity bits equal to approximately:

$$S \equiv c \ln(K/\delta) \cdot \sqrt{K} \tag{2}$$

rather than one, throughout the decoding process. The parameter δ is a bound on the probability that the decoding fails to run to completion after a certain number K' of bits were received. The parameter c is a constant of order one. A positive function can now be defined [7]

$$\tau(d) = \begin{cases} \frac{S}{Kd} & \text{for } d = 1, 2, \cdots, (K/S) - 1 \\ \frac{S}{K} \ln(S/\delta) & \text{for } d = K/S \\ 0 & \text{for } d > K/S \end{cases} \tag{3}$$

To obtain the robust soliton distribution μ [7], ρ and τ are added and normalized:

$$\mu(d) = \frac{\rho(d) + \tau(d)}{Z} \tag{4}$$

where $Z = \sum_d \rho(d) + \tau(d)$. The number of encoded bits required at the receiving end to ensure that the decoding runs to completion, with probability at least 1 - δ, is $K' = KZ$. Some degrees have such low probabilities p_d that parity bits of degree d are absent. Instead these probabilities can be used to reinforce the amount of degree-one parity bits. This is done by introducing an extra factor [14]

$$v(d) = \begin{cases} \sum \mu(d_i) & \text{for } d = 1 \\ 0 & \text{for } d > 1 \end{cases} \tag{5}$$

where d_i represents the degree-i term of the distribution, satisfying the following inequalities

$$\begin{cases} \frac{\frac{1}{d_i(d_i-1)} + \frac{S}{Kd_i}}{Z} K < 1 & \text{for } 2 \le d_i \le \frac{K}{S} - 1 \\ \frac{1}{d_i(d_i-1)} \cdot \frac{K}{Z} < 1 & \text{for } (\frac{K}{S} + 1) \le d_i \le K \end{cases} \tag{6}$$

The improved robust soliton distribution $\kappa(d)$ [14], can now be defined as

$$\kappa(d) = \frac{\rho(d) + \tau(d) + v(d)}{Z} \tag{7}$$

where $Z = \sum_d \rho(d) + \tau(d) + v(d)$. In short an LT code can be uniquely defined through the distribution (K, δ, c) it uses

during encoding. The LT code used for the noisy channel was designed using the exact same method as described above since [15] shows some linkage between good erasure codes and other channels.

C. Noisy Decoder

To decode an LT code on a noisy channel it is important to construct the factor graph as shown in Fig. 1. A code node usually represents a received bit and is assigned a channel log-likelihood ratio (LLR) $\Lambda(m_n)$ in the case of binary phase shift keying (BPSK) this equals [11] [16] [17]

$$\Lambda(m_n) = -\frac{4}{N_0}\alpha_n r_n \tag{8}$$

where $\Lambda(m_n) = \ln\left[\frac{\Pr(m_n=0)}{\Pr(m_n=1)}\right]$, m_n is the random binary variable containing the prior probability that the n^{th} received bit is equal to either a 1 or a 0, r_n is the soft value of the n^{th} received bit, α_n is the average fading amplitude of r_n and N_0 is the single-sided noise power spectral density

As can be seen from Fig. 1 an LT factor graph consists of two node types, check and code nodes. Each of these types of nodes use different formula's to update its branches. The input LLR's of a code node are summed together to produce the output LLR (including channel LLR), and is described as [9] [10] [11] [12]:

$$\Lambda(m_o) = \sum_{i\neq o}\Lambda(m_i) + \Lambda(m_n) \tag{9}$$

For a check node

$$\Lambda(m_o) = 2\tanh^{-1}\left[\prod_{i\neq o}\tanh(0.5\Lambda(m_i))\right] \tag{10}$$

is used. Where $\Lambda(m_o)$ represent output branch LLR values, $\Lambda(m_i)$ are input LLR values and m_x is the appropriate conditional binary random variable associated with a branch. Take note that the output branch is not used in the calculation. To update a node each branch of a node needs to be updated with the above formula's. All branches are usually set to zero for the first iteration.

The order in which the nodes are updated can be chosen uniquely for each factor graph. Standard LDPC iterative belief propagation involves updating the code nodes first and then the check nodes. After such an iteration has been completed, the algorithm can stop or continue with another cycle. Usually the algorithm stops if a codeword is found or a certain predetermined amount of cycles are reached. The decoded bit value is calculated by performing an appropriate inverse "hard" decision (with the decision point being 0) on the final posterior LLR value of a bit calculated with

$$\Lambda(m_f) = \sum_{i}\Lambda(m_i) + \Lambda(m_n) \tag{11}$$

What is clear from the above is that this algorithm will work better on systematic than non-systematic codes. If the code is non-systematic the message nodes will all have initial channel LLR values equal to 0. The 0 values will hamper efficient belief propagation, making almost all messages equal to zero.

III. Incremental Redundancy

Incremental redundancy is implemented in EDGE by using a punctured convolutional code, see Fig. 2. During the first transmission the data (1) is encoded by a rate 1/3 convolutional encoder [1] [3] and punctured using puncture pattern 1 of MCS_x (2). Since only one transmission was made the combination stage is skipped (3). If the decoding (4) fails the loop starts again. The second transmission punctures the encoded data by using pattern 2 of MCS_x (2). Due to differences in the puncturing patterns the receiver will be able to fill in some punctures of the first transmission. If some bits (in the same bit positions) were sent twice, they are averaged (since they are LLR) to obtain a new soft value for that bit. These calculations are performed at the combination stage (3). Decoding takes place again, if decoding fails again the procedure needs to be repeated. The amount of cycles depend on the MCS scheme used. For MCS5 and 6, only two transmissions are possible (due to two puncture patterns). For MCS7 to 9 three transmissions are possible (due to three puncture patterns). If the maximum puncturing pattern is reached a decoding failure is declared (2). This is however not the only approach that can

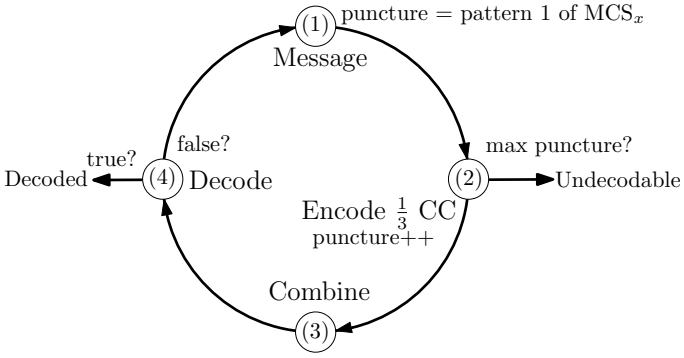

Fig. 2. IR implemented with punctured convolutional codes

be followed. EDGE can also use a fountain code to implement an IR scheme (see Fig. 3). LT codes can be used to replace

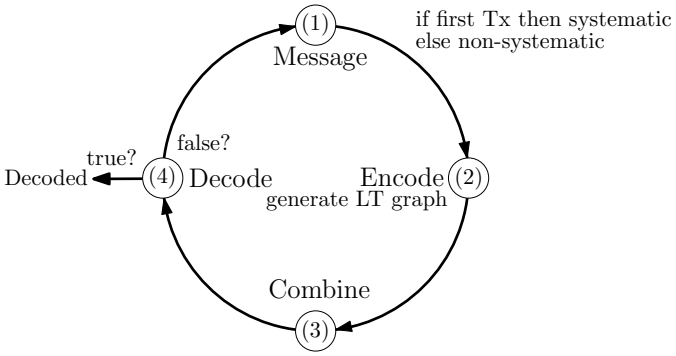

Fig. 3. IR implemented with fountain codes

the puncturing scheme [10] [12] [20] [21], e.g. a message can be encoded using a systematic LT code [9] (2) and if the

frame at the decoder (4) is invalid the transmitter retransmits the message using a different non-systematic LT structure (2). Now the receiver can use both transmissions for successful decoding (3). The cycle can be continued until a valid frame is received. In contrast to punctured convolutional codes, fountain codes have the ability to utilize all the retransmitted information for decoding and can have an endless amount of retransmissions. The reason being it grows dynamically (rate less), and does not fill up punctures.

A Raptor [18] [19], code can also be used, with the first transmission being a pre-code, such as an LDPC code. From there on the scheme is exactly the same. A systematic fountain code is used on the first transmission since noisy decoding of non-systematic codes performs poorly, see Section II-C. For the second transmission a non-systematic LT code (only parity) is used, because the graphs can be combined at the receiver, see Section II-A.

IV. Simulation: EDGE Platform

Some simulation specific parameters describing the EDGE platform needs to be defined so the results can be repeated; these parameters are given in Table I. The different channel models for EDGE are described in the documents 3GPP TS 45.005 V7.9.0 (2007-2) and 3GPP TS 05.05 V8.20.0 (2005-11). For the fountain approach a systematic LT code was used to implement the IR scheme. The LT codes used for each MCS scheme are different and can be uniquely defined by the distribution (K, δ, c). The LT codes used for this simulation can all be described by $(x, 0.01, 0.06)$, where $x = (468, 612, 468, 564, 612)$ for MCS5 to 9 respectively. The first transmission consists of x message bits and $1248 - x$ parity bits in the case of MCS5 and MCS6. All other retransmissions send 1248 parity bits. In the case of MCS7, MCS8 and MCS9 two data payloads are sent, each consisting of 612 bits. For the first transmission one data payload consists of x message bits and $612 - x$ parity bits. During the second transmission only parity bits are sent. Belief propagation was used as decoding algorithm, 200 iterations (see Section II-C).

TABLE I
Parameters defining EDGE platform used for simulations

Parameter	Value
Maximum Doppler Frequency	41.6Hz
Fading	Frequency selective
Channel Model	Typically Urban
Co channel interference	None
Channel state information	Estimated (real world receiver)

A. Simulated BLER Curves: EDGE

The results obtained via computer simulation of IR (EDGE system) using fountain and convolutional codes for each MCS scheme is displayed in Fig. 4 - 8.

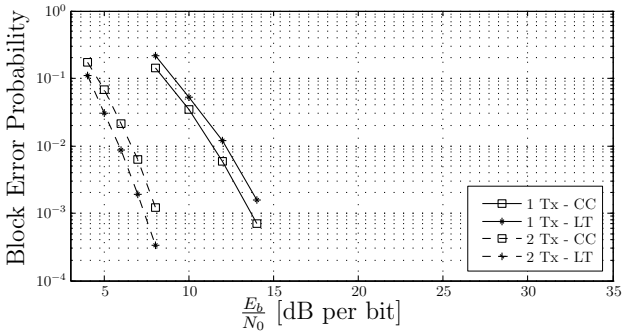

Fig. 4. BLER performance of IR implemented on EDGE (MCS5)

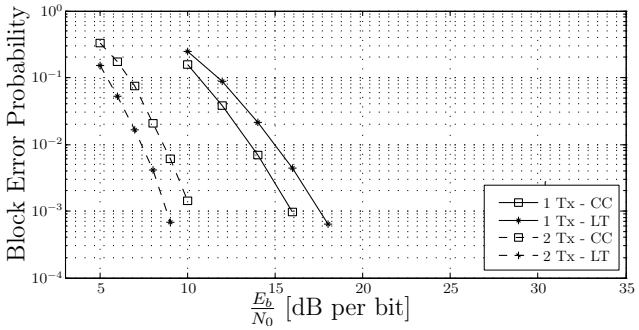

Fig. 5. BLER performance of IR implemented on EDGE (MCS6)

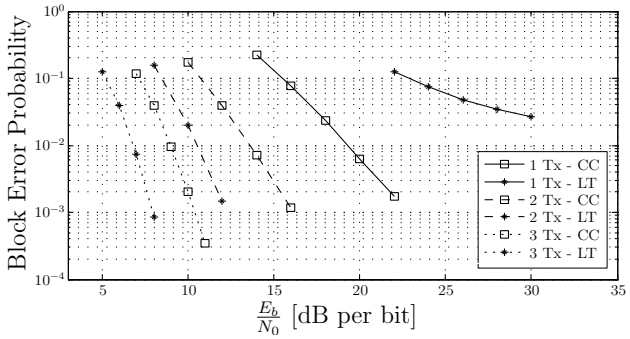

Fig. 6. BLER performance of IR implemented on EDGE (MCS7)

B. Discussion of Results

The actual dB improvement due to the fountain approach on each transmission for each MCS scheme can be found in Table II.

TABLE II
Improvement of fountain approach when compared to convolutional approach

MCS	TX 1 [dB]	TX 2 [dB]	TX 3 [dB]
5	-0.6	0.75	-
6	-1.75	1.5	-
7	-8	4	3
8	-4	1	1.2
9	-1.8	6	1.8

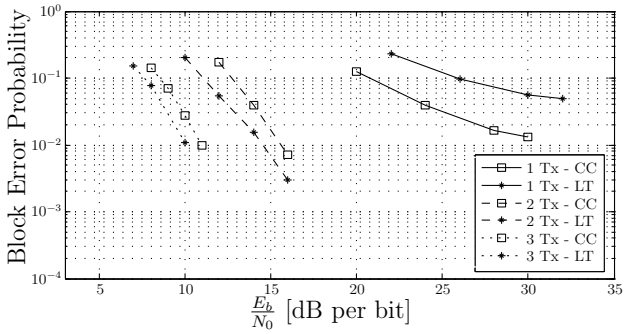

Fig. 7. BLER performance of IR implemented on EDGE (MCS8)

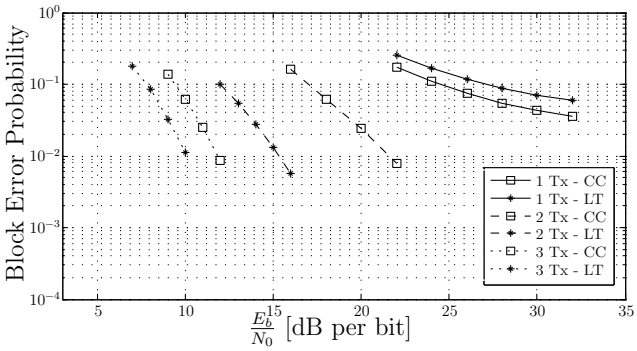

Fig. 8. BLER performance of IR implemented on EDGE (MCS9)

V. Conclusions

The following conclusions can be drawn from the observations gathered from the simulation results of Section IV-A:

1) Implementing incremental redundancy on EDGE using the convolutional approach outperforms the fountain approach on the first transmission, but the fountain approach starts gaining over the convolutional approach after the second retransmission. The above is true for all the MCS schemes.
2) The reason the first transmission of MCS5 and MCS6 using the fountain incremental redundancy approach performs only marginally worse than the convolutional approach, is because these schemes only have one data payload per Medium Access Control/Radio Link Control (MAC/RLC) block [1]. This implies longer codewords than in the case of MCS7, MCS8 and MCS9. Fountain codes perform better with longer codewords and they perform especially well when their code rate $R_c < 0.5$.
3) MCS7 and MCS8 perform very bad on the first transmission if fountain codes are used due to very small code lengths. MCS7, MCS8 and MCS9 have two data payloads per MAC/RLC block implying extremely small code lengths. Any kind of sparse graph code performs bad if the code length is small. In the case of the systematic LT code, the code can not cover all of its message bits with so few parity bits.
4) For MCS9 the code rate is 1, which makes the uncoded scheme (fountain scheme) almost as good as using a punctured convolutional code (since the code rate is 1 only message bits are sent during the first transmission if the fountain approach is used).
5) The fountain approach has an endless amount of retransmissions and uses all of the resent bits. The convolutional approach has some repetitions and does not have an endless amount of retransmissions. The drawback of the fountain approach is that the decoding structure grows larger after each transmission, slowing the decoding time.

References

[1] M. Hakaste, E. Nikula, and S. Hamiti, *GSM, GPRS and EDGE Performance*, 2nd ed. John Wiley & Sons, Ltd., 2003.

[2] S. Nanda, K. Balachandran, and S. Kumar, "Adaptation Techniques in Wireless Packet Data Services," *IEEE Commun. Mag.*, January 2000.

[3] M. H. Ismail, H. M. Mourad, and M. El-Soudani, "A newproposal for Link Adaptation in EDGE system based on the use of Turbo codes," *Twentieth National Radio Science Conference*, March 2003.

[4] R. V. Nobelen and N. Seshadri, "Incremental redundancy transmission for EDGE," *ETSI SMG 2*, August 1998.

[5] J. Byers, M. Luby, and M. Mitzenmacher, "A Digital Fountain Approach to Asynchronous Reliable Multicast," *IEEE Journal on Selected Areas in Communication*, vol. 20, no. 8, pp. 1528–1540, October 2002.

[6] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *Proc. ACM SIG-COMM*, pp. 56–67, August 1998.

[7] M. Luby, "LT codes," *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 271–282, 2002.

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

[9] T. Nguyen, L. Yang, and L. Hanzo, "Systematic Luby Transform codes and their soft decoding," *IEEE Workshop on Signal Processing Systems (SiPS)*, October 2007.

[10] T. Stockgammer, H. Jenkac, T. Mayer, and W. Xu, "Soft decoding of LT-codes for wireless broadcast," *in Proc. IST Mobile*, 2005.

[11] J. D. Vlok, "Sparse graph codes on multi-dimensional WCDMA platform," Master's dissertation, University of Pretoria, South Africa, 2007.

[12] T. L. Grobler, J. C. Olivier, and J. D. Vlok, "Fountain codes and their possible application in standards like GSM," *Southern African Telecommunication Networks and Applications Conference*, September 2007.

[13] P. Elias, "Coding for two noisy channels," *Information Theory, 3rd London Symp*, pp. 61–67, 1955.

[14] R. Tee, T. Nguyen, L. Yang, and L. Hanzo, "Serially concatenated luby transform coding and bit-interleaved coded modulation using iterative decoding for the wireless internet," *In Proceedings of IEEE VTC'06 Spring 1*, pp. 22–26, 2006.

[15] O. Etesami, "Relations between belief propagation on erasure and symmetric channels," *Proceedings of the International Symposium on Information Theory*, June 2004.

[16] R. H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, 1st ed. Chichester, UK: John Wiley & Sons Limited, 2004.

[17] L. Hanzo, T. H. Liew, and B. L. Yeap, *Turbo-Coding, Turbo Equalisation and Space-Time Coding for transmission over Fading Channels*, 1st ed. Chichester, UK: John Wiley & Sons Limited, 2002.

[18] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[19] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Transactions on Information Theory*, vol. 52, no. 5, pp. 2033–2051, May 2006.

[20] T. L. Grobler, "Fountain codes and their typical application in wireless standards like EDGE," Master's dissertation, University of Pretoria, South Africa, 2008.

[21] H. A. Ngo, T. D. Nguyen, and L. Hanzo, "HARQ Aided Systematic LT Coding for Amplify-Forward and Decode-Forward Cooperation," *in Proceedings of IEEE 71st Vehicular Technology Conference (VTC 2010-Spring)*, 2010.