Progressive encoding with non-linear source codes for compression of low-entropy sources

Francisco Ramirez Javega[1], Meritxell Lamarca [1,2], and Javier Garcia-Frias[2]

[1] Universitat Politecnica de Catalunya (UPC), Barcelona, Spain
[2] Dept. of ECE, University of Delaware, USA

***Abstract*-. We propose a novel scheme for source coding of non-uniform memoryless binary sources based on progressively encoding the input sequence with non-linear encoders. At each stage, a number of source bits is perfectly recovered, and these bits are thus not encoded in the next stage. The last stage consists of an LDPC code acting as a source encoder over the bits that have not been recovered in the previous stages.**

I. INTRODUCTION

In [1] a new family of non-linear graph-based codes named hybrid Low Density Product Check – Low Density Parity Check (LDPrC-LDPC) codes was introduced to compress binary asymmetric sources. Non-linear codes are potentially more powerful than linear ones, since they include the latter as a particular case. This potential is of special interest in the case of non-uniform sources. The reason is that linear codes possess identical distance profiles for all codewords, while non-linear codes have different distance properties for different codewords. This can be exploited to guarantee better distance profiles for the most likely information sequences, which should lead to better performance. In spite of this potential advantage, there has been relatively little work on non-linear codes, probably due to the fact that linear codes are known to be asymptotically optimum in channel coding for infinite block lengths.

The key idea for the definition of the non-linear codes in [1] was the use of non-linear nodes that perform the AND operation over their binary inputs. The proposed structure can be seen as a non-linear generalization of LDPC codes, which includes them as a particular case while maintaining many of their desirable features. Namely, i) the proposed non-linear codes can be graphically represented by means of a factor graph, ii) they can be decoded using belief propagation, and iii) their performance can be predicted, and the codes analyzed, using density evolution, and thus they can be easily designed when long block lengths are considered. This distinguishes the proposed scheme from the few non-linear codes that have been recently proposed for lossless and lossy compression, which are not easy to analyze and generalize. Preliminary results obtained for single source coding utilizing regular codes of the proposed family show that they easily outperform their linear counterpart (LDPC codes).

In this paper we unveil some of the key aspects that define the behavior of LDPrC-LDPC codes and propose a modification of the original scheme that can be employed to obtain a family of codes with better performance while keeping complexity low. The proposed procedure is named "progressive hybrid LDPrC-LDPC codes" and is based on the generation of the encoded bits by stages rather than all at the same time.

The proposed scheme allows to achieve compression of very low entropy sources with rate losses around 20%. While this figure might seem poor taking into account that optimum source encoding schemes exist in the literature (e.g. entropy coding), this scheme departs from these optimum procedures in that the complexity is moved from the encoder to the decoder. LDPC [2] and turbo codes [3,4] have been proposed in the past to achieve this goal, but the performance of linear codes for compression experiences significant degradation when the source entropy decreases (the smaller the entropy the higher the relative gap between the code performance and the theoretical limit).

This work has been partially funded by the Spanish Science and Technology Commissions and by FEDER funds from the European Commission: TEC2007-68094-C02-02, CSD2008-00010, 2009SGR-01236.

II. SYSTEM SET-UP

We consider the problem of almost lossless source coding of an asymmetric memoryless binary source with p(1) > p(0). We consider fixed-length block source codes, where a sequence of k information bits, $b_1 b_2 \ldots b_k$, is compressed into a codeword of $n < k$ bits, so that a code with compression rate $r = n/k$ is obtained.

III. REVIEW OF HYBRID LDPC-LDPRC CODES

Hybrid LDPrC-LDPC codes are constructed as a parallel concatenation of two block codes: a fraction α of coded bits is generated by a nonlinear Low Density Product Check (LDPrC) code and the remainder fraction, 1 − α, by a linear LDPC code.

The linear block is encoded as in a standard LDPC code. Defining a generator matrix **G** of size $k \times (1 - \alpha)\, n$, the encoding process can be expressed as:

$$\mathbf{c} = \mathbf{bG}, \quad \mathbf{c} = \left[c_1 \ldots c_{(1-\alpha)n} \right], \ \mathbf{b} = \left[b_1 \ldots b_k \right]. \tag{1}$$

For the LDPrC code, each coded bit, p_j, is obtained as the product (AND) of a few information bits b_i. Thus, we generate a codeword of length αn as

$$p_j = \prod_{i \in S_j} b_i, \qquad j = 1 \ldots \alpha n, \tag{2}$$

where S_j is the set of d_{pj} indices ($1 \le d_{pj} \le k$) that defines which information bits are used to generate each product bit p_j. Analogously to the LDPC code, the encoding process can be described in a compact form by defining a $k \times \alpha n$ generator matrix **P** whose (i,j) entry is 1 if the information bit b_i is

978-1-4244-6746-4/10/$26.00 ©2010 IEEE

employed in the computation of the coded bit p_j, and 0 otherwise. We thus represent the encoding process as

$$\mathbf{p} = \mathbf{b} \boxdot \mathbf{P},\ \mathbf{p} = [\mathrm{p}_1 \ldots \mathrm{p}_{\alpha n}],\ \mathbf{b} = [\mathrm{b}_1 \ldots \mathrm{b}_{\mathrm{k}}].\,, \tag{3}$$

where $\boxdot$ indicates the product over the bits selected by the corresponding column of matrix **P**.

The LDPrC-LDPC codeword is built as [**p c**]. Therefore, matrices **G** and **P** fully characterize the hybrid LDPC-LDPrC code. These matrices are sparse and have random appearance. They are characterized by the degree profiles of the bit nodes (both for the linear and non-linear parts), of the parity check and product check nodes. The analysis in [1] focused on the design of regular codes by means of density evolution. In this paper we propose a modification of the original LDPrC-LDPC codes resulting in a simple design of codes with irregular bit and product degree profiles.

The proposed codes are constructed using a sparse **P** and **G**. Hence, if the codeword is long enough and the matrix has been properly designed, there will be few cycles in the graph and belief propagation will provide a quite accurate approximation of maximum-a-posteriori decoding. The message passing equations for the variable nodes are the same as in LDPC codes, whereas new equations must be derived for the product nodes. As indicated in [1], if we consider the case of a two-input AND operator $z=x\cdot y$ and we denote by $L_{\times\to v}$ and $L_{v\to\times}$ the log-likelihood ratios (LLR) messages that go from the product node $\times$ to variable node v (being v either z, x or y) and vice versa, where $\mathrm{LLR(v)} = \log\frac{\mathrm{p(v=1)}}{\mathrm{p(v=0)}}$, then, we can write the decoding equations for this product node of degree two as

$$\mathrm{L}_{\times\to\mathrm{y}} = \log\left(\frac{1+2\mathrm{e}^{\mathrm{L}_{\mathrm{x}\to\times}+\mathrm{L}_{\mathrm{z}\to\times}}}{1+2\mathrm{e}^{\mathrm{L}_{\mathrm{x}\to\times}}}\right) \tag{4}$$

$$\mathrm{L}_{\times\to\mathbf{z}} = \mathrm{L}_{\mathbf{x}\to\times} + \mathrm{L}_{\mathbf{y}\to\times} - \log\left(1+\mathrm{e}^{\mathrm{L}_{\mathbf{x}\to\times}}+\mathrm{e}^{\mathrm{L}_{\mathbf{y}\to\times}}\right). \tag{5}$$

For product nodes of higher degree, the messages can be computed recursively from the expressions above.

IV. "ERASURE DECODING" OF THE LDPRC CODE

In this section we introduce a low complexity decoder for LDPrC codes that will be subsequently used in section V to improve the performance of hybrid LDPrC-LDPC codes. This decoder relies on the specific behavior of the AND operator and it does not exploit the knowledge of the source entropy.

IV.1. Erasure decoder

As indicated in [1], the product nodes are much more informative when they are equal to '1' than when they are equal to '0', since knowledge that the product is equal to 1 removes all uncertainty on the value of the operands (all inputs must be 1). By exploiting this fact some source bits can be easily recovered even if the source statistics are not taken into account, as shown next.

Based on the behavior of the AND operator, a very low complexity decoder for the LDPrC code can be envisaged. It is a decoder that only considers three possible values for the messages exchanged between the nodes: 1 (for a source bit that is perfectly known with LLR $+\infty$), 0 (for a source bit that is perfectly known with LLR $-\infty$) and '?' (for a source bit that has not been recovered yet). At the first decoder iteration, all source bits connected to a product node that has value '1' are identified; at the second iteration the knowledge of these bits is employed to recover some of the source bits that are '0' (according to the notation introduced in the next subsection those bits that are '0' and are connected to a type II product node are recovered in the second iteration). After the second iteration the decoding process is stopped, since no more source bits can be recovered by this low complexity decoder.

The performance of such a simple decoder is clearly suboptimal, since source statistics are not taken into account and soft values are not exchanged between nodes. However, it has the nice feature that no errors are made, i.e. the BER is zero for all bits identified as '1' or '0'. Based on the similarities of this decoder with that one employed in LDPC codes for the binary erasure channel, this low complexity decoder will be denoted in the sequel as the "erasure decoder". We next analyze the performance of this decoder depending on the degree of the product nodes and the source entropy.

IV.2. The behaviour of the AND operator of degree $\mathrm{d_p}$

Let us consider an AND operator of degree d_p and the amount of information on the input bits that can be inferred from the value of its product. Three different situations arise in the erasure decoder:

- Type I: All source bits are equal to '1'. Then their AND operation is also '1' and this single product bit identifies completely the value of the d_p source bits.
- Type II: All source bits but one are equal to '1', so that their AND operation is '0'. In this case, when the d_p -1 bits that are equal to '1' are perfectly known, the remaining bit can be identified as a '0'. However, the AND operator is not useful to recover information on any of the bits that are equal to '1' even if the bit that is '0' is perfectly known.
- Type III: At least two source bits are equal to '0'. In this case the only information that can be extracted from this product bit is that some source bits are zero. Perfect knowledge of any of the source bits does not convey any further information on the value of the other ones.

Given a source entropy and given a degree of the AND operator, the fraction of product nodes that correspond to each of these three types is perfectly determined. Figure 1 shows the probability that the product of d_p source bits belongs to each one of the three types listed above for the case of a source with entropy 0.1. Note that the fraction of nodes of type I is a decreasing function of the degree d_p, whereas the fraction of nodes of type II exhibits a maximum that can be shown to appear when $d_{\mathbf{p}} = -\log^{-1}(p(1))$.

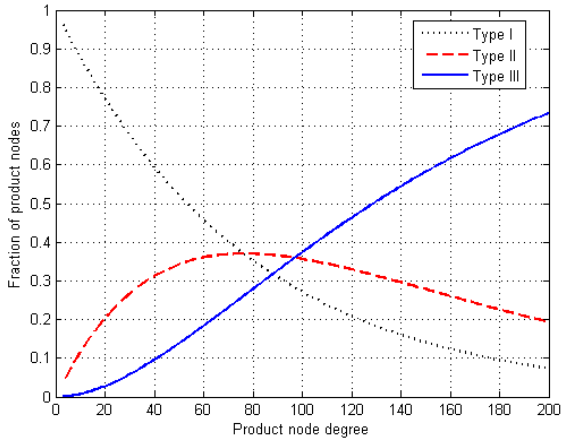

Fig. 1. Distribution of the type I, II and III products as a function of the product node degree for a source of entropy 0.1.

From the point of view of source compression it is thus clear that:

- The source bits that are '1' can only be recovered from the LDPrC code by the erasure decoder if they are connected to at least one type I product node, in which case it does not matter whether they have degree equal to 1 or higher than 1.
- The source bits that are '0' can only be recovered from the LDPrC code by the erasure decoder if they are connected to a type II product node and all other source bits in that product node have been recovered.

V. PROGRESSIVE ENCODING

V.1. Encoding procedure

The results in the previous section indicate that in the design of LDPrC codes there is a trade-off in the selection of the product node degree. The higher the product node degree the higher the compression rate of the code but also the higher the probability of generating Type III product nodes and the lower the probability of generating Type I product nodes.

Taking into account this trade-off, it is apparent that the performance of the non-linear code can be improved if source bits equal to '1' connected to Type I products have the smallest bit node degree (so no graph edges are "wasted" trying to determine them) and source bits equal to '0' connected to type II products also have a small degree (so they can be recovered but they have a small contribution to generate products that are '0'). This is possible if the encoding procedure for the hybrid codes proposed in [1] is modified so that it is performed in N successive stages. At each stage a subset of product bits is obtained, so if we denote as $\mathbf{p}_i$ the product bits generated in the *i-th* stage then

$$\mathbf{p} = \left[\mathbf{p}_1 \quad \cdots \quad \mathbf{p}_N\right] \tag{6}$$

The basic idea is to employ the erasure decoder described previously at the encoder to identify in the earliest possible stage the maximum number of source bits based on a reduced set of coded bits, and to devote the remaining coded bits to encode those bits that remain as '?' after the use of the erasure decoder in that stage. The procedure is as follows. At a first stage, all source bits are encoded with AND operators of degree d_{p1} so $n_1 = k/d_{p1}$ coded bits are generated:

$$\mathbf{p}_1 = \left[\mathrm{p}_1^{(1)} \quad \cdots \quad \mathrm{p}_{n_1}^{(1)}\right] = \mathbf{b} \boxdot \mathbf{P}_1 \tag{7}$$

where $\mathbf{P_1}$ is a matrix of size $k \times n_1$ that has one '1' in each row and d_{p1} '1's per column. Thus, the first edge for all source bits is defined and they all have degree one. Afterwards, the encoder tries to recover the source bits from these n_1 coded bits employing the erasure decoder. Denote as $(1-f_1)\cdot k$ the amount of source bits that can be recovered, and as $f_1 \cdot k$ the bits that remain as '?'.

At the second stage, those bits that remain as '?' are encoded with AND operators of degree d_{p2} and $n_2 = f_1 \cdot k/d_{p2}$ coded bits are generated. This procedure can be expressed as

$$\mathbf{P}_2 = \mathbf{\Pi}_2 \left[\mathbf{0}_2 \quad \mathbf{M}_2^{\mathrm{T}}\right]^{\mathrm{T}}$$
$$\mathbf{p}_2 = \left[\mathrm{p}_1^{(2)} \quad \cdots \quad \mathrm{p}_{n_2}^{(2)}\right] = \mathbf{b} \boxdot \mathbf{P}_2 \tag{8}$$

where $\mathbf{0_2}$ is a zero matrix of size $((1-f_1)\cdot k) \times n_2$, $\mathbf{M}_2$ is a matrix of size $(f_1 \cdot k) \times n_2$ that has one '1' in each row and d_{p2} '1's per column, $\mathbf{\Pi_2}$ is a permutation matrix of size $k \times k$ that rearranges the rows of $\mathbf{P_2}$ so that the non-zero rows are mapped to the source bits that remain as '?' after the first stage. Alternatively, this procedure can be written in terms of a permutation of the source bits:

$$\mathbf{p}_2 = \left[\mathrm{p}_1^{(2)} \quad \cdots \quad \mathrm{p}_{n_2}^{(2)}\right] = \left(\mathbf{b}\mathbf{\Pi}_2^{-1}\right) \boxdot \left[\mathbf{0} \quad \mathbf{M}_2^{\mathrm{T}}\right]^{\mathrm{T}} \tag{9}$$

so in this case the permutation matrix $\mathbf{\Pi_2}$ sorts the source bits placing first those bits that were recovered in the first stage and afterwards those that are encoded in the second stage with matrix $\mathbf{M}_2$. Note that the source bit nodes for the latter have degree 2. Note also that these two stages can be regarded as a single code with an irregular bit and product degree profile:

$$\left[\mathbf{p}_1 \quad \mathbf{p}_2\right] = \mathbf{b} \boxdot \left[\mathbf{P}_1 \quad \mathbf{P}_2\right] \tag{10}$$

Next, the encoder tries to recover the source bits that remained as '?' employing the 'erasure decoder' over the $n_1 + n_2$ coded bits of the equivalent code of rate $(n_1 + n_2)/k$.

Denote as $f_2 \cdot k$ the bits that remain as '?' after decoding the second stage. These bits are further encoded in stage 3 with $n_3 = f_2 \cdot k/d_{p3}$ AND operators of degree d_{p3} so their bit node degree is 3. Afterwards the 'erasure decoder' is applied again and the procedure follows with as many stages as desired.

Hence, for N stages $n_1+n_2+\ldots+n_N = k/d_{p1} + f_1 \cdot k/d_{p2} + \ldots + f_{N-1} \cdot k/d_{pN}$ coded bits are generated, so the total code rate for all stages is $1/d_{p1} + f_1/d_{p2} + \ldots + f_{N-1}/d_{pN}$, and $f_N \cdot k$ bits remain as '?' at the end of this procedure. This process can be written as using the same notation as in (3)

$$\mathbf{P}_i = \mathbf{\Pi}_i \left[\mathbf{0}_i \quad \mathbf{M}_i^{\mathrm{T}}\right]^{\mathrm{T}}$$
$$\mathbf{P} = \left[\mathbf{P}_1 \quad \cdots \quad \mathbf{P}_N\right] \tag{11}$$

Note that at every stage the fraction of bits that remain as '?' and need to be further encoded is smaller ($1 > f1 > f2 > f3 \ldots$). Furthermore, as most of the source bits being identified in early encoding stages are '1', the entropy of these remaining bits increases. This procedure can be continued until no bits remain as '?' or the entropy is very close to one.

Alternatively, this procedure can also be stopped at any number of stages and the bits remaining as '?' can be then encoded with a classic LDPC code of smaller size.

Note that although the coded bits are generated in several stages, the whole encoding process can be regarded as a single hybrid LDPrC-LDPC code with irregular degree profiles whose matrix depends on the source sequence to be compressed. This dependency appears through the permutation matrices $\mathbf{\Pi}_2$ … $\mathbf{\Pi}_N$, but the matrices $\mathbf{P_1}$, $\mathbf{M}_2$ … $\mathbf{M}_N$ are fixed.

It is important to remark that the same 'erasure decoding' process can be employed at the encoder and the decoder, so the decoder can recover the permutation matrices $\mathbf{\Pi}_2$ … $\mathbf{\Pi}_N$ that were employed in the encoder. Therefore, it has all the information required to retrieve the data dependent matrix that defines the matrix $\mathbf{P}$ in equation (3). At the receiver side, once this matrix has been obtained the fully-fledged decoder described in [1] for hybrid LDPrC-LDPC codes (i.e. the decoder that exploits the knowledge of source entropy and exchanges soft messages) can be applied to get the best performance.

V.2. Analysis

The improved performance of the proposed procedure is due to the fact that those source bits that are more difficult to recover are those that are most protected by the compression code, whereas those that can be easily recovered have smaller degree.

The simulation results in section VI indicate that the proposed procedure provides a low complexity method to compress very low entropy sources. It is well known that the design of linear codes for very high compression rates is difficult: in practical code designs the relative gap to the theoretical limits grows when the entropy decreases. However, the proposed procedure maps this problem into the generation of a set of coded bits by AND operators of high degree and the compression of a remaining sequence of bits of higher entropy, which can be efficiently encoded with a state-of-the-art LDPC code. For example, in the next section it is shown how a source with entropy 0.05 can be compressed using four non-linear stages and an LDPC code of rate 0.5. Thus, the problem of compressing a long source of low entropy is mapped into that of compressing a short sequence of higher entropy. The only drawback of this procedure from the code design point of view is that the LDPC code will operate with a smaller codeword length (its input block length is $f_N \cdot k$ rather than k), so a careful design of its parity check matrix must be made to guarantee good performance.

Although the encoding procedure described above results in a code with bit node degrees equal to i for those bits retrieved in the i-th stage, the generalization of the proposed procedure to obtain overall hybrid LDPrC-LDPC codes with more irregular degree profiles is straightforward. In the simulations in section VI the degrees of each stage d_{pi} were selected as ones that made the probability of product nodes of type I equal to 0.5, i.e. $\mathrm{d}_{\mathbf{p}}^{\mathbf{i}} = -log_2^{-1}\left(\mathrm{p}^{\mathbf{i}}(1)\right)$ or its closest integer, being $p^i(1)$ the probability of '1' at the input of the *i-th* stage. This criterion leads to an encoded sequence with equally likely 0s and 1s (which is a desirable feature in a compressed sequence), and preliminary simulations have shown that it provides the best results among the designs employing regular degree products. However, a deeper analysis is required to see if the performance can be improved by using other criteria in the selection of the product degree profile of each stage. Proceeding in this way, the number of coded bits generated at each stage keeps approximately constant ($n_1 \approx \ldots \approx n_N$), and on average at least half the bit nodes are determined at each stage.

The number of source bits that remain after each encoding stage (i.e. the values of $f_1 \ldots f_N$) depends on the information sequence, and it has a variance that is larger for latter encoding stages and decreases when the source word length k increases. Hence, the progressive encoding procedure is most effective when long codewords are considered. In order to tackle with this variable length without requiring the generation of a new encoding matrix for every information sequence to be encoded, a procedure must be proposed. In the simulation results in the next section the product degrees d_{pi} for each matrix $\mathbf{M}_2$ … $\mathbf{M}_N$ were originally designed according to the average input length in the corresponding stage, which can be computed from the knowledge of the source entropy. Then these matrices where extended adding additional rows corresponding to a number Δ_i of permutation matrices of size n_i, so the degree of the product nodes was increased to $d_{pi}+\Delta_i$ and the number of bits entering the i-th encoding stage was increased from $n_i \cdot d_{pi} = f_{i-1} \cdot k$ to $n_i \cdot (d_{pi}+\Delta_i)$. In those source words where some of these bits (usually the $n_i \cdot \Delta_i$ first ones) have been recovered in previous stages, no further encoding is necessary and the source can be replaced at the input of the product node by '1', thereby reducing the effective degree of the product node and getting it closer to the original value of d_{pi}. In those source words when the sequence length at the input of the i-th stage exceeds $n_i \cdot (d_{pi}+\Delta_i)$ an unrecoverable error will occur.

In the final linear coding stage, a similar procedure must be proposed to cope with the variable input length. In the simulations in section VI this issue was approached by increasing the LDPC codeword length while keeping the code rate fixed. This procedure results in a rate loss; the search for more efficient methods is a topic of current research.

Regarding implementation complexity, note that the operation at the encoder is very simple (only LDPrC-LDPC encoding and erasure decoding are required). It is also important to remark that the data-dependency is only introduced through the permutation matrices $\mathbf{\Pi}_i$, which act over the information bits. Hence, the conventional LDPrC-LDPC decoder presented in [1] can be employed at the receiver by introducing data-dependent interleavers. Note also that the size of submatrix $\mathbf{M}_i$ is reduced at every stage, and this fact can be also exploited to reduce decoder complexity.

VI. Simulation Results

The proposed procedure has been employed to compress two sources with entropy 0.091 and 0.05. In both cases four stages of coded bits generated by means of product nodes were employed, and the bits that remained as '?' after these stages were encoded with a linear code.

In the case of source entropy 0.091 a block of 103250 bits was compressed into 11950 coded bits, so a code rate of 0.1157 was obtained. After Montecarlo simulation of 16600

codewords, 9 wrong codewords were obtained and the average BER was 1.53×10^{-6}. The parameters for the non-linear stages are listed in Table 1. As indicated there, an average of 6670 bits remained to be identified after these four stages, and their average entropy was 0.70. These bits where then compressed with an LDPC code of rate 0.5 obtained from [2, 6]. Note that the code rate is smaller than the entropy of these bits. Operation in this regime was possible because the LDPC decoder did not operate alone: it was assisted by the non-linear stages, since the optimum decoder for the hybrid LDPC-LDPrC code was employed. In order to take into account the variability in the number of bits that remain undetermined at the end of the fourth stage the number of parity checks was increased from $f_4 \cdot k$=6670 to 9900, maintaining constant the total code rate. This performance compares favorably with that obtained when a single LDPC code is employed for the same task: in [2] an LDPC code was optimized to compress a source with entropy 0.091 and the minimum rate required for it was 0.125.

In the case of source entropy 0.05 a block of 1230000 bits were compressed into 85500 coded bits, so a code rate of 0.064 was obtained. After Monte Carlo simulation of 2600 codewords 1 wrong codeword was obtained and the average BER was 2.2×10^{-6}. The parameters for the non-linear stages are listed in table 2. In this case a linear code of rate 0.425 was employed [5].

VII. Conclusions

The application of the erasure decoder to LDPrC codes has been employed in this paper to propose a new source code whose encoding matrix (and the corresponding graph) depends on the information word. The new encoding procedure is implemented in successive stages and results in an adaptive graph where the nodes corresponding to those bits that are easy/difficult to recover have low/high degree. In spite of its data dependent nature the proposed encoding procedure has low complexity.

The simulation results evidence that the proposed codes outperform linear codes when compressing very low entropy sources. An analytical analysis and the search for more efficient and less complex methods to exploit adaptive graph construction of codes are topics of current research.

VIII. References

[1] D. Matas, M. Lamarca, and J. Garcia-Frias, "Non-linear graph-based codes for source coding," *Proc. Information Theory Workshop, 2009,* October 2009.

[2] A. D. Liveris, Z. Xiong, and C. N. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," IEEE *Communications Letters, IEEE*, pp. 440- 442, October 2002.

[3] J. Garcia-Frias and Y. Zhao, "Compression of correlated binary sources using turbo codes", IEEE *Communications Letters, pp. 417-419*, October 2001.

[4] J. Garcia-Frias and Y. Zhao, "Compression of binary memoryless sources using punctured turbo codes", *IEEE Communications Letters,* pp. 394-396, September 2002.

[5] D. H. Schonberg, "Practical Distributed Source Coding and Its Application to the Compression of Encrypted Data", *PhD dissertation, ECCS Department, University of California, Berkeley,* 2007

[6] S.-Y. Chung, "On the construction of some capacity-approaching coding schemes," *Ph.D. dissertation, Massachusetts Institute of Technology*, 2000

Table 1. Code parameters for source with entropy H=0.091 and k=103250. Rate = 0.1157

		NL stage 1	NL stage 2	NL stage 3	NL stage 4	LDPC
Product node degree ($d_{pi}+\Delta_i$)		59	29+5	14+5	7+5	
Bits to be encoded at that stage	p_1^i	0.9888	0.9767	0.9533	0.9074	0.8199
	Entropy	0.091	0.1596	0.2720	0.4449	0.6799
	Length ($n_i\, d_{pi} + n_i\, \Delta_i$)	103250	59500	33250	21000	9900
Number of coded bits (n_i)		1750	1750	1750	1750	4950
Contribution to overall code rate (n_i /k or LDPC code rate)		0.0169	0.0169	0.0169	0.0169	0.0479
Bits that remain as '?' after that stage ($f_i \cdot k$)		51379	25688	12968	6670	

Table 2. Code parameters for source with entropy H=0.05 and k=1230000 Rate=0.064

		NL stage 1	NL stage 2	NL stage 3	NL stage 4	LDPC
Product node degree ($d_{pi}+\Delta_i$)		123	61+7	30+7	15+7	
Bits to be encoded at that stage	p_1^i	0.9944	0.9775	0.9775	0.9552	0.9115
	Entropy	0.05	0.0888	0.1550	0.2637	0.4313
	Length ($n_i\, d_{pi} + n_i\, \Delta_i$)	1230000	680000	370000	220000	91100
Number of coded bits (n_i)		10000	10000	10000	10000	38781
Contribution to overall code rate (n_i /k or LDPC code rate)		0.0081	0.0081	0.0081	0.0081	0.0315
Bits that remain as '?' after that stage ($f_i \cdot k$)		613477	306614	153888	77893	