UNIVERSAL DECREMENTAL REDUNDANCY COMPRESSION WITH FOUNTAIN CODES

F.P.S. Luus*, A. McDonald and B.T. Maharaj*****

** Sentech Chair in Broadband Wireless Multimedia Communications, University of Pretoria, Lynnwood Road, 0002, Tshwane, South Africa. E-mail: fpsluus@gmail.com*
*** Dept. of Electrical, Electronic & Computer Engineering, University of Pretoria, Lynnwood Road, 0002, Tshwane, South Africa. E-mail: andre.mcdonald@up.ac.za*
**** Sentech Chair in Broadband Wireless Multimedia Communications, University of Pretoria, Lynnwood Road, 0002, Tshwane, South Africa. E-mail: sunil.maharaj@up.ac.za*

Abstract: A new universal noise-robust lossless compression algorithm based on a decremental redundancy approach with Fountain codes is proposed. The binary entropy code is harnessed to compress complex sources with the addition of a preprocessing system in this paper. Both the whole binary entropy range compression performance and the noise-robustness of an existing incremental redundancy Fountain code compression technique are exceeded. A new autocorrelation-based symbol length estimator, the Burrows-Wheeler block sorting transform (BWT) and Move-to-Front transformation (MTF) with a new entropy ordered MTF indices transformation reduces the binary entropy of a universal data source. The preprocessed input source is coded with a new modified incremental degree LT-code (Luby Transform) and a low-complexity decremental redundancy algorithm is used to compress the Fountain-coded source. The improved compression and robustness against transmission errors with our novel incremental degree puncturing decremental redundancy algorithm is shown. The universal (complex memory source) compression performance of the proposed system is shown to achieve appreciable compression.

Key words: symbol length estimator, EOI, entropy ordered indices, decremental redundancy, incremental degree LT-code, LT-IDP, LT-CLID, adaptive successive rate refinement.

1. INTRODUCTION

An integral part in the determination of the overall performance and usage efficiency in communication is source compression [1]. However, in wireless networks with residual channel noise and errors, the traditional data compression techniques, including Huffman [2], Lempel-Ziv [3], Run-length, Tunstall [4], arithmetic coding and entropy encoders that describe compressed data with variable-length symbols suffer from catastrophic error propagation for even single errors [5-6]. Reducing the input block length of these compressors, to minimise the effect of error propagation, results in poorer compression.

This paper extends the noise-robust binary entropy coder of our previous work in [7], which uses error correction codes to diminish the effects of catastrophic error propagation. A preprocessing system is added to produce binary compressible data from a source with complex memory so that the compressor in [7] can be harnessed as a universal compressor.

Linear error correction codes can achieve a coding rate equal to the source entropy for asymptotically large discrete memoryless sources [8]. Non-universal attempts for compression of memoryless sources with linear codes [9-11] performed poorly compared to traditional compression algorithms.

Sparse-graph low-density parity check (LDPC) block error correcting codes have been used for lossless compression of processed sources [12-15] and lossy compression of binary symmetric sources [16]. The variable length coding rates supported by Fountain codes [17] lend itself toward a more natural compressor implementation than the fixed-rate LDPC codes, so the focus in this paper is on rateless Fountain codes.

Belief-propagation decoding (BP) and a priori log-likelihood ratio information can be used for decompression, and the incremental redundancy iterative CLID (Closed-loop iterative doping) algorithm can utilise the BP variable node reliability information to improve compression speed and ratio. A novel two-stage approach in conjunction with CLID was presented [17] to reach comparable performance to the LDPC compressor.

The possibility of using only LT-codes (Luby Transform), which are patented [18], and CLID is mentioned in [17] as an inferior technique (LT-CLID), as well as the use of a decremental-redundancy approach with an LT-code and systematic precode (Raptor code).

Rectangular puncturing matrix and adaptive successive rate refinement strategies are proposed [19] for decremental redundancy compression with forward error correcting codes. Turbo codes have been used for

This research was funded by the Sentech Chair in BWMC, UP, Lynnwood Road, 0002, Tshwane, South Africa.
An earlier version of this paper was first published in IEEE WiMOB'08, Avignon, France, and with approval, this is an extended article.

decremental redundancy source coding [1], where encoded data were punctured sufficiently to produce compressed data that were losslessly decodable.

Universality is ensured with a new autocorrelation-based symbol length estimator (SLE), the Burrows-Wheeler block sorting transform (BWT) and Move-to-Front transformation (MTF) with a new entropy ordered MTF indices (EOI) transformation. The binary entropy of a universal source is reduced by exploiting higher order source entropy with a combination of these preprocessing techniques. The SLE improves the exploitation of source memory with the BWT and the proposed EOI after-MTF transform further reduces the binary entropy.

The decremental redundancy approach to compression with binary Fountain codes, originally proposed by us in [7], is included to make this paper self-contained. The purpose of the decremental redundancy approach is to increase the robustness to transmission errors over that of the existing incremental redundancy closed-loop iterative doping (LT-CLID) algorithm [17]. The noise-robust decompression of corrupted compressed data should minimize the symbol error probability in the decompressed data.

The noise-robust lossless compression algorithm in [7], which is based on a decremental redundancy approach with Fountain codes, is used for compressing binary memoryless sources. The whole binary entropy range compression performance and the noise-robustness of LT-CLID are again shown to be exceeded by the decremental redundancy approach.

It is shown that appreciable compression of the Calgary and Canterbury compression benchmark corpora [20] can be achieved with our universal compressor when it uses the proposed entropy coder.

Section 2 explains the different algorithms and transforms involved and how they are combined to operate as a universal data compressor. Section 3 introduces the SLE for determining a suitable symbol length for a BWT on a binary represented source with unknown content. A new after-MTF transform, namely the EOI transform, is proposed in Section 4 for improving the overall compression. The use of Fountain codes in a decremental redundancy setting and the performance variations with a systematic precode, a constant input degree distribution and a low-complexity puncturing distribution are revisited in Section 5. Finally, in Section 6, the combined universal compression performance of the proposed scheme is evaluated against existing compressors and algorithms.

2. COMPRESSION SYSTEM OVERVIEW

A memoryless entropy coder can be used in a universal data compressor as shown in Fig. 1, where the necessary preprocessing stages reduce the binary entropy of the input. As a universal data compressor this system processes a binary string input, with no a priori knowledge of the nature of the source, to give an approximately memoryless output, which is then segmentally compressed. The entropy of the input source is reduced for appreciable compression with the Fountain code and redundancy-varying algorithm. This study was limited to binary Fountain codes, and hence the binary entropy is reduced to increase the achievable compression.

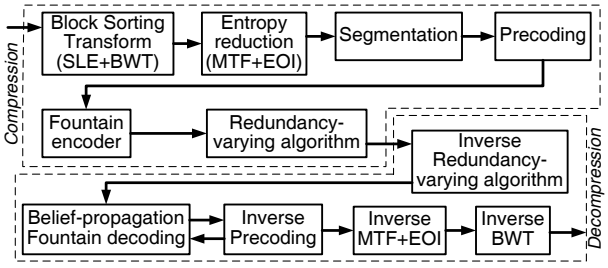

Figure 1: Universal compression system

The BWT [21] is used to shift redundancy in the source memory to redundancy in the marginal distributions. An SLE determines a source symbol bit length, which will increase the exploitation of memory with the BWT. The MTF [22] reduces the binary entropy of the BWT output, whilst a one-dimensional EOI transformation further lowers the entropy of the MTF transform output.

Compression is achieved through entropy encoding of the preprocessed source. A Fountain code in the form of a binary LT-code [23] is used to produce output symbols, in addition to a decremental redundancy algorithm to remove redundancy in the output, to produce the compressed data. The decompression involves the mapping of the compressed data to a reconstructed bipartite graph which is decoded with belief-propagation (BP). Lossless compression can be verified since decompression at the compressor is possible.

3. SYMBOL LENGTH ESTIMATION

The digital input source, for the universal compression application, can be primitively represented as a bit string. The BWT should receive a string of symbols that best reveal the memory present in the source. Most uncompressed real-world data sources are described with q-ary (non-binary) symbols, e.g. ASCII characters are 8 bit symbols.

The type of input data is not known a priori, so a mechanism is required to choose a suitable source symbol bit length, which produces the most memory in the source. The SLE produces an estimate of the symbol length which shows the highest correlation in the input source, and it is a heuristic approach.

Estimation of the symbol length is based on an autocorrelation over fixed sized blocks into which the input bit string is divided. The usefulness of a symbol

length to reveal source memory is determined, for a low-order SLE, as the correlation magnitude calculated for adjacent symbols. A higher-order SLE correlates non-adjacent symbols in a structured progressive manner. The algorithm details for a low-order estimator are explained as follows.

a) The input bit string is divided into M blocks of length $2N$ bits, for detecting symbol lengths of up to N bits. The last block can be padded to $2N$ bits, since M is generally a large number, thus the adverse effect on the accuracy of the symbol length estimation algorithm is negligible.
b) Replace all binary 0 symbols with -1, to ensure an antipodal autocorrelation.
c) Starting with the first block $m : 1 \le m \le M$, calculate the autocorrelation sum $\varphi_m(\tau) = \sum_{n=1}^{2N-\tau} b_n^m b_{n+\tau}^m$, where b_n^m is the value of the n^{th} bit of the m^{th} block, for all symbol lengths $\tau : 1 \le \tau \le N$.
d) Keep a mean symbol length autocorrelation sum $\phi_m(\tau) = m^{-1} \sum_{k=1}^{m} \varphi_k(\tau)$, for each symbol length τ after processing each block m. A mean symbol length autocorrelation is kept rather than a cumulative autocorrelation to facilitate an implementation that requires less memory.
e) Find $\tau_m = \arg\max_{\tau:\tau>1} \phi_m(\tau)$ and update a symbol length frequency counter $\vartheta(\tau_m) \leftarrow \vartheta(\tau_m) + 1$ after processing each block m.
f) After all M blocks have been processed, find the symbol length estimate $\hat{\tau} = \arg\max_{\tau:\tau>1} \vartheta(\tau)$. This is the symbol length that has the highest autocorrelation sum most frequently.

By calculating a mean autocorrelation function $\phi_m(\tau)$, the algorithm is able to distinguish a smaller constantly occurring autocorrelation peak among larger peaks with an average value of zero (when averaged over all blocks). By using the most frequently occurring peak over M functions, instead of solely using the peak of the final mean autocorrelation function $\phi_M(\tau)$, we avoid a situation in shorter files where a few blocks with very large autocorrelation peaks at the incorrect lag, lead to incorrect estimates. This situation might occur in shorter 24 bits-per-pixel image files with a header that contains strongly correlated data with 8 bits per symbol.

The performance of the preprocessing stages in the exploitation of memory, for different source symbol bit lengths for the Calgary and Canterbury corpora [20], were determined. This performance dependency on symbol length was used as a benchmark to evaluate the accuracy of the symbol length estimator. The statistics of $\vartheta_s(\tau_m) = \vartheta(\tau_m) \Big/ \sum_{t=1}^{N} \vartheta(\tau_t)$ are shown in Fig. 2 where multiples of 8 bit symbols are most prominent. Accurate source symbol length estimation ensures that the BWT will remove as much memory-based redundancy as possible, which maximises the compression with a binary fountain code and redundancy-varying algorithm.

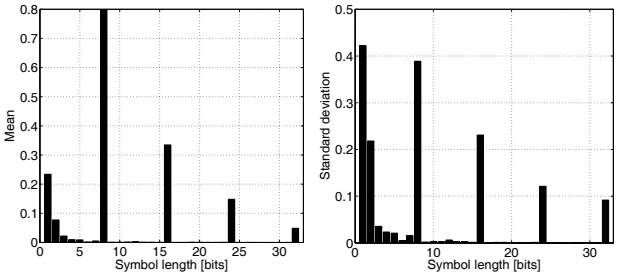

Figure 2: Statistical analysis on the symbol estimation results

The best symbol length is the one that allows for the maximal removal of redundancy. The SLE results in Fig. 2 show a good correlation with the removed redundancy in Fig. 3, showing the effectivity of the symbol length estimation used in the preprocessing.

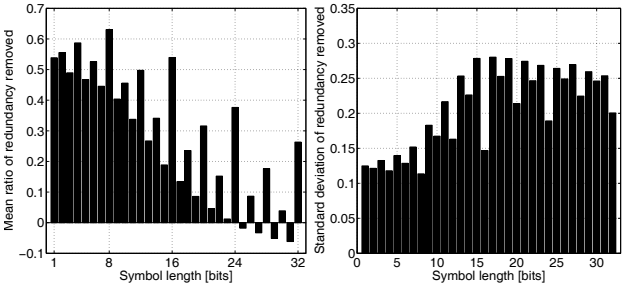

Figure 3: Binary entropy reduction performance for the Calgary and Canterbury corpora for varying symbol lengths

4. ENTROPY ORDERED INDICES TRANSFORM

The MTF transform replaces the normally high binary entropy representations of the symbols in the BWT output, with low value number representations. An optimal preprocessor for the binary Fountain code compressor should minimize the binary entropy of the input. The MTF transform with entropy ordered indices replaces the normally high binary entropy representations with low binary entropy representations.

4.1 Entropy ordered recurrence list creation

As a recurrence time coding scheme, the MTF transform replaces a symbol with the number of distinct symbols that have appeared since the previous occurrence of the current symbol. For a source with memory, the BWT output contains segments of a repeated symbol so that the MTF transform produces a low value number output, with the most common index value being 0.

The EOI transform replaces the recurrence index values in the MTF transform output with entropy ordered index values. A one-dimensional transformation vector must be generated to map the normal MTF transform output values to low entropy values.

Let $W=\{w_1,w_2,\ldots,w_{2^\alpha}\}$ denote the entropy ordered index list, where the amount of unique symbols in the BWT output is $|A|$ and $\alpha=\lfloor \log_2|A| \rfloor+1$. For a binary one biased transform, the elements w_i of list W are ordered according to the amount of binary ones, $n_1(w_i)$, in the binary representation of w_i. The output of concatenated indices has as few binary ones as possible to minimise the binary entropy.

a) Initialize the list W so that $w_i=i-1$ for $i\in[1,2^\alpha]$.
b) To sort W according to element-wise entropies, create an intermediate vector $G=\{g_0,g_1,\ldots,g_{2^\alpha-1}\}$, where $g_i \leftarrow n_1(w_{i+1}),\ \text{for } i\in[0,2^\alpha-1]$.
   i) Start with $g_0\leftarrow 0$, since $n_1(w_1)=0$.
   ii) Increment the list index counter, i.e. $i\leftarrow i+1$.
   iii) Generate the next elements g_i to g_{2i-1} so that $g_{i+j}\leftarrow g_j+1,\ \text{for } 0\le j<i.$
   iv) Update the index counter to the new size of G, that is $i\leftarrow 2i$.
   v) Repeat the generation process from iii) to v) until $i=2^\alpha-1$ is satisfied.

   This algorithm is $96\cdot 2^\alpha/\left(2\cdot 2^\alpha+\log_2\cdot 2^\alpha\right)\approx 48$ times less computationally complex than a brute-force list generation, for 32 bit integer index values.
c) Sort $W: g_{w_{i-1}}\le g_{w_i},\ \text{for } 1<i\le 2^\alpha$ according to G.

4.2 Entropy ordered indices transformation

The entropy ordered indices transform output is $V=\{v_1,v_2,\ldots,v_{|A|}\}$, where $|A|$ is also the length of the MTF transform output. Set each element v_i of V to the value w_j in W, where the original index value is $u_i=j-1$, in the MTF transform output U, such that $v_i=w_{u_i+1},\ \text{for } 1\le i\le|A|.$

4.3 Inverse transformation

The entropy ordering of the index values is a repeatable process, and an inverse transform exists. The original MTF transform output U can be obtained from the entropy ordered indices transform V as follows.

a) Generate the same entropy ordered index list W at the receiver. Determine $W'=\{w'_1,w'_2,\ldots,w'_{2^\alpha}\}$ where $w'_{w_i+1}=i,\ \text{for } 1\le i\le 2^\alpha.$
b) Reconstruct the original MTF transform output $U: u_i=w'_{v_i+1}-1,\ \text{for } 1\le i\le 2^\alpha$ from V and W'.

4.4 Analysis

The EOI transform will reduce the entropy of the MTF transform as shown in the following theorems.

Lemma 1: There exists a set of number pairs $J=\{(a,b)\in\mathbb{Z}\times\mathbb{Z}: a>b\}$, where $|J|\ge 1$ and $\beta=\lfloor \log_2 a \rfloor+1\ge 4$, so that $n_1(a)<n_1(b)$.

Proof: Let the binary equivalent the pair $(a,b)\in J$ be represented in $\mathbb{F}_2^\beta$, and $h(p,q,r,s):\mathbb{F}_2^q\mapsto\mathbb{F}_2^{(s-r+1)}$ denote the binary value formed by the bits $p_k: k\in[r,s]$ corresponding to positions $2^{(k-1)}$ in $p\in\mathbb{F}_2^q$.

The set of lengths of most significant sub-symbols $\{\vartheta\in[1,\beta-2]: h(a,\beta,\beta-\vartheta+1,\beta)=h(b,\beta,\beta-\vartheta+1,\beta)+1\}$ ensures that $a>b$, where $n_1(h(a,\beta,\beta-\vartheta+1,\beta))=n_1(h(b,\beta,\beta-\vartheta+1,\beta))+1$.

The size of the set $Y=\{y\in\mathbb{F}_2^{(\beta-\vartheta)}: n_1(y)=m\}$ with m binary ones is $|Y|=(\beta-\vartheta)!/((\beta-\vartheta-m)!m!)$. Then for $n_1(h(b,\beta,1,\beta-\vartheta))\ge n_1(h(a,\beta,1,\beta-\vartheta))+2$ the size of J is $|J|=\sum_{\vartheta=1}^{\beta-2}\sum_{i=2}^{\beta-\vartheta}\frac{(\beta-\vartheta)!}{(\beta-\vartheta-i)!i!}\sum_{j=0}^{i-2}\frac{(\beta-\vartheta)!}{(\beta-\vartheta-j)!j!}$.

Thus $n_1(b)>n_1(a)$ for $a>b$, and $|J|\ge 1$ for $\beta\ge 4$.

Theorem 1: For MTF transform U with the normal list $\{q_i=i-1: i\in[1,2^\alpha]\}$ of index values, the relation $H(V)=H\left(\sum_{i=1}^{|A|}\frac{n_1(v_i)}{\alpha|A|}\right)\le H(U)=H\left(\sum_{j=1}^{|A|}\frac{n_1(u_j)}{\alpha|A|}\right)$ holds.

Proof: By design, the entropy ordered list indices $\{w_i\}$ have the relation $n_1(w_i)\le n_1(w_j),\ \text{for } i<j$. According to Lemma 1 the relation for the normal MTF transform indices $\{q_i\}$ are $n_1(q_i)>n_1(q_j),\ \text{for } (q_j,q_i)\in J$.

For U with a ratio r_i of occurrences of index value q_i we have $\sum_{i=1}^{2^\alpha} r_i=1$. The MTF transform is characterised by $r_i>r_{i+1}$, and the following reasonable assumptions $\sum_{i=1}^{2^{\alpha-1}} r_i \gg \sum_{j=2^{\alpha-1}+1}^{2^\alpha} r_j$ and $\sum_{i=1}^{2^{\alpha-1}} r_i n_1(q_i)\big/\alpha\le 0.5$ are made.

Since $r_i>r_{i+1}$ the MTF transform indices will then result in a transform with more binary ones, for a binary one biased approach, and consequently a higher entropy such

that $\sum_{i=1}^{2^\alpha} r_i n_1(w_i)\big/\alpha \le \sum_{i=1}^{2^\alpha} r_i n_1(q_i)\big/\alpha$. An approximation of $1-\sum_{i=1}^{2^\alpha} r_i n_1(w_i)\big/\alpha \ge \sum_{i=1}^{2^\alpha} r_i n_1(q_i)\big/\alpha$ is

$$\begin{aligned}
1-\sum_{i=1}^{2^\alpha}\frac{r_i n_1(w_i)}{\alpha} &= 1-\sum_{i=1}^{2^{\alpha-1}} r_i\frac{n_1(w_i)}{\alpha}-\sum_{j=2^{\alpha-1}}^{2^\alpha} r_j\frac{n_1(w_j)}{\alpha}\\
&\approx 1-\sum_{i=1}^{2^{\alpha-1}} r_i\frac{n_1(w_i)}{\alpha} \quad \left(\text{since} \quad \sum_{i=1}^{2^{\alpha-1}} r_i \gg \sum_{j=2^{\alpha-1}+1}^{2^\alpha} r_j\right)\\
&\ge 0.5 \quad \left(\text{since} \quad \frac{n_1(w_i)}{\alpha}\le 0.5, \ \text{for } 1<i<2^{\alpha-1}\right)\\
&\ge \sum_{i=1}^{2^{\alpha-1}} r_i\frac{n_1(q_i)}{\alpha} \quad \left(\text{since} \quad \sum_{i=1}^{2^{\alpha-1}} r_i\frac{n_1(q_i)}{\alpha}\le 0.5\right)\\
&\approx \sum_{i=1}^{2^\alpha} r_i\frac{n_1(q_i)}{\alpha} \quad \left(\text{since} \quad \sum_{i=1}^{2^{\alpha-1}} r_i \gg \sum_{j=2^{\alpha-1}+1}^{2^\alpha} r_j\right).
\end{aligned}$$

The relation between the normal list indices $\{q_i\}$ and entropy ordered indices $\{w_i\}$ is then given as $\sum_{i=1}^{2^\alpha} r_i\frac{n_1(w_i)}{\alpha} \le \sum_{i=1}^{2^\alpha} r_i\frac{n_1(q_i)}{\alpha} \le 1-\sum_{i=1}^{2^\alpha} r_i\frac{n_1(w_i)}{\alpha}$.

For the binary entropy function $H(\cdot)$ and the above relation, the theorem holds for the assumptions made.

4.5 Experimental results

The binary entropy reductions $H(U)$ and $H(V)$ were measured for the Calgary and Canterbury corpora [20] with the application of the preprocessing, as shown in Table 1. The SLE, BWT and MTF transform were used in addition to the entropy ordered indices transform (EOI) for determining $H(V)$. The binary entropy values in Table 1 are multiplied with the ratio of the transform filesize to the original filesize. For a sufficiently large input source with an original BWT symbol bit length being greater than $\lfloor \log_2|A| \rfloor + 1$, the MTF transform filesize is smaller than the original filesize.

Table 1: Preprocessing binary entropy reduction comparison

Corpus	**MTF transform**		**MTF+EOI transform**	
	$H(U)$	Std. Dev.	$H(V)$	Std. Dev.
Calgary	0.4468	0.1984	0.3947	0.2054
Canterbury	0.4149	0.0845	0.3639	0.0824
Calgary	Additional binary entropy reduction with EOI transform			0.0521
Canterbury				0.0510

The EOI transform further reduces the binary entropy of real-world sources by more than 5% when compared to the use of only the MTF transform as post-BWT stage. The preprocessing appreciably reduces the binary entropy from more than 0.9 to less than 0.4, such that a binary entropy coder can achieve meaningful compression. The effectivity of the EOI transform in reducing binary entropy of the Canterbury corpus is displayed in Fig. 4.

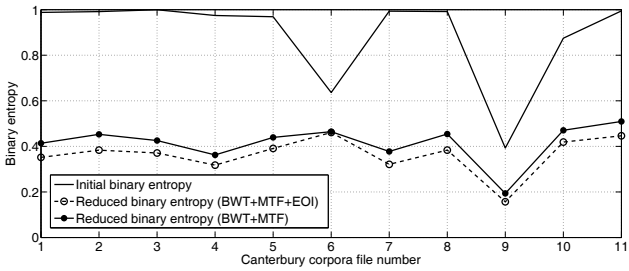

Figure 4: Binary entropy reduction of Canterbury corpus with preprocessing stages

5. DECREMENTAL REDUNDANCY COMPRESSION WITH FOUNTAIN CODES

The Fountain code compressor that we developed in [7] is described in detail in this section, and the benefits of the code are shown. Entropy encoding of the preprocessed source is required for compression to be achieved. A Fountain code in the form of a binary LT-code [23] is used to produce output symbols in addition to a decremental redundancy algorithm to remove redundancy in the output to produce the compressed data. During decompression the compressed data is mapped to a reconstructed bipartite graph which is decoded with belief-propagation (BP).

5.1 Belief-propagation decoding

Appreciable compression is possible with the soft-decoding of the LT-code when a priori log-likelihood information is included [17]. The BP algorithm is a message-passing algorithm, and the natural log-likelihood domain is used for all message calculations.

The log-likelihood ratio of a random variable x_n, describing the bit value associated with a node n, is $L(x_n) = \ln\left(p(x_n=0)/p(x_n=1)\right)$. The compressed data includes the quantized uniform a priori log-likelihood ratio value $L(x_i)$ for the original input source. The log-likelihood $L(x_o)$ for each individual output node o is set at the start of the decoding based on the unpunctured output values as $L(x_o) = \pm L_\infty$, or $L(x_o) = 0$ if punctured. The theoretical maximum achievable compression with a binary entropy encoder in terms of the a priori log-likelihood ratio is $H\left(1\big/\left(1+e^{L(x_i)}\right)\right)$.

Messages sent in the ℓth iteration are denoted by $m^{(\ell)}$. The message from output node o to input node i is denoted by $m_{oi}^{(\ell)}$, and $m_{io}^{(\ell)}$ denotes the message from the input node i to the output node o. Message updates during each iteration for BP are given as follows:

$$m_{oi}^{(\ell)} = 2\tanh^{-1}\left(\tanh\left(\frac{L(x_o)}{2}\right)\prod_{i'\in I_o\setminus\{i\}}\tanh\left(\frac{m_{i'o}^{(\ell)}}{2}\right)\right) \quad (1)$$

$$m_{io}^{(\ell)} = \begin{cases} L(x_i), & \text{if } \ell=0 \\ L(x_i)+\sum_{o'\in O_i\setminus\{o\}} m_{o'i}^{(\ell-1)}, & \text{if } \ell>0 \end{cases} \quad (2)$$

where O_i is the set of output nodes incident on input node i and I_o the set of input nodes incident on output node o.

The value $x_i = \chi^{-1}\left(\text{sign}\left(L(x_i)+\sum_{o\in O_i} m_{oi}^{(\ell)}\right)\right)$ is the bit value of an input node i for the isomorphism $\chi : GF(2) \mapsto \{-1,+1\}$, where $\chi(0)=+1$ and $\chi(1)=-1$.

5.2 Compressor

The procedure followed for compression of a memoryless source is explained as follows.

1) Choose a coding rate Z, where $Z-1$ is the proportional output overhead of the LT-code.
2) Create a robust soliton degree distribution sampling pool and construct the bipartite graph, with K input nodes and KZ output nodes. A pseudo-random number sequence is used with the seeding value included in the compressed data.
3) Map all source input bits x_i to each associated input node i.
4) Encode all output values x_o, according to $GF(2)$ addition, so that $x_o = \sum_{i\in I_o} x_i$.
5) Determine the uniform a priori input log-likelihood ratio $L(x_i)$ for the entire input source.
6) Puncture or remove uniformly at random, a fraction $1-\chi$ of the output values. In practice χ is adjusted with adaptive successive rate refinement [19], which is a binary search algorithm, of depth γ.
   i) Choose two puncturing extremes with $\chi_H = 1$ and $\chi_L = H\left(1\Big/\left(1+e^{L(x_i)}\right)\right)$.
   ii) Determine the mean $\chi = 0.5(\chi_L+\chi_H)$ and randomly puncture $KZ(1-\chi)$ output values.
   iii) With the decompressor, verify that lossless decompression is possible. In the case that it is not, set $\chi_L = \chi$, else set $\chi_H = \chi$.
   iv) Repeat the search $\gamma-1$ times with the updated puncturing ratio extremes.
7) Form the compressed data with the unpunctured output values and input log-likelihood ratio, in addition to necessary overhead information such as the original source length and utilized compression parameters.

5.3 Decompressor

The decompressor receives the compressed data as input which contains the unpunctured output values and input a priori log-likelihood ratio. Lossless decompression is ensured for perfect transmission of the compressed data. The decompression procedure is given as follows.

1) Recreate the bipartite graph by using the same seeding value and creation parameters.
2) Map the available output values to the graph by retracing the puncturing order.
3) Execute a limited amount of BP iterations.
4) Calculate the associated input node values, which will be the preprocessed source input.
5) Determine the validity of the input by calculating and comparing parity or checksum information included in the compressed data header.

5.4 Robust Soliton optimisation for compression

The Robust Soliton c and δ variables were optimised as in Fig. 5, for all fountain codes used. A varying binary entropy source was generated for this experiment, where 10000 bit blocklengths were compressed, such that it contained 16 blocks in the useful binary entropy range of 14% to 88%.

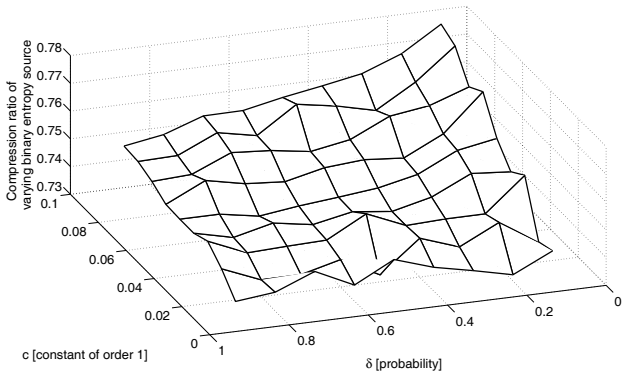

Figure 5: Detailed optimising of Robust Soliton variables for maximum compression.

For all the robust soliton output degree distribution implementations $c = 0.05$ and $\delta = 0.4$ were used, as it is in the good performance region of Fig. 5.

5.5 Puncturing influence on decoding

The decremental redundancy approach involves the use of puncturing to increase the compression ratio, but the removal of output values in the compressed data affects the bipartite graph and belief-propagation decoding.

The amount of incident opposing nodes is called the degree d_n of a node n. Let the total amount of edges emanating from the input nodes be denoted by E. If O_i contains an output node o with a punctured value, the input node i has modified degree $d_i' = d_i - 1$ and the

new effective set of incident output nodes is $O_i' = O_i \setminus \{o\}$, the amount of bipartite graph edges are reduced to $E' < E$. The reason is that the log-likelihood ratio for output nodes with punctured values is $L(x_o) = 0$, such that $m_{oi}^{(\ell)}$ cannot contribute to the recovery of the value of any node present in the bipartite graph.

Theorem 2: The probability of an input symbol value being unrecoverable is increased during puncturing so that $p(d_i' = 0) > p(d_i = 0)$ for any input node.

Proof: Let the resulting amount of edges, after puncturing be denoted by E'. The input degrees are Poisson distributed for a large E, so that the probability of an unconnected input node is $\mathrm{p}(d_i' = 0) = \frac{e^{(-E'/K)}(E'/K)^0}{0!}$ $= e^{(-E'/K)} > e^{(-E/K)} = p(d_i = 0)$.

5.6 Coding amendments

Changes to the input- and output degree distributions are presented in this section to reduce the probability $p(d_i' = 0)$ of a zero degree input node i. Several degree distribution modifications were investigated as follows.

Systematic precoding: The average input degree is increased with a systematic precode that adds high degree output symbols. The systematic precode output consists of the original input and additional output symbols. The precode is simultaneous since the LT-code input involves only the systematic part of the precode output. The number of unique input nodes participating in each of the additional high degree output nodes is maximized to ensure the lowest $p(d_i' = 0)$.

Effectively, with the LT-coding done only on the systematic part, the combination of the precode and LT-code produces a modified coding rate and output degree distribution. The effective bipartite graph can be used for simultaneous decoding of the precode and the LT-code.

Input degree equalization: The LT-code input degree distribution can be changed from a Poisson distribution, for a large amount of edges E, to a constant distribution. The input nodes will have an approximate equal degree of $d_i = E/K$, so that output value puncturing for this amendment will produce a minimum $p(d_i' = 0)$ for an unchanged output degree distribution and coding rate.

Precoding and equalization: The modification of the input degree distribution to a constant distribution used in conjunction with a systematic precode to increase the probability of lossless decoding of the input for the same puncturing rates.

Incremental degree puncturing compressor: A low-complexity puncturing distribution can be used with a modified LT-code to significantly improve the decremental redundancy compression performance. This useful puncturing distribution is used in the LT-code incremental degree puncturing compressor (LT-IDP). An incremental degree puncturing distribution produces much better compression than a random or decremental degree puncturing distribution. With an incremental degree puncturing distribution, $KZ(1-\chi)$ values of the lowest degree output nodes are removed. The iterative puncturing algorithm searches for an unpunctured value associated with an output node with the lowest degree of all the output nodes with unpunctured values.

Normally output degrees are sampled uniformly at random from a degree pool for LT-coding, so that:

1) For an increasing output size, the output degree distribution will be approximately representative of the robust soliton degree distribution.
2) Transmission burst errors will not as severely reduce the lossless decoding probability.

In order to reduce the linear computational complexity of finding the lowest degree output node with an unpunctured value to a constant complexity, an incremental degree LT-code can be used. LT-code output degrees are sampled incrementally and a random interleaver may be used prior to transmission to increase lossless decoding probability in the case of burst-type noise channels. The addition of a precode and/or the modification of the input degree distribution does not increase the performance of this puncturing distribution.

5.7 Performance comparison

The binary compression performance of the various discussed existing and proposed compressors are compared in this subsection.

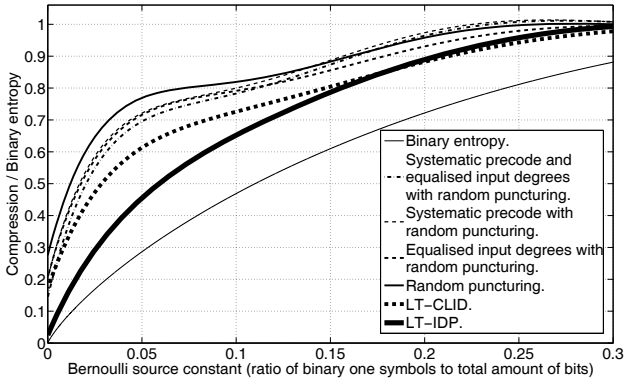

Figure 6: Comparative memoryless source compression performance

The comparative memoryless source compression performance for $K = 10000$ bit input blocks, $Z = 1.1$ (except for LT-CLID), with an LT-code with a robust soliton output degree distribution with $c = 0.05$ and $\delta = 0.4$ (as previously optimised), belief propagation

decoding with a maximum LLR of $L_\infty = 25.0$ and 8 a priori LLR bits, is given in Fig. 6. The source is a collection of Bernoulli sources with varying binary entropies.

The LT-CLID compressor uses 300 BP iterations (per epoch in a $\gamma = 7$ binary search) with a doping delay of $D_d = 10$ iterations and a doping frequency of $D_f = 1$ iterations per bit. The remaining compressors use 40 BP iterations (per epoch in a $\gamma = 7$ binary search). The systematic precodes produce 100 additional output nodes of degree $d_o = 100$, with edges to input nodes such that each input node has approximately 1 edge incident on one of the extra output nodes.

The graphs are 10th degree polynomial approximations of the achieved compression ratios, including overhead information, for 5 repetitions of 200 input blocks of length 10000 bits over the entire binary entropy range.

The mean compression ratio and standard deviations for the entire binary entropy range of the varying entropy Bernoulli sources for the investigated amendments are shown in Table 2. The systematic precode improves the compression performance of the random puncturing algorithm although not as much with the equalized input degrees. The LT-CLID incremental redundancy algorithm performs significantly better than the random puncturing decremental redundancy algorithm although the LT-IDP compressor outperforms the LT-CLID algorithm with a reduced computational complexity.

Table 2: Comparative statistics of memoryless source compression performance

Algorithm	**Mean**	**Std. Dev.**
Systematic precode and random puncturing	0.907	0.164
Equalized input degrees and random puncturing	0.893	0.163
Systematic precode and equalized input degrees	0.896	0.176
Random puncturing	0.913	0.148
LT-CLID	0.857	0.189
LT-IDP	0.828	0.244

5.8 Noise robustness

The purpose of this compression approach is to achieve meaningful compression whilst being more robust against error propagation and transmission errors than conventional compressors. The ability of the proposed LT-IDP compressor to recover from data corruption during decompression is compared with that of LT-CLID.

An input source with 10 corruption repetitions, in a 10 times 10000 bit block, over the 0.141 to 0.722 binary entropy range was used. The compressed header was not corrupted, as it has a small size relative to the payload. The LT-IDP compressor has an LT-code with $Z = 1.1$, robust soliton output degree distribution with $c = 0.05$ and $\delta = 0.4$, Poisson input degree distribution with 40 BP iterations (per epoch in $\gamma = 7$ binary search).

The LT-CLID compressor is optimized for higher noise robustness with a doping delay of $D_d = 50$ iterations and a doping frequency of $D_f = 5$ iterations per bit, for 300 BP iterations (per epoch in $\gamma = 7$ binary search). Both compressors use belief propagation decoding with a maximum LLR of $L_\infty = 25.0$ and 8 a priori LLR bits.

The noise robustness on a flat fading AWGN (Additive White Gaussian Noise) channel for binary antipodal modulation is shown in Fig. 7.

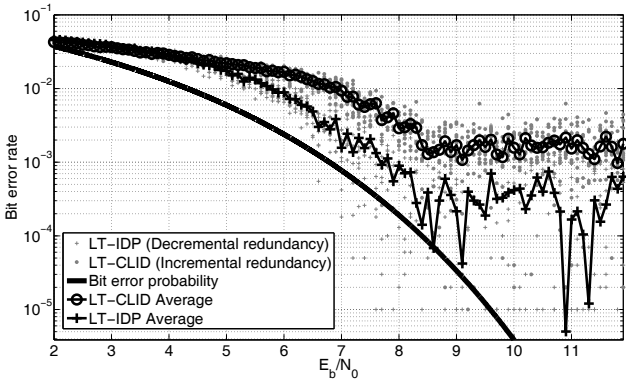

Figure 7: Noise robustness on a flat fading BI-AWGN channel

The graphs for LT-CLID and LT-IDP are the BER of the decompressed data and the line BER graph is the corruption ratio of the compressed data. The BSC (Binary Symmetric Channel) noise robustness is compared in Fig. 8 for the same compressor specifications as for AWGN.

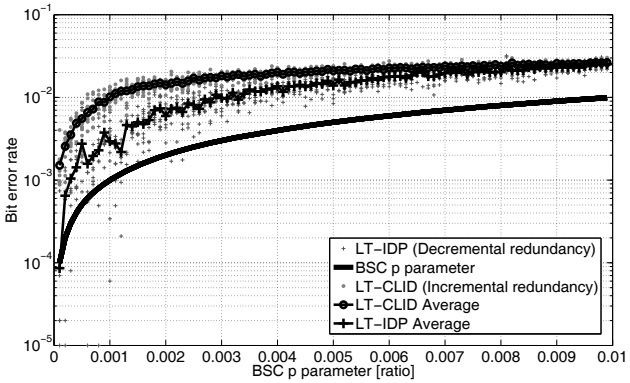

Figure 8: Noise robustness on the BSC channel

For the BSC channel, only LT-IDP is able to sustain the error fraction of the compressed data and for low p values it can reduce the error fraction in the decompressed data. The graph means, for BI-AWGN and BSC respectively, are 19.08% and 36.64% higher for LT-CLID. For both channels the higher noise-robustness of LT-IDP is apparent from Figs. 7 and 8.

6. UNIVERSAL COMPRESSION PERFORMANCE

The file compression of the Calgary and Canterbury corpora were measured with the Bzip2 (Huffman coding), Gzip (Lempel-Ziv coding), PAQ8L [24] (predictive arithmetic coding) and the Fountain code compressors.

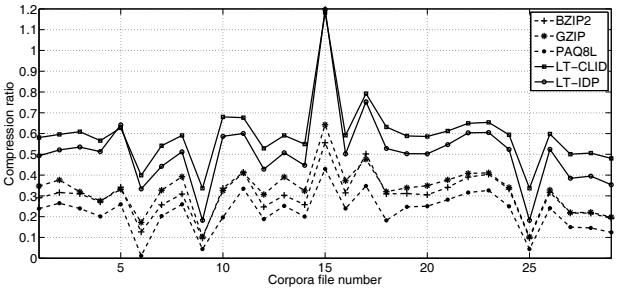

Figure 9: Compressor performance comparison for Calgary and Canterbury corpora

The BWT with symbol lengths determined by the SLE, the MTF and EOI transforms were used to pre-process the corpora files for the Fountain code compressors. The LT-CLID incremental redundancy and LT-IDP decremental redundancy algorithms were both separately used to compress the preprocessed files. For both of the redundancy-varying algorithms, blocks of 10000 bits were individually compressed.

The LT-CLID algorithm was optimized for best compression with a doping frequency of $D_f = 1$ iterations per bit and a doping delay of $D_d = 10$ iterations for 300 BP iterations (per epoch in a $\gamma = 7$ binary search). The LT-IDP algorithm used only 40 BP iterations (per epoch in a $\gamma = 7$ binary search). The results are compared for the Calgary and Canterbury corpora in Table 3.

Table 3: Compression performance statistics

	BZip2	**GZip**	**PAQ8L**	**LT-CLID**	**LT-IDP**
Mean	0.299	0.328	0.223	0.592	0.512
Std. Dev.	0.102	0.108	0.092	0.149	0.179

The LT-IDP compressor has a compression ratio for the Calgary and Canterbury corpora which is approximately 0.08 less than the ratio achieved by the LT-CLID compressor. There was only one benchmark file, namely *geo* in the Calgary corpus, which could not be compressed by either the LT-IDP or LT-CLID compressors. The preprocessor appreciably translated the original complex entropy, approximately represented by the performance of PAQ8L to the binary entropy of the preprocessor output. The Bzip2, Gzip and PAQ8L compressors performed significantly better than the Fountain code compressors as shown in Fig. 9.

7. CONCLUSION

A new method of improving the memory exploitation with the BWT, by determining a suitable symbol length with an SLE, was introduced. A new preprocessing strategy for translating complex source entropy to binary entropy was used to allow for universal compression with a non-universal binary entropy coder. This strategy included a new after-MTF stage transform, the EOI transform, which further reduced the binary entropy by more than 5%. A novel low-complexity implementation for the EOI transform is given, with a complexity which is 48 times less than that of a brute-force approach.

Compression with LT-codes with a decremental redundancy-varying algorithm, as proposed by us in [7], was revisited, and the binary compression performance of a basic random puncturing scheme was shown to be improved by 97.82%, with a low-complexity incremental degree puncturing distribution combined with a modified LT-code. This decremental redundancy algorithm (LT-IDP) has an increased compression performance of 20.88% over that of the existing LT-CLID method. Amendments to the application of the LT-code were discussed to improve the compression performance of a random puncturing.

The critical property of resistance to catastrophic error propagation during decompression with error-correction codes was considered. It was numerically shown that the LT-IDP algorithm provides greater noise robustness than the LT-CLID method for the BI-AWGN and BSC channels over the useful binary entropy range.

The combination of the preprocessing and the binary Fountain code entropy coder as a universal compressor removed 62.79% of the maximum redundancy according to the best possible compression estimate (PAQ8L), for the Calgary and Canterbury corpora.

8. REFERENCES

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

[2] J.S. Bhullar, P.S. Sandhu and M. Gupta, “A Huffman Codes Based Approach to Achieve Minimum Redundancy”, *International Conference on Computer Modeling and Simulation (ICCMS 2010)*, pp. 436-439, 22-24 Jan. 2010.

[3] J. Ziv, “The Universal LZ77 Compression Algorithm is Essentially Optimal for Individual Finite-Length N -Blocks”, *IEEE Trans. Information Theory*, vol. 55, no. 5, pp. 1941-1944, May 2009.

[4] M. Drmota, Y.A. Reznik and W. Szpankowski, “Tunstall Code, Khodak Variations, and Random Walks”, *IEEE Trans. Information Theory*, vol. 56, no. 6, pp. 2928-2937, June 2010.

[5] P. Mitran and J. Bajcsy, "Turbo source coding: A noise-robust approach to data compression", *Proceedings: DCC'02*, pp. 465-465, April 2002.

[6] J. Almeida and J. Barros, "Joint compression and data protection", *47th Annual Allerton Conference on Comunication, Control, and Computing, 2009*, pp. 835-842, Sept. 30-Oct. 2 2009.

[7] F.P.S. Luus and B.T. Maharaj, "Decremental redundancy compression with fountain codes", *IEEE Int. Conf. on Wireless and Mobile Computing (Wimob 2008)*, pp. 328-332, 12-14 Oct. 2008.

[8] I. Csiszar and J. Korner, Information Theory: Coding Theorems for Discrete Memoryless Systems, Academic, New York, 1981.

[9] P. E. Allard and A. W. Bridgewater, "A source encoding technique using algebraic codes", *Proceedings: 1972 Canadian Computer Conference*, pp. 201-213, June 1972.

[10] H. Ohnsorge, "Data compression system for the transmission of digitalized signals", *Conf. Rec. IEEE Int. Conf. on Communications*, vol. II, pp. 485-488, June 1973.

[11] T. Ancheta, "Syndrome source coding and its universal generalization", *IEEE Trans. Information Theory*, vol. 22, no. 4, pp. 432-436, July 1976.

[12] G. Caire, S. Shamai and S. Verdu, "A new data compression algorithm for sources with memory based on error correcting codes", *2003 IEEE Workshop on Information Theory*, pp. 291-295, March 30-April 4, 2003.

[13] G. Caire, S. Shamai and S. Verdu, "Lossless data compression with error correction codes", *2003 IEEE Int. Symp. on Information Theory*, p. 22, June 29-July 4 2003.

[14] G. Caire, S. Shamai and S. Verdu, "Universal data compression with LDPC codes", *Third International Symposium On Turbo Codes and Related Topics*, pp. 55-58, Brest, France, September 1-5, 2003.

[15] D. Matas, M. Lamarca and J. Garcia-Frias, "Non-linear graph-based codes for source coding", *IEEE Information Theory Workshop (ITW 2009)*, pp. 318-322, 11-16 Oct. 2009.

[16] A. Braunstein, R. Zecchina and F. Kayhan, "Efficient LDPC codes over GF(q) for lossy data compression", *IEEE International Symposium on Information Theory (ISIT 2009)*, pp. 1978-1982, June 28-July 3 2009.

[17] G. Caire, S. Shamai, A. Shokrollahi and S. Verdu, "Fountain Codes for Lossless Data Compression," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 2005. [Online]. Available: http://www.princeton.edu/~verdu/reprints/dimacs2005.pdf

[18] A. Shokrollahi, S. Lassen, and M. Luby. "Multi-stage code generator and decoder for communication systems". *U.S. Patent Application #20030058958*, 2003.

[19] N. Dutsch, "Code optimisation for lossless compression of binary memoryless sources based on FEC codes", *Euro. Trans. Telecomms*, 17, pp. 219-229, Wiley InterScience, 2006.

[20] R. Arnold and T. Bell, "A corpus for the evaluation of lossless compression algorithms," *Proceedings: 1997 Data Compression Conference*, pp. 201-210, 1997.

[21] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," *Digital Systems Res. Ctr.*, Palo Alto, CA, Tech. Rep. SRC 124, May 1994.

[22] P. Elias, "Interval and recency rank source coding: two on-line adaptive variable-length schemes," *IEEE Trans. Inform. Theory*, vol. 33, pp. 3-10, 1987.

[23] M.G. Luby, "LT codes," *Proceedings: 43rd IEEE Symp. Foundations of Computer Science*, pp. 271-280, 2002.

[24] M. Mahoney, S. Osnach and B. Pettis, PAQ8L software, 2007. [Online]. Available: http://cs.fit.edu/~mmahoney/compression/paq8l.zip