DIMACS Series in Discrete Mathematics
and Theoretical Computer Science

Noiseless Data Compression with Low-Density Parity-Check Codes

Giuseppe Caire, Shlomo Shamai, and Sergio Verdú

Abstract. This paper presents a new approach to universal noiseless compression based on error correcting codes. The scheme is based on the concatenation of the Burrows-Wheeler block sorting transform (BWT) with the syndrome former of a Low-Density Parity-Check (LDPC) code. The proposed scheme has linear encoding and decoding times and uses a new closed-loop iterative doping algorithm that works in conjunction with belief-propagation decoding. Unlike the leading data compression methods our method is resilient against errors, and lends itself to joint source-channel encoding/decoding; furthermore it offers very competitive data compression performance.

1. Introduction

Lossless data compression algorithms find numerous applications in information technology. To name some of the major ones:

- packing utilities (such as `gzip`) in operating systems such as Windows, Linux and Unix;
- modem standards such as V.32bis and V.42bis;
- fax standards such as CCITT;
- the back-end of lossy compression algorithms such as JPEG and MPEG;
- compression of headers of TCP/IP packets in wireless networks.

Indeed, the field of lossless data compression has achieved a state of maturity, with algorithms that admit fast (linear-complexity) implementations and achieve asymptotically the fundamental information theoretic limits.

However, emerging high-speed wireless data transmission systems send their payloads uncompressed. The main reasons for the failure of the state-of-the-art in wireless data networks to take into account source redundancy in those applications are:

- Lack of resilience of data compressors to transmission errors.

1991 *Mathematics Subject Classification.* Primary 68P30, 94A29; Secondary 94A45, 62B10.
Key words and phrases. Noiseless Data Compression, Universal algorithms, Error Correcting Codes, Source Coding, Sources with Memory, Block Sorting Transform.

- Packetized transmission/recording often means that the compression output length is only relevant up to an integer multiple of a given packet length.
- The regime for which the algorithms exhibit efficient performance requires much longer packet lengths than those typically used in modern high-speed wireless applications.

We propose a new approach to lossless data compression based on error correcting codes and the block-sorting transform. This approach turns out to be quite competitive with existing algorithms even in the purely lossless data compression setting, particularly for relatively short lengths (of the order of kilobits) of interest in packet wireless transmission.

Section 2 explains the role of linear channel codes in noiseless and almost-noiseless source coding. Section 3 shows how to use LDPC codes and the belief propagation algorithm for compressing and decompressing binary and nonbinary sources. Important gains in performance are possible by devising schemes that take into account that, unlike the noisy channel setting, here the encoder has access to all the information available to the decoder. Section 4 shows several ways to capitalize on this possibility and in particular it describes our closed-loop iterative doping scheme. The challenge of how to adapt the schemes for memoryless compression to general sources with memory is addressed in Section 6. This is solved by the dovetailing of the Block Sorting transform with LDPC encoding and the belief-propagation algorithm. Section 7 describes how to make the overall scheme universal and shows some experimental comparisons with state-of-the-art data compression algorithms.

2. Linear Channel Codes in Data Compression

The Shannon-MacMillan theorem [**Sha48, McM53**] states that for stationary ergodic sources and all $\delta > 0$ there exist fixed-length n-to-m compression codes of any rate m/n exceeding the entropy rate plus δ with vanishing block error probability as the blocklength goes to infinity. While almost-lossless fixed-length data compression occupies a prominent place in information theory it has had no impact in the development of data compression algorithms. However, *linear* error correction codes for additive-noise discrete memoryless channels can be used as the basis for almost-lossless data compressors for memoryless sources because of the following analogy.

A linear fixed-length data compressor is characterized by an $m \times n$ matrix $\mathbf{H}$ which simply maps $\mathbf{s}$, the source n-vector, to the compressed m-vector[1] $\mathbf{Hs}$. The maximum-likelihood decoder selects $g_{\mathbf{H}}(\mathbf{Hs})$, the most likely source vector $\mathbf{u}$ that satisfies $\mathbf{Hu} = \mathbf{Hs}$. Note that the encoder can recognize if the source vector will be decompressed successfully by simply checking whether $g_{\mathbf{H}}(\mathbf{Hs}) = \mathbf{s}$. If this is not the case, it can output an error message, or it can try a different (possibly with larger m) encoding matrix.

On the other hand, consider an additive-noise[2] discrete channel $\mathbf{y} = \mathbf{x} + \mathbf{u}$ driven by a linear channel code with parity check matrix $\mathbf{H}$. Since every valid codeword satisfies $\mathbf{Hx} = \mathbf{0}$, it is easy to see that the maximum-likelihood decoder

[1]The arithmetic is in the finite field on which the source is defined.

[2]The sum is that of the finite field on which the noise and channel inputs and outputs are defined.

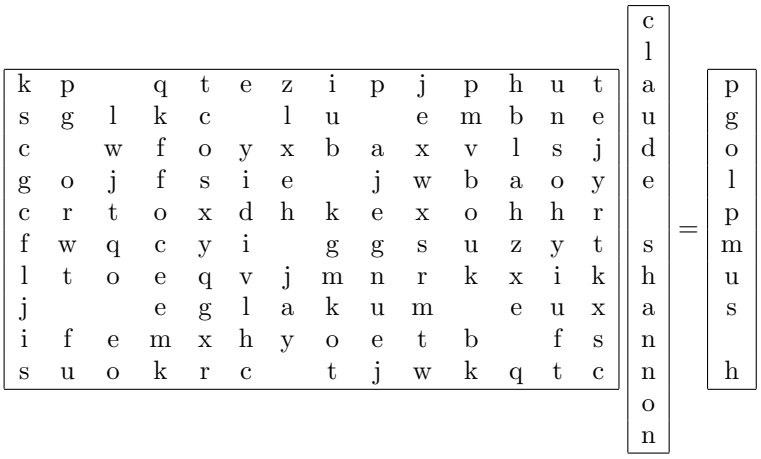

FIGURE 1. Example of linear data compression in GF(27)

selects the codeword $\mathbf{y} - g_{\mathbf{H}}(\mathbf{Hy})$, i. e. it selects the most likely noise realization among those that lead to the same *syndrome* $\mathbf{Hy}$. Thus, the problems of almost-noiseless fixed-length data compression and almost noiseless coding of an additive-noise discrete channel whose noise has the same statistics as the source are strongly related. Indeed since the maximum-likelihood channel decoder selects the wrong codeword if and only if $g_{\mathbf{H}}(\mathbf{Hu}) \neq \mathbf{u}$, the block error rate achieved by the corresponding maximum-likelihood decoders are identical. This strongly suggests using the parity-check matrix of a channel code as the source encoding matrix. One of the (neglected) consequences of this powerful equivalence relates to the construction of optimum n-to-m source codes, which are well-known to be those that assign a unique codeword to each of the n-sequences with the largest probabilities. For memoryless sources, this can be accomplished for certain (m, n) by using the parity-check matrix of *perfect* codes[3]. To see this (in the binary case) note that using the parity-check matrix of a perfect linear code of radius e, all the sourcewords with weight $0, 1, \ldots, e$ are mapped to distinct codewords. Otherwise, when the code is used for the binary channel there would be different error patterns of weight less than or equal e leading to the same syndrome. For example, in the binary case, we can obtain optimum 7-to-3 and 23-to-11 source codes by using the Hamming and Golay codes respectively. Figure 2 compares the probability that an optimal (Hamming) 7-to-3 source code is successful for a Bernoulli source with bias p with the probability that a Huffman (minimum average length) code operating on 7-tuples outputs a string with no more than 3 bits.

Unfortunately, the foregoing analogy between linear source codes and linear parity-check channel codes seems to have been all but neglected in the literature and we can find few instances in which error correcting codes have been used as data compressors. Dealing with source sequences known to have a limited Hamming weight, Weiss [**Wei62**] noted that since the 2^{n-k} cosets of an e-correcting (n, k) binary linear code can be represented by their unique codewords having weight less than or equal to e, the binary input n-string can be encoded with $n - k$ bits

[3]A perfect code is one where Hamming spheres of a fixed radius around the codewords fill the complete space of n-tuples without overlap (e.g. [**McE02**]) .

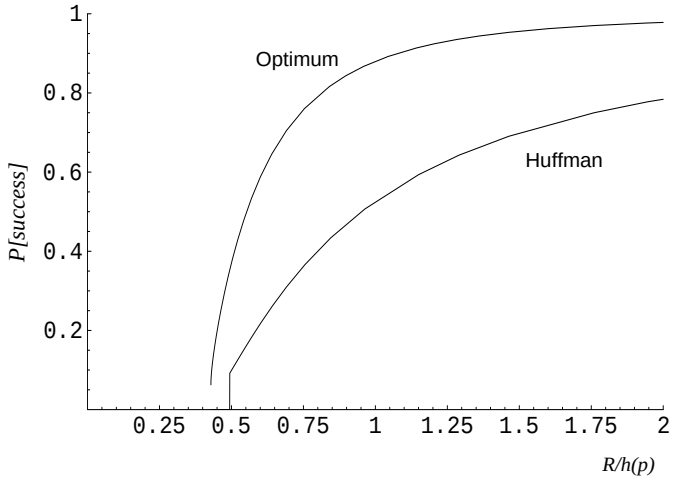

FIGURE 2. 7-to-3 encoding of a biased coin.

that label the coset it leads. The decompressor is simply the mapping that identifies each coset with its leader. The fact that this is equivalent to computing the syndrome that would result if the source sequence were the channel output was explicitly recognized by Allard and Bridgewater [**AB72**]. Both [**AB72**] and Fung et al [**FTS73**] remove Weiss' bounded weight restriction and come up with variable-to-fixed-length schemes where zeros are appended prior to syndrome formation so that the resulting syndrome is correctable. Also in the setting of binary memoryless sources, Ancheta [**Anc76**] considered fixed-to-fixed linear source codes based on syndrome formation of a few BCH codes. Since the only sources the approach in [**Wei62, AB72, FTS73, Anc76**] was designed to handle were memoryless sources (essentially biased coins) and since the practical error correcting codes known at that time did not offer rates close to channel capacity, that line of research rapidly came to a close with the advent of Lempel-Ziv coding. Also limited to memoryless binary sources is another, more recent, approach [**GFZ02**] in which the source is encoded by a Turbo code whose systematic outputs as well as a portion of the non-systematic bits are discarded; and decompression is carried out by a Turbo decoder that uses knowledge of the source bias. Germane to this line of research is the use of Turbo and LDPC codes [**BM01, GFZ01, Mur02, LXG02**] for Slepian-Wolf coding of correlated memoryless sources as specific embodiments of the approach proposed in [**Wyn74**].

The restriction for a fixed-length source code to be linear is known not to incur any loss of asymptotic optimality for memoryless sources. This can be shown through a standard random coding argument [**CK81**], which is essentially the same used in [**Eli55**] for the optimality of random channel codes. The idea is that, using random codebooks, the intersection of the coset represented by the syndrome formed with the source sequence and the set of typical sequences has one and only one element with high probability as the blocklength goes to infinity. We have extended this result (without recourse to typicality) to completely general sources (with memory and nonstationarity):

THEOREM 2.1. [**CSV**] *If the source has sup-entropy rate*[4] $\bar{H}$, $\frac{m}{n} > \bar{H} + \delta$, *the encoding matrix entries are independent, equiprobable on the alphabet, then the average (over the ensemble of matrices and source realizations) block error probability* $\to 0$ *as* $n \to \infty$

For general linear codes, syndrome forming (source encoding) has quadratic complexity in the blocklength and maximum-likelihood decoding has exponential complexity in the blocklength. These complexities compare unfavorably with the linear encoding/decoding complexities achieved by Lempel-Ziv coding. To address that shortcoming we consider a specific class of error-correction codes in the next section.

3. LDPC Data Compression for Memoryless Sources

By definition, the parity-check matrix $\mathbf{H}$ of a low-density parity-check code is a sparse matrix with a number of nonzero entries growing linearly with the blocklength, and, thus, the linear source compressor that uses $\mathbf{H}$ has linear complexity in the blocklength. Note that this is in contrast to the encoding of LDPC codes for channel encoding as the multiplication by the generator matrix is far less straightforward, requiring the application of the technique in [**RU01b**] to achieve nearly linear complexity.

The sparseness of $\mathbf{H}$ does not imply that a polynomially-complex maximum-likelihood decoder exists. For LDPCs the suboptimal iterative technique known as *Belief-Propagation* (BP) decoding has proved to yield very good results on a variety of memoryless channels. The BP decoders used in channel decoding of binary symmetric channels [**Gal63, RU01a**] operate with the channel outputs rather than with the syndrome vector. However, counterparts to those algorithms that give an approximation to the syndrome-decoding function $g_{\mathbf{H}}$ can easily be derived. In these algorithms, the compressed data is not present at the variable nodes (which represent the uncompressed bits) but at the check nodes, since in the source-coding case, each parity-check equation has a value given by the compressed data. For simplicity and concreteness we specify the algorithm in the binary memoryless case with nonstationary probabilities (which will be relevant in Section 6). Fix the realization of the input to the decoder, $\mathbf{z}$. The set of checknodes in which the bitnode $k \in \{1, \dots n\}$ participates is denoted by $\mathcal{A}_k \subset \{1, \dots m\}$, and the set of bitnodes which are connected to checknode $j \in \{1, \dots m\}$ is denoted by $\mathcal{B}_j \subset \{1, \dots n\}$. With the a priori probability that the kth source symbol equal to 1 denoted by p_k, define the a priori source log-ratios

$$\mathcal{L}_k = \log \frac{1 - p_k}{p_k}, \quad k \in \{1, \dots n\}, \tag{3.1}$$

and the isomorphism $\phi : GF(2) \mapsto \{-1, +1\}$, where $\phi(0) = +1$ and $\phi(1) = -1$. For each iteration $t = 1, 2, \dots$, the algorithm computes the value of the bitnodes

$$\widehat{x}_k = \phi^{-1}\left(\text{sign}\left\{\mathcal{L}_k + \sum_{j \in \mathcal{A}_k} \mu_{j \to k}^{(t)}\right\}\right)$$

by updating the messages sent by the checknodes to their neighboring bitnodes and by the bitnodes to their neighboring checknodes, denoted respectively by $\mu_{j \to k}^{(t)}$ and

[4]See [**HV93**] for the definition of sup-entropy rate.

by $\nu_{k\to j}^{(t)}$, according to the message-passing rules

$$\nu_{k\to j}^{(t)} = \mathcal{L}_k + \sum_{j'\in\mathcal{A}_k-\{j\}} \mu_{j'\to k}^{(t-1)} \tag{3.2}$$

and

$$\mu_{j\to k}^{(t)} = 2\phi(z_j)\tanh^{-1}\left(\prod_{k'\in\mathcal{B}_j-\{k\}} \tanh\left(\frac{\nu_{k'\to j}^{(t)}}{2}\right)\right) \tag{3.3}$$

with the initialization $\mu_{j\to k}^{(0)} = 0$ for all $j \in \{1,\ldots,m\}$. At each iteration, the parity-check equations

$$\sum_{k\in\mathcal{B}_j} \widehat{x}_k = z_k, \quad j = 1,\ldots,m$$

are evaluated. If they are all satisfied, the BP algorithm stops. If after a maximum number of allowed iterations some parity-check equations are not satisfied, a block-error is declared.

In the q-ary case, the BP algorithm is more cumbersome since the distributions are no longer succinctly represented by a scalar. However, a linear-time BP algorithm akin to the one in [**DM98**] can be given [**CSV**] Using the FFT in GF(q), the computational complexity per iteration is $O(E\mathsf{q}\log\mathsf{q})$ if E is equal to the number of nonzero entries in $\mathbf{H}$.

We have shown [**CSV**] that the same pointwise correspondence between channel decoding and source decoding stated for the ML decoder in Section 2 holds also for the BP decoder: the sets of source sequences and noise sequences that lead to errors of the respective BP decoders are identical. This equivalence has important consequences in the design of the LDPC matrix for data compression. Consider an arbitrary LDPC ensemble with BSC-threshold $p^\star$. In the standard terminology, this means that as the blocklength goes to infinity, the bit error rate achieved by BP averaged over the ensemble vanishes for every BSC with crossover probability less than $p^\star$, but not higher. Similarly, we can define the Bernoulli-source threshold of an arbitrary LDPC ensemble for fixed-length data compression, as the supremum of the source biases for which the bit error rate achieved by BP averaged over the ensemble vanishes as the blocklength goes to infinity. Because of the equivalence in performance of the corresponding BP decoders, for a given ensemble the BSC-threshold is equal to the Bernoulli-source threshold. It follows that optimal LDPC ensembles for the BSC yield immediately optimal LDPC ensembles for source coding of biased coins.

To end this section, we present an alternative to the foregoing approach to fixed-length data compression based on error correcting codes for an additive-noise discrete memoryless channel. This alternative approach uses erasure channel coding/decoding designed for the conventional stationary erasure channel. At the compressor, first we apply the n-binary string produced by the source to a rate-R erasure channel encoder. Then, we discard the systematic part and the remaining $n(\frac{1}{R}-1)$ bits are passed through an interleaver; the compressed m-binary string is given by the uppermost $1-p$ fraction of the interleaver outputs. The other interleaver outputs are also discarded (Figure 3). Thus,

$$\frac{m}{n} = (1-p)(\frac{1}{R}-1).$$

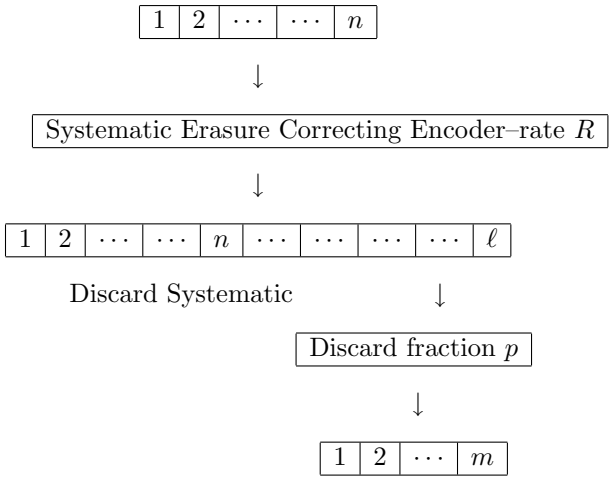

FIGURE 3. Data compression with erasure correcting codes.

There is some freedom in the choice of p and R, respecting the constraint:

$$(1-p)(\frac{1}{R}-1) \approx H + \delta.$$

or equivalently,

$$R \approx \frac{1-p}{1-p+H+\delta}.$$

The data decompressor first appends to the compressed sequence a string of $m\frac{p}{1-p}$ erasures,then it deinterleaves (using the inverse interleaver used at the compressor), it appends n erasures (the missing systematic part), and it feeds the whole string of m bits and $n((\frac{1}{R}-1)p+1)$ erasures (a total of n/R symbols) to the erasure decoder which takes into account the a priori probabilities of the source as follows: Now there are n/R variable nodes, which from top to bottom consist of

- n erasures corresponding to bits whose a priori probabilities are equal to $P_{s_1}(1), \ldots P_{s_n}(1)$;
- $mp/(1-p)$ erasures corresponding to bits which are approximated as being equiprobable
- m nonerased bits.

The iterations proceed using the BP algorithm described above once we do the following preprocessing. For each of the $(1/R-1)n$ check nodes, we assign a value to check node i given by the sum of all the information bits contained in the nonerased subset of variable nodes connected to that check node. Now, discard all nonerased variable nodes and their connections. All the remaining variable nodes correspond to erasures; n of them have the a priori probabilities of the source and their loglikelihood updates proceed exactly as in (3.2). For the $mp/(1-p)$ remaining erasure variable nodes, (3.2) also applies but with the first term equal to 0 (corresponding to 1/2 prior probabilities). The methods in Sections 4,6,7 are described in the context of the error-correcting approach but apply verbatim to the erasure-correcting approach.

4. Decoding at the Encoder

In contrast to channel decoding, in noiseless data compression, the source encoder has the same information available to the decoder. Thus, not only it can check whether the original sourceword will be correctly decompressed but it has the luxury of running a mirror image of the decoder iterations.

There are several ways in which this capability can be taken advantage of in order to counter the inherent inefficiencies incurred by the use of LDPC codes and the suboptimal BP decoder.

First note that the "almost noiseless" fixed-to-fixed compression algorithm can be easily adapted to zero-error variable-to-fixed or fixed-to-variable operation by tuning the input/output length in the compression to guarantee decoding success. For example, it is possible to successively generate parity-check equations until successful decoding is achieved. To that end, the use of raptor codes [**Sho03**] is more natural than LDPC codes as shown in [**CSSV**].

In general, the encoder can try several parity-check matrices (with equal or different rates) from a library known at the decoder until success is achieved. A header identifying the actual matrix sent is part of the compressed sequence.

Symbol Doping, i.e. informing the decoder about sure bits, which are treated by the decoder as perfectly known (i.e., having infinite reliability) is another strategy that accelerates convergence and improves the block error probability. It can be done in open-loop mode [5] where the location of the doped symbols is pre-agreed or in closed-loop mode depending on the evolution of the algorithm. However, it would be very costly to need to identify the location of each bit to be doped. A much better alternative is our *Closed-Loop Iterative Doping Algorithm* which dopes the bit that most needs it, i.e. the one with the lowest reliability. The algorithm generates one doped bit per iteration as follows.

1) Initialization: $\mu_{j\to k}^{(0)} = 0$ for all $j \in \{1, \ldots, m\}$.

2) Repeat the following steps for $t = 1, 2, \ldots$ (iteration count) until successful decoding is reached:

- For all bitnodes $k = 1, \ldots, n$ compute
$$\nu_k^{(t)} = \mathcal{L}_k + \sum_{j \in \mathcal{A}_k} \mu_{j\to k}^{(t-1)}$$
and find the node that achieves the lowest reliability:
$$\widehat{k} = \arg \min_{k=1,\ldots,n} \{|\nu_k^{(t)}|\},$$
- Least-reliable symbol doping: Feed the symbol $x_{\widehat{k}}$ directly to the decoder and let
$$\mathcal{L}_{\widehat{k}} = \begin{cases} +\infty & \text{if } x_{\widehat{k}} = 0 \\ -\infty & \text{if } x_{\widehat{k}} = 1 \end{cases}$$
- Bitnode update (3.2) and Checknode update (3.3).

Some qualitative properties of the Iterative Doping Algorithm are:

(1) Because of reliability sorting, the position of doped symbols need not be explicitly communicated to the decoder. Hence, the cost of doping is the (random) number of iterations necessary to achieve successful decoding.
(2) The algorithm never dopes twice the same symbol.

[5] Traditional doping (e.g. [**tB00**]) follows that paradigm.

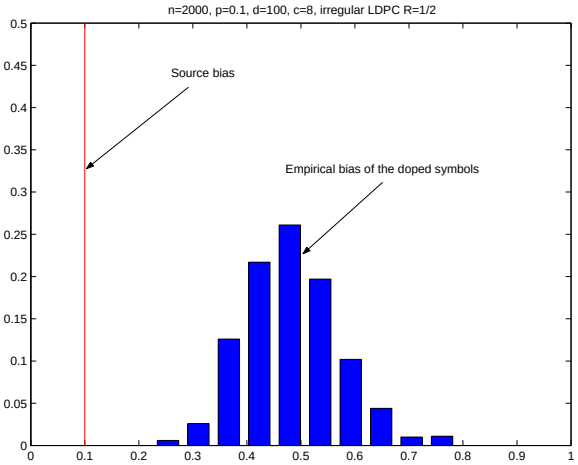

FIGURE 4. Empirical Distribution of the bits generated by the Closed-Loop Iterative Doping Algorithm.

(3) Doping the symbol with least reliability is an efficient strategy. In practice this is done only if the reliability is below a certain threshold, in which case the doped bits (conditioned on the knowledge accumulated up until their reception by the decompressor) are approximately fair coin flips. Figure 4 contrasts the distribution of the source (a very biased coin) with the empirical distribution of the bits generated by the Closed-Loop Iterative Doping Algorithm, which is very close to fair-coin flips. In that experiment, the empirical entropy of the doped bits is above 0.95 bits with probability around 0.92. While at the very beginning the lowest-reliability bit has the same bias as the source, after the first iterations it quickly drops below the threshold. Figure 5 illustrates the typical evolution of the average and minimum reliability of the source bits as a function of the BP iterations.

The closed-loop iterative doping algorithm can either be used in fixed-length mode with a prefixed number of doping bits or in zero-error variable-length mode, where the number of doped bits is dictated by the source realization. The mean and variance of the number of doped symbols can be reduced by using a library of several parity-check matrices, [6] by applying the above algorithm to each matrix, choosing the one that achieves successful decoding with the smallest length, and indicating the label of the best matrix to the decoder. Note that doping need not take place at every iteration. In fact, a tradeoff exists between decompression time and encoding efficiency which can be controlled by the doping schedule. Furthermore the doping schedule can also be "closed-loop", controlled by the instantaneous values of the reliabilities computed by the BP algorithm. Although explained in conjunction with the error-correcting approach, the iterative doping algorithm can also be applied to the erasure-correcting approach to data compression explained in Section 3. Figure

[6]Even after the BP algorithm has converged, there exists the (extremely unlikely) possibility that the chosen codeword is erroneous. In that event the compressor can choose another parity-check matrix (e.g. the identity) to ensure strictly zero-error decompression.

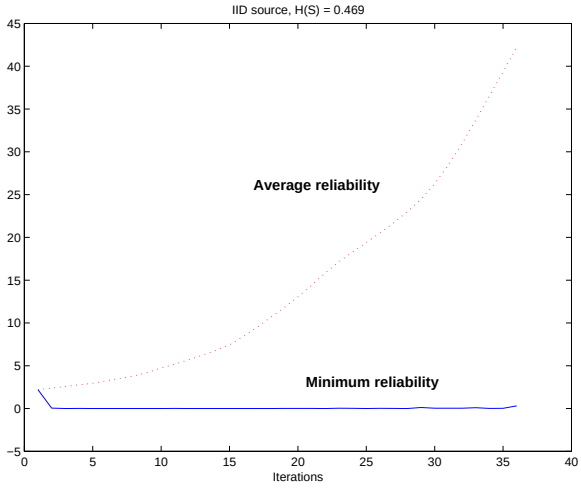

FIGURE 5. Typical evolution of the minimum and average reliability of the BP decoder.

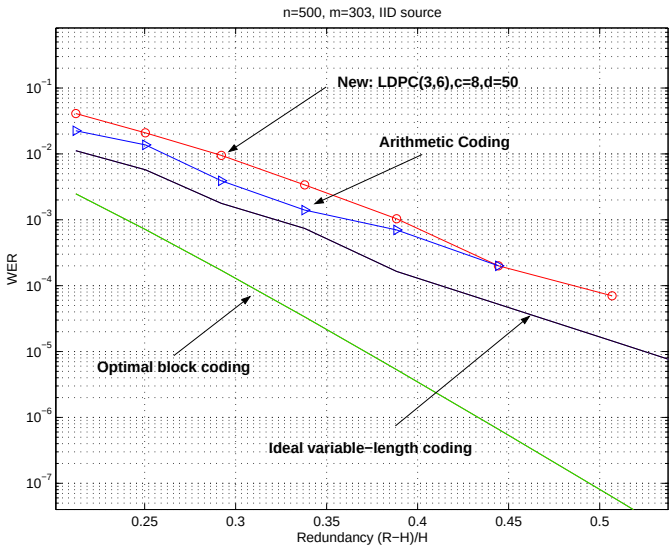

FIGURE 6. Block error rate of 500-to-303 source codes for a biased coin.

6 shows the distribution (complementary cumulative distribution function) of the achieved compression for several schemes operating on a biased coin for biases p ranging from 0.11 to 0.08. Since the code rate is $R = 303/500$, this translates into normalized redundancies $(R - h(p))/h(p)$ ranging from 0.21 to 0.51, where $h(p)$ is the binary entropy function. Note that by means of the distribution of redundancies it is possible to compare fixed-length and variable-length schemes on an equal footing: for a variable-length scheme the "WER" in Figure 6 is simply the probability of the set of source realizations that cannot be encoded with the corresponding redundancy.

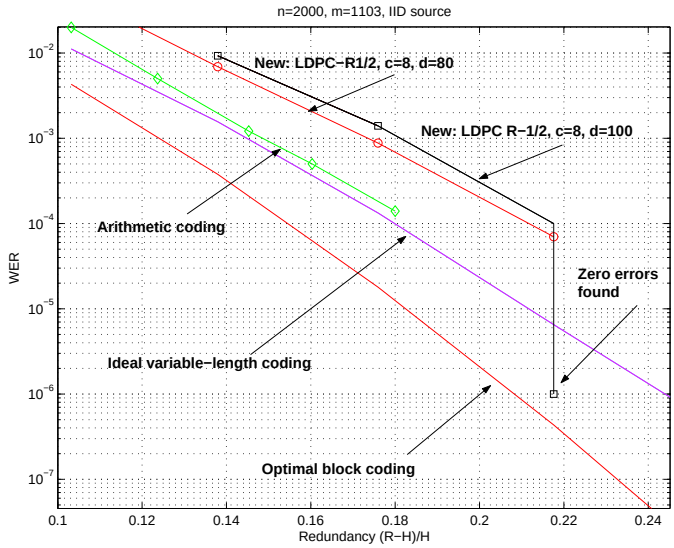

FIGURE 7. Block error rate of 2000-to-1103 source codes for a biased coin.

The schemes shown in Figure 6 are:

- *Optimal block coding*: optimal code that assigns a distinct codeword to the most probable 2^{303} source realizations.
- *"Ideal" variable-length coding* (or Information Spectrum (cf. [**HV93**]): where the 'length' of the codeword assigned to $x_1, \ldots x_n$ is equal to

$$-\log_2 P_{X_1,\ldots X_n}(x_1, \ldots, x_n).$$

- *Arithmetic Coding:* Nonuniversal, source exactly known.
- *New code:* with a library of 8 (3,6) regular LDPCs, and 50 closed-loop iterative doped bits.

Figure 7 shows the results of an analogous experiment with a longer blocklength and a library of 8 rate-$\frac{1}{2}$ irregular LDPCs drawn at random from the ensemble designed for the BSC in [**Urb02**]. We show the effect of varying the number of bits allocated to doping from 80 to 100. At the point marked at 10^{-6} no errors were detected in 10,000 trials.

The main benefit accrued by the new scheme over arithmetic coding is much higher resilience to errors at the cost of a small decrease in performance efficiency. Figure 8 illustrates the very different resilience of arithmetic coding and the new scheme against channel erasures. It should be noted that in neither case we take any countermeasures against the channel erasures. In [**CSV04**], we report on source/channel schemes derived from the basic scheme of this paper which exhibit much more robustness against channel noise. A source block of 3000 biased coin flips is compressed into a block of 1500 syndrome bits, 200 doping bits and 3 bits identifying the LDPC. The syndrome bits are decompressed by the conventional BP algorithm with the only difference that a subset of the check nodes are now missing due to the channel erasures. As we vary the coin bias from 0.1 to 0.08, we observe an 'error floor' in Figure 8 which is dominated by the fact that the 203 doping and code library bits are sent completely unprotected. The abscissa in

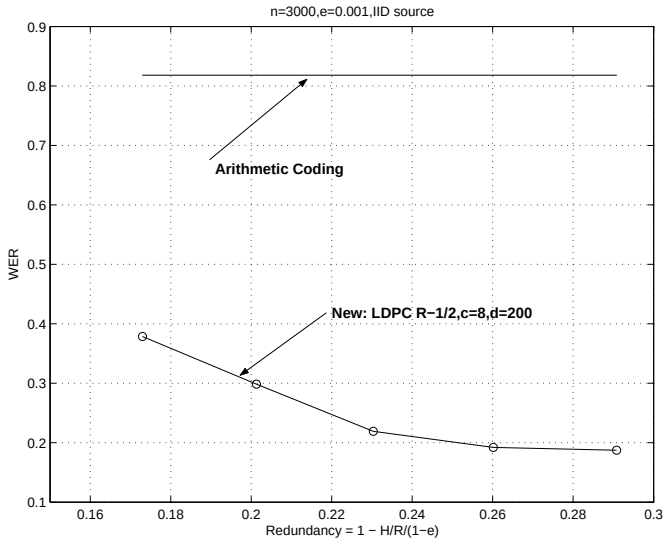

FIGURE 8. Block error rate of a 3000-to-1703 source code and arithmetic coding for a biased coin and a channel with erasure probability equal to $e = 0.001$

Figure 8 is a normalized measure of the distance from the fundamental limit that would be achievable for asymptotically long blocklengths, namely $\frac{h(p)}{1-e}$ where e is the channel erasure probability.

5. Non-binary Sources: Multilevel Compression Codes

In this section we propose an approach based on multilevel coding and multistage successive decoding to handle q-ary sources, where $\mathsf{q} = 2^L$. In the realm of channel coding, multilevel coding and successive decoding is a well-known technique to construct efficient codes for high-order modulations, see for example [**WFH99**] and more recently [**CS02**] with application to LDPC-coded modulation.

Although as we mentioned in previous sections, both the encoders and decoders can be generalized to the nonbinary case, there are at least two obstacles for the practical application of our method to general q-ary sources:

(1) The design of q-ary LDPC ensembles for an arbitrary q-ary additive noise channel $\mathbf{y} = \mathbf{x} + \mathbf{z}$, where addition is in GF(q), or in the ring $\mathbb{Z}_{\mathsf{q}}$, and $\mathbf{z}$ obeys an arbitrary q distribution is still not mature even in the simplest case where $\mathbf{z}$ is i.i.d. See [**DM98, SF02**].
(2) The FFT-based q-ary BP decoder mentioned in Section 3 has linear complexity in the source blocklength, but grows as $O(\mathsf{q} \log \mathsf{q})$ with the alphabet size.

By restricting the source alphabet to have cardinality equal to a power of two (e.g., $\mathsf{q} = 2^7$ in ASCII), we are able to use efficiently standard LDPC binary codes and achieve an overall complexity $O(n \log \mathsf{q})$.

For the sake of simplicity, we outline the proposed multilevel approach to data compression for an i.i.d. source in the non-universal setting (i.e., assuming that the

source statistics is perfectly known to both the encoder and decoder). The methods in Sections 4,6,7 apply to this multilevel approach also.

Let X be an i.i.d. source $\sim P_X$, defined over an alphabet $\mathcal{X}$ of cardinality 2^L. Define an arbitrary one-to-one binary labeling of the source alphabet $\mu : \mathcal{X} \to \mathrm{GF}(2)^L$, such that $\mu(x) = (b_1, \ldots, b_L)$ is the binary label corresponding to x, for all $x \in \mathcal{X}$. The mapping μ and the source probability assignment P_X induce a probability assignment $P_{B_1,\ldots,B_L}(b_1, \ldots, b_L)$ over $\mathrm{GF}(2^L)$ such that $P_{B_1,\ldots,B_L}(\mu(x)) = P_X(x)$. Without loss of generality, consider the binary random variables B_ℓ in the natural order $\ell = 1, \ldots, L$. We have

$$\begin{aligned} H(X) &= H(B_1, \ldots, B_L) \\ &= \sum_{\ell=1}^{L} H(B_\ell | B_1, \ldots, B_{\ell-1}) \end{aligned} \tag{5.1}$$

From the above decomposition, we see that each *level* ℓ can be thought as a binary piecewise i.i.d. source having up to $2^{\ell-1}$ segments and entropy $H(B_\ell|B_1, \ldots, B_{\ell-1})$. Each segment corresponds to a different value of the conditioning variables $(B_1, \ldots, B_{\ell-1})$. The conditional probability of $B_\ell = 1$ given $(B_1, \ldots, B_{\ell-1}) = (b_1, \ldots, b_{\ell-1})$ is given by

$$p_\ell(b_1, \ldots, b_{\ell-1}) \triangleq \frac{\sum_{(b_{\ell+1},\ldots,b_L)\in\{0,1\}^{L-\ell}} P_{B_1,\ldots,B_L}(b_1, \ldots, b_{\ell-1}, 1, b_{\ell+1}, \ldots, b_L)}{\sum_{(b_\ell,\ldots,b_L)\in\{0,1\}^{L-\ell+1}} P_{B_1,\ldots,B_L}(b_1, \ldots, b_L)} \tag{5.2}$$

Therefore, the foregoing discussion on the compression of binary piecewise i.i.d. sources can be applied to each level ℓ. The only difference is that now the segments for the binary source at level ℓ are defined by the realizations of the previous levels $1, \ldots, \ell-1$, i.e., encoding and decoding must be performed in a successive fashion.

The multilevel coding algorithm is briefly summarized as follows: we assume that the source distribution P_X is known, therefore, also the conditional marginal probabilities $p_\ell(b_1, \ldots, b_{\ell-1})$ defined in (5.2) and the corresponding ℓ-level entropies $H_\ell \triangleq H(B_\ell|B_1, \ldots, B_{\ell-1})$ are known. Then, we select L LDPC ensembles such that the ℓ-th ensemble has rate R_ℓ larger than, but close to, H_ℓ. In practice, we can optimize each ℓ-th LDPC ensemble for a BSC with parameter $p_\ell = h^{-1}(H_\ell)$. For each ℓ-th ensemble, we construct c parity-check matrices $\{\mathbf{H}_{\ell,i} : i = 1, \ldots, c\}$.

Let $\mathbf{s}$ denote the q-ary source sequence of length n to be compressed and, for each level ℓ, let $\mathbf{b}_\ell = (b_{\ell,1}, \ldots, b_{\ell,n})$ denote the binary sequence obtained by projecting the binary representation of $\mathbf{s}$ onto its ℓ-level binary component. For levels $\ell = 1, 2, \ldots, L$, we produce the binary encoded sequences $\mathbf{z}_\ell$ successively, by applying the LDPC compression method described before, based on the library of parity-check matrices $\{\mathbf{H}_{\ell,i} : i = 1, \ldots, c\}$ and the CLID algorithm. As anticipated above, the ℓ-th level encoder/decoder depends on the realization of the binary sequences at levels $1, \ldots, \ell-1$ since it makes use of the conditional statistics of $\mathbf{b}_\ell$ given $\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1}$. This conditioning is incorporated into the BP decoder by using the sequence of conditional probability log-ratios

$$\mathcal{L}_k = \log \frac{1 - p_\ell(b_{1,k}, \ldots, b_{\ell-1,k})}{p_\ell(b_{1,k}, \ldots, b_{\ell-1,k})}$$

for $k = 1, \ldots, n$. Notice that at each level ℓ the symbols $b_{1,k}, \ldots, b_{\ell-1,k}$ are known at the decoder due to successive decoding. In the lossless setting there is no error propagation between the levels.

In the realm of channel coding, multilevel coding and successive decoding is a well-known technique to construct efficient codes for high-order modulations, see for example [**WFH99**] and more recently [**CS02**] with application to LDPC-coded modulation. In the realm of lossy image coding, the multilevel coding idea is reminiscent of the so-called "bit-plane coding" (e.g., see [**TM**]), where a sequence of L-bit integer samples output by some transform coder is partitioned into L binary sequences called "bit-planes". Each ℓ-th sequence is formed by the ℓ-th most significant bits of the original samples, and it is separately compressed by adaptive arithmetic coding.

The resulting encoded binary sequence corresponding to $\mathbf{s}$ is given by the concatenation of all syndromes $\mathbf{z}_\ell$ and corresponding doping symbols, for $\ell = 1, \ldots, L$. Successive encoding and decoding complexity scales linearly with the number of levels, i.e., logarithmically with the alphabet size, as anticipated above.

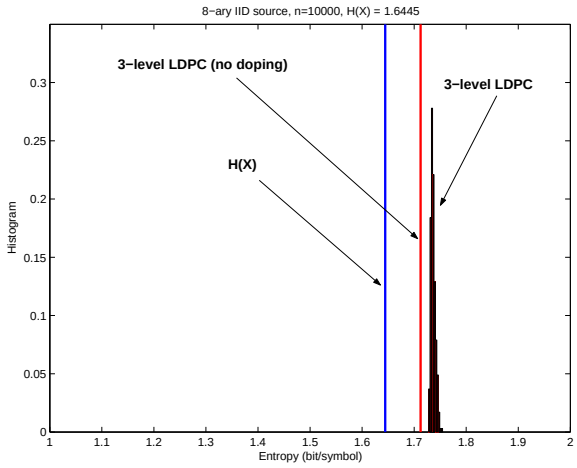

FIGURE 9. Histogram of compression rates with multilevel scheme.

Figure 9 shows the compression rate histogram achieved by the proposed scheme for a 8-ary source ($L = 3$ levels). The source has entropy $H(X) = 1.6445$, and probability distribution

$$P_X = [0.5666, 0.2833, 0.0850, 0.0028, 0.0028, 0.0283, 0.0028, 0.0283]$$

Encoding the 8 letters by the binary representation $000, 001, \ldots, 111$ (level 1 is the least-significant bit), we obtain the level entropies $H_1 = 0.9274, H_2 = 0.5233$ and $H_3 = 0.1938$. We chose LDPC ensembles with slightly conservative rates equal to $R_1 = 0.95, R_2 = 0.55$ and $R_3 = 0.22$, and for each ensemble we used 8 independently generated parity-check matrices. Figure 9 shows the distribution of the achieved coding rates. The vertical line labelled "3-level LDPC (no doping)" corresponds to the compression rate of the above scheme due exclusively to the LDPC encoding in the absence of doping (and thus achieving nonzero block error rate), with the same LDPC codes used in the histogram shown with doping. Notice that the largest

contribution to redundancy is due to the conservative choice of the LDPC coding rates, and the Closed Loop Iterative Doping adds a small variable fraction to the overall redundancy. Also, notice that thanks to the iterative doping each level achieves zero decoding error probability. Hence, no catastrophic error propagation between the levels occurs.

6. Sources with Memory

In Sections 3, 4 and 5 we have limited our discussion to encoding memoryless nonstationary sources. However we know from Theorem 2.1 that linear codes can be used to approach the entropy rate of the source even if it has memory. However, the direct application of a syndrome former to a source with memory faces a serious hurdle: it is very cumbersome to design an optimum or near-optimum decoder that takes the source statistics into account. As we saw, it is easy to incorporate the knowledge about the source marginals in the BP decoding algorithm. However, for sources with memory the marginals alone do not suffice for efficient data compression. While the incorporation of Markov structure is in principle possible at the BP decoder by enlarging the Tanner graph, the complexity grows very rapidly with the source memory. In addition it would be futile to search for encoders as a function of the memory structure of the source. We next describe a design approach that enables the use of the BP (marginal based) decoding algorithm in Section 3 while taking full advantage of the source memory.

We propose to use a one-to-one transformation, called the block-sorting transform or Burrows-Wheeler transform (BWT) [**BW94**] which performs the following operation: after adding a special End-of-file symbol, it generates all cyclic shifts of the given data string and sorts them lexicographically. The last column of the resulting matrix is the BWT output from which the original data string can be recovered.

Note that the BWT performs no compression. Fashionable BWT-based universal data compression algorithms (e.g. *bzip*) have been proposed which are quite competitive with the Lempel-Ziv algorithm. To understand how this is accomplished, it is best to consider the statistical properties of the output of the BWT. It is shown in [**EVKV02**] that the output of the BWT (as the blocklength grows) is asymptotically piecewise i.i.d. For stationary ergodic tree sources the length, location, and distribution of the i.i.d. segments depend on the statistics of the source. The universal BWT-based methods for data compression all hinge on the idea of compression for a memoryless source with an adaptive procedure which learns implicitly the local distribution of the piecewise i.i.d. segments, while forgetting the effect of distant symbols.

Our approach is to let the BWT be the front-end. Then we apply the output of the BWT to the LDPC parity-check matrix, as explained in Section 3 for memoryless nonstationary sources.[7] The decompressor consists of the BP decoder, making use of the prior probabilities p_i. The locations of the transitions between the segments exhibit some deviation from there expected values which can be explicitly communicated to the decoder. Once the string has been decompressed we apply the inverse BWT to recover the original string. Figure 10 shows a block diagram of the nonuniversal version of the compression/decompression scheme.

[7]It is sometimes advantageous to set a threshold H_{th} reasonably close to 1 bit/symbol and feed directly to the output the symbols on segments whose entropy exceeds H_{th}.

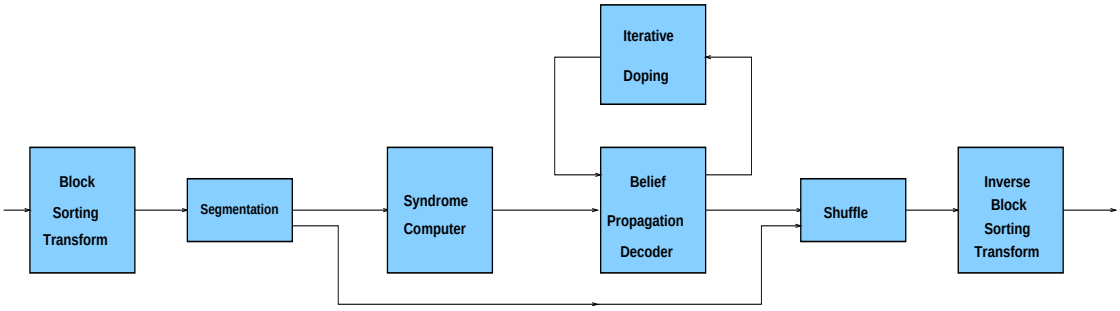

FIGURE 10. Compression/Decompression Scheme: Nonuniversal Version

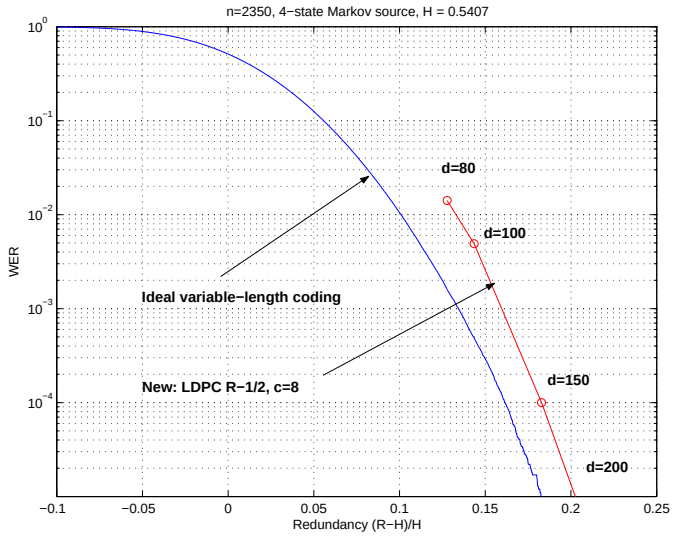

FIGURE 11. Block error rate of a 2350-to-$(1350+d)$ source code for a four-state Markov chain.

The proof of optimality of the scheme for stationary ergodic sources using capacity-approaching channel codes uses the tools developed in [**EVKV02**].

As in the case of memoryless sources, the computational complexities of both compression and decompression grow linearly with the data size.

Figure 11 shows the results obtained with a 4-state Markov chain with conditional biases equal to $(0.1, 0.6, 0.4, 0.9)$ leading to an entropy rate of 0.5407 bits per symbol. The coding scheme does not code 350 of the source symbols (corresponding to the segments at the output of the BWT with highest entropy). Varying the number of symbols allocated to closed-loop iterative doping (which now operates in fixed-length mode) yields different rate values. Again, the degree of approximation to the ideal coding curve is rather encouraging.

Our basic scheme can be adapted to Slepian-Wolf coding of correlated stationary ergodic sources for which the fundamental limits were proved in [**Cov75**]. Note that the constructive approaches for Slepian-Wolf coding proposed so far in the literature [**BM01, GFZ01, Mur02, LXG02**] are limited to memoryless sources.

Suppose that the sources are correlated according to $\mathbf{s}_1 = \mathbf{s}_2 + \mathbf{u}$ where $\mathbf{u}$ is a Markov process independent of the stationary ergodic process $\mathbf{s}_2$. To approach the point in the Slepian-Wolf-Cover region given by the entropy rate of source 2 and the conditional entropy rate of source 1 given source 2, we may use the BWT-BP-based approach explained above to encode and decode source 2, and then run a BP to recover $\mathbf{s}_1$ on a Tanner graph that incorporates not only the parity-check matrix of the encoder applied to source 1 but the Markov graphical model of $\mathbf{u}$. Note that in this case, no BWT of source 1 is carried out at the expense of higher complexity at the decoder.

7. Universal Scheme

No knowledge of the source is required by the BWT. However, we need a robust LDPC design procedure that yields ensembles of codes that perform close to capacity for nonstationary memoryless channels. Since the capacity-achieving distribution is the same (equiprobable) regardless of the noise distribution, Shannon theory (e.g. [**CK81**]) guarantees that without knowledge of the channel sequence at the encoder it is possible to attain the channel capacity (average of the maximal single-letter mutual informations) with the same random coding construction that achieves the capacity of the stationary channel.

In our universal implementation, the encoder estimates the piecewise constant first-order distribution at the output of the BWT, uses it to select a parity-check matrix rate, and communicates a rough piecewise approximation of the estimated distributions to the decoder using a number of symbols that is a small fraction of the total compressed sequence. After each iteration, the BP decoder refines its estimates of individual probabilities on a segment-by-segment basis by weighing the impact of each tentative decision with the current reliability information. Naturally, such procedures are quite easy to adapt to prior knowledge that may exist from the encoding of the previous data blocks. Moreover, the encoder can monitor the statistics from block to block. When it sees that there is a large change in statistics it can alert the decoder to the segments that are most affected, or even back off in rate.

If the encoder knew that the source is Markov source with S states, then it could detect the segment transitions (or boundary points) T_i just by looking at the first $M = \log_2 S$ columns of the BWT array. In this case, each probability p_i can be estimated by the empirical frequency of ones in the i-th segment. If the source memory is not known a priori, then all memories $M = 0, 1, \ldots$, up to some maximum D can be considered, and the corresponding empirical frequencies can be obtained hierarchically, on a tree. In fact, due to the lexicographic ordering of the BWT, segments corresponding to states $(s, 0)$ and $(s, 1)$ of a memory $M+1$ model can be merged into state s of a memory M model, and the empirical frequency of ones in state s segment can be computed recursively by

$$\begin{aligned} n_1(s) &= n_1(s0) + n_1(s1) \\ n(s) &= n(s0) + n(s1) \end{aligned} \tag{7.1}$$

where $n_1(s)$ and $n(s)$ denote the number of ones and the segment length in segment s (e.g. [**BB**]).

A coarse description of the segmentation and associated probabilities p_i is supplied to the iterative decoder. As outlined above, the decoder may be given the

option of refining this statistical description iteratively. The discrete set of possible sources $\mathcal{S}$ communicated to the decompressor corresponds to all possible quantized probabilities and quantized transition points, for all possible source memories. We denote the quantized parameters of a particular instance of the source model by $\theta \in \mathcal{S}$. We wish to encode the transition points and the probabilities p_i by using

$$L_n(\theta) = \frac{S(\theta)}{2} \log_2 n + O(1) \quad \text{bits} \tag{7.2}$$

where $S(\theta)$ is the number of states in the model θ. Apart from the value of the constant term, this is the best possible redundancy to encode the source model, according to the minimum description length (MDL) principle [**Ris84**].

A simple heuristic method approaching this lower bound is the following. For each given memory M, let the exact transition points of the segments identified by the first M columns of the BWT array be denoted by $T_i(M)$, for $i = 1, \ldots, 2^M - 1$. We can write

$$T_i(M) = \kappa_i \sqrt{n} + \zeta_i$$

where $\zeta_i = T_i(\theta)$ modulo $\sqrt{n}$ and $\kappa_i = \lfloor T_i(M)/\sqrt{n} \rfloor$. Then, we quantize the remainder ζ_i by using b_1 bits. Hence, the location of the transition point $T_i(M)$ is known up to a maximum error of $\sqrt{n} 2^{-b_1}$. We let $\widehat{T}_i(M)$ denote the quantized value corresponding to $T_i(M)$. Notice that after quantization some of the originally distinct transition points might have been quantized to the same value, i.e., some segments after quantization have disappeared from the resulting source model. Let $\{\widehat{T}_j(\theta) : j = 1, \ldots, S(\theta) - 1\}$, denote the set of *distinct* transition points after quantization, where $S(\theta)$ denotes the number of states in the source model θ. Notice that, by construction, $S(\theta) \leq 2^M$. Let p_j denote the empirical frequency of ones in the j-th segment identified by the transition points $\{\widehat{T}_j(\theta)\}$. We use b_2 bits to encode the log-ratios $\mathcal{L}_j(\theta) = \log(1-p_j)/p_j$. The decoder will apply the (quantized) log-ratio $\mathcal{L}_j(\theta)$ on the positions of segment j.

Clearly, each κ_i can be encoded by $\frac{1}{2} \log_2 n$, therefore the description length for the model θ is

$$L_n(\theta) = (S(\theta) - 1)(\frac{1}{2} \log_2 n + b_2) + S(\theta) b_1$$

which is compliant with the MDL principle (7.2).

The degrees of freedom available to the encoder are the model to be described $\theta \in \mathcal{S}$ and the ensemble of LDPC matrices to be chosen. For every matrix the scheme is the variable-length version explained in Section 6 using the closed-loop iterative doping algorithm.

We consider a collection of Q LDPC ensembles (defined by the left and right degree distributions (λ_q, ρ_q) for $q = 1, \ldots, Q$) of rates $0 < R_1 < \ldots < R_Q < 1$, and, for each q-th ensemble, a set of c parity-check matrices

$$\mathcal{H}_q = \left\{ \mathbf{H}_{i,q} \in \{0,1\}^{m_q \times n} \ : \ i = 1, \ldots, c \right\} \tag{7.3}$$

such that $m_q/n = 1 - R_q$ and each matrix $\mathbf{H}_{i,q}$ is independently and randomly generated over the ensemble (λ_q, ρ_q). Then, in order to encode $\mathbf{x}$, the proposed scheme finds the model $\theta \in \mathcal{S}$ and the set of parity-check matrices $\mathcal{H}_q$ that minimize the overall output length, i.e., it finds

$$(\widehat{q}, \widehat{\theta}) = \arg \min_{q,\theta} \{ L_n(\theta) + M_n(q, \theta, \mathbf{x}) \} \tag{7.4}$$

with $M_n(q,\theta,\mathbf{x})$ equal to the output length of the basic LDPC compression scheme using $\mathcal{H}_q$ and θ at the decoder when applied to sourceword $\mathbf{x}$. It then encodes the source model $\widehat{\theta}$ using $L_n(\widehat{\theta})$ bits, and the sourceword $\mathbf{x}$ with the basic LDPC compression scheme using the parity-check matrices in $\mathcal{H}_{\widehat{q}}$.

It is problematic to evaluate $M_n(q,\theta,\mathbf{x})$, as this would require the application of the Closed-loop Iterative Doping Algorithm to $\mathbf{x}$, for all parity-check matrices $\{\mathcal{H}_q : q = 1,\ldots,Q\}$ and all models $\theta \in \mathcal{S}$. However, we make the following observation. Let $\widehat{H}_\theta(\mathbf{x})$ be the empirical entropy of $\mathbf{x}$ according to the probability model θ. Then, if $R_q > 1 - \widehat{H}_\theta(\mathbf{x})$, the number of doping bits $d(\mathbf{x})$ is very large. On the contrary, if $R_q < 1 - \widehat{H}_\theta(\mathbf{x})$, $d(\mathbf{x})$ is very small with respect to n. Hence, we shall use the heuristic approximation:

$$M_n(q,\theta,\mathbf{x}) = m_{q(\theta)} \tag{7.5}$$

where

$$q(\theta) = \max\{1 \le q \le Q : R_q < 1 - \widehat{H}_\theta(\mathbf{x})\}.$$

In other words, we choose the LDPC ensemble with largest rate not above the *normalized information density* of the deterministically time-varying BSC determined by θ with noise realization $\mathbf{x}$.

As explained in [**BB**], the BWT and the recursive segment determination can be obtained with complexity linear in n by using suffix-trees methods. This makes the overall complexity of our algorithm linear with n, although the (constant with n) complexity due to BP is large with respect to other universal linear complexity algorithms based on sequential arithmetic coding.

We compare the performance of the proposed data compression algorithm with the standard compression software `gzip` (based on the Lempel-Ziv algorithm) and `bzip` (based on postprocessing the BWT output using Move-to-Front runlength coding and adaptive arithmetic coding).

Figure 12 shows the histogram of the normalized output lengths obtained from 2000 independent trials for a four-state binary Markov source with entropy rate 0.4935 bit/symbols, for the new scheme, `gzip` and `bzip`, for blocklength $n = 10,000$. We used $b_2 = 8$ quantization bits for the log-ratios, $b_1 = 3$ quantization bits for the transition points and a collection of LDPC ensembles with rates equally spaced from 0.005 to 0.0995. For each ensemble, $c = 8$ parity-check matrices were randomly generated.

Instead of testing a given source, our last experiment considers an ensemble of randomly generated binary Markov sources with number of states equally likely to be 1, 2, 4, 8 and 16 (i.e., with memory equal to 0, 1, 2, 3 and 4). The Markov source ensemble is obtained by generating independently the memory length, and then conditional distributions are also generated randomly with a distribution that puts more weight on values either close to 0 or to 1. The ensemble is restricted to produce sources with entropy ranging from 0.05 to 0.75 bits per symbol. Figures 13 and 14 examine histograms of the normalized redundancy (difference between the compression rate and the empirical entropy of the source) for our universal scheme, `gzip` and `bzip` for blocklengths equal to 3,000 and 10,000 respectively. For the latter case, the memory of the Markov source can be up to 5. For our scheme the parameters b_1 and b_2 that govern the quantization coarseness in the description of the segmentation are adapted using the Minimum Description Length principle described above.

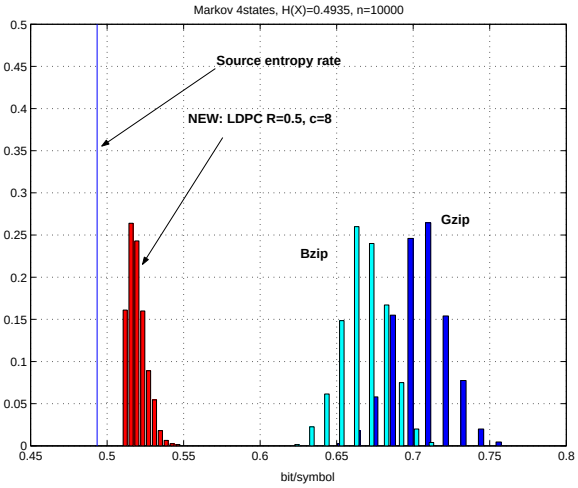

FIGURE 12. Histogram of normalized output lengths for the LDPC compressor, `gzip` and `bzip` for a Markov source with entropy = 0.4935 bit/symbol.

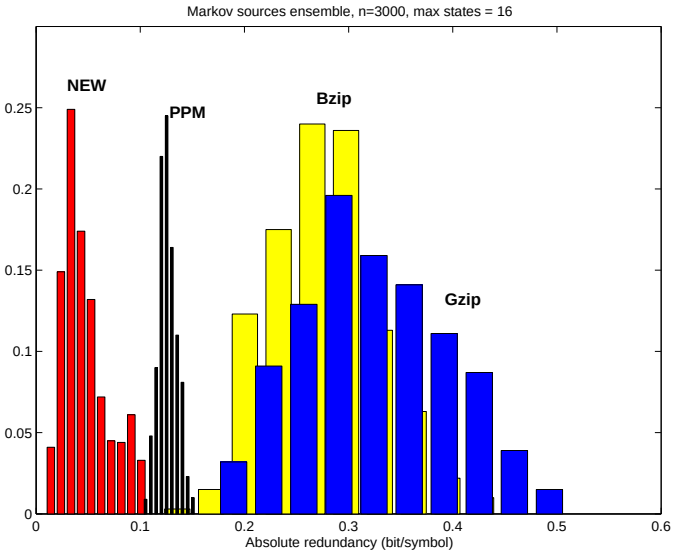

FIGURE 13. Random ensemble of Markov sources: redundancies for the new universal compressor, PPM, `bzip` and `gzip` with source blocklength equal to 3,000.

We see that as blocklength increases all schemes tend to become more efficient, but the new scheme (which requires linear compression and decompression complexity) maintains an edge even with respect to the PPM scheme which requires quadratic computational complexity. The resilience against errors and the application of this approach to joint source-channel coding are treated in [**CSV04**].

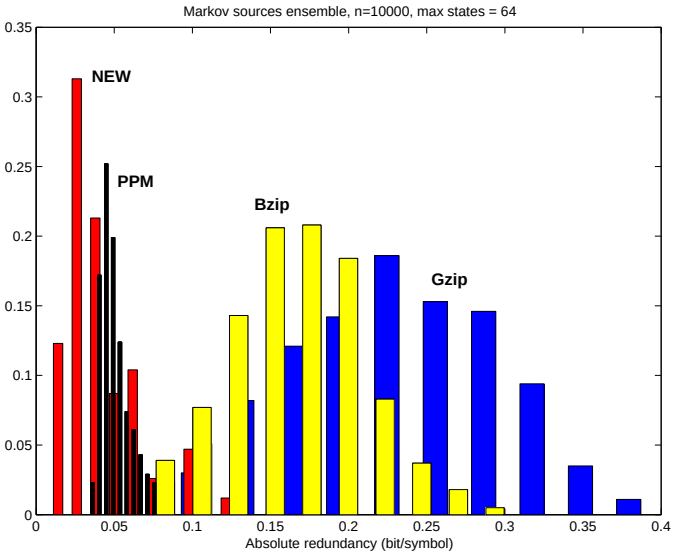

FIGURE 14. Random ensemble of Markov sources: redundancies for the new universal compressor, PPM, `bzip` and `gzip` with source blocklength equal to 10,000.

References

[AB72] P. E. Allard and A. W. Bridgewater, *A source encoding technique using algebraic codes*, Proc. 1972 Canadian Computer Conference (June 1972), 201–213.

[Anc76] T. Ancheta, *Syndrome source coding and it universal generalization*, IEEE Trans. Information Theory **22, no. 4** (July 1976), 432 – 436.

[BB] D. Baron and Y. Bresler, *A work-efficient parallel two-pass MDL context tree algorithm for universal source coding*, submitted for publication.

[BM01] J. Bajcsy and P. Mitran, *Coding for the Slepian-Wolf problem with Turbo codes*, Proc. 2001 IEEE Globecom (2001), 1400–1404.

[BW94] M. Burrows and D. J. Wheeler, *A block-sorting lossless data compression algorithm*, Tech. Rep. SRC 124, May 1994.

[CK81] I. Csiszar and J. Korner, *Information theory: Coding theorems for discrete memoryless systems*, Academic, New York, 1981.

[Cov75] T. M. Cover, *A proof of the data compression theorem of slepian and wolf for ergodic sources*, IEEE Trans. Information Theory **IT-22** (1975), 226–228.

[CS02] G. Caire and S. Shamai, *Writing on dirty tape with LDPC codes*, DIMACS Workshop on Sig. Proc. for Wireless Transmission (Rutgers University, NJ, USA), October 2002.

[CSSV] G. Caire, S. Shamai, A. Shokrollahi, and S. Verdú, *Fountain codes for data compression*, in preparation.

[CSV] G. Caire, S. Shamai, and S. Verdú, *Data compression with error correction codes*, in preparation.

[CSV04] ———, *Almost-noiseless joint source-channel coding-decoding of sources with memory*, 5th International ITG Conference on Source and Channel Coding (SCC) (Jan 14-16, 2004).

[DM98] M. C. Davey and D. MacKay, *Low density parity check codes over GF(q)*, IEEE Communications Letters **2** (June 1998), 165–167.

[Eli55] P. Elias, *Coding for noisy channels*, IRE Convention record **4** (1955), 37–46.

[EVKV02] M. Effros, K. Visweswariah, S. Kulkarni, and S. Verdú, *Universal lossless source coding with the Burrows-Wheeler transform*, IEEE Trans. on Information Theory **48, no. 5** (May 2002), 1061–1081.

[FTS73] K. C. Fung, S. Tavares, and J. M. Stein, *A comparison of data compression schemes using block codes*, Conf. Rec. IEEE Int Electrical and Electronics Conf. (Oct. 1973), 60–61.

[Gal63] R. G. Gallager, *Low-density parity-check codes*, MIT Press, 1963.

[GFZ01] J. Garcia-Frias and Y. Zhao, *Compression of correlated binary sources using Turbo codes*, IEEE Communication Letters **5** (Oct. 2001), 417–419.

[GFZ02] ———, *Compression of binary memoryless sources using punctured Turbo codes*, IEEE Communication Letters **6** (Sep. 2002), 394–396.

[HV93] T. S. Han and S. Verdú, *Approximation theory of output statistics*, IEEE Trans. Information Theory **39** (May 1993), 752–772.

[LXG02] A. D. Liveris, Z. Xiong, and C. N. Georghiades, *Compression of binary sources with side information at the decoder using LDPC codes*, IEEE Communication Letters **6, no. 10** (Oct. 2002), 440–442.

[McE02] R.J. McEliece, *The theory of information and coding: Second edition*, Cambridge University Press, Cambridge, U. K., 2002.

[McM53] B. McMillan, *The basic theorems of information theory*, Ann. Math. Statist. **24** (June 1953), 196–219.

[Mur02] T. Murayama, *Statistical mechanics of the data compression theorem*, J. Phys. A: Math. Gen. **35** (2002), L95L100.

[Ris84] J. Rissanen, *Universal coding, information prediction and estimation*, IEEE Trans. on Information Theory **30, No. 4** (July 1984), 629–636.

[RU01a] T. J. Richardson and R. L. Urbanke, *The capacity of low-density parity-check codes under message-passing decoding*, IEEE Trans. Information Theory **47** (Feb. 2001), 599–618.

[RU01b] ———, *Efficient encoding of low-density parity-check codes*, IEEE Trans. Information Theory **47** (Feb. 2001), 638–656.

[SF02] D. Sridhara and T. E. Fuja, *Low density parity check codes over groups and rings*, Proc. 2002 IEEE Information Theory Workshop (Oct. 2002), 163–166.

[Sha48] C. E. Shannon, *A mathematical theory of communication*, Bell Sys. Tech. J. **27** (Jul.-Oct. 1948), 379–423, 623–656.

[Sho03] A. Shokrollahi, *Raptor codes*, Preprint, 2003.

[tB00] S. ten Brink, *Designing iterative decoding with the extrinistic information transfer chart*, AEU. Int. J. Electron. Commun. **34, No. 6** (2000), 389–398.

[TM] D. S. Taubman and M. W. Marcellin, *JPEG2000: standard for interactive imaging.*

[Urb02] R. Urbanke, *http://lthcwww.epfl.ch/research/ldpcopt/*, 2002.

[Wei62] E. Weiss, *Compression and coding*, IRE Transactions on Information Theory (Apr. 1962), 256–257.

[WFH99] U. Wachsmann, R. Fisher, and J. Huber, *Multilevel codes: theoretical concepts and practical design rules*, IEEE Trans. Information Theory **45** (1999), no. 5, 1361–1391.

[Wyn74] A. D. Wyner, *Recent results in the Shannon theory*, IEEE Trans. Information Theory **IT-20** (Jan. 1974).

Acknowledgment

Thanks to Mr. H. Cai for his help with the simulations of `bzip`, `gzip` and `PPM`.

Institut Eurecom, Sophia Antipolis, France
E-mail address: `giuseppe.caire@eurecom.fr`

Technion, Haifa, Israel
E-mail address: `sshlomo@ee.technion.ac.il`

Princeton University
E-mail address: `verdu@princeton.edu`