Complexity-Compression Tradeoffs in Lossy Compression via Efficient Random Codebooks and Databases

Christos Gioran, Ioannis Kontoyiannis, *†‡

Abstract

The compression-complexity trade-off of lossy compression algorithms that are based on a random codebook or a random database is examined. Motivated, in part, by recent results of Gupta-Verdú-Weissman (GVW) and their underlying connections with the pattern-matching scheme of Kontoyiannis' lossy Lempel-Ziv algorithm, we introduce a non-universal version of the lossy Lempel-Ziv method (termed LLZ). The optimality of LLZ for memoryless sources is established, and its performance is compared to that of the GVW divide-and-conquer approach. Experimental results indicate that the GVW approach often yields better compression than LLZ, but at the price of much higher memory requirements. To combine the advantages of both, we introduce a hybrid algorithm (HYB) that utilizes both the divide-and-conquer idea of GVW and the single-database structure of LLZ. It is proved that HYB shares with GVW the exact same rate-distortion performance and implementation complexity, while, like LLZ, requiring less memory, by a factor which may become unbounded, depending on the choice of the relevant design parameters. Experimental results are also presented, illustrating the performance of all three methods on data generated by simple discrete memoryless sources. In particular, the HYB algorithm is shown to outperform existing schemes for the compression of some simple discrete sources with respect to the Hamming distortion criterion.

Keywords: Lossy data compression, rate-distortion theory, pattern matching, Lempel-Ziv, random codebook, fixed database

1 Introduction

One of the last major outstanding classical problems of information theory is the development of general-purpose, practical, efficiently implementable lossy compression algorithms. The corresponding problem for lossless data compression was essentially settled in the late 1970s by the advance of the Lempel-Ziv (LZ) family of algorithms [64][62][65] and arithmetic coding [46][43][29]; see also the texts [22][6]. Similarly, from the early- to mid-1990s on, efficient channel coding strategies emerged that perform close to capacity, primarily using sparse graph

*C. Gioran and I. Kontoyiannis are with Department of Informatics, Athens University of Economics and Business, Patission 76, Athens 10434, Greece. e-mail: himicos@gmail.com, yiannis@aueb.gr

†I. Kontoyiannis was supported in part by a Marie Curie International Outgoing Fellowship, PIOF-GA-2009-235837.

‡A preliminary version of these results was presented at the 2009 IEEE Information Theory Workshop, in Volos, Greece, June 2009.

codes, turbo codes, and local message-passing decoding algorithms; see, e.g., [53][33][48][34], the texts [16][32][44], and the references therein.

For lossy data compression, although there is a rich and varied literature on both theoretical results and practical compression schemes, near-optimal, efficiently implementable algorithms are yet to be discovered. From rate-distortion theory [7][47] we know that it is possible to achieve a sometimes dramatic improvement in compression performance by allowing for a certain amount of distortion in the reconstructed data. But the majority of existing algorithms are either compression-suboptimal or they involve exhaustive searches of exponential complexity at the encoder, making them unsuitable for realistic practical implementation.

Until the late 1990s, most of the research effort was devoted to addressing the issue of universality, see [26] and the references therein, as well as [61][41][63][42][40][59][60][55]; algorithms emphasizing more practical aspects have been proposed in [57]. In addition to many application-specific families of compression standards (e.g., JPEG for images and MPEG for video), there is a general theory of algorithm design based on vector quantization; see [17][30][8][19] and the references therein. Yet another line of research, closer in spirit to the present work, is on lossy extensions of the celebrated Lempel-Ziv schemes, based on approximate pattern matching; see [39][49][58][31][3][56][4][14][27][1].

More recently, there has been renewed interest in the compression-complexity trade-off, and in the development of low-complexity compressors that give near-optimal performance, at least for simple sources with known statistics. The lossy LZ algorithm of [27] is rate-distortion optimal and of polynomial complexity, although, in part due to the penalty paid for universality, its convergence is slow. For the uniform Bernoulli source, [37][38][36] present codes based on sparse graphs, and, although their performance is promising, like earlier approaches they rely on exponential searches at the encoder. In related work, [52][9] present sparse-graph compression schemes with much more attractive complexity characteristics, but suboptimal compression performance. Rissanen and Tabus [45] describe a different method which, unlike most of the earlier approaches, is not based on a random (or otherwise exponentially large) codebook. It has linear complexity in the encoder and decoder and, although it appears to be rate-distortion suboptimal, it is an effective practical scheme for Bernoulli sources. Sparse-graph codes that are compression-optimal and of subexponential complexity are constructed in [20]. A simulation-based iterative algorithm is presented in [24] and it is shown to be compression-optimal, although its complexity is hard to evaluate precisely as it depends on the convergence of a Markov chain Monte Carlo sampler. The more recent work [23] on the lossy compression of discrete Markov sources also contains promising results; it is based on the combination of a Viterbi-like optimization algorithm at the encoder followed by universal lossless compression.

The present work is partly motivated by the results reported in [21] by Gupta-Verdú-Weissman (GVW). Their compression schemes are based on the "divide-and-conquer" approach, namely the idea that instead of encoding a long message $x_1^n = (x_1, x_2, \ldots, x_n)$ using a classical random codebook for blocks of length n, it is preferable to break up x_1^n into shorter sub-blocks of shorter length ℓ, say, and encode the sub-blocks separately. The main results in [21] state that, with an appropriately chosen sub-block length ℓ, it is possible to achieve asymptotically optimal rate-distortion performance with low implementation complexity (in a sense made precise in Section 3 below).

Our starting point is the observation that there is a closely related, in a sense dual, point of view. On a conceptual as well as mathematical level, the divide-and-conquer approach is very closely related to a pattern-matching scheme with a restricted database. In the divide-and-conquer setting, given a target distortion level D and an $\ell \geq 1$, each sub-block of length ℓ in the original message x_1^n is encoded using a random codebook consisting of $\approx 2^{\ell R(D)}$ codewords, where $R(D)$ is the rate-distortion function of the source being compressed (see the following section for more details and rigorous definitions). To encode each sub-block, the encoder searches all $2^{\ell R(D)}$ entries of the codebook, in order to find the one which has the smallest distortion with respect to that sub-block.

Now suppose that, instead of a random codebook, the encoder and decoder share a random database with length $M \approx 2^{\ell R(D)}$, generated from the same distribution as the Shannon-optimal codebook. As in [27], the encoder searches for the longest prefix $x_1^L = (x_1, x_2, \ldots, x_L)$ of the message x_1^n that matches somewhere in the database with distortion D or less. Then the prefix x_1^L is described to the decoder by describing the position and length of the match in the database, and the same process is repeated inductively starting at x_{L+1}. Although the match-length L is random, we know [14][27] that, asymptotically, it behaves like,

$$L \approx \frac{\log M}{R(D)} \approx \ell, \quad \text{with high probability.}$$

Therefore, because the length M of the database was chosen to be $\approx 2^{\ell R(D)}$, in effect both schemes will individually encode sub-blocks of approximately the same length ℓ, and will also have comparable implementation complexity at the encoder.[1]

Thus motivated, after reviewing the GVW scheme in Section 2 we introduce a (non-universal) version of the lossy LZ scheme in [27], termed LLZ, and we compare its performance to that of GVW. Theorem 1 shows that LLZ is asymptotically optimal in the rate-distortion sense for compressing data from a known discrete memoryless source with respect to a single-letter distortion criterion. Simulation results are also presented, comparing the performance of LLZ and GVW on a simple Bernoulli source. These results indicate that for message lengths around 1000 bits, GVW offers better compression than LLZ at a given distortion level, but it requires significantly more memory for its execution. [The same findings are also confirmed in the simulation examples presented in Section 4.]

In order to combine the different advantages of the two schemes, in Section 3 we introduce a hybrid algorithm (HYB), which utilizes both the divide-and-conquer idea of GVW and the single-database structure of LLZ. In Theorems 2 and 3 we prove that HYB shares with GVW the exact same rate-distortion performance and low implementation complexity. Moreover, like LLZ, the HYB scheme requires much less memory, by an unbounded factor, depending on the choice of parameters in the design of the two algorithms. Experimental results are presented in Section 4, comparing the performance of GVW and HYB. These confirm the theoretical findings, and indicate that HYB outperforms existing schemes for the compression of some simple discrete sources with respect to the Hamming distortion criterion. The earlier theoretical results stating that HYB's rate-distortion performance is the same as GVW's are

[1] It is well-known that the main difficulty in designing effective lossy compressors is in the implementation complexity of the *encoder*. Therefore, in all subsequent results dealing with complexity issues we focus on the case of the encoder. Moreover, it is easy to see that the *decoding* complexity of all the schemes considered here is linear in the message length.

confirmed empirically, and it is also shown that, again for lengths of approximately 1000 symbols, the HYB scheme requires much less memory, by a factor ranging between 15 and 240.

Finally we note that the main motivation for this work was the introduction of memory considerations in the classical compression-complexity trade-off. Of course, such considerations have been implicit in much of the existing literature, most notably in work related to so-called "structured source coding" and "structured vector quantization." In this connection, in addition to the references discussed earlier we mention tree codes developed by Jelinek and others [25][2][17, Ch. 15]; and source codes based on trellises [51][18][35][50]. Also, there is a long line of work on compression algorithms based on linear codes, of which the most complexity-efficient ones are those that combine a linear code with an encoder utilizing sparse-graph properties or a message-passing-type algorithm; see, e.g., [36][52] and the references therein.

After a brief discussion on potential extensions of the present results, some conclusions are collected in Section 5. The appendix contains the proofs of the theorems in Sections 2 and 3.

2 The GVW and LLZ algorithms

After describing the basic setting within which all later results will be developed, in Section 2.2 we recall the divide-and-conquer idea of the GVW scheme, and in Section 2.3 we present a new, non-universal lossy LZ algorithm and examine its properties.

2.1 The setting

Let $\{X_n\} = \{X_1, X_2, \ldots\}$ be a memoryless source on some finite alphabet A and suppose that its distribution is described by a known probability mass function P on A. The objective is to compress $\{X_n\}$ with respect to a sequence of single-letter distortion criteria,

$$\rho_n(x_1^n, y_1^n) = \frac{1}{n}\sum_{i=1}^{n} \rho(x_1, y_1), \quad n \geq 1,$$

where $x_1^n = (x_1, x_2, \ldots, x_n) \in A^n$ is an arbitrary source string to be compressed, $y_1^n = (y_1, y_2, \ldots, y_n)$ is a potential reproduction string taking values in a finite reproduction alphabet $\hat{A}$, and $\rho : A \times \hat{A} \to [0, \infty)$ is an arbitrary distortion measure. We make the customary assumption that for any source letter x there is a reproduction letter y with zero distortion,

$$\max_{x \in A} \min_{y \in \hat{A}} \rho(x, y) = 0.$$

The best achievable rate at which data from the source $\{X_n\}$ can be compressed with distortion not exceeding $D \geq 0$ is given by the *rate-distortion function* [47][7][10],

$$R(D) = \inf_{W(y|x):\sum_{x,y\in A} P(x)W(y|x)\rho(x,y)\leq D} I(X;Y), \tag{1}$$

where $I(X;Y)$ denotes the mutual information between a random variable X with the same distribution P as the source and a random variable Y with conditional distribution $W(\cdot|x)$

given $X = x$.[2] Let $D_{\max} = \min_{y\in\hat{A}} E_P[\rho(X,y)]$; in order to avoid the trivial case where $R(D)$ is identically equal to zero, $D_{\max}$ is assumed to be strictly positive. It is well-known and easy to check that, for all distortion values in the nontrivial range $0 < D < D_{\max}$, there is a conditional distribution $W^*(\cdot|\cdot)$ that achieves the infimum in (1), and this induces a distribution Q^* on $\hat{A}$ via $Q^*(y) = \sum_{x\in A} P(x)W^*(y|x)$, for all $y \in \hat{A}$. With a slight abuse of terminology (as Q^* may not be unique) we refer to Q^* as *the optimal reproduction distribution at distortion level* D. Recall also the analogous definition of the *distortion-rate function* $D(R)$ of the source; cf. [7][10].

2.2 The GVW algorithm

The GVW algorithm[3] is a fixed-rate, variable-distortion code for messages of length n, with target distortion $D \in (0, D_{\max})$. It is described in terms of two parameters; a "small" $\gamma > 0$, and an integer ℓ so that $n = k\ell$.

Given the target distortion level D, let $R = R(D) + \gamma$, and take,

$$\overline{D} = R^{-1}\Big(R(D) + \gamma/2\Big) = D\Big(R(D) + \gamma/2\Big) \leq D. \tag{2}$$

First a fixed-rate code of blocklength ℓ and rate R is created according to Shannon's classical random codebook construction. Letting Q^* denote the optimal reproduction distribution at level $\overline{D}$, the codebook consists of $\lfloor 2^{\ell R}\rfloor$ i.i.d. codewords of length ℓ, each generated i.i.d. from Q^*. Writing $x_1^n = x_1^\ell * x_{\ell+1}^{2\ell} * \cdots * x_{(k-1)\ell+1}^{k\ell}$, as the concatenation of k sub-blocks, each sub-block is matched to its ρ_ℓ-nearest neighbor in the codebook, and it is described to the decoder using $\lceil \log\lfloor 2^{\ell R}\rfloor\rceil \approx \ell R$ bits to describe the index of that nearest neighbor in the codebook.

This code is used k times, once on each of the k sub-blocks, to produce corresponding reconstruction strings $y_{(i-1)\ell+1}^{i\ell}$, for $i = 1, 2, \ldots, k$. The description of x_1^n is the concatenation of the descriptions of the individual sub-blocks, and the reconstruction string itself is the concatenation of the corresponding reproduction blocks, $y_1^n = y_1^\ell * y_{\ell+1}^{2\ell} * \cdots * y_{(k-1)\ell+1}^{k\ell}$. The overall description length of this code is $k\lceil \log\lfloor 2^{\ell R}\rfloor\rceil \leq k(\ell R + 1) = nR + k$ bits, so the (fixed, asymptotic) rate of this code is $\leq R$ bits/symbol, and its (variable) distortion is $\rho_n(x_1^n, y_1^n)$.

2.3 The lossy Lempel-Ziv algorithm LLZ

The LLZ algorithm described here can be seen as a simplified (in that it is non-universal) and modified (to facilitate the comparison below) version of the algorithm in [27]. It is a fixed-distortion, variable-rate code for messages of length n, described in terms of three parameters; an integer blocklength $\ell \leq n$, and "small" $\alpha, \gamma > 0$.[4] The algorithm will be presented in a setting "dual" to that of the GVW algorithm, in the sense that was described in the Introduction. The main difference is that the source sting x_1^n will be parsed into substrings of *variable* length, not of fixed length ℓ.

[2] The mutual information, rate-distortion function, and all other standard information-theoretic quantities here and throughout are expressed in bits; all logarithms are taken to be in base 2, unless stated otherwise.

[3] To be more precise, this is one of two closely related schemes discussed in [21]; see the relevant comments in Section 3.

[4] Note that in [27] a fixed-rate, variable-distortion *universal* code is also described, but we restrict attention here to the conceptually simpler fixed-distortion algorithm.

Given n and a target distortion level D, define $R = R(D) + \gamma$, take,

$$\overline{D} = R^{-1}\Big(R(D) - \gamma/2\Big) = D\Big(R(D) - \gamma/2\Big) \geq D,$$

and let Q^* denote the optimal reproduction distribution at level $\overline{D}$. [Note that, although the quantity $\overline{D}$ defined here is different from that defined for the GVW in (2), it plays exactly the same role.] Then generate a single i.i.d. database $Y_1^m = (Y_1, Y_2, \ldots, Y_m)$ of length,

$$m = m(\ell) = \lfloor 2^{\ell R} \rfloor + \ell - 1, \tag{3}$$

and make it available to both the encoder and decoder.

The encoding algorithm is as follows: The encoder calculates the length of the longest match (up to $(1+\alpha)\ell$-many symbols) of an initial portion of the message x_1^n, within distortion $\overline{D}$, in the database. Let $L_{\ell,1}$ denote the length of this longest match,

$$L_{\ell,1} = \max\{1 \leq k \leq (1+\alpha)\ell \,:\, \rho_k(x_1^k, Y_i^{i+k-1}) \leq \overline{D} \text{ for some } 1 \leq i \leq m-k+1\},$$

and let $Z^{(1)} = x_1^{L_{\ell,1}}$ denote the initial phrase of length $L_{\ell,1}$ in x_1^n. Then the encoder describes to the decoder:

(a) the length $L_{\ell,1}$; this takes $\lceil \log((1+\alpha)\ell) \rceil$ bits;

(b) the position i in the database where the match occurs; this takes $\lceil \log m \rceil$ bits.

From (a) and (b) the decoder can recover the string $\hat{Z}^{(1)} = Y_i^{i+L_{\ell,1}-1}$, which is within distortion $\overline{D}$ of $Z^{(1)}$.

Alternatively, $Z^{(1)}$ can be described with *zero* distortion by first describing its length $L_{\ell,1}$ as before, and then describing $Z^{(1)}$ itself directly using,

$$\lceil L_{\ell,1} \log |\hat{A}| \rceil \quad \text{bits.} \tag{4}$$

The encoder uses whichever one of the two descriptions is shorter. [Note that is not necessary to add a flag to indicate which one was chosen; the decoder can simply check if $\lceil L_{\ell,1} \log |\hat{A}| \rceil$ is larger or smaller than $\lceil \log m \rceil$.] Therefore, from (a), (b), and (4) the length of the description of $Z^{(1)}$ is,

$$\lceil \log((1+\alpha)\ell) \rceil + \min\{\lceil \log m \rceil, \lceil L_{\ell,1} \log |\hat{A}| \rceil\} \quad \text{bits.} \tag{5}$$

After $Z^{(1)}$ has been described within distortion $\overline{D}$, the same process is repeated to encode the rest of the message: The encoder finds the length $L_{\ell,2}$ of the longest string starting at position $(L_{\ell,1}+1)$ in x_1^n that matches within distortion $\overline{D}$ into the database, and describes $Z^{(2)} = x_{L_{\ell,1}+1}^{L_{\ell,1}+L_{\ell,2}}$ to the decoder by repeating the above steps. The algorithm is terminated, in the natural way, when the entire string x_1^n has been exhausted. At that point, x_1^n has been parsed into $\Pi_\ell = \Pi_\ell(x_1^n, D)$ distinct phrases $Z^{(k)}$, each of length $L_{\ell,k}$, $x_1^n = Z^{(1)} * Z^{(2)} * \cdots * Z^{(\Pi_\ell)}$, with the possible exception of the last phrase, which may be shorter. Since each substring $Z^{(k)}$ is described within distortion $\overline{D}$, also the concatenation of all the reproduction strings, call it $\psi_1^n := \hat{Z}^{(1)} * \hat{Z}^{(2)} * \cdots * \hat{Z}^{(\Pi_\ell)}$, will be within distortion $\overline{D}$ of x_1^n.

The distortion achieved by this code is $\rho_n(x_1^n, \psi_1^n)$, and it is guaranteed to be $\leq \overline{D}$ by construction. Regarding the rate, if we write $\Lambda(x_1^n) = \Lambda(x_1^n, \ell, D)$ for the overall description length of x_1^n, then from (5),

$$\Lambda(x_1^n) = \sum_{k=1}^{\Pi_\ell} \Big[\lceil \log((1+\alpha)\ell) \rceil \ + \min\{ \lceil \log m \rceil, \ \lceil L_{\ell,k} \log |\hat{A}| \rceil \} \Big] \ \text{bits}, \tag{6}$$

and the rate achieved by this code is $\Lambda(x_1^n)/n$ bits/symbol.

Remark. As mentioned in the Introduction, there are two main differences between the GVW algorithm and the LLZ scheme. The first one is that while the GVW is based on a Shannon-style random codebook, the LLZ uses an LZ-type random database. The second is that GVW divides up the message x_1^n into fixed-length sub-blocks of size ℓ, whereas LLZ parses x_1^n into variable-length strings of (random) lengths $L_{\ell,k}$. But there is also an important point of solidarity between the two algorithms. Recall [15, Theorem 23] that, for large ℓ, the match length $L_{\ell,1}$ behaves logarithmically in the size of the database; that is, with high probability,

$$L_{\ell,1} \approx \frac{\log m(\ell)}{R(\overline{D})} \approx \ell,$$

where the second approximation follows by the choice of $m(\ell)$ and of $\overline{D}$. Therefore, both algorithms end up parsing the message x_1^n into sub-blocks of length $\approx \ell$ symbols.

Our first result shows that LLZ is asymptotically optimal in the usual sense established for fixed-database versions of LZ-like schemes; see [54][27]. Specifically, it is shown that by taking ℓ large enough and γ small enough, the LLZ comes arbitrarily close to any optimal rate-distortion point $(R(D), D)$. Note that $\alpha > 0$ is a parameter that simply controls the complexity of the best-match search, and its influence on the rate-distortion performance is asymptotically irrelevant. The proof is given in the appendix.

Theorem 1. [LLZ Optimality] Suppose the LLZ with parameters ℓ, α and γ is used to compress a memoryless source $\{X_n\}$ with rate-distortion function $R(D)$ at a target distortion rate $D \in (0, D_{\max})$. For any $\delta > 0$, the parameter $\gamma > 0$ can be chosen small enough such that:
(a) For any choice of ℓ and any message length n, the distortion achieved by LLZ is no greater than $D + \delta$.
(b) Taking ℓ large enough, the asymptotic rate of LLZ achieves the rate-distortion bound, in that,

$$\limsup_{\ell\to\infty} \limsup_{n\to\infty} \ E\left\{ \frac{\Lambda(X_1^n, \ell, D)}{n} \ \Big| \ X_1^n \right\} \ \leq \ R(\overline{D}) = R(D) - \gamma/2 \quad \text{bits/symbol, w.p.1}, \tag{7}$$

where the expectation is over all databases. Therefore, also,

$$\limsup_{\ell\to\infty} \limsup_{n\to\infty} \ E\left\{ \frac{\Lambda(X_1^n, \ell, D)}{n} \right\} \ \leq \ R(\overline{D}) = R(D) - \gamma/2 \quad \text{bits/symbol}, \tag{8}$$

with the expectation here being over both the message and the databases.

Next, the performance of LLZ is compared with that of GVW on data generated from a Bernoulli source with parameter $p = 0.4$ and with respect to Hamming distortion. Simulation results at different target distortions are shown in Figure 1 and Table 1 below; see Section 4 for details on the choice of parameter values. It is clear from these results that, at the same distortion level, the GVW algorithm typically gives a better rate than LLZ. In terms of implementation complexity, the two algorithms have comparable execution times, but the LLZ uses significantly less memory. The same pattern – GVW giving better compression but using much more memory than LLZ – is also confirmed in the other examples we consider in Section 4.

Note that, like for the case of GVW, more can be said about the implementation complexity of LLZ and how it depends on the exact choice of parameters ℓ, α and γ. But since, as we will see next, the performance of both algorithms is dominated by that of a different algorithm (HYB), we do not pursue this direction further.

3 The HYB algorithm

In order to combine the rate-distortion advantage of GVW with the memory advantage of LLZ, in this section we introduce a hybrid algorithm and examine its performance.

The new algorithm, termed HYB, uses the divide-and-conquer approach of GVW, but based on a random database like the LLZ instead of a random codebook. It is a fixed-rate, variable-distortion code for messages of length n, with target distortion $D \in (0, D_{\max})$, and it is described in terms of two parameters; a "small" $\gamma > 0$, and an integer ℓ so that $n = k\ell$.

Like with the GVW, given a target distortion level D, let $R = R(D) + \gamma$ and take $\overline{D}$ as in (2). Now, like for the LLZ algorithm, let $m = m(\ell) = \lfloor 2^{\ell R} \rfloor + \ell - 1$ as in (3), and generate a random database $Y_1^m = (Y_1, Y_2, \ldots, Y_m)$, where the Y_i are drawn i.i.d. from the optimal reproduction distribution at level $\overline{D}$. The database is made available to both the encoder and the decoder, and the message x_1^n to be compressed is parsed into $k = n/\ell$ non-overlapping blocks, $x_1^n = x_1^\ell * x_{\ell+1}^{2\ell} * \cdots * x_{(k-1)\ell+1}^{k\ell}$.

The first sub-block x_1^ℓ is matched to its ρ_ℓ-nearest neighbor in the database, where we consider each possible $Y_i^{i+\ell-1}$, $i = 1, 2, \ldots, \lfloor 2^{\ell R} \rfloor$ as a potential reproduction word. Then x_1^ℓ is described to the decoder by describing the position of its matching reproduction block in the database using $\approx \ell R$ bits, and the same process is repeated on each of the k sub-blocks, to produce k reconstruction strings. The description of x_1^n is the concatenation of the descriptions of the individual sub-blocks, and the reconstruction string itself is the concatenation of the corresponding reproduction blocks. The overall description length of this code is $k\lceil \log \lfloor 2^{\ell R} \rfloor \rceil \leq k\ell R = nR$ bits.

The following result, proved in the appendix, shows that the HYB algorithm shares the exact same rate-distortion performance, as well as the same implementation complexity characteristics, as the GVW. Let:

$$\hat{\gamma} = \min\{1, 2(R(D/2) - R(D))\}.$$

Theorem 2. [HYB Compression/Complexity Trade-off] Consider a memoryless source $\{X_n\}$ with rate-distortion function $R(D)$, which is to be compressed at target distortion level $D \in (0, D_{\max})$. There exists an $\hat{\epsilon} > 0$ such that, for any $0 < \epsilon < \hat{\epsilon}$, the HYB algorithm with parameters $0 < \gamma < \hat{\gamma}$ and ℓ as in (12) achieves a rate of $R = R(D) + \gamma$ bits/symbol, its expected distortion is less than $D + \epsilon$, and moreover:

– Encoding time per source symbol is proportional to $(\lambda_1/\epsilon)^{\lambda_2(D)/\gamma^2}$,
– Decoding time per symbol is independent of γ and ϵ,
where λ_1 and $\lambda_2(D)$ are independent of ϵ and γ.

Remarks.

1. Theorem 2 is an exact analog of Theorem 1 proved for GVW in [21], the only difference being that we consider average distortion instead of the probability-of-excess distortion criterion. The reason is that, instead of presenting an existence proof for an algorithm with certain desired properties, here we examine the performance of the HYB algorithm itself. Indeed, the proof of Theorem 2 can easily be modified to prove the stronger claim that there exists *some* instance of the random database Y_1^m such that, using that particular database, the HYB algorithm also has the additional property that the probability of excess distortion vanishes as $n \to \infty$. The same comments apply to Theorem 3 below.

2. In [21] a similar result is proved with the roles and ϵ and γ interchanged. In fact, it should be pointed out that the scheme we call "the" GVW algorithm here corresponds to the scheme used in the proof of [21, Theorem 1]. A slight variant (having to do with the choice of parameter values and not with the mechanics of the algorithm itself) is used to prove [21, Theorem 2]. Having gone over the proofs, it would be obvious to the reader that, once the corresponding changes are made for HYB, an analogous result can be proved for HYB. The straightforward but tedious details are omitted.

3. In terms of memory, the GVW scheme requires $\ell\lfloor 2^{\ell R}\rfloor$ reproduction symbols for storing the codebook, while using the same memory parameters the HYB algorithm needs $m(\ell) = \lfloor 2^{\ell R}\rfloor + \ell - 1$ symbols. The ratio between the two is,

$$\frac{\text{memory for GVW}}{\text{memory for HYB}} = \frac{\ell\lfloor 2^{\ell R}\rfloor}{\lfloor 2^{\ell R}\rfloor + \ell - 1} \approx \ell,$$

so that the GVW needs $\approx \ell$ *times* more memory than HYB. Moreover, the closer we require the algorithm to come to achieving an optimal $(D, R(D))$ point, the smaller the values of ϵ and α need to be taken in Theorem 2, and the larger the corresponding value of ℓ; cf. equation (12). Therefore, not only the difference, but even the ratio of the memory required by GVW compared to HYB, is unbounded.[5]

On the other hand, it is easily seen that if we were to decrease the rate in the GVW algorithm by $(\log \ell)/\ell$ bits/symbol, for large ℓ we would obtain essentially the same performance. This parameter change is insignificant for large ℓ, but it would certainly force the compression achieved to deteriorate for small values of ℓ. Indeed, the focus of this work, as well as that of [21], is on relatively small values of ℓ. This can be seen both from the parameter choices in the experimental results presented later on, as well as from the actual expression for ℓ that appears in equation (12); there, ℓ is defined by an expression of the form,

$$\ell = \lceil A(D)\gamma^{-2}\log(B(D)\epsilon^{-1})\rceil,$$

where the constants $A(D)$ and $B(D)$ depend only on D. In particular, note that ℓ does *not* depend on the message length n.

[5] Note that the memory usage of HYB can be reduced further to $\lfloor 2^{\ell R}\rfloor$ symbols, if a cyclic indexing convention is used as in, e.g., [24].

In order to clarify this point further, we look briefly at the specific values of the parameters involved in the first simulation experiment reported in the following section. There, as well as in the corresponding experiments for the GVW given in [21], ℓ is chosen as $\ell \approx 22/R(D)$, where $R(D)$ is the rate-distortion function of a Bernoulli source with parameter $p = 0.4$. For distortion values in the rage $0.05 \leq D \leq 0.3$, the corresponding values of ℓ range approximately between 30 and 220, so that the corresponding *rate increase* that we would need to incur in order for the GVW and HYB to use the same memory ranges between 0.035 and 0.16 bits/symbol. Thus the amount by which the compression performance would degrade is quite substantial – and certainly not negligible.

The next result shows examines, for a particular choice of the parameters ℓ and γ in HYB, the tradeoff achieved between compression performance and encoding/decoding complexity. It is a parallel result to [21, Theorem 3].

Theorem 3. [HYB Near-Linear Complexity] For a memoryless source $\{X_n\}$ with rate-distortion function $R(D)$, a target distortion level $D \in (0, D_{\max})$, and an *arbitrary* increasing and unbounded function $g(n)$, the HYB algorithm with appropriately chosen parameters $\ell = \ell(n)$ and $\gamma = \gamma(n)$, achieves a limiting rate equal to $R(D)$ bits/symbol and limiting average distortion D. The encoding and decoding complexities are $O(ng(n))$ and $O(n)$ respectively.

The proof is given in the appendix. The actual empirical performance of HYB on simulated data is compared to that of GVW and LLZ in the following section.

4 Simulation results

Here the empirical performance of the HYB scheme is compared with that of GVW and LLZ, on two simulated data sets from simple memoryless sources.[6] The following parameter values were used in all of the experiments. For the GVW and HYB algorithms, ℓ was chosen as in [21] to be $\ell = \lceil \frac{22}{R(D)+\gamma} \rceil$, where $R(D)$ is the rate-distortion function of the source, and γ was taken equal to 0.002. Similarly, for LLZ we took $\ell = \lceil 22/R(D) \rceil$, $\gamma = 0.03$ and $\alpha = 0.1$. Note that, with this choice of parameters, the complexity of all three algorithms is essentially identical. All experiments were performed on a Sony Vaio laptop running Ubuntu Linux with 4MB of memory, under identical conditions.[7]

First we revisit the example of Section 2; $n = 1050$ bits generated by a Bernoulli source with parameter $p = 0.4$, are compressed by all three algorithms at various different distortion levels with respect to Hamming distortion. Figure 1 shows the rate-distortion pairs achieved.

Rate-distortion performance. It is evident that the compression performance obtained by GVW and HYB is near-identical, and better than that of LLZ. This example was also examined by Rissanen and Tabus in [45], where it was noted that it is quite hard for any implementable scheme to produce rate-distortion pairs below the straight line connecting the end-points $(D, R(D))$ of the rate-distortion curve corresponding to $D = 0$ and $D = 0.4$. As

[6] We do not present comparison results with earlier schemes apart from the GVW, since extensive such studies already exist in the literature; in particular, the GVW is compared in [21] with the algorithms proposed in [52], [20] and [45].

[7] Although there is a wealth of efficient algorithms for the problem of approximate string matching (see, e.g., [12][3][5][11] and the references therein), since HYB clearly outperforms LLZ, our version of the LLZ scheme was implemented using the naive, greedy scheme consistent with the definition of algorithm.

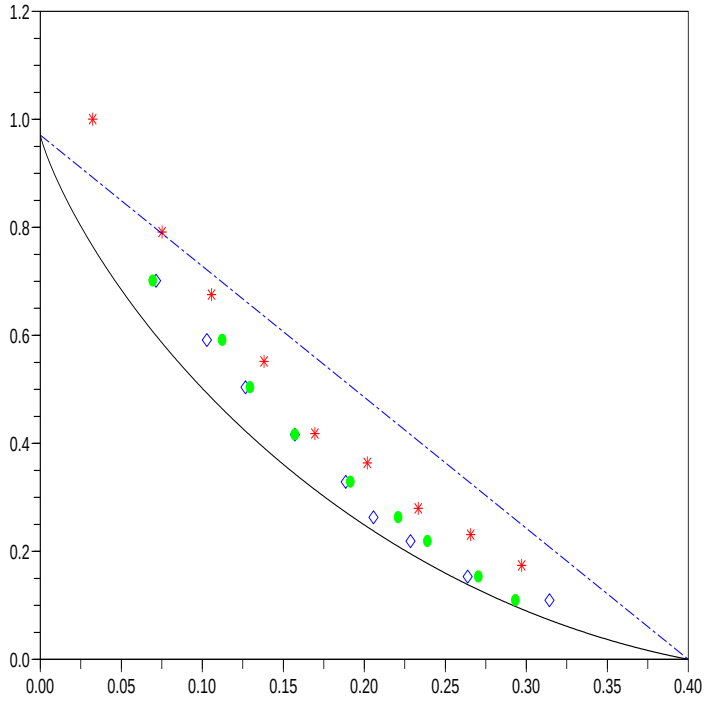

Fig. 1: Comparison of the rate-distortion performance of GVW, LLZ and HYB on a data string of length $n = 1050$ bits generated from a Bernoulli source with parameter $p = 0.4$. The solid convex curve is the rate-distortion function; the rate-distortion pairs achieved by GVW are shown as blue diamonds; by LLZ as red stars; and by HYB as bold green dots.

noted in [21], the Rissanen-Tabus scheme produces results slightly below the straight line, and it is one of the best implementable schemes for this problem.

Memory and complexity. Table 1 contains a complete listing off all performance parameters obtained in the above experiment, including the execution time required for the encoder and the total amount of memory used. As already observed in Section 2, the LLZ scheme requires much less memory that GVW, and so does the hybrid algorithm HYB. In fact, while GVW and HYB produce essentially identical rate-distortion performance, the HYB algorithm requires between 32 and 213 *times* less memory than GVW. [Note that these figures are deterministic; the memory requirement is fixed by the description of the algorithm and it is not subject to random variations produced by the simulated data.] In terms of the corresponding execution times, the GVW and HYB share the exact same theoretical complexity in their implementation. Nevertheless, because of the vastly different memory requirements, in practice we find that the execution times of HYB were approximately 3 to 10 times faster than GVW.

Bern(0.4) source, Hamming distortion					
	Performance parameters				
Algorithm	D_{target}	D_{achieved}	rate	memory	time
GVW	0.05	0.07143	0.70095	26MB	27m53s
GVW	0.08	0.10286	0.59143	23MB	21m11s
GVW	0.11	0.12667	0.50381	27MB	20m48s
GVW	0.14	0.15714	0.41619	31MB	19m52s
GVW	0.17	0.18857	0.32857	36MB	18m48s
GVW	0.2	0.20571	0.26286	46MB	19m18s
GVW	0.23	0.22857	0.21905	57MB	18m42s
GVW	0.26	0.26381	0.15333	79MB	19m46s
GVW	0.29	0.31429	0.10952	113MB	20m29s
LLZ	0.05	0.03238	1.00029	1.5MB	4m23s
LLZ	0.08	0.07524	0.79129	1.28MB	6m15s
LLZ	0.11	0.10571	0.6754	1.46MB	8m53s
LLZ	0.14	0.1381	0.55171	1.69MB	11m18s
LLZ	0.17	0.16952	0.41827	2.6MB	18m15s
LLZ	0.2	0.2019	0.36381	3.6MB	20m09s
LLZ	0.23	0.23333	0.27975	6.2MB	41m32s
LLZ	0.26	0.26571	0.23102	13MB	63m56s
LLZ	0.29	0.29714	0.1741	47MB	165m54s
HYB	0.05	0.06952	0.70095	0.79MB	2m45s
HYB	0.08	0.11238	0.59143	0.6MB	3m06s
HYB	0.11	0.12952	0.50381	0.59MB	3m33s
HYB	0.14	0.15714	0.41619	0.56MB	4m06s
HYB	0.17	0.19143	0.32857	0.52MB	4m40s
HYB	0.2	0.22095	0.26286	0.53MB	5m21s
HYB	0.23	0.23905	0.21905	0.51MB	5m26s
HYB	0.26	0.27048	0.15333	0.53MB	6m27s
HYB	0.29	0.29333	0.10952	0.53MB	6m56s

Tab. 1: Performance achieved by the HYB algorithm on a data string of length $n = 1050$ bits generated from a Bernoulli source with parameter $p = 0.4$.

In the second example $\{X_n\}$ is taken as a memoryless source uniformly distributed on $\{0, 1, 2, 3\}$, to be compressed with respect to Hamming distortion. The empirical results are shown in Figure 2 and Table 2.

In both these cases, the same qualitative conclusions are drawn. The rate-distortion performance of the GVW and HYB algorithms is essentially indistinguishable, while the compression achieved by LLZ is generally somewhat worse, though in several instances not significantly so. In the second example note that the memory required by HYB is smaller than that of GVW by a factor that ranges between 44 and 242, while in the third example the corresponding factors are between 16 and 218. And again, although the theoretical implementation complexity of

GVW and HYB is identical, because of their different memory requirements the encoding time of HYB is smaller than that of GVW by a factor ranging between approximately 3 and 9 in the second example, and between 1.25 and 1.5 in the third example.

Finally, from both examples above we observe that, at low rates, the LLZ algorithm's memory usage is large its execution times are slow. HYB, on the other hand, is both fastest and most memory-efficient among all three schemes, both low and high rates.

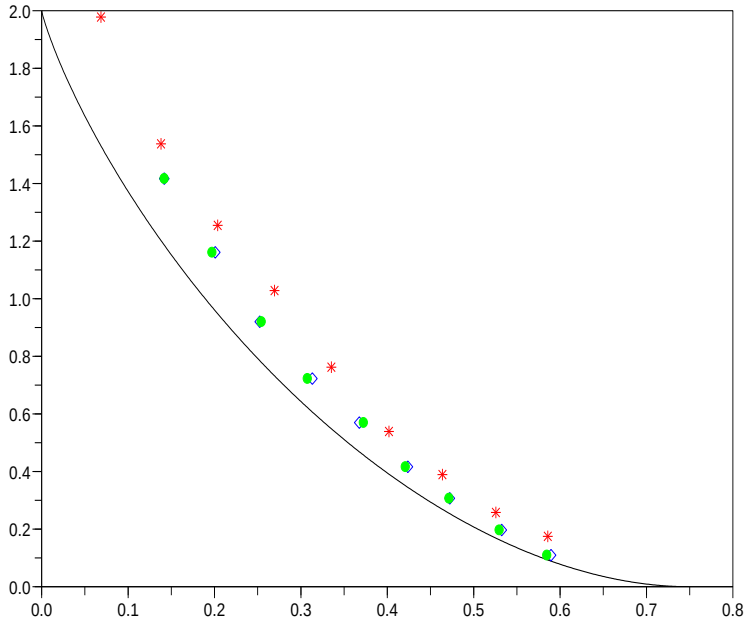

Fig. 2: Comparison of the rate-distortion performance of GVW, LLZ and HYB on a data string of length $n = 1050$ symbols generated from the Uniform source on $\{0, 1, 2, 3\}$. The solid curve is the rate-distortion function; the rate-distortion pairs achieved by GVW are shown as blue diamonds; by LLZ as red stars; and by HYB as bold green dots.

5 Conclusions and Extensions

The starting point for this work was the observation that there is a certain duality relationship between the divide-and-conquer compression schemes of Gupta-Verdú-Weissman (GVW) in [21], and certain lossy Lempel-Ziv schemes based on a fixed-database as in [27]. To explore this duality, LLZ, a new (non-universal) lossy LZ algorithm was introduced, and it was shown to be asymptotically rate-distortion optimal. To combine the low-complexity advantage of GVW with the low-memory requirement of LLZ, a hybrid algorithm, called HYB, was then proposed, and its properties were explored both theoretically and empirically.

$U\{0,1,2,3\}$ **source, Hamming distortion**					
	Performance parameters				
Algorithm	D_{target}	D_{achieved}	rate	memory	time
GVW	0.1	0.1419	1.41714	43MB	10m27s
GVW	0.16	0.20095	1.16095	24MB	6m44s
GVW	0.22	0.25238	0.92	31MB	8m19s
GVW	0.28	0.31333	0.72286	44MB	11m12s
GVW	0.34	0.36762	0.56952	43MB	9m45s
GVW	0.4	0.42381	0.41619	65MB	12m29s
GVW	0.46	0.47238	0.30667	92MB	13m59s
GVW	0.52	0.53238	0.19714	124MB	14m12s
GVW	0.58	0.58952	0.10952	229MB	17m30s
LLZ	0.1	0.06857	1.97778	3.597MB	9m54s
LLZ	0.16	0.1381	1.53794	1.79MB	7m46s
LLZ	0.22	0.20381	1.25461	2.04MB	12m50s
LLZ	0.28	0.26952	1.02841	2.61MB	18m51s
LLZ	0.34	0.33524	0.76228	3.445MB	28m25s
LLZ	0.4	0.4019	0.5393	3.49MB	30m37s
LLZ	0.46	0.46381	0.3893	5.44MB	46m19s
LLZ	0.52	0.52571	0.25807	14.6MB	105m56s
LLZ	0.58	0.58571	0.17475	104MB	62m16s
HYB	0.1	0.1419	1.41714	2.58MB	7m49s
HYB	0.16	0.19714	1.16095	1.22MB	5m06s
HYB	0.22	0.25429	0.92	1.26MB	6m37s
HYB	0.28	0.30762	0.72286	1.39MB	8m48s
HYB	0.34	0.37238	0.56952	1.05MB	7m42s
HYB	0.4	0.42095	0.41619	1.18MB	9m39s
HYB	0.46	0.47143	0.30667	1.15MB	10m34s
HYB	0.52	0.52952	0.19714	1.01MB	10m14s
HYB	0.58	0.58476	0.10952	1.05MB	11m43s

Tab. 2: Comparison of the performance of GVW, LLZ and HYB on a data string of length $n = 1050$ symbols generated from the Uniform source on $\{0, 1, 2, 3\}$.

The main contribution of this paper is the introduction of memory considerations in the usual compression-complexity trade-off. Building on the success of the GVW algorithm, it was shown that the HYB scheme simultaneously achieves three goals: 1. Its rate-distortion performance can be made arbitrarily close to the fundamental rate-distortion limit; 2. The encoding complexity can be tuned in a rigorous manner so as to balance the trade-off of encoding complexity vs. compression redundancy; and 3. The memory required for the execution of the algorithm is much smaller than that required by GVW, a difference which may be made arbitrarily large depending on the choice of parameters.

Moreover, empirically, for messages of length of the order of thousands, the HYB scheme

appears to outperform existing schemes for the compression of simple memoryless sources with respect to Hamming distortion.

Lastly, we briefly mention that the results presented in this paper can be extended in several directions. First we note that the finite-alphabet assumption was made exclusively for the sake of simplicity of exposition and to avoid cumbersome technicalities. While keeping the structure of all three algorithms exactly the same, this assumption can easily be relaxed, at the price of longer, more technical proofs, along the lines of arguments, e.g., in [55][27][28][21]. For example, Theorem 4 of [21] which gives precise performance and complexity bounds for the GVW used with general source and reproduction alphabets and with respect to an unbounded distortion measure, can easily be generalized to HYB. Similarly, Theorem 5 of [21] which describes the performance of a universal version of GVW can also be generalized to the corresponding statement for a universal version of HYB (with obvious modifications), although, as noted in [21], the utility of that result is purely of theoretical interest.

A Proof of Theorem 1.

Recall that under the present assumptions the rate-distortion function $R(D)$ is continuous, differentiable, convex and nonincreasing [7][13]. Given $D \in (0, D_{\max})$ and $\delta > 0$, assume without loss of generality that $D+\delta < D_{\max}$; then we can choose $\gamma > 0$ according to $R(D+\delta) = R(D) - \gamma/2$, so that $\overline{D} = D + \delta$. [As it does not change the asymptotic analysis below, we take $\alpha > 0$ fixed and arbitrary.] Then the distortion part of the theorem is immediate by the construction of the algorithm.

Before considering the rate, we record two useful asymptotic results for the match-lengths $L_{\ell,k}$. Let $R = R(D) + \gamma$, and $m = m(\ell) = \lfloor 2^{\ell R} \rfloor + \ell - 1$ as in (3). Then [15, Theorem 23] immediately implies that,

$$\lim_{\ell\to\infty} \frac{L_{\ell,1}}{\log m(\ell)} = \frac{1}{R(\overline{D})} \quad \text{w.p.1.}$$

Moreover, for any $\epsilon > 0$, the following more precise asymptotic lower bound on $L_{\ell,1}$ holds: As $\ell \to \infty$,

$$(\log m(\ell)) \Pr\left\{ L_{\ell,1} \le \frac{\log m(\ell)}{R(\overline{D}) + \epsilon} \;\middle|\; X_1^n \right\} \;\to\; 0 \quad \text{w.p.1.} \tag{9}$$

The proof of (9) is a straightforward simplification of the proof of [27, Corollary 3], and therefore omitted.

Now let $\epsilon > 0$ arbitrary. The encoder parses the message X_1^n into Π_ℓ distinct words $Z^{(k)}$, each of length $L_{\ell,k}$. We let $N = (\log m(\ell))/(R(\overline{D}) + \epsilon)$ and following [54] we assume, without loss of generality, that N is an integer and that the last phrase is complete, i.e.,

$$Z^{(\Pi_\ell)} \;\text{ has length }\; L_{\ell,\Pi_\ell}.$$

To bound above the rate obtained by LLZ, we consider phrases of different lengths separately. We call a phrase $Z^{(k)}$ *long* if its length satisfies $L_{\ell,k} > N$, and we call $Z^{(k)}$ *short* otherwise. Recalling (6), the total description length of the LLZ can be broken into two parts as,

$$\Lambda(X_1^n) \le \sum_{k:\, Z^{(k)} \text{ is short}} \Big[\lceil \log((1+\alpha)\ell) \rceil + \lceil L_{\ell,k} \log |\hat{A}| \rceil \Big] + \sum_{k:\, Z^{(k)} \text{ is long}} \Big[\lceil \log((1+\alpha)\ell) \rceil + \lceil \log m \rceil \Big]. \tag{10}$$

For the first sum we note that, by the choice of $m(\ell)$ and the definition of a short phrase, each summand is bounded above by a constant times N, at least for all ℓ large enough; therefore, the conditional expectation of the whole sum given X_1^n,

$$E\left\{C_1\, N \sum_{k=1}^{\Pi_\ell} \mathbb{I}_{\{L_{\ell,k}\leq N\}} \,\Big|\, X_1^n\right\}$$

is bounded above by,

$$C_2 \log m(\ell)\, n \,\Pr\left\{L_{\ell,1} \leq \frac{\log m(\ell)}{R(\overline{D})+\epsilon} \,\Big|\, X_1^n\right\},$$

where $\mathbb{I}_F$ denotes the indicator function of an event F, and the inequality follows by considering not just all k's, but all the possible positions in X_1^n where a short match can occur. Dividing by n and letting $n \to \infty$, from (9) we get that this expression converges to zero w.p. 1, so that the conditional expectation of the first term in (10) also converges to zero, w.p.1.

For the second and dominant term in (10), let Π'_ℓ be the number of long phrases $Z^{(k)}$. Since each long $Z^{(k)}$ has length $L_{\ell,k} \geq N$, we must have $N\Pi'_\ell \leq n$, so that

$$\frac{\Pi'_\ell}{n} \leq \frac{R(\overline{D})+\epsilon}{\log m(\ell)}. \tag{11}$$

Also, by the definition of $m(\ell)$, for all ℓ large enough (independently of n), we have,

$$\log((1+\alpha)\ell) \leq \epsilon \log m(\ell).$$

Therefore, the second sum in (10) can be bounded above by,

$$\Pi'_\ell\,(1+\epsilon)\log m(\ell) \leq n(1+\epsilon)(R(\overline{D})+\epsilon).$$

Combining this with the fact that the first term in (10) vanishes, immediately yields that,

$$\limsup_{\ell\to\infty} \limsup_{n\to\infty}\; E\left\{\frac{\Lambda(X_1^n, D, \ell)}{n} \,\Big|\, X_1^n\right\}$$

is bounded above w.p.1 by,

$$(R(\overline{D})+\epsilon)(1+\epsilon),$$

and since $\epsilon > 0$ was arbitrary we get the first claim in the theorem. Finally, the second claim follows from the first and Fatou's lemma. □

B Proof of Theorem 2.

The proof of the theorem is based on Lemma 1 below, which plays the same role as [21, Lemma 1] in the proof of [21, Theorem 1]. The rest of of the proof is identical, except for the fact that we do not need to invoke the law of large numbers, since here we do not claim that the probability of excess distortion goes to zero.

Before stating the lemma, we define the following auxiliary quantities: $D_1 = D/2$, $K(D) = (D - D_1)/(R(D_1) - R(D))$,

$$C(D) = \min\Big\{\frac{K(D)^2}{8{D_{\max}}^2}, \frac{1}{32(R'(D/2)D_{\max})^2}, \frac{1}{4}\Big\},$$

and,

$$\hat{\epsilon} = \min\Big\{\frac{\exp\{16C(D)\}}{3(D_{\max} - D)}, 3e^{-1}(D_{\max} - D)\Big\}.$$

Lemma 1. Consider a memoryless source $\{X_n\}$ to be compressed at target distortion level $D \in (0, D_{\max})$. Then for any $0 < \epsilon < \hat{\epsilon}$, the HYB algorithm with parameters $0 < \gamma < \hat{\gamma}$ and

$$\ell = \left\lceil \frac{1}{C(D)\gamma^2} \log \frac{3(D_{\max} - D)}{\epsilon} \right\rceil, \tag{12}$$

when applied to a single block X_1^ℓ achieves rate $R = R(D) + \gamma$ on any X_1^ℓ, and its expected distortion (averaged over all source strings X_1^ℓ) is less than $D + \epsilon$.

Proof. Given $\epsilon > 0$, choose a positive $\epsilon' < \epsilon$ such that,

$$\frac{\epsilon'}{C(D)} \log \frac{3(D_{\max} - D)}{\epsilon'} < \epsilon.$$

Now follow the proof of [21, Lemma 1] with ϵ' in place of ϵ, until the beginning of the computation of the probability of excess distortion. The key observation is that, for HYB, this probability can be bounded above by the excess-distortion probability with respect to a random codebook with

$$\frac{1}{\ell} 2^{\ell R(\overline{D})} = 2^{\ell(R(D)+\gamma-\frac{\log \ell}{\ell})}$$

words, by just considering possible matches starting at positions $i = 1, \ell + 1, 2\ell + 1, \ldots$, making the corresponding potentially matching blocks in the database independent. Therefore, following the same computation, the required probability can be bounded above exactly as in the proof of [21, Lemma 1] by,

$$2(2^{-\ell C(D)\gamma^2}) + \ell 2^{-\ell\gamma/4}. \tag{13}$$

The first term is bounded above by,

$$\frac{2\epsilon'}{(D_{\max} - D)},$$

as before, and in order to show that the expected distortion is less than ϵ it suffices to show that the last term satisfies,

$$(D_{\max} - D)\ell 2^{\ell(R(D)+\gamma)} < \epsilon/3. \tag{14}$$

Substituting the choice of ℓ from (12), it becomes,

$$\frac{(D_{\max} - D)}{C(D)\gamma^2} \log\Big(\frac{3(D_{\max} - D)}{\epsilon'}\Big) 2^{-\frac{1}{4\gamma C(D)} \log(3(D_{\max} - D)/\epsilon')}$$

and since γ is restricted to be less than one, this can in turn be bounded above, uniformly in $\gamma \in (0,1)$, by its value at $\gamma = 1$. [To see that, note that the function $f(x) = Ax^2 \exp\{-Bx\}$ is increasing for $x < 2/B$ and decreasing for $x > 2/B$. By our choice of $\hat{\epsilon}$, the maximum above is achieved at the point $x = 1/\gamma = 1$.] Therefore, noting also that $4C(D) \leq 1$, this term is bounded above by,

$$\frac{(D_{\max} - D)}{C(D)} \log\Big(\frac{3(D_{\max} - D)}{\epsilon'}\Big) 2^{-\log(3(D_{\max}-D)/\epsilon')},$$

which, after some algebra, simplifies to,

$$\frac{\epsilon'}{3C(D)} \log\Big(\frac{3(D_{\max} - D)}{\epsilon'}\Big),$$

and this is less than $\epsilon/3$ by the choice of ϵ'. This establishes (14) and completes the proof of the lemma. □

C Proof of Theorem 3.

Taking $c > 0$ arbitrary, we let, as in the proof of [21, Theorem 3],

$$\ell(n) = \left\lceil \frac{\log g(n)}{R(D) + c} \right\rceil \quad \text{and} \quad \gamma(n) = \sqrt{\frac{\log \ell(n)}{\ell(n)}}.$$

For each n we use HYB with the corresponding parameters; the rate result follows from the construction of the algorithm, which has rate no larger than,

$$R(D) + \gamma(n) \to R(D) \quad \text{bits/symbol},$$

as $n \to \infty$.

Regarding the distortion, equation (13) in the proof of Theorem 2 shows that that the probability of the event that the distortion of the ith block will exceed D is bounded above by,

$$2(2^{-\ell(n)C(D)\gamma(n)^2}) + \ell(n)2^{-\ell(n)\gamma(n)/4}.$$

It is easily seen that, for large n, this is dominated by the second term,

$$\ell(n)2^{-(1/4)\sqrt{\ell(n)\log\ell(n)}}.$$

Therefore, the distortion of any one ℓ-block is bounded above by,

$$D + D_{\max}\ell(n)2^{-(1/4)\sqrt{\ell(n)\log\ell(n)}}.$$

Noting that the excess term goes to zero as $n \to \infty$, it will still go to zero when averaged out over all $n/\ell(n)$ sub-blocks, and, therefore, the expected distortion over the whole message X_1^n will converge to D.

Finally, the complexity results are straightforward by construction; the details can be found in the corresponding discussion for the GVW scheme in [21, Section II-A]. □

References

[1] M. Alzina, W. Szpankowski, and A. Grama. 2D-pattern matching image and video compression. *IEEE Trans. Image Processing*, 11:318–331, 2002.

[2] J.B. Anderson and F. Jelinek. A 2-cycle algorithm for source coding with a fidelity criterion. *IEEE Trans. Information Theory*, IT-19(1):77–92, 1973.

[3] D. Arnaud and W. Szpankowski. Pattern matching image compression with prediction loop: Preliminary experimental results. In *Proc. Data Compression Conf. – DCC 97*, Los Alamitos, California, 1997. IEEE, IEEE Computer Society Press.

[4] M. Atallah, Y. Génin, and W. Szpankowski. Pattern matching image compression: Algorithmic and empirical results. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21:618–627, 1999.

[5] M. Atallah, Y. Génin, and W. Szpankowski. Pattern matching image compression: Algorithmic and empirical results. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21, 1999.

[6] J.G. Bell, T.C. Cleary and I.H. Witten. *Text Compression*. Prentice Hall, New Jersey, 1990.

[7] T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.

[8] P.A. Chou, M. Effros, and R.M. Gray. A vector quantization approach to universal noiseless coding and quantizations. *IEEE Trans. Inform. Theory*, 42(4):1109–1138, 1996.

[9] S. Ciliberti, M. Mézard, and R. Zecchina. Message-passing algorithms for non-linear nodes and data compression. *ComPlexUs*, 3:58–65, 2006.

[10] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. J. Wiley, New York, 1991.

[11] M. Crochemore and T. Lecroq. Pattern-matching and text-compression algorithms. *ACM Computing Surveys*, 28(1):39–41, 1996.

[12] M. Crochemore and W. Rytter. *Text Algorithms*. Oxford University Press, New York, 1994.

[13] I. Csiszár and J. Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, New York, 1981.

[14] A. Dembo and I. Kontoyiannis. The asymptotics of waiting times between stationary processes, allowing distortion. *Ann. Appl. Probab.*, 9:413–429, 1999.

[15] A. Dembo and I. Kontoyiannis. Source coding, large deviations, and approximate pattern matching. *IEEE Trans. Inform. Theory*, 48:1590–1615, June 2002.

[16] B.J. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, Cambridge, MA, USA, 1998.

[17] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[18] R.M. Gray. Time-invariant trellis encoding of ergodic discrete-time sources with a fidelity criterion. *IEEE Trans. Information Theory*, IT-23(1):71–83, 1977.

[19] R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Trans. Inform. Theory*, 44(6):2325–2383, 1998.

[20] A. Gupta and S. Verdú. Nonlinear sparse-graph codes for lossy compression. *Information Theory, IEEE Transactions on*, 55(5):1961 –1975, May 2009.

[21] A. Gupta, S. Verdú, and T. Weissman. Rate-distortion in near-linear time. In *IEEE International Symposium on Information Theory*, pages 847–851, July 2008.

[22] D. Hankerson, G.A. Harris, and P.D. Johnson, Jr. *Introduction to Information Theory and Data Compression*. CRC Press LLC, 1998.

[23] S. Jalali, A. Montanari, and T. Weissman. An implementable scheme for universal lossy compression of discrete Markov sources. *Preprint*, 2008.

[24] S. Jalali and T. Weissman. Rate-distortion via Markov chain Monte Carlo. In *Proc. of the IEEE International Symposium on Inform. Theory*, pages 852–856, Toronto, Canada, July 2008.

[25] F. Jelinek. Tree encoding of memoryless time-discrete sources with a fidelity criterion. *IEEE Trans. Information Theory*, IT-15:584–590, 1969.

[26] J.C. Kieffer. A survey of the theory of source coding with a fidelity criterion. *IEEE Trans. Inform. Theory*, 39(5):1473–1490, 1993.

[27] I. Kontoyiannis. An implementable lossy version of the Lempel-Ziv algorithm – Part I: Optimality for memoryless sources. *IEEE Trans. Inform. Theory*, 45(7):2293–2305, November 1999.

[28] I. Kontoyiannis. Pointwise redundancy in lossy data compression and universal lossy data compression. *IEEE Trans. Inform. Theory*, 46(1):136–152, January 2000.

[29] Jr. Langdon, G.G. An introduction to arithmetic coding. *IBM J. Res. Develop.*, 28(2):135–149, 1984.

[30] T. Linder, G. Lugosi, and K. Zeger. Rates of convergence in the source coding theorem, in empirical quantizer design, and in universal lossy source coding. *IEEE Trans. Inform. Theory*, 40(6):1728–1740, 1994.

[31] T. Łuczak and W. Szpankowski. A suboptimal lossy data compression algorithm based on approximate pattern matching. *IEEE Trans. Inform. Theory*, 43(5):1439–1451, 1997.

[32] D.J.C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, New York, 2003.

[33] D.J.C. Mackay and R.M. Neal. Good codes based on very sparse matrices. In *Cryptography and Coding. 5th IMA Conference, number 1025 in Lecture Notes in Computer Science*, pages 100–111. Springer, 1995.

[34] Y. Mao and A. Banihashemi. Design of good LDPC codes using girth distribution. In *Int. Symp. Inform. Theory*, Sorrento, Italy, 2000.

[35] M.W. Marcellin and T.R. Fischer. Trellis coded quantization of memoryless and Gauss-Markov sources. *IEEE Trans. Comm.*, 38(1):82–93, 1990.

[36] E. Martinian and M. Wainwright. Low density codes achieve the rate-distortion bound. In *Proc. Data Compression Conf. – DCC 2006*, pages 153–162, Snowbird, UT, March 2006.

[37] Y. Matsunaga and H. Yamamoto. A coding theorem for lossy data compression by LDPC codes. *IEEE Trans. Inform. Theory*, 49(9):2225–2229, Sept. 2003.

[38] S. Miyake. Lossy data compression over Zq by LDPC code. In *Proc. of the IEEE International Symposium on Inform. Theory*, page 813, Seattle, WA, July 2006.

[39] H. Morita and K. Kobayashi. An extension of LZW coding algorithm to source coding subject to a fidelity criterion. In *4th Joint Swedish-Soviet Int. Workshop on Inform. Theory*, pages 105–109, Gotland, Sweden, 1989.

[40] J. Muramatsu and F. Kanaya. Distortion-complexity and rate-distortion function. *IEICE Trans. Fundamentals*, E77-A:1224–1229, 1994.

[41] R.M. Neuhoff, D.L. Gray and L.D. Davisson. Fixed rate universal block source coding with a fidelity criterion. *IEEE Trans. Inform. Theory*, 21(5):511–523, 1975.

[42] D. Ornstein and P.C. Shields. Universal almost sure data compression. *Ann. Probab.*, 18:441–452, 1990.

[43] R.C. Pasco. *Source Coding Algorithms for Fast Data Compression.* PhD thesis, Dept. of Electrical Engineering, Stanford, CA, USA, 1976.

[44] T. Richardson and R. Urbanke. *Modern Coding Theory*. Cambridge University Press, Cambridge, UK, 2008.

[45] J. Rissanen and I. Tabus. Rate-distortion without random codebooks. In *Workshop on Information Theory and Applications (ITA)*, UCSD, San Diego, CA, January 2006.

[46] J.J. Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM J. Res. Develop.*, 20(3):198–203, 1976.

[47] C.E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec.*, part 4:142–163, 1959. Reprinted in D. Slepian (ed.), *Key Papers in the Development of Information Theory*, IEEE Press, 1974.

[48] M. Sipser and D.A. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42:1710–1722, 1996.

[49] Y. Steinberg and M. Gutman. An algorithm for source coding subject to a fidelity criterion, based upon string matching. *IEEE Trans. Inform. Theory*, 39(3):877–886, 1993.

[50] R.J. van der Vleuten and J.H. Weber. Construction and evaluation of trellis-coded quantizers for memoryless sources. *IEEE Trans. Information Theory*, 41(3):853–859, 1995.

[51] A.J. Viterbi and J.K. Omura. Trellis encoding of memoryless discrete-time sources with a fidelity criterion. *IEEE Trans. Information Theory*, IT-20:325–332, 1974.

[52] M.J. Wainwright and E. Maneva. Lossy source encoding via message-passing and decimation over generalized codewords of LDGM codes. In *Proc. of the IEEE International Symposium on Inform. Theory*, pages 1493–1497, Adelaide, Australia, Sept. 2005.

[53] N. Wiberg, H.-A. Loeliger, and R. Koetter. Codes and iterative decoding on general graphs. *European Transactions in Telecommunication*, 6:513–525, 1995.

[54] A.D. Wyner and J. Ziv. Fixed data base version of the Lempel-Ziv data compression algorithm. *IEEE Trans. Inform. Theory*, 37(3):878–880, 1991.

[55] E.-H. Yang and J.C. Kieffer. Simple universal lossy data data compression schemes derived from the Lempel-Ziv algorithm. *IEEE Trans. Inform. Theory*, 42(1):239–245, 1996.

[56] E.-H. Yang and J.C. Kieffer. On the performance of data compression algorithms based upon string matching. *IEEE Trans. Inform. Theory*, 44(1):47–65, 1998.

[57] E.-H. Yang, Z. Zhang, and T. Berger. Fixed-slope universal lossy data compression. *IEEE Trans. Inform. Theory*, 43(5):1465–1476, 1997.

[58] R. Zamir and K. Rose. Natural type selection in adaptive lossy compression. *IEEE Trans. Inform. Theory*, 47(1):99–111, 2001.

[59] Z. Zhang and V.K. Wei. An on-line universal lossy data compression algorithm by continuous codebook refinement – Part I: Basic results. *IEEE Trans. Inform. Theory*, 42(3):803–821, 1996.

[60] Z. Zhang and E.-H. Yang. An on-line universal lossy data compression algorithm by continuous codebook refinement – Part II: Optimality for phi-mixing models. *IEEE Trans. Inform. Theory*, 42(3):822–836, 1996.

[61] J. Ziv. Coding of sources with unknown statistics – Part II: Distortion relative to a fidelity criterion. *IEEE Trans. Inform. Theory*, 18(3):389–394, 1972.

[62] J. Ziv. Coding theorems for individual sequences. *IEEE Trans. Inform. Theory*, 24(4):405–412, 1978.

[63] J. Ziv. Distortion-rate theory for individual sequences. *IEEE Trans. Inform. Theory*, 26(2):137–143, 1980.

[64] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inform. Theory*, 23(3):337–343, 1977.

[65] J. Ziv and A. Lempel. Compression of individual sequences by variable rate coding. *IEEE Trans. Inform. Theory*, 24(5):530–536, 1978.