RJ 10286 (declassified 6/26/03) (A0302-015) February 19, 2003
Computer Science

IBM Research Report

The Art of *Making* Errors: A Quadratic-time, Sequential, Adaptive Algorithm for Lossy Compression

Dharmendra Modha
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Art of *Making* Errors: A Quadratic-time, Sequential, Adaptive Algorithm for Lossy Compression

Dharmendra S. Modha
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
email: dmodha@almaden.ibm.com

March 6, 2003

Abstract

We propose a new algorithm for variable-rate lossy compression of memoryless sources at a fixed distortion. The algorithm uses approximate pattern matching and is modelled after the Lempel-Ziv (LZ78) algorithm. As the key new idea, we introduce a codeword selection principle, *Codelet Parsing*, to choose from various approximately matching codewords of possibly different lengths. To implement this selection, the algorithm examines the "empirical mutual information" between the source and the distorted sequences. The algorithm is sequential, adaptive, and can be implemented in quadratic-time in the length of the source sequence. For several memoryless Bernoulli sources with Hamming distortion, we experimentally demonstrate the surprising result that the empirical distributions of the distorted sequences are close to their respective "optimal reproduction distributions."

Keywords: Rate-distortion function, Bernoulli processes, memoryless sources, lossy coding, data compression, incremental parsing, Lempel-Ziv algorithm, approximate pattern matching, Hamming distance, fidelity criterion, induced channels, Blahut-Arimoto algorithm, side information

1 Introduction

1.1 Motivation

Shannon [1] introduced the idea of source coding with a fidelity criterion. We summarize the problem of lossy data compression at a fixed distortion level; for an extensive survey, see [2]. Suppose we are given a finite string $x_1^n \equiv x_1, x_2, \ldots, x_n$ of length n drawn from a finite source alphabet B. We would like to find a distorted or lossy version of x_1^n, say, $y_1^n = y_1, y_2, \ldots, y_n$, drawn from a finite reproduction alphabet $\tilde{B}$ such that the average single-letter distortion between the two strings is at most D (according to some bounded, non-negative distortion measure d) and that the lossy sequence y_1^n is highly compressible. We would like to minimize the per-symbol *rate* at which y_1^n must be transmitted subject to the *distortion* constraint. The central result of rate-distortion theory is that for source sequences generated by a stationary, ergodic stochastic process, asymptotically (as $n \to \infty$), the *rate-distortion* function is an achievable lower bound on the compression rate.

We first set some terminology. We say that a sequence of codes is *asymptotically optimal* for some source distribution if it achieves the rate-distortion bound as $n \to \infty$. We say that a sequence of codes is *universal* if it is asymptotically optimal for a class of sources without any *a priori*

knowledge of which specific source in the class generated the given source sequence that is to be compressed. Roughly, we say that a code is *sequential* or *online* if its delay in encoding m-th source letter is $o(m)$; see, [3]. We say that a sequential code is *adaptive* if no codebook (or other information) needs to be transmitted separately by the encoder to the decoder. In other words, an adaptive code builds its codebook on-the-fly in response to an observed source sequence. Moreover, both the encoder and decoder can keep updating their codebooks by the same rule. Finally, we are interested in *polynomial-time* algorithms with computational complexity of the form $O(n^k)$ for some (hopefully, small) k.

Equipped with the terminology, we can state that the central problem of lossy source coding is to find an universal (for stationary, ergodic sources), sequential, adaptive, and polynomial-time algorithm. The quest for such algorithms is important in theory as well in practice owing to broadband applications such as streaming multimedia [4], images, audio, cellular voice, and text.

When no distortion is desired, that is, $D = 0$, the lossy coding problem simplifies to the well researched problem of lossless data compression. Various well known algorithms for lossless coding are dynamic Huffman coding [5], adaptive arithmetic coding [6, 7], Lempel-Ziv algorithms [8, 9], locally adaptive schemes. [10, 11], and grammar codes [12]. These algorithms constitute universal (for stationary, ergodic sources), sequential, adaptive, and polynomial-time algorithms for lossless data compression.

While such algorithms exist when $D = 0$, so far, no algorithm attaining all the desiderata is known when $D > 0$. Indeed, prominent researchers have attested to this fact, for example, [2, p. 2709] noted that "All universal lossy coding schemes found to date lack the relative simplicity that imbues Lempel-Ziv coders and arithmetic coders with economic viability. ... This suggests it is unlikely that the "holy grail" of implementable universal lossy source coding will be discovered soon.". When speaking about Lempel-Ziv-type algorithms for lossy data compression, [13, p. 2054] noted that "It seems, however, that low computational complexity is achievable only at the expense of yielding a nonoptimal distortion." Also, [14] noted that "...it is our belief that a universal lossy source coding scheme with attractive computational complexity aspects will never be found." Finally, it was suggested that [15] "further investigations of suboptimal and practical heuristics for lossy compression are needed."

1.2 Our Contribution

We study a quadratic-time, sequential, and adaptive Lempel-Ziv-type algorithm for variable-rate lossy compression of memoryless sources. Our algorithm is in the spirit of the incremental parsing Lempel-Ziv (LZ78) algorithm [9] for lossless coding, and, when no distortion is desired, the algorithm simplifies to LZ78. As an important side benefit, the algorithm outputs a distorted sequence that can be naturally losslessly compressed using the LZ78 algorithm.

The algorithm *sequentially* parses the source sequence into phrases, say, *sourcelets*, and maps each sourcelet to a distorted phrase, say, a *codelet*, such that the per-letter distortion between the two phrases does not exceed the desired distortion. The algorithm *adaptively* maintains a codebook (a set of codewords[1]) which contains all one-letter extensions of previously emitted codelets; hence, it generates and continually grows the codebook on-the-fly in response to the observed source sequence without any *a priori* knowledge of the source. Both the encoder and decoder maintain the same codebook. The algorithm carries out a sequential procedure by iterating the following steps: (i) given the current codebook find the set of all codewords that match (at a given distortion) the

[1] We use the term *codelet* to denote a previously emitted distorted phrase and *codeword* to denote a phrase that is a one-letter extension of a codelet and is itself not a codelet (yet).

corresponding prefixes of the unparsed source sequence; and (ii) from all these matching codewords select one.

Typically, more than one approximately matching codewords of possibly different lengths will be found at each epoch. Multiple matches are a sign of underlying redundancy, and, hence, are a symptom of the fact that we are not operating near the rate-distortion curve. We focus on the following key question:

How to select between multiple approximately matching codewords at each epoch?

The chosen codeword becomes a part of the codebook, and, hence, affects future code rate. Also, the length of the chosen codeword affects the per-letter code rate in the current epoch. We would like to carefully choose the codeword that best balances between the per-letter code rate in the current epoch and the quality of the resulting codebook for future epochs. This suggests that lossy compression is not just an information-theoretic problem but a control-theoretic problem as well.

To further understand this choice, recall that for lossless compression LZ78 achieves universality for stationary, ergodic sources, since, asymptotically, all codewords are typical with respect to the source distribution. In lossy compression, to achieve asymptotically optimal compression, the codewords must be typical with respect to an[2] optimal reproduction distribution. In general, the source distribution need not be an optimal reproduction distribution, and, hence, using the source distribution for codeword selection leads to asymptotically sub-optimal compression, see, [16]. To further complicate the matters, we need to drive the first-order empirical distribution of the distorted sequence towards an optimal reproduction distribution but not the entire joint distribution of the entire distorted sequence towards the product of first-order optimal reproduction distributions. In fact, we need to drive the joint distribution of the entire distorted sequence so that it can be compressed close to the optimal. Achieving these two goals simultaneously is the fundamental difficulty and challenge of designing algorithms for lossy compression. For a related discussion in the context of channel coding, see [17].

Each of the approximately matching codeword, if selected, will extend the distorted sequence differently. We study the "induced empirical forward channel" that is implied by each such extended distorted sequence and the source sequence. Equivalently, we study the "empirical mutual information" between extended distorted sequences and the source sequence. To compute empirical mutual information between two sequences, we use lossless compression with side information as a subroutine. To the best our knowledge, empirical mutual information has not previously been used in the context of lossy coding. As our **main** contribution, we propose the following principle.

> Select the codeword that minimizes the (additional) per-symbol description complexity of the corresponding extended distorted sequence and the (additional) per-symbol empirical mutual information between the source sequence and the corresponding extended distorted sequence.

We think of such selection as a kind of "gold-washing" [18] or "natural-type selection" [19] to select good codewords, and refer to our selection mechanism as: *Codelet Parsing*. This has the desired effect of favoring the codewords that are typical with respect to an (*unknown*) optimal reproduction distribution and simultaneously generating a distorted sequence that has low description complexity. The effect of Codelet Parsing is that we do not always select the longest match, but rather select the longest match that best balances between two terms. Hence, Codelet Parsing is a type of "regularized pattern matching". The choice of one amongst many approximately matching codeword really amounts to selecting between multiple values and locations for intentionally making errors; and, hence, we are really concerned with the "art of *making* errors".

[2] In general, optimal reproduction distribution need not be unique.

At this stage of research, the algorithm appears difficult to theoretically analyze. As our **second** contribution, for several memoryless Bernoulli sources with Hamming distortion, we experimentally demonstrate that the empirical distribution of the distorted sequences is close to their corresponding optimal reproduction distributions. In other words, our algorithm that does not know the source statistics, empirically discovers an optimal reproduction distribution. Thus, our algorithm may be thought of as an adaptive, sample-path version of the Blahut-Arimoto algorithm [20, 21] that leads not only to an optimal reproduction distribution, but also to a good code. At the very least, our algorithm provides an upper bound on the rate-distortion function that can be computed using a sample-path.

On one hand, we empirically demonstrate that the greedy strategy of always selecting the longest match does not always favor codewords that are typical with respect to an optimal reproduction distribution. On the other hand, we empirically demonstrate that the strategy of selecting the match that minimizes empirical mutual information alone does not seem to favor longer codewords, and, hence, does not lead to a good rate. In contrast, Codelet Parsing seems to empirically discover an optimal reproduction distribution *and* seems to lead to a good code.

A preliminary version of Codelet Parsing based on a quite different estimator of the empirical mutual information between two sequences was announced in [22]. This paper is based on internal IBM Research Reports [23, 24].

1.3 Outline of the Paper

In Section 2, we describe various known algorithms and put our work in context. In Section 3, we present the necessary preliminaries. In Section 4, we present the Codelet Parsing algorithm for memoryless sources and discuss some its properties. In Section 5, we experimentally demonstrate the algorithm for several memoryless Bernoulli sources. Finally, in Section 6, we present conclusions.

2 Prior Work

We briefly and non-exhaustively review various known lossy coding schemes with a focus on algorithmic results. For references to earlier results on existence of universal lossy codes involving exponential-time constructions, see, Kieffer [25]. We will confine our discussion to finite (discrete) source and reproduction alphabets; for an extensive survey of results for real-valued sources, see [26].

Cheung and Wei [27] extended the move-to-front algorithm [10, 11] to lossy source coding. The algorithm adaptively builds its codebook from source words. Since, generally, the source distribution need not be an optimal reproduction distribution, the algorithm is known to be sub-optimal [16]. Later, Zhang and Wei [18] proposed an on-line lossy coding algorithm for the fixed-rate case that uses a "gold-washing" mechanism for promoting frequently used codewords (during a time interval of specified length) to permanent status while randomly generating new candidate codewords. Their algorithm is universal for memoryless sources. Later variants [28] have been shown to be universal for certain phi-mixing sources.

Another group of papers have focussed on lossy extensions of the Lempel-Ziv algorithm. The central idea is to use approximate string matching [29, 30] instead of exact string matching used in the Lempel-Ziv algorithms.

Morita and Kobayashi [31] extended the LZW algorithm, say, ELZW. There are several key differences between Codelet Parsing and ELZW. At every step, ELZW finds the longest source phrase such that the source phrase and all its prefixes match at least one codeword and its corresponding prefixes within the desired distortion. We do not require such strict sequentiality, and,

hence, can often find longer matches. Due to strong sequentiality, in certain cases, ELZW has the property that it transmits the first few letters of every phrase undistorted. Second, amongst multiple codeword matches, ELZW selects the one with the smallest distortion. Finally, like the LZW algorithm, after ELZW has found the longest matching source phrase and has selected a corresponding codeword, it adds the newly selected codeword extended by the next source letter to the codebook. As a result, each codeword is composed by sampling different source letters and by concatenating them. At least in the memoryless case, the end effect is that each codeword is now drawn from the source distribution. As a result, the algorithm is known to be sub-optimal for memoryless sources [16]. In comparison, we add all one-letter extensions of the selected codeword to the codebook. From these multiple codewords, we adaptively pick the codewords that prove to be useful. Hence, the codewords in our algorithm are not all drawn from the source distribution.

Constantinescu and Storer [32, 33] combined ideas from lossless Lempel-Ziv algorithms and vector quantization to design first practical implementations of lossy image compression based on approximate pattern matching. The problem of "selecting amongst multiple matches" mentioned above was termed the "Match Heuristic" in their work; see, also, Storer [34, p. 111].

Later, Steinberg and Gutman [35] and Luczak and Szpankowski [15] considered the fixed-database version of the Lempel-Ziv algorithm, and provided sub-optimal performance guarantees. Finally, Yang and Kieffer [16] established that all previous fixed-database extensions of the Lempel-Ziv algorithm are suboptimal; intuitively, the distribution generating the fixed database (the so-called training sequence) need not be an optimal reproduction distribution. Kontoyiannis [36] presented a scheme where multiple databases (each drawn according to a different reproduction distribution) are used at the encoder and must also be known to the decoder. When the reproduction alphabet is large, the number of training databases is unreasonably large. Since Codelet Parsing uses previously emitted codelets to construct a codebook, unlike the algorithms in [35, 15, 16, 36], it does not require training sequences that are *a priori* known to both the encoder and decoder.

Atallah et al. [37] considered a cubic-time, adaptive algorithm (PMIC) in the spirit of LZ77. Their algorithm is not sequential in the sense of [3], since its encoding delay grows faster than $o(n)$. Alzina et al. [38] combined ideas from [37] and [32, 33] to propose a 2D-PMIC algorithm that is more suited for 2D images. We note that while we have developed Codelet Parsing in the framework of LZ78, our codeword selection mechanism can be easily adapted to ELZW of [31] and LZ77-based PMIC of [37].

Continuing the quest for Lempel-Ziv-type lossy algorithms, Zamir and Rose [39] further studied the algorithm in [31]. From the multiple codewords that may match a source word, they suggest choosing one "at random". From a theoretical perspective, by assuming uniqueness, Zamir and Rose [19] proposed a natural type selection scheme for finding the type of the optimal reproduction distribution. Their procedure can be thought of as a stochastic simulation of the Blahut-Arimoto algorithm for computing the rate-distortion function. In later work, Kochman and Zamir [40] pointed out that the theoretical procedure in [19] is in itself not practical and demonstrated an application of natural-type selection to on-line codebook selection from a parametric class.

Along a different line, Yang and Kieffer [14] have proposed exponential-time Lempel-Ziv-type block codes that are universal (for stationary, ergodic sources and for individual sequences). In a related work, Yang and Zhang [41] presented fixed-slope universal lossy coding schemes that search for the reproduction sequence through a trellis in a fashion reminiscent of the Viterbi algorithm. While finding the optimal reproduction sequence still takes exponential-time, the trellis structure allows computationally efficient heuristic codes that are also sequential in nature. Recently, Zhou and Zhang [42] characterized the redundancy of trellis lossy codes for discrete memoryless sources.

3 Preliminaries

3.1 Basic Definitions

All logarithms are base 2, that is, $\log \equiv \log_2$. The set of natural numbers is written as $\mathbb{N}$.

Let λ denote the empty string. Let A denote a finite alphabet. Let $|A|$ denote the number of symbols in A. We will refer to elements of A as *symbols* or *letters*. For $m \in \mathbb{N}$, let $A^m = \{(a_1, a_2, \ldots, a_m) : a_i \in A, 1 \leq i \leq m\}$ denote the set of all sequences (strings, words, or phrases) of length m over A. By convention, $A^0 = \{\lambda\}$. Let $A^* = \cup_{m \geq 0} A^m$ denote the set of all sequences of finite length over A. For any string a of finite length, let $|a|$ denotes the length of a. Let A^∞ denote the set of one-sided infinite sequences of the form $(a_1, a_2, \ldots)$, where $a_i \in A, i \in \mathbb{N}$. For $i \leq j$, we let $a_i^j \equiv a_i, a_{i+1}, \ldots, a_j$. If $j < i$, then a_i^j denotes the empty string λ. Let $\circ$ denote the string concatenation operator.

We refer to a finite set of finite strings as a *dictionary*. A dictionary is termed *proper* if no string in the dictionary is a prefix of another. A dictionary is termed *complete* if every one-sided infinite sequence has a prefix in the dictionary.

Let B denote a *source* alphabet, and let $\tilde{B}$ denote a *reproduction* alphabet.

Assumption 3.1 *Alphabets B and $\tilde{B}$ are finite.*

We now define a *distortion measure* d that quantifies the loss of fidelity in representing a symbol in B using a symbol in $\tilde{B}$.

Assumption 3.2 *The function $d : B \to \tilde{B}$ is bounded, non-negative, and satisfies*

$$\max_{b \in B} \min_{\tilde{b} \in \tilde{B}} d(b, \tilde{b}) = 0. \tag{1}$$

For example, for binary alphabets $B = \tilde{B} = \{0, 1\}$, d may simply be the Hamming distance.

The distortion measure d is a single-letter measure in that it measures the loss of fidelity in representing a single symbol in B using a symbol in $\tilde{B}$. We now extend d to strings. For $m \in \mathbb{N}$, define the function $d_m : B^m \to \tilde{B}^m$ as $d_m(b_1^m, \tilde{b}_1^m) = \frac{1}{m} \sum_{i=1}^m d(b_i, \tilde{b}_i)$, where $b_1^m \in B^m$ and $\tilde{b}_1^m \in \tilde{B}^m$. For $m \in \mathbb{N}$, we say that a sequence $\tilde{b}_1^m \in \tilde{B}^m$ is a *ϵ-match*, $\epsilon \geq 0$, of a sequence $b_1^m \in B^m$ with respect to the single-letter distortion measure d_m, if

$$d_m(b_1^m, \tilde{b}_1^m) \leq \epsilon.$$

3.2 Rate-Distortion Theory

Let P denote a probability distribution function on B, that is, $P(x) \geq 0, x \in B$, and $\sum_{x \in B} P(x) = 1$.

Let $\mathcal{W}$ denote the set of all conditional distributions of the form $W(y|x)$, where $y \in \tilde{B}$ and $x \in B$. We refer to each element in $\mathcal{W}$ as a *forward channel*. Let D denote a desired distortion. The *rate-distortion* function is defined as

$$R(P, D) = \min_{W \in \mathcal{W} : d(P,W) \leq D} I(P, W),$$

where $I(P, W)$ denotes the *mutual information* between P and W, namely,

$$I(P, W) = \sum_{x \in B, y \in \tilde{B}} P(x) W(y|x) \log \frac{W(y|x)}{\sum_{x' \in B} P(x') W(y|x')}$$

and $d(P, W)$ denotes the average distortion induced by W, namely,

$$d(P, W) = \sum_{x \in B, y \in \tilde{B}} d(x, y) P(x) W(y|x).$$

The fundamental significance of rate-distortion function stems from the fact that it is an asymptotic lower bound for the compression rate of codes operating at the fixed distortion D. Let $W^\star$ denote a *optimal forward channel* that achieves the rate-distortion function. Define $J^\star = P \circ W^\star$ and $Q^\star$ as

$$\begin{aligned} J^\star(x, y) &= P(x) \cdot W^\star(y|x), \ x \in B \text{ and } y \in \tilde{B} \\ Q^\star(y) &= \sum_{x \in B} J^\star(x, y), y \in \tilde{B}. \end{aligned}$$

Distribution $Q^\star$ is known as an *optimal reproduction distribution.* Define

$$D^\star \equiv D^\star(P) = \inf\{D : R(P, D) = 0\}.$$

Example 3.1 *Let $B = \tilde{B} = \{0, 1\}$, and let d be the Hamming distortion. Let $P(1) = p$ and $P(0) = 1 - p$. Assume that $0 \leq p \leq 1/2$. In this case, it is known that*

$$D^\star = p \tag{2}$$

$$R(P, D) = \begin{cases} H(p) - H(D) & \text{for } 0 \leq D \leq D^\star, \\ 0 & D^\star < D, \end{cases} \tag{3}$$

$$Q^\star(1, D) \equiv Q^\star(1) = \begin{cases} \frac{p - D}{1 - 2D} & \text{for } 0 \leq D \leq D^\star, \\ \frac{D - D^\star}{1 - 2D^\star} & D^\star < D \leq 1/2. \end{cases} \tag{4}$$

For $D^\star < D \leq 1/2$, optimal reproduction distribution is not unique. For example, $Q^\star(1, D) = 0$ is also an optimal reproduction distribution. Different optimal reproduction distributions differ in the average distortion that they introduce. The choice in (4) corresponds to average distortion D, whereas the choice $Q^\star(1, D) = 0$ corresponds to average distortion $D^\star$.

3.3 The Problem

Practically, we are interested in the following problem. Given a source sequence $x_1^n \equiv x_1, x_2, \ldots, x_n \in B^n$ and target distortion D, we would like to find a *reproduced sequence* or *distorted sequence* $y_1^n \equiv y_1, y_2, \ldots, y_n \in \tilde{B}^n$ such that $d_n(x_1^n, y_1^n) \leq D$ and that y_1^n is highly compressible.

Assumption 3.3 (memorless sources) *The B-valued random variables $x_1, x_2, \ldots$ are independently and identically distributed with marginal distribution P.*

Under the above assumption, $R(P, D)$ is the lower bound on the per-symbol rate at which the sequence $y_1, y_2, \ldots$ can be compressed. Note that we do not assume that the source distribution P is known.

3.4 Compression using Side Information: Memoeyless Sources

The development of this section is strongly tailored to Assumption 3.3.

For $k \in \mathbb{N}$, let

$$u_1^k = u_1, u_2, \ldots, u_k \in B^k$$
$$v_1^k = v_1, v_2, \ldots, v_k \in \tilde{B}^k,$$

denote two sequences. Write $(uv)_i$ to mean the pair (u_i, v_i), and $(uv)_i^j$ to mean (u_i^j, v_i^j).

We define the *joint empirical distribution* of $(uv)_1^i$, $i \geq 0$, as:

$$\hat{J}^i(u, v; (uv)_1^i) \equiv \hat{J}^i(u, v) = \frac{1 + \sum_{j=1}^{i} \mathbb{I}(u_j = u, v_j = v)}{|B| \cdot |\tilde{B}| + i},$$

where $\mathbb{I}$ denotes the indicator function, $u \in B$, and $v \in \tilde{B}$. Observe that distribution $\hat{J}^i$ is a function only of the first i letters from each sequence. Also, observe that $\hat{J}^i$ is always positive.

For $\hat{J}^i$, $i \geq 0$, let us define the induced *marginal* and *conditional* distributions as:

$$\hat{P}^i(u; u_1^i) \equiv \hat{P}^i(u) = \sum_{v \in \tilde{B}} \hat{J}^i(u, v), u \in B, \tag{5}$$

$$\hat{Q}^i(v; v_1^i) \equiv \hat{Q}^i(v) = \sum_{u \in B} \hat{J}^i(u, v), v \in \tilde{B}, \tag{6}$$

$$\hat{W}^i(v|u; (uv)_1^i) \equiv \hat{W}^i(v|u) = \frac{\hat{J}^i(u, v)}{\hat{P}^i(u)}, u \in B, v \in \tilde{B}, \tag{7}$$

$$\hat{V}^i(u|v; (uv)_1^i) \equiv \hat{V}^i(u|v) = \frac{\hat{J}^i(u, v)}{\hat{Q}^i(v)}, u \in B, v \in \tilde{B}. \tag{8}$$

Observe that $\hat{P}^i(u)$ is a function only of u_1^i, and $\hat{Q}^i(u)$ is a function only of v_1^i. For $i \geq 0$, we refer to $\hat{W}^i$ as the *empirical forward channel* induced by the joint sequence $(uv)_1^i$.

With respect to the sequence of distributions $\hat{J}^0, \hat{J}^1, \hat{J}^2, \ldots$, we can losslessly compress the sequence $(uv)_1^k$ in an on-line fashion as follows. By using arithmetic coding, use the initial distribution $\hat{J}^0$ to compress (u_1, v_1), compute $\hat{J}^1$ and use it to compress (u_2, v_2), and so on. The number of bits required for such a code is:

$$L^{B\tilde{B}}((uv)_1^k) = \sum_{i=1}^{k} -\log \hat{J}^{i-1}(u_i, v_i). \tag{9}$$

In a similar fashion, we can losslessly compress the sequence u_1^k and v_1^k using the following code lengths, respectively:

$$L^B(u_1^k) = \sum_{i=1}^{k} -\log \hat{P}^{i-1}(u_i) \tag{10}$$

$$L^{\tilde{B}}(v_1^k) = \sum_{i=1}^{k} -\log \hat{Q}^{i-1}(v_i). \tag{11}$$

Now, consider a scenario where the sequence v_1^k is known to both the encoder and the decoder. By taking advantage of this *side information*, we would like to compress the sequence u_1^k; see, Figure 1. To accomplish this task, we start with the conditional distribution $\hat{V}^0(u|v)$ and use that

to encode u_1 given v_1, next, we estimate $\hat{V}^1(u|v)$ and use that to encode u_2 given v_2, and so on. By using arithmetic coding, the necessary code length is:

$$L^{B|\tilde{B}}(u_1^k|v_1^k) = \sum_{i=1}^{k} -\log \hat{V}^{i-1}(u_i|v_i). \tag{12}$$

Similarly, we can write the code length required to compress v_1^k using side information u_1^k as:

$$L^{\tilde{B}|B}(v_1^k|u_1^k) = \sum_{i=1}^{k} -\log \hat{W}^{i-1}(v_i|u_i). \tag{13}$$

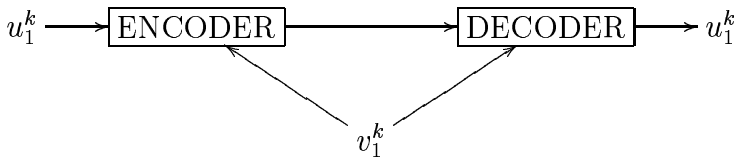

Figure 1: We would like to transmit the source sequence u_1^k from the ENCODER to the DECODER while taking advantage of the side information sequence v_1^k that is known at both the ends.

In this paper, we are only interested in the code lengths $L^B(u_1^k)$, $L^{\tilde{B}}(v_1^k)$, $L^{B|\tilde{B}}(u_1^k|v_1^k)$ and $L^{\tilde{B}|B}(v_1^k|u_1^k)$, and not in the actual codes. The following equalities can be easily seen by simple algebraic manipulations.

Proposition 3.1 *For any sequence $u_1^k \in B^k$ and any sequence $v_1^k \in \tilde{B}^k$, where $k \in \mathbb{N}$, we have that*

$$\begin{aligned} L^{B\tilde{B}}((uv)_1^k) &= L^{\tilde{B}}(v_1^k) + L^{B|\tilde{B}}(u_1^k|v_1^k) && (14)\\ &= L^{B}(u_1^k) + L^{\tilde{B}|B}(v_1^k|u_1^k). && (15) \end{aligned}$$

We define the *empirical mutual information* between the sequences u_1^k and v_1^k as

$$\hat{I}^{B\tilde{B}}(u_1^k, v_1^k) = L^{\tilde{B}}(v_1^k) - L^{\tilde{B}|B}(v_1^k|u_1^k) \stackrel{(a)}{=} L^{B}(u_1^k) - L^{B|\tilde{B}}(u_1^k|v_1^k), \tag{16}$$

where the equality (a) follows from Proposition 3.1. This quantity will play a fundamental role in our algorithm. For $u' \in B^*$ and $v' \in \tilde{B}^*$ such that $|u'| = |v'|$, define the following notation that will be used later:

$$\hat{I}^{B\tilde{B}}(u', v'|(uv)_1^k) = \hat{I}^{B\tilde{B}}(u_1^k \circ u', v_1^k \circ v') - \hat{I}^{B\tilde{B}}(u_1^k, v_1^k). \tag{17}$$

3.5 LZ78

We now describe LZ78 (also known as *incremental parsing*). For $k \in \mathbb{N}$, let $v_1^k \in \tilde{B}^k$ denote a source sequence to be compressed. The idea is to append the sequence on the left side with the empty phrase, and, then, sequentially parse the resulting sequence into phrases as

$$v_1^k = \lambda \circ v_1^{\tau(1)} \circ v_{\tau(1)+1}^{\tau(2)} \circ \ldots \circ v_{\tau(c_{\text{LZ}}(v_1^k)-1)+1}^{\tau(c_{\text{LZ}}(v_1^k))} \circ v_{\tau(c_{\text{LZ}}(v_1^k))+1}^{k},$$

where every new phrase (except perhaps the last phrase) is the shortest prefix of the unparsed sequence that is distinct from all previously parsed phrases and $c_{\text{LZ}}(v_1^k)$ denotes the number of

distinct phrases. Every phrase (including the last one) can be encoded by pointing to its longest proper prefix amongst all previously parsed phrases, and by describing its last symbol. Hence, we can write the Lempel-Ziv codeword length function as:

$$L_{\mathrm{LZ}}^{\tilde{B}}(u_1^k) = \sum_{i=1}^{c_{\mathrm{LZ}}(v_1^k)+1} \log(i|\tilde{B}|), \tag{18}$$

where $\log(i)$ is the number of bits required to identify the matching prefix and $\log|\tilde{B}|$ is the number of bits required to describe the last symbol.

For $v' \in \tilde{B}^*$, write

$$L_{\mathrm{LZ}}^{\tilde{B}}(v'|v_1^k) = L_{\mathrm{LZ}}^{\tilde{B}}(v_1^k \circ v') - L_{\mathrm{LZ}}^{\tilde{B}}(v_1^k). \tag{19}$$

4 Codelet Parsing

We are now ready to describe our lossy compression algorithm. At the block level, the structure of the code is described in Figure 2.

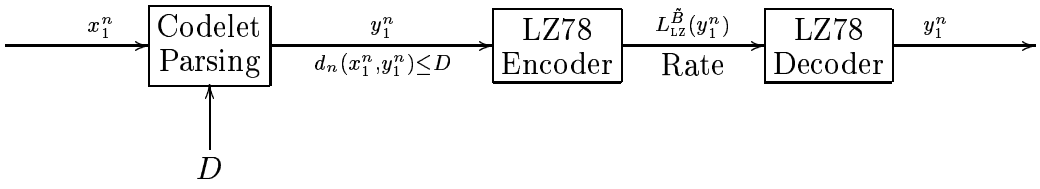

Figure 2: The Codelet Parsing algorithm, ϕ_n, consumes a source sequence $x_1^n \in B^n$ and outputs a distorted sequence $y_1^n \in \tilde{B}^n$ such that $d_n(x_1^n, y_1^n) \leq D$. The distorted sequence y_1^n is then compressed without loss and transmitted using LZ78 at the per-symbol rate of $L_{\mathrm{LZ}}^{\tilde{B}}(y_1^n)/n$.

Let D denote the target distortion. In Figure 3, we define a code ϕ_n, termed *Codelet Parsing*, that maps arbitrary source sequences in B^n to (a subset of) reproduction sequences in $\tilde{B}^n$ such that for every sequence $x_1^n \in B^n$

$$d_n(x_1, \phi_n(x_1^n, D)) \leq D. \tag{20}$$

The distorted sequence $y_1^n \equiv \phi_n(x_1^n, D)$ will then be losslessly compressed using LZ78 at the per-symbol rate of

$$\frac{L_{\mathrm{LZ}}^{\tilde{B}}(y_1^n)}{n}.$$

We now explain the workings of the code. The code parses a given source sequence into source phrases, termed as *sourcelets*, and maps each sourcelet to an equi-length distorted phrase, termed as a *codelet*, such that the average per-symbol distortion between the two phrases is no more than D. We can write a generic such parsing as follows. Append the source sequence on the left with the empty string, and, then sequentially parse the source sequence x_1^n into sourcelets as

$$x_1^n = \lambda \circ x_1^{\hat{\tau}(1)} \circ x_{\hat{\tau}(1)+1}^{\hat{\tau}(2)} \circ \ldots \circ x_{\hat{\tau}(\hat{c}(x_1^n)-1)+1}^{\hat{\tau}(\hat{c}(x_1^n))} \circ x_{\hat{\tau}(\hat{c}(x_1^n))+1}^{n},$$

where quantities $\hat{c}(x_1^n)$ and $\hat{\tau}(1), \hat{\tau}(2), \ldots, \hat{\tau}(\hat{c}(x_1^n))$ are defined in Figure 3. The code constructs the distorted sequence by mapping each sourcelet to a codelet as

$$y_1^n = \lambda \circ y_1^{\hat{\tau}(1)} \circ y_{\hat{\tau}(1)+1}^{\hat{\tau}(2)} \circ \ldots \circ y_{\hat{\tau}(\hat{c}(x_1^n)-1)+1}^{\hat{\tau}(\hat{c}(x_1^n))} \circ y_{\hat{\tau}(\hat{c}(x_1^n))+1}^{n}. \tag{22}$$

Code ϕ_n
Input: the source sequence x_1^n, $n \geq 1$, and the target distortion D.

By convention, $x_1^0 = \lambda$ and $y_1^0 = \lambda$. Set the codebook $S_0 = \{\lambda\}$.
Set $\hat{\tau}(0) = \hat{\tau}(-1) = 0$.
Set the initial distributions $\hat{Q}^0(y; y_1^0)$ and $\hat{W}^0(y|x; (xy)_1^0)$ as in (6) and (7), respectively.
Set the iteration index $i = 1$.
while (the source sequence is not exhausted, that is, $(\hat{\tau}(i-1) < n)$)

1. Construct the new codebook as

$$S_i = \left\{ S_{i+1} \setminus y_{\hat{\tau}(i-2)+1}^{\hat{\tau}(i-1)} \right\} \cup \left\{ y_{\hat{\tau}(i-2)+1}^{\hat{\tau}(i-1)} \circ \tilde{b} : \tilde{b} \in \tilde{B} \right\},$$

   that is, by deleting the chosen codeword from S_i and by adding all one-letter extensions of the chosen codeword.

2. Compute the set of codewords $C_i \subset S_i$ that D-match the corresponding equi-length prefix of $x_{\hat{\tau}(i-1)+1}^n$.

3. From this set of matching codewords, choose

$$y_{\hat{\tau}(i-1)+1}^{\hat{\tau}(i)} = \arg\min_{v \in C_i} \left[\underbrace{\frac{L_{\mathrm{LZ}}^{\tilde{B}}(v|y_1^{\hat{\tau}(i-1)})}{|v|}}_{I_1} + \underbrace{\frac{\hat{I}^{B\tilde{B}}(u, v|(xy)_1^{\hat{\tau}(i-1)})}{|v|}}_{I_2} \right], \tag{21}$$

   where $u = x_{\hat{\tau}(i-1)+1}^{\hat{\tau}(i-1)+|v|}$, $L_{\mathrm{LZ}}^{\tilde{B}}$ is as in (19), and $\hat{I}^{B\tilde{B}}$ is as in (17). Output the codelet $y_{\hat{\tau}(i-1)+1}^{\hat{\tau}(i)}$. Note that $\hat{\tau}(i)$ is also implicitly set in (21).

4. Update the distributions: $\hat{Q}^{\hat{\tau}(i)}(v; y_1^{\hat{\tau}(i)})$ and $\hat{W}^{\hat{\tau}(i)}(v|u; (uv)_1^{\hat{\tau}(i)})$ as in (6) and (7), respectively.

5. Advance the iteration index $i = i + 1$.

endwhile
Write $\hat{c}(x_1^n) = \hat{\tau}(i-2)$.

Output: The codelets $y_1^{\hat{\tau}(1)}$, $y_{\hat{\tau}(1)+1}^{\hat{\tau}(2)}$, $\ldots$, $y_{\hat{\tau}(\hat{c}(x_1^n)-1)+1}^{\hat{\tau}(\hat{c}(x_1^n))}$, $y_{\hat{\tau}(\hat{c}(x_1^n))+1}^{n}$.

Figure 3: The Codelet Parsing Algorithm.

We think of the code as going through epochs or iterations. At epoch 0, the code reads the sourcelet λ and emits the codelet λ, at epoch 1, the code reads $x_1^{\hat{\tau}(1)}$ and emits $y_1^{\hat{\tau}(1)}$, and so forth and so on. For convenience, we write $\hat{\tau}(-1) = \hat{\tau}(0) = 0$.

We introduce the notion of a *codebook* or a *dictionary*, namely, a set of *codewords*. Intuitively, a codeword is a one-letter extension of a previously emitted codelet and is itself not yet a codelet. At epoch i, $i \geq 1$, let S_i' denote the set of all previously emitted codelets:

$$S_i' = \left\{\lambda, y_1^{\hat{\tau}(1)}, y_{\hat{\tau}(1)+1}^{\hat{\tau}(2)}, \ldots, y_{\hat{\tau}(i-2)+1}^{\hat{\tau}(i-1)}\right\}.$$

Now, we will define the codebook

$$S_i = \left\{z \circ \tilde{b} : z \in S_i' \text{ and } \tilde{b} \in \tilde{B}\right\} \setminus S_i'$$

that is obtained by taking all one-letter extensions of all previous codelets and by deleting previously emitted codelets. Note that the words *codelet* and *codeword* are used for different purposes. We use the term *codelet* to denote a previously emitted distorted phrase and *codeword* to denote a phrase that is a one-letter extension of a codelet and is itself not a codelet (yet). In Step 1, the code updates the codebook according to the codelet chosen in the previous step. The codebook S_i can be organized in a tree whose internal nodes constitute all previously emitted codelets and whose leaf nodes constitute all codewords that are available for future use. The set S_i is a complete and proper dictionary over $\tilde{B}$.

In Step 2 of every epoch, the code computes a set of codewords C_i from amongst all codewords in S_i that D-match the corresponding equi-length prefix of $x_{\hat{\tau}(i-1)+1}^n$. Since S_i is a complete and proper dictionary over $\tilde{B}$, it now follows from Assumption 3.2 that C_i must contain at least one phrase corresponding to a string in S_i that matches an equi-length prefix of $x_{\hat{\tau}(i-1)+1}^n$ without any distortion. Every element of C_i is a D-match of the corresponding sourcelet. Hence, trivially, we have that (20) is satisfied.

Note that C_i may contain codewords of different lengths. By definition of the codebook, every codeword in S_i, and, hence, in C_i, is of the form $z \circ \tilde{b}$, $z \in \tilde{B}^*$ and $\tilde{b} \in \tilde{B}$, such that z is one of the previous codelets. Hence, any choice from C_i will lead to a new codelet $y_{\hat{\tau}(i-1)+1}^{\hat{\tau}(i)}$ that is a shortest phrase that is not one of the previous codelets. In other words, for any choice from C_i, the parsing in (22) is the incremental parsing of y_1^n according to LZ78. Hence, the distorted sequence y_1^n can be naturally losslessly compressed using LZ78 (see Figure 2). Furthermore, as soon as a new codelet is computed in Step 2, it can be immediately sent. Thus, the code ϕ_n has low delay.

Typically, C_i will contain more than one approximate match. The main question is how to resolve this ambiguity. Excess matches are a sign of underlying redundancy. Amongst many matching codewords in C_i, in Step 3, the code carefully selects one. This step is the heart of the code. Selecting one of the longest codeword from C_i will minimize the instantaneous rate at which that selected codelet can be compressed using LZ78. However, note that the selected codeword will affect the codebook for future encoding (see Step 1), and, hence, will affect the future code rate. Thus, a good code should strive to balance between the code rate in the current step versus the quality of the resulting codebook for the future. Under Assumption 3.3, we would like to drive the distribution of the selected codeword towards an optimal reproduction distribution $Q^\star$. The choice in (21) is an attempt to achieve a fine balance between the code rate in the current step (see term I_1) and the quality of the resulting codebook for the future (see term I_2). The term I_1 indicates that we will like to minimize the per-letter code rate in compressing the distorted sequence. The term I_2 attempts to minimize the per-symbol empirical mutual information between the source and the distorted sequence. By minimizing this term, we hope to drive the symbol-wise (first-order

order) empirical distribution of the distorted sequence towards $Q^\star$. Hence, the term I_2 serves to improve the quality of the codebook for future. Observe (21) will not always choose the longest match that simply minimizes I_1. It will choose the longest match that minimizes the sum of I_1 and I_2. Hence, we think of the term I_2 as "regularizing" the codeword choice, and think of (21) as "regularized pattern matching."

Different codewords in C_i correspond to different error values and locations, and, hence, poetically, choosing amongst them can now be thought of as the art of making errors. The process of distilling the best codeword from amongst many is used to find the induced "channel that is just right for the source and allowed distortion level" [1] and to simultaneously find a distorted sequence that is highly compressible. We view the choice of the codeword as a form of control with which to affect the future code rate.

In Step 5, the code updates the distributions $\hat{Q}^{\hat{\tau}(i)}(v; y_1^{\hat{\tau}(i)})$ and $\hat{W}^{\hat{\tau}(i)}(v|u; (uv)_1^{\hat{\tau}(i)})$ and as in (6) and (7), respectively. These distributions are used to compute $L^{\tilde{B}}$ and $L^{\tilde{B}|B}$, respectively, which are then used to compute $\hat{I}^{B\tilde{B}}$ in (21).

Our description of the algorithm is now complete. We now discuss some of its properties.

Remark 4.1 (Goal of Codelet Parsing) The algorithm is trying to drive the quantities $L_{\mathrm{LZ}}^{\tilde{B}}(y_1^n)$ and $\hat{I}^{B\tilde{B}}(x_1^n, y_1^n)$ to $R(D,P)$ and to drive the marginal first-order distribution $\hat{Q}^n$ to $Q^\star$. The quantity $\hat{I}^{B\tilde{B}}(x_1^n, y_1^n)$ can be thought of as an upper bound on the rate-distortion function that is derived purely from a sample-path without knowledge of the source distribution P. As we will demonstrate in the next section, $\hat{I}^{B\tilde{B}}(x_1^n, y_1^n)$ seems to converge to $R(P,D)$ fairly quickly.

Remark 4.2 (Subtlety of Codelet Parsing) Observe that in (21) we minimize a sum of two terms I_1 and I_2. The term I_2 is intended to drive the empirical first-order distribution of the distorted sequence y_1^n towards an optimal reproduction distribution $Q^\star$. This does not mean that we are trying to drive the joint distribution of the entire distorted sequence y_1^n towards the product distribution $Q^\star \times Q^\star \times \ldots Q^\star$. Quite the contrary, we are trying to drive the distribution of the entire distorted sequence towards a distribution that minimizes $L_{\mathrm{LZ}}^{\tilde{B}}(y_1^n)$. This is achieved by the term I_1. To put it another way, the code corresponding to code length $L^{\tilde{B}}(y_1^n)$ in (11) is, in general, not a good code for y_1^n, but the Lempel-Ziv code is a good code for y_1^n. Hence, the fundamental idea of Codelet Parsing is to try to kill (achieve) two birds (distributions) with one stone (codeword choice). To illustrate this subtlety with a striking example, consider the sequence S1 described in Section 5. This is a sequence of independent bits that are identically distributed with $P(1) = P(0) = 1/2$. In this case, when $D = 1/2$, Codelet Parsing algorithm drives the empirical first-order distribution of the distorted sequence towards $Q^\star(1) = Q^\star(0) = 1/2$. This does not mean, of course, that the distorted sequence is drawn i.i.d. with $Q^\star$. The Lempel-Ziv complexity of the distorted sequence is 0.0192579 bits per input symbol which is essentially zero, and, hence, not equal to the entropy of $Q^\star$ which is one.

Remark 4.3 (Two Variants) Observe that the term I_2 which represents the additional empirical mutual information can be negative, and, hence, a simple variant of Codelet Parsing is to replace I_2 in (21) by $|I_2|$. As another variant, instead of (21), we can use the following choice leading to a slightly different algorithm in the same spirit.

$$y_{\hat{\tau}(i-1)+1}^{\hat{\tau}(i)} = \arg\min_{v \in C_i} \left[\underbrace{\frac{L_{\mathrm{LZ}}^{\tilde{B}}(y_1^{\hat{\tau}(i-1)} \circ v)}{\hat{\tau}(i-1) + |v|}}_{I_3} + \underbrace{\frac{\hat{I}^{B\tilde{B}}(x_1^{\hat{\tau}(i-1)} \circ u, y_1^{\hat{\tau}(i-1)} \circ v)}{\hat{\tau}(i-1) + |v|}}_{I_4} \right], \tag{23}$$

where $u = x_{\hat{\tau}(i-1)+1}^{\hat{\tau}(i-1)+|v|}$, $L_{\text{LZ}}^{\tilde{B}}$ is as in (18), and $\hat{I}^{B\tilde{B}}$ is as in (16). In empirical studies, both of the above algorithms produce results that are qualitatively similar to those exhibited in the next section.

Remark 4.4 (Sequential) When computing the set C_i, observe that the amount of look-ahead from $x_{\hat{\tau}(i-1)}$ is exactly equal to the length of the longest codeword in S_i, and, hence, cannot be larger that $O(\sqrt{\hat{\tau}(i-1)})$. Hence, the total encoding delay in encoding t-th source letter is $o(t)$. Hence, our code is *sequential* with vanishing look-ahead [3]. Furthermore, note that the length n of the source sequence is not used anywhere in the algorithm except to determine when to stop.

Remark 4.5 (Adaptive) The code is *adaptive* in the sense that it does not require any *a priori* knowledge of the source distribution P. It learns the source distribution online. Even more so, it seems (see Section 5) to learn an optimal reproduction distribution. Also, no *a priori* information such as common randomness or common databases need to be transmitted from the encoder to the decoder.

Remark 4.6 (Quadratic-time) From an implementation perspective, finding the set C_i is reasonably easy. Organize the codebook S_i as a tree. Traverse the tree in a depth-first fashion and at every leaf compare the codeword corresponding to the leaf to the source phrase of the same length. Add any D-matching codeword to the set C_i. Total amount of computation required is proportional to the length of the string processed so far, and, hence, the total overall computation is at most quadratic in the worst case. A finer argument which takes into account the average phrase lengths can show that the total overall computation is actually $O(n^2/\log n)$. Also, using a similar depth-first strategy, (21) can also be computed in computation proportional to the length of the string processed so far. Hence, the total computational cost of the code is at most *quadratic-time*. Generically, the fewer the previous y-phrases, the faster the algorithm, and, hence, for larger distortions (when there are generally fewer y-phrases) the algorithm is generally faster than for smaller distortions.

5 Empirical Results

5.1 The Sequences

We generated four binary sequences S1, S2, S3, and S4 as follows.

Sequence	Length	fractions of ones
S1	4194304	0.499588
S2	2398065	0.374379
S3	2797618	0.249762
S4	2398108	0.124777

The sequence S1 was downloaded from [43] that generates random numbers using atmospheric noise. As can be seen above this sequence possesses nearly even number of 1s and 0s. The sequence S2 was generated by making a one-pass through S1 and making following substitutions:

$$\{0, 100\} \to 0, \{101, 110, 111\} \to 1.$$

The idea was to generate a sequence with distribution of 1s to be roughly 0.375. Similarly, the sequence S3 was generated by making a one-pass through S1 and making following substitutions:

$$\{0, 10\} \to 0, 11 \to 1.$$

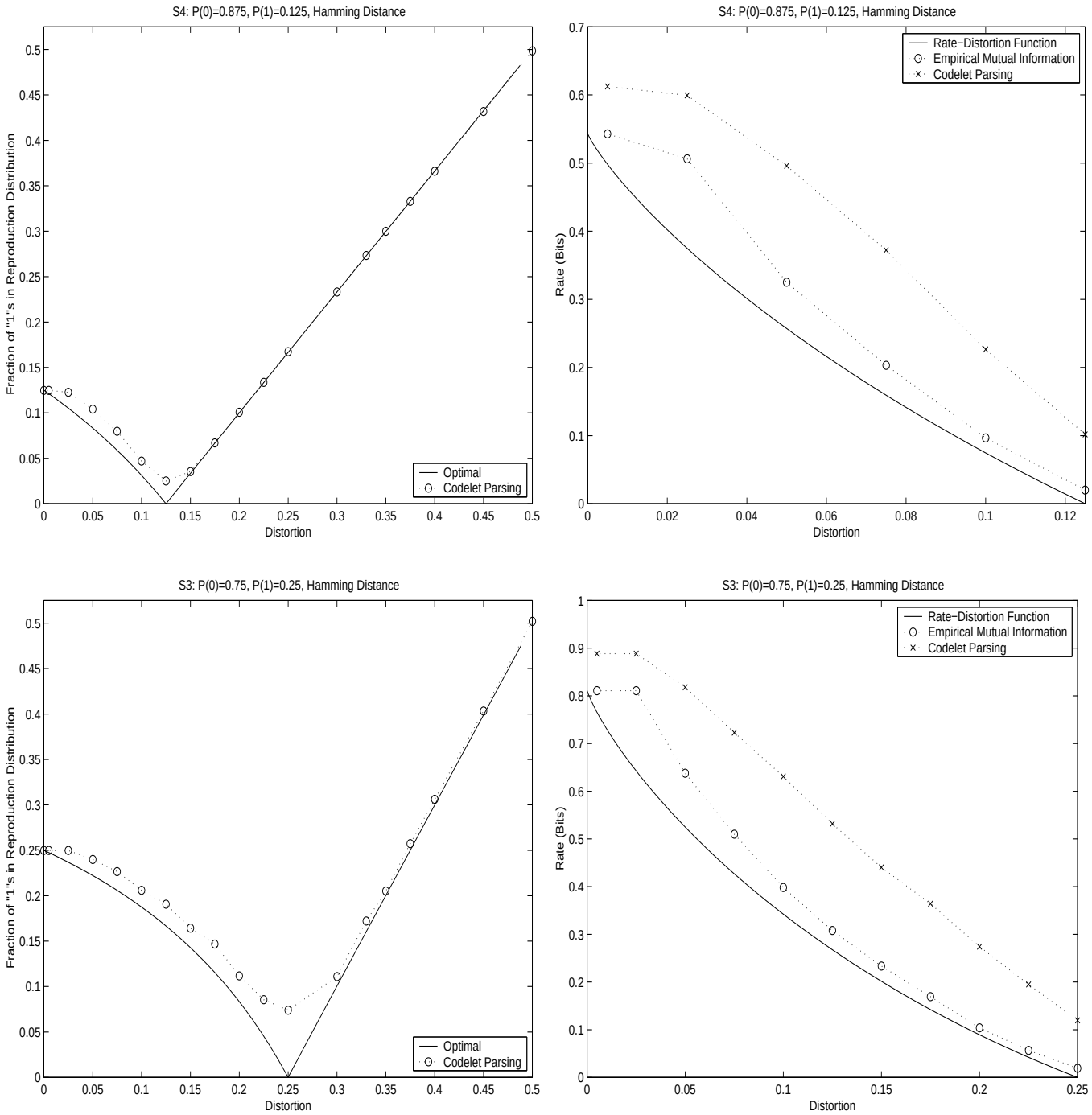

Figure 4: For the sequence S4 (resp. S3), in the top (resp. bottom) left plot, for various distortions, we compare the optimal reproduction distribution in (4) to the empirical distribution of the distorted sequence produced by Codelet Parsing. Similarly, for the sequence S4 (resp. S3), in the top (resp. bottom) right plot, for various distortions, we compare the compression rate of Codelet Parsing to the rate-distortion function. We also plot the empirical mutual information between the source sequence and the distorted sequence.

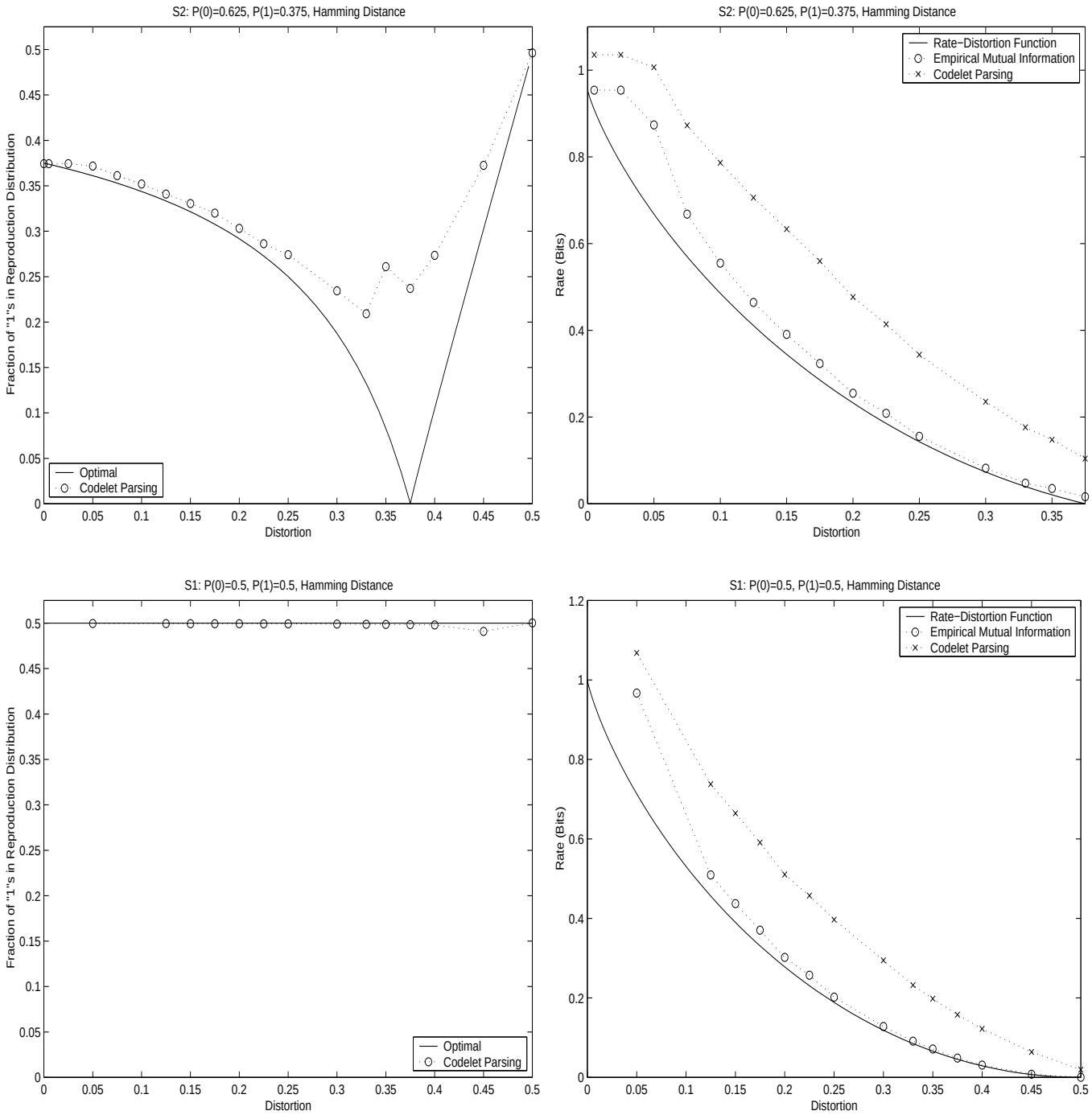

Figure 5: For the sequence S2 (resp. S1), in the top (resp. bottom) left plot, for various distortions, we compare the optimal reproduction distribution in (4) to the empirical distribution of the distorted sequence produced by Codelet Parsing. Similarly, for the sequence S2 (resp. S1), in the top (resp. bottom) right plot, for various distortions, we compare the compression rate of Codelet Parsing to the rate-distortion function. We also plot the empirical mutual information between the source sequence and the distorted sequence.

The idea was to generate a sequence with distribution of 1s to be roughly 0.25. Finally, the sequence S4 was generated by making a one-pass through S1 and making following substitutions:

$$\{0, 10, 110\} \to 0, 111 \to 1.$$

The idea was to generate a sequence with distribution of 1s to be roughly 0.125.

5.2 Results

In this subsection, we shall restrict attention to Hamming distance as the distortion measure. The reader may want to recall Example 3.1.

In the top left plot of Figure 4, for the sequence S4 and for various distortions, we compare the optimal reproduction distribution in (4) to the empirical distribution of the distorted sequence produced by Codelet Parsing. It can be seen that the empirical distribution of the distorted sequence produced by Codelet Parsing seems to be quite close to the optimal. This is a quite surprising result that demonstrates that Codelet Parsing seems to empirically discover the optimal reproduction distribution. Furthermore, in the region $D^\star < D \leq 1/2$, optimal reproduction distribution is not unique. Amongst these different distributions, Codelet Parsing seems to prefer the one with the largest distortion, namely, D.

Similarly, in the top right plot of Figure 4, for the sequence S4 and for various distortions, we compare the compression rate of Codelet Parsing to the rate-distortion function. We also plot the empirical mutual information between the source sequence and the distorted sequence. It can be seen that the empirical mutual information seems to be close to the rate-distortion function. This once again demonstrates that Codelet Parsing seems to empirically discover the optimal reproduction distribution. However, the actual compression rate seems to be far from the rate-distortion function. This continues to hold even for $D = 0$. But, for $D = 0$, Codelet Parsing collapses to LZ78 which is known to be asymptotically optimal. For $D = 0$, this gap is a result of the extremely slow redundancy rate of LZ78, see, for example, [44]. We expect that Codelet Parsing also has a similar redundancy rate, and, hence, the gap between the optimal and the actual rates.

For similar results concerning sequences S1, S2, and S3, see Figures 4 and 5. It can be seen from all the left hand plots in Figures 4 and 5 that the probability of one according to the optimal reproduction distribution as a function of the distortion is discontinuous at $D^\star$. In all the plots, this discontinuity seems to be the hardest operating point for Codelet Parsing.

It can be seen from the top right-hand plot in Figure 5 that, for sequence S2, empirical mutual information is fairly close to the rate-distortion function at high distortions, but begins to veer away from the rate-distortion function at small distortions, and once again steers back to the rate-distortion function at zero distortion. Essentially, the same phenomenon also holds for other sequences. To explain this behavior, in Figure 6, we plot the actual distortion achieved by Codelet Parsing versus the target distortion. It can be seen that for small distortions Codelet Parsing does not make enough errors. This is an artifact of pattern matching and can be explained as follows. We require that the average per-letter distortion between every sourcelet and the corresponding codelet be less that the target distortion. But, when the target distortion is very small, even to make one error, we must have fairly large phrases. For example, suppose that $D = 0.025$. This means that for all sourcelets of length less than $1/D = 40$, the algorithm cannot make even a single error. For all such small sourcelets, the empirical mutual information is high, and, hence, the observed behavior. However, asymptotically, this transitory, start-up effect should vanish.

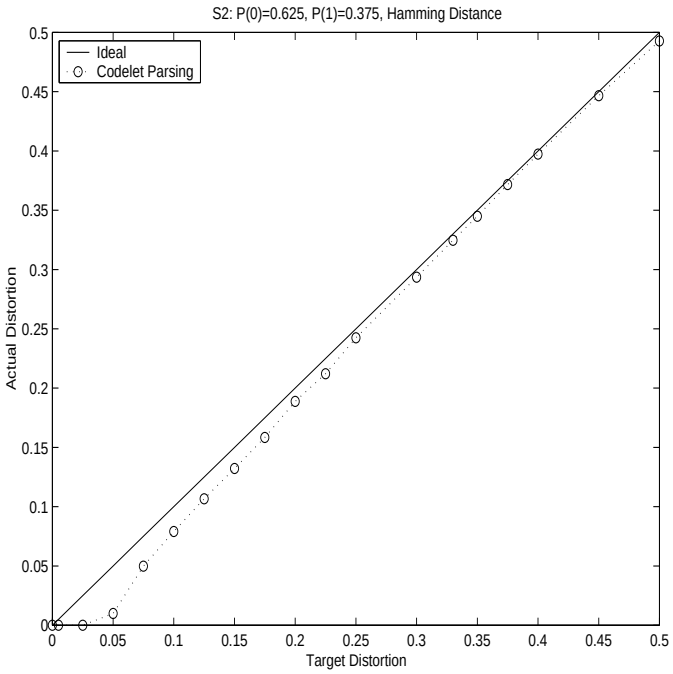

Figure 6: For the sequence S2 and with Hamming distortion, we plot the actual distortion achieved by Codelet Parsing versus the target distortion. It can be seen that for small distortions the actual distortion is less than the specified target distortion.

5.3 Greedy is Not Enough

The essence of Codelet Parsing is in the choice of the codeword in Step 3 of Figure 3. A trivial, greedy way to make this choice is to select one of the longest codewords from the set C_i. Ties are broken by selecting the most recent match and further ties are broken by selecting the lowest distortion match. This algorithm was already known to Storer in 1988 [34, p. 112]. In essence, this algorithm amounts to ignoring the term I_2 in (21). We now create an artificial example to illustrate that this greedy strategy is unlikely to be universal. Let the source and reproduction alphabets be binary. Define the distortion measure d as:

$$d(0,0) = d(1,1) = 0, d(0,1) = 1, d(1,0) = 5.$$

For this asymmetric distortion measure, we consider a source with $P(0) = 0.75$ and $P(1) = 0.25$. We chose the target distortion to be $D = D^\star = 0.75$. As can be seen from Figures 4 and 5, due to the discontinuity at $D^\star$, this operating point is a particularly hard spot. The Blahut-Arimoto algorithm (and simple common sense) indicates that $Q^\star(0) = 0$ and $Q^\star(1) = 1$ is the optimal reproduction distribution. For the sequence S3, the fraction of zeroes in the distorted sequence generated by the greedy algorithm is 0.479145, whereas the fraction of zeroes in the distorted sequence generated by the Codelet Parsing algorithm is 0.123059. To further illustrate this difference, in Figure 7, we plot the length of the LZ78 phrases (corresponding to the two distorted sequences) versus their empirical distribution (or "type") for both the algorithms. It can be seen that Codelet Parsing is quite focussed, while greedy seems to be attempting to generate phrases of many different types. The compression rate (per source bit) of the greedy algorithm is 0.1154, whereas that of the Codelet Parsing is 0.0559.

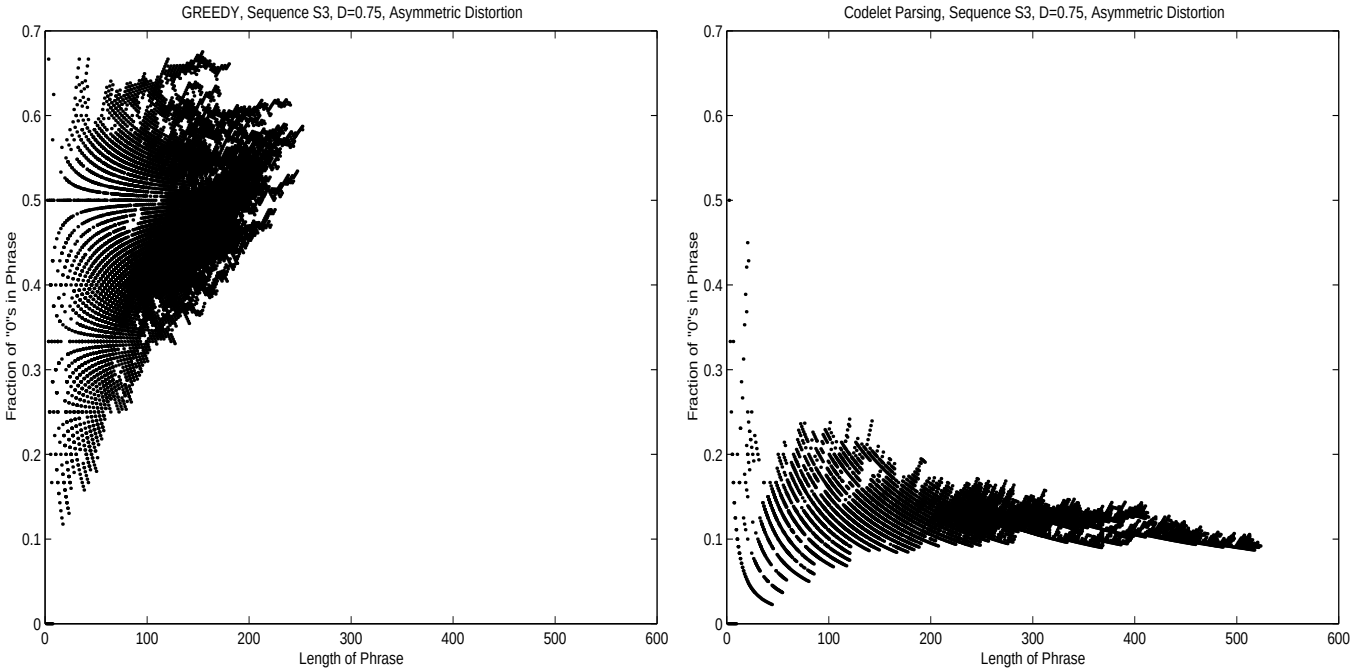

Figure 7: For the sequence S3, with asymmetric distortion and with $D = 0.75$, we ran the greedy algorithm to produce a distorted sequence. We took all LZ78 phrases of this distorted sequence. In the left plot, we plot length of each phrase versus the fraction of zeroes in each phrase. Similarly, for the sequence S3, with asymmetric distortion, and with $D = 0.75$, we ran the Codelet Parsing algorithm to produce a distorted sequence. We took all LZ78 phrases of this distorted sequence. In the right plot, we plot length of each phrase versus the fraction of zeroes in each phrase.

5.4 Empirical Mutual Information is Not Enough

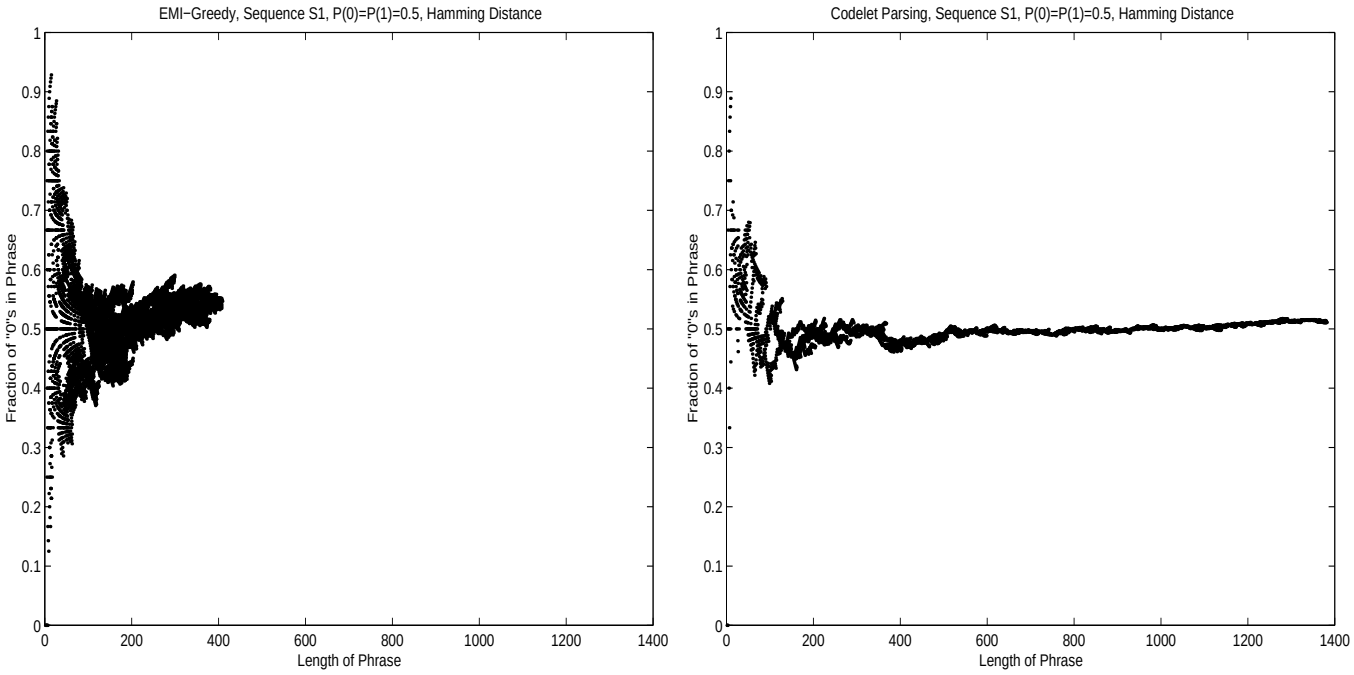

Figure 8: For the sequence S1, with Hamming distortion and with $D = 0.5$, we ran the EMI-Greedy algorithm to produce a distorted sequence. We took all LZ78 phrases of this distorted sequence. In the left plot, we plot length of each phrase versus the fraction of zeroes in each phrase. Similarly, for the sequence S1, with Hamming distortion and with $D = 0.5$, we ran the Codelet Parsing algorithm to produce a distorted sequence. We took all LZ78 phrases of this distorted sequence. In the right plot, we plot length of each phrase versus the fraction of zeroes in each phrase.

Let us consider another greedy algorithm that leaves out the term I_1 in (21), and considers only I_2. In words, the algorithm selects the codeword that minimizes additional per-letter empirical mutual information between the extended distorted sequence and the corresponding source sequence. We refer to this algorithm as EMI-Greedy (for empirical mutual information). We now illustrate via an example that this strategy is unlikely to always lead to good codes. Let the source and reproduction alphabets be binary. Let us consider Hamming distance as distortion. We consider a source with $P(0) = P(1) = 0.5$. We chose the target distortion to be $D = D^\star = 0.5$. In this case, $Q^\star(0) = Q^\star(1) = 0.5$ is the optimal reproduction distribution. For the sequence S1, the fraction of zeroes in the distorted sequence generated by EMI-Greedy and Codelet Parsing is, respectively, 0.5236 and 0.4997. Hence, both algorithms seem to discover the optimal reproduction distribution. However, the compression rate (per source bit) of EMI-Greedy and Codelet Parsing is, respectively, 0.06199 and 0.01926. Hence, Codelet Parsing leads to a better code. The number of phrases in the LZ78 parsing of the distorted sequences generated by EMI-Greedy and Codelet Parsing was, respectively, 18891 and 6597. To further accentuate the difference between the two algorithms, in Figure 8, we plot the length of the LZ78 phrases (corresponding to the two distorted sequences) versus their empirical distribution (or "type") for both the algorithms. It can be seen that Codelet Parsing generates longer and fewer phrases, while EMI-Greedy generates shorter and numerous phrases.

6 Conclusions

We have presented a quadratic-time, adaptive, sequential algorithm for variable-rate lossy compression of memoryless sources at a fixed distortion. As a key new idea, our algorithm carefully selects between multiple matching codewords. The process of distilling the best codeword from amongst many is used to find the induced "channel that is just right for the source and allowed distortion level" [1] and to simultaneously find a distorted sequence that is highly compressible. We have empirically demonstrated that, for several sources, the algorithm seems to discover the optimal reproduction distribution, and, hence, for memoryless sources, we are tempted to conjecture that our algorithm or a suitable variant is universal.

Acknowledgements. I am grateful to Corneliu Constantinescu, Daniela De Farias, Robert Hecht-Nielsen, Brian Marcus, Elias Masry, and Nimrod Megiddo for valuable discussions. I am also grateful to Moidin Mohiuddin for his encouragement during this work.

References

[1] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," in *Institute of Radio Engineers, Int. Convention Record*, vol. 7, pp. 142–163, 1959.

[2] T. Berger and J. D. Gibson, "Lossy source coding," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2693–2723, 1998.

[3] J. C. Kieffer and E.-H. Yang, "Sequential codes, lossless compression of individual sequences, and kolmogorov complexity," *IEEE Trans. Inform. Theory*, vol. 42, no. 1, pp. 29–39, 1996.

[4] G. J. Conkin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippmann, and Y. A. Reznik, "Video coding for streaming media delivery on the internet," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 269–281, 2001.

[5] R. G. Gallager, "Variation on a theme by Huffman," *IEEE Trans. Inform. Theory*, vol. 24, pp. 668–674, 1978.

[6] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM J. Res. Dev.*, vol. 20, no. 3, pp. 198–203, 1976.

[7] J. Rissanen and G. G. Langdon Jr., "Arithmetic coding," *IBM J. Res. Dev.*, vol. 23, no. 2, pp. 149–162, 1979.

[8] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[9] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[10] P. Elias, "Interval and recency rank source coding: Two on-line adaptive variable-length schemes," *IEEE Trans. Inform. Theory*, vol. 33, no. 3, pp. 3–10, 1987.

[11] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, "A locally adaptive data compression scheme," *Comm. ACM*, vol. 29, no. 4, pp. 320–330, 1986.

[12] J. C. Kieffer and E.-H. Yang, "Grammar-based codes: A new class of universal lossless source codes," *IEEE Trans. Inform. Theory*, vol. 46, pp. 737–754, 2000.

[13] A. D. Wyner, J. Ziv, and A. J. Wyner, "On the role of pattern matching in information theory," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2045–2056, 1998.

[14] E.-H. Yang and J. C. Kieffer, "Simple universal lossy data compression schemes derived from the Lempel-Ziv algorithm," *IEEE Trans. Inform. Theory*, vol. 42, no. 1, pp. 239–245, 1996.

[15] T. Luczak and W. Szpankowski, "A suboptimal lossy data compression based on approximate pattern matching," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1439–1451, 1997.

[16] E.-H. Yang and J. C. Kieffer, "On the performance of data compression algorithms based upon string matching," *IEEE Trans. Inform. Theory*, vol. 44, pp. 47–65, 1998.

[17] S. Shamai and S. Verdú, "The empirical distribution of good codes," *IEEE Trans. Inform. Theory*, vol. 43, pp. 836–846, May 1997.

[18] Z. Zhang and V. K. Wei, "An on-line universal lossy data compression algorithm via continuous codebook refinement–part i: Basic results," *IEEE Trans. Inform. Theory*, vol. 42, no. 3, pp. 803–821, 1996.

[19] R. Zamir and K. Rose, "Natural type selection in adaptive lossy compression," *IEEE Trans. Inform. Theory*, vol. 47, no. 1, pp. 99–111, 2001.

[20] R. E. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Trans. Inform. Theory*, vol. 18, pp. 460–473, July 1972.

[21] S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *IEEE Trans. Inform. Theory*, vol. 18, pp. 14–20, January 1972.

[22] D. S. Modha, "Codelet Parsing: Quadratic-time, sequential, adaptive algorithms for lossy compression," in *Proc. Data Compression Conf. (DCC), Snowbird, UT*, March 24–27, 2003.

[23] D. S. Modha, "Codelet Parsing: Quadratic-time, sequential, adaptive algorithms for lossy compression," Tech. Rep. RJ 10267 (A0211-021), IBM Almaden Research Center, San Jose, CA, November 5, 2002.

[24] D. S. Modha, "The art of making mistakes: A quadratic-time, sequential, adaptive algorithm for lossy compression," Tech. Rep. RJ 10286 (A0302-015), IBM Almaden Research Center, San Jose, CA, February 19, 2003.

[25] J. C. Kieffer, "A survey of the theory of source coding with a fidelity criterion," *IEEE Trans. Inform. Theory*, vol. 39, no. 5, pp. 1473–1490, 1993.

[26] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[27] K. Cheung and V. K. Wei, "A locally adaptive source coding scheme," in *Bilkent Conf. on New Trends in Comm., Cont., and Signal Proc.*, pp. 1473–1482, 1990.

[28] Z. Zhang and E.-H. Yang, "An on-line universal lossy data compression algorithm via continuous codebook refinement–part ii: Optimality for phi-mixing source models," *IEEE Trans. Inform. Theory*, vol. 42, no. 3, pp. 822–836, 1996.

[29] M. J. Atallah, F. Chyzak, and P. Dumas, "A randomized algorithm for approximate string matching," *Algorithmica*, vol. 29, pp. 468–486, 2001.

[30] G. Navarro, "A guide to approximate string matching," *ACM Computing Surveys*, vol. 33, no. 1, pp. 31–88, 2001.

[31] H. Morita and K. Kobayashi, "An extension of LZW coding algorithm to source coding subject to a fidelity criterion," in *Proc. 4th Joint Swedish-Soviet Int. Workshop on Information Theory, Gotland, Sweden*, pp. 105–109, 1989.

[32] C. Constantinescu and J. A. Storer, "On-line adaptive vector quantization with variable size codebook entries," in *Data Compression Conf.*, pp. 32–41, 1993.

[33] C. Constantinescu and J. A. Storer, "Improved techniques for single-pass vector quantization," *Proceedings of the IEEE*, vol. 82, no. 6, pp. 933–939, 1994.

[34] J. A. Storer, *Data Compression: Methods and Theory.* Rockville, Maryland: Computer Science Press, 1988.

[35] Y. Steinberg and M. Gutman, "An algorithm for source coding subject to a fidelity criterion, based on string matching," *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 877–886, 1993.

[36] I. Kontoyiannis, "An implementable lossy version of the Lempel-Ziv algorithm–part i: Optimality for memoeyless sources," *IEEE Trans. Inform. Theory*, vol. 45, no. 7, pp. 2293–2305, 1999.

[37] M. Atallah, Y. Genin, and W. Szpankowski, "Pattern matching image compression: Algorithmic and empirical results," in *Proc. Int. Conf. Image Processing, Lausanne, Switzerland*, vol. II, pp. 349–352, 1996.

[38] M. Alzina, W. Szpankowski, and A. Grama, "2D-pattern matching image and video compression," in *Data Compression Conf.*, pp. 424–433, 1999.

[39] R. Zamir and K. Rose, "Towards lossy Lempel-Ziv: Natural type selection," in *Proc. Inform. Theory Workshop, Haifa, Israel*, p. 58, 1996.

[40] Y. Kochman and R. Zamir, "Adaptive parametric vector quantization by natural type selection," in *Data Compression Conference*, pp. 392–401, 2002.

[41] E.-H. Yang and Z. Zhang, "Variable-rate trellis source encoding," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 586–608, 1999.

[42] G. Zhou and Z. Zhang, "On the redundancy of trellis lossy source coding," *IEEE Trans. Inform. Theory*, vol. 48, pp. 205–218, 2002.

[43] M. Haahr. http://www.random.org.

[44] S. A. Savari, "Redundancy of the Lempel-Ziv incremental parsing rule," *IEEE Transactions on Information Theory*, vol. 43, pp. 9–21, January 1997.