Local Decode and Update for Big Data Compression

Shashank Vatedka, Aslan Tchamkerten

Local Decode and Update for Big Data Compression

Shashank Vatedka, *Member, IEEE,* Aslan Tchamkerten, *Senior Member, IEEE,*

Abstract

This paper investigates data compression that simultaneously allows local decoding and local update. The main result is a universal compression scheme for memoryless sources with the following features. The rate can be made arbitrarily close to the entropy of the underlying source, contiguous fragments of the source can be recovered or updated by probing or modifying a number of codeword bits that is on average linear in the size of the fragment, and the overall encoding and decoding complexity is quasilinear in the blocklength of the source. In particular, the local decoding or update of a single message symbol can be performed by probing or modifying a constant number of codeword bits. This latter part improves over previous best known results for which local decodability or update efficiency grows logarithmically with blocklength.

I. Introduction

Recent articles [2]–[4] point to the mismatch between the amount of generated data, notably genomic data [5]–[7], and hardware and software solutions for cloud storage. There is a growing need for space-optimal cloud storage solutions that allow efficient remote interaction, as frequent remote access and manipulation of a large dataset can generate a large volume of internet traffic [8]–[10].

Consider for instance compressing and storing DNA sequences in the cloud. If compression is handled via traditional methods, such as Lempel-Ziv [11], [12], then to retrieve say a particular gene, typically a few tens of thousands of base pairs, we need to decompress the entire DNA sequence, about three billion base pairs. Similarly, the update of a small fraction of the DNA sequence requires to update the compressed data entirely. Solutions have been proposed, typically using modifications of Lempel-Ziv and variants, to address some of these issues (see *e.g.*, [13]–[17] and the references therein).

In this paper we investigate lossless data compression with both local decoding and local update properties. Accordingly, consider a rate R compression of an i.i.d.$\sim p_X$ sequence X^n. Let $d(s)$ denote the *average* (over the randomness in the source X^n) number of bits of the codeword sequence that need to be probed, possibly adaptively, to decode an arbitrary length s contiguous substring of X^n. Similarly, let $u(s)$ denote the *average* number of codeword bits that need to be read and written, possibly adaptively, in order to update an arbitrary length s contiguous substring of X^n. The basic question addressed here is whether it is possible to design a compression scheme such that the operations of local decoding and local update involve a number of bits that is proportional to the number of bits to be retrieved or updated. Specifically, is it possible to design a coding scheme such that, for any R larger than the entropy $H(p_X)$,

$$(d(s), u(s)) = (O(s), O(s)) \quad \text{for any } 1 \leqslant s \leqslant n \ ?$$

As we show in this paper, the answer is positive. Given $\varepsilon > 0$, we exhibit a compressor, a local decoder and a local updater with the following properties:

- The compressor achieves rate $R = H(p_X) + \varepsilon$ universally.
- The local decoder achieves constant decodability
$$d(1) = \alpha_1 \left(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon} \right)$$
for some constant $\alpha_1 < \infty$ that only depends on p_X.
- the local updater achieves constant update
$$u(1) = \alpha_2 \left(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon} \right)$$
for some constant α_2 that only depends on p_X.
- For all $s \geqslant 3$
$$d(s) < s \cdot d(1)$$
and
$$u(s) < s \cdot u(1).$$

This work was supported by Nokia Bell Labs France within the framework "Computation over Encoded Data with Applications to Large Scale Storage." This work was presented in part at the 2019 IEEE International Symposium on Information Theory, Paris, France [1].

S. Vatedka and A. Tchamkerten are with the Department of Communications and Electronics, Telecom Paris, Paris, France. Email: {shashank.vatedka, aslan.tchamkerten}@telecom-paristech.fr

Moreover, if the source is non-dyadic then there exists $\alpha_3 > 0$ independent of n, ε such that for all $s > \alpha_3/\varepsilon^2$, we have

$$d(s) < s \cdot d^*(1)$$

where $d^*(1)$ denotes the minimum average local decodability that can possibly be achieved by any compression scheme having rate $R \leqslant H(p_X) + \varepsilon$.[1]

- The compression scheme has an overall encoding and decoding computational complexity that is quasilinear in n.

We also show, through a second scheme, that it is possible to achieve $(O(\log\log n), O(\log\log n))$ *worst-case* local decodability and average update efficiency for any R larger than the entropy $H(p_X)$ of the underlying source.

Related works: word-RAM and bitprobe models

There has been a lot of work related to local decoding of compressed data structures; see, *e.g.*, [18]–[23] and the references therein. Most of these results hold under the word-RAM model which assumes that operations are on blocks of $\Theta(\log n)$ bits, where n denotes the length of the source sequence. It is assumed that operations (memory access, arithmetic operations) on words of $\Theta(\log n)$ bits take constant time, and the efficiency of a scheme is measured in terms of the time complexity required to perform local decoding. Therefore, algorithms in all these papers must probe $\Omega(\log n)$ bits of the codeword even if only to recover a single bit of the source sequence.

In the word-RAM model it is possible to compress any sequence to its empirical entropy and still be able to locally decode any message symbol in constant time [18], [19]. In particular, [18] gives a multilevel encoding procedure that is conceptually related to our first scheme—the difference will be discussed later in Section IV-E. Another compression scheme is due to Dutta *et al.* [24] which achieves compression lengths within a $(1 + \varepsilon)$ multiplicative factor of that of LZ78 while allowing local decoding of individual symbols in $O(\log n + 1/\varepsilon^2)$ time on average. Bille *et al.* [25] gave a scheme that allows one to modify any grammar-based compressor (such as Lempel-Ziv) to provide efficient local decodability under the word-RAM model. Viola *et al.* [26] recently gave a scheme that achieves near-optimal compression lengths for storing correlated data while being able to locally decode any data symbol in constant time. There is a long line of work, *e.g.*, [27]–[30], on compression schemes that allow efficient local recovery of length $m > 1$ substrings of the message.

Concerning local update, Makinen and Navarro [31] designed an entropy-achieving compression scheme that supports insertion and deletion of symbols in $O(\log n)$ time. Successive works [32]–[34] gave improved compressors that support local decoding, updates, insertion and deletion of individual symbols in $O(\log n/\log\log n)$ time.

While the word-RAM model is natural for on-chip type of applications where data transfer occurs through a memory bus (generally $\Omega(\log n)$ bits), it is perhaps less relevant for (off-chip) communication applications such as between a server, hosting the compressed data set, and the client. In this context it is desirable to minimize the number of bits exchanged, and a more relevant model is the so-called bitprobe model [35] where the complexity of updating or decoding is measured by the number of bits of the compressed sequence that need to be read or modified to recover or update a single bit of the raw data.

Under the bitprobe model, it is known that a single bit of an n-length source sequence can be recovered by accessing a constant (in n) number of bits of the codeword sequence [36]–[39]. However, these works typically assume that the source sequence is deterministic and chosen from a set of allowed sequences, and the complexity of local decoding or update is measured for the worst-case allowed sequence.

The problem of locally decodable source coding of random sequences has received attention very recently. Mazumdar *et al.* [40] gave a fixed-blocklength entropy-achieving compression scheme that permits local decoding of a single bit efficiently. For a target rate of $H(p_X) + \varepsilon$ the decoding of a single bit requires to probe $d(1) = \Theta(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ bits on the compressed codeword. They also provided a converse result for non-dyadic sources: $d(1) = \Omega(\log(1/\varepsilon))$ for any compression scheme that achieves rate $H(p_X) + \varepsilon$. Tatwawadi *et al.* [41] extended the achievability result to Markov sources and provided a universal scheme that achieves $d(1) = \Theta(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon})$. It should perhaps be stressed that the papers [40], [41] only investigate local decoding of a single bit and, in particular, they leave open the question whether we can achieve $d(s) < sd^*(1)$ for $s > 1$. It should also be noted that the construction in these papers make use of the bitvector compressor of Buhrman *et al.* [36] which in turn is a nonexplicit construction based on expander graphs. It is also unclear whether their encoding and decoding procedures can be peformed with low (polynomial-time) computational complexity. Relatedly, Makhdoumi *et al.* [42] showed interestingly that any linear source code that achieves $d(1) = \Theta(1)$ necessarily operates at a trivial compression rate ($R = 1$ for binary sources).

All the above papers on the bit-probe model consider fixed-length block coding. Variable-length source coding was investigated by Pananjady and Courtade [43] who gave upper and lower bounds on the achievable rate for the compression of sparse sequences under local decodability constraints.

Update efficiency was studied in [44], which used sparse-graph codes to design an entropy-achieving compression scheme for which an update to any single message bit can be performed by modifying at most $u(1) = \Theta(1)$ codeword bits. The authors remarked that their scheme cannot simultaneously achieve $d(1) = \Theta(1)$ and $u(1) = \Theta(1)$. Related to update efficiency is the

[1]We guarantee that local decompression of *contiguous* substrings of the message can be performed more efficiently than repeated local decompression of the individual bits. If we want to recover s arbitrary non-contiguous message symbols, it is not clear if we can simultaneously achieve rate close to entropy and $d(s) < sd^*(1)$.

notion of malleability [45], [46], defined as the average fraction of codeword bits that need to be modified when the message is updated by passing through a discrete memoryless channel.

Also related is the notion of local encodability, defined to be the maximum number of message symbols that influence any codeword symbol. Note that this is different from update efficiency, which is the number of codeword symbols that are influenced by any message symbol. Mazumdar and Pal [47] observed the equivalence of locally encodable source coding with a problem of semisupervised clustering, and derived upper and lower bounds on the local encodability. Locality has been well studied in the context of channel coding—see, *e.g.*, [48]–[53] and the references therein.

An outline of this paper is as follows. In Section II, we describe the model. In Section III, we present our results which are based on two schemes. The first achieves $(d(1), u(1)) = (\Theta(1), \Theta(1))$ and the the second scheme achieves $(d_{\mathrm{wc}}(1), u(1)) = (O(\log\log n), O(\log\log n))$. The detailed description of these schemes as well as the proof of the main results appear in Sections IV and V. In Section VI, we provide a few concluding remarks. We end this section with notational conventions.

Notation

We use standard Bachmann-Landau notation for asymptotics. All logarithms are to the base 2. Curly braces denote sets, *e.g.*, $\{a, b, c\}$, whereas parentheses are used to denote ordered lists, *e.g.*, (a, b, c). The set $\{1, 2, \ldots, n\}$ is denoted by $[n]$, whereas for any positive integers i, m, we define $i : i + m$ to be $\{i, i + 1, \ldots, i + m\}$. The set of all finite-length binary sequences is denoted by $\{0, 1\}^*$.

Random variables are denoted by uppercase letters, *e.g.*, X, Y. Vectors of length n are indicated by a superscript n, *e.g.*, x^n, y^n. The ith element of a vector x^n is x_i. Uppercase letters with a superscript n indicate n-length random vectors, *e.g.*, X^n, Y^n. A substring of a vector x^n is represented as $x_i^{i+m} \stackrel{\text{def}}{=} (x_i, x_{i+1}, \ldots, x_{i+m})$.

Let $\mathcal{X}$ be a finite set. For any $x^n \in \mathcal{X}^n$, let $\hat{p}_{x^n}$ be the type/histogram of x^n, *i.e.*, $\hat{p}_{x^n}(a) = \frac{\sum_{i=1}^n 1_{\{x_i = a\}}}{n}$. We say that x^n is ε-typical with respect to a distribution p_X if for all $a \in \mathcal{X}$, we have $|\hat{p}_{x^n}(a) - p_X(a)| \leqslant \varepsilon p_X(a)$. Let $\mathcal{T}_\varepsilon^n$ denote the set of all n-length sequences that are ε-typical with respect to p_X. We impose an ordering (which may be arbitrary) on $\mathcal{T}_\varepsilon^n$. If $x^n \in \mathcal{T}_\varepsilon^n$ is the ith sequence in $\mathcal{T}_\varepsilon^n$ according to the order, then we say that the index of x^n in $\mathcal{T}_\varepsilon^n$ (denoted by $\mathtt{index}(x^n; \mathcal{T}_\varepsilon^n)$) is i.

II. Querying and updating compressed data

The source is specified by a distribution p_X over a finite alphabet $\mathcal{X}$. Unless otherwise mentioned, a source sequence or a message refers to n i.i.d. realizations X^n of the source.

Definition II.1 (Compression scheme)**.** *A rate R length n compression scheme, denoted as (n, R) compression scheme or (n, R) fixed-length compression scheme, is a pair of maps $(\mathrm{ENC}, \mathrm{DEC})$ consisting of*

- *An encoder* $\mathrm{ENC} : \mathcal{X}^n \to \{0, 1\}^{nR}$*, and*
- *A decoder* $\mathrm{DEC} : \{0, 1\}^{nR} \to \mathcal{X}^n$*.*

The probability of error is the probability of the event that codeword $\mathrm{ENC}(X^n)$ is wrongly decoded, that is

$$P_e \stackrel{\text{def}}{=} \Pr_{X^n}[\mathrm{DEC}(\mathrm{ENC}(X^n)) \neq X^n].$$

A. Queries

Given a compression scheme, a local decoder is an algorithm which takes $(i, s) \in [n]^2$ as input, adaptively queries (a small number of) bits of the compressed sequence C^{nR} and outputs X_i^{i+s-1}.

Given $s \in [n]$ and codeword c^{nR} corresponding to source sequence x^n, let $d^{(s)}(i, x^n)$ denote the number of symbols of c^{nR} that need to be queried by the local decoder in order to decode x_i^{i+s-1} without error. The average local decodability of the code is defined as

$$d(s) \stackrel{\text{def}}{=} \max_{i \in [n-s+1]} \mathbb{E}[d^{(s)}(i, X^n)],$$

where the average is taken over X^n and possibly any randomness in the query algorithm. Hence, if say $d(3) = 20$ then the local decoder that can recover any length 3 contiguous substring of the source by probing on average 20 symbols from the codeword sequence.

The worst-case local decodability is defined as

$$d_{\mathrm{wc}}(s) \stackrel{\text{def}}{=} \max_{i, x^n} d^{(s)}(i, x^n).$$

B. Updates

Given $s \in [n]$, suppose a subsequence x_i^{i+s-1} of the original sequence x^n is updated to $\tilde{x}_i^{i+s-1}$ so that x^n becomes $x^{i-1}\tilde{x}_i^{i+s-1}x_{i+s}^n$. A local updater is an algorithm which takes $(i, \tilde{x}_i^{i+s-1})$ as input, probes (a small number of) bits of the compressed sequence c^{nR}, and modifies a small number of bits of c^{nR} such that the new codeword $\tilde{c}^{nR}$ corresponds to the message $x^{i-1}\tilde{x}_i^{i+s-1}x_{i+s}^n$. We assume here that the update algorithm probes and modifies c^{nR} given $(i, \tilde{x}_i^{i+s-1})$ only, without prior knowledge of (x^n, c^{nR}).

Accordingly, let $u_{\mathrm{rd}}^{(s)}(i, x^n, \tilde{x}_i^{i+s-1})$ and $u_{\mathrm{wr}}^{(s)}(i, x^n, \tilde{x}_i^{i+s-1})$ denote the number of symbols of c^{nR} that need to be read and modified, respectively, and let

$$u_{\mathrm{tot}}^{(s)}(i, x^n, \tilde{x}_i^{i+s-1}) \stackrel{\text{def}}{=} u_{\mathrm{rd}}^{(s)}(i, x^n, \tilde{x}_i^{i+s-1}) + u_{\mathrm{wr}}^{(s)}(i, x^n, \tilde{x}_i^{i+s-1}).$$

The average update efficiency of the code is defined as

$$u(s) \stackrel{\text{def}}{=} \max_{i\in[n-s+1]} \mathbb{E}\left[u_{\mathrm{tot}}^{(s)}(i, X^n, \tilde{X}_i^{i+s-1})\right]$$

where the update $\tilde{X}_i^{i+s-1}$ is supposed to be independent of the original sequence X^n but is drawn from the same i.i.d.$\sim p_X$ distribution. Hence, updates do not modify the distribution of the original message. The worst-case update efficiency is defined as

$$u_{\mathrm{wc}}(s) \stackrel{\text{def}}{=} \max_{i,x^n,\tilde{x}_i^{i+s-1}} u_{\mathrm{tot}}^{(s)}(i, x^n, \tilde{x}_i^{i+s-1}).$$

This paper is concerned about the design of $(n, H(p_X)+\varepsilon)$ compression schemes with vanishingly small probability of error that allows the recovery and update of short fragments (contiguous symbols) of the message efficiently.

III. Main results

A naive approach to achieve compression with locality is to partition the message symbols into nonoverlapping blocks of equal size b and compress each block separately with a $(b, H(p_X)+\varepsilon)$ fixed-length compression scheme. The probability of error for each block can be made to go to zero as $2^{-\Theta(b)}$ (see, *e.g.*, [54]). From the union bound, the overall probability of error is at most $(n/b)2^{-\Theta(b)}$. Hence, as long as $b = \Omega(\log n)$ we have $P_e = o(1)$. Since the blocks are encoded and decoded independently,

$$d_{\mathrm{wc}}(1) = u_{\mathrm{wc}}(1) = O(b) = O(\log n)$$

where the constant in the order term does not depend on ε. The overall computational complexity is at most $(n/b)2^{\Theta(b)}$, which is polynomial in n. Noticing that every subsequence of length $s > 1$ is contained in at most $\lceil s/b\rceil + 1$ blocks, we have:[2]

Theorem III.1 (Fixed-length neighborhood and compression)**.** *For every $\varepsilon > 0$, the naive scheme achieves a rate-locality triple of*

$$(R, d_{\mathrm{wc}}(1), u_{\mathrm{wc}}(1)) = (H(p_X)+\varepsilon, O(\log n), O(\log n)).$$

Moreover,

$$d_{\mathrm{wc}}(s) = \begin{cases} \Theta(\log n), & \text{if } s \leqslant b \\ \Theta(s), & \text{if } s > b \end{cases}$$

$$u_{\mathrm{wc}}(s) = \begin{cases} \Theta(\log n), & \text{if } s \leqslant b \\ \Theta(s), & \text{if } s > b \end{cases}$$

where all the order terms are independent of ε. The overall computational complexity required for compression/decompression is polynomial in n.

It is easy to see that the above analysis is essentially tight as the naive scheme achieves vanishingly small error probabilities for overall compression and decompression only if $b = \Omega(\log n)$.

In the naive scheme, the recovery or update of a particular symbol X_i involves an $O(\log n)$-size neighborhood of that symbol which is compressed by means of a fixed-length compression scheme. To improve upon the $O(\log n)$ locality, we consider two other schemes. In the first, neighborhoods are of variable lengths and are compressed using a fixed length block code. The length of the neighborhood of a particular symbol X_i is defined as the length of the smallest typical set that contains X_i. To find this smallest neighborhood, the algorithm proceeds iteratively by considering larger and larger neighborhoods of X_i

[2]In case b does not divide n, we can compress the last block of size $b+n-\lfloor n/b\rfloor b$ separately using a $(b+n-\lfloor n/b\rfloor b, H(p_X)+\varepsilon)$-fixed length compression scheme. The local decodability and update efficiency would increase by a factor of less than 2, and therefore remain $O(\log n)$. A similar argument can be made for all the multilevel schemes in the rest of this paper and overall will only introduce an additional constant multiplicative factor. For ease of exposition, we will conveniently assume in all our proofs that the size of each block divides n.

until it finds a neighborhood that is typical. Local decoding and local recovery of X_i are performed by decompressing and recompressing this neighborhood. This scheme is formally described in Section IV where we prove the following result:

Theorem III.2 (Variable-length neighborhood and fixed length compression)**.** *Fix $\varepsilon > 0$. There exists a scheme which universally over i.i.d. sources with common known finite alphabet achieves rate $R = H(p_X) + \varepsilon$, and probability of error $\Pr[\mathrm{DEC}(\mathrm{ENC}(X^n)) \neq X^n] = 2^{-2^{\Omega(\sqrt{\log n})}}$. The average local decodability and update efficiency is*

$$d(s) \leqslant \begin{cases} \alpha_1 \frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon} & \text{if } s \leqslant \alpha_1'' \left(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon}\right) \\ \alpha_1' s & \text{if } s > \alpha_1'' \left(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon}\right) \end{cases},$$

$$u(s) \leqslant \begin{cases} \alpha_2 \frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon} & \text{if } s \leqslant \alpha_2'' \left(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon}\right) \\ \alpha_2' s & \text{if } s > \alpha_2'' \left(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon}\right) \end{cases},$$

where the constants $\alpha_i, \alpha_i', \alpha_i''$, $i = 1, 2$, are independent of n, ε but dependent on p_X. Moreover, the overall computational complexity of encoding and decoding X^n is $O(n \log n)$. For $1 \leqslant s \leqslant n$, the expected computational complexity for local decoding or updating a fragment of size s is $\Theta(s)$, where the proportionality constant depends only on ε and p_X.[3]

Mazumdar *et al.* [40] proved that $d^*(1) = \Omega(\log(1/\varepsilon))$ for non-dyadic sources.[4] Hence, from Theorem III.2 we get:

Corollary III.1. *There exists a universal constant $\alpha_f > 0$ such that for all non-dyadic sources, the scheme of Theorem III.2 achieves $d(s) < s d^*(1)$ whenever $s \geqslant \alpha_f/\varepsilon^2$.*

Given Theorem III.1, the interesting regime of Corollary III.1 is when attempting to locally decode a substring of size s that satisfies

$$\Omega(1/\varepsilon^2) \leqslant s \leqslant o(\log n).$$

Theorem III.2 involves average local decoding and average local update. A natural question is whether we can achieve the same performance but under worst-case locality, *i.e.*, can we achieve for any $1 \leqslant s \leqslant n$

$$(d_{\mathrm{wc}}(s), u_{\mathrm{wc}}(s)) = (O(s), O(s))?$$

While this question remains open we show that it is possible to achieve $(d_{\mathrm{wc}}(s), u(s)) = (O(s)), O(s))$ whenever $s = \Omega(\log\log(n))$. This result is obtained by means of a second scheme where neighborhoods are of fixed length, as in the naive scheme, but compressed with a variable length code. Using such as a code raises the problem of efficiently encoding the start and end locations of each subcodeword. Indeed, were we to store an index of the locations of each subcodeword, and since there are n/b subcodewords, the index would take approximately $(n \log n)/b$ additional bits of space. Hence, only to ensure that the rate remains bounded would require $b = \Omega(\log n)$, which would further imply that $d_{\mathrm{wc}}(1)$ and $u_{\mathrm{wc}}(1)$ are still $O(\log n)$. It turns out that the location of individual subcodewords can be done much more efficiently by means of a particular data structure for subcodeword location as we show in Section V:

Theorem III.3 (Fixed-length neighborhood and variable-length compression)**.** *Fix $\varepsilon > 0$. There exists a scheme which univerally over i.i.d. sources with common known finite alphabet achieves a rate-locality triple of*

$$(R, d_{\mathrm{wc}}(1), u(1)) = (H(p_X) + \varepsilon, O(\log\log n), O(\log\log n))$$

where order terms are independent of ε.

Moreover, for any $s > 1$,

$$d_{\mathrm{wc}}(s) \leqslant \begin{cases} 2 d_{\mathrm{wc}}(1) & \text{if } s \leqslant b_1 \\ s(H(p_X) + \varepsilon) + 2 d_{\mathrm{wc}}(1) & otherwise, \end{cases}$$

and

$$u(s) \leqslant \begin{cases} 2u(1) & \text{if } s \leqslant b_1 \\ 2s(H(p_X) + \varepsilon) + 2u(1) & otherwise, \end{cases}$$

where $b_1 = O(\log\log n)$. The overall computational complexity of encoding and decoding is polynomial in n.

Analogously to the derivation of Corollary III.1 we get:

Corollary III.2. *For non-dyadic sources, there exists a constant $\alpha_v > 0$ such that the scheme of Theorem III.3 achieves $d_{\mathrm{wc}}(s) < s d^*_{\mathrm{wc}}(1)$ whenever $s \geqslant \alpha_v(\log\log n)$.*

All our results easily extend to variable-length codes with zero error—See Appendix C-1.

[3]In comparison, the naive scheme requires computational complexity $\Omega(\log n)$ to locally decode or update even a single symbol.

[4]Recall that $d^*(1)$ denotes the minimum average local decodability that can be achieved by any compression scheme having rate $R \leqslant H(p_X) + \varepsilon$.

Discussion

Mazumdar *et al.* [40] gave a compression scheme that achieves $R = H(p_X) + \varepsilon$ and $d_{\mathrm{wc}}(1) = \Theta(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$. The probability of error decays as $2^{-\Theta(n)}$. This suggests that we can achieve $u_{\mathrm{wc}}(1) = O(\log n)$ using the following scheme. Split the message into blocks of $O(\log n)$ symbols each, and use the scheme of Mazumdar *et al.* in each block. We can choose the size of each block so that the overall probability of error decays polynomially in n. Since each block of size $O(\log n)$ is processed independently of the others, the overall computational complexity (which may be exponential in the size of each block) is only polynomial in n. This gives us the following result:

Lemma III.1 (Corollary to [40])**.** *For every $\varepsilon > 0$, a rate-locality triple of*

$$(R, d_{\mathrm{wc}}(1), u_{\mathrm{wc}}(1)) = \left(H(p_X) + \varepsilon, \Theta\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right), O(\log n)\right)$$

is achievable with $\mathrm{poly}(n)$ *overall encoding and decoding complexity.*

Although the above scheme has $\mathrm{poly}(n)$ computational complexity, this could potentially be a high-degree polynomial. Moreover, we do not know if the above scheme can achieve $d_{\mathrm{wc}}(s) < sd_{\mathrm{wc}}(1)$ for $1 < s = o(\log n)$.

Montanari and Mossel [44] gave a compressor that achieves update efficiency $u_{\mathrm{wc}}(1) = \Theta(1)$. The construction is based on syndrome decoding using low-density parity-check codes. Arguing as above we deduce the following lemma:

Lemma III.2 (Corollary to [44])**.** *For every $\varepsilon > 0$, a rate-locality triple of*

$$(R, d_{\mathrm{wc}}(1), u_{\mathrm{wc}}(1)) = (H(p_X) + \varepsilon, O(\log n), \Theta(1))$$

is achievable with $\mathrm{poly}(n)$ *overall encoding and decoding complexity.*

The local decodability of the compressor in [44] cannot be improved as it uses a linear encoder for the compression of each block, and Makhdoumi *et al.* [42] showed that for such a compression scheme local decodability ($d_{\mathrm{wc}}(1)$) necessarily scales logarithmically with block size, hence in our case $d_{\mathrm{wc}}(1) = \Omega(\log n)$. Hence, linearity in the encoding impacts local decodability.

As we note in the following lemma, if we impose the decoder to be linear then it is impossible to even obtain nontrivial rates of compression:

Lemma III.3. *Fix $p \in (0, 1/2]$. Any fixed-length compression scheme with linear decoder and achieving vanishingly small probability of error for a Bernoulli(p) source has asymptotic rate equal to one.*

Proof. Consider any fixed-length scheme having a linear decoder specified by an $n \times nR$ matrix A. The decoded message is therefore $\hat{x}^n = Ac^{nR}$. Clearly, the reconstructed message must lie within the column space of A. Therefore, the probability of error is upper bounded by

$$\Pr[\hat{X}^n \neq X^n] \leqslant \Pr[X^n \notin \mathcal{S}(A)],$$

where $\mathcal{S}(A)$ denotes the column space of A and has dimension at most nR. From [42, Lemma 1],

$$\Pr[X^n \notin \mathcal{S}(A)] \leqslant 1 - p^{n(1-R)},$$

which can be $o(1)$ only if $R = 1 - o(1)$. In other words, we cannot achieve an asymptotic rate of less than one. □

IV. Proof of Theorem III.2

We now present our compression scheme which achieves constant $(d(1), u(1))$. We assume first that the source distribution p_X is known, as it is conceptually simpler. The universal scenario is handled separately in Section IV-I.

Before giving a formal description of our scheme, let us give some intuition.

A. Intuition

The main idea is to analyze the message sequence at multiple levels: At the coarsest level, we view the message as a single block of size n. At the finest level, we view it as a concatenation of blocks of size $b_0 = \Theta(1)$. As depicted in Figure 1, we can refine this by saying that at level ℓ, the message is viewed as a concatenation of n_ℓ-sized neighborhoods, where $b_0 < n_1 < n_2 < \ldots < n$. If $b_0 = \Theta(1)$, then a positive fraction of the level-0 neighborhoods are atypical with high probability, while neighborhoods at higher levels are more likely to be typical.

Corresponding to each b_0-sized block, we identify the smallest typical neighborhood containing the block. In the example of Figure 1, the smallest typical neighborhood of $x^{b_0}(1)$ is $x^{n_1}(1,1)$ at level 1, while that of $x^{b_0}(2)$ is $x^{b_0}(2)$ itself. The main idea in our scheme is to efficiently encode typical neighborhoods at each level, and local decoding/update of a symbol is performed by decompressing/recompressing only the smallest typical neighborhood containing it.

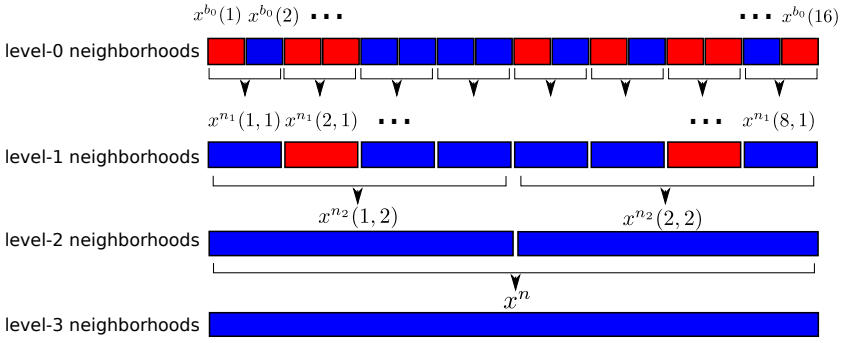

Fig. 1: Intution for the multilevel compression scheme in Section IV-B. We group together symbols to form larger neighborhoods. If we have an efficient means to compress these neighborhoods, then we can locally decode a block by decompressing the smallest typical neighborhood of that block. Blocks colored blue are typical, while the red blocks are atypical.

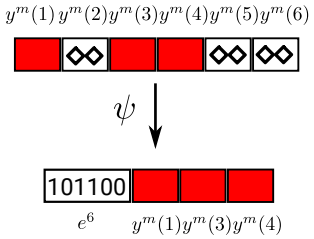

Fig. 2: Illustrating the compression scheme for levels $\ell \geqslant 1$ as described in Definition IV.2. In this example, we have used $b = 6$ and $\varepsilon = 1/2$.

Our actual scheme is more nuanced. We will view the message sequence at different levels, but use a different definition of typicality at each level. Compression of the neighborhoods is performed in an iterative fashion, starting from level 0, and then moving to higher levels. At level ℓ, we only compress the residual information of each neighborhood, *i.e.,* that which is not recoverable from the first $\ell - 1$ levels.

Local decoding of a symbol is performed by successively answering the question "Is the level-ℓ neighborhood typical?" for $\ell = 0, 1, \ldots$, till we get a positive answer. The desired symbol can be recovered from the typical neighborhood.

We proceed with the formal description of our scheme.

B. Compression scheme

Fix $\varepsilon_0 > 0$. Let $b_0 \stackrel{\text{def}}{=} n_0 \stackrel{\text{def}}{=} \Theta(\frac{1}{\varepsilon_0^2} \log \frac{1}{\varepsilon_0})$ where the implied constant is chosen so that

$$\Pr[X^{b_0} \notin \mathcal{T}_{\varepsilon_0}^{b_0}] \leqslant \varepsilon_0^4,$$

and let

$$k_0 \stackrel{\text{def}}{=} \lceil (H(p_X) + \varepsilon_0) b_0 \rceil.$$

For $\ell \geqslant 1$, let

$$\varepsilon_\ell = \varepsilon_{\ell-1}/2,$$

$$b_\ell = 4 b_{\ell-1},$$

$$n_\ell = b_\ell n_{\ell-1},$$

and let $\ell_{\max}$ be the largest ℓ such that $n_\ell \leqslant n$.

Notice that $\ell_{\max} = \Theta(\sqrt{\log n})$.

The overall encoding/decoding involves a multilevel procedure over $\ell_{\max}$ levels. At each level, we generate a part of the codeword and modify the input string in an entropy decreasing manner until the string becomes a constant. The scheme uses a special marker symbol, referred to as $\diamond$, that is not in $\mathcal{X}$. This symbol will be used to denote that we have been able to successfully compress a part of the message at an earlier stage.

Definition IV.1 ($\diamond$ blocks and non-$\diamond$ blocks)**.** *A vector v^m is said to be a $\diamond$-block if $v_i = \diamond$ for all i. It is called a non-$\diamond$ block if there exists an i such that $v_i \neq \diamond$.*

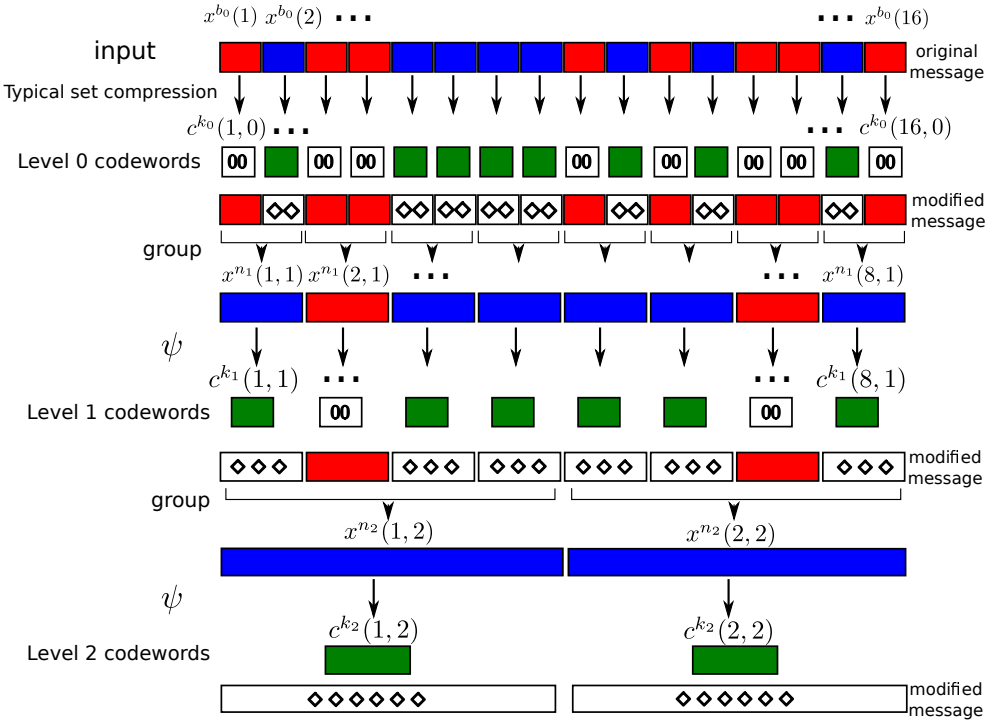

Fig. 3: Illustrating the multilevel compression scheme. Red and blue blocks denote atypical and typical blocks respectively, while green blocks denote nonzero codewords. For ease of illustration, we have used $b_\ell = 2b_{\ell-1}$.

1) Level $\ell = 0$: partition x^n into n/b_0 blocks of length $b_0 = n_0$ each. Let $x^{n_0}(j) \stackrel{\text{def}}{=} x_{(j-1)n_0+1}^{jn_0}$ denote the jth block of the message symbols. Blocks at level $\ell = 0$ are processed independently of each other. For each $x^{n_0}(j)$, we generate a codeword block $c^{k_0}(j,0)$ and possibly modify $x^{n_0}(j)$:

- If $x^{n_0}(j)$ is typical, then $c^{k_0}(j,0)$ is assigned the index of $x^{n_0}(j)$ in $\mathcal{T}_{\varepsilon_0}^{n_0}$, else $c^{k_0}(j,0) = 0^{k_0}$.
- If $x^{n_0}(j)$ is typical, then $x^{n_0}(j,0)$ is modified to a diamond block $\diamond^{n_0}$ and if $x^{n_0}(j)$ is not typical then $x^{n_0}(j,0)$ is kept unchanged. The message sequence after possible modifications of each block $x^{n_0}(j,\ell=0)$, $j = 1,2,\ldots$ is denoted by $x^n(\ell=0)$.

For compression at higher levels, we make use of the following code

Definition IV.2 (Code for levels $\ell \geqslant 1$)**.** *Fix any positive integers b, m. Let $\mathcal{X}$ be a finite alphabet, and $\diamond$ be a symbol such that $\diamond \notin \mathcal{X}$. Let $\mathcal{S} \subset (\mathcal{X} \cup \{\diamond\})^{mb}$ be the set of all sequences of the form $y^{mb} = (y^m(1), y^m(2), \ldots, y^m(b))$ such that $y^m(j) \in (\mathcal{X} \cup \{\diamond\})^m$ and at least $(1-\varepsilon)b$ fraction of the $y^m(j)$'s are $\diamond$ blocks.*

For any sequence $y^{mb} \in \mathcal{S}$, let $j_1, j_2, \ldots, j_k$ denote the locations of the non-$\diamond$ blocks. Let $e^b = \phi(y^{mb}; b, m)$ be the b-length indicator vector for the non-$\diamond$ blocks, i.e., the jth element of $\phi(y^{mb}; b, m)$ is 1 iff $y^{mb}(j)$ is a non-$\diamond$ block. Let

$$\psi(y^{mb}; b, m, \varepsilon) \stackrel{def}{=} (e^b, y^{mb}(j_1), , \ldots, y^{mb}(j_k), \diamond^{m(\varepsilon b - k)}).$$

In other words, ψ consists of a header e^b to locate the non-$\diamond$ blocks, followed by a concatenation of all the non-$\diamond$ blocks. The binary representation of ψ requires $b + \varepsilon mb \log(|\mathcal{X}|+1)$ bits. The mapping ψ is one-to-one on $\mathcal{S}$. Both ψ and ψ^{-1} (for any element in the range of ψ) can be computed using $\Theta(mb)$ operations. An example is illustrated in Figure 2.

2) Levels $\ell \geqslant 1$: having generated codewords up to level $\ell - 1$ and having modified the message if necessary, we form groups of b_ℓ consecutive blocks from $x^n(\ell-1)$ to obtain blocks of size $n_\ell = b_\ell n_{\ell-1}$. The jth block at level ℓ, denoted $x^{n_\ell}(j,\ell)$, is therefore

$$(x^{n_{\ell-1}}((j-1)b_\ell + 1, \ell-1), \ldots, x^{n_{\ell-1}}(jb_\ell + 1, \ell-1)).$$

Similarly to level $\ell = 0$, for each of these blocks of size n_ℓ, we generate a codeword and modify it if necessary:

- If $x^{n_\ell}(j,\ell)$ is "typical," *i.e.*, has at least $(1-\varepsilon_\ell)b_\ell$ $\diamond$-blocks (of size $n_{\ell-1}$), then we set the subcodeword $c^{k_\ell}(j,\ell)$ of length $k_\ell = b_\ell + \varepsilon_\ell n_\ell \log(|\mathcal{X}|+1)$ using the scheme described in Definition IV.2.[5] If this block is "atypical," *i.e.*, has fewer than $(1-\varepsilon_\ell)b_\ell$ many $\diamond$ blocks, then $c^{k_\ell}(j,\ell) = 0^{k_\ell}$.

[5]One could use a more sophisticated scheme to get better performance. However, we can get order-optimal (d,u) even with this very simple scheme.

- If $x^{n_\ell}(j,\ell)$ has at most $\varepsilon_\ell b_\ell$ many non-$\diamond$-blocks, then we modify $x^{n_\ell}(j,\ell)$ to a diamond block $\diamond^{n_\ell}$. Otherwise, the group is left untouched.

Hence, at each level the input sequence gets updated with more and more $\diamond$'s as larger and larger subsequences become typical. As we show in Section IV-F, the entropy of the message keeps decreasing till it becomes zero, once it becomes the all-$\diamond$ sequence. Finally, the stored codeword is the concatenation of codewords of all levels:

$$c^{nR} = (c^{k_0}(1 : n/n_0, 0), \ldots, c^{k_{\ell_{\max}}}(1 : n/n_{\ell_{\max}}, \ell_{\max})).$$

Example IV.1 (Figure 3)**.** *An example of the encoding process is illustrated in Figure 3 where the blue blocks refer to typical blocks whereas the red blocks refer to atypical blocks.*

At level 0*, the subcodewords* $c^{k_0}(i,0)$ *are obtained using typical set compression. The subcodeword* $c^{k_0}(i,0)$ *is zero if the block is atypical, and nonzero (depicted in green in the figure) if it is typical. We then modify the message, replacing each typical level-*0 *block with* $\diamond^{b_0}$*.*

For ease of illustration, we select $b_1 = 2$ *and* $\varepsilon_1 = 1/2$*. Hence the blocks are grouped in pairs to obtain* $x^{n_1}(i,1)$*,* $1 \leqslant i \leqslant 8$*. A block* $x^{n_1}(i,1)$ *is typical if it contains at most one non-*$\diamond$ *block of length* n_0*. Therefore, only* $x^{n_1}(2,1)$ *and* $x^{n_1}(7,1)$ *are atypical. These blocks are compressed to get the level-*1 *codewords* $c^{k_1}(i,1)$ *for* $1 \leqslant i \leqslant 8$*. As earlier, typical blocks are encoded to nonzero codewords, while atypical blocks are compressed to the zero codeword. Post compression, we again modify the message by replacing typical blocks with* $\diamond^{n_1}$*.*

The encoding process proceeds in an identical fashion for level 2*, where we have selected* $b_2 = 4$ *and* $\varepsilon_2 = 1/4$*.*

C. Local decoding

Suppose that we are interested in recovering the mth message symbol x_m, where $m \in (j-1)n_0 : jn_0$.

- We probe $c^{k_0}(j,0)$. If the block $x^{n_0}(j)$ is typical, then we can directly recover $x^{n_0}(j)$ from $c^{k_0}(j,0)$.
- If $x^{n_0}(j)$ is not typical, we probe higher levels successively till we reach the smallest level ℓ for which the block that includes $x^{n_0}(j)$, which we denote as $x^{n_\ell}(q_\ell(j),\ell)$ is a diamond $\diamond^{n_\ell}$- block. This can be determined by reading the first b_i bits of $c^{k_i}(q_i(j),i)$, $i = 1,2,\ldots,\ell$ since this corresponds to the indicator vector of the non-$\diamond$ blocks at each level $i \leqslant \ell$. If we can recover $x^{b_0}(j)$ by probing up to the first ℓ levels, then we say that the jth block is encoded at the ℓth level.
- Using this approach, we automatically recover the entire block $x^{b_0}(j)$—not only an individual message symbol. If we want to recover multiple message blocks, we repeatedly employ the same algorithm on each block.[6]

We revisit our earlier example to illustrate the local decoder.

Example IV.2 (Figure 3)**.** *Suppose that we are interested in recovering* $x^{b_0}(2)$*. The local decoder first probes* $c^{k_0}(2,0)$*. Since this is a nonzero codeword,* $x^{b_0}(2)$ *can be obtained by decompressing* $c^{k_0}(2,0)$*. In this process, the local decoder probes* k_0 *bits.*

Suppose that we are instead interested in recovering $x^{b_0}(3)$*. On probing* $c^{k_0}(3,0)$*, the local decoder obtains a zero codeword. Next, it probes* $c^{k_1}(2,1)$*. This is also zero. Finally, the local decoder probes* $c^{k_2}(1,2)$ *which is nonzero, and* $x^{b_0}(3)$ *can be obtained by decompressing this codeword. In this case, the local decoder probes* $k_0 + k_1 + k_2$ *bits.*

D. Local updating

The local updating rule is a little more involved. Assume that the jth block $x^{n_0}(j)$ is to be updated with $\tilde{x}^{n_0}(j)$.

- If both $x^{n_0}(j)$ and $\tilde{x}^{n_0}(j)$ are typical, only $c^{n_0}(j,0)$ needs to be updated. Whether $x^{n_0}(j)$ is typical or not can be determined by reading $c^{k_0}(j,0)$.
- If both $x^{n_0}(j)$ and $\tilde{x}^{n_0}(j)$ are atypical, then we probe higher levels till we reach the level ℓ where $x^{n_0}(j)$ is encoded, and update $c^{k_0}(q_\ell(j),\ell)$.
- If $x^{b_0}(j)$ is typical and $\tilde{x}^{b_0}(j)$ is atypical, then we need to update $c^{b_0}(j,0)$ and the blocks at higher levels. Due to the atypicality, the number of non-$\diamond$ blocks for level 1 increases by 1, and hence $c^{\ell_1}(q_1(j),1)$ must be updated. If the number of non-$\diamond$ blocks now exceeds $\varepsilon_1 b_1$, then we would also need to update the codeword at level 2, and so forth.
- If $x^{b_0}(j)$ is atypical and $\tilde{x}^{b_0}(j)$ is typical, then the number of non-$\diamond$ blocks at each level might decrease by 1 (or 0). If $x^{b_0}(j)$ were encoded at level i, then we might need to update the codeword blocks up to level i.

Let us illustrate the local updater in the context of our earlier example.

Example IV.3 (Figure 3)**.** *Suppose that we want to replace* $x^{b_0}(5)$ *with* $\widetilde{x}^{b_0}(5)$*. The local updater first probes* $c^{k_0}(5,0)$ *to conclude that* $x^{b_0}(5)$ *is encoded at level* 0*.*

If $\widetilde{x}^{b_0}(5)$ *is also typical, then only* $c^{k_0}(5,0)$ *needs to be updated, and the rest of the codeword remains untouched. The updater probes* k_0 *bits and modifies* k_0 *bits.*

[6]We can actually do much better than naively repeating the algorithm for multiple blocks. However, for ease of exposition and proofs, we use the naive algorithm.

In case $\widetilde{x}^{b_0}(5)$ is atypical, then the local updater first sets $c^{k_0}(5,0)$ to 0^{k_0}. It then probes $c^{k_1}(3,1)$ and decompresses this to recover $x^{n_1}(3,1)$. This block is updated with $\widetilde{x}^{b_0}(5)$, and the new level 1 block $\widetilde{x}^{n_1}(3,1)$ is typical. Therefore, $c^{k_1}(3,1)$ is updated with the codeword corresponding to $\widetilde{x}^{n_1}(3,1)$, and the update process is terminated. In this scenario, the updater probes k_0+k_1 bits and modifies k_0+k_1 bits.

E. Connections with Pătraşcu's compressed data structure [18]

In [18], Pătraşcu gave an entropy-achieving compression scheme that achieves constant-time local decoding in the word-RAM model. The compressor has a multilevel structure whose concept inspired our work.

The basic idea in [18] is the following. At level 0, split the message into blocks of b_0 symbols each, compress each block using an entropy-achieving variable-length compression scheme, and store a fixed number of the compressed bits of each block. The remainder is called the "spill," and is encoded in higher levels. At level $i \geqslant 1$, the spills from each block of level $i-1$ are grouped together to form larger blocks, and compressed in a fashion similar to level 0. Reconstruction of any block necessarily requires both the codeword at level-0 and the spill. As a result, the local decoder of [18] must always probe subcodewords of all levels, and the number of bitprobes required to recover even one symbol is $\Omega(\log n)$.

In our scheme on the other hand encoding is such that the number of levels that the local decoder needs to probe to retrieve one block depends on the realization of the source message. In particular, the local decoder need not always probe all levels—and indeed, probes a small number of levels.

Hence, in Pătraşcu's scheme the information about a particular block is spread across multiple levels whereas in our scheme this information is stored at a particular level that depends on the realization of the message.

In the next section we establish Theorem III.2 assuming the underlying source p_X is known. Universality is handled separately in Section IV-I.

F. Bounds on $d(1)$ and $u(1)$

We now derive bounds on the average local decodability and update efficiency. In the following, we will make use of some preliminary results that are derived in Appendix A.

Lemma IV.1. *If $\varepsilon_0 < 1/2$, then*

$$d(1) \leqslant 2b_0.$$

Proof. We can assume without loss of generality that we want to recover X_1.

If X_1 is encoded at level i, then the local decoder probes $\sum_{i_1=0}^{i} k_{i_1}$ bits. Therefore,

$$\begin{aligned}
\mathbb{E}[d^{(1)}(X^n, b_0)] &\leqslant b_0 + \sum_{i=1}^{\ell_{\max}} \left(\Pr[X_1 \text{ is encoded at level } i] \sum_{i_1=0}^{i} k_{i_1} \right) \\
&\leqslant b_0 + \sum_{i=1}^{\ell_{\max}} \left(\Pr[X_1 \text{ is encoded at level } i] \sum_{i_1=0}^{i} n_{i_1} \right) \\
&\leqslant b_0 + \sum_{i=1}^{\ell_{\max}} \Big((i+1) n_i \Pr[X_1 \text{ is encoded at level } i] \Big). \qquad (1)
\end{aligned}$$

Let $\delta^{(1)}_{i \to i+1}$ denote the conditional probability that $x^{n_i}(1,i)$ is not the all-$\diamond$ block given that $x^{n_{i-1}}(1,i-1)$ is not a $\diamond$-block. Then,

$$\Pr[X_1 \text{ is encoded at level } i] \leqslant \delta_{0 \to 1} \prod_{i_1=1}^{i-1} \delta^{(1)}_{i_1 \to i_1+1}$$

From Lemma A.3, specifically (8), we know that $\delta^{(1)}_{i \to i+1} \leqslant \varepsilon_i^{\beta 2^{i-1}}$ for $i \geqslant 1$. The quantity β is defined in (5). Therefore,

$$\Pr[X_1 \text{ is encoded at level } i] \leqslant \varepsilon_i^{\beta 2^{i-1}}. \qquad (2)$$

Since $n_{i_1} = b_0^{i_1+1} 2^{i_1(i_1+1)}$, we have

$$(i+1) n_i \leqslant (i+1) b_0^{i+1} 2^{i(i+1)}.$$

Using this and (2) in (1), we have

$$\mathbb{E}[d^{(1)}(X^n, b_0)] \leqslant b_0 + \sum_{i=1}^{\ell_{\max}} (i+1) b_0^{i+1} 2^{i(i+1)} \varepsilon_i^{\beta 2^{i-1}}. \qquad (3)$$

It is easy to show that $(i+1)b_0^{i+1}2^{i(i+1)}\varepsilon_i^{\beta 2^{i-1}} \leqslant \varepsilon_0^i$ for all $i \geqslant 1$ (see Lemma A.4 for a proof). Therefore,

$$d(1) = \mathbb{E}[d^{(1)}(X^n, b_0)] \leqslant b_0 + \varepsilon_0 b_0 \sum_{i=1}^{d_{\max}} \varepsilon_0^i < 2b_0$$

if $\varepsilon_0 < 1/2$. This completes the proof. □

Lemma IV.2. *If $\varepsilon_0 < 1/2$, then*

$$u(1) \leqslant 8b_0.$$

Proof. The calculations are identical to those in Lemma IV.1, so we will only highlight the main differences. Again, we can assume that the first symbol needs to be updated.

Suppose $U^{b_0}(1)$ is the new realization of the message block that needs to be updated. Let i_{old} denote the level at which $X^{b_0}(1)$ is encoded in the codeword for X^n, and let i_{new} be the level at which $U^{b_0}(1)$ is encoded in the codeword for $U^{b_0}(1), X^{(b_0)}(2), \ldots, X^{b_0}(n/b_0)$. The number of bits that need to be read is upper bounded by

$$u_{rd} \leqslant \max\{(i_{\text{old}}+1)n_{i_{\text{old}}}, (i_{\text{new}}+1)n_{i_{\text{new}}}\} \leqslant (i_{\text{old}}+1)n_{i_{\text{old}}} + (i_{\text{new}}+1)n_{i_{\text{new}}}.$$

Likewise, the number of bits that need to be written is

$$u_{wr} \leqslant \max\{(i_{\text{old}}+1)n_{i_{\text{old}}}, (i_{\text{new}}+1)n_{i_{\text{new}}}\} \leqslant (i_{\text{old}}+1)n_{i_{\text{old}}} + (i_{\text{new}}+1)n_{i_{\text{new}}}.$$

Since the $U^{b_0}(i)$ is independent of everything else and does not change the message distribution, $u_{\text{wc}}(1)$ is at most 4 times the upper bound in (1). Using the calculations in the proof of Lemma IV.1, the expected number of bits to be read and written is at most $8b_0$. □

G. Proof of Theorem III.2 assuming that p_X is known

1) Rate of the code: Recall that k_i is the length of a subcodeword in the ith level. The achievable rate is given by

$$R = \frac{1}{n}\sum_{i=0}^{\ell_{\max}} k_i \frac{n}{n_i} = \sum_{i=0}^{\ell_{\max}} \frac{k_i}{n_i}.$$

We have $k_0 \leqslant (H(p_X)+\varepsilon_0)b_0$. From Definition IV.2, we have

$$\begin{aligned} k_i &= b_i + \varepsilon_i n_i \log(|\mathcal{X}|+1) \\ &= n_i\left(\frac{1}{n_{i-1}} + \varepsilon_i \log(|\mathcal{X}|+1)\right) \\ &\leqslant n_i\left(\frac{\varepsilon_0}{2^{i(i-1)}} + \frac{\varepsilon_0}{2^i}\log(|\mathcal{X}|+1)\right) \\ &\leqslant n_i(1+\log(|\mathcal{X}|+1))\frac{\varepsilon_0}{2^i}. \end{aligned}$$

Therefore,

$$\begin{aligned} R &\leqslant H(p_X) + \varepsilon_0 + (1+\log(|\mathcal{X}|+1))\sum_{i=1}^{d_{\max}} \frac{\varepsilon_0}{2^i} \\ &\leqslant H(p_X) + \varepsilon_0(2+\log(|\mathcal{X}|+1)). \end{aligned}$$

Hence, the rate is $H(p_X) + \Theta(\varepsilon_0)$.

We show in Corollary A.1 (See Appendix A) that the probability of error is upper bounded by $2^{-2^{O(\sqrt{\log n})}}$.

2) Average local decodability and update efficiency: In Lemmas IV.1 and IV.2, we have established that $d(1)$ and $u(1)$ are both $\Theta(\frac{1}{\varepsilon_0^2}\log\frac{1}{\varepsilon_0})$.

Any sequence of s consecutive message symbols is spread over at most $\lceil m/b_0 \rceil + 1$ level-0 blocks. For any $s \leqslant b_0$, it is clear that $d(s) \leqslant 2d(1)$. For $s > b_0$,

$$d(s) \leqslant (\lceil s/b_0 \rceil + 1)\,\alpha_{ld} b_0 = \alpha_1 s,$$

for some absolute constant α_1 independent of ε_0 and n. Likewise,

$$u(s) = \alpha_2 s.$$

for some α_2 independent of n, ε_0.

3) Computational complexity: Since b_0 is a constant independent of n, the total complexity for encoding/decoding all the codewords at level zero is $\Theta(n)$. From Definition IV.2, the computational complexity of decoding a block at level i is linear in n_i, and there are n/n_i blocks at level i. Since the total number of levels $\ell_{\max}$ is $O(\log n)$, the overall computational complexity is $O(n \log n)$. A similar argument can be made to show that the expected computational complexity for local decoding/updating of a fragment of length s is $\Theta(s)$.

H. Variable-length source code with zero error

Note that Theorem III.2 guarantees the existence of a fixed-length source code with a vanishing probability of error. However, in most applications, we want zero error source codes. The scheme of Appendix C-1 allows us to modify our code to give a locally decodable and update efficient variable-length compressor.

After the modification in Appendix C-1, $d(1)$ can increase by no more than 1. If the probability of error P_e is $o(1/n)$, then the expected update efficiency also remains $\Theta\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)$. If the original fixed-length code has rate $H(p_X)+\varepsilon$ and probability of error P_e, then the new code has rate $(1-P_e)(H(p_X)+\varepsilon)+P_e$, which asymptotically approaches $H(p_X)+\varepsilon$ if $P_e = o(1)$.

I. Universal compression using Lempel-Ziv as a subcode

We show that the performance by the coding scheme described above can be achieved even if the source p_X is unknown to the encoder and local decoder/updater.

Let $\mathcal{C}_i$ denote the $(n_i, k_i/n_i)$ fixed-length compression scheme at level i in Section IV-B. In Section IV-B, we chose $\mathcal{C}_0$ to be the typical set compressor. In this section, we will replace this with a fixed-length compressor based on LZ78 [12].

We first redefine what it means for a sequence to be typical.

Definition IV.3. *For any $\delta > 0$ and $b \in \mathbb{Z}_+$, we say that $x^b \in \mathcal{X}^b$ is δ-LZ typical with respect to p_X if the length of the LZ78 codeword corresponding to x^b, denoted $\ell_{LZ}(x^b)$, is less than $b(H(p_X)+\delta)$.*

The above notion of typicality leads to a natural computationally-efficent fixed-length compression scheme.

Definition IV.4 (Fixed-length compression scheme derived from LZ78)**.** *Let $\mathcal{T}^b_{\delta,LZ}$ denote the set of all sequences that are δ-LZ typical with respect to p_X. Associated with this is a natural $(b, H(p_X)+\delta)$ fixed-length compression scheme which we denote $\mathcal{C}_{LZ}(b, H(p_X), \delta)$: For any $x^b \in \mathcal{X}^b$, the corresponding codeword in $\mathcal{C}_{LZ}(b, H(p_X), \delta)$ is given by*

$$y^{b(H(p_X)+\delta)} = \begin{cases} [1, \mathrm{ENC}_{\mathrm{LZ}}(x^b),\ 0^{b(H(p_X)+\delta)-\ell_{LZ}(x^b)}] & \text{if } \ell_{LZ}(x^b) < b(H(p_X)+\delta) \\ 0^{b(H(p_X)+\delta)} & \text{otherwise,} \end{cases}$$

where $\mathrm{ENC}_{\mathrm{LZ}}$ denotes the LZ78 encoder.

We can now describe the modifications required in the scheme of Section IV-B in order to achieve universal compression.

The universal compressor with locality: The global encoder uses the empirical estimate of p_X to choose b_0 and k_0, which are encoded in the first $\Theta(1)$ bits (the preamble) of the compressed sequence[7]. The parameter ε_0 can be fixed beforehand, or otherwise stored in the preamble. The rest of the codeword is generated as in Section IV-B but with $\mathcal{C}_0$ being $\mathcal{C}_{LZ}$.

The following theorem summarizes the main result of this section, and completes the proof of Theorem III.2. The proof uses some technical lemmas that are formally proved in Appendix B.

Theorem IV.1. *Fix a small $\varepsilon > 0$. The coding scheme in Section IV-B with $\mathcal{C}_0$ chosen to be $\mathcal{C}_{LZ}$ achieves rate*

$$R = H(p_X) + \varepsilon,$$

probability of error

$$\Pr[\mathrm{DEC}(\mathrm{ENC}(X^n)) \neq X^n] = 2^{-2^{\Omega(\sqrt{\log n})}},$$

and average local decodability and update efficiency

$$d(s) \leqslant \begin{cases} \alpha_1 \frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon} & \text{if } s = O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right) \\ \alpha_1' s & \text{if } s = \Omega\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right) \end{cases},$$

$$u(s) \leqslant \begin{cases} \alpha_2 \frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon} & \text{if } s = O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right) \\ \alpha_2' s & \text{if } s = \Omega\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right) \end{cases},$$

where $\alpha_1, \alpha_1', \alpha_2, \alpha_2'$ are constants independent of n, ε but dependent on p_X.

The overall computational complexity of encoding and decoding X^n is $O(n \log n)$.

[7]One way to store b_0 (resp. k_0) is by $1^{b_0}0^{k_b-b_0}$ (resp. $1^{k_0}0^{k_b-k_0}$) for a large enough predetermined value of $k_b = o(n)$.

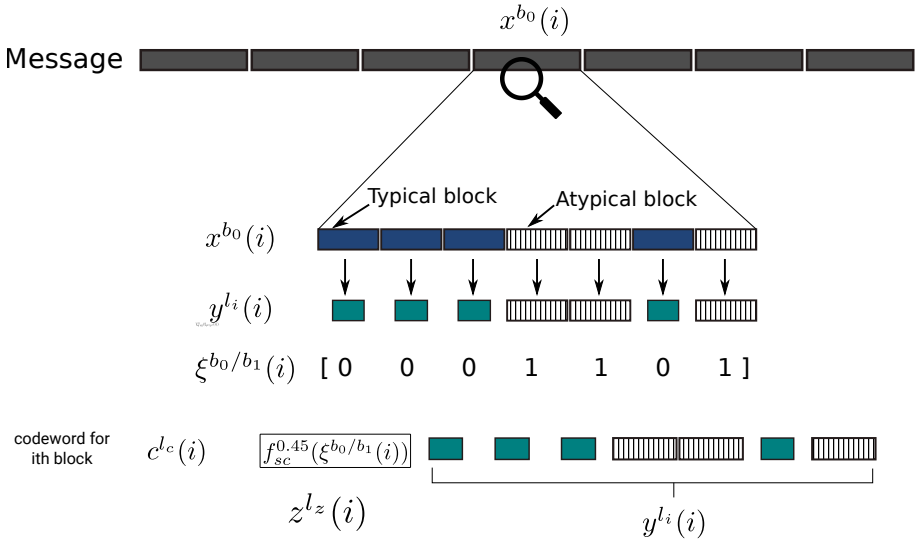

Fig. 4: Compression of each block as described in Section V-1. Typical subblocks are compressed to $\approx b_1 H(p_X)$ bits, while atypical subblocks are stored without compression. The address of $y^{l_i}(i)$ on disk can be easily computed using rank and select operations on $\xi^{b_0/b_1}(i)$.

Proof. We set $k_0 = b_0(H(p_X) + \xi(\varepsilon_0, b_0))$, where

$$\xi(\varepsilon_0, b_0) \coloneqq \left(2 + \max_{a \in \mathcal{X}} \log \frac{1}{p_X(a)}\right)\varepsilon_0 + \frac{c \log\log b_0}{\log b_0}.$$

In the above, c denotes the constant that appears in Lemma B.1. Clearly, $k_0 = b_0(H(p_X) - \Theta(\varepsilon_0))$. At level 0, we use $\mathcal{C}_0 = \mathcal{C}_{LZ}(b_0, H(p_X), \xi(\varepsilon_0, b_0))$. The rest of the compression scheme is exactly as in Section IV. From our choice of parameters and Lemma B.1, it is easy to see that $\ell_{LZ}(x^{b_0}(j)) \leqslant k_0 - 1$ as long as $x^{b_0}(j) \in \mathcal{T}_{\varepsilon_0}^{b_0}$. Therefore, the calculations in the proof of Theorem III.2 can be invoked to complete the proof.

The rate is $H(p_X) + \Theta(\varepsilon_0)$, while $d(s)$ and $u(s)$ are (up to constants depending only on p_X) the same as in Theorem III.2. □

V. Proof of Theorem III.3

We now describe our algorithm which achieves *worst-case* local decodability and *average* update efficiency of $O(\log\log n)$. The basic idea is the following: We partition the message symbols into blocks of $O(\log\log n)$ symbols each, and compress each block using a simple variable-length compression scheme. To locate the codeword corresponding to each block, we separately store a data structure that takes $o(n)$ space. This data structure allows us to efficiently query certain functions of the message.

For ease of exposition, we assume that p_X is known. Universality can be achieved by replacing the typical set compressor in our scheme with a universal compressor such as LZ78 (as we did in Section IV-I).

Definition V.1 (Rank)**.** *Let z^m denote an m-length binary sequence. For any $i \in [m]$, the* rank, $\texttt{RNK}_i(z^m)$ *denotes the number of 1's in (or the Hamming weight of) z_1^i.*

Our construction for efficient local decoding and updates is based on the existence of compressed data structures that allow query-efficient computation of rank. Let $h(\cdot)$ denote the binary entropy function.

Lemma V.1 ([33])**.** *Let m be a sufficiently large integer, and fix $0 < \alpha < 1/2$. Then, there exists a mapping $f_{\mathrm{sc}}^{(\alpha)} : \{0,1\}^m \to \{0,1\}^{m(h(\alpha)+o(1))}$ such that for every $x^n \in \{0,1\}^m$ with Hamming weight at most αm,*

- *x^m can be recovered uniquely from $f_{\mathrm{sc}}^{(\alpha)}(x^m)$*
- *For every $1 \leqslant i \leqslant m$, the rank $\texttt{RNK}_i(x^m)$ can be computed by probing at most $O(\log m)$ bits of $f_{\mathrm{sc}}^{(\alpha)}(x^m)$ in the worst case.*

1) Encoding: We partition the source sequence x^n into blocks of $b_0 = O(\log n)$ symbols each: $x^n = (x^{b_0}(1), \ldots, x^{b_0}(n/b_0))$. We further subdivide each block into subblocks of b_1 symbols each, *i.e.*, $x^{b_0}(i)$ is partitioned into $(x^{b_1}(i,1), \ldots, x^{b_1}(i, b_0/b_1))$. The symbols $x^{b_0}(i)$'s are encoded independently of each other using a fixed length code which has a vanishingly small probability of error. The codeword for each block consists of two parts:

- Corresponding to every $x^{b_1}(i,j)$, we generate $y^{l_{ij}}(i,j)$, which is given by

$$y^{l_{ij}}(i,j) = \begin{cases} \mathrm{index}(x^{b_1}(i,j); \mathcal{T}_{\varepsilon_0}^{b_1}) & \text{if } x^{b_1}(i,j) \in \mathcal{T}_{\varepsilon_0}^{b_1} \\ x^{b_1}(i,j) & \text{otherwise.} \end{cases}$$

Observe that the above is not a fixed-length code. The length of the (i,j)th codeword l_{ij} is equal to $\log|\mathcal{T}_{\varepsilon_0}^{b_1}|$ if $x^{b_1}(i,j)$ is typical and b_1 otherwise. Additionally, let

$$\xi(i,j) = \begin{cases} 0 & \text{if } x^{b_1}(i,j) \in \mathcal{T}_{\varepsilon_0}^{b_1} \\ 1 & \text{otherwise.} \end{cases}$$

be an indicator of whether the (i,j)th block $x^{b_1}(i,j)$ is atypical. Let $\xi^{b_0/b_1}(i) = (\xi(i,1),\dots\xi(i,b_0/b_1))$ and define

$$z^{\ell_z}(i) \stackrel{\text{def}}{=} \begin{cases} f_{\text{sc}}^{(\varepsilon_1)}(\xi^{b_0/b_1}(i)) & \text{if } \xi^{b_0/b_1} \text{ has Hamming weight at most } \varepsilon_0 b_0/b_1 \\ 0^{\ell} & \text{otherwise,} \end{cases}$$

where f_{sc} is the compressed data structure in Lemma V.1. Let $\ell_y \stackrel{\text{def}}{=} (1-2\varepsilon_0)(H(p_X)+\varepsilon)b_0 + 2\varepsilon_0 b_0 \log|\mathcal{X}|$ and $l_i' = \ell_y - \sum_j l_{ij}$

$$y^{\ell_y}(i) \stackrel{\text{def}}{=} \begin{cases} (y^{l_{i1}}(i,1),\dots,y^{l_{ib_0/b_1}}(i,b_0/b_1),0^{l_i'}) & \text{if } \sum_j l_{ij} \leqslant \ell_y \\ 0^{\ell_y} & \text{otherwise.} \end{cases}$$

The second case would correspond to an error.

- The codeword $c^{\ell_c}(i)$ corresponding to $x^{b_0}(i)$ is a sequence of length $\ell_c = \ell_y + \ell_z$, and is equal to the concatenation of $z^{\ell_z}(i)$ and $y^{\ell_y}(i)$.

Example V.1 (Figure 4)**.** *Consider the encoding of each b_0-length block as illustrated in Figure 4. In this example, $b_0/b_1 = 7$. Subblocks $4,5,7$ are atypical. Therefore, $y^{l_{ij}}(i,j) = x^{b_1}(i,j)$ and $l_{ij} = n_1$ for $j = 4,5,7$. The remaining subblocks are compressed using the typical set compressor. The indicator vector $\xi^6(i) = [0001101]$, and is compressed to get $z^{\ell_z}(i)$ using the scheme in Lemma V.1. The overall codeword for block i is the concatenation of $z^{\ell_z}(i)$ and $y^{l_{ij}}(i,j)$, $1 \leqslant j \leqslant 7$.*

2) Local decoding of a subblock: Our scheme allows us to locally decode an entire b_1-length subblock and local recovery of a single symbol is performed by locally decoding the subblock containing it.

Suppose that we want to locally decode $x^{b_1}(i,j)$. Our local decoder works as follows:

- Compute n_{atyp}, the number of atypical subblocks in the first j subblocks of the ith block. This is equal to $\texttt{RNK}_j(\xi^{b_0/b_1}(i))$ and can be obtained by probing $O(\log(b_0/b_1))$ bits of $z^{\ell_z}(i)$.
- Compute $\xi(i,j)$ from $z^{\ell_z}(i)$. This could be recovered by first decoding $\texttt{RNK}_{j+1}(\xi^{b_0/b_1}(i))$ and subtracting $\texttt{RNK}_j(\xi^{b_0/b_1}(i))$ from this. This tells us whether the block we we want to decode is atypical.
- Given the above information, it is easy to decode the (i,j)th block. Let $k_1 = n_{atyp}b_1 + (j-1-n_{atyp})b_1(H(p_X+\varepsilon_0))$.

$$\hat{y}^{\ell_{ij}}(i,j) = \begin{cases} y_{k_1}^{k_1+b_1(H(p_X+\varepsilon_0)} & \text{if } \xi(i,j) = 0 \\ y_{k_1}^{k_1+b_1} & \text{otherwise.} \end{cases}$$

The estimate of the message block $x^{b_1}(i,j)$ is obtained by decompressing $\hat{y}^{\ell_{ij}}(i,j)$.

Let us revisit the previous example.

Example V.2 (Figure 4)**.** *Figure 4. Suppose that we are interested in recovering $x^{n_1}(i,5)$.*

The local decoder first finds $\texttt{RNK}_4(z^{\ell_z}(i)) = 1$ and $\texttt{RNK}_5(z^{\ell_z}(i)) = 2$ using the probing scheme in Lemma V.1. This reveals that $x^{n_1}(i,5)$ is atypical, and one out of four subblocks prior to $x^{n_1}(i,5)$ is atypical. The starting location of $x^{n_1}(i,5)$ in $y^{\ell_i}(i)$ is $m \stackrel{def}{=} 3n_1(H(p_X)+\varepsilon) + n_1 + 1$. The desired block is recoverable from $y_m^{m+n_1-1}(i)$.

3) Update algorithm: We consider update of $x^{b_1}(i,j)$ with a new symbol denoted $\widetilde{x}^{b_1}$. Let

$$\tilde{y}^{\ell} = \begin{cases} (\text{index}(\widetilde{x}^{b_1}); \mathcal{T}_{\varepsilon_0}^{b_1}) & \text{if } \widetilde{x}^{b_1} \in \mathcal{T}_{\varepsilon_0}^{b_1} \\ \widetilde{x}^{b_1} & \text{otherwise.} \end{cases}$$

The update algorithm works as follows:

- Compute n_{atyp} and $x^{b_1}(i,j)$ by running the local decoding algorithm above.
- If both $\widetilde{x}^{b_1}$ and $x^{b_1}(i,j)$ are typical (or both atypical), then updating the codeword is trivial as it only requires replacing $y^{\ell_{ij}}(i,j)$ with $\tilde{y}^{\ell}$. In this case, only $O(\log b_0)$ bits need to be read and written in order to update the codeword.
- If only one of $\widetilde{x}^{b_1}$ and $x^{b_1}(i,j)$ is typical, then the entire code block $c^{\ell_c}(i)$ is rewritten with the encoding of

$$\widetilde{x}^{b_0} \stackrel{\text{def}}{=} (x^{b_1}(i,1),\dots,\widetilde{x}^{b_1},\dots,x^{b_1}(i,b_0/b_1)).$$

In this case, a total of $O(b_0)$ bits need to be read and modified to effect the update.

A. Proof of Theorem III.3

We choose $b_0 = c_0 \log n$ and $b_1 = c_1 \log\log n$, where c_0 and c_1 are constants that need to be chosen appropriately. The probability that a subblock is atypical is $p_0 = 2^{-\Theta(\varepsilon_0^2 b_1)}$. We choose c_1 so that this probability is at most $1/\log^2 n$. Recall that a b_0-block is in error if more than $2\varepsilon_0$ fraction of the subblocks are atypical. Using Chernoff bound, this is at most $p_1 = 2^{-\Omega(b_0/b_1)}$. We can choose c_0 so as to ensure that p_1 is at most n^{-2}. The probability that the overall codeword is in error is at most $np_1 = o(1)$.

We therefore have a fixed-length compression scheme with a vanishingly small probability of error. The worst-case local decodability is $d_{\mathrm{wc}}(1) = \Theta(b_1)$. Updating a subblock might lead to a typical block becoming atypical (or vice versa). Therefore, the average update efficiency is

$$u(1) = (1-p_1)\Theta(b_1) + p_1\Theta(b_0) = O(\log\log n).$$

This gives the first part of the theorem.

Any s-length substring is contained in at most $\lceil s/b_1 \rceil + 1$ subblocks of size b_1. We can therefore locally decode/update any m-length substring by separately running the local decoding/update algorithm for each of the $\lceil s/b_1 \rceil + 1$ subblocks. Therefore,

$$\begin{aligned} d_{\mathrm{wc}}(s) &\leqslant \left(\left\lceil \frac{s}{b_1} \right\rceil + 1 \right) d_{\mathrm{wc}}(1) \\ &\leqslant \begin{cases} 2d_{\mathrm{wc}}(1) & \text{if } s \leqslant b_1 \\ s(H(p_X)+\varepsilon) + 2d_{\mathrm{wc}}(1) & \text{otherwise.} \end{cases} \end{aligned}$$

The calculation of $u_{\mathrm{wc}}(s)$ proceeds identically. This completes the proof of the second part of Theorem III.3. □

VI. Concluding remarks

In this paper, we gave an explicit, computationally efficient entropy-achieving scheme that achieves constant *average* local decodability and update efficiency. Our scheme also allows efficient local decoding and update of contiguous substrings. For $s = \Omega(1/\varepsilon^2)$, both $d(s)$ and $u(s)$ grow as $\Theta(s)$, where the implied constant is independent of n and ε.

It still remains an open problem as to whether $(d_{\mathrm{wc}}(1), u_{\mathrm{wc}}(1)) = (\Theta(1), \Theta(1))$ is achievable. We described a scheme with $(d_{\mathrm{wc}}(1), u(1)) = (O(\log\log n), O(\log\log n))$. Even showing that $(d_{\mathrm{wc}}(1), u_{\mathrm{wc}}(1)) = (O(\log\log n), O(\log\log n))$ is achievable would be an interesting step in this direction.

Although we did not optimize the hidden constants in our analysis, the dependence on ε in our scheme cannot be improved by using tighter bounds. This is because we used a lossless compression scheme at level 0, and we require $b_0 = \Omega(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon})$ to guarantee concentration. Mazumdar *et al.* [40] used a slightly different approach, and gave a two-level construction with a lossy source code at the zeroth level. This allowed them to achieve $d_{\mathrm{wc}}(1) = \Theta(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$. Finding the right dependence of $(d_{\mathrm{wc}}(1), u_{\mathrm{wc}}(1))$ or $(d(1), u(1))$ on ε is an interesting open question.

Appendix A
Preliminary lemmas for the proof of Theorem III.2

Lemma A.1. *Let X^b be a b-length i.i.d. sequence where the components are drawn according to p_X. For any positive α, and $0 < \varepsilon < 1/2$, if*

$$b \geqslant 3(\alpha + \log|\mathcal{X}|)\left(\max_{a\in\mathcal{X}} \frac{1}{p_X(a)}\right)\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right),$$

then

$$\Pr[X^b \notin \mathcal{T}_\varepsilon^b] \leqslant \varepsilon^\alpha.$$

Moreover,

$$|\mathcal{T}_\varepsilon^b| \leqslant 2^{b(H(p_X)+\varepsilon)}.$$

Proof. The first part can be easily derived using Chernoff and union bounds. The second part is a standard property of typical sets. See, *e.g.*, the book by El Gamal and Kim [55] for a proof. □

Lemma A.2. *Let $\delta_{i-1\to i}$ denote the probability that the message block from level $i-1$, say $x^{n_{i-1}}(j, i-1)$, is not the all-$\diamond$ block. If $\varepsilon_0 < 1/2$, $\delta_{i-1\to i} \leqslant \varepsilon_i^4$, and*

$$b_0 \geqslant 3(8 + \log|\mathcal{X}|)\left(\max_{a\in\mathcal{X}} \frac{1}{p_X(a)}\right)\left(\frac{1}{\varepsilon_0^2}\log\frac{1}{\varepsilon_0}\right),$$

Then,

$$\delta_{i\to i+1} \leqslant \varepsilon_i^{\beta 2^i}, \tag{4}$$

where

$$\beta = 9(8 + \log|\mathcal{X}|)\left(\max_{a\in\mathcal{X}} \frac{1}{p_X(a)}\right)\left(\frac{1}{\varepsilon_0}\log\frac{1}{\varepsilon_0}\right) \tag{5}$$

This implies that

$$\delta_{i\to i+1} \leqslant \varepsilon_{i+1}^4. \tag{6}$$

Proof. Recall that a message block from level i is not a $\diamond$-block only if there are more than $\varepsilon_i b_i$ non-$\diamond$-blocks from level $i-1$. Therefore,

$$\begin{aligned}\delta_{i\to i+1} &\leqslant \binom{b_i}{\varepsilon_i b_i}\delta_{i-1\to i}^{\varepsilon_i b_i}\\ &\leqslant \left(\frac{\delta_{i-1\to i}}{\varepsilon_i}\right)^{\varepsilon_i b_i}\\ &\leqslant \varepsilon_i^{3\varepsilon_i b_i}\end{aligned} \tag{7}$$

However,

$$\varepsilon_i b_i = \varepsilon_0 2^{-i} b_0 2^{2i} = \varepsilon_0 b_0 2^i.$$

Using the lower bound for b_0 in the above equation and substituting in (7) gives us (4). Inequality (6) follows from (4) by observing that $\varepsilon_0 < 1/2$. □

The probability of error therefore decays quasiexponentially in n as described by the following corollary.

Corollary A.1. *Suppose we use the parameters as defined in Lemma A.2, and choose $b_i = 2^{2i}b_0$ and $\varepsilon_i = \varepsilon_0/2^i$. Then, the probability that the encoder makes an error, i.e., that the message is not compressed within $\ell_{\max}$ levels, is $2^{-\Omega(\ell_{\max}2^{\ell_{\max}})}$. If the number of levels is $\Theta(\sqrt{\log n})$, then this is $2^{-2^{\Omega(\sqrt{\log n})}}$.*

The following lemma will be used to compute the average local decodability and update efficiency.

Lemma A.3. *Let $\delta_{i\to i+1}^{(1)}$ be the conditional probability that the message block from level i, say $x^{n_i}(1,i)$ is not the all-$\diamond$ block given that a fixed block from level $i-1$, say $x^{n_{i-1}}(1,i-1)$, is not a $\diamond$-block. If $\varepsilon_0 < 1/2$, $\delta_{i-1\to i}^{(1)} \leqslant \varepsilon_i^4$, and*

$$b_0 \geqslant 3(8 + \log|\mathcal{X}|)\left(\max_{a\in\mathcal{X}} \frac{1}{p_X(a)}\right)\left(\frac{1}{\varepsilon_0^2}\log\frac{1}{\varepsilon_0}\right).$$

Then,

$$\delta_{i\to i+1}^{(1)} \leqslant \varepsilon_i^{\beta 2^{i-1}}, \tag{8}$$

where $\beta = 9(8 + \log|\mathcal{X}|)\left(\max_{a\in\mathcal{X}} \frac{1}{p_X(a)}\right)\left(\frac{1}{\varepsilon_0}\log\frac{1}{\varepsilon_0}\right)$. This implies that

$$\delta_{i\to i+1}^{(1)} \leqslant \varepsilon_{i+1}^4. \tag{9}$$

Proof. Clearly,

$$\delta_{i\to i+1}^{(1)} \leqslant \binom{b_i - 1}{\varepsilon_i b_i - 1}\left(\delta_{i-1\to i}^{(1)}\right)^{\varepsilon_i b_i - 1}.$$

The remainder of the proof is almost identical to that of Lemma A.2, and we skip the details. □

The following result will be useful when bounding the average local decodability in Lemma IV.1.

Lemma A.4. *For all $i \geqslant 1$ and $b_0 \geqslant 3$, we have*

$$(i+1)b_0^{i+1}2^{i(i+1)}\varepsilon_i^{\beta 2^{i-1}} \leqslant \varepsilon_0^i.$$

Proof. Let $\chi(i) \stackrel{\text{def}}{=} (i+1)b_0^{i+1}2^{i(i+1)}\varepsilon_i^{\beta 2^{i-1}}$ for $i \geqslant 1$.

Note that β, defined in (5), is equal to $3b_0$. Therefore,

$$\chi(1) = 8b_0^2\left(\frac{\varepsilon_0}{2}\right)^{3b_0} < \varepsilon_0,$$

where the last step holds for all $b_0 \geqslant 3$. For any $i \geqslant 2$,

$$\begin{aligned}\frac{\chi(i)}{\chi(i-1)} &= \left(\frac{i+1}{i}\right) b_0 2^{2i} \left(\frac{\varepsilon_0}{2^i}\right)^{3b_0(2^{i-1}-2^{i-2})} \\ &\leqslant \left(\frac{i+1}{i}\right) b_0 2^{2i} \left(\frac{\varepsilon_0}{2^i}\right)^{6b_0} \\ &\leqslant 2b_0 \left(\frac{\varepsilon_0}{2^i}\right)^{6b_0} \\ &\leqslant \varepsilon_0.\end{aligned}$$

Therefore, $\xi(i) \leqslant \varepsilon_0^i$. □

APPENDIX B
PRELIMINARY LEMMAS FOR THE PROOF OF THEOREM IV.1

In order to compute bounds on the rate and expected local decodability and update efficiency, we must find the probability that the length of an LZ78 codeword exceeds a certain amount. To help us with that, we have the following lemma:

Lemma B.1 ([54])**.** *Let $\mathcal{X}$ be a finite alphabet and b be a positive integer. For any $x^b \in \mathcal{X}^b$, let $\ell_{LZ}(x^b)$ denote the length of the LZ78 codeword for x^b. For every $k \in \mathbb{Z}_+$, we have*

$$\ell_{LZ}(x^b) \leqslant bH_k(x^b) + \frac{ckb \log\log b}{\log b},$$

where $H_k(x^b)$ denotes the kth order empirical entropy of the sequence x^b, and c is an absolute constant.

The above lemma says that the length of the LZ78 codeword is close to the empirical entropy of the string. The following lemma lets us conclude that if a sequence is typical, then the empirical entropy is close to the true entropy.

Lemma B.2. *Fix any two probability mass functions p, q on $\mathcal{X}$, and $0 < \varepsilon < 1/2$. If $|p(a) - q(a)| \leqslant \varepsilon p(a)$ for all $a \in \mathcal{X}$, then*

$$|H(p) - H(q)| \leqslant \left(2 + \max_{a\in\mathcal{X}} \log \frac{1}{p(a)}\right) \varepsilon.$$

Proof. Consider

$$\begin{aligned}\Delta_a &\coloneqq p(a)\log p(a) - q(a)\log q(a) \\ &= p(a)\log p(a) - q(a)\log p(a) + q(a)\log p(a) - q(a)\log q(a) \\ &= (p(a) - q(a))\log p(a) - q(a)\log\frac{q(a)}{p(a)}\end{aligned}$$

However,

$$\begin{aligned}|H(p) - H(q)| &\leqslant \sum_a |\Delta_a| \\ &\leqslant \sum_a \left(|p(a) - q(a)| \log \frac{1}{p(a)} + q(a)\left|\log\frac{q(a)}{p(a)}\right|\right) \\ &\leqslant \varepsilon \max_a \log\frac{1}{p(a)} + \log\frac{1}{1-\varepsilon}.\end{aligned}$$

For $\varepsilon < 1/2$, we have $\log\frac{1}{1-\varepsilon} \leqslant 2\varepsilon$. Using this in the above completes the proof. □

APPENDIX C
FIXED V/S VARIABLE-LENGTH COMPRESSION

We briefly show how to achieve zero-error data compression and still achieve the performance stated in Theorems III.2 and III.3. This is obtained by using a variable length code instead of a fixed-length code.

Definition C.1 (Variable-length compression)**.** *An (n, R) variable-length compression scheme is a pair of maps* $(\mathrm{ENC}, \mathrm{DEC})$ *consisting of*

- *an encoder* $\mathrm{ENC} : \mathcal{X}^n \to \{0,1\}^*$*, and*
- *a decoder* $\mathrm{DEC} : \{0,1\}^* \to \mathcal{X}^n$ *satisfying*

$$\mathrm{DEC}(\mathrm{ENC}(X^n)) = X^n, \quad \forall X^n \in \mathcal{X}^n$$

For any $Y^l \in \{0,1\}^*$, let $\ell(Y^l)$ denote the length of the sequence Y^l. The quantity R is the rate of the code, and is defined to be

$$R \stackrel{def}{=} \frac{1}{n}\mathbb{E}[\ell(\mathrm{ENC}(X^n))]$$

where the averaging is over the randomness in the source.

It is generally desired for a variable-length source code be prefix free: For every distinct pair of inputs $X^n, Y^n \in \mathcal{X}^n$, the codeword $\mathrm{ENC}(X^n)$ must not be a prefix of $\mathrm{ENC}(Y^n)$.

1) Converting a fixed-length compressor to a prefix-free variable-length compressor: Given any (n, R) fixed-length compression scheme $(\mathrm{ENC}_{\mathrm{fix}}, \mathrm{DEC}_{\mathrm{fix}})$ with a probability of error $P_e = o(1)$, it is easy to construct a prefix-free $(n, R + o(1))$ variable-length compressor $(\mathrm{ENC}_{\mathrm{var}}, \mathrm{DEC}_{\mathrm{var}})$. The following is one-such construction:

$$\mathrm{ENC}_{\mathrm{var}}(X^n) \stackrel{\text{def}}{=} \begin{cases} (0, \mathrm{ENC}_{\mathrm{fix}}(X^n)) & \text{if } \mathrm{DEC}_{\mathrm{fix}}(\mathrm{ENC}_{\mathrm{fix}}(X^n)) = X^n \\ (1, X^n) & \text{otherwise.} \end{cases}$$

Clearly, the compressor is prefix free. The rate of $(\mathrm{ENC}_{\mathrm{var}}, \mathrm{DEC}_{\mathrm{var}})$ is equal to

$$\begin{aligned} R_{\mathrm{var}} &= 1/n + R \times (1 - P_e) + \log|\mathcal{X}| \times P_e \\ &= R + o(1). \end{aligned}$$

For all $s \geqslant 1$, the local decodability of the new variable-length scheme $d(s), d_{\mathrm{wc}}(s)$ is at most 1 more than that of the original fixed-length scheme, and $u(s), u_{\mathrm{wc}}(1)$ can increase by at most 2.

Due to the above transformation, we have devoted most of our attention to constructing fixed-length compression schemes.

REFERENCES

[1] S. Vatedka and A. Tchamkerten, "Local decoding and update of compressed data," in *Proceedings of the 2019 IEEE International Symposium on Information Theory (ISIT)*, Paris, France, 2019.

[2] D. Pavlichin, T. Weissman, and G. Mably, "The quest to save genomics: Unless researchers solve the looming data compression problem, biomedical science could stagnate," *IEEE Spectrum*, vol. 55, no. 9, pp. 27–31, 2018.

[3] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information sciences*, vol. 275, pp. 314–347, 2014.

[4] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information systems*, vol. 47, pp. 98–115, 2015.

[5] M. P. Ball, J. V. Thakuria, A. W. Zaranek, T. Clegg, A. M. Rosenbaum, X. Wu, M. Angrist, J. Bhak, J. Bobe, M. J. Callow *et al.*, "A public resource facilitating clinical use of genomes," *Proceedings of the National Academy of Sciences*, vol. 109, no. 30, pp. 11 920–11 927, 2012.

[6] U. consortium *et al.*, "The uk10k project identifies rare variants in health and disease," *Nature*, vol. 526, no. 7571, p. 82, 2015.

[7] J. M. Gaziano, J. Concato, M. Brophy, L. Fiore, S. Pyarajan, J. Breeling, S. Whitbourne, J. Deen, C. Shannon, D. Humphries *et al.*, "Million veteran program: a mega-biobank to study genetic influences on health and disease," *Journal of clinical epidemiology*, vol. 70, pp. 214–223, 2016.

[8] E. E. Schadt, M. D. Linderman, J. Sorenson, L. Lee, and G. P. Nolan, "Computational solutions to large-scale data management and analysis," *Nature reviews genetics*, vol. 11, no. 9, p. 647, 2010.

[9] M. Vivien, "The big challenges of big data," *Nature*, vol. 498, p. 255, June 2013.

[10] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson, "Big data: astronomical or genomical?" *PLoS biology*, vol. 13, no. 7, p. e1002195, 2015.

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

[12] ——, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[13] M. C. Brandon, D. C. Wallace, and P. Baldi, "Data structures and compression algorithms for genomic sequence data," *Bioinformatics*, vol. 25, no. 14, pp. 1731–1738, 2009.

[14] S. Deorowicz and S. Grabowski, "Robust relative compression of genomes with random access," *Bioinformatics*, vol. 27, no. 21, pp. 2979–2986, 2011.

[15] A. J. Cox, M. J. Bauer, T. Jakobi, and G. Rosone, "Large-scale compression of genomic sequence databases with the burrows–wheeler transform," *Bioinformatics*, vol. 28, no. 11, pp. 1415–1419, 2012.

[16] S. Deorowicz, A. Danek, and S. Grabowski, "Genome compression: a novel approach for large collections," *Bioinformatics*, vol. 29, no. 20, pp. 2572–2578, 2013.

[17] K. Tatwawadi, M. Hernaez, I. Ochoa, and T. Weissman, "GTRAC: Fast retrieval from compressed collections of genomic variants," *Bioinformatics*, vol. 32, no. 17, pp. i479–i486, 2016.

[18] M. Patrascu, "Succincter," in *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2008, pp. 305–313.

[19] Y. Dodis, M. Patrascu, and M. Thorup, "Changing base without losing space," in *Proceedings of the forty-second ACM symposium on Theory of computing*. ACM, 2010, pp. 593–602.

[20] J. I. Munro and Y. Nekrich, "Compressed data structures for dynamic sequences," in *Algorithms-ESA 2015*. Springer, 2015, pp. 891–902.

[21] R. Raman and S. S. Rao, "Succinct dynamic dictionaries and trees," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2003, pp. 357–368.

[22] V. Chandar, D. Shah, and G. W. Wornell, "A locally encodable and decodable compressed data structure," in *Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2009, pp. 613–619.

[23] V. B. Chandar, "Sparse graph codes for compression, sensing and secrecy," Ph.D. dissertation, MIT, 2010.

[24] A. Dutta, R. Levi, D. Ron, and R. Rubinfeld, "A simple online competitive adaptation of lempel-ziv compression with efficient random access support," in *Proceedings of the Data Compression Conference (DCC)*. IEEE, 2013, pp. 113–122.

[25] P. Bille, G. M. Landau, R. Raman, K. Sadakane, S. R. Satti, and O. Weimann, "Random access to grammar-compressed strings," in *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2011, pp. 373–389.

[26] E. Viola, O. Weinstein, and H. Yu, "How to store a random walk," *arXiv preprint arXiv:1907.1087*, 2019.

[27] K. Sadakane and R. Grossi, "Squeezing succinct data structures into entropy bounds," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics, 2006, pp. 1230–1239.

[28] R. González and G. Navarro, "Statistical encoding of succinct data structures," in *Annual Symposium on Combinatorial Pattern Matching*. Springer, 2006, pp. 294–305.

[29] P. Ferragina and R. Venturini, "A simple storage scheme for strings achieving entropy bounds," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 690–696.

[30] S. Kreft and G. Navarro, "LZ77-like compression with fast random access," in *2010 Data Compression Conference*. IEEE, 2010, pp. 239–248.

[31] V. Mäkinen and G. Navarro, "Dynamic entropy-compressed sequences and full-text indexes," in *Annual Symposium on Combinatorial Pattern Matching*. Springer, 2006, pp. 306–317.

[32] J. Jansson, K. Sadakane, and W.-K. Sung, "Cram: Compressed random access memory," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2012, pp. 510–521.

[33] R. Grossi, R. Raman, S. S. Rao, and R. Venturini, "Dynamic compressed strings with random access," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2013, pp. 504–515.

[34] G. Navarro and Y. Nekrich, "Optimal dynamic sequence representations," *SIAM Journal on Computing*, vol. 43, no. 5, pp. 1781–1806, 2014.

[35] P. K. Nicholson, V. Raman, and S. S. Rao, "A survey of data structures in the bitprobe model," in *Space-Efficient Data Structures, Streams, and Algorithms*. Springer, 2013, pp. 303–318.

[36] H. Buhrman, P. B. Miltersen, J. Radhakrishnan, and S. Venkatesh, "Are bitvectors optimal?" *SIAM Journal on Computing*, vol. 31, no. 6, pp. 1723–1744, 2002.

[37] M. Lewenstein, J. I. Munro, P. K. Nicholson, and V. Raman, "Improved explicit data structures in the bitprobe model," in *European Symposium on Algorithms*. Springer, 2014, pp. 630–641.

[38] M. Garg and J. Radhakrishnan, "Set membership with a few bit probes," in *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2015, pp. 776–784.

[39] ——, "Set Membership with Non-Adaptive Bit Probes," in *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), H. Vollmer and B. Vallee, Eds., vol. 66. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[40] A. Mazumdar, V. Chandar, and G. W. Wornell, "Local recovery in data compression for general sources," in *Proceedings of the 2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 2984–2988.

[41] K. Tatwawadi, S. Bidokhti, and T. Weissman, "On universal compression with constant random access," in *Proceedings of the 2018 IEEE International Symposium on Information Theory*, 2018, pp. 891–895.

[42] A. Makhdoumi, S.-L. Huang, M. Médard, and Y. Polyanskiy, "On locally decodable source coding," in *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 4394–4399.

[43] A. Pananjady and T. A. Courtade, "The effect of local decodability constraints on variable-length compression," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2593–2608, 2018.

[44] A. Montanari and E. Mossel, "Smooth compression, Gallager bound and nonlinear sparse-graph codes," in *Proceedings of the 2008 IEEE International Symposium on Information Theory*. IEEE, 2008, pp. 2474–2478.

[45] L. R. Varshney, J. Kusuma, and V. K. Goyal, "On palimpsests in neural memory: An information theory viewpoint," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 2, no. 2, pp. 143–153, 2016.

[46] ——, "Malleable coding for updatable cloud caching," *IEEE Transactions on Communications*, vol. 64, no. 12, pp. 4946–4955, 2016.

[47] A. Mazumdar and S. Pal, "Semisupervised clustering, AND-queries and locally encodable source coding," in *Advances in Neural Information Processing Systems*, 2017, pp. 6489–6499.

[48] S. Yekhanin *et al.*, "Locally decodable codes," *Foundations and Trends® in Theoretical Computer Science*, vol. 6, no. 3, pp. 139–255, 2012.

[49] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Transactions on Information theory*, vol. 58, no. 11, pp. 6925–6934, 2012.

[50] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4661–4676, 2014.

[51] V. R. Cadambe and A. Mazumdar, "Bounds on the size of locally recoverable codes," *IEEE transactions on information theory*, vol. 61, no. 11, pp. 5787–5794, 2015.

[52] A. Mazumdar, V. Chandar, and G. W. Wornell, "Update-efficiency and local repairability limits for capacity approaching codes," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 976–988, 2014.

[53] I. Tamo, A. Barg, and A. Frolov, "Bounds on the parameters of locally recoverable codes," *IEEE Transactions on information theory*, vol. 62, no. 6, pp. 3070–3083, 2016.

[54] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012.

[55] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge university press, 2011.