$O(\log\log n)$ Worst-Case Local Decoding and Update Efficiency for Data Compression

Shashank Vatedka*, Venkat Chandar[†], Aslan Tchamkerten[‡]
*Dept. of Electrical Engineering, Indian Institute of Technology Hyderabad, India
[†]DE Shaw, New York, USA
[‡]Dept. of Communications and Electronics, Telecom Paris, France

***Abstract*—This paper addresses the problem of data compression with local decoding and local update. A compression scheme has worst-case local decoding d_{wc} if any bit of the raw file can be recovered by probing at most d_{wc} bits of the compressed sequence, and has update efficiency of u_{wc} if a single bit of the raw file can be updated by modifying at most u_{wc} bits of the compressed sequence. This article provides an entropy-achieving compression scheme for memoryless sources that simultaneously achieves $O(\log\log n)$ local decoding and update efficiency. Key to this achievability result is a novel succinct data structure for sparse sequences which allows efficient local decoding and local update.**

Under general assumptions on the local decoder and update algorithms, a converse result shows that d_{wc} and u_{wc} must grow as $\Omega(\log\log n)$.

I. Introduction

Consider a source sequence X^n with independent and identically distributed (i.i.d.) components having probability mass function p_X on a finite alphabet $\mathcal{X}$. For simplicity we assume here that p_X is a known distribution over $\mathcal{X} = \{0,1\}$.[1]

A fixed-length compression scheme of rate R consists of a pair of algorithms, the encoder ENC and the decoder DEC. The encoder maintains for every X^n, a codeword $C^{nR} \in \{0,1\}^{nR}$ such that $\text{DEC}(C^{nR})$ is a good estimate of X^n. The probability of error of the compression scheme is defined as

$$P_{\text{glob}} \coloneqq \Pr[\text{DEC}(C^{nR}) \neq X^n].$$

From the source coding theorem, we know that there exist sequences of codes with rate arbitrarily close to the entropy $H(p_X)$ and error probability vanishing in n.

Our goal is to design a fixed-length compression scheme that additionally supports local encoding and decoding. A locally decodable and updatable compression scheme consists of a global encoder and decoder pair (ENC, DEC) and in addition, a local decoder and a local updater:

- **Local decoder:** A local decoder is an algorithm which given $i \in [n]$, probes (possibly adaptively) a small number of bits of C^{nR} to output $\widehat{X}_i$. Here, $\widehat{X}_i$ is the ith symbol of $\widehat{X}^n \overset{\text{def}}{=} \text{DEC}(C^{nR})$. The worst-case local decodability d_{wc} of the scheme is the maximum number of bits probed by the local decoder for any i. The average local decodability d is the expected number of bits (averaged over the source distribution) probed to recover any $\widehat{X}_i$.
- **Local updater:** A local updater is an algorithm which given $i \in [n]$ and $\widetilde{X}_i \in \{0,1\}$, adaptively reads and modifies a small number of bits of C^{nR} to give $\widetilde{C}^{nR}$ such that $\widetilde{C}^{nR} = \text{ENC}(X_1, \ldots, X_{i-1}\widetilde{X}_i, X_{i+1}, \ldots, X_n)$. In particular, the local updater has no prior knowledge of X^n or C^{nR} and must probe C^{nR} to obtain such information.
  The worst-case update efficiency u_{wc} is defined as the maximum of the sum of the bits read and written in order to update any i. Also, we assume that $\widetilde{X}_i$ is distributed according to p_X and is independent of X^n. Likewise, the average update efficiency u is the sum of the average number of bits probed and written in order to update any X_i.

It was recently shown in [1] that

$$(d, u) = (O(1), O(1))$$

is achievable. In that paper the authors also gave a separate compression scheme achieving

$$(d_{\text{wc}}, u) = (O(\log\log n), O(\log\log n)).$$

In particular, the question of whether

$$(d_{\text{wc}}, u_{\text{wc}}) = (O(\log\log n), O(\log\log n))$$

is achievable was left open. In this paper we answer this question in the affirmative. We also show that under certain additional assumptions on the local decoder and the local updater this locality is order optimal.

Our achievability proof is based on a novel succinct data structure for $O(b/\log b)$-sparse sequences of length b in the bitprobe model which for any $0 < \delta < 1$ takes space $O(\delta b)$ while enabling local decode and update using at most $O(\log b)$ and $O(\frac{1}{\delta}\log b)$ bit reads/writes respectively. Our restricted converse is based on an analysis of bipartite graphs that represent the encoding and decoding algorithms.

A. Prior work

Local decoding and update for entropy-achieving compression schemes have been studied mostly in isolation. The

[1] This can be generalized to the scenario where p_X is unknown to the encoder and decoder by first estimating p_X and then using this for designing the compression scheme as in [1]. Similarly, our results can be generalized to nonbinary alphabets.

problem of locally decodable source coding of random sequences has received attention very recently following [2], [3]. Mazumdar *et al.* [4] gave a fixed-length compressor of rate of $H(p_X)+\varepsilon$ with $d_{\mathrm{wc}}(1)=\Theta(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$. They also provided a converse result for non-dyadic sources: $d_{\mathrm{wc}}(1)=\Omega(\log(1/\varepsilon))$ for any compression scheme that achieves rate $H(p_X)+\varepsilon$. Similar results are known for variable length compression [5] and universal compression of sources with memory [6]. Likewise, there are compressors that achieve [7] rate $R=H(p_X)+\varepsilon$ and update efficiency $u_{\mathrm{wc}}=O(1)$.

In the computer science community, the literature has mostly focused on the word-RAM model [8], [9], [10], [11], [12], [13], where each operation (read/write/arithmetic operations) is on words of size $w=O(\log n)$ bits each, and the complexity is measured in terms of the number of word operations required for local decoding/update. However, in this case the number of bitprobes required is $\Omega(\log n)$. For random messages, it is almost trivial to obtain local decoding/update efficiency of $O(\log n)$ bitprobes by partitioning the n message symbols into blocks of size $O(\log n)$ and compressing each block separately.

II. Contributions

Before we present our main results we make a few observations aimed at justifying our model, and in particular the requirements we impose on the local decoder (see previous section).

Note that for any index i the local decoder must output $\widehat{X}_i$, the i-th estimate of the global decoder. We could potentially relax this constraint, by requiring that the local decoder produces an estimate $\widehat{X}_i^{(\mathrm{loc})}$, potentially different from $\widehat{X}_i$, with small error probability

$$P_{\mathrm{loc}} \overset{\mathrm{def}}{=} \max_i \Pr[\widehat{X}_i^{(\mathrm{loc})} \neq X_i].$$

As we argue next, if we impose P_{loc} to be small (but non-vanishing) with either no global decoder or with a separate global decoder that achieves vanishing error probability, then from a coding perspective the solution is essentially trivial.

1) Only local decodability and non-vanishing error: If we only require P_{loc} to be small without constraints on the global decoder, then we can easily achieve

$$(d_{\mathrm{wc}},u_{\mathrm{wc}})=\left(O\left(\log\frac{1}{P_{\mathrm{loc}}}\right),O\left(\log\frac{1}{P_{\mathrm{loc}}}\right)\right).$$

This can be obtained by partitioning the length$-n$ message into blocks of size $b_0=O(\log\frac{1}{P_{\mathrm{loc}}})$, and compressing each block using an entropy-achieving fixed-length compression scheme—notice that the probability of wrongly decoding any particular block vanishes exponentially with b_0. Hence, for any small but constant P_{loc} we can achieve

$$(d_{\mathrm{wc}},u_{\mathrm{wc}})=(O(1),O(1)).$$

2) Separate local and global decoders: Suppose that in addition to 1) we also want $P_{\mathrm{glob}}=o(1)$ using a separate global decoder to recover $\widehat{X}^n$. This can be obtained by using a low-density parity check (LDPC) code with $O(1)$ maximum variable and check node degrees. The codeword consists of two parts:

$$C^{nR}=(C^{n(R-\delta)}(1),C^{\delta n}(2)),$$

where $C^{n(R-\delta)}(1)$ is obtained as in the previous case by dividing the message into constant size b_0 blocks and separately encoding each, while $C^{\delta n}(2)$ is obtained as the syndrome (of the LDPC code) of the (Hamming) error vector between X^n and the decoding of $C^{n(R-\delta)}(1)$.

The local decoder only probes $C^{n(R-\delta)}(1)$, while the local updater needs to update both $C^{n(R-\delta)}(1)$ and $C^{n\delta}(2)$. Since we are using an LDPC code, $C^{n\delta}(2)$ can be updated using $O(1)$ bit modifications. Therefore,

$$(d_{\mathrm{wc}},u_{\mathrm{wc}})=(O(\log\frac{1}{P_{\mathrm{loc}}}),O(\log\frac{1}{P_{\mathrm{loc}}})).$$

The global decoder decodes both $C^{n(R-\delta)}(1)$ and $C^{\delta n}(2)$ and can recover X^n with $o(1)$ probability of error.

As we see, 1) and 2) are essentially trivial cases from a coding perspective. Note also that the solution to 2) generally depends on the rate of decay of P_{loc}. Requiring the local decoder to output $\widehat{X}_i$ removes this degree of freedom—since $P_{\mathrm{glob}}=o(1)$ implies $P_{\mathrm{loc}}=o(1)$—and, as we argue below, is more interesting from a coding perspective. This setup is perhaps more interesting also practically since we can parallelize global decoding. If we have a large number of parallel processors, then the runtime of global decoding can be made sublinear in n.

We will henceforth only consider compression schemes with local decoders that output $\widehat{X}_i$.

The main result of this article is the following:

Theorem II.1. *For any $\varepsilon>0$ there exists a compression scheme for Bernoulli(p) sources that achieves*

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

and the overall computational complexity of global encoding/decoding is quasilinear in n.

The above theorem is formally proved in Section III-A. The proof of the above theorem is based on a novel dynamic succinct data structure for sparse sequences that achieves $O(\log n)$ locality in the bitprobe model.

Lemma II.1. *Fix any $\delta>0$. For every $\beta=o(b/\log b)$, there exists a dynamic succinct data structure for b-length binary vectors of sparsity at most β with the following properties. Any such vector occupies at most $\delta b(1+o(1))$ bits, has worst-case local decoding $O(\log b)$ and worst-case update efficiency at most $O(\frac{1}{\delta}\log b)$.*

To prove a lower bound, we make three assumptions

(A1) *Global encoding and decoding using local algorithms:* We assume that C^{nR} is obtained by running the local

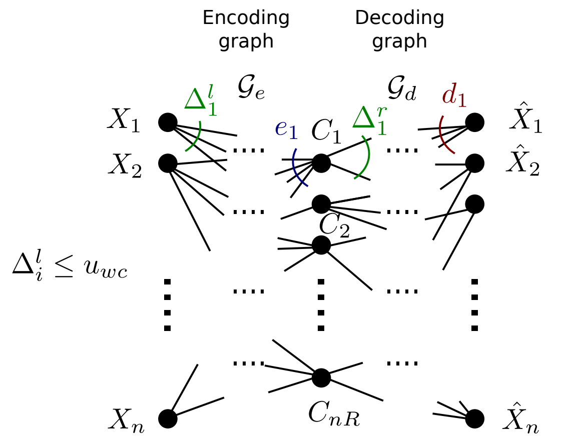

Fig. 1: Encoding and decoding graphs under assumptions (A1) and (A2). The degree of the ith left vertex in $\mathcal{G}_e$ is Δ_i^l, which is upper bounded by u_i, the update efficiency for message symbol i. The jth right vertex is the local encodability of the ith symbol e_j, defined in Sec. IV. The degree of a right vertex in $\mathcal{G}_d$ is equal to the local decoding of the ith symbol d_i.

updater on each message symbol, and $\widehat{X}^n$ by running the local decoder for each bit. In other words, there is no separate global encoder or decoder.

(A2) *Function assumption:* The local update function for the t'th update is a deterministic function of X^n, and is independent of the sequence of the previous $t-1$ updates. Likewise, the local decoder does not depend on the sequence of updates that have occurred previously.

(A3) *Bounded average-to-worst case influence:* For nonadaptive schemes, we can construct the corresponding local encoding graph $\mathcal{G}_e$ and the local decoding graph $\mathcal{G}_d$ as follows—see Fig. 1. Two vertices (i,j) in $\mathcal{G}_e$ are adjacent if C_j is a function of X_i. Likewise, (j,i) are adjacent in $\mathcal{G}_d$ if $\widehat{X}_i$ is a function of C_j. We assume that the ratio of the average to the worst case degrees of the right vertices of $\mathcal{G}_e$ is bounded from below by a constant independent of n, and similarly for the left vertices of $\mathcal{G}_d$.

Remark: Under (A1) and (A2), any adaptive scheme (whether for update or decoding) achieving worst-case locality l can be converted to a non-adaptive scheme with locality $\leqslant 2^l$. This is because an adaptive scheme with locality l may depend on up to 2^l different bits. Therefore a lower bound on the locality of a non-adaptive scheme translates into a lower bound on the logarithm of the locality of an adaptive scheme.

Theorem II.2. *For any adaptive scheme that satisfy (A1)–(A3) with $R < 1$, we have*

$$d_{\mathrm{wc}} + u_{\mathrm{wc}} = \Omega(\log\log n).$$

As mentioned earlier, it is possible to achieve $(d_{\mathrm{wc}}, u_{\mathrm{wc}}) = (O(1), O(1))$ without assumptions (A1)–(A3). We conjecture that Theorem II.2 holds even without assumption (A3), but were unsuccessful in proving this. We also conjecture that it holds even without (A2). As we will see later, the data structure that leads to Lemma II.1 does not satisfy (A2).

III. Achievability

The high-level structure of our scheme is inspired by the locally decodable compressor in [4]. The idea in [4] is to partition the set of message symbols into constant-sized blocks and use a fixed-length compressor for each block. The residual error vector is then encoded using the succinct data structure in [14] that occupies negligible space but allows local decoding of a single bit using $O(1)$ bitprobes. However, this data structure is static, in the sense that it does not allow efficient updates and hence does not get us small u_{wc}.

A well-known dynamic data structure in the word-RAM model is the van Emde Boas tree [15] which takes space $O(b)$ but allows local retrieval, insert and delete in $O(\log\log b)$ time (equivalently $O(\log b\,\log\log b)$ bitprobes). If we use the van Emde Boas tree for encoding the residual error vector, then we can achieve rate close to $H(p_X)$ but a higher locality of $O(\log\log n\ \mathrm{poly}(\log\log\log n))$.

Our main contribution is a novel dynamic succinct data structure (used to encode the residual error) which occupies roughly the same space as [14] but allows local decode *and* update using only $O(\log b)$ bitprobes.

A. Proof of Theorem II.1

We partition the n-length message sequence x^n into blocks $x^{b_1}(1), \ldots, x^{b_1}(n/b_1)$ of size $b_1 = O(\log n)$ each. Each block i is further partitioned into subblocks $(x^{b_0}(i,1), \ldots, x^{b_0}(i, b_1/b_0))$ of b_0 symbols each. Each subblock is compressed independently using a fixed-length lossy compression scheme of rate $H(p)+\epsilon$ and average per-letter distortion ϵ. Let $c^{b_0(H(p)+\epsilon)}(i,j)$ denote this subcodeword for the (i,j)th subblock. In addition, the error vectors (denoted $e^{b_0}(i,j)$ and equal to $x^{b_0}(i,j)$ if $x^{b_0}(i,j)$ is atypical and 0^{b_0} otherwise) are concatenated and for each i, $e^{b_1}(i) \overset{\mathrm{def}}{=} (e^{b_0}(i,1), \ldots, e^{b_0}(i, b_1/b_0))$ is compressed using the scheme in Lemma II.1 to give $\overline{c}^{\epsilon b_1}(i)$ with $\delta = \epsilon$. The overall codeword is the concatenation of $(c^{b_0(H(p)+\epsilon)}(i,j) : 1 \leqslant i \leqslant n/b_1, 1 \leqslant j \leqslant b_1/b_0)$ and $(\overline{c}^{\epsilon b_1}(i) : 1 \leqslant i \leqslant n/b_1)$.

As long as the sparsity of $e^{b_1}(i)$ is less than $\alpha b_1/\log b_1$ for a suitably chosen $\alpha > 0$, we can recover the ith message block (or any symbol within it) without error.

We choose $b_1 = \alpha_1(\log n \log\log n)$ and $b_0 = \alpha_2(\log\log n)$.

Using Azuma's inequality (and carefully choosing α_1, α_2), the probability that the distortion in each block is greater than $\alpha \log n/\log\log n$ falls as $o(1/n)$.

The worst-case local decoding is at most $O(\log b_1) = O(\log\log n)$, while the worst-case update efficiency is $O(\frac{1}{\epsilon}\log\log n)$. The compression rate is $H(p) + 2\epsilon$, and the overall probability of error (using the union bound over blocks) is $o(1)$. This completes the proof. □

All that remains is to prove Lemma II.1.

B. A succinct data structure achieving $O(\log b)$ locality for sparse sequences of length b

The high-level idea in our data structure is to split the b symbols into chunks of $O(\log b)$ symbols each, and maintain a dynamic memory table where we only store the chunks with

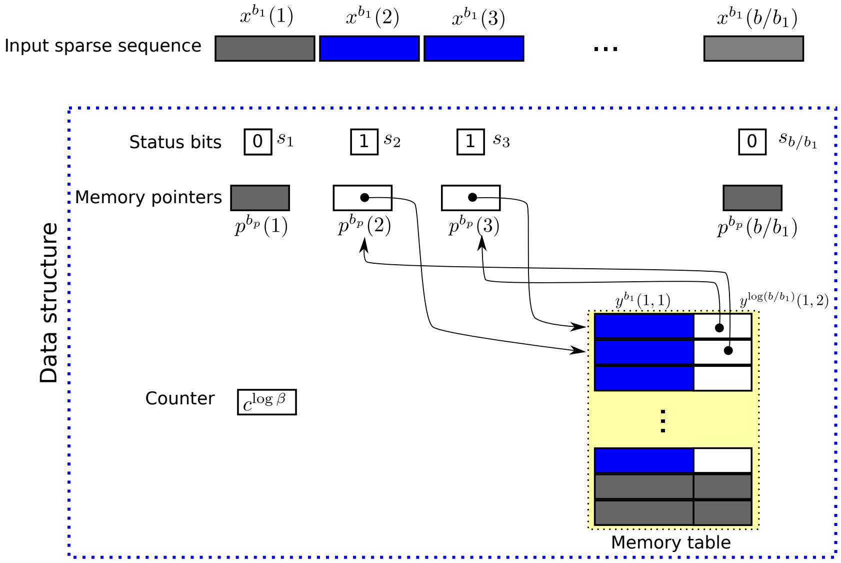

Fig. 2: Illustrating the data structure in Sec. III-B. For each message block, we store a status bit to denote whether the block is nonzero, and a memory pointer indicating the corresponding location in the memory table. Additionally, we store a counter with the number of nonzero blocks and a memory table. Each chunk in the memory table contains some data and a reverse pointer to the corresponding block.

nonzero Hamming weight. Addressing is resolved by storing for each chunk a pointer which indicates the location in the memory table where the chunk is encoded.

We split the b-length sequence x^b into blocks of b_1 consecutive symbols each: $x^{b_1}(1),\ldots,x^{b_1}(b/b_1)$.

The data structure has the following components:

- Status bits: b/b_1 many bits $s_1,\ldots,s_{b/b_1}$, one for each block. The status bit s_i is set to 1 if the Hamming weight of $x^{b_1}(i)$ is greater than zero, and set to zero otherwise.
- Memory table: β many chunks $y^{b_m}(1),\ldots,y^{b_m}(\beta)$ of $b_m = b_1+\log(b/b_1)$ bits each. The ith chunk $y^{b_m}(i)$ is split into two parts: $y^{b_1}(i,1)$ having b_1 bits, and $y^{\log(b/b_1)}(i,2)$ having $\log(b/b_1)$ bits. Here, $y^{b_1}(i,1)$ is a vector which stores a nonzero block $x^{b_1}(j)$ for some (suitably defined later) j, while $y^{\log(b/b_1)}(i,2)$ is a reverse pointer which encodes j in $\log(b/b_1)$ bits.
- Memory pointers: b/b_1 words $p^{b_p}(1),\ldots,p^{b_p}(b/b_1)$ of $b_p=\log\beta$ bits each, one for each block.
- Counter for number of nonzero blocks: $c^{\log\beta}$ is a vector of length $\log\beta$ bits which stores the number of nonzero blocks in x^b.

The overall codeword is a bit sequence obtained by the concatenation of the status bits, memory table, memory pointers and the counter. The total space required is

$$k_{ds} = \frac{b}{b_1} + \beta(b_1+\log(b/b_1)) + \frac{b}{b_1}\log\beta + \log\beta \qquad (1)$$

The data structure is illustrated in Fig. 2.

1) Initial encoding: Let k denote the number of nonzero blocks in x^b.

- Status bits: If $x^{b_1}(i)$ has nonzero Hamming weight, then $s_i=1$. Otherwise, it is set to zero.
- Memory pointers: If there are $j-1$ nonzero blocks among $x^{b_1}(1),\ldots,x^{b_1}(i-1)$ and $x^{b_p}(i)$ is nonzero, then $p^{b_p}(i)=j$ (or more precisely, the binary representation of j).
- Counter for number of nonzero blocks: $c^{\log\beta}$ is set to the number of nonzero blocks.
- Memory table: For every i, if $s_i=1$ and $x^{b_p}(i)=j$, then the jth chunk contains information about $x^{b_1}(i)$. Specifically, $y^{b_1}(j,1) = x^{b_1}(i)$, and $y^{\log(b/b_1)}(i,2)$ is equal to the binary representation of i.

2) Local decoding: Suppose that we want to recover x_i which happens to be the i_1th bit in the i_2th block (*i.e.,* $i=(i_2-1)b_1+i_1$).

- If $s_{i_2}=0$, then output 0. This is because $s_{i_2}=0$ implies that the entire block is zero.
- If not, then read $p^{b_p}(i_2)$. If $p^{b_p}(i_2)=j$, then output the i_1th bit in $y^{b_1}(j,1)$.

The maximum number of bits probed is

$$d_{\mathrm{wc}} = 1+b_p+1 = 2+\log\beta.$$

3) Local update: Suppose that we want to update x_i (which happens to be the i_1th bit in the i_2th block) with $\widetilde{x}_i$. The update algorithm works as follows:

- Suppose that $x_i=0$ and $\widetilde{x}_i=1$. The updater first reads s_{i_2}.
  - If $s_{i_2}=1$, then it reads $p^{b_p}(i_2)$. Suppose that $p^{b_p}(i_2)=j$. Then it writes $\widetilde{x}_i$ into the i_1th location of $y^{b_1}(j,1)$.
  - If $s_{i_2}=0$, then it means that the block was originally a zero block. The updater sets s_{i_2} to 1, and increments the counter for the number of nonzero blocks $c^{\log\beta}$ by 1. Suppose that after incrementing, $c^{\log\beta}=j$.

Then the updater sets $p^{b_p}(i_2) = j$, writes $\widetilde{x}^{b_1}(i_2)$ into $y^{b_1}(j,1)$, and sets $y^{\log(b/b_1)}(j,1)$ to i_2.

- Suppose that $x_i = 1$ and $\widetilde{x}_i = 0$. The updater first reads s_{i_2}. Clearly, this should be equal to 1. The updater reads $p^{b_p}(i_2)$ (suppose that it is equal to j), and then $y^{b_1}(j,1)$ to compute $x^{b_1}(i_2)$.
  - If $x^{b_1}(i_2)$ has Hamming weight greater than 1, then it flips the i_1th bit of $y^{b_1}(j,1)$.
  - If not, then it implies that $\widetilde{x}^{b_1}(i_2) = 0^{b_1}$. The updater next sets s_{i_2} to 0. It then decrements $c^{\log\beta}$. It next overwrites $y^{b_m}(j)$ with the contents of $y^{b_m}(c^{\log\beta})$, and sets $p^{b_p}(y^{\log(b/b_1)}(c^{\log\beta},2))$ to j. This is to consistently ensure that the first $c^{\log\beta}$ chunks of the memory table always contains all the information about nonzero blocks.

  The maximum number of bits that need to be read and written in order to update a single message bit is

$$u_{\rm wc} = 2 + b_p + 2\log\beta + 2b_m + b_p = 2 + b_1 + 4\log\beta + \log\frac{b}{b_1}.$$

4) Proof of Lemma II.1: Let us now prove the statement. We use the above scheme with $b_1 = O\left(\frac{1}{\delta}\log b\right)$. From (1), the total space used is

$$k_{ds} \leqslant \delta b(1 + o(1)).$$

The worst-case local decoding is equal to $O(\log b)$ and the worst-case update efficiency is equal to $O\left(\frac{1}{\delta}\log b\right)$. This completes the proof. □

Remark III.1. *The succinct data structure here satisfies (A1) but not (A2). Clearly, the order of the chunks in the memory table depends on the sequence of updates performed previously. For example, the first chunk could initially contain data of the first subblock. After a number of updates (e.g., involving setting all bits of the first block to zero, inserting bits in other blocks, and then repopulating the first block with ones), the first block could be stored in chunk $l > 1$.*

IV. Lower bounds on simultaneous locality

To obtain lower bounds, we introduce an additional parameter that we might be interested in minimizing: the worst-case *local encodability*, $e_{\rm wc}$, defined to be the maximum number of input symbols that any single codeword bit can depend on. Note that this is different from the update efficiency. This was studied in [16], where the authors related this quantity to a problem of semisupervised learning. It has been established in the literature that separately, each of $d_{\rm wc}, u_{\rm wc}, e_{\rm wc}$ can be made $O(1)$ for near-entropy compression. However, it is not known if

$$(d_{\rm wc}, u_{\rm wc}, e_{\rm wc}) = (\Theta(1), \Theta(1), \Theta(1))$$

can be simultaneously achieved.

In this section, we assume (A1)–(A3) and that the scheme is nonadaptive.

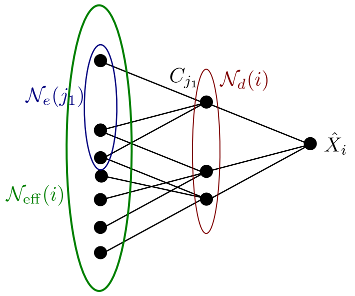

Fig. 3: Illustrating various neighbourhoods used in the proofs.

1) The local encoding and decoding graphs: Our derivation of the lower bound involves analyzing the connectivity properties of two bipartite graphs that describe the local encoding and decoding functions. As we will demonstrate, the various probabilities of error are influenced by the degrees of these bipartite graphs, whose values are governed by $d_{\rm wc}$, $e_{\rm wc}$ and $u_{\rm wc}$.

- Under (A1) and (A2), the jth compressed bit C_j can be written as $C_j = f_j(X_{\mathcal{N}_e(j)})$ for some function f_j, where $\mathcal{N}_e(j)$ is the set of message locations that C_j can depend on.
- The ith decoded bit $\hat{X}_i$ can be written as $\hat{X}_i = g_i(C_{\mathcal{N}_d(i)})$ for some function g_i, where $\mathcal{N}_d(i)$ is the set of codeword locations that need to be probed in order to recover X_i.
- We can construct an $n \times nR$ encoder bipartite graph $\mathcal{G}_e$ where (i,j) is an edge only if $i \in \mathcal{N}_e(j)$, and an $nR \times n$ decoder bipartite graph $\mathcal{G}_d$ where $(j,i) \in [nR] \times n$ is an edge if $j \in \mathcal{N}_d(i)$. See Fig. 1 for an illustration.
- Let $e_j \overset{\rm def}{=} |\mathcal{N}_e(j)|$ be the degree of the jth right vertex in $\mathcal{G}_e$. This is equal to the local encodability of the ith codeword symbol and satisfies $e_i \leqslant d_{\rm wc}$.
- This gives a natural lower bound on the update efficiency for the ith message symbol: it must be greater than or equal to Δ_i^l, the degree of the ith left vertex in $\mathcal{G}_e$.
  The average degree of a vertex in the left (corresponding to a message symbol) is equal to R times the average degree of a right vertex (which corresponds to codeword bits). This implies that $u_{\rm wc}$ is lower bounded by the average (arithmetic mean of) the individual local encodabilities of the individual codeword bits.
- Likewise, we denote the degree of the jth left vertex of $\mathcal{G}_d$ by Δ_j^r and that of the ith right vertex in $\mathcal{G}_d$ by d_i. Note that d_i is equal to the local decodability of $\widehat{X}_i$ (maximum number of codeword bits to be probed to recover $\widehat{X}_i$). By definition, $d_i \leqslant d_{\rm wc}$ for all i.
- We also define the so-called effective neighbourhood of $\widehat{X}_i$ to be

$$\mathcal{N}_{\rm eff}(i) \overset{\rm def}{=} \{X_l : l \in \bigcup_{j \in \mathcal{N}_d(i)} \mathcal{N}_e(j)\}.$$

The ith decoded symbol is a function of only those symbols in $\mathcal{N}_{\rm eff}(i)$, i.e., there exists a function h_i such that $\hat{X}_i = h_i(X_{\mathcal{N}_{\rm eff}(i)})$. This is illustrated in Fig. 3.

Let us now obtain a lower bound on the bit error probability of any compression scheme satisfying the above properties. We would like to point out that the following Lemmas IV.1 and IV.2 only make use of assumptions (A1) and (A2), and do not require (A3).

Lemma IV.1. *The probability of bit error, $P_e^{(i)} \overset{def}{=} \Pr[\hat{X}_i \neq X_i]$ satisfies*

$$P_e^{(i)} = \begin{cases} \geqslant (1-p)^{|\mathcal{N}_{\text{eff}}(i)|} \geqslant (1-p)^{e_{\text{wc}}d_{\text{wc}}}, & \textit{or,} \\ 0. \end{cases}$$

Proof. For every $i \in [n]$, the decoded symbol $\hat{X}_i$ is a deterministic function (the composition of g_i and f_j's) of $\mathcal{N}_{\text{eff}}(i) \overset{\text{def}}{=} \{X_l : l \in \bigcup_{j \in \mathcal{N}_d(i)} \mathcal{N}_e(j)\}$. But we have $|\mathcal{N}_{\text{eff}}(i)| \leqslant e_{\text{wc}}d_{\text{wc}}$. The probability of error is given by[2]

$$\begin{aligned} P_e^i &= \sum_x \left(\prod_{l \in \mathcal{N}_{\text{eff}}(i)} \Pr[X_l = x_l] \right) 1_{\{X_i = \hat{X}_i\}} \\ &\geqslant \sum_x \left(\prod_{l \in \mathcal{N}_{\text{eff}}(i)} (1-p) \right) 1_{\{X_i = \hat{X}_i\}}. \end{aligned}$$

If there is even a single configuration of X^n for which $X_i \neq \hat{X}_i$, then $P_e^i \geqslant (1-p)^{|\mathcal{N}_{\text{eff}}(i)|} \geqslant (1-p)^{e_{\text{wc}}d_{\text{wc}}}$. □

Remark: Observe that Lemma IV.1 holds even without assumption (A3). Also, if $R < 1$, then by the pigeonhole principle $P_e^{(i)} > 0$ for a nonvanishing fraction of i's. Since $P_{\text{glob}} \geqslant \max_i P_e^{(i)}$, we can conclude that any compression scheme satisfying (A1)–(A2) and having vanishing probability of error and nontrivial rate must satisfy $u_{\text{wc}}u_{\text{wc}} = \omega(1)$. In other words, one cannot achieve $(d_{\text{wc}}, u_{\text{wc}}) = (O(1), O(1))$.

A similar observation was made in [3] for linear locally decodable (but not updatable) compression schemes. If the encoder of the compressor must be linear, then $d_{\text{wc}} = \omega(1)$ if we desire a vanishingly small probability of error.

We can use the above lemma to prove a lower bound on simultaneous local encodability and decoding.

Lemma IV.2. *Any fixed-length compression scheme achieving $R < 1$, a vanishing probability of error, and satisfying assumptions (A1) and (A2) must necessarily satisfy*

$$e_{\text{wc}}d_{\text{wc}} = \Omega(\log n)$$

Proof. If we operate at a rate $R < 1$, then by the pigeonhole principle, it must be the case that at least $n(1-R)$ bits have a nonzero bit error probability. Call this set of bits/locations $\mathcal{E}$.

We will now select a subset $\mathcal{S}$ of $O(n/e_{\text{wc}}d_{\text{wc}})$ many X_i's whose probabilities of error are all nonzero but whose error events are statistically independent of each other. We can construct such an $\mathcal{S}$ in a greedy manner.

For every $i \in \mathcal{E}$, the corresponding $\hat{X}_i$ is a deterministic function of $\{X_l : l \in \mathcal{N}_{\text{eff}}(i)\}$. In other words, each $\hat{X}_i$ depends on at most $e_{\text{wc}}d_{\text{wc}}$ many X_l's. We can start with the empty set, and iteratively put those X_i's in $\mathcal{S}$ which are in $\mathcal{E}$ but not in $\bigcup_{i_1 \in \mathcal{S}} \mathcal{N}_{\text{eff}}(i_1)$. It is clear that

$$|\mathcal{S}| \geqslant n(1-R)/(d_{\text{wc}}e_{\text{wc}}).$$

Moreover, by construction, we have that all the events $1_{X_i \neq \hat{X}_i}$ are independent for all $i \in \mathcal{S}$ (since the corresponding sets of X_l's they depend on are disjoint).

Let $m \overset{\text{def}}{=} n(1-R)/(e_{\text{wc}}d_{\text{wc}})$. Using Lemma IV.1, the probability of block error is bounded as

$$\begin{aligned} P_e &\geqslant 1 - (1-(1-p)^{d_{\text{wc}}e_{\text{wc}}})^m \geqslant 1 - e^{-m(1-p)^{d_{\text{wc}}e_{\text{wc}}}} \\ &\geqslant m(1-p)^{d_{\text{wc}}e_{\text{wc}}}. \end{aligned}$$

The above quantity is vanishing in n only if $d_{\text{wc}}e_{\text{wc}} = \Omega(\log n)$ □

A. Lower bound on $d_{\text{wc}}u_{\text{wc}}$ and proof of Theorem II.2

We can in fact tighten the analysis above to prove an identical lower bound on $d_{\text{wc}}u_{\text{wc}}$.

Let $\eta_i := \mathcal{N}_{\text{eff}}(i)$. Let u_i, e_j, d_i respectively denote the update efficiency for the ith message symbol, the local encodability of the jth codeword symbol, and the local decoding of the ith message symbol. These are also respectively greater than or equal to the degrees of the ith left vertex in $\mathcal{G}_e$, the jth right vertex in $\mathcal{G}_e$, and the ith right vertex in $\mathcal{G}_d$.

Lemma IV.3. *Consider any fixed-length compression scheme achieving vanishing probability of error and nontrivial compression rate $R < 1$. There exists a set $\mathcal{S} \subset [n]$ of message symbols with $\mathcal{S} = \Theta(n)$ such that for all $i \in \mathcal{S}$,*

$$\eta_i = \Omega(\log n).$$

The proof is very similar to that of Lemma IV.2, so we only sketch the details. We select a subset $\mathcal{S}'$ of $[n]$ such that $P_e^i > 0$ for all $i \in \mathcal{S}'$. We know that $|\mathcal{S}'| \geqslant n(1-R)$. A more careful rederivation of Lemma IV.1 gives us $P_e^i \geqslant (1-p)^{\eta_i}$ for all $i \in \mathcal{S}'$. Hence,

$$P_e \geqslant 1 - \prod_{i=1}^{n} (1 - P_e^i) \tag{2}$$

$$\geqslant \sum_{i \in \mathcal{S}'} (1-p)^{\eta_i}, \tag{3}$$

which is nonvanishing in n if $\eta_i < \log n$ for any subset of $\mathcal{S}'$ of size $n(1-R)/2$. Therefore, there must exist a subset $\mathcal{S} \subset \mathcal{S}'$ of size $n(1-R)/2$ where $\eta_i > \log n$ for all $i \in \mathcal{S}$. This completes the proof. □

This leads us to the following result:

Theorem IV.1. *Any fixed-length compression scheme achieving vanishing probability of error and $R < 1$, and satisfying assumptions (A1)—(A3) and nonadaptive local algorithms must have*

$$d_{\text{wc}}u_{\text{wc}} = \Omega(\log n).$$

[2] For an event $\mathcal{E}$, $1_{\mathcal{E}}$ is the indicator function which takes value 1 if $\mathcal{E}$ occurs, and zero otherwise.

Proof. From Lemma IV.3, there exists a set $\mathcal{S}$ of size $\Theta(n)$ such that for all $i \in \mathcal{S}$, $\eta_i = \Omega(\log n)$. However, $\eta_i \leqslant d_{\mathrm{wc}} \sum_{j \in \mathcal{N}_d(i)} e_j$. From the graph $\mathcal{G}_e$, we also have

$$d_{\mathrm{wc}} \sum_{j \in [nR]} e_j = d_{\mathrm{wc}} \sum_{i \in [n]} \Delta_i^l \leqslant d_{\mathrm{wc}} \sum_{i \in [n]} u_i \leqslant n d_{\mathrm{wc}} u_{\mathrm{wc}}, \quad (4)$$

where Δ_i^l denotes the degree of the ith left vertex in $\mathcal{G}_e$.

Using Lemma IV.3, we have

$$|\mathcal{S}| \Omega(\log n) \leqslant \sum_{i \in \mathcal{S}} \eta_i \leqslant \sum_{i=1}^{n} \eta_i \leqslant d_{\mathrm{wc}} \sum_{j \in [nR]} e_j .$$

From (4) and using the fact that $|\mathcal{S}| = \Theta(n)$, we have $n d_{\mathrm{wc}} u_{\mathrm{wc}} = \Omega(n \log n)$, which implies $d_{\mathrm{wc}} u_{\mathrm{wc}} = \Omega(\log n)$, completing the proof. □

Using Theorem IV.1 and our remark on adaptive vs non-adaptive schemes just prior to Theorem II.2, we get Theorem II.2.

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] A. Makhdoumi, S.-L. Huang, M. Medard, and Y. Polyanskiy, “On locally decodable source coding,” *arXiv preprint arXiv:1308.5239*, 2013.

[3] 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)*, pp. 4394–4399, IEEE, 2015.

[4] 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)*, pp. 2984–2988, IEEE, 2015.

[5] 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.

[6] 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*, pp. 891–895, 2018.

[7] 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*, pp. 2474–2478, IEEE, 2008.

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

[9] M. Patrascu and M. Thorup, “Dynamic integer sets with optimal rank, select, and predecessor search,” in *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 166–175, IEEE, 2014.

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

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

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

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

[14] 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.

[15] P. van Emde Boas, “Preserving order in a forest in less than logarithmic time,” in *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pp. 75–84, IEEE, 1975.

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