On the relaxed LDC of BGHSV: A survey that corrects the record

Oded Goldreich

April 28, 2024

Abstract

A locally decodable code (LDC) is an error correcting code that allows for recovery of any desired bit in the message based on a constant number of randomly selected bits in the possibly corrupted codeword. A relaxed LDC requires correct recovery only in case of actual codewords, while requiring that for strings that are (only) close to the code, with constant probability, the local decoder outputs either the correct value or a special failure symbol (but not a wrong value).

We survey the relaxed LDC presented by Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan (in 2004). For every constant $\alpha > 0$, their code has block-length $n = k^{1+\alpha}$, whereas their decoder makes $O(1/\alpha)$ queries. Unfortunately, their paper claims a $O(1/\alpha^2)$-query decoder, but the inferior complexity bound is merely due to a trivial oversight. This survey was triggered by a wish to correct the historical record.

Contents

1 Introduction

A locally decodable code (LDC) is a (binary) error correcting code that allows for the recovery of any desired bit in the message based on a constant number of (randomly selected) bits in the possibly corrupted codeword.

Locally decodable codes, or rather family of such codes, have several parameters: The length of the message, denoted k, the length of codewords, denoted n (and viewed as a function of k), the number of queries, denoted q, and the tolerated corruption rate, denoted δ. We shall view $q \in \mathbb{N}$ and $\delta > 0$ as fixed constants, whereas k and $n = n(k)$ are viewed as varying parameters. (This regime is fundamentally different from the one in [11] (and subsequent works), where $n = O(k)$ and q is allowed to be a function of k.)

The conjecture that locally decodable codes require large length (e.g., n must be super-polynomial in k)[1], which was supported by the super-linear lower bound (i.e., $n = \Omega(k^{q/(q-1)})$) of [9], led [2] to suggest a relaxed notion of LDCs. In this relaxation, hereafter referred to as *relaxed LDCs*, the decoder is allowed to announce failure and is required to satisfy the following two conditions (see Definition 1):

1. When given access to a valid codeword, the local decoder always recovers the desired bit.[2]
2. When given access to a string that is δ-close to a valid codeword (i.e., the relative Hamming distance between the string and the codeword is at most δ), with probability at least $2/3$, the local decoder does not err; that is, with probability at least $2/3$, it outputs either the desired bit or a special failure symbol.

As shown in [2, Sec. 4.2.1], such a relaxed LDC can be transformed into one in which a $1-O(\Delta(w))$ fraction of the bits of the message can be correctly recovered (with probability at least $2/3$) when the local decoder is given access to a string w that is $\Delta(w)$-close to the code. Furthermore, if the local decoder satisfies the *average smoothness condition* (i.e., when the desired bit position is uniformly distributed in $[k]$, each of the queries of the local decoder is almost uniformly distributed in $[n]$), then the original decoder correctly recovers a $1-O(\Delta(w))$ fraction of the bits of the message with probability at least $5/9$.

More importantly, as shown in [2, Sec. 4.2.2], relaxed LDCs of polynomial length exist. Specifically, for every sufficiently large constant q, one can obtain a q-query relaxed LDC of length $n = k^{1+O(1/q)}$. Unfortunately, as stated in the abstract, [2, Thm. 1.5] states a weaker result (i.e., $n = k^{1+O(1/\sqrt{q})}$), but this is due to a trivial calculation oversight. In this note, we survey the relevant construction and its analysis.

Organization The core of this note is Section 3, which surveys the relaxed LDC of [2]. This section is preceded by Section 2, which provides the actual definitions, and is followed by Section 4, which provides an alternative presentation. We end (in Section 5) with some reflections.

[1] It was even conjectured that $n > \exp(k^{\Omega(1)})$, but this conjecture was refuted in [13, 4].

[2] This is the "one-sided error" version. In the "two sided error" version, the decoder is required to recover the desired bit with probability at least $2/3$.

2 Formal Setting

For the sake of good order, we recall the standard definition of a *relaxed* locally decodable code (relaxed LDC), while viewing $\delta > 0$ and $q \in \mathbb{N}$ as fixed constants, whereas k and $n = n(k)$ are viewed as varying parameters. Still, at times we shall use notations such as $O(q)$, which assert a universal dependence on q.

Definition 1 (relaxed LDC): *We say that $C : \{0,1\}^k \to \{0,1\}^n$ is a q-*query relaxed locally decodable code (relaxed LDC) *if for some constant $\delta > 0$ there exists a randomized oracle machine D, called a* local decoder*, that makes q queries such that the following two conditions hold.*

1. *For every $x = (x_1, ..., x_k) \in \{0,1\}^k$ and $i \in [k]$,*
$$\Pr[D^{C(x)}(i) = x_i] = 1.$$
(A two-sided error version only requires that $\Pr[D^{C(x)}(i) = x_i] \geq 2/3$.)
2. *For every $x \in \{0,1\}^k$, every $w \in \{0,1\}^n$ that is δ-close to $C(x)$, and every $i \in [k]$,*
$$\Pr[D^w(i) \in \{x_i, \perp\}] \geq 2/3$$
*where $\perp \notin \{0,1\}$ is a special symbol and w is δ-*close *to y if $\Delta(w,y) \stackrel{\text{def}}{=} \frac{|\{j \in [n] : w_j \neq y_j\}|}{n} \leq \delta$.*

The fixed parameter δ is called the decoding distance of *D.*

Indeed, δ is typically smaller than half the relative distance of C, and q is the query complexity of the local decoder. The (original) *non-relaxed* notion of a locally decodable code is obtained by requiring that D never outputs $\perp$.

Stronger definitions are presented in [2, Sec. 4.2.1]. In particular, in [2, Def. 4.5], it is required that *for a $1 - O(\Delta_C(w))$ fraction of the i's it holds that* $\Pr[D^w(i) = x_i] \geq 2/3$, where $\Delta_C(w) \stackrel{\text{def}}{=} \min_x\{\Delta(w, C(x))\}$. Note that this additional condition implies a two-sided error version of Condition 1. On the other hand, the additional condition (essentially) follows from Condition 1 whenever the decoder satisfies the average smoothness condition, which requires that, *for a uniformly distributed $i \in [k]$, each of the queries of the decoder is almost uniformly distributed in* $[n]$.[3]

The relaxed locally decodable code of Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [2]. As stated upfront, we shall survey the proof of the following result of [2].

Theorem 2 (the relaxed LDC of [2, Sec. 4.2.2]): *For every constant $\ell \in \mathbb{N}$, there exists an $O(\ell)$-query relaxed locally decodable code $C : \{0,1\}^k \to \{0,1\}^n$ for decoding distance $\Omega(1/\ell)$ such that $n = O(k^{1+(1/\ell)+o(1)})$. Furthermore, the decoder satisfies the average smoothness condition.*

Recall that [2, Thm. 1.5] asserts that the decoder makes $O(\ell^2)$ queries, although it is stated in [2, Sec. 4.2.2] that the query complexity of [2, Construction 4.14] is $O(\ell/\delta_{\mathtt{pcpp}})$ and that $\delta_{\mathtt{pcppp}} = \Omega(1)$. (The oversight in [2] is that the text that follows [2, Construction 4.14] uses $\delta_{\mathtt{pcppp}} = \Omega(1/\ell)$ rather than $\delta_{\mathtt{pcppp}} = \Omega(1)$.)

[3] The O-notation hides a factor that is proportional to the query complexity, whereas "almost uniformly (in $[n]$)" means a distribution in which each $j \in [n]$ occurs with probability $\Theta(1/n)$, and "essentially follows" means that for such i's it holds that $\Pr[D^w(i) = x_i] \geq 5/9$. The point is that, on input a uniformly distributed $i \in [k]$, the view of the q-query decoder when querying w is $O(q \cdot \Delta_C(w))$-close to its view when querying $C(x)$. Hence, the views are 0.1-close for a $1 - O(\Delta_C(w))$ fraction of the i's.

3 The Relaxed Locally Decodable Code of [2, Sec. 4.2.2]

This section provides a revised presentation of [2, Sec. 4.2.2]. Our presentation differs from the original only in low-level choices (e.g., notations, constants, and calculations) and in dealing directly with the general case (i.e., any $\ell \in \mathbb{N}$).[4]

3.1 The code and its (relaxed) local decoder

We start by presenting the code constructed in [2, Sec. 4.2.2]. This code combines an asymptotically good code, denoted $C_0 : \{0,1\}^k \to \{0,1\}^n$, for which encoding can be performed by (almost) linear-size circuits, with a *PCP of Proximity* for CVAL (Circuit Value) that has almost linear length. That is, viewing k and $n = n(k)$ as varying parameters, we assume the following.

- The code C_0 has constant relative distance, denoted δ_0, it holds that $n = O(k)$, and there is an $n^{1+o(1)}$-size circuit for evaluating C_0 (i.e., computing the encoding function).

  (Recall that expander codes can be evaluated by uniform linear-size circuits [12].)

- A PCP of Proximity (PCPP) for CVAL that utilize proof-oracles of size that is almost linear in the circuit size.

  - Recall that a PCP of Proximity for a set S [2] is given query access to an input-oracle and to a proof-oracle. For a constant proximity parameter $\epsilon > 0$, the verifier makes a constant number of queries (to both oracles) and is required to always accept any input in S (when given an adequate proof-oracle), and reject with probability at least $2/3$ any input that is ϵ-far from S (regardless of the choice of the proof-oracle).
  - The set CVAL refers to a fixed uniform family of *non-deterministic* circuits, and the input is a string that is supposed to satisfy the relevant circuit.

  As shown in [2], CVAL has a PCP of Proximity that makes $O(1/\epsilon)$ queries and uses a proof-oracle of length $s^{1+o(1)}$, where s is the size of the circuit. Furthermore, each query made to the input-oracle (resp., proof-oracle) is almost uniformly distributed in it. We assume, without loss of generality, that the verifier makes the same number of queries to both oracles.

Using these ingredients, the basic idea (which handles the case of $\ell = 1$), is to use a code that consists of three equal-length blocks. On input $x \in \{0,1\}^k$, the first block consists of an adequate number of repetitions of the codeword $C_0(x)$, the second block consists of an adequate number of repetitions of x itself, whereas the third block consists of k proof-oracles such that the i^{th} proof-oracle refers to the claim that the value provided for x_i is consistent with the codeword $C_0(x)$. (The repetitions are required because the length of each proof-oracle is $n^{1+o(1)}$, whereas $|C_0(x)| = n$ and $|x| < n$.) On input $i \in [k]$ and oracle access to a purported $\overline{w}$, the local decoder checks that the first block of $\overline{w}$ actually consists of repetitions of some n-bit long string, denoted w, and verifies that the purported i^{th} bit of x (recovered from an adequate random position in the second block) is consisted with w. The verification is performed by invoking the PCP of Proximity while using the i^{th} sub-block of the third block as a proof-oracle (where the input-oracle is essentially a random sub-block of the first block).

[4] Indeed, we skip Proposition 4.13 of [2, Sec. 4.2.2], and proceed directly to Construction 4.14 and Proposition 4.15.

The general case (of arbitrary $\ell \in \mathbb{N}$) is handled by extending the foregoing strategy. Specifically, in case of $\ell = 2$, in addition to using copies of $C_0(x)$ and x, we use copies of $C_0(x_{i_1})$ for every $i_1 \in [k^{1/2}]$, where $x \equiv (x_1, ..., x_{k^{1/2}}) \in (\{0,1\}^{k^{1/2}})^{k^{1/2}}$. Instead of directly verifying the consistence of bits in x with $C_0(x)$, we verify the consistency of such bits with the corresponding $C_0(x_{i_1})$ and the consistency of the latter with $C_0(x)$. Hench, rather than having k proof-oracles that refer to n-bit long input-oracles, we have $m \stackrel{\text{def}}{=} k^{1/2}$ proof-oracles that refer to $2n$-bit long input-oracles and k proof-oracles that refer to $(2n/m)$-bit long input-oracles. In general, for $\ell \geq 2$, we use $\ell - 1$ intermediate blocks, and verify consistency between adjacent blocks. This yields [2, Construction 4.14], which is presented next.

Construction 3 (the code): *For a constant parameter $\ell \in \mathbb{N}$, we view the message $x \in \{0,1\}^k$ as an ℓ-dimensional tensor $(x_{i_1,...,i_\ell})_{i_1,...,i_\ell \in [m]}$ of bits, where $k = m^\ell$. For each $j \in [\ell - 1]$ and $(i_1, ..., i_j) \in [m]^j$, we view the $(\ell - j)$-dimensional tensor $(x_{i_1,....,i_j,i_{j+1},...,i_\ell})_{i_{j+1},...,i_\ell \in [m]}$ as an $m^{\ell-j}$-bit long string, and denote it by $x_{i_1,....,i_j}$.*[5] *For $t \in \mathbb{N}$ to be determined below, we present the code*

$$C(x) \stackrel{\text{def}}{=} (C_0(x)^t, (C_0(x_{i_1})^t)_{i_1 \in [m]}, ..., (C_0(x_{i_1,...,i_\ell})^t)_{i_1,...,i_\ell \in [m]}, \overline{\pi}_1(x), ..., \overline{\pi}_\ell(x))$$

where $t = |\overline{\pi}_j(x)|/n$ and $\overline{\pi}_j(x) = (\pi^{t_j}_{i_1,...,i_j})_{i_1,...,i_j \in [m]}$ such that $\pi_{i_1,...,i_j}$ is a proof-oracle (of the PCP of Proximity) *for the claim that $(C_0(x_{i_1,....,i_{j-1}}), C_0(x_{i_1,....,i_j})^m)$ is in the set*

$$S_{j,i_j} \stackrel{\text{def}}{=} \{(y, z) : \exists x'_1, ..., x'_m \in \{0,1\}^{m^j} \text{ s.t. } C_0(x'_1 \cdots x'_m) = y \ \& \ C_0(x'_{i_j})^m = z\}.$$

Hence, a codeword consists of $(2\ell + 1)$ equal length blocks, where each block has length $t \cdot n$. For every $j \in [\ell + 1]$, the j^{th} block contains t copies of $C_0(x_{i_1,...,i_{j-1}})$ for each $(i_1, ..., i_{j-1}) \in [m]^{j-1}$. Likewise, for every $j \in [\ell]$ and $(i_1, ..., i_j) \in [m]^j$, the $(\ell+1+j)^{\text{th}}$ block contains t_j copies of $\pi_{i_1,...,i_j}$, where $t_\ell > \cdots > t_2 > t_1 = 1$.

Note that membership in S_{j,i_j} can be decided by non-deterministic circuits of linear size. Alternatively, assuming that errorless decoding for C_0 can be computed by linear size circuits (see [12]), it follows that membership in S_{j,i_j} can be decided by (deterministic) circuits of linear size.

The length of the codewords of C is $(2\ell + 1) \cdot t \cdot n$, where t was set such that $t \cdot n = |\overline{\pi}_1(x)| = |(\pi_1, ..., \pi_m)|$, because the π_h's are the longest proof-oracles (since they refer to inputs of length $2n$). Recalling that $|\pi_h| = (2n)^{1+o(1)}$, it follows that the length of the codewords of C is $O(m) \cdot n^{1+o(1)} = n^{1+(1/\ell)+o(1)}$.

The relative distance of the code C is at least $\delta_0/(2\ell + 1)$, because different codewords of C must differ on the first block, which consists of repetitions of some codeword of C_0.

Construction 4 (the decoder): *For $\ell \in \mathbb{N}$ and $C : \{0,1\}^{m^\ell} \to \{0,1\}^{(2\ell+1)\cdot tn}$ as in Construction 3, on input $i = (i_1, ..., i_\ell) \in [m]^\ell$, when given oracle access to $(\overline{w}_0, \overline{w}_1, ..., \overline{w}_\ell, \overline{p}_1, ..., \overline{p}_\ell) \in (\{0,1\}^{tn})^{2\ell+1}$, the decoder proceeds as follows.*

1. *It tests that $\overline{w}_0$ is in $\{w^t : w \in \{0,1\}^n\}$.*

[5] The reader may envision an m-ary tree of depth ℓ with $x \equiv x_\lambda$ at its root, and $x_{i_1,...,i_j}$ being the i_j^{th} child of $x_{i_1,...,i_{j-1}}$. Below, we apply C_0 to $m^{\ell-j}$-bit long strings, for each $j \in \{0, 1, ..., \ell\}$, and rely on the fact that in each case C_0 stretches its input by a factor of n/k.

Letting $\overline{w}_0 = (w_1, ..., w_t) \in (\{0,1\}^n)^t$, this is done by repeating the following check $O(1)$ times: Select uniformly $r', r'' \in [t]$ and $s \in [n]$, and check whether the s^{th} bit of $w_{r'}$ equals the s^{th} bit of $w_{r''}$.

The test is performed so that tn-bit long strings that are $0.01 \cdot \delta_0$-far from $\{w^t : w \in \{0,1\}^n\}$ are rejected with probability at least $2/3$. Such rejection leads the decoder to halt with output $\perp$. Otherwise, the decoder selects uniformly $r \in [t]$ and $(r_j, r'_j) \in [t] \times [t_j]$ for each $j \in [\ell]$, and proceeds to the next step.

2. *For each $j \in [\ell]$, let $\overline{w}_j = (w_1^{(i'_1,...,i'_j)}, ..., w_t^{(i'_1,...,i'_j)})_{i'_1,...i'_j \in [m]} \in ((\{0,1\}^{(n/k) \cdot m^{\ell-j}})^{m^j})^t$, and let $v_j = w_{r_j}^{(i_1,...,i_j)}$; that is, $v_j \in \{0,1\}^{(n/k) \cdot m^{\ell-j}}$ is the r_j^{th} string in the sequence of t strings associated with $(i_1, ..., i_j) \in [m]^j$ in $\overline{w}_j$. Similarly, let p_j be the r'^{th}_j string in the sequence of t_j strings* (i.e., the sequence $(p_1^{(i_1,...,i_j)}, ..., p_{t_j}^{(i_1,...,i_j)})$) *associated with $(i_1, ..., i_j) \in [m]^j$* (within $\overline{p}_j = (p_1^{(i'_1,...,i'_j)}, ..., p_{t_j}^{(i'_1,...,i'_j)})_{i'_1,...i'_j \in [m]}$).

   For every $j \in [\ell]$, the decoder invokes the PCP of Proximity for S_{j,i_j}, with proximity parameter $0.1 \cdot \delta_0$, providing it with access to the input-oracle $(v_{j-1}, v_j^m))$ and the proof-oracle p_j, where $v_0 = w_r$ (i.e., the r^{th} string in $\overline{w}_0$). *If any of these invocations rejects, then the decoder halts with output $\perp$.*

3. *If $v_\ell = C_0(\sigma)$ for some $\sigma \in \{0,1\}$, then the decoder outputs σ. Otherwise, it outputs $\perp$.*

Note that the decoder makes a constant number of queries to each of the $2\ell+1$ blocks of the alleged codeword. By adding dummy queries, we can make the decoder query each block the same number of times. (Indeed, the dummy queries to each block can be chosen to be uniformly distributed in the block.)

Assuming that the queries of the PCP of Proximity are each uniformly distributed in the relevant oracle (see [2]), it follows that the queries of the decoder to each of the $2\ell + 1$ blocks are uniformly distributed in that the relevant part of that block (i.e., the part that corresponds to $(i_1, ..., i_{j-1})$ in the j^{th} and $(\ell + j)^{\text{th}}$ block). Hence, for a uniformly distributed $(i_1, ..., i_\ell) \in [m]^\ell$, the queries to each block are uniformly distributed. It follows that the decoder satisfies the average smoothness condition.

3.2 Analysis of the (relaxed) local decoder

It is easy to see that the decoder always recovers the correct bit from a valid codeword; that is, for every $x = (x_{i_1,...,i_\ell})_{i_1,...,i_\ell \in [m]}$ and every $i = (i_1, ..., i_\ell) \in [m]^\ell$, on input i and oracle access to x, the decoder always outputs x_i. Hence, we focus on proving the following.

Lemma 5 (the local decoder satisfies Condition 2 of Definition 1): *Let C be as in Construction 3. Then, for every $x = (x_{i'_1,...,i'_\ell})_{i'_1,...,i'_\ell \in [m]}$, if $\overline{w} = (\overline{w}_0, \overline{w}_1, ..., \overline{w}_\ell, \overline{p}_1, ..., \overline{p}_\ell)$ is $\delta_0/7\ell$-close to $C(x)$, then, for every $i = (i_1, ..., i_\ell) \in [m]^\ell$, on input i, with probability at least $2/3$, the decoder described in Construction 4 outputs a value in $\{x_i, \perp\}$.*

Note that there is a unique codeword of C that is $\delta_0/7\ell$-close to w, because the relative distance of C is at least $\delta_0/(2\ell + 1) > 2 \cdot \delta_0/7\ell$.

Proof Sketch: We consider three cases regarding $\overline{w}_0$ (i.e., the first block of $\overline{w}$).

Case 1: $\overline{w}_0 = (w_1, ..., w_t)$ ***is*** $0.01 \cdot \delta_0$***-far from*** $\{w^t : w \in \{0,1\}^n\}$**.** In this case,

$$\mathrm{E}_{r',r'' \in [t]}[\Delta(w_{r'}, w_{r''})] > 0.01\delta_0,$$

and Step 1 rejects (w.p. at least 2/3).

Hence, in the other cases, we may assume that, for at least a 0.9 fraction of the r's in $[t]$, it holds that $\overline{w}_0$ is $0.1 \cdot \delta_0$-close to w_r^t. In the following cases, we assume that such an r was selected in Step 1.

Case 2: w_r ***is*** $0.2 \cdot \delta_0$***-far from a codeword of*** C_0**.** In this case, for every $v \in \{0,1\}^{n/m}$, it holds that (w_r, v^m) is 0.1-far from the set S_{1,i_1}, and Step 2 rejects (with high probability), due to the PCPP invocation with $j = 1$.

Hence, we may assume that w_r is $0.2 \cdot \delta_0$-close to a codeword of C_0, denoted c, which implies that $\overline{w}_0$ is $0.3 \cdot \delta_0$-close to c^t (since $\overline{w}_0$ is $0.1 \cdot \delta_0$-close to w_r^t). Using the hypothesis that $\overline{w}$ is $\delta_0/7\ell$-close to $C(x)$, it follows that $\overline{w}_0$ is $\frac{2\ell+1}{7\ell} \cdot \delta_0$-close to $C_0(x)^t$, which implies that $c = C_0(x)$ (since $\frac{2\ell+1}{7\ell} \cdot \delta_0 \leq \frac{3}{7} \cdot \delta_0$). Thus, w_r is $0.2 \cdot \delta_0$-close to $C_0(x)$.

Case 3: w_r ***is*** $0.2 \cdot \delta_0$***-close to*** $C_0(x)$**.** In this case, we fix any sequence $(r_1, ..., r_\ell) \in [t]^\ell$, and consider the corresponding v_j's as defined in Step 2 (i.e., $v_0 = w_r$ and $v_j = w_{r_j}^{(i_1,...,i_j)}$ for every $j \in [\ell]$). If v_j is $0.2 \cdot \delta_0$-close to $C_0(x_{i_1,...,i_j})$ for every $j = 0, 1, ..., \ell$, then Step 3 outputs either $x_{i_1,...,i_\ell}$ or $\perp$. Otherwise, we consider the smallest $j \in [\ell]$ such that v_j is $0.2 \cdot \delta_0$-far from $C_0(x_{i_1,...,i_j})$, and observe that (v_{j-1}, v_j^m) is $0.1 \cdot \delta_0$-far from S_{j,i_j}, which means that Step 2 rejects (with high probability), due to the PCPP invocation with this j.

Recalling that, with probability at least 0.9, the decoder either rejects or reaches Case 3, the claim follows. ■

4 An Alternative Presentation

We discovered the calculation oversight in [2] after establishing Theorem 2 using an alternative approach (described below).[6] At that point, we realized that our approach yields a construction that is very similar to the one in [2, Sec. 4.2.2], and wondered why [2, Thm. 1.5] claims an inferior result. (The answer, of course, is that there is a calculation oversight in [2, Sec. 4.2.2].)

Our alternative presentation is iterative. Using C_0 and PCPs of Proximity as in Section 3.1, we consider a sequence of relaxed LDCs $C_1, C_2, ...$ such that C_1 is the code that corresponds to $\ell = 1$ (in Section 3.1). In general, for every $\ell \in \mathbb{N}$, using the $O(\ell)$-query relaxed LDC $C_\ell : \{0,1\}^k \to \{0,1\}^n$, where $n = k^{1+(1/\ell)+o(1)}$, we construct $C_{\ell+1} : \{0,1\}^{k^{(\ell+1)/\ell}} \to \{0,1\}^{n'}$, where $n' = (k^{(\ell+1)/\ell})^{1+(1/(\ell+1))+o(1)}$. Loosely speaking, $C_{\ell+1}(x)$ consists of three equal-length blocks: The first block consists of an adequate number of repetitions of the codeword $C_0(x)$, the second block consists of an adequate number of repetitions the codewords $C_\ell(x_1), ..., C_\ell(x_{k^{1/\ell}})$, where $x = (x_1, ..., x_{k^{1/\ell}}) \in (\{0,1\}^k)^{k^{1/\ell}}$, whereas the third block consists of $k^{1/\ell}$ proof-oracles such that

[6] We mention that a different alternative construction was proposed in [1]. Their construction is pivoted at the Reed-Muller code over a large finite field (i.e., ℓ-variate polynomials of total degree at most m over a finite field of size at least $3\ell \cdot m$) and utilizes consistency tests that refer to a random walk among the planes in the corresponding ℓ-dimensional hyperspace.

the i^{th} proof-oracle refers to the claim that the codeword $C_\ell(x_i)$ is consistent with the codeword $C_0(x)$. On input $(i,i') \in [k^{1/\ell}]\times[k]$ and oracle access to a purported $\overline{w}$, the local decoder checks that the first block of $\overline{w}$ actually consists of repetitions of some n-bit long string, denoted w, and verifies that the purported i^{th} codeword of C_ℓ (recovered from an adequate random position in the second block) is consisted with w. The verification is performed by invoking the PCP of proximity while using the i^{th} sub-block of the third block as a proof-oracle (where the input-oracle is essentially a random sub-block of the first block). If all these checks were successful, the local tester for $C_{\ell+1}$ invokes the local tester for C_ℓ on input i', while providing it access to the foregoing purported codeword of C_ℓ. This strategy is detailed next.

Construction 6 (constructing $C_{\ell+1}$ based on C_ℓ): *Let $m = k^{1/\ell}$. Recall that given an $O(\ell)$-query relaxed LDC $C_\ell : \{0,1\}^k \to \{0,1\}^n$, where $n = k^{1+(1/\ell)+o(1)}$, we wish to construct a realxed LDC $C_{\ell+1} : \{0,1\}^{m\cdot k} \to \{0,1\}^{n'}$, where $n' = (m\cdot k)^{1+(1/(\ell+1))+o(1)}$.*

***The code*:** *For $x = (x_1, ..., x_m) \in (\{0,1\}^k)^m$, we let*

$$C_{\ell+1}(x) \stackrel{\text{def}}{=} (C_0(x)^t, (C_\ell(x_1), ..., C_\ell(x_m))^{t'}, (\pi_1(x), ..., \pi_m(x)))$$

where $\pi_j(x) = \pi(C_0(x), j, C_\ell(x_j))$ is a PCP of Proximity for the claim that $C_0(x)$ is consistent with $C_\ell(x_j)$, and $t \cdot |C(x)| = (t'/\ell)\cdot m \cdot n = m \cdot |\pi_j(x)|$.

That is, $\pi(y,j,z)$ is a proof that there exists an $x = (x_1, ..., x_m)$ such that $y = C_0(x_1, ..., x_m)$ and $z = C_\ell(x_j)$. Note that t' is set so that the C_ℓ codewords occupy an $\ell/(\ell+1)$ fraction of the length of $C_{\ell+1}$.

Note that the length of the codewords of $C_{\ell+1}$ is $O(m)\cdot(2k^{1+(1/\ell)})^{1+o(1)} = k^{1+(2/\ell)+o(1)}$, which equals $(k^{(\ell+1)/\ell})^{1+(1/(\ell+1))+o(1)}$.

***The decoder*:** *On input $(i,i') \in [m]\times[k]$, given oracle access to $((w_1, ..., w_t), (u_1, ..., u_{t'\cdot m}), (\pi_1, ..., \pi_m))$ such that $|w_1| = \cdots = |w_t| = O(mk)$, $|u_1| = \cdots = |u_{t'\cdot m}| = n$, and $|\pi_1| = \cdots = |\pi_m|$, the local decoder proceeds as follows.*

1. *It tests that $w_1 = \cdots = w_t$ by repeating the following check $O(1)$ times: Select uniformly $r', r'' \in [t]$ and $s \in [n]$, and check whether the s^{th} bit of $w_{r'}$ equals the s^{th} bit of $w_{r''}$.*
   The test is performed so that $t\cdot mn$-bit long strings that are $0.01\cdot\delta_0$-far from $\{w^t : w \in \{0,1\}^{mn}\}$ are rejected with probability at least $2/3$. Such rejection leads the decoder to halt with output $\perp$. Otherwise, the decoder selects uniformly $r \in [t]$ and $r' \in [t']$, and proceeds to the next step.
2. *It uses π_i as a proof-oracle for a PCP of proximity that verifies whether there exists an $x = (x_1, ..., x_m)$ such that $w_r = C_0(x_1, ..., x_m)$ and $u_{(r'-1)\cdot m+i} = C_\ell(x_i)$.*
   The corresponding verifier rejects if for every $x = (x_1, ..., x_m)$ either w_r is $0.1\cdot\delta_0$-far from $C_0(x)$ or $u_{(r'-1)\cdot m+i}$ is $0.1\cdot\delta_0$-far from $C_\ell(x_i)$. Such rejection leads the decoder to halt with output $\perp$. Otherwise, it proceeds to the next step.
3. *It invokes the local decoder of C_ℓ on input i' in order to recover the i'^{th} bit that is encoded by $u_{(r'-1)\cdot m+i}$.*

The local decoder rejects if either Step 1 or Step 2 rejects. Otherwise, it outputs the output obtained in Step 3.

Note that the query complexity of the foregoing decoder exceeds the query complexity of the decoder for C_ℓ only by an additive constant (i.e., $O(1/\delta_0)$). The error probability of the local decoder remains upper-bounded by 1/3 (i.e., its error probability is the maximum of the error probabilities of Steps 1–3, not their sum).

The analysis of Construction 6 is analogous to the proof of Lemma 5. The fact that Steps 1 and 2 accept with high probability implies that, with high probability, w_r is $0.2 \cdot \delta_0$-close to some codeword $C(x_1, ..., x_m)$ and that $u_{(r'-1)\cdot t'+i}$ is $0.1 \cdot \delta$-close to $C_\ell(x_i)$. Hence, if this happens, then Step 3 yields an adequate answer (per the induction hypothesis).

There is, however, a problem with the foregoing description. Specifically, this description presumes that the PCP of Proximity utilized in Step 2 uses proof-oracles that are almost linear in the length of the assertion. What we know (see, e.g., [2]) is that these proof-oracles have almost linear length in the size of the circuit that verifies the NP-claim (regarding consistency of codewords). The question at hand is what is the size of circuits computing the encodings for C_0 and C_ℓ. Since C_0 is chosen by us, we may choose an adequate code that has linear-size encoding circuits (cf. [12]). But what about C_ℓ?

The difficulty in computing C_ℓ is that it calls for computing proof-oracles for PCP of Proximity (of claims that refer to C_0 and $C_{\ell-1}$). It is quite likely that the PCPs of Proximity presented in [2] have proof-oracles that can be computed by circuits of almost linear size, but confirming this conjecture is beyond the scope of the current note. Alternatively, we observe that we can just as well use proof-oracles (and consequently codes C_ℓ's) that can be computed by almost linear size *non-deterministic circuits*, provided that these (valid) proof-oracles are unique (a.k.a canonical, see [7, Sec. 5.3] and [6, Sec. 2.4]). In such a case these proof-oracles for PCPs of randomness complexity r can be computed by a non-deterministic circuit of size $\widetilde{O}(2^r)$ (and using the bound on the randomness complexity of the PCPs of Proximity in [2] suffices). Unfortunately, verifying that the proof-oracles for the PCPs in [2] are canonical or can be made so (while maintaining the bound on the randomness complexity) is beyond the scope of the current note.

Recap. Note that C_ℓ encodes k-bit strings by codewords of length $n = k^{1+(1/\ell))+o(1)}$ and that it has a $O(\ell)$-query local decoder for decoding distance $\Omega(\delta_0/\ell)$. Specifically, the ecoding distance of C_ℓ is a $\frac{\ell-1}{\ell}$ factor of the decoding distance of $C_{\ell-1}$, which means that it is a $1/\ell$ factor of the decoding distance of C_1 (which is $\Omega(\delta_0)$).

Unravelling the recursion we get a sequence of C_0-codewords that equals the first $\ell + 1$ blocks in Construction 3, and a sequence of corresponding proof-oracles for a PCP of proximity. However, the latter proof-oracles refer to somewhat different claims than the proof-oracles in the last ℓ blocks of Construction 3.

5 Reflections

We mention that Theorem 2 is quite tight: It establishes a q-query relaxed LDC of length $n = k^{1+O(1/q)}$, whereas any q-query relaxed LDC must have length $n \geq k^{1+\widetilde{\Omega}(1/q)^2}$ (see [8, 3], and a related exposition [5]). As indicated by this note, the historical view that describes an asymptotic improvement in the length of known q-query relaxed LDCs (from $n = k^{1+O(1/q^{1/2})}$ in [2] to $n = k^{1+O(1/q)}$ in [1]) was wrong; in fact, no asymptotic improvement over the original construction of [2] has been made in twenty years. Hence, we highlight the following

Open Problem 7 (can the length of relaxed LDCs be improved?): *For any natural number $q \geq 3$, is there a q-query relaxed locally decodable code $C : \{0,1\}^k \to \{0,1\}^n$ such that $n \leq k^{1+(2/q)}$? Furthermore, can one achieve $n \leq k^{1+o(1/q)}$?*

The furthermore challenge sounds more natural, but the main challenge is very interesting per se: Although it would not improve asymptotically over $n \leq k^{1+O(1/q)}$, it would provide a separation between relaxed LDCs and plain LDCs. This is the case because q-query LDCs must have length $n = \Omega(k/\log k)^{1+\frac{1}{\lceil q/2 \rceil - 1}} > k^{1+(2/q)}$ (cf. [10, Thm. 7], improving over [9]).

Indeed, an alternative route for separating relaxed LDCs and plain LDCs is to improve the lower bound on the length of the latter. This begging and famous challenge has been with us for almost 25 years now.

References

[1] Vahid Asadi and Igor Shinkar. Relaxed Locally Correctable Codes with Improved Parameters. In *48th ICALP*, pages 18:1–18:12, 2021.

[2] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, Vol. 36 (4), pages 889–974, 2006.

[3] Marcel Dall'Agnol, Tom Gur and Oded Lachish. A Structural Theorem for Local Algorithms with Applications to Coding, Testing, and Privacy. In *32nd ACM-SIAM Symposium on Discrete Algorithms*, pages 1651–1665, 2021.

[4] Klim Efremenko. 3-query locally decodable codes of subexponential length. *SICOMP*, Vol. 41 (6), pages 1694–1703, 2012.

[5] Oded Goldreich. On the Lower Bound on the Length of Relaxed Locally Decodable Codes. *ECCC*, TR23-064, 2023.

[6] Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong Locally Testable Codes with Relaxed Local Decoders. *ACM Trans. Comput. Theory*, Vol. 11 (3), Art. 17:1–17:38, 2019.

[7] Oded Goldreich and Madhu Sudan. Locally Testable Codes and PCPs of Almost-Linear Length. *Journal of the ACM*, Vol. 53 (4), pages 558–655, 2006.
See errata on `https://www.wisdom.weizmann.ac.il/~oded/p_ltc.html`

[8] Tom Gur and Oded Lachish. On the Power of Relaxed Local Decoding Algorithms. *SIAM Journal on Computing*, Vol. 50 (2), pages 788–813, 2021.

[9] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *32nd ACM Symposium on the Theory of Computing*, pages 80–86, 2000.

[10] Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *Journal of Computer and System Science*, Vol. 69(3), pages 395–420, 2004.

[11] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-Rate Locally Correctable and Locally Testable Codes with Sub-Polynomial Query Complexity. *Journal of the ACM*, Vol. 64 (2), pages 11:1–11:42, 2017.

[12] Daniel Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, Vol. 42 (6), pages 1723–1731, 1996.

[13] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM*, Vol. 55(1), pages 1:1–1:16, 2008.