Local list decoding with a constant number of queries

Avraham Ben-Aroya  Klim Efremenko
Computer Science Dept., Computer Science Dept.,
Tel-Aviv University, Israel, 69978 Tel-Aviv University, Israel, 69978
abrahambe@post.tau.ac.il klimefrem@gmail.com

Amnon Ta-Shma
Computer Science Dept,
Tel-Aviv University, Israel, 69978
amnon@post.tau.ac.il

March 23, 2010

Abstract
Recently Efremenko showed locally-decodable codes of sub-exponential length. That result showed that these codes can handle up to $\frac{1}{4}$ fraction of errors. In this paper we show that the same codes can be locally unique-decoded from error rate $\frac{1}{2} - \alpha$ for any $\alpha > 0$ and locally list-decoded from error rate $1 - \alpha$ for any $\alpha > 0$, with only a constant number of queries and a constant alphabet size. This gives the first sub-exponential codes that can be locally list-decoded with a constant number of queries.

1 Introduction
Locally decodable codes (LDCs) are codes that allow retrieving any symbol of an encoded message by reading only a constant number of symbols from its codeword, even if a large fraction of the codeword is corrupted. Formally, a code $C$ is said to be locally decodable with parameters $(\alpha, q, \epsilon)$ if it is possible to recover any symbol $x_i$ of a message $x$ by making at most $q$ queries to $C(x)$, such that even if up to a $1 - \alpha$ fraction of $C(x)$ is corrupted, the decoding algorithm returns the correct answer with probability at least $1 - \epsilon$.

The first formal definition of locally decodable codes was given by Katz and Trevisan in [KT00]. The Hadamard code is the best-known 2-query locally decodable code, and its length is $2^n$ (where $n$ is the message length). For 2-query LDCs tight lower bounds on the block length of $2^{\Theta(n)}$ were given in [GKST02] for linear codes and in [KdW03] for general codes. For an arbitrary constant number of queries $q$, there are weak polynomial bounds, see [KT00, KdW03, Woo07].

The first sub-exponential LDCs (with a constant number of queries) were obtained by Yekhanin in [Yek08]. Yekhanin obtained 3-query LDCs with sub-exponential length under a highly believable number theoretic conjecture. Later, Efremenko [Efr09] gave an unconditional construction of sub-exponential LDCs. That construction also allowed a tradeoff between the number of queries and the codeword length. Unfortunately, these constructions could handle only $\frac{1}{q}$ fraction of errors (where $q$ is the number of queries) over a large alphabet and
over the binary alphabet. In [Woo08], Woodruff showed how to increase the handled error rate to $\frac{1}{q}$ over binary alphabets. Very recently, independent of our work, Dvir, Gopalan and Yekhanin [DGY10], showed how to handle $\frac{1}{8}$ fraction of errors.

Locally decodable codes have many applications in cryptography and complexity theory, see surveys [Tre04, Gas04]. Many of these applications require LDCs that can handle high error rates. Therefore, the question of local decoding from a high error rate attracted much attention.

The goal of this paper is to construct LDCs that can handle $1 - \alpha$ fraction of errors. Clearly, when the error rate of a code is above half its distance, it is impossible to find a unique answer. Thus, we have to consider list-decoding. A code $C$ is said to be $(1 - \alpha, L)$-list-decodable if for every word, the number of codewords within relative distance $1 - \alpha$ from that word is at most $L$. The notion of list-decoding dates back to works by Elias [Eli57] and Wozencraft [Woz58] in the 50s. Roughly speaking, a code $C$ is locally list-decodable if it is $(1 - \alpha, L)$-list-decodable, and given a corrupted word $w$, an index $k \in [L]$ and a target bit $j$, the decoder returns the $j$’th message bit of the $k$’th codeword that is close to $w$. As expected, there are some subtleties in the definition. The main issue is guaranteeing that for a fixed $k$, all answers for inputs $(k, j)$ correspond to the same codeword. More formally, a local list-decoding algorithm generates $L$ machines $\{M_k\}$, such that the machine $M_k$ locally decodes one codeword that is close to $w$, and the machines $\{M_k\}$ together cover all the codewords that are close to $w$ (for a formal definition, see Section 2).

The notion of local list-decoding is a central one in the theory of computer science. It first (implicitly) appeared in the celebrated Goldreich-Levin result [GL89], that can be seen as a local list-decoding algorithm for the Hadamard code. Later on many local list-decoding algorithms were studied. Most of the currently known locally list-decodable can be divided into three categories: Reed-Muler codes [GRS00, AS03, STV01, GKZ08], direct product and XOR codes [IW97, IJK06, IJKW08] and low-rate random codes [KS09]. Many of these results play an important role in complexity theory.

Our Results  In this paper we show how to locally list-decode the codes given in [Efr09] (and which have sub-exponential length) with only a constant number of queries. We also show that one can unique decode this code up to radius close to $\frac{1}{2}$. The code we work with is a linear code over a finite field $\mathbb{F}$ of constant-size, i.e., $|\mathbb{F}| = f(k, \alpha) = \Theta(1)$, where $f$ is some function. For unique local decoding we show:

**Theorem 1** (Unique decoding). For every $k \geq 2, \alpha > 0$, there exists a $(\frac{1}{2} + \alpha, q, \epsilon)$ LDC of dimension $n$ over $\mathbb{F}$ of length

$$\exp(\exp(O(\sqrt[3]{\log n \log \log n})^{k-1})))$$

with $q = \Theta\left(\frac{k^k \log(\frac{1}{\epsilon})}{\alpha^2 + k}\right) = \Theta(1)$ queries.

Independent of our work Dvir, Gopalan and Yekhanin in [DGY10] show a restricted version of this theorem for $\alpha \geq \frac{1}{4}$.

For local list-decoding we show:
Theorem 2 (List Decoding). For every \( k \geq 2, \alpha > 0 \), there exists a code of dimension \( n \) over \( \mathbb{F} \) of length
\[
\exp(\exp(O(\sqrt[k]{\log n (\log \log n)^{k-1}})))
\]
which is \((\alpha, L, q, \epsilon)\) locally list-decodable code with probabilistic advice. The number of queries is \( q = O(\frac{k^h \log \frac{1}{\epsilon}}{\alpha^n}) = \Theta(1) \) and the list size is \( L = |\mathbb{F}|^{O(\log \frac{1}{\epsilon})} = \text{poly}(n) \).

As we said before, the above code is a linear code over a finite field \( \mathbb{F} \) of constant-size, i.e., \(|\mathbb{F}| = f(k, \alpha) = \Theta(1)\), where \( f \) is some function. We can get a locally list-decodable binary code, by concatenating the code of Theorem 2 with a good binary code, namely,

Theorem 3. For every \( k \geq 2, \alpha > 0 \), there exists a binary code of dimension at least \( n \) and length
\[
\exp(\exp(O(\sqrt[k]{\log n (\log \log n)^{k-1}}))) \cdot |\mathbb{F}|
\]
which is \((\alpha, L, q, \epsilon)\) locally list-decodable with probabilistic advice. The number of queries is \( q = O(\frac{k^h \log \frac{1}{\epsilon}}{\alpha^n}) \cdot \text{poly}(\log |\mathbb{F}|) = \Theta(1) \) and the list size is \( L = |\mathbb{F}|^{O(\log \frac{1}{\epsilon})} = \text{poly}(n) \). Furthermore, if the field \( \mathbb{F} \) is of characteristic two, the binary code is linear.

We remark that in [Efr09] it is shown that we can use a field \( \mathbb{F} \) of characteristic 2 and of size \( f(k, \alpha) \leq 2^m \) where \( m = (k/\alpha)^{O(k)} \). With this field \( \mathbb{F} \) the resulting binary code is linear. Alternatively, using the Prime Number Theorem for arithmetic progressions it can be shown that we can use a field \( \mathbb{F} \) of prime order \( f(k, \alpha) \approx m \log m \) (for the above \( m \)), which results in a shorter code, fewer queries and shorter output lists, but produces a non-linear binary code.

The rest of the paper is organized as follows. In Section 2 we give the necessary preliminaries. Section 3 gives the formal definitions of locally decodable and list-decodable codes. In Section 4 we revise the construction of the code and analyze its local structure. Sections 5 and 6 contains the proofs of Theorems 1 and 2, respectively. The proof of Theorem 3 will appear in the full version of this paper.

2 Preliminaries

We use the following standard mathematical notation:

- \([s] = \{1, \ldots, s\}\);
- \(\mathbb{F}\) is a finite field;
- \(\mathbb{F}^*\) is the multiplicative group of the field;
- \(\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z}\), the integers modulo \( m \);
- \(\Delta(x, y)\) denotes the relative Hamming distance between vectors \( x, y \in \Sigma^n \), i.e. \(\Delta(w, w') = \Pr_{i \in [n]}[w_i \neq w'_i]\);
- \(\text{Ag}(w, w') \triangleq 1 - \Delta(w, w')\), i.e. \(\text{Ag}(w, w') = \Pr_{i \in [n]}[w_i = w'_i]\);
- \(A^B\) denotes the set of functions from \( B \) to \( A \), i.e., \(A^B = \{f : B \to A\}\). We identify \(A^{|m|}\) with \(A^m\).
A code is a function \( C : \Sigma^n \rightarrow \Sigma^\bar{n} \). We identify a code \( C \) with its image \( C = \{ C(\lambda) \mid \lambda \in \Sigma^n \} \).

The distance \( d \) of the code is the minimum distance between two codewords in \( C \) and the relative distance is \( \delta = d/n \). The Hamming balls of radius \( d/2 \) around codewords are disjoint, and therefore one can uniquely correct up to so many errors. If we allow more than \( d/2 \) errors several decodings are possible. In many cases one can allow almost up to the distance errors and still get only few possible decodings.

For \( w \in \Sigma^n \) and \( \alpha > 0 \), define

\[
\mathcal{L}_C(w, \mu) = \{ z \in C : \Delta(w, z) \leq \mu \}.
\]

When the code \( C \) is implicit from the text we abbreviate \( \mathcal{L}_C(\cdot) \) to \( \mathcal{L}(\cdot) \).

**Definition 1.** We say that a code \( C \) is \((\mu, L)\) list-decodable if for every \( w \in \Sigma^\bar{n} \) there are at most \( L \) codewords within distance \( \mu \) from \( w \), i.e. \( |\mathcal{L}(w, \mu)| \leq L \).

**Fact 4 (The Johnson bound).** Let \( C \) be a code with relative distance \( \delta \). Then, for every \( \alpha > \sqrt{1 - \delta} \), the code \( C \) is \((1 - \alpha, \alpha - (1 - \delta) \alpha^2)\) list decodable.

### 3 Locally Decodable and List Decodable Codes

As always one can study the combinatorial properties of a code, or ask for an explicit decoding algorithm. If the decoding algorithm makes only a few queries to the corrupted word, we say it is local. We begin with a formal definition of local unique decoding:

**Definition 2.** We say a probabilistic oracle machine \( M^w \) locally outputs a string \( s \) with confidence \( 1 - \epsilon \), if

\[
\forall i \Pr[M^w(i) = s_i] \geq 1 - \epsilon,
\]

where the probability is taken over the randomness of \( M \).

**Definition 3 (Local Unique Decoding).** A code \( C \) over a field \( \mathbb{F} \), \( C : \mathbb{F}^n \rightarrow \mathbb{F}^\bar{n} \) is said to be \((q, \alpha, \epsilon)\) locally decodable if there exists a probabilistic oracle machine \( M^w \) (the decoding algorithm) with oracle access to a received codeword \( w \) such that:

1. For every message \( \lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathbb{F}^n \) and for every \( w \in \mathbb{F}^\bar{n} \) such that \( \text{Ag}(C(\lambda), w) \geq \alpha^1 \), it holds that \( M^w \) locally outputs \( \lambda \) with confidence \( 1 - \epsilon \).
2. \( M^w(i) \) makes at most \( q \) queries to \( w \) for all \( i \in [n] \).

Recall that a code \( C \) is list decodable if for every codeword \( w \) there are a few codewords near \( w \). Let \( C(y_1), C(y_2), \ldots, C(y_L) \) be the list of codewords near \( w \). Roughly speaking, a code \( C \) is locally list decodable if there exists a machine \( M \) that given \( i, j \) and an oracle access to the received word \( w \), outputs the \( j \)th symbol of \( y_i \). The locality property requires that the machine \( M \) makes a few queries to \( w \). To make this formal:

**Definition 4.** Let \( C : \Sigma^n \rightarrow \Sigma^\bar{n} \) be a code. We say a set of probabilistic oracle circuits \( M_1, \ldots, M_L \) with oracle queries to \( w \), \( (\alpha, L, q, \epsilon) \) local list-decodes \( C \) at the word \( w \in \Sigma^\bar{n} \), if.

\[\text{Note that here } \alpha \text{ denotes agreement and not distance.} \]
• Every oracle circuit $M_j$ makes at most $q$ queries to the input word $w$.

• For every codeword $c \in C$ with $\text{Ag}(c, w) \geq \alpha$, there exists some $k \in [L]$, such that $M_k^w$ locally outputs $c$ with confidence $1 - \epsilon$.

**Definition 5** (Locally List Decodable Codes with deterministic advice). We say a deterministic algorithm $A(\alpha, L, q, \epsilon)$ local list-decodes the code $C : \Sigma^n \rightarrow \Sigma\bar{n}$, if on input $1^n$, $A$ outputs probabilistic oracle circuits $M_1 \ldots M_L$ which $(\alpha, L, q, \epsilon)$ local list-decode $C$ at every word $w \in \Sigma\bar{n}$.

The code $C$ is $(\alpha, L)$ list-decodable and therefore every $w \in \Sigma^n$ has at most $L \alpha$-close codewords $c_1, \ldots, c_L$. Each such codeword $c_i = C(\lambda_i)$ is represented by a probabilistic circuit $M_i$ such that $\forall j M_i(j) = \lambda_j$ (recall that $M_i$ is a probabilistic circuit, and therefore $M_i(j) = \lambda_j$ means that $M_i$ outputs $\lambda_j$ with probability at least $1 - \epsilon$). The algorithm $A$ outputs $L$ machines that are good for every $w \in \Sigma\bar{n}$. One way to think about it, is that $i \in [L]$ is an advice telling which of the $L$ solutions corresponds to the codeword we are interested in.

For instance, in [STV01] the advice string is a position in the codeword and a value of the codeword at this position. Since the code considered in [STV01] is of polynomial length, there is a polynomial number of advice strings. Therefore, the local list decoding algorithm of [STV01] works with deterministic advice. However, for codes with super-polynomial length (such as the Hadamard code and the code considered in this paper) one cannot have a position in a codeword as part of the advice string and still maintain a polynomial number of advice strings. Indeed, the local list decoding for the Hadamard code [GL89] has probabilistic advice, as we define now:

**Definition 6** (Locally List Decodable Codes with probabilistic advice). We say a probabilistic algorithm $A(\alpha, L, q, \epsilon)$ local list-decodes the code $C : \Sigma^n \rightarrow \Sigma\bar{n}$, if on input $1^n$, $A$ outputs probabilistic oracle circuits $M_1 \ldots M_L$ such that for every word $w \in \Sigma^n$, with probability $2/3$ over the random coins of $A$,

$$\forall w \in \Sigma^n \Pr_A \left[ \forall \lambda \left( \text{Ag}(C(\lambda), w) \geq \alpha \Rightarrow \exists i \forall j \Pr[M_i(j) = \lambda_j] \geq 1 - \epsilon \right) \right] \geq 2/3.$$  

In the [GL89] local list decoding algorithm, the algorithm $A$ (which generates the probabilistic circuits $M_1, \ldots, M_L$) picks a random subset of evaluation points, and takes as advice the value of the codeword on these evaluation points. Therefore, the [GL89] local list decoding algorithm works with probabilistic advice. Notice the order of the quantifiers: for every $w \in \Sigma^n$ most sets (i.e., random coins of $A$) are good for $w$, however, it is not the case that most choices of $A$ are good for every $w$.

### 4 The Code

In this section we will define the code and study its local properties.

#### 4.1 Definition of the Code

We first review the definition of the code from [Efr09]. Fix a composite number $m = p_1 \cdot p_2 \ldots p_k$ which is a product of $k$ distinct primes. The definition of the code will depend only on $m$.  

5
In order to define the code we need the following definition:

**Definition 7.** A family of vectors \( \{ u_i \}_{i=1}^n \subseteq \mathbb{Z}_m^h \) is said to be \( S \)-matching if the following conditions hold:

1. \( \langle u_i, u_i \rangle = 0 \) for every \( i \in [n] \).
2. \( \langle u_i, u_j \rangle \in S \) for every \( i \neq j \).

Grolmusz [Gro00] showed how to construct a large set of \( S \)-matching vectors \( \{ u_i \}_{i=1}^n \), \( u_i \in \mathbb{Z}_m^h \), for 
\[ S = \{ x \in \mathbb{Z}_m \setminus \{ 0 \} \mid \forall i, \ x \text{ mod } p_i \in \{ 0, 1 \} \} . \]

Let \( F \) be a field that contains an element \( \gamma \in F \) of order \( m \), i.e. \( \gamma^m = 1 \) and \( \gamma^i \neq 1 \) for \( i < m \). We define a code \( C : F^n \to F^{m^h} \), where we think of a codeword as a function from \( \mathbb{Z}_m^h \) to \( F \). The encoding of the message \( \lambda_1, \lambda_2, \ldots, \lambda_n \) is the function:
\[
C(\lambda_1, \lambda_2, \ldots, \lambda_n)(x) \triangleq \sum_{i=1}^n \lambda_i \gamma^{\langle u_i, x \rangle} .
\]

Equivalently, we can write
\[
C(\lambda_1, \lambda_2, \ldots, \lambda_n) = \sum_{i=1}^n \lambda_i f_i , \tag{1}
\]
where \( f_i(x) \triangleq \gamma^{\langle u_i, x \rangle} \). We will denote the codeword length by \( \bar{n} = m^h \). An asymptotic relation between \( n \) and \( \bar{n} \) is:
\[
\bar{n} = \exp(\exp(O(\sqrt[4h]{\frac{\log n}{\log \log n}})^{k-1}))).
\]

Note that the asymptotic rate of the code depends only on \( k \), the number of different primes dividing \( m \).

For simplicity sometimes we denote \( G \triangleq \mathbb{Z}_m^h \).

### 4.2 Local Properties of the Code

In this subsection we study local properties of the code. Specifically we study the restriction of the code to lines.

**Definition 8.** (line) Let \( v, u \in G \). The line through \( v \) in direction \( u \) is the function \( \ell = \ell_{v,u} \in G^{[m]} \) defined by \( \ell(t) = v + tu \).

**Definition 9** (restriction). Let \( \ell \in G^{[m]} \) be a line.

- For a function \( f \in F^G \), the restriction of \( f \) to \( \ell \), denoted by \( f|_\ell \in F^{m^h} \) is defined by \( f|_\ell(t) = f(\ell(t)) \).
- For a code \( C : F^n \to F^G \), the restriction of \( C \) to \( \ell \), denoted by \( C|_\ell : F^n \to F^m \), is the vector space \( \{ C(\lambda)|_\ell \mid \lambda \in F^n \} \).
Now, we analyze the restriction of the code in direction $u_j$. Observe that
\[
\mathcal{C}(\lambda_1, \ldots, \lambda_n)(v + tu_j) = \sum_i \lambda_i \gamma^{(u_i, v + tu_j)} = \sum_i \lambda_i \gamma^{(u_i, v)} (\gamma^{(u_i, v)})^t
\]
\[
= \sum_{b \in S \cup \{0\}} \left[ \sum_{i: (u_i, u_j) = b} \lambda_i \gamma^{(u_i, v)} \right] (\gamma^t)^b.
\]

Define $p : \mathbb{F} \to \mathbb{F}$ by $p(x) = \sum_{b \in S \cup \{0\}} a_b x^b$, where $a_b = \sum_{i: (u_i, u_j) = b} \lambda_i \gamma^{(u_i, v)}$, then $\mathcal{C}|_{\ell_{u_j}}(\lambda)(t) = p(\gamma^t)$. Furthermore, $a_0 = \lambda_j \gamma^{(v, v)}$, and so when $\lambda_j \neq 0$, $p$ is a non-zero polynomial.

**Lemma 5.** Let $\mathcal{C}$ be the code above. For every $v \in G$ and $j \in [n]$, the code $\mathcal{C}|_{\ell_{u_j}} : \mathbb{F}^m \to \mathbb{F}^m$ is of dimension at most $2^k$ and distance $d \geq 1 - \sum_{i=1}^{k} \frac{1}{p_i}$.

**Proof:** In order to prove the lemma we need to show that the polynomial $p(x) = \sum_{b \in S \cup \{0\}} a_b x^b$ does not have too many roots in the set $H = \{ \gamma^t 0 \leq i < m \}$. Recall that the set $S$ is
\[
S = \{ x \in \mathbb{Z}_m \setminus \{0\} \mid \forall i \mod p_i \in \{0, 1\} \}.
\]
Notice that $p$ might have a large degree, and therefore might have a large number of roots in $\mathbb{F}$. Nevertheless, we show that the number of roots $p$ has in $H$ is at most $\sum_{i=1}^{k} \frac{m}{p_i}$. To see that denote $\tilde{p}(x) = p(x^\frac{\sum_{i} m}{p_i})$. We show that $\tilde{p}$ has the same number of roots as $p$. Let $s = \sum_{i} \frac{m}{p_i}$. Then,
\[
s \mod p_i = \frac{m}{p_i} \mod p_i \neq 0.
\]
Therefore, $\gcd(s, m) = 1$, that is, $s$ is invertible in $\mathbb{Z}_m$. This implies the mapping $\psi : H \to H$, $\psi(x) = x^s$ is a bijection.

Thus, in order to show $p$ has few roots in $H$, it suffices to show that $\tilde{p}$ is a low-degree polynomial. Each monomial of $\tilde{p}$ is of degree $b \cdot s \mod m$ for some $b \in S \cup \{0\}$. Notice that for every $1 \leq i, j \leq k$,
\[
\frac{m}{p_i} \cdot b \mod p_j = \begin{cases} 0 & j \neq i \\ 0 & (j = i) \land (b \mod p_i = 0) \\ \frac{m}{p_i} \mod p_i & (j = i) \land (b \mod p_i = 1) 
\end{cases}
\]
This implies that for every $i$,
\[
\frac{m}{p_i} \cdot b \mod m = \begin{cases} 0 & b \mod p_i = 0 \\ \frac{m}{p_i} & b \mod p_i = 1 
\end{cases}
\]
Hence, $b \cdot s \mod m \leq \sum_{i} \frac{m}{p_i} \mod m \leq \sum_{i} \frac{m}{p_i}$. We conclude that $\tilde{p}$ has at most $\sum_{i} \frac{m}{p_i}$ roots in $H$ and therefore so does $p$.

For a polynomial $p : \mathbb{F} \to \mathbb{F}$ define the vector $\overline{p} \in \mathbb{F}^m$ by $\overline{p}(t) = p(\gamma^t)$. Then, $\mathcal{C}|_{\ell_{u_j}}$ is a linear subspace of the vector-space $\text{Span}\{ \overline{x}^b : b \in S \cup \{0\} \}$, which is a dimension $2^k$ $\mathbb{F}$-subspace. Every non-zero codeword corresponds to a non-zero polynomial that can have at most $\sum_{i} \frac{m}{p_i}$ roots. As the elements $\gamma^t$ are distinct for $1 \leq t \leq m$, every codeword has at most that many zeroes.
Let \( \mathcal{C} \) be the code above. Let \( v \in G \) and \( j \in [n] \). Then every codeword of \( \mathcal{C}|_{\ell_{v,u_j}} \) corresponds to a polynomial with \( 2^k \) monomials, where the free coefficient is \( \lambda_j \gamma^{(u_j,v)} \). Thus, any restricted codeword \( z \in \mathcal{C}|_{\ell_{v,u_j}} \) contains information about \( \lambda_j \).

**Definition 10.** Using the above notation, we denote \( D_{v,j}(z) = \lambda_j \).

In particular,

**Corollary 6.** Let \( \mathcal{C} \) be the code above. Let \( v \in G \) and \( j \in [n] \). If \( z, z' \in \mathcal{C}|_{\ell_{v,u_j}} \) and \( D_{v,j}(z) \neq D_{v,j}(z') \) then \( z \neq z' \) and therefore \( \Delta(z, z') \geq \delta \).

Another corollary is,

**Corollary 7.** The distance of the code \( \mathcal{C} \) is at least \( \delta \), where \( \delta = 1 - \sum_{i=1}^k \frac{1}{p_i} \).

**Proof:** Look at two different codewords \( \mathcal{C}(\lambda) \) and \( \mathcal{C}(\tilde{\lambda}) \) for some \( \lambda \neq \tilde{\lambda} \). Then, there exists some \( j \in [n] \) such that \( \lambda_j \neq \tilde{\lambda}_j \). We can now partition \( G \) to disjoint lines in direction \( u_j \). From Corollary 6 it follows that on each of these lines the restrictions of \( \mathcal{C}(\lambda) \) and \( \mathcal{C}(\tilde{\lambda}) \) are different. From Lemma 5 we know that the distance on each of these lines is at least \( \delta \). It follows that the distance between \( \mathcal{C}(\lambda) \) and \( \mathcal{C}(\tilde{\lambda}) \) is at least \( \delta \).

**Remark 8.** By taking all \( p_i \)'s of the same order we get that \( \delta = 1 - O\left(\frac{k}{\sqrt{m}}\right) \). In this paper we assume that \( m \) is such a product.

## 5 Local unique decoding

We are given some word \( w \in \mathbb{F}^G \) that has agreement \( \frac{1}{2} + \alpha \) with some codeword \( \mathcal{C} = \mathcal{C}(\lambda) \). We are also given some \( j \in [n] \). Our goal is to recover (with a good probability) \( \lambda_j \). A first attempt at local decoding is restricting the code to a random line \( \ell_{v,u_j} \) in direction \( u_j \). Intuitively, this is a good step because we restrict the global code to a small fragment of constant size \( m \), while still keeping information about \( \lambda_j \). Specifically, by Lemma 5, \( \mathcal{C}|_{\ell_{v,u_j}} \) is a linear code with a large distance, and by Corollary 6, a codeword \( z = \mathcal{C}(\lambda)|_{\ell_{v,u_j}} \in \mathcal{C}|_{\ell_{v,u_j}} \) corresponds to a polynomial with \( g^k \) monomials, where the free coefficient is \( \lambda_j \gamma^{(u_j,v)} \).

As \( \mathcal{C}(\lambda) \) has \( \frac{1}{2} + \alpha \) agreement with \( w \), when we pick a random line in direction \( u_j \), the expected agreement between \( w|_{\ell_{v,u_j}} \) and \( \mathcal{C}(\lambda)|_{\ell_{v,u_j}} \) is \( \frac{1}{2} + \alpha \). The problem is that it may still happen that with high probability the agreement between \( w|_{\ell_{v,u_j}} \) and \( \mathcal{C}(\lambda)|_{\ell_{v,u_j}} \) is less then \( \frac{1}{2} \) and we will decode wrong value. In order to overcome this problem we can sample \( K = O\left(\frac{\log(\frac{1}{\alpha})}{\alpha^2}\right) \) independent lines. Then with high probability the agreement between \( w \) and \( \mathcal{C}(\lambda) \) is at least \( \frac{1}{2} + \frac{\alpha}{2} \) on the sampled lines. Note that the code \( \mathcal{C}(\lambda) \) restricted to the union of independent lines in direction \( u_j \) may not have a good distance, as two different codewords may coincide on a restriction to a line. However, for any two codewords \( \mathcal{C}(\lambda) \) and \( \mathcal{C}(\tilde{\lambda}) \), where \( \lambda_j \neq \tilde{\lambda}_j \), the distance between the restrictions of these two codewords on each line must be large because of Corollary 6.

Let \( \alpha \geq 2(1-\delta) \) (where \( \delta \) is the distance of the code, and by Lemma 5 is at least \( 1 - \sum_i \frac{1}{p_i} \)).

The unique decoding algorithm for \( \frac{1}{2} + \alpha \) agreement is as follows:
Input:
- $w \in \mathbb{F}^G$ that has agreement $\frac{1}{2} + \alpha$ with some codeword $C$,
- $j \in [n]$,
- $\epsilon > 0$

Randomness: A set of $K = \Theta\left(\frac{\log\left(\frac{1}{\epsilon}\right)}{\alpha^2}\right)$ random elements in $G$, $\overline{v} = (v_1, \ldots, v_K) \in G$.

Queries: For each $k \in [K]$, the algorithm queries all points on the line $\ell_{v_k,u_j}$.

Algorithm: For every $k \in [K]$ and for every symbol $\sigma \in \mathbb{F}$, the algorithm computes
$$\text{weight}_k(\sigma) = \max \left\{ \text{Ag}(w, z) : z \in C|_{\ell_{v_k,u_j}}, D_{v_k,j}(z) = \sigma \right\}.$$  

The algorithm then computes $\text{weight}(\sigma) = \frac{1}{K} \sum_{k=1}^{K} \text{weight}_k(\sigma)$. The output of the algorithm is the symbol $\sigma$ with the highest weight.

**Theorem 9.** Assume $\alpha \geq 2(1 - \delta)$. For every $\lambda \in \mathbb{F}^n$, $w \in \mathbb{F}^G$ with $\text{Ag}(w, C(\lambda)) \geq \frac{1}{2} + \alpha$ and every $j \in [n]$, 
$$\text{Pr}_{\overline{v}}[\text{The algorithm outputs } \lambda_j] \geq 1 - \epsilon.$$  

The algorithm uses $\Theta\left(\frac{\log\left(\frac{1}{\epsilon}\right)}{\alpha^2} \cdot m\right)$ queries.

**Proof:** Suppose that $C(\lambda)$ is a codeword which has $\frac{1}{2} + \alpha$ agreement with the received word $w$. Then 
$$\mathbb{E}_{v \in G} \left[ \text{Ag}(w|_{\ell_{v,u_j}}, C(\lambda)|_{\ell_{v,u_j}}) \right] = \frac{1}{2} + \alpha.$$  

We say $\overline{v} = (v_1, \ldots, v_k)$ is good, if 
$$\frac{1}{K} \sum_{k=1}^{K} \left[ \text{Ag}(w|_{\ell_{v_k,u_j}}, C(\lambda)|_{\ell_{v_k,u_j}}) \right] \geq \frac{1 + \alpha}{2}.$$  

By a standard application of the Chernoff bound, 
$$\text{Pr}_{\overline{v}}[\text{overline{v} is not good}] \leq 2^{-\Omega(\alpha^2 K)} = \epsilon.$$  

We now prove that if $\overline{v}$ is good the algorithm outputs the correct answer. Denote $ag_k = \text{Ag}(w|_{\ell_{v_k,u_j}}, C(\lambda)|_{\ell_{v_k,u_j}})$. Then,
- For every $k$, $\text{weight}_k(\lambda_j) \geq \text{Ag}(w|_{\ell_{v_k,u_j}}, C(\lambda)|_{\ell_{v_k,u_j}}) \geq ag_k$ and so $\text{weight}(\lambda_j) \geq \mathbb{E}_k[ag_k] \geq \frac{1 + \alpha}{2}$,
- Fix any $\sigma \neq \lambda_j$ and $k \in [K]$. Let $z \in C|_{\ell_{v_k,u_j}}$ be such that $D_{v_k,j}(z) = \sigma$. Then, by the triangle inequality, 
$$\delta \leq \Delta(z, C(\lambda)|_{\ell_{v_k,u_j}}) \leq \Delta(C(\lambda)|_{\ell_{v_k,u_j}}, w|_{\ell_{v_k,u_j}}) + \Delta(w|_{\ell_{v_k,u_j}}, z).$$  

Thus, $\Delta(w|_{\ell_{v_k,u_j}}, z) \geq \delta + ag_k - 1$, and $\text{weight}_k(\sigma) \leq 1 - ag_k + 1 - \delta$. In particular, 
$$\text{weight}(\sigma) \leq 1 - \delta + \mathbb{E}_k[1 - ag_k] \leq \frac{1}{2} + 1 - \frac{\alpha}{2} = \frac{1}{2}.$$  

9
Thus, whenever \( \tau \) is good the algorithm outputs \( \lambda_j \).

We are now ready to prove Theorem 1.

**Proof of Theorem 1:** The code \( \mathcal{C} \) has distance at least \( \delta = 1 - O\left(\frac{k}{m^{1/r}}\right) \) and the code length is

\[
\exp(\exp(O(\frac{k}{\log n}\log(\log n)^{k-1}))).
\]

We take \( m \) to be a product of \( m \) almost equal primes. From Theorem 9, for every \( \alpha \geq 2(1-\delta) = O\left(\frac{k}{m^{1/r}}\right) \), the code is \( (\frac{1}{2} + \alpha, q, \epsilon) \) locally decodable with \( q = \Theta\left(\frac{\log(\frac{1}{\epsilon})}{\alpha^2} \cdot m\right) \) queries. We think of \( k \) as a constant, and \( m \) as depending on \( \alpha \), growing to accommodate the required error rate. Thus \( \alpha = 2(1-\delta) \approx \frac{2k}{m^{1/r}}, \) or equivalently, \( m \approx (\frac{2k}{\alpha})^k \). Thus, the number of queries is \( \Theta\left(\frac{m\log(\frac{1}{\epsilon})}{\alpha^2}\right) = \Theta(k^k \cdot \alpha^{-(k+2)} \cdot \log(\frac{1}{\epsilon})) \). For \( k = 2 \) the number of queries is \( \Theta(\alpha^{-4} \cdot \log(\frac{1}{\epsilon})) \).

---

### 6 Local list decoding with probabilistic advice

We first remind the reader of the setting. A probabilistic algorithm \( A \) has to produce \( L \) probabilistic circuits \( M_1, \ldots, M_L \) that \( (\alpha, L, q, \epsilon) \) local list-decode \( \mathcal{C} \). \( A \) uses its internal random coins to sample a random subset \( \Lambda \subseteq G \) of cardinality \( \Theta\left(\frac{\log n}{\alpha} \right) \). The list size \( L \) is \( |\mathbb{F}^\Lambda| \) and corresponds to all possible values a codeword may take on \( \Lambda \). We identify an index of a machine \( i \in [L] \) with a function \( \text{ad} : \Lambda \mapsto \mathbb{F} \) of values of a codeword on the set \( \Lambda \). The machine \( M_{\text{ad}} \) locally outputs a message \( \lambda \) such that \( C(\lambda) \) has \( \alpha \) agreement with \( w \) and \( \text{ad} = C(\lambda)|\Lambda \).

Given a corrupted word \( w \in \mathbb{F}^G \) and a value \( j \in [n] \), \( M_{\text{ad}} \)'s goal is to find (the hopefully unique) codeword \( c \in C \) that is \( \alpha \) close to \( w \), and that is consistent with the given advice \( \text{ad} \in \mathbb{F}^\Lambda \). To do so, \( M_{\text{ad}} \) does the following: \( M_{\text{ad}} \) picks \( K \) (and \( K \) will turn out to be constant) random lines in direction \( u_j \) that pass through some point in \( \Lambda \). For each such line, \( M_{\text{ad}} \) queries \( w \) on the line, and finds all the restricted codewords that are close to the \( w \) (on the line). We say that a line is good if among all those codewords, exactly one matches the value \( \text{ad} \) gives on the point from \( \Lambda \). For each good line, \( M_{\text{ad}} \) extracts from this unique codeword the value \( \lambda_j \) and adds it to the candidates list. The output of \( M_{\text{ad}} \) is the most common value in the candidates list. More formally, the algorithm \( M_{\text{ad}} \) is defined as follows.

**A’s random coins:** A random subset \( \Lambda \) of of cardinality \( \Theta\left(\frac{\log \frac{1}{\epsilon}}{\alpha^2}\right) \).

**Advice:** Values of some codeword \( c \) on \( \Lambda \).

**Input:** \( w \in \mathbb{F}^G, j \in [n] \).

**M’s randomness:** A random subset \( \{s_1, \ldots, s_K\} \) of \( \Lambda \) of cardinality \( K = \Theta\left(\frac{\log(\frac{1}{\epsilon})}{\alpha}\right) \).

**Queries:** For each \( k \in [K], M \) queries the values of \( w \) on the \( K \) lines \( \ell_{s_k, u_j} \).

**Algorithm:** For every \( k \), the algorithm goes over all codewords of \( \mathcal{C}' = \mathcal{C}|\ell_{s_k, u_j} \). For every such \( k \), if there exists exactly one codeword \( z \) of \( \mathcal{C}' \) with:

- \( \text{Ag}(z, w|\ell_{s_k, u_j}) \geq \frac{\alpha}{2}, \) and,
• \( z(s_k) = \text{ad}(s_k) \)

then the algorithm adds the value \( D_{v,j}(z) \) to the candidates list.

**Output:** The most common value in the candidates list.

**Theorem 10.** For any \( \alpha \geq 8\sqrt{1-\delta} \), \( \epsilon > 0 \) and \( L = |\mathbb{F}^\Lambda| = q^{O\left( \frac{\log \frac{1}{\alpha}}{\alpha} \right)} \), \( q = Km = O\left( \frac{m \log \left( \frac{1}{\alpha} \right)}{\alpha} \right) = O\left( \log \left( \frac{1}{\epsilon} \right) \right) \). The above algorithm is a probabilistic polynomial-time \((\alpha, L, q, \epsilon)\) local list-decoding algorithm.

Theorem 2 follows immediately from Theorem 10.

**Proof of Theorem 2:** We take \( m \) a product of \( k \) distinct almost equal primes. From Theorem 10 we know that for any \( \alpha > 8\sqrt{1-\delta} = O\left( \frac{\sqrt{k}}{\sqrt{m}} \right) \) the code is \((\alpha, L, q, \epsilon)\) local list-decodable with \( q = O\left( \frac{m \log \left( \frac{1}{\alpha} \right)}{\alpha} \right) \). Therefore, \( m = O\left( \frac{k^2}{\alpha} \right) \) and \( q = O\left( k^{k} \cdot \alpha^{-2k+1} \cdot \log \left( \frac{1}{\epsilon} \right) \right) \) with a codeword length:

\[
\exp\left( \exp\left( O\left( k^{k} \log n \log \log n^{k-1} \right) \right) \right)
\]

We are left to prove Theorem 10.

### 6.1 Proof of correctness

We need to show that for every received word \( w \), with high provability over the choice of the set \( \Lambda \), for every codeword \( c = C(\lambda) \) that has \( \alpha \) agreement with \( w \), when the advice is \( \text{ad} = c|_\Lambda \), it holds that for every \( j \in [n] \), \( \Pr[M_{ad}^w(j) = \lambda_j] \geq 1 - \epsilon \), where the probability is over the randomness of \( M \).

Fix \( w \in \mathbb{F}^G \), a codeword \( c = C(\lambda) \) and \( j \in [n] \). For \( v \in G \) the machine \( M_{ad} \) considers the set

\[
U_j(v) = \left\{ z \in C|_{\ell_{v,u_j}} : (\text{Ag}(z, w|_{\ell_{v,u_j}}) \geq \alpha/2) \land (z(v) = \text{ad}(v)) \right\}.
\]

In the \( k \)th iteration, \( M_{ad} \) adds \( \lambda_j \) to the candidates list if \( U_j(s_k) = \left\{ c|_{\ell_{v,u_j}} \right\} \).

We say that \( v \) is **useful** if \( c|_{\ell_{v,u_j}} \in U_j(b) \). Notice that \( c|_{\ell_{v,u_j}}(v) = \text{ad}(v) \), hence for \( v \) to be useful we only need a high agreement between \( v \) and \( w \) on the line \( \ell_{v,u_j} \). We say that \( v \) **filters** if \( U_j(v) \subseteq \left\{ c|_{\ell_{v,u_j}} \right\} \), i.e., for any codeword in the restricted code \( z \in C|_{\ell_{v,u_j}} \) such that \( z \neq c \) it holds that \( z \notin U_j(v) \).

**Lemma 11.** For any \( \alpha \geq 8\sqrt{1-\delta} \) it holds that

- \( \Pr_{v \sim G}[v \text{ is useful}] \geq \frac{\alpha}{2} \)
- \( \Pr_{v \sim G}[v \text{ does not filter}] \leq \frac{4 \alpha}{\alpha} \cdot (1 - \delta) \leq \frac{\alpha}{16} \).
Proof: Since $E_v[Ag(w|_{\ell_v,u_j}, c|_{\ell_v,u_j})] = \alpha$, an averaging argument implies that the probability $v \in G$ is useful is at least $\alpha/2$.

We turn to the second item. A point $v$ does not filter if there is a restricted codeword $z \in C|_{\ell_v,u_j}$ such that $z \not= c|_{\ell_v,u_j}$ and $z \in U_j(v)$. A restricted codeword $z$ is in $U_j(v)$ if it is in the list $L(w|_{\ell_v,u_j}, \alpha/2)$ and $z(v) = c(v)$. One way to choose $v$ uniformly from $G$ is by first choosing a random line $\ell$ in direction $u_j$, and then choosing a random point $v$ on the line. For any line $\ell$ in direction $u_j$, $C' = C|_{\ell}$ has distance $\delta$. Therefore, for any $z \not= c$ the probability that $z(v) = c(v)$ is at most $1 - \delta$. By the Johnson bound (see Fact 4), the number of codewords with $\alpha/2$ agreement with $w|_{\ell}$ satisfies

$$|L(w|_{\ell_v,u_j}, \alpha/2)| \leq \frac{\alpha/2 - (1-\delta)}{\alpha^2/4 - (1-\delta)} \leq \frac{4}{\alpha},$$

when $\alpha \geq 2\sqrt{2(1-\delta)}$. The probability that such a codeword $z$ agrees with $c$ at $v$ is at most $1 - \delta$. The lemma follows from the union bound. 

\begin{definition}
For $w \in \mathbb{F}^G$, a set $\Lambda \subseteq G$ is good for $w$, if for every $c \in L(w, \alpha/2)$ and every $j \in [n]$,

\begin{itemize}
  \item $Pr_{v \in \Lambda}[v \text{ is useful and filters for } (w, c, j)] \geq \frac{\alpha}{7}$.
  \item $Pr_{v \in \Lambda}[v \text{ does not filter } (w, c, j)] \leq \frac{\alpha}{8}$.
\end{itemize}
\end{definition}

\begin{lemma}
Fix $w \in \mathbb{F}^G$. Pick a set $\Lambda$ uniformly at random from $G$. The probability $\Lambda$ is not good for $w$ is at most $\frac{\alpha}{n} \cdot 2^{-\Omega(\alpha|\Lambda|)}$.
\end{lemma}

Proof: For any $w, j$ and $c \in L(w, \alpha)$, the probability a single $v$ is useful and filters, by Lemma 11, is at least $\frac{\alpha}{7}$. By the Chernoff bound, the probability we do not have $\frac{\alpha}{7}$ fraction of good vectors in the sample $\Lambda$ is at most $2^{-\Omega(\alpha|\Lambda|)}$.

Similarly, by Lemma 11, for any $w, j$ and $c \in L(w, \alpha)$, the probability a single $v$ does not filter $(w, c, j)$, is at most $\frac{\alpha}{8}$. By the Chernoff bound, the probability we have more than $\frac{\alpha}{8}$ fraction of vectors that do not filter $(w, c, j)$ in the sample $\Lambda$ is at most $2^{-\Omega(\alpha|\Lambda|)}$.

The lemma follows from a union bound over $j$ and $c \in L(w, \alpha)$.

Assume $\Lambda$ is good for $w$. The probability that at the $i$th iteration, $M_{ad}$ adds the correct value $\lambda_j$ to the candidates list is at least the probability that $v$ is useful and filters. By Definition 11 this probability is at least $\frac{\alpha}{7}$. The probability that $M_{ad}$ adds a wrong value to the candidates list is bounded by the probability that $v$ does not filter, which is at most $\frac{\alpha}{8}$. Therefore, by the Chernoff bound, it follows that after $\Theta(\frac{\log(\frac{1}{\alpha})}{\alpha})$ iterations the probability that $\lambda_j$ is the most common value in the candidates list is at least $1 - \epsilon$. Theorem 10 follows from the above lemma, since for every $w$, $\Lambda$ is good for $w$ with probability at least $\epsilon$ (by the choice of the cardinality of $\Lambda$).

Remark 13. If $\Lambda$ is chosen to be a random subset $G$ of size $K = \Theta(\frac{\log(\frac{1}{\alpha})}{\alpha})$ (which is constant) then $M_{ad}$ becomes deterministic. In this case, the above lemma shows the for any fixed $j$, the machine $M_{ad}$ outputs the correct answer with probability at least $1 - \epsilon$. However, the definition requires that $M_{ad}$ outputs the correct answer for every $j$. 

12
References


