IEICE TRANS. INF. & SYST., VOL.E98–D, NO.3 MARCH 2015

PAPER Special Section on Foundations of Computer Science—New Spirits in Theory of Computation and Algorithm—

A Fourier-Analytic Approach to List-Decoding for Sparse Random Linear Codes

Akinori KAWACHI[†a)], *Member and* **Ikko YAMANE**[†], *Nonmember*

SUMMARY It is widely known that decoding problems for random linear codes are computationally hard in general. Surprisingly, Kopparty and Saraf proved query-efficient list-decodability of sparse random linear codes by showing a reduction from a decoding problem for sparse random linear codes to that for the Hadamard code with small number of queries even under high error rate [11]. In this paper, we show a more direct list-decoding algorithm for sparse random linear codes with small number of queries from a Fourier-analytic approach.

key words: *list-decoding, Fourier analysis*

1. Introduction

The error-correcting code is one of the most significant technologies in the field of computer science. Recently, the so-called list-decoding algorithms have been intensively studied in the coding theory and theoretical computer science since it revealed deep connections among computational complexity theory, cryptography and coding theory. (See [15], [16] for the connections.)

While the standard decoding for a code C is required to output a single correct message x from a received word w that is provided by adding errors to codeword $C(x)$, the task of list-decoding is to output a short list of candidate messages that contains the correct x from w even if it is impossible to uniquely decode x from w because of high error rate.

Historically, the concept of list-decoding was proposed in the context of coding theory by Elias and Wozencraft [4], [17]. However, it was in the field of cryptography that the first efficient list-decoding algorithm for an explicit error-correcting code was discovered by Goldreich and Levin [6].

The Goldreich-Levin theorem provides a general construction for a hardcore predicate, which generates a cryptographic pseudorandom bit, from any one-way function. The statement of the Goldreich-Levin theorem was quite far from notions of coding theory, but the proof of the theorem essentially constructed an efficient algorithm, which we call the GL algorithm here, that list-decodes the Hadamard code. This list-decoding algorithm is developed by a clever combination of strong tools from complexity theory and cryptography such as pair-wise independent sampling and self-correcting of the Hadamard code.

Manuscript received March 28, 2014.
Manuscript revised August 1, 2014.

[†]The authors are with Tokyo Institute of Technology, Tokyo, 152–8552 Japan.
a) E-mail: kawachi@is.titech.ac.jp

DOI: 10.1587/transinf.2014FCP0016

Later, the Goldreich-Levin theorem was reproved by an entirely different approach. Kushilevitz and Mansour developed an efficient simple algorithm for learning decision trees using a Fourier-analytic technique [9]. It turned out afterwards that the Kushilevitz-Mansour algorithm (KM algorithm) was applicable to list-decoding for the Hadamard code.

The KM algorithm is not only mathematically elegant but also suggestive of a wide range of applications. As mentioned above, the KM algorithm was originally developed for machine learning, and thus, it has essentially shown an application of the list-decoding algorithm to machine learning. Furthermore, Akavia, Goldwasser and Safra showed a unified framework that provides simple proofs for many previously known hardcore predicates by extending the KM algorithm rather than the GL algorithm [1]. From these examples, we can say that KM algorithm has a great potential to find new applications of list-decoding.

After the discovery of the connections, list-decoding algorithms have been given to a lot of error-correcting codes. (See [7], [8] for details.) Recently, studies of random linear codes have proceeded rapidly in the context of list-decoding [5], [10], [11]. It is known that the random linear codes can achieve excellent performance with high probability, but the decoding problem is strongly believed to be computationally hard in general.

For example, the decoding problem is known to be NP-hard [3] even in an approximate version with some preprocessing [12]. Even in the case of average-case hardness, it is shown that some integer lattice problems, which are also believed to be hard, reduce to decoding random linear codes in some setting [2]. From such reasons, the problem of decoding random linear codes has been applied to cryptographic constructions (See, e.g., [13], [14]).

As mentioned above, the problem of decoding random linear codes is widely believed to be hard. However, Kopparty and Saraf most recently showed how to list-decode sparse random linear codes efficiently from a viewpoint of query complexity, i.e., the number of bits to which an algorithm makes access in a received word [11].

They provided a reduction from decoding random linear codes to decoding the Hadamard code, which is efficiently list-decodable by the GL and KM algorithms.

In this paper, we construct a more direct algorithm that list-decodes sparse random linear codes efficiently in the same sense as Kopparty and Saraf's result from a Fourier-analytic approach by extending the KM algorithm. This

provides a new mathematical viewpoint of list-decoding for random linear codes as in the case of the KM algorithm for the Hadamard code.

Let us observe more details of the previous results and ours below. We define an inner product $\langle \cdot, \cdot \rangle$ as $\langle x, y \rangle = x_1 \cdot y_1 + \cdots + x_n \cdot y_n \bmod 2$, where x_i denotes the i-th coordinate of $x \in \{0,1\}^n$. Then, a codeword of the Hadamard code of a message x is defined as $\mathrm{Had}(x) := (\langle x, 0\cdots0\rangle, \ldots, \langle x, 1\cdots1\rangle)$, or equivalently, $\mathrm{Had}(x) := xG_{\mathrm{Had}}$, where the generating matrix $G_{\mathrm{Had}} \in \{0,1\}^{n\times 2^n}$ consists of all the column vectors of n bits in the lexicographical order.

Note that the length of $\mathrm{Had}(x)$ is exponential in n. We need to construct a decoding algorithm that only reads a part of a received word for efficiency of decoding. Indeed, the GL and KM algorithms give efficient local list-decoding algorithms for the Hadamard code, which means that the algorithms can output a short list of candidate messages only by locally reading a small portion of a long received word.

The result of Kopparty and Saraf provides some reduction from decoding random linear codes to decoding the Hadamard code. The generating matrix of random linear code C of length N is given by a random matrix $G_C \in \{0,1\}^{n\times N}$. We say C is sparse if $N \geq 2^{\delta n}$ for some constant $\delta > 0$. Then, we can identify $C(x)$ as random N bits sampled from $\mathrm{Had}(x)$. (Note that a set of column vectors in G_C is a subset of those in G_{Had}.) If C is sparse, their reduction can simulate access to $\mathrm{Had}(x)$ from $C(x)$ even under high error rate by randomly XORing a constant number of bits sampled from $C(x)$ under some special distribution.

Therefore, the combination of their reduction with the GL algorithm can work as a local list-decoding algorithm for sparse random linear codes. The number of bits that the algorithm reads is quite small, i.e., the algorithm is query-efficient, but we need to solve some sampling problem from the special distribution for the reduction. If we wish to make the algorithm time-efficient, i.e., to run it in polynomial time in message length n, we need to solve the sampling problem efficiently in the sense of time complexity. As discussed in [11], it is unknown whether this sampling can be done efficiently in the sense of time complexity so far.

Suppose that a received word is of error rate $(1-\varepsilon)/2$, that is, the received word is generated by flipping $(1-\varepsilon)N/2$ bits of some codeword. The Kopparty-Saraf reduction provides a list-decoding algorithm that outputs a list of $\varepsilon^{-O(1)}$ candidate messages, which contains a correct message with a constant probability, say $2/3$, by making $\varepsilon^{-O(1)} \cdot O(n \log n)$ queries to a received word of error rate $(1-\varepsilon)/2$. The time complexity is $\varepsilon^{-O(1)} \cdot O(T_{\mathrm{KS}}(n)\, n \log n))$, where $T_{\mathrm{KS}}(n)$ denotes the time complexity for determining a query position by solving the above sampling problem. (In fact, the GL algorithm can produce the i-th bit of the j-th candidate message for any given i, j by reading only $\varepsilon^{-O(1)}$ bits of a received word, and therefore, the list-decoding algorithm from their reduction with the GL algorithm can obtain a list of candidate messages by repeatedly running it for every bit of the candidate messages.)

Our algorithm, which is an extension of the KM algorithm and we call the extended KM algorithm below, is a local list-decoder for sparse random linear codes as in the case of the KM algorithm for the Hadamard code.

The (extended) KM algorithm is based on the depth-first search for candidate messages with pruning. Our main technical contribution is to provide a new criterion suitable to sparse random linear codes for the pruning from a viewpoint of the Fourier analysis.

Similarly to the reduction of Kopparty and Saraf, we need to solve some other sampling problem for applying the criterion, and thus, even in the extended KM algorithm, it is also unknown whether we can list-decode sparse random linear codes time-efficiently.

The extended KM algorithm outputs a list of $\varepsilon^{-O(1)}$ candidate messages with a constant probability by making $\varepsilon^{-O(1)} \cdot O(n \log n)$ queries to a received word of error rate $(1-\varepsilon)/2$. The time complexity is $\varepsilon^{-O(1)} \cdot O(T_{\mathrm{EKM}}(n)\, n \log n)$, where $T_{\mathrm{EKM}}(n)$ denotes the time complexity for determining a query position by solving the sampling problem in our case.

Comparing Kopparty and Saraf's and our approaches, they have polynomial list size in ε^{-1} and polynomial query complexity in ε^{-1} and n. Their time complexities depend on those of the sampling problems. By trivial algorithms, we have $T_{\mathrm{KS}}(n) \leq 2^{O(n)}$ and $T_{\mathrm{EKM}}(n) \leq 2^{O(n)}$.

However, we will show an efficient sampling procedure that approximately solves our sampling problem by assuming an efficient sampling procedure for Kopparty and Saraf's problem in this paper. This would suggest that our sampling problem is easier and our Fourier analytic approach has a higher potential to discover efficient list-decoding algorithms for random linear codes.

In addition, the KM algorithm provided new applications of list-decoding algorithms to cryptography and machine learning, as mentioned above. Our approach extending the KM algorithm would also have a potential to new applications to other fields like the KM algorithm.

The rest of this paper is organized as follows. In Sect. 2, we introduce basic notions and notation necessary to show the extended KM algorithm, and we briefly review the KM algorithm. In Sect. 3, we describe the extended KM algorithm and prove its correctness and performance. We conclude this paper with some remarks on the sampling problems of Kopparty and Saraf and of ours in Sect. 4.

2. Preliminaries

We deal with only $\mathbb{F}_2 := \{0,1\}$ as alphabet in this paper. For $k \geq 1$, we denote by w_i the i-th coordinate of a row vector, or a word, $w = (w_1, \ldots, w_k)$. Let $|w|$ denote the length of w and let vw denote a concatenation of two vectors v and w. We denote by 0^k the word of k zeros. It is also denoted by $\mathbf{0}$ if it is clear from the context. The relative Hamming weight of a word w is defined as $\mathrm{wt}(w) := |\{i \in [k] \mid w_i \neq 0\}|/|w|$, where $[k] := \{1, \ldots, k\}$. The relative Hamming distance is defined as $\Delta(v, w) := \mathrm{wt}(v - w)$.

2.1 List-Decoding Problem

We discuss decoding problems for error-correcting codes in this paper. Our error model is the same as the previous work [11], i.e., bit-flipping errors $e \in \mathbb{F}_2^N$ are adversarially added to a codeword $C(x) \in \mathbb{F}_2^N$ (of a message $x \in \mathbb{F}_2^n$) at communication from a sender to a receiver. In this model, we can only guarantee that the number of the errors (and equivalently the error rate $\mathrm{wt}(e)$) is bounded by some predetermined value. We say a received word $w = C(x) + e$ has an advantage $\varepsilon \in (0, 1)$ if $\mathrm{wt}(e) \leq (1 - \varepsilon)/2$.

The list-decoding problem that we consider in this paper is the problem of outputting a short list of candidate messages from a given received word:

Problem 2.1 (List-Decoding for C) Given a received word $w \in \mathbb{F}_2^N$ as an oracle and advantage $\varepsilon \in (0, 1)$, find a list L satisfying $\{x \in \mathbb{F}_2^n \mid \Delta(C(x), w) \leq (1 - \varepsilon)/2\} \subseteq L$.

Performance of list-decoding algorithms is often evaluated from two measures, the list size and the number of queries. The list size is given by $|L|$, i.e., the number of elements in L. The number of queries is given by the number of the bits that the decoding algorithm makes access to the received word.

In the standard setting of error-correcting codes, decoding algorithms are normally supposed to read all the bits in a given received word. However, it is significant for applications in theoretical computer science to suppose decoding algorithms that only make local access to a received word. In this setting, the received word can be formulated as an oracle $w : [N] \to \mathbb{F}_2$. When a decoding algorithm makes a query i to the oracle w, it answers the i-th bit w_i of the received word. Thus, the number of queries is defined by how many times the decoding algorithm asks the oracle to output L.

2.2 Random-Linear Codes

The main target of this paper is to decode random linear codes. A code C is a map from a message space to a codeword space. In this paper, we focus on binary codes $C : \mathbb{F}_2^n \to \mathbb{F}_2^N$. For a message $x \in \mathbb{F}_2^n$, $C(x)$ denotes the corresponding codeword of x. Then, message length of C is n and codeword length N. We sometimes identify the map C as a set of codewords, i.e., $C := \{C(x) \in \mathbb{F}_2^N : x \in \mathbb{F}_2^n\}$.

A random linear code is given from a random generating matrix $G \in \mathbb{F}_2^{n \times N}$. Every entry of G is determined uniformly at random over $\mathbb{F}_2$. The linear code is defined from the generating matrix as $C = \{xG \in \mathbb{F}_2^N : x \in \mathbb{F}_2^n\}$. We say the code C is sparse if $2^n \leq \mathrm{poly}(N)$ for message length n and codeword length N. Namely, the codeword length of a sparse code is exponentially longer than the message length.

The bias of a code C is defined as $\mathrm{bias}(C) := \max_{c \in C - \{\mathbf{0}\}} |1/2 - \mathrm{wt}(c)|$, and we say C is unbiased if there is a constant $\gamma > 0$ such that $\mathrm{bias}(C) \leq N^{-\gamma}$. It is easy to show the following proposition by the Chernoff bound and union bound.

Proposition 2.2: For every $\beta \in (0, 1)$ and every $n, N \in \mathbb{N}$, we have $\mathrm{bias}(C) \leq \beta$ with probability at least $1 - O(2^n \exp\{-\beta^2 N/6\})$.

Actually, the extended KM algorithm we show in this paper can work for every unbiased linear codes as well as Kopparty and Saraf's result [11]. From the above theorem, the extended KM algorithm can also work for sparse random linear codes with high probability, and thus, we only focus on decoding unbiased linear codes rather than sparse random-linear codes.

2.3 Fourier Analysis

The Fourier analysis is a strong tool for our target. In this section, we briefly review fundamentals of the Fourier analysis to be used in our analysis.

Below, we consider randomized functions. If we invoke a randomized function $f : \mathbb{F}_2^k \to \mathbb{R}$ on a fixed input $x \in \mathbb{F}_2^k$, it outputs an independent random sample in $\mathbb{R}$. Then, $f(x)$ denotes the sample in $\mathbb{R}$ and a random variable over $\mathbb{R}$ of the output of f on x. Let $\mathcal{F}_k$ be a set of randomized functions $f : \mathbb{F}_2^k \to \mathbb{R}$ for $k \in \mathbb{N}$. For two randomized functions $f, g \in \mathcal{F}_k$, we define the inner product $\langle f, g \rangle := \mathbf{E}_{X \sim U_k, f(X), g(X)}[f(X)g(X)]$, where U_k is the uniform distribution over $\mathbb{F}_2^k$. Since a (deterministic) function is a special case of the randomized functions, we can similarly define the inner products of functions and of randomized and deterministic ones.

For $x, i \in \mathbb{F}_2^k$, let $\chi_x(i) := (-1)^{\langle x, i \rangle}$, where $\langle x, i \rangle = x_1 i_1 + \cdots + x_k i_k$. It is easy to check that $\sum_{v \in \mathbb{F}_2^k} \chi_v(i) = 2^k \mathbf{E}_{X \sim U_k}[\chi_i(X)] = 2^k \delta_k(i)$, and $\langle \chi_x, \chi_y \rangle = \delta_k(x + y)$ if $|x| = |y|$, where $\delta_k(x) = 1$ if $x = 0^k$ and $\delta_k(x) = 0$ otherwise. $\{\chi_x\}_{x \in \mathbb{F}_2^k}$ is an orthonormal basis, called the Fourier basis. We call each χ_x a Fourier basis vector. From a property of an orthogonal basis, we have $f(y) = \sum_{x \in \mathbb{F}_2^k} \langle f, \chi_x \rangle \chi_x(y)$.

For technical reasons, we exploit some extensions of the inner product and self-convolution in this paper.

Definition 2.3 (μ-semi-inner product, μ-self-convolution) Let μ be any distribution over $\mathbb{F}_2^k$. For $f, g \in \mathcal{F}_k, l \in \mathbb{N}$, we define μ-semi-inner product of f and g as $\langle f, g \rangle_\mu := \mathbf{E}_{Y \sim \mu, f(Y), g(Y)}[f(Y)g(Y)]$. For $Y^{(1)}, \ldots, Y^{(l)} \overset{\text{i.i.d.}}{\sim} \mu$, denoting by $\mu^{(l)}$ the distribution according to $Y^{(1)} + \cdots + Y^{(l)}$, we define l-th μ-self-convolution of f as $f^{[\mu, l]}(y) := \mathbf{E}\left[\prod_{i=1}^{l} f(Y^{(i)}) \mid \sum_{i=1}^{l} Y^{(i)} = y\right]$, where the expectation is taken over the random variables $Y^{(1)}, \ldots, Y^{(l)}$ and $f(Y^{(1)}), \ldots, f(Y^{(l)})$, if $\mu^{(l)}(y) > 0$, and as $f^{[\mu, l]}(y) := 0$ if $\mu^{(l)}(y) = 0$. Hence, $f^{[\mu, l]}$ is a deterministic function.

Note that U_k-inner product and U_k-self-convolution coincides with the standard inner product and self-convolution. We can show the following proposition as a basic property of these notions.

Proposition 2.4: For every $f \in \mathcal{F}_k$, every $x \in \mathbb{F}_2^k$ and every distribution μ over $\mathbb{F}_2^k$, we have $\left\langle f^{[\mu,l]}, \chi_x \right\rangle_{\mu^{(l)}} = \langle f, \chi_x \rangle_\mu^l$.

Proof. Let $Y \sim \mu^{(l)}$ and $Y^{(1)}, \ldots, Y^{(l)} \overset{\text{i.i.d.}}{\sim} \mu$. Then, we have

$$\begin{aligned}
&\left\langle f^{[\mu,l]}, \chi_x \right\rangle_{\mu^{(l)}} = \mathbf{E}_Y \left[f^{[\mu,l]}(Y) \chi_x(Y) \right] \\
&= \mathbf{E}_Y \left[\mathbf{E}_{\{Y^{(i)}\}_{i=1}^l, \{f(Y^{(i)})\}_{i=1}^l} \left[\prod_{i=1}^l f(Y^{(i)}) \;\middle|\; \sum_{i=1}^l Y^{(i)} = Y \right] \chi_x(Y) \right] \\
&= \mathbf{E}_{\{Y^{(i)}\}_{i=1}^l, \{f(Y^{(i)})\}_{i=1}^l} \left[\prod_{i=1}^l f(Y^{(i)}) \chi_x(Y^{(i)}) \right] \\
&= \prod_{i=1}^l \mathbf{E}_{Y^{(i)}, f(Y^{(i)})} \left[f(Y^{(i)}) \chi_x(Y^{(i)}) \right] \\
&= \langle f, \chi_x \rangle_\mu^l.
\end{aligned}$$

□

2.4 Kushilevitz-Mansour Algorithm

The Kushilevitz-Mansour algorithm (KM algorithm) was originally developed for learning decision trees [9]. This provides an efficient list-decoding algorithm for the Hadamard code from a Fourier-analytic approach.

First, we review a Fourier-analytic property of the Hadamard code. As described in Sect. 1, a codeword of the Hadamard code is defined as $\mathrm{Had}(x) = (\langle x, 0\cdots0\rangle, \ldots, \langle x, 1\cdots1\rangle)$ for a message $x \in \mathbb{F}_2^n$.

Recall the definition of a Fourier basis vector $\chi_x(i) = (-1)^{\langle x,i\rangle}$. Then, a vector $(\chi_x(0\cdots0), \ldots, \chi_x(1\cdots1))$ exactly corresponds to $\mathrm{Had}(x)$ by conversions $0 \mapsto +1$ and $1 \mapsto -1$. If a received word w is error-free, decoding w corresponds to finding a Fourier basis vector χ_x satisfying $\langle \chi_x, \tilde{w} \rangle = 1$ for $\tilde{w} : i \mapsto (-1)^{w_i}$ since the Fourier basis is orthogonal.

Even if adding relatively small errors to the codeword, we can expect $\langle \chi_x, \tilde{w} \rangle$ is relatively large. So, the list-decoding problem for the Hadamard code can be formalized by the following problem:

Problem 2.5 (Enumeration problem of Fourier basis vectors) Given a randomized function $f : \mathbb{F}_2^n \to [-1, +1]$ as an oracle $i \mapsto f(i)$ and a threshold $\varepsilon \in (0, 1)$, find a list $L \subseteq \mathbb{F}_2^n$ satisfying $\{x \in \mathbb{F}_2^n \mid \varepsilon \le |\langle f, \chi_x \rangle|\} \subseteq L$.

The KM algorithm actually solves this problem efficiently.

Now we explain the behavior of the KM algorithm briefly. We consider the following complete binary tree. Each node corresponds to a distinct element in $\mathbb{F}_2^{\le n}$ and the root corresponds to the null word. The node $\alpha \in \mathbb{F}_2^{<n}$ has two children $\alpha 0$ and $\alpha 1$. The leaves correspond to messages in $\mathbb{F}_2^n$. We say $x \in \mathbb{F}_2^n$ is good if it satisfies $\varepsilon \le |\langle f, \chi_x \rangle|$.

The KM algorithm traverses this tree like the depth-first search, but the algorithm evaluates at every node α if a subtree rooted at α has good leaves or not by some criterion. If the algorithm decides the subtree has no good leaves, it prunes the subtree.

In order to give the criterion for pruning, we define an estimator $S_2^{\mathrm{KM}}(\alpha) := \sum_{\alpha' \in \mathbb{F}_2^{n-k}} \langle f, \chi_{\alpha\alpha'} \rangle^2$ for $\alpha \in \mathbb{F}_2^k$. If $S_2^{\mathrm{KM}}(\alpha) < \varepsilon^2$, the subtree rooted at α has no good leaves. Thus, the algorithm can prune the subtree. Due to pruning, we can show that the number of the nodes the KM algorithm visits is bounded by $\mathrm{poly}(n, 1/\varepsilon)$, and thus, the size of the list that the algorithm outputs is also bounded by $\mathrm{poly}(n, 1/\varepsilon)$.

Notice that it possibly takes exponential time to exactly apply the criterion $S_2^{\mathrm{KM}}(\alpha) < \varepsilon^2$ for pruning. However, by using the orthogonal property of the Fourier basis, the criterion S_2^{KM} can be transformed to the following form: $S_2^{\mathrm{KM}}(\alpha) = \mathbf{E}_{X^{(1)}, X^{(2)} \overset{\text{i.i.d.}}{\sim} U_n, f(X_1), f(X_2)} [f(X^{(1)}) f(X^{(2)}) \chi_\alpha(X^{(1)}{}_{\le k} + X^{(2)}{}_{\le k}) \mid X^{(1)}{}_{>k} + X^{(2)}{}_{>k} = \mathbf{0}]$, where $X_{\le k}$ denotes $(X_1, \ldots, X_k)$ for $X = (X_1, \ldots, X_n)$ and $X_{>k}$ denotes $(X_{k+1}, \ldots, X_n)$. From the above form, we can efficiently approximate $S_2^{\mathrm{KM}}(\alpha)$ by sampling $X^{(1)}$ and $X^{(2)}$. By applying the approximate criterion efficiently at each node, we can run the KM algorithm efficiently in total.

3. Fourier-Analytic List-Decoding Algorithm

As mentioned in Section 2.2, a sparse random linear code is unbiased with high probability, and thus, our goal is reduced to a construction of list-decoding algorithm for unbiased linear code. The following is the main theorem of this paper, which gives a query-efficient list-decoding algorithm for unbiased linear codes.

Theorem 3.1: There exists a randomized algorithm that solves Problem 2.1 for any unbiased linear code C with probability at least $2/3$. Then, the algorithm outputs a list of size at most $\varepsilon^{-O(1)}$ by at most $\varepsilon^{-O(1)} \cdot O(n \ln n)$ queries in $\varepsilon^{-O(1)} \cdot 2^{O(n)}$ time.

We prove this theorem in this section.

In order to prove this theorem, we reduce Problem 2.1 to another problem, Problem 3.2, in Section 3.1, and then, we construct the extended KM algorithm in Section 3.2. We also analyze the algorithm in Section 3.3.

If $1/\varepsilon = 2^{\Omega(n)}$ a trivial algorithm suffices for the proof of Theorem 3.1. Thus, we suppose $1/\varepsilon = 2^{o(n)}$ in what follows.

3.1 Enumeration Problem of Fourier Basis Vectors for μ-Semi-Inner Product

We now show a reduction from Problem 2.1 to a somewhat general problem using μ-semi-inner product:

Problem 3.2 (Enumeration problem of Fourier basis vectors for μ-semi-inner product) Given a randomized function $f : \mathbb{F}_2^n \to [-1, 1]$ as an oracle $i \mapsto f(i)$ and threshold $\varepsilon \in (0, 1)$, find a list $L \subseteq \mathbb{F}_2^n$ satisfying the following two conditions: (1) $\{x \in \mathbb{F}_2^n \mid \varepsilon \le |\langle f, \chi_x \rangle_\mu|\} \subseteq L$ and (2) $|L| = \mathrm{poly}(n, 1/\varepsilon)$, where μ is a distribution over $\mathbb{F}_2^n$.

Let $G := (g_1, \ldots, g_N)$ be a generating matrix of a linear

code C and let V be a multi-set $\{g_1{}^{\mathsf{T}}, \ldots, g_N{}^{\mathsf{T}}\}$. (Hence, it may hold that $g_i{}^{\mathsf{T}} = g_j{}^{\mathsf{T}}$ for some $i \neq j$ in V.) For a received word w, we define a randomized function $f_w : \mathbb{F}_2^n \to [-1, 1]$ as $f_w(x) := (-1)^{w_i}$ for $i \sim U_x$, where U_x is the uniform distribution over $\{i \in [N] : g_i = x\}$. We then have $f_w(x) := 0$ for $x \notin V$ with probability 1. We define a distribution μ associated with C as the uniform distribution over V. Then, the following lemma holds:

Lemma 3.3: If the Problem 3.2 for μ associated with C can be solved with Q queries and list size $|L|$, the Problem 2.1 can be solved with Q queries and list size $|L|$.

Proof. Our goal is to solve Problem 2.1 on a given received word $w \in \mathbb{F}_2^N$ and advantage $\varepsilon \in (0, 1)$. Note that an answer $f_w(x)$ from an oracle f_w to a query x can be simulated with an oracle $w : i \mapsto w_i$ by making 1 query to w.

Suppose that we have an algorithm for Problem 3.2. This algorithm outputs a list L on f_w and ε for a distribution μ associated with C. By a simple calculation, if $\Delta(w, C(a)) \leq (1 - \varepsilon)/2$ for a candidate message a, we have $\varepsilon \leq \langle f_w, \chi_a \rangle_\mu$. Since $a \in L$ from the condition (1) of Problem 3.2, L is a solution to Problem 2.1. □

3.2 Construction of Extended KM Algorithm

The most naive approach to solving Problem 3.2 is to directly apply the original KM algorithm. However, this approach fails since the original KM algorithm tries to find good leaves defined from the standard inner product, while Problem 3.2 requires us to find good leaves in the sense of the semi-inner product.

More specifically, if we apply the KM algorithm to Problem 3.2 in our setting for the list-decoding that codeword length N of a code C is quite smaller than 2^n, say, $N = 2^{.1n}$, by setting μ to a distribution associated with C, the threshold given to the algorithm can be exponentially small. Let $C(x) = (\langle x, g_1 \rangle, \ldots, \langle x, g_N \rangle)$ be a codeword of a message $x \in \mathbb{F}_2^n$ and let w be a received word given from $C(x)$ satisfying $\Delta(C(x), w) \leq (1 - \varepsilon)/2$. Note that $\chi_x(g_i) = (-1)^{C(x)_i}$. Therefore, we have $|\langle f_w, \chi_x \rangle| = N/2^n |\langle f_w, \chi_x \rangle_\mu| \geq 2^{-.9n}\varepsilon$, where f_w is the randomized function given below Problem 3.2. Then, the KM algorithm needs exponentially many queries to the oracle f_w for more precise approximation.

The next attempt is to fit the algorithm to μ-semi-inner product. We replace the estimator S_2^{KM} in the original KM algorithm to $S_2(\alpha) := \sum_{\alpha' \in \mathbb{F}_2^{n-k}} \langle f_w, \chi_{\alpha\alpha'} \rangle_\mu^2$ for a distribution μ associated with C. Then, we need to prune a subtree rooted at α if (an approximated value of) $S_2(\alpha)$ is lower than some threshold. Unfortunately, the pruning with $S_2(\alpha)$ does not work correctly from the following reason.

As briefly reviewed in Section 2.4, the original KM algorithm decides if the subtree rooted at α should be pruned or not by the estimator S_2^{KM} computed from the standard inner products $\langle \cdot, \cdot \rangle$ between an oracle f and leaves, or equivalently, Fourier basis vectors, of the subtree. Recall $\langle \chi_a, \chi_b \rangle = 1$ if $a = b$ and $\langle \chi_a, \chi_b \rangle = 0$ otherwise. If f is close enough to some leaf x of the subtree rooted at α (namely, the subtree has a good leaf x), $S_2^{\mathrm{KM}}(\alpha)$ takes a large value since so does $\langle f_w, \chi_x \rangle$. Furthermore, S_2^{KM} should be low enough if x is not among the leaves of the subtree rooted at α (namely, the subtree has no good leaf), since $\langle f_w, \chi_y \rangle$ should be almost zero for every $y \neq x$. Therefore, the KM algorithm can distinguish whether the subtree should be pruned or not by approximating S_2^{KM}.

In the case of the μ-semi-inner product for a distribution μ associated with C, $\langle \chi_a, \chi_b \rangle_\mu = 1$ if $a = b$, but we cannot say $\langle \chi_a, \chi_b \rangle_\mu = 0$ for $a \neq b$ if $\mathrm{bias}(C) > 0$. Therefore, if f is close enough to χ_x, $\langle f_w, \chi_x \rangle_\mu$ is high, but $\langle f_w, \chi_y \rangle_\mu$ for $y \neq x$ can be relatively high depending on $\mathrm{bias}(C)$. As a result, it would be impossible to distinguish between two cases whether α has good leaves or not since there would be no gap at S_2 between these two cases.

In order to circumvent this difficulty, we introduce an l-th estimator

$$S_l(\alpha) := \sum_{\alpha' \in \mathbb{F}_2^{n-k}} \langle f, \chi_{\alpha\alpha'} \rangle_\mu^l$$

for a randomized function f, where l is an even number. This coincides with S_2 if $l = 2$. The parameter l controls how much we amplify gaps among absolute values of the semi-inner product. One can see that, if $\langle f_w, \chi_x \rangle$ is large and $y \neq x$, a ratio $\langle f_w, \chi_x \rangle_\mu{}^l / \langle f_w, \chi_y \rangle_\mu{}^l$ grows up rapidly in l. This property enables us to distinguish whether α has good leaves or not query-efficiently by balancing the number of queries for approximating S_l and the gap amplification depending on $\mathrm{bias}(C)$.

It is easy to see that $\varepsilon^l \leq S_l(\alpha)$ holds for every prefix α if $\varepsilon \leq |\langle f_w, \chi_a \rangle_\mu|$ for $a \in \mathbb{F}_2^n$. Therefore, even if we use the condition $S_l(\alpha) < \varepsilon^l$ for pruning, the algorithm does not fail to find the correct message with high probability.

We give the description of the extended KM algorithm as Algorithm 1. Similarly to the original KM algorithm, the algorithm performs pruning with approximated values of $S_l(\alpha)$ for query-efficiency. This approximation is given by $\bar{S}_l^f(\alpha)$ in Algorithm 2. The definition of the distribution $\mu|_{\leq k}^l$ will be given later.

Algorithm 1 $\mathrm{EKM}^f(\alpha, \varepsilon)$

1: **if** $\bar{S}_l^f(\alpha, \varepsilon) < \varepsilon^l/2$ **then return** . //Pruning
2: **else if** $|\alpha| = n$ **then** Output α.
3: **else** $\mathrm{EKM}^f(\alpha 0, \varepsilon)$, $\mathrm{EKM}^f(\alpha 1, \varepsilon)$.
4: **end if**

Algorithm 2 $\bar{S}_l^f(\alpha, \varepsilon)$

1: $m \leftarrow O(\varepsilon^{-2l} \ln(n/\varepsilon)), k \leftarrow |\alpha|$
2: $(z^{(i,1)}, \ldots, z^{(i,l)}) \leftarrow \mu|_{\leq k}^l$ for $i \in [m]$
3: **return** $\frac{1}{m} \sum_{i=1}^m f\left(z^{(i,1)}\right) \cdots f\left(z^{(i,l)}\right) \chi_\alpha\left(z^{(i,1)}{}_{\leq k} + \cdots + z^{(i,l)}{}_{\leq k}\right)$

3.3 Correctness and Performance of the Extended KM Algorithm

We prove the correctness and performance of the extended KM algorithm in this section. Below, let λ denote the null string and let $\rho(\mu) := \max_{a\neq b}|\langle\chi_a,\chi_b\rangle_\mu| \quad (a,b\in\mathbb{F}_2^n)$.

Theorem 3.4: $\mathrm{EKM}^f(\lambda,\varepsilon)$ solves Problem 3.2 with probability at least 2/3 for every distribution μ over $\mathbb{F}_2^n$ if $\rho(\mu) = 2^{-\Omega(n)}$ and $1/\varepsilon = 2^{o(n)}$. $\mathrm{EKM}^f(\lambda,\varepsilon)$ then outputs the list of size $O(\varepsilon^{-l})$ by at most $O(n\varepsilon^{-3l}\ln(n/\varepsilon))$ queries for some constant $l>0$ that depends solely on $\rho(\mu)$.

$\rho(\mu)$ intuitively measures non-orthogonality of the Fourier basis vectors with respect to the semi-inner product $\langle\cdot,\cdot\rangle_\mu$. The smaller $\rho(\mu)$ becomes, the stronger orthogonality the Fourier basis vectors have with respect to the semi-inner product.

Note that $\rho(\mu) = 2\,\mathrm{bias}(C)$ holds for a distribution μ associated with a code C since $\rho(\mu) = \max_{a\neq b}|\mathbf{E}_{X\sim\mu}[\chi_{a-b}(X)]| = \max_{a\neq b}|\mathbf{E}_{X\sim\mu}[(-1)^{\langle a,X\rangle}]| = \max_{c\in C-\{\mathbf{0}\}}|\mathbf{E}_{X\sim\mu}[(-1)^{c_X}]| = \max_{c\in C-\{\mathbf{0}\}}|1-2\,\mathrm{wt}(c)| = 2\,\mathrm{bias}(C)$. Since C is unbiased and sparse, we have $\mathrm{bias}(C)\le N^{-\gamma}\le 2^{-\gamma\delta n}$ for some constants γ and δ from the definitions. Thus, $\rho(\mu)=2^{-\Omega(n)}$. It follows that Theorem 3.4 implies Theorem 3.1 immediately. (For the time complexity, see Remark 3.14.)

The algorithm EKM uses an approximated version $\bar{S}_l^f(\alpha,\varepsilon) < \varepsilon^l/2$ of the criterion $S_l(\alpha) < \varepsilon^l$ for pruning. First, we prove how precisely $\bar{S}_l^f$ can approximate S_l. Let $Y^{(1)},\dots,Y^{(l)}\sim\mu$ be independent random variables and let $\mu|_{\le k}^l$ be a distribution of $(Y^{(1)},\dots,Y^{(l)})$ under the condition ${Y^{(1)}}_{>k}+\cdots+{Y^{(l)}}_{>k}=\mathbf{0}$. We define $E_l(\alpha) := \mathbf{E}\left[f(Z^{(1)})\cdots f(Z^{(l)})\chi_\alpha(Z^{(1)}_{\le k}+\cdots+Z^{(l)}_{\le k})\right]$, where the expectation is taken over $(Z^{(1)},\dots,Z^{(l)})\sim\mu|_{\le k}^l$ and $f(Z^{(1)}),\dots,f(Z^{(l)})$. In fact, $E_l(\alpha)$ is close to $S_l(\alpha)$ for sufficiently large l.

Lemma 3.5: If $\rho(\mu)=2^{-\Omega(n)}$ and $1/\varepsilon=2^{o(n)}$, there exists some constant even number $l>0$ such that we have $|S_l(\alpha)-E_l(\alpha)|\le\varepsilon^l/8$ for every $k\le n$ and every $\alpha\in\{0,1\}^k$.

In addition to Lemma 3.5, assuming that the randomized algorithm $\bar{S}_l^f(\alpha,\varepsilon)$ approximates $E_l(\alpha)$, we can prove that $\bar{S}_l^f(\alpha,\varepsilon)$ also approximates $S_l(\alpha)$ from the triangle inequality. Therefore, our task is to prove the following lemma.

Lemma 3.6: We say $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds at α if the outcome of $\bar{S}_l^f(\alpha,\varepsilon)$ invoked in $\mathrm{EKM}^f(\lambda,\varepsilon)$ satisfies $|\bar{S}_l^f(\alpha,\varepsilon)-E_l(\alpha)|\le\varepsilon^l/8$. Let VISIT be a set of nodes α that $\mathrm{EKM}^f(\lambda,\varepsilon)$ takes as an input in all the recursive invocations. Then, $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds at every $\alpha\in\mathrm{VISIT}$ with probability at least 2/3 for some even number $l\in\mathbb{N}$.

Furthermore, we can show the following lemma on the output of EKM and the number of queries.

Lemma 3.7: If $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds at every $\alpha\in\mathrm{VISIT}$, the following three hold: (a) $\{x\in\mathbb{F}_2^n \mid \varepsilon\le\langle f,\chi_x\rangle_\mu\}\subseteq L$, (b) $|L|\le O(\varepsilon^{-l})$, and (c) $\mathrm{EKM}^f(\lambda,\varepsilon)$ makes at most $O(n\varepsilon^{-3l}\ln(n/\varepsilon))$ queries.

Assuming these lemmas, we can immediately prove Theorem 3.4. So, we now prove the lemmas. We identify a distribution μ over $\mathbb{F}_2^n$ as a function $\mu: x\mapsto\Pr_{X\sim\mu}[X=x]$ and a vector $(\mu(0\cdots0),\dots,\mu(1\cdots1))\in[0,1]^{2^n}$ in the proofs. Also, let $Y\sim\mu^{(l)}$ be a random variable.

Proof of Lemma 3.5. This lemma immediately follows from Claims 3.8, 3.9 and 3.10. We show these three claims below.

Claim 3.8: $S_l(\alpha)=2^{n-k}\Pr[Y_{>k}=\mathbf{0}]E_l(\alpha)$.

Proof. For $f\in\mathcal{F}_s, g\in\mathcal{F}_t$, we define $f\bullet g\in\mathcal{F}_{s+t}$ as $f\bullet g: xy\mapsto f(x)g(y)$. Note that $\chi_{xy}=\chi_x\bullet\chi_y$ holds for every $x\in\mathbb{F}_2^s, y\in\mathbb{F}_2^t$. Then, we have by Proposition 2.4

$$\begin{aligned}S_l(\alpha)&=\sum_{\alpha'}\left\langle f^{[\mu,l]},\chi_{\alpha\alpha'}\right\rangle_{\mu^{(l)}}=\left\langle f^{[\mu,l]},\chi_\alpha\bullet\sum_{\alpha'}\chi_{\alpha'}\right\rangle_{\mu^{(l)}}\\&=2^{n-k}\left\langle f^{[\mu,l]},\chi_\alpha\bullet\delta_{n-k}\right\rangle_{\mu^{(l)}}.\end{aligned}$$

Note that $\sum_{\alpha'}\chi_{\alpha'}(x)=\sum_{\alpha'}\chi_x(\alpha')=2^{n-k}\delta_{n-k}(x)$, where $\delta_{n-k}(x)=1$ if $x=0^{n-k}$ and $\delta_{n-k}(x)=0$ otherwise. Furthermore, we have

$$\begin{aligned}&\left\langle f^{[\mu,l]},\chi_\alpha\bullet\delta_{n-k}\right\rangle_{\mu^{(l)}}\\&=\mathbf{E}\left[f^{[\mu,l]}(Y)(\chi_\alpha\bullet\delta_{n-k})(Y)\right]\\&=\mathbf{E}\left[f^{[\mu,l]}(Y)\chi_\alpha(Y_{\le k})\,\delta_{n-k}(Y_{>k})\right]\\&=\Pr[Y_{>k}=\mathbf{0}]\,\mathbf{E}\left[f^{[\mu,l]}(Y)\chi_\alpha(Y_{\le k})\mid Y_{>k}=\mathbf{0}\right]\\&=\Pr[Y_{>k}=\mathbf{0}]E_l(\alpha).\end{aligned}$$

Therefore, the claim holds. □

Claim 3.9: $|\Pr[Y_{>k}=\mathbf{0}]-1/2^{n-k}|\le 2^n\rho(\mu)^l$.

Proof. Firstly, we have

$$\left|\Pr[Y_{>k}=\mathbf{0}]-1/2^{n-k}\right|\le\sum_{x\in\mathbb{F}_2^n}\left|\mu^{(l)}(x)-U_n(x)\right|.$$

Secondly, from the Fourier expansion and the fact that $U_n=\chi_{0^n}/2^n$, we have

$$\begin{aligned}&\left|\mu^{(l)}(x)-U_n(x)\right|\\&=\left|\sum_{a\in\mathbb{F}_2^n}\left(\left\langle\mu^{(l)},\chi_a\right\rangle-\langle\chi_{0^n}/2^n,\chi_a\rangle\right)\chi_a(x)\right|\\&\le\sum_{a\in\mathbb{F}_2^n}\left|\left\langle\mu^{(l)},\chi_a\right\rangle-\langle\chi_{0^n}/2^n,\chi_a\rangle\right|\\&=\sum_{a\neq0^n}\left|\left\langle\mu^{(l)},\chi_a\right\rangle\right|.\end{aligned}$$

Thirdly, $|\langle \mu^{(l)}, \chi_a \rangle| \leq 2^{-n}\rho(\mu^{(l)})$ for every $a \neq 0^n$ holds from the definition of $\rho(\cdot)$ and the following calculation:

$$\begin{aligned}\left\langle \mu^{(l)}, \chi_a \right\rangle &= \mathbf{E}_{X\sim U_n}[\mu^{(l)}(X)\chi_a(X)] \\ &= 2^{-n}\sum_{x\in\mathbb{F}_2^n} \mu^{(l)}(x)\chi_a(x) \\ &= 2^{-n}\,\mathbf{E}_{Y\sim\mu^{(l)}}[\chi_a(Y)] \\ &= 2^{-n}\langle \chi_{0^n}, \chi_a\rangle_{\mu^{(l)}}.\end{aligned}$$

Finally, we also have

$$\begin{aligned}\rho(\mu^{(l)}) &= \max_{a\neq b} \left|\mathbf{E}_{Y\sim\mu^{(l)}}[\chi_{a-b}(Y)]\right| \\ &= \max_{a\neq 0^n} \left|\mathbf{E}[\chi_a(Y^{(1)}+\cdots+Y^{(l)})]\right| \\ &= \max_{a\neq 0^n} \left|\mathbf{E}[\chi_a(Y^{(1)})]\cdots\mathbf{E}[\chi_a(Y^{(1)})]\right| \\ &\leq \rho(\mu)^l.\end{aligned}$$

Thus the inequality holds. □

Claim 3.10: If $\rho(\mu) = 2^{-\Omega(n)}$ and $1/\varepsilon = 2^{o(n)}$, there exists some constant even number $l \in \mathbb{N}$ such that $2^{2n}\rho(\mu)^l \leq \varepsilon^l/8$ for every sufficiently large n.

Proof. There exists a constant $d > 0$ such that $\rho(\mu) \leq 2^{-n/d}$ for every sufficiently large n. Set l to the minimum even number strictly larger than $2d$. Then, there exists a constant $\kappa > 0$ such that $2^{2n}\rho(\mu)^l \leq 2^{-\kappa n}$ for every sufficiently large n. Since $\varepsilon = 2^{-o(n)}$, we have $2^{2n}\rho(\mu)^l \leq \varepsilon^l/8$ for every sufficiently large n. □

Lemma 3.5 holds from the above three claims. □

In what follows, we consider only μ satisfying $\rho(\mu) = 2^{-\Omega(n)}$, and we fix l to a constant even number given from Claim 3.10.

In order to prove Lemma 3.6, we first show Claims 3.11, 3.12 and 3.13.

Claim 3.11: Let α be any node at which $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds. If $\varepsilon^l \leq S_l(\alpha)$, we have $\varepsilon^l/2 \leq \bar{S}_l^f(\alpha,\varepsilon)$. Conversely, if $\varepsilon^l/2 \leq \bar{S}_l^f(\alpha,\varepsilon)$, we have $\varepsilon^l/4 \leq S_l(\alpha)$.

Proof. From Lemma 3.5, $|\bar{S}_l^f(\alpha,\varepsilon) - S_l(\alpha)| \leq |\bar{S}_l^f(\alpha,\varepsilon) - E_l(\alpha)| + |S_l(\alpha) - E_l(\alpha)| \leq \varepsilon^l/4$, and thus, the claim holds. □

We define $\mathrm{SURVIVE} := \{\alpha \in \mathrm{VISIT} \mid \varepsilon^l/2 \leq \bar{S}_l^f(\alpha,\varepsilon)\}$, $\mathrm{VISIT}_k := \mathrm{VISIT} \cap \mathbb{F}_2^k$, $\mathrm{SURVIVE}_k := \mathrm{SURVIVE} \cap \mathbb{F}_2^k$, $\mathrm{VISIT}_{\leq k} := \bigcup_{i\leq k}\mathrm{VISIT}_i$, $\mathrm{SURVIVE}_{\leq k} := \bigcup_{i\leq k}\mathrm{SURVIVE}_i$, and $A_k(\theta) := \{\alpha \in \mathbb{F}_2^k \mid \theta \leq S_l(\alpha)\}$. Then, the following claim holds.

Claim 3.12: For every $k \in [n]$, if $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds at every $\alpha \in \mathrm{VISIT}_{\leq k}$, we have $A_k(\varepsilon^l) \subseteq \mathrm{SURVIVE}_k \subseteq A_k(\varepsilon^l/4)$.

Proof. By induction on k. In the base case $k = 0$, the claim holds by Claim 3.11 if $\alpha \in \mathrm{VISIT}_{\leq k} = \{\lambda\}$. Assume that the claim holds for $k = k' - 1 < n$. Then, if $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds at every $\alpha \in \mathrm{VISIT}_{\leq k'-1}$, we have $A_{k'-1}(\varepsilon^l) \subseteq \mathrm{SURVIVE}_{k'-1} \subseteq A_{k'-1}(\varepsilon^l/4)$. Assume also that $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds at every $\alpha \in \mathrm{VISIT}_{\leq k'}$. Fix any $\beta \in A_{k'}(\varepsilon^l)$. From the monotonicity of S_l, a prefix β' of β of length $|\beta| - 1$ satisfies $\beta' \in A_{k'-1}(\varepsilon^l) \subseteq \mathrm{SURVIVE}_{k'-1}$. Then $\beta \in \{\beta'0, \beta'1\} \subseteq \{\alpha'0, \alpha'1 \mid \alpha' \in \mathrm{SURVIVE}_{k'-1}\} = \mathrm{VISIT}_{k'}$. Since it succeeds at β, we have $\beta \in \mathrm{SURVIVE}_{k'}$ from Claim 3.11. Namely, $A_{k'}(\varepsilon^l) \subseteq \mathrm{SURVIVE}_{k'}$. We next fix any $\beta \in \mathrm{SURVIVE}_k \subseteq \mathrm{VISIT}_k$. Since it succeeds at β, we have $\beta \in A_{k'}(\varepsilon^l/4)$ from Claim 3.11. □

Claim 3.13: $|A_k(\theta)| = O(1/\theta)$.

Proof. Let $Y \sim \mu^{(l)}$. Then, it follows that

$$\begin{aligned}|A_k(\theta)|\,\theta &\leq \sum_{\alpha\in A_k(\theta)} S_l(\alpha) \leq \sum_{\alpha\in\mathbb{F}_2^k} \left\langle f^{[\mu,l]}, \sum_{\alpha'\in\mathbb{F}_2^{n-k}} \chi_{\alpha\alpha'} \right\rangle_{\mu^{(l)}} \\ &= \left\langle f^{[\mu,l]}, \sum_{a\in\mathbb{F}_2^n} \chi_a \right\rangle_{\mu^{(l)}} = 2^n \left\langle f^{[\mu,l]}, \delta_n \right\rangle_{\mu^{(l)}} \\ &= 2^n \Pr[Y = 0] f^{[\mu,l]}(0) \leq 2^n \Pr[Y = 0].\end{aligned}$$

(Recall that $\delta_n(x) = 1$ if $x = 0^n$ and $\delta(x) = 0$ otherwise.) By Claim 3.9, the righthand side is bounded by $1 + \varepsilon^l/8 = O(1)$ since $2^{2n}\rho(\mu)^l \leq \varepsilon^l/8$. □

We now prove Lemma 3.6 by these claims.

Proof of Lemma 3.6. By the Höffding bounds, the probability that $\bar{S}_l^f(\alpha,\varepsilon)$ fails, namely, $|\bar{S}_l^f(\alpha,\varepsilon) - E_l(\alpha)| > \varepsilon^l/8$ is at most $q := 2\exp\left\{-\varepsilon^{2l}m/128\right\}$ independently of α. From the union bound, the probability that $\mathrm{EKM}^f(\lambda,\varepsilon)$ does not succeed at some $\alpha \in \mathrm{VISIT}_k$ is at most $|\mathrm{VISIT}_k| \cdot q$. Assume that $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds at every $\alpha \in \mathrm{VISIT}_{\leq k-1}$. By Claim 3.12, we have $\mathrm{SURVIVE}_{k-1} \subseteq A_{k-1}(\varepsilon^l/4)$. Since $\mathrm{VISIT}_k = \{\alpha 0, \alpha 1 \mid \alpha \in \mathrm{SURVIVE}_{k-1}\}$, $|\mathrm{VISIT}_k| = 2|\mathrm{SURVIVE}_{k-1}| \leq 2|A_{k-1}(\varepsilon^l/4)| = O(1/\varepsilon^l)$ by Claim 3.13. Therefore, it holds that

$$\begin{aligned}\Pr\left[\bigcup_{i=1}^n \mathrm{SUCCESS}_i\right] &= \Pr[\mathrm{SUCCESS}_0] \\ &\quad\times \prod_{i=1}^n \Pr[\mathrm{SUCCESS}_i \mid \mathrm{SUCCESS}_{i-1}] \\ &\geq (1 - O(1/\varepsilon^l)q)^{n+1} \geq 1 - O(n/\varepsilon^l)q,\end{aligned}$$

where $\mathrm{SUCCESS}_i$ is the event that $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds at every $\alpha \in \mathrm{VISIT}_i$. The righthand side is at least $2/3$ by setting $m = c\varepsilon^{-2l}\ln(n/\varepsilon)$ for some appropriate constant c. □

We finally prove Lemma 3.7.

Proof of Lemma 3.7. Assume that $\mathrm{EKM}^f(\lambda,\varepsilon)$ succeeds

at every $\alpha \in \mathrm{VISIT}$. (a) For every $x \in \mathbb{F}_2^n$, if $\varepsilon \le \langle f, \chi_x \rangle_\mu$, we have $x \in A_n(\varepsilon^l)$. Also, by Claim 3.12, $A_n(\varepsilon^l) \subseteq \mathrm{SURVIVE}_n = L$. Thus, $x \in L$. (b) By Claim 3.12, we have $L = \mathrm{SURVIVE}_n \subseteq A_n(\varepsilon^l/4)$. Also, $|A_n(\varepsilon^l/4)| = O(1/\varepsilon^l)$ from Claim 3.13. (c) $\mathrm{VISIT}_k = \{\alpha 0, \alpha 1 \mid \alpha \in \mathrm{SURVIVE}_{k-1}\}$ for every $k \in [n]$ by definition. By Claims 3.13 and 3.12, $|\mathrm{VISIT}_k| = 2|\mathrm{SURVIVE}_{k-1}| \le 2|A_{k-1}(\varepsilon^l/4)| = O(1/\varepsilon^l)$. Since $|\mathrm{VISIT}| \le (n+1)O(1/\varepsilon^l) = O(n/\varepsilon^l)$, the number of invocations of $\bar{S}_l^f$ is at most $O(n/\varepsilon^l)$. Since each invocation of $\bar{S}_l^f$ makes $ml = O(\varepsilon^{-2l} \ln(n/\varepsilon))$ queries, the total number of queries is at most $O(n\varepsilon^{-3l} \ln(n/\varepsilon))$. □

Remark 3.14: It is easy from the analysis of the query complexity to see that the running time of the extended KM algorithm for Problem 3.2 is $T(\mu, l) \cdot O(n\varepsilon^{-3l} \ln(n/\varepsilon))$, where $T(\mu, l)$ denotes the time complexity to obtain a sample from $\mu\,|^l_{\le k}$. In the case of list-decoding for the sparse random linear codes, T is bounded by $O(N^l) \cdot n^{O(1)} = 2^{O(n)}$ using the following trivial algorithm: Generate a list of all l-tuples $(y^{(1)}, \ldots, y^{(l)})$ from $\{g_1, \ldots, g_N\}$ and pick up a uniform sample satisfying $y^{(1)}_{>k} + \cdots + y^{(l)}_{>k} = 0$. Taking into accout the time complexity for the oracle simulation in Lemma 3.3, the total time complexity for the list-decoding is at most $\varepsilon^{-O(1)} \cdot 2^{O(n)}$, as given in Theorem 3.1.

4. Concluding Remarks

Finally, we conclude this paper with some remarks on sampling problems of Kopparty and Saraf and of ours.

Let μ be any distribution over $\mathbb{F}_2^n$ such that $\rho(\mu) \le 2^{-n/d}$ for some constant $d > 1$. If the following sampling problem for μ can be solved efficiently, Kopparty amd Saraf's result provides an efficient local-list decoding algorithm for sparse random linear codes, as mentioned in [11]. We suppose l be some sufficiently large constant.

Problem 4.1 (Kopparty and Saraf's back-sampling problem) Given $x \in \mathbb{F}_2^n$, sample $y^{(1)}, \cdots, y^{(l)}$ from μ under the condition $\sum y^{(i)} = x$.

On the other hand, our algorithm works in polynomial time if we can perform the sampling from the distribution $\mu|^l_{\le k}$ in polynomial time in n.

Problem 4.2 (Our sampling problem) Given $k \in [n]$, sample $y^{(1)}, \cdots, y^{(l)}$ from μ under the condition that $\sum (y^{(i)})_{>k} = 0^{n-k}$.

Inspecting these two sample problems, it seems to be easier to solve the second problem since the condition of the distribution in the first problem must be controlled by an arbitrarily given x, but that in the second problem is partially fixed only by 0^{n-k}.

Actually, an approximate version of the second problem is reduced to the first problem, and hence we can obtain an efficient list decoding algorithm for sparse random linear codes from our Fourier-analytic approach if we have an efficient algorithm that solves the first problem approximately.

Theorem 4.3: Assume that there exists a randomized algorithm A_1 that outputs samples under the distribution D_1 of Problem 4.1 in polynomial time in n. Then, there exists a randomized algorithm A_2 that samples under the distribution D_2' which is statistically $2^{-\Omega(n)}$-close to the distribution D_2 of Problem 4.2, i.e., $\frac{1}{2}\sum_{x \in \mathbb{F}_2^n} |\Pr_{X \sim D_2'}[X = x] - \Pr_{X \sim D_2}[X = x]| = 2^{-\Omega(n)}$.

Proof. We explicitly describe the sampling procedure A_2 by using A_1: Sample $x \in \mathbb{F}_2^k$ uniformly at random, and then, output the samples from A_1 on input $x0^{n-k}$.

We now observe why this procedure works well. If we could sample $x0^{n-k}$ from the distribution $\mu^{(l)}$ under the condition that the lower $n-k$ bits are all zeros, we could exactly solve Problem 4.2 by using A_1 on input $x0^{n-k}$. However, it does not look to be easy to directly do the sampling yet. Instead of the sampling, we just sample x uniformly at random as done in A_2. Actually, $\mu^{(l)}$ is statistically close to U_n by taking a sufficiently large l from Lemma 4.1 in [11].

Therefore, the statistical distance Δ between $\mu^{(l)}$ under the condition that the lower $n-k$ bits are all zeros and U_n under the same condition is bounded from above by $2^{-\Omega(n)}$ as follows.

Let $X \sim \mu^{(l)}$ and $Y \sim U_n$. The statistical distance Δ between the two distribution is

$$\begin{aligned}\Delta &= \frac{1}{2}\sum_{x \in \mathbb{F}_2^k} |\Pr[X = x\mathbf{0} | X_{>k} = \mathbf{0}] - \Pr[Y = x\mathbf{0} | Y_{>k} = \mathbf{0}]| \\ &= \frac{1}{2}\sum_{x \in \mathbb{F}_2^k} |\Pr[X = x\mathbf{0}] / \Pr[X_{>k} = \mathbf{0}] - 2^{-k}]|.\end{aligned}$$

Let $\Pr[X = x\mathbf{0}] := 2^{-n} + d_x$ and $\Pr[X_{>k} = \mathbf{0}] := 2^{-(n-k)} + e$. Then,

$$\Delta = \frac{1}{2}\sum_{x \in \mathbb{F}_2^k} \left| \frac{d_x - 2^{-k}e}{2^{-(n-k)} + e} \right| \le \frac{1}{2} \frac{\sum_{x \in \mathbb{F}_2^k} |d_x| + |e|}{2^{-(n-k)} - |e|}$$

by the triangle inequality. Then, we can show Δ is bounded by $2^{-\Omega(n)}$ by taking a sufficiently large l from Lemma 4.1 in [11] since $\sum_{x \in \mathbb{F}_2^k} |d_x|$ and $|e|$ are bounded by the statistical distance between $\mu^{(l)}$ and U_n.

Thus, the uniform sampling x works well alternatively, and the sampling procedure A_2 approximately solves Problem 4.2. □

This reduction suggests that the first problem is as at least hard as the second problem. Therefore, it would be easier to improve our algorithm by adding further algorithmic tricks towards efficient list-decoding of random linear codes.

As briefly mentioned in Sect. 1, the Fourier-analytic approach for the Hadamard code shown by the KM algorithm provided many applications by introducing a new mathematical viewpoint. Similarly, our Fourier-analytic approach

to random linear codes would be suggestive of further applications.

References

[1] A. Akavia, S. Goldwasser, and S. Safra, "Proving hard-core predicates using list decoding," Proc. 44th IEEE Symposium on Foundations of Computer Science, pp.146–157, 2003.
[2] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors," Proc. 45th Annual ACM Symposium on Theory of Computing, pp.575–584, 2013.
[3] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg, "On the inherent intractability of certain coding problems," IEEE Trans. Inf. Theory, vol.24, pp.384–386, 1978.
[4] P. Elias, "List decoding for noisy channels," 1957.
[5] V. Guruswami, J. Håstad, and S. Kopparty, "On the list-decodability of random linear codes," IEEE Trans. Inf. Theory, vol.57, no.2, pp.718–725, 2011.
[6] O. Goldreich and L.A. Levin, "A hard-core predicate for all one-way functions," STOC '89, pp.25–32, 1989.
[7] V. Guruswami, "List decoding of error-correcting codes," LNCS 3238, 2002. Winning Thesis of the 2002 ACM Doctoral Dissertation Competition.
[8] V. Guruswami, "List decoding of binary codes – A brief survey of some recent results," 2nd International Workshop on Coding and Cryptography, pp.97–106, 2009.
[9] E. Kushilevitz and Y. Mansour, "Learning decision trees using the fourier spectrum," SIAM J. Comput., vol.22, no.6, pp.1331–1348, 1993.
[10] T. Kaufman and M. Sudan, "Sparse random linear codes are locally decodable and testable," FOCS '07, pp.590–600, 2007.
[11] S. Kopparty and S. Saraf, "Local list-decoding and testing of random linear codes from high error," SIAM J. Comput., vol.42, no.3, pp.1302–1326, 2013.
[12] O. Regev, "Improved inapproximability of lattice and coding problems with preprocessing," IEEE Trans. Inf. Theory, vol.50, no.9, pp.2031–2037, 2004.
[13] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," J. ACM, vol.56, no.6, article 34, pp.1–40, 2009.
[14] O. Regev, "The learning with errors problem," Proc. 25th Annual IEEE Conference on Computational Complexity, pp.191–204, 2010.
[15] M. Sudan, "List decoding: Algorithms and applications," in Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics, pp.25–41, Springer Berlin Heidelberg, 2000.
[16] S. Vadhan, "The unified theory of pseudorandomness," SIGACT News, vol.38, no.3, pp.39–54, 2007.
[17] J.M. Wozencraft, "List decoding," Quarterly Progress Report, vol.48, pp.90–95, 1958. Research Laboratory of Electronics, MIT.

Akinori Kawachi is an assistant professor of Department of Mathematical and Computing Sciences, Tokyo Institute of Technology. Received B.E., M.Info., and Ph.D. degrees from Kyoto University in 2000, 2002, and 2004, respectively. His research interests are computational complexity, quantum computing, and foundations of cryptography.

Ikko Yamane is a master student of Department of Computer Science, Tokyo Institute of Technology. Received a B.E. degree from Tokyo Institute of Technology in 2013. His research interests are randomized algorithms and machine learning.