Electronic Colloquium on Computational Complexity, Report No. 124 (2011)

Algorithms for the Coin Weighing Problems with the Presence of Noise

Nader H. Bshouty
Technion, Israel
bshouty@cs.technion.ac.il

Hanna Mazzawi
Technion, Israel
hanna@cs.technion.ac.il

Abstract

The coin weighing problem is the following: Given n coins for which m of them are counterfeit with the same weight. The problem is to detect the counterfeit coins with minimal number of weighings. This problem has many applications in compressed sensing, multiple access adder channels, etc.

This problem was solved when m is unknown. An old optimal non-adaptive polynomial time algorithm of Lindstrom does $O(n/\log n)$ weighings can detect the counterfeit coins. In this paper we give a non-adaptive polynomial time algorithm that does $O(n/\log n)$ weighings and detect the counterfeit coins even if n^c of the answers of the weighings received are incorrect or missing for any constant $c < 1$.

When m is known we give an adaptive polynomial time algorithm that does $O((m \log n)/\log m)$ weighings and detect the counterfeit coins even if m^c of the answers of the weighings received are incorrect or missing for any constant $c < 1$.

We then study the problem when m is known and the counterfeit coins have different weights. We show that there is an optimal non-adaptive algorithm for detecting the counterfeit coins with $O((m \log n)/\log m)$ weighing even if 11 percent of the answers are incorrect or missing.

A simple information theoretical argument shows that all the above algorithm are optimal.

1 Introduction

The coin weighing problems, also known as the problems of reconstructing hidden vectors, are the following. Let v be a size n hidden vector. Suppose that we are allowed to ask queries of the form

$$Q(x) = v^T x,$$

where $x \in \{0,1\}^n$. Suppose that the some of the answers received are incorrect (we call them *errors*). Also assume that some of the answers received are equal to "?", where "?" indicates that the answer to the query is unknown (we call these answers *erasures*). Our goal is to exactly reconstruct the hidden vector with minimal number of queries.

We distinguish between two type of algorithms for solving our problems. Non-adaptive algorithms are algorithms that ask all queries in advance, before receiving any answer. On the other hand, adaptive algorithms are algorithms that take into account the outcome of previous queries.

We denote by $\text{coin}_B(n)$ the coin weighing problem where the hidden vector v is a (0,1)-vector of size n. We denote by $\text{coin}_B(n,m)$ the coin weighing problem where the hidden vector v is a (0,1)-vector of size n and Hamming weight that is bounded by m. Finally, we denote by $\text{coin}_{\mathbb{R}}(n,m)$ the coin weighing problem where the hidden vector is in $\mathbb{R}^n$ and has at most m non-zero entries.

The coin weighing problems were heavily studied in the noiseless case, that is, the case where all the answers are available and correct. For the $\text{coin}_B(0,1)$ weighing problem in the noiseless case, the informa-

ISSN 1433-8092

tion theoretic lower bound for the query complexity of reconstructing the hidden vector is

$$O\left(\frac{n}{\log n}\right).$$

This problem was studied by Cantor in [8], by Soderberg and Shapiro in [34] by Erdös and Rényi in [18], by Lindström in [26, 27, 28, 29] and by Cantor and Mills in [10]. Lindström [26] and independently Cantor and Mills [10] showed a non-adaptive polynomial time algorithm for the problem with query complexity that matches the lower bound. Simplifications appear in [29, 1, 5].

For the $\text{coin}_B(n, m)$ weighing problem in the noiseless case, the information theoretic lower bound for the query complexity is

$$O\left(\frac{m \log \frac{n}{m}}{\log m}\right).$$

In [19] Grebinski and Kucherov showed a non-constructive non-adaptive algorithm for the problem that matches this lower bound for all m. In [5], Bshouty gave a polynomial time adaptive algorithm for the problem with query complexity that matches the information theoretic lower bound.

Finally, for the $\text{coin}_{\mathbb{R}}(n, m)$ in the noiseless case, the information theoretic lower bound for the query complexity is

$$O\left(\frac{m \log n}{\log m}\right).$$

For this problem, in [14] Choi and Han Kim showed a tight non-constructive non-adaptive algorithm for reconstructing a hidden vector with non-zero entries that are real numbers that are bounded by n^{-a} and n^b for any constant a and b. In [6] Bshouty and Mazzawi, removed the condition on the magnitudes of the non-zero numbers by giving a tight non-constructive non-adaptive algorithm for the general problem. They also give an almost tight polynomial time adaptive algorithm in [7].

The coin weighing problems are combinatorial search problems that are motivated by real-life problems. As such, it is important to study them in the presence of noise.

For the $\text{coin}_B(n)$ weighing problem with noise, we show the following.

Theorem 1. *Let $v \in \{0,1\}^n$ be a hidden vector. There exists a non-adaptive reconstructing algorithm that reconstructs v using*

$$O\left(\frac{n}{\epsilon \log n}\right)$$

queries.

This algorithm reconstructs v correctly in $O(n^{3/2})$ time when the number of errors e_1 and the number of erasures e_2 are bounded by $c\left(\frac{n}{\epsilon \log n}\right)^{\frac{1-\epsilon}{4}}$, for any constant $c < 1/2$.

The proof of the above theorem relies on properties of the fourier spectrum of function we prove. The algorithm above, although fast, it cannot handle large amount of errors. In the following theorem, we show that one can compromise the running time to handle more errors and erasures. We show a technique that uses generating matrices of linear codes to non-adaptively solve multiple vector reconstructing problems altogether while gaining resistance to noise in the process.

Theorem 2. *Let $v \in \{0,1\}^n$ be a hidden vector. There exists a non-adaptive polynomial time reconstructing algorithm that reconstructs v using*

$$k = O\left(\frac{n}{\epsilon \log n}\right)$$

queries. This algorithm reconstructs v correctly when the number of incorrect answers e_1 and the number of unknown answers e_2 are bounded by $\Omega(n^{1-\epsilon})$. Moreover, denote by $(q_1, q_2, \ldots, q_k)$ the vector of answers, that is, q_i equals the answer to the ith query, for $i \in [k]$. If the errors/erasures are consecutive, then, the algorithm works correctly even if their number is $\Omega(\frac{n}{\epsilon \log n})$.

Using the same mentioned technique, we obtain the following result for the $\text{coin}_B(n, m)$ weighing problem.

Theorem 3. *There is a polynomial time algorithm for reconstructing any hidden vector $v \in \{0,1\}^n$ that uses*

$$O\left(\frac{m \log n}{\epsilon \log m}\right)$$

queries, where m is the number of non-zeros in v. This algorithm reconstructs v correctly even in the presence of $O(m^{1-\epsilon})$ errors or erasures.

Finally, for the $\text{coin}_{\mathbb{R}}(n,m)$ weighing problem we show the following.

Theorem 4. *Let $p = \omega(1)$ be a prime number and $\epsilon \leq 0.22$. There exists a matrix $M \in \{0,1\}^{k \times n}$, where*

$$k = O\left(m + \frac{m \log \frac{n}{m}}{\log \min(m,p)}\right),$$

such that: For every submatrix M^* formed by selecting arbitrary $k(1-\epsilon)$ rows of M, every m columns in M^* are linearly independent over $\mathbb{Z}_p$.

This theorem yields the existence of a non-adaptive algorithm for the $\text{coin}_{\mathbb{R}}(n,m)$ weighing problem with $O(m \log n / \log m)$ query complexity. That is, query complexity that matches the information theoretic lower bound for the noiseless case. This algorithm works correctly even if 11 percent of the answers received are incorrect or missing.

1.1 Applications

In this subsection we introduce one application for the coin weighing problem with noise. We introduce the signature coding problem for the multiple access adder channels [26, 10, 11, 23, 17, 15, 25, 3].

Consider n stations or users that transmit information using a common channel. Each user i has a component code $C_i = \{0^k, x_i\} \subset \{0,1\}^k$. The codes C_i for $i \in [n]$ are of the same length k. Each user i wants to send one of the words $y_i \in C_i$. Assume that codewords sent through the channel by the users are bit and block synchronized. The codeword go through an adder that sends

$$Y = \sum_{i=1}^{n} y_i.$$

through a noisy channel. We denote by $\tilde{Y}$ the output of the channel. The goal is to recover all the messages of the users using $\tilde{Y}$. See Figure 1.

There are two models for this problem, the *permanently active* model where all stations are active all the time and the *partially active* model where at most m stations are active in each transmission (the inactive stations turn off their transmitter, an action that is equivalent to sending the zero codeword 0^k).

A non-adaptive algorithm for the $\text{coin}_B(n)$ weighing problems with the presence of noise yields signature codes for noisy channels in the permanently active model. The existence of such codes was studied in the literature [11, 17, 35, 12, 15]. In [11], Chang and Weldon first showed a technique to reconstruct signature codes for noisy channels. Using this technique, one can build a codes of length $O(\frac{n}{\epsilon \log n})$ for n users, where a messages can be correctly recovered if the error vector, $\tilde{Y} - Y$, has L_1 norm that is smaller that $O(n^{1-\epsilon})$ (note that unlike what is presented in Figure 1, in the literature the received codeword does not include erasures. Therefore, when talking about previous related work, always assume that the received word $\tilde{Y}$ does not contain erasures, that is, the "?" symbol). In other words, the users' messages can be recovered from $\tilde{Y}$ if

$$\sum_i |\tilde{Y}_i - Y_i| = O(n^{1-\epsilon}).$$

In [35], Wilson showed a similar technique. As in the previous technique, using Wilson's technique, one can reconstruct codes of length $O(\frac{n}{\epsilon \log n})$ for n users, where a message can be correctly recovered if the error vector, $\tilde{Y} - Y$, has L_1 norm that is smaller than $O(n^{1-\epsilon})$. Moreover, the message can be recovered correctly even if we have $O(\frac{n^\epsilon}{\epsilon \log n})$ consecutive errors of magnitude $O(n^{1-\epsilon})$. That is, if the error vector $\tilde{Y} - Y$ has only $O(\frac{n^\epsilon}{\epsilon \log n})$ non-zero entries, where these non-zero entries are consecutive and each entry is bounded by $O(n^{1-\epsilon})$. Additional constructions also appear in [13, 12]. Finally, in [15], show a code for n users of length n, where the messages can be recovered if the error vector has L_1 norm that is smaller than $\lfloor (n/2-1)/2 \rfloor$.

To our knowledge, all codes in the literature for the multiple access decode the received message correctly

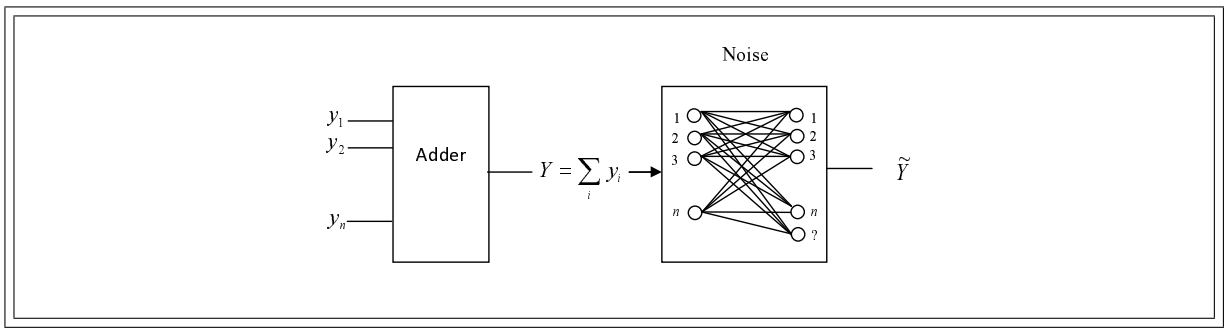

Figure 1: Multiple access adder channel with noise

if the error vector, $\tilde{Y} - Y$, has a bounded L_1 norm. Our theorem, Theorem 2, yields signature codes for the permanently active model of length

$$O\left(\frac{n}{\epsilon \log n}\right),$$

where the receiver is able to decode all the users' messages correctly with the presence of $O(n^{1-\epsilon})$ errors and erasures. That is, if the error vector has *Hamming weight* (rather than the L_1 norm) that is smaller than $O(n^{1-\epsilon})$. Moreover, if the errors are consecutive, then the message can be recovered even when we have $O(\frac{n}{\epsilon \log n})$ errors.

Similarly, Theorem 4 yields signature codes for the partially active model of length $k = O(m \log n / \log m)$. The messages are recovered correctly if the Hamming weight of the error vector is bounded by $0.11k$.

2 Preliminaries

In this section, we introduce some notation, the well known Fourier basis and some known results we use in this paper.

We denote by $\mathbb{R}$ the set of real numbers. We denote by $\mathbb{Z}_p$ the field of integers modulo a prime p. We also denote by $\mathbb{N}_0$ the set of non-negative integers. For a constant c, we denote by $[c]$ the set $[1, 2, \ldots, c]$. We denote by $[c]_0$ the set $[c] \cup \{0\}$.

For a matrix M, we denote by $r(M)$ the rank of M.

An $[n, k, d]$ *linear code* $\mathcal{C}$ over a field $\mathbb{F}$ is a linear code of length n, dimension k and minimal distance d.

We now present the well known Fourier representation of functions. Let $x = (x_1, x_2, \ldots, x_\ell)$ be variables. Define the following basis

$$B = \left\{ \chi_a(x) \triangleq \prod_{i:a_i=1} x_i \;\middle|\; a \in \{-1,1\}^\ell \right\},$$

where $\chi_a : \{-1,1\}^\ell \to \{-1,1\}$. It is known that every function $f : \{-1,1\}^\ell \to \mathbb{R}$ has a unique representation of the form

$$f(x) = \sum_{a \in \{-1,1\}^\ell} \hat{f}(a)\chi_a(x),$$

where for every $a \in \{-1,1\}^\ell$, $\hat{f}(a)$ is a real number and is called the Fourier coefficient of χ_a in f. For a vector $a \in \{-1,1\}^\ell$ we will abuse the standard notation and denote by $||a||$ the number of ones in a (the size of the Fourier coefficient).

In the paper, we also make use of a family of matrices defined in [5] that optimally solve the $\mathrm{coin}_B(n)$ weighing problem in the noiseless case. We sketch their construction for convenience in Appendix A.

Finally, we make use of the following lemmas.

Lemma 5. *[6] Let $M \in \{0,1\}^{k \times m}$ be a matrix of rank $r = r(M) < m$. For a uniformly randomly chosen row vector $y \in \{0,1\}^m$ the rank of the matrix*

$$M' = \begin{pmatrix} M \\ y \end{pmatrix}$$

over $\mathbb{Z}_p$ is r with probability at most $1/2$.

Lemma 6. *[6] Let $M \in \{0,1\}^{k\times m}$ be a matrix of rank $r(M) = r < m$. Suppose that every s columns of M are linearly independent. Then, for a uniformly randomly chosen row vector $y \in \{0,1\}^m$ the rank of the matrix*

$$M' = \begin{pmatrix} M \\ y \end{pmatrix}$$

over $\mathbb{Z}_p$ is r with probability at most $\max\left(\frac{1}{s^\beta}, \frac{1}{p^{1/2}}\right)$. Here $\beta = \frac{1}{2+\log 3} = 0.278943\cdots$.

3 Non-constructive Algorithm

In this section, we prove Theorem 4.

Theorem 4. *Let $p = \omega(1)$ be a prime number and $\epsilon \le 0.22$. There exists a matrix $M \in \{0,1\}^{k\times n}$, where*

$$k = O\left(m + \frac{m\log\frac{n}{m}}{\log\min(m,p)}\right),$$

such that: For every submatrix M^* formed by selecting arbitrary $k(1-\epsilon)$ rows of M, every m columns in M^* are linearly independent over $\mathbb{Z}_p$.

Proof. We may assume that $m = \omega(1)$ since otherwise if $m = O(1)$ then $k = O(m\log n)$ and the result follows by a simple use of the probabilistic method. Take a randomly chosen matrix $M \in \{0,1\}^{k\times n}$, where

$$k = c_1\left(m + \frac{m\log\frac{n}{m}}{\log\min(m,p)}\right),$$

and c_1 is a constant that will be determined later. Look at any matrix M^* formed by selecting arbitrary $k^* = (1-\epsilon)k$ rows of M. We now show that every m columns in M^* are linearly independent with probability that is strictly greater than

$$1 - \frac{1}{\binom{k}{k^*}}$$

Once we prove the above statement, by union bound the result immediately follows.

Take any $t = m/\log^2 m$ columns, $M^*_{i_1}, M^*_{i_2}, \ldots, M^*_{i_t}$, of M^*. Denote by M' the matrix

$$M' = \left[M^*_{i_1} | M^*_{i_2} | \ldots | M^*_{i_t}\right].$$

Denote by $M'^{[j]}$ be the first j rows of M'. Consider the random variable $X_j \in \{0,1\}$ where $X_j = 1$ if and only if $r(M'^{[j-1]}) = t$ or the jth row $M'^{(j)}$ increases the rank of $M'^{[j-1]}$, i.e., $r(M'^{[j]}) = r(M'^{[j-1]}) + 1$. Let $k_1 = c_2 k^*$, where $c_2 = 1 - o(1)$ is a positive constant. By Lemma 5,

$$\mathbf{Pr}[X_j = 0 | X_1, X_2, \ldots, X_{j-1}] \le 1/2.$$

The probability that $M'^{[k_1]}$ is of rank smaller than t is

$$\begin{aligned}
\mathbf{Pr}[r(M'^{[k_1]}) < t] &= \mathbf{Pr}[X_1 + \cdots + X_{k_1} \le t-1] \\
&= \sum_{\substack{\xi_1+\cdots+\xi_{k_1}\le t-1,\\ \xi_j\in\{0,1\}}} \mathbf{Pr}[X_1 = \xi_1, \ldots, X_{k_1} = \xi_{k_1}] \\
&\le \frac{\sum_{i=0}^{t-1}\binom{k_1}{i}}{2^{k_1-t+1}} \le t2^{t-1}\frac{\binom{k_1}{t}}{2^{k_1}} \qquad (1) \\
&< \frac{t2^{t-1}\left(\frac{k_1 e}{t}\right)^t}{2^{k_1}} < \frac{n^t}{2^{k_1}} = \frac{2^{\frac{m\log n}{\log^2 m}}}{2^{k_1}} \\
&= \frac{1}{2^{(1-o(1))k_1}}. \qquad (2)
\end{aligned}$$

Equation (1) follows since $t = o(k_1)$. Let $\mathcal{A}$ denote the event where there exists $m/\log^2 m$ columns in $M^{*[k_1]}$ that are linearly dependent. Using (2) and the union bound, we have

$$\mathbf{Pr}[\mathcal{A}] \le \frac{\binom{n}{t}}{2^{(1-o(1))k_1}} = \frac{1}{2^{(1-o(1))k_1}}.$$

Note that for $\epsilon < 0.22$ and $c_2 = 1 - o(1)$, we have that

$$\begin{aligned}
\mathbf{Pr}[\mathcal{A}] &\le \frac{\binom{n}{t}}{2^{(1-o(1))k_1}} = \frac{1}{2^{(1-o(1))k_1}} \\
&= \frac{1}{2^{(1-o(1))(1-\epsilon)k}} < \frac{1}{2\cdot 2^{H(\epsilon)k}} < \frac{1}{2\binom{k}{k^*}}. \qquad (3)
\end{aligned}$$

Where $H(\epsilon)$ is the binary entropy of ϵ, that is,

$$H(\epsilon) = -\epsilon\log\epsilon - (1-\epsilon)\log(1-\epsilon).$$

We now analyze the probability that there exists m columns in M^* that are linearly dependent. We denote this event by $\mathcal{B}$.

$$\begin{aligned}
\mathbf{Pr}[\mathcal{B}] &= \mathbf{Pr}[\mathcal{B}|\mathcal{A}]\mathbf{Pr}[\mathcal{A}] + \mathbf{Pr}[\mathcal{B}|\neg\mathcal{A}](1 - \mathbf{Pr}[\mathcal{A}]) \\
&\le \mathbf{Pr}[\mathcal{A}] + \mathbf{Pr}[\mathcal{B}|\neg\mathcal{A}]. \qquad (4)
\end{aligned}$$

Given that every $m/\log^2 m$ columns in $M^{*[k_1]}$ are linearly independent, that is, given $\neg\mathcal{A}$, look at any matrix M'' formed by selecting arbitrary m columns of M^*. That is,

$$M'' = \left[M^*_{j_1}|M^*_{j_2}|\dots|M^*_{j_m}\right].$$

Denote by $M''^{[j]}$ be the first j rows of M''. For $j \in [k^*] \setminus [k_1]$, consider the random variable $Y_j \in \{0,1\}$ that is equal to one if and only if $r(M''^{[j-1]}) = m$ or the row jth row, $M''^{(j)}$, increases the rank of $M''^{[j-1]}$, that is, $r(M''^{[j]}) = r(M''^{[j-1]}) + 1$. Let $q = min(t^\beta, p^{1/2})$, where $\beta = 0.2789$ as defined in Lemma 6. Using Lemma 6, the probability that M'' is of rank smaller than m given $\neg\mathcal{A}$ is,

$$\begin{aligned}
&\mathbf{Pr}[r(M'') < m|\neg\mathcal{A}] \\
&\quad\le \mathbf{Pr}[Y_{k_1} + Y_{k_1+1} + \dots + Y_{k^*} \le m-1|\neg\mathcal{A}] \\
&\quad\le \frac{\sum_{i=0}^{m-1}\binom{k^*-k_1}{i}}{q^{k^*-k_1-m+1}} < \frac{2^{k^*-k_1}}{q^{k^*-k_1-m+1}} \qquad (5)\\
&\quad= \frac{1}{q^{(k^*-k_1)(1-o(1))-m+1}} \qquad (6)\\
&\quad= \frac{1}{q^{(1-\epsilon)(1-c_2)(1-o(1))k-m+1}}.
\end{aligned}$$

Again, in (5) we use Lemma 6. In (6) we use that $m = \omega(1)$ and $p = \omega(1)$. Therefore, the probability that there exists m columns in M^* that are linearly independent given that every $m/\log^2 m$ columns are linearly independent in $M^{*[k_1]}$ is

$$\mathbf{Pr}[\mathcal{B}|\neg\mathcal{A}] \le \frac{\binom{n}{m}}{q^{(1-\epsilon)(1-c_2)(1-o(1))k-m+1}}.$$

Since $k = c_1\left(m + \frac{m\log\frac{n}{m}}{\log\min(m,p)}\right)$ and $\binom{n}{m} \le 2^{m\log\frac{ne}{m}}$, then, for large enough constant c_1, we have that

$$\mathbf{Pr}[\mathcal{B}|\neg\mathcal{A}] \le \frac{1}{q^{O\left(m+\frac{m\log\frac{n}{m}}{\log\min(m,p)}\right)}} < \frac{1}{2^k} < \frac{1}{2\binom{k}{k^*}} \qquad (7)$$

Finally, from (3), (4) and (7), we get that the probability that there exists m columns that are linearly independent in M^* is

$$\begin{aligned}
\mathbf{Pr}[\mathcal{B}] &\le \mathbf{Pr}[\mathcal{A}] + \mathbf{Pr}[\mathcal{B}|\neg\mathcal{A}] \\
&< \frac{1}{2\binom{k}{k^*}} + \frac{1}{2\binom{k}{k^*}} = \frac{1}{\binom{k}{k^*}} \qquad (8)
\end{aligned}$$

Now, using union bound, the main result follows immediately. □

Corollary 1. *Let $\epsilon < 0.22$ be a constant, there exists a non-adaptive algorithm for reconstructing a hidden vector $v \in \mathbb{R}^n$, $wt(v) \le m$, using*

$$k = O\left(\frac{m\log n}{\log m}\right)$$

queries. The algorithm reconstruct the hidden vector correctly when the number of errors e_1 and the number of erasures e_2 is bounded by $\frac{\epsilon}{2}k$, that is, $e_1 + e_2 < \frac{\epsilon}{2}k$.

4 Polynomial Time Algorithms

In this section, we present polynomial time algorithms for the $\text{coin}_B(n)$ and the $\text{coin}_B(n,m)$ weighing problems. We start with the $\text{coin}_B(n)$ weighing problem.

4.1 Fourier Based Algorithm

In this subsection, we show a fast algorithm for reconstructing a $(0,1)$ vector using queries. We show the following,

Theorem 1. *Let $v \in \{0,1\}^n$ be a hidden vector. There exists a non-adaptive reconstructing algorithm that reconstructs v using*

$$O\left(\frac{n}{\epsilon\log n}\right)$$

queries.

This algorithm reconstructs v correctly in $O(n^{3/2})$ time when the number of errors e_1 and the number of erasures e_2 are bounded by $\Omega\left(\left(\frac{n}{\epsilon\log n}\right)^{\frac{1-\epsilon}{4}}\right)$.

To prove the above theorem, we start by showing some properties of functions in $\mathbb{R}^{\{-1,1\}^n}$.

Lemma 7. *Let $f \in \mathbb{R}^{\{-1,1\}^n}$. If f has s non-zero values, then, there is $q \in \{-1,1\}^n$ where $||q|| \le \lceil\log s\rceil$ and the fourier coefficient $\hat{f}(q)$ is non-zero. Recall that $||q||$ denotes the number of ones in q.*

Proof. Let

$$\lambda_1 = f(a_1),\ \lambda_2 = f(a_2),\ \ldots, \lambda_s = f(a_s),$$

be the non-zero values of f. Suppose of the contrary that for all $q \in \{-1,1\}^n$, where $||q|| \leq \lceil \log s \rceil$, we have $\hat{f}(q) = 0$. That is,

$$\begin{aligned} \hat{f}(q) &= E[f \cdot \chi_q] \qquad (9) \\ &= \lambda_1 \chi_q(a_1) + \lambda_2 \chi_q(a_2) + \ldots + \lambda_s \chi_q(a_s) = 0. \end{aligned}$$

Since we have s vectors, $a_1, a_2, \ldots, a_s$, then, there is a vector a_ℓ where $\ell \in [s]$ that has $t = \lceil \log s \rceil$ entries $j_1, j_2, \ldots, j_t$ that uniquely identify it. That is,

$$(a_{\ell j_1}, a_{\ell j_2}, \ldots, a_{\ell j_t}) \neq (a_{w j_1}, a_{w j_2}, \ldots, a_{w j_t}),$$

for every $w \in [s]$ such that $w \neq \ell$.

Now, let h be a function in $\mathbb{R}^{\{-1,1\}^t}$, where

$$h(b) = \begin{cases} 1 & b = (a_{\ell j_1}, a_{\ell j_2}, \ldots, a_{\ell j_t}) \\ 0 & \text{otherwise.} \end{cases}$$

For a vector $a \in \{-1,1\}^n$, denote by $a|_{(j_1,j_2,\ldots,j_t)}$, the t vector where the ith entry equals a_{j_i}. Denote by $a^{(j_1,j_2,\ldots,j_t)}$ the size n vector where the ith entry equals a_i if $i \in \{j_1, j_2, \ldots, j_t\}$ and -1 otherwise. From (9), we have that for every $b \in \{-1,1\}^n$,

$$\sum_{i=1}^{s} \lambda_i \chi_{b^{(j_1,j_2,\ldots,j_t)}}(a_i)$$

Let $b_1, b_2, \ldots, b_{2^t}$ be all the vectors in $\{b^{(j_1,j_2,\ldots,j_t)} | b \in \{-1,1\}^n\}$. We have that

$$\begin{aligned} 0 &= \sum_{j=1}^{2^t} \hat{h}(b_j|_{(j_1,j_2,\ldots,j_t)}) \sum_{i=1}^{s} \lambda_i \chi_{b_j}(a_i) \\ &= \sum_{i=1}^{s} \lambda_i \sum_{j=1}^{2^t} \hat{h}(b_j|_{(j_1,j_2,\ldots,j_t)}) \chi_{b_j}(a_i) \\ &= \sum_{i=1}^{s} \lambda_i h(a_i|_{(j_1,j_2,\ldots,j_t)}) \\ &= \lambda_\ell. \end{aligned}$$

Contradiction, since λ_ℓ is a non-zero value. □

Corollary 2. *Let $f \in \mathbb{R}^{\{-1,1\}^n}$ be a hidden function. Suppose f has at most s non-zero values. The function f can be uniquely identified given $\hat{f}(q)$ for all $q \in \{-1,1\}^n$ where $||q|| \leq \lceil \log s \rceil + 1$.*

Proof. Suppose of the contrary that we have $f_1, f_2 \in \mathbb{R}^{\{-1,1\}^n}$ where f_1 and f_2 have at most s non-zero values each and $\hat{f}_1(q) = \hat{f}_2(q)$ for all $q \in \{-1,1\}^n$ where $||q|| \leq \lceil \log s \rceil + 1$. Define $f = f_1 - f_2$. Clearly, f has at most $2s$ non-zero values. Moreover, we have $\hat{f}(q) = 0$ for all $q \in \{-1,1\}^n$ where $||q|| \leq \lceil \log s \rceil + 1 = \lceil \log 2s \rceil$. By Lemma 7, this is a contradiction. □

Lemma 8. *Let $f \in \mathbb{R}^{\{-1,1\}^n}$ be a hidden function. Suppose f has at most s non-zero values. There is an $O\left(s^2 n \sum_{i=0}^{2\log s+1} \binom{n}{i}\right)$ time algorithm that reconstruct f from the fourier coefficients $\hat{f}(q)$ for all $q \in \{-1,1\}^n$ where $||q|| \leq 2\lceil \log s \rceil + 1$.*

Proof. before giving the algorithm, we start by giving some notation. Given a function $f(x_1, x_2, \ldots, x_n) \in \mathbb{R}^{\{-1,1\}^n}$, denote by $f|_{x_{i_1}=\sigma_1, x_{i_2}=\sigma_2, \ldots, x_{i_r}=\sigma_r}$ the function f when fixing x_{i_j} to σ_j for all $j \in [r]$.

We start with the subroutine **Find_Non_Zero**, presented in Figure 2. This subroutine finds at least one vector $a \in \{-1,1\}^n$ for which $f(a) \neq 0$.

The subroutine is recursive. Each recursive call gets a non-zero function f' of the following form

$$f' = f|_{x_{i_1}=\sigma_1, x_{i_2}=\sigma_2, \ldots, x_{i_t}=\sigma_t},$$

where $\sigma_1, \sigma_2, \ldots, \sigma_t \in \{-1,1\}$ and $i_1, \ldots, i_t \in [n]$. Here t is called the *depth* of the recursive call. We also call $i_1, i_2, \ldots, i_t$ *fixed* indices, all the other indices are called *free* indices.

The subroutine works only if the depth of the recursive call is at most $\lceil \log s \rceil$ (Line 1). The subroutine examines the following functions

$$f'_{j,1} = f'|_{x_j=-1} \text{ and } f'_{j,2} = f'|_{x_j=1},$$

for every j that is a free index (Lines 4, 5). At any depth $t \leq \log s$ the subroutine is able to calculate the fourier coefficient $f'_{j,1}(q)$ and $f'_{j,2}(q)$ for every $q \in \{-1,1\}^n$ such that

$$||q|| \leq 2\lceil \log s \rceil + 1 - (t+1).$$

Subroutine **Find_Non_Zero(** $f' = f|_{x_{i_1}=\sigma_1, x_{i_2}=\sigma_2, \ldots, x_{i_t}=\sigma_t}$, depth**)**

1. **If** depth $> \lceil \log s \rceil$ **return**.
2. $a \leftarrow 0^n$.
3. **For all** $j \in [n] \setminus \{i_1, i_2, \ldots, i_t\}$ **do**
4. $\quad f'_{j,1} \leftarrow f'|_{x_j=-1}$.
5. $\quad f'_{j,2} \leftarrow f'|_{x_j=1}$.
6. $\quad$ **if** $f'_{j,1} \neq 0$ **and** $f'_{j,1} \neq 0$ **then**
7. $\quad\quad$ **Find_Non_Zero**$(f'_{j,1} = f|_{\sigma_1, x_{i_2}=\sigma_2, \ldots, x_{i_t}=\sigma_t, x_j=-1}$, depth + 1),
8. $\quad\quad$ **Find_Non_Zero**$(f'_{j,2} = f|_{\sigma_1, x_{i_2}=\sigma_2, \ldots, x_{i_t}=\sigma_t, x_j=1}$, depth + 1).
9. $\quad\quad$ **Return**.
10. $\quad$ **else if** $f'_{j_1} \neq 0$ **and** $f'_{j_1} = 0$ **then**
11. $\quad\quad$ $a_j \leftarrow -1$.
12. $\quad$ **else**
13. $\quad\quad$ $a_j \leftarrow 1$.
14. $\quad$ **end if**
15. **end for**.
16. **For all** $j \in [t]$ **do**
17. $\quad$ $a_{i_j} \leftarrow \sigma_j$.
18. **end for.**
19. **output** $(a, 2^{n-t}\hat{f}'(-1^{n-t}))$.
20. **return**.

Figure 2: Subroutine for finding non-zero values.

Thus, since $t \leq \lceil \log s \rceil$, the subroutine is able to calculate the fourier coefficient $f'_{j,1}(q)$ and $f'_{j,2}(q)$ for every $q \in \{-1, 1\}^n$ where

$$||q|| \leq \lceil \log s \rceil.$$

Therefore, by Lemma 7, the subroutine is able to determine whether the function $f'_{j,k}$ is the zero function or not, for every j that is a free index and $k \in [2]$.

The subroutine tries to find a free index j for which $f'_{j,1}$ and $f'_{j,2}$ are not zero (Line 6). If found, two recursive calls are made with the input $f'_{j,1}$ and $f'_{j,2}$ (Lines 7, 8). Otherwise, in case there is no such index j, then the function f' has only one non-zero value (recall that f' is a non-zero function). In this case, since it is known which of $f'_{j,1}$ and $f'_{j,2}$ is the zero function and which is not for every free index j (Lines 10-14), the subroutine knows the non-zero value of f'. It outputs a corresponding non-zero value of f (Line 19), that is, a vector $a \in \{-1, 1\}$ for which $f(a) \neq 0$.

Now, when running the subroutine **Find_Non_Zero** with f as an input, it must output at least one vector $a \in \{-1, 1\}^n$ for which $f(a) \neq 0$. This follows from the fact that f has at most s non-zero values.

Now, given the above subroutine and f, the algorithm is simple. It is presented in Figure 3. It runs the subroutine with f as an input. The subroutine returns vectors $a_1, a_2, \ldots, a_r \in \{-1, 1\}^n$ such that $f(a_i) \neq 0$ for all $i \in [r]$. The algorithm defines the following function

$$h(x) = \begin{cases} f(x) & x \in \{a_1, a_2, \ldots, a_r\} \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm then, runs recursively with input $f - h$. It does so iteratively until we are remained with the zero function.

During the running of the algorithm,

Algorithm **Reconstruct_using_Fourier_Coefficients(f)**

1. **if** $f = 0$ **return**.
2. $h \leftarrow 0$. // zero function over $\{-1,1\}^n$.
3. **Find_Non_Zero**(f).
4. **For all** $(a, f(a)) \in$ output of **Find_Non_Zero**, **do**
5. $h(a) \leftarrow f(a)$.
6. **end for**.
7. **Reconstruct_using_Fourier_Coefficients**($f - h$).

Figure 3: Algorithm for reconstructing a function f using its fourier coefficients.

Find_Non_Zero reveals all non-zero values and therefore, the function f is reconstructed.

As for complexity analysis each call of **Find_Non_Zero** calculate fourier coefficients of $f'_{j,1}$ and $f'_{j,2}$ for every free index j. This takes at most

$$O\left(n \sum_{i=0}^{2\log s+1} \binom{n}{i}\right),$$

time. In total, the algorithm runs at most s calls; within each such call at most s recursive calls are made to **Find_Non_Zero**. Therefore, the total time complexity is

$$O\left(s^2 n \sum_{i=0}^{2\log s+1} \binom{n}{i}\right).$$

□

Now after proving the above lemmas, we are ready to prove our main result, that is, prove Theorem 1.

Proof. Let ℓ be the smallest integer where

$$\sum_{i=\frac{\ell}{2}}^{\ell} \epsilon i \binom{\ell}{i} \geq n$$

Consider the matrix M_t defined in Appendix A, where $t = \ell 2^{\ell-1}$. Recall that every column of this matrix corresponds to a function $f_{a,k}$ where $a \in \{-1,1\}^\ell$ and $k \in [||a||]$. Remove All columns that correspond to functions $f_{a,k}$ where $||a|| \leq \ell/2$. Also remove all columns that corresponds to functions $f_{a,k}$ where k is less than $\frac{(1-\epsilon)\ell}{2}$. Denote the resulting matrix by M'. Note that the number of columns might be greater than n. If this occurs, remove arbitrary columns until we are remained with a matrix M'_n with n columns. Note that the number of rows in M_n is $2^\ell = O(n/\epsilon \log n)$. Also note that for all functions $f_{a,k}$ that correspond to a column in M'_n we have that all fourier coefficient $\hat{f}(q)$ where $||q|| < \frac{(1-\epsilon)\ell}{2}$ are equal to zero with the exception of $\hat{f}(-1^\ell)$ that is equal to $1/2$. Now, consider the following matrix

$$M^* = \left(\frac{M'_n}{1^{n^{1/4}\times n}}\right).$$

We argue that given $r = M^*v + e$, where $v \in \{0,1\}^n$, $e \in \mathbb{R}^n$ and $wt(e) \leq c\left(\frac{n}{\epsilon \log n}\right)^{\frac{1-\epsilon}{4}}$, one can reconstruct v in $O(n^{3/2})$ time. Denote by e' and r' the 2^ℓ vectors that contains the first 2^ℓ entries of e and r, respectively. We have that

$$r' = M'_n v + e'.$$

We view r', $M'_n v$ and e' as functions over $\{-1,1\}^\ell$ just as we did with the columns of M'_n. All fourier coefficients of $\hat{M'_n v}(q)$ for all q where $||q|| < \frac{(1-\epsilon)\ell}{2}$ are equal to zero with the exception of $\hat{M'_n v}(-1^\ell)$ that is equal to $\frac{v^T 1^n}{2}$. This follows from the fact

that all functions $f_{a,k}$ that correspond to a column in M'_n we have that all fourier coefficient $\hat{f}(q)$ where $||q|| < \frac{(1-\epsilon)\ell}{2}$ are equal to zero with the exception of $\hat{f}(-1^\ell)$ that is equal to 1/2. Given that $wt(e) \leq c\left(\frac{n}{\epsilon \log n}\right)^{\frac{1-\epsilon}{4}}$, where $c < 1/2$ is a constant, then a majority vote over the last $n^{1/4}$ entries of r determines $v^T 1^n$. Thus, using the value $v^T 1^n$ and all the above, we get that one can find all the fourier coefficients $\hat{e'}(q)$ for all q where $||q|| < \frac{(1-\epsilon)\ell}{2}$ by finding the fourier coefficients $\hat{r'}(q)$ for all q where $||q|| < \frac{(1-\epsilon)\ell}{2}$. Now, since $wt(e') \leq c\left(\frac{n}{\epsilon \log n}\right)^{\frac{1-\epsilon}{4}}$, by Lemma 8, we are able in $O(n^{3/2})$ time to reconstruct e' using the Fourier coefficients $\hat{e'}(q)$ for all q where $||q|| < \frac{(1-\epsilon)\ell}{2}$. This follows from the fact that

$$2^{\frac{(1-\epsilon)\ell}{4}-1} = O\left(\frac{n}{\epsilon \log n}\right)^{\frac{1-\epsilon}{4}}$$

In other words, we are able to find

$$M'_n v = r' - e'.$$

Finally, note that M'_n is a matrix formed by selecting columns from M_t. Therefore, using the reconstructing algorithm in [5], we are able to reconstruct the hidden vector v. □

4.2 Reconstructing (0,1)-vectors with Heavy Noise

The algorithm in the above subsection, although fast, it cannot handle large amount of errors. In this subsection, we show that one can compromise the running time to handle more errors and erasures. To do so, we rely on tools from coding theory. We show the following,

Theorem 2. *Let $v \in \{0,1\}^n$ be a hidden vector. There exists a non-adaptive polynomial time reconstructing algorithm that reconstructs v using*

$$k = O\left(\frac{n}{\epsilon \log n}\right)$$

queries. This algorithm reconstructs v correctly when the number of incorrect answers e_1 and the number of unknown answers e_2 are bounded by $\Omega(n^{1-\epsilon})$. Moreover, denote by $(q_1, q_2, \ldots, q_k)$ the vector of answers, that is, q_i equals the answer to the ith query, for $i \in [k]$. If the errors/erasures are consecutive, then, the algorithm works correctly even if their number is $\Omega(\frac{n}{\epsilon \log n})$.

Proof. To prove this theorem, we give a search matrix for the problem. That is, we build a matrix B such that given $Bv + e$, where $v \in \{0,1\}^n$ and e is any n-vector with Hamming weight smaller than $n^{1-\epsilon}$, one can reconstruct v in polynomial time.

We start by proving the following lemma.

Lemma 9. *Let $\mathcal{C}$ be a linear code $[p,k,d]$ over $\mathbb{Z}_2$ with the generating matrix G. Let $\mathcal{D}$ be a polynomial time decoding algorithm for $\mathcal{C}$. Suppose $\mathcal{D}$ is able to decode correctly in the presence of $d' \leq \frac{d-1}{2}$ errors. Let $\bar{G} \in \mathbb{Z}^{k \times p}$ be equal to G. Given $\bar{G}^T w + e$, where $w \in \mathbb{N}_0^k$ and $e \in \mathbb{Z}^p$ is any p-vector such that $wt(e) < d'$, one can reconstruct w in polynomial time in p.*

Proof. Let

$$b = \bar{G}^T w + e.$$

Let $w_{i,s}\, w_{i,s-1} \ldots w_{i,1}$ denote the binary representation of w_i for $i \in [n]$. Let $b_{i,t}\, b_{i,t-1} \ldots b_{i,1}$ denote the binary representation of b_i for $i \in [k]$. That is,

$$b = \begin{pmatrix} b_{1,t} \ldots b_{1,1} \\ b_{2,t} \ldots b_{2,1} \\ \vdots \\ b_{k,t} \ldots b_{p,1} \end{pmatrix} \quad \text{and} \quad w = \begin{pmatrix} w_{1,s} \ldots w_{1,1} \\ w_{2,s} \ldots w_{2,1} \\ \vdots \\ w_{k,s} \ldots w_{k,1} \end{pmatrix}.$$

Denote by $w^{\{j\}}$ (for $j \in [s]$) and by $b^{\{\ell\}}$ (for $\ell \in [t]$) the vectors

$$\begin{pmatrix} w_{1,j} \\ w_{2,j} \\ \vdots \\ w_{k,j} \end{pmatrix} \text{ and } \begin{pmatrix} b_{1,\ell} \\ b_{2,\ell} \\ \vdots \\ b_{p,\ell} \end{pmatrix},$$

respectively. Note that $b^{\{1\}} = b \mod 2$ and $\bar{G}^T w^{\{1\}} = \bar{G}^T w \mod 2$. Thus,

$$b^{\{1\}} = \bar{G}^T w^{\{1\}} \mod 2.$$

That is, we have that $\mathcal{D}(b^{\{1\}}) = w^{\{1\}}$. Now, using $\mathcal{C}$'s decoding algorithm $\mathcal{D}$, and the fact that

$$\mathcal{D}(b^{\{1\}}) = w^{\{1\}},$$

the algorithm can reconstruct $w^{\{1\}}$ simply by decoding $b^{\{1\}}$. Now, after finding $w^{\{1\}}$, the algorithm calculates $G^T w^{\{1\}}$. The algorithm calculates

$$b' = \bar{G}^T(w - w^{\{1\}}) + e.$$

It divides every entry in the vector b' by 2 and continue recursively to reconstruct $w^{\{2\}}$. This completes the proof. □

Now, recall the matrix M_{n^ϵ} defined in Appendix A. This matrix is of size

$$\frac{2n^\epsilon}{\epsilon \log n} \times n^\epsilon.$$

In the following, we abbreviate M_{n^ϵ} to M. Let $\mathcal{C}$ be any linear code $[c_1 n^{1-\epsilon}, n^{1-\epsilon}, c_2 n^{1-\epsilon}]$ over $\mathbb{Z}_2$ with generating matrix G and polynomial time decoding algorithm $\mathcal{D}$. Concatenated codes are an example to such code. Let,

$$B = G^T \otimes M = \begin{pmatrix} g_{1,1}M & g_{2,1}M & \cdots & g_{n^{1-\epsilon},1}M \\ g_{1,2}M & g_{2,2}M & \cdots & g_{n^{1-\epsilon},2}M \\ \vdots & \vdots & \ddots & \vdots \\ g_{1,c_1 n^{1-\epsilon}}M & g_{2,c_1 n^{1-\epsilon}}M & \cdots & g_{n^{1-\epsilon},c_1 n^{1-\epsilon}}M \end{pmatrix}.$$

The matrix B is of size $n/(\epsilon \log n) \times n$. We argue that given $Bv + e$, where $e \in \mathbb{Z}^n$ and $wt(e) \leq c_3 n^{1-\epsilon}$, one can reconstruct v in polynomial time.

Divide v into size n^ϵ blocks

$$v = \begin{pmatrix} v_1 \\ \hline v_2 \\ \hline \vdots \\ \hline v_{n^{1-\epsilon}} \end{pmatrix}.$$

Our goal is to reconstruct the vectors

$$Mv_1, Mv_2, \ldots, Mv_{n^{1-\epsilon}}.$$

Then, using the algorithm in [5], we reconstruct the vectors $v_1, v_2, \ldots, v_{n^{1-\epsilon}}$ and therefore, reconstruct v.

Denote by p the number of rows in M, that is, $p = \frac{2n^\epsilon}{\epsilon \log n}$. Denote by $x^{(i)}$ the vector

$$((Mv_1)_i, (Mv_2)_i, \ldots, (Mv_{n^{1-\epsilon}})_i),$$

for $i \in [p]$. Denote by $y^{(i)}$, for $i \in [p]$, the vector

$$((Bv)_i, (Bv)_{i+p}, (Bv)_{i+2p}, \ldots, (Bv)_{i+(c_1 n^{1-\epsilon}-1)p}).$$

Note that,

$$y^{(i)} = G^T x^{(i)}.$$

By Lemma 9, we are able to reconstruct $x^{(i)}$ from $y^{(i)} + e^{(i)}$ where $e^{(i)}$ is any vector in $\mathbb{Z}^{c_1 n^{1-\epsilon}}$ with Hamming weight bounded by $(c_2 n^{1-\epsilon} - 1)/2$. Therefore, by applying p times the reconstructing algorithm in Lemma 9, we are able from $Bv + e$ to reconstruct all $x^{(i)}$'s. This completes the proof. □

4.3 Reconstructing Bounded Weight Vector with Noise

In this subsection, we show an algorithm for reconstructing a (0,1) vector with at most m non-zero entries in the presence of noise. That is, we prove Theorem 3.

Theorem 3. *There is a polynomial time algorithm for reconstructing any hidden vector $v \in \{0,1\}^n$ that uses*

$$O\left(\frac{m \log n}{\epsilon \log m}\right)$$

queries, where m is the number of non-zeros in v. This algorithm reconstructs v correctly even in the present of $O(m^{1-\epsilon})$ errors or erasures.

To prove the above theorem, we start by extending the result from previous subsection. We give an algorithm for reconstructing any hidden positive integer vector in the presence of noise. We prove the following.

Theorem 10. *Let $v \in [d_1]_0 \times [d_2]_0 \times \cdots \times [d_n]_0$ be a hidden vector, where $d_1, \ldots, d_n$ are positive integers and $[d_i]_0 = [d_i] \cup \{0\}$. There exists a non-adaptive algorithm that reconstructs v using k queries, where*

$$k = O\left(\frac{n \log\left(\frac{\sum_i d_i}{n} + \frac{\max_i d_i}{n^\epsilon}\right)}{\epsilon \log n}\right)$$

This algorithm reconstructs v correctly when the number of errors e_1 and the number of erasures e_2 are bounded by $\Omega(n^{1-\epsilon})$. Moreover, if the errors are consecutive the algorithm works correctly even if $e_1 \leq \Omega(\frac{n}{\epsilon \log n})$.

Proof. We make use of the search matrices from [5]. As mentioned in Appendix A (Lemma 11), these matrices are used to reconstruct vectors $v \in [d_1]_0 \times [d_2]_0 \times \cdots \times [d_n]_0$ without the presence of noise, where the d_i's are positive integers. The number of rows k in such matrix is bounded by

$$k(\log k - 4) \leq 2n \log \frac{\sum_i d_i}{n}.$$

We now show how to use these matrices to build a search matrix for the problem with noise.

Assume w.l.o.g that $d_1 \geq d_2 \geq \ldots \geq d_n$. We divide our reconstruction problem (reconstructing v) into smaller vector reconstruction problems as follows: For $i \in [n^{1-\epsilon}]$, denote by M_i the search matrix defined in [5] for reconstructing the vector [1]

$$\begin{aligned} u_i &= (v_i, v_{i+n^{1-\epsilon}}, v_{i+2n^{1-\epsilon}}, \ldots, v_{i+n-n^{1-\epsilon}}) \\ &\in [d_i]_0 \times [d_{i+n^{1-\epsilon}}]_0 \times \cdots \times [d_{i+n-n^{1-\epsilon}}]_0. \end{aligned}$$

Since we assumed that $d_1 \geq d_2 \geq \ldots \geq d_n$, we know that the matrix M_1 has the maximal number of rows. We add zero rows to $M_2, \ldots, M_{n^{1-\epsilon}}$ so that all the matrices M_i have the same number of rows. Now, let $\mathcal{C}$ be any linear code $[c_1 n^{1-\epsilon}, n^{1-\epsilon}, c_2 n^{1-\epsilon}]$ over $\mathbb{Z}_2$ with generating matrix $G = (g_{ij})$ and polynomial time decoding algorithm $\mathcal{D}$. Consider the matrix

$$B = \begin{pmatrix} g_{1,1}M_1 & \ldots & g_{n^{1-\epsilon},1}M_{n^{1-\epsilon}} \\ g_{1,2}M_1 & \ldots & g_{n^{1-\epsilon},2}M_{n^{1-\epsilon}} \\ \vdots & \ddots & \vdots \\ g_{1,c_1 n^{1-\epsilon}}M_1 & \ldots & g_{n^{1-\epsilon},c_1 n^{1-\epsilon}}M_{n^{1-\epsilon}} \end{pmatrix}.$$

We argue that BP is a search matrix for our problem, where P is a permutation matrix that reorder v so that the vector Pv equals the vector

$$Pv = (u_1|u_2|\ldots|u_{n^{1-\epsilon}})^T.$$

[1] note that we abuse notation here. The matrices M_i are not the ones constructed in Appendix A, they are only mentioned in the appendix in Lemma 11

Denote by k_1 the number of rows in M_1. Denote by x_i the vector

$$((M_1 u_1)_i, (M_2 u_2)_i, \ldots, (M_{n^{1-\epsilon}} u_{n^{1-\epsilon}})_i),$$

for $i \in [k_1]$. Denote by y_i the vector $((BPv)_i, (BPv)_{i+k_1}, \ldots, (BPv)_{i+(c_1 n^{1-\epsilon}-1)k_1})$ for $i \in [k_1]$. Note that

$$y_i = G^T x_i.$$

Therefore, using Lemma 9, we are able to reconstruct every x_i. Once the vectors $M_i u_i$ are known, we are able to reconstruct u_i for every i, that is, reconstruct the hidden vector v.

As for the query complexity, the number of rows in B is equal to the number of rows in M_1 times the number of rows in G^T. The number of rows in M_1, k_1, is bounded by

$$k_1(\log k_1 - 4) \leq 2n^{\epsilon} \log \frac{d_1 + d_{1+n^{1-\epsilon}} + \cdots + d_{n-n^{1-\epsilon}+1}}{n^{\epsilon}}.$$

Now, we have

$$\begin{aligned} &\frac{d_1 + d_{1+n^{1-\epsilon}} + \cdots + d_{n-n^{1-\epsilon}+1}}{n^{\epsilon}} \\ &= \frac{n^{1-\epsilon}(d_1 + d_{1+n^{1-\epsilon}} + \cdots + d_{n-n^{1-\epsilon}+1})}{n} \\ &\leq \frac{d_1 n^{1-\epsilon} + \sum_{i=1}^{n} d_i}{n} \\ &= \frac{\sum_{i=1}^{n} d_i}{n} + \frac{d_1}{n^{\epsilon}} \end{aligned}$$

Summing all the above, we get that the number of rows k is bounded by

$$k = O\left(\frac{n \log\left(\frac{\sum_i d_i}{n} + \frac{d_1}{n^{\epsilon}}\right)}{\epsilon \log n}\right)$$

This completes the proof. □

Corollary 3. *Let $v \in [d_1]_0 \times [d_2]_0 \times \cdots \times [d_{n'}]_0$ be a hidden vector, where $d_1, \ldots, d_{n'}$ are positive integers and $n' \leq n$. Suppose $\max_i d_i \leq n^{\epsilon}$. There exists a non-adaptive algorithm that reconstructs v using k queries, where*

$$k = O\left(\frac{n}{\epsilon \log n} \log\left(\frac{\sum_i d_i}{n}\right)\right).$$

This algorithm reconstructs v correctly when the number of errors e_1 and the number of erasures e_2 are bounded by $\Omega(n^{1-\epsilon})$.

We are now ready to prove our main result, that is, Theorem 3.

Proof. The algorithm we present contains two stages. In the first stage, the algorithm's goal is to divide the set of entries in v into disjoint sets $S_1, S_2, \ldots, S_t$ where the sum of the entries in each set S_i is bounded from above by $m^{\epsilon/2}$.

For any k, let $\mathcal{C}_k$, be any linear $[p, k, d]$ code over $\mathbb{Z}_2$, where k/p and d/p are $\Omega(1)$. Let G_k be its generating matrix and $\mathcal{D}$ be its polynomial time decoding algorithm.

Let $S = S_1, S_2, \ldots, S_t$ be disjoint sets, where $S_i \subseteq [n]$. We denote by u^S the t-vector for which

$$u_i^S = \sum_{j \in S_i} v_j.$$

The algorithm's first stage is presented in Figure 4

The first stage uses the divide and conquer approach, it finds a set $H = \{S_1, S_2, \ldots, S_t\}$, where for every i, we have $S_i \subseteq [n]$, $0 < \sum_{j \in S_i} v_j \leq m^{\epsilon/2}$ and the sets in H are disjoint. Also, the set $\bigcup_{S_i \in H} S_i$ contains all non-zero entries in v.

The first stage is iterative. At the beginning of each iteration, we have disjoint sets $S_1, S_2, \ldots, S_t \subseteq [n]$, where the sum of the entries in each set is at least $m^{\epsilon/2}$. The algorithm divides each set S_i into equal size sets. It continue dividing those sets until we have more than $m^{1-\epsilon/2}$ sets. Then, using Lemma 9, it finds the sum of the entries in each new set. The algorithm throws all new sets for which the sum of the entries is zero. Sets for which the sum of the entries is between "1" and $m^{\epsilon/2}$ are added to the output. Finally, all the other sets remain or advance to the next iteration.

The second stage is also iterative. It continues to use the divide and conquer approach. For $S_i \subseteq [n]$, denote by $X(S_i)$ the sum of entries $\sum_{j \in S} v_j$. At each iteration, the algorithm have sets $S_1, S_2, \ldots, S_t$ of indices. The algorithm knows $X(S_i)$ for every set S_i. The algorithm divides each set S_i into two equal size sets (up to ± 1), $S_{i,1}, S_{i_2}$. The algorithm uses Corollary 3 to reconstruct the hidden vector

$$(X(S_{1,1}), X(S_{2,1}), \ldots, X(S_{t,1})) \in [X(S_1)]_0 \times [X(S_2)]_0 \times \ldots \times [X(S_t)]_0$$

It throw all set $S_{i,j}$ for which $X(S_{i,j})$ equal zero and advanced to the next iteration. After at most $\log n$ iteration, the algorithm knows the all the non-zero entries in the hidden vector v.

As for complexity analysis, the algorithm's first stage asks at most

$$O(m^{1-\epsilon/2}) \log \frac{n}{m^{\epsilon/2}}$$

queries. In the second stage the algorithm asks, each iteration (each time it applies Corollary 3) asks

$$O\left(\frac{m}{\epsilon \log m}\right)$$

queries. Since we have at most $\log n$ iteration in the second stage, our total query complexity is

$$O\left(\frac{m \log n}{\epsilon \log m}\right)$$

This completes the proof. □

References

[1] M. Aigner. Combinatorial Search. *John Wiley and Sons*, 1988.

[2] N. Alon and V. Asodi. Learning a Hidden Subgraph. *SIAM J. Discrete Math*, 18, 4, 697–712, 2005.

[3] E. Biglieri and L. Györfi. Multiple Access Channels Theory and Practice Volume 10 NATO Security through Science Series - D: Information and Communication Security, April 2007.

[4] L. Bruneau and F. Germinet. On the singularity of random matrices with independent entries *Proc. Amer. Math. Soc.* 137 (2009), 787-792.

Algorithm's first stage

1. $S \leftarrow \{[n]\}$
2. $H \leftarrow \emptyset$.
3. **while** $(|S| < m^{1-\epsilon/2})$
4. Split every $S_i \in S$ into two equal (up to ± 1) sets.
6. **End while**
7. Find (by asking queries) $G^T_{|S|}\, u^S$.
8. Use Lemma 9 to reconstruct u^S.
9. For every $S_i \in S$ for which $u^S_i = \sum_{j \in S_i} v_j \leq m^{\epsilon/2}$.
10. Remove S_i from S.
11. **if** $(u^S_i > 0)$ **then** add S_i to H.
12. **End for**
13. **if** $(|S| = 0)$ **then** finish **else** Goto 3.

Figure 4: Reconstructing with noise. Algorithm's first stage.

[5] N. H. Bshouty. Optimal Algorithms for the Coin Weighing Problem with a Spring Scale. *Conference on Learning Theory*, 2009.

[6] N. H. Bshouty and H. Mazzawi. On Parity Check $(0,1)$-Matrix over $\mathbb{Z}_p$. *SODA*, 2011.

[7] N. H. Bshouty and H. Mazzawi. Toward a Deterministic Polynomial Time Algorithm with Optimal Additive Query Complexity. *MFCS*, 2010.

[8] D. Cantor. Determining a set from the cardinalities of its intersections with other sets, *Canadian Journal of Mathematics*, V. 16, 94–97, 1962.

[9] J. Cheng, K. Kamoi and Y. Watanabe. User Identification by Signature Code for Noisy Multiple-Access Adder Channel. *ISIT*, 2006.

[10] D. Cantor, W. Mills. Determining a Subset from Certain Combinatorial Properties. *Canad. J. Math.* V. 18, 42–48, 1966.

[11] S.C. Chang and E.J. Weldon. Coding for T-user multiple access channels. IEEE Transactions on Information Theory, 25(6), 684–691, 1979.

[12] J. Cheng and Y. Watanabe. A Multiuser k-Ary Code for the Noisy Multiple-Access Adder Channel. *IEEE Transactions on Information Theory*, vol 47, 6, 2001.

[13] J. Cheng and Y. Watanabe. Affine Code for T-User Noisy Multiple Access Adder Channel. *IEICE Trans. Fundamentals*, vol. E83-A, no. 3, 2000.

[14] S. Choi, J. Han Kim. Optimal Query Complexity Bounds for Finding Graphs. *STOC*, 749–758, 2008.

[15] J. Cheng, K. Kamoi and Y. Watanabe. User Identification by Signature Code for Noisy Multiple-Access Adder Channel. *IEEE International Symposium on Information Theory*, 1974–1977, 2006.

[16] D. Du and F. K. Hwang. Combinatorial group testing and its application, Volume 3 of Series on applied mathematics. *World Science*, 1993.

[17] D. Danev, B. Laczay and M. Ruszinkó. Multiple Access Adder Channel. *Multiple Access Channels - Theory and Practice, IOS Press*, 26–53, 2007.

[18] Erdös and A. Rényi. On two problems of information theory. *Publ. Math. Inst. Hung. Acad. Sci.* V. 8, 241–254, 1963.

[19] V. Grebinski and G. Kucherov. Optimal Reconstruction of Graphs Under the Additive Model. *Algorithmica* , 28(1), 104–124, 2000.

[20] V. Grebiniski and G. Kucherov. Reconstructing a hamiltonian cycle by querying the graph: Application to DNA physical mapping. *Discrete Applied Mathematics*, 88, 147–165, 1998.

[21] V. Grebinski. On the Power of Additive Combinatorial Search Model. *COCOON*, 194–203 , 1998.

[22] P. Indyk, M. Ruzic. Near-Optimal Sparse Recovery in the L1 Norm. FOCS 2008: 199-207, 2008.

[23] G.K. Khachatrian, S.S. Martirossian. Codes for T-user Noiseless Adder Channel. Problems of Control and Information Theory, vol 16, 187–192, 1987.

[24] J. Komlós, On the determinant of matrices, *Studia. Sci. Math. Hungar.*. 2, 7-21 (1967).

[25] B. Laczay. Coding for the Multiple Access Adder Channel. 2003.

[26] B. Lindström. On a combinatorial problem in number theory. *Canad. Math. Bull.* 8, 477–490, 1965.

[27] B. Lindström. On a combinatorial detection problem II. *Studia Scientiarum Mathematicarum Hungarica*. 1. 353–361, 1966.

[28] B. Lindström. On Möbius functions and a problem in combinatorial number theory. *Canad. Math. Bull.* 14(4), 513–516, 1971.

[29] B. Lindström. Determining subsets by unramified experiments. In J.N. Srivastava, editor, *A Survey of Statistical Designs and Linear Models.* North Holland, Amsterdam, 407–418, 1975.

[30] M. Li, P. M. B. Vitányi. Combinatorics and Kolmogorov Complexity. *Structure in Complexity Theory Conference*. 154–163. 1991.

[31] L. Moser. The second moment method in combinatorial analysis. In Combinatorial Structure and their applications, Gordon and Breach, 283–384, 1970.

[32] N. Pippenger. An Informtation Theoretic Method in Combinatorial Theory. *J. Comb. Theory, Ser. A*. 23(1), 99–104, 1977.

[33] N. Pippenger. Bounds on the performance of protocols for a multiple-access broadcast channel. *IEEE Transactions on Information Theory.* 27(2), 145–151, 1981.

[34] S. Soderberg, H. S. Shapiro. A combinatory detection problem. *American Mathematical Monthly*, 70, pp. 1066–1070, 1963.

[35] J.H. Wilson. Error-Correcting Codes for a T-User Binary Adder Channel. *IEEE Transactions of Information Theory*, vol. 34, 4, 1988.

Appendix A

In this section we present a family of matrices defined in [5] that optimally solve the $\text{coin}_B(n)$ weighing problem in the noiseless case.

Let $a \in \{-1,1\}^\ell$. Let $j_1, j_2, \ldots, j_{||a||}$ be the indices of the entries in a that are equal to one (Recall that $||a||$ denotes the number of ones in a). For $k \in [||a||]$, define the following function

$$g_{a,k}(x) = \left(2\prod_{i=1}^{k}\frac{x_{j_i}+1}{2} - 1\right) x_{j_{k+1}} x_{j_{k+2}} \cdots x_{j_{||a||}}.$$

Now, let $f_{a,k}(x) = (g_{a,k}(x)+1)/2$. Define the following family of functions

$$F_\ell = \{f_{a,k} | a \in \{-1,1\}^\ell \text{ and } k \in [||a||]\}.$$

Define the matrix $M_t \in \{0,1\}^{2^\ell \times \ell 2^{\ell-1}}$ in the following way (we assume w.l.o.g. that $t = \ell 2^{\ell-1}$ for some integer ℓ): First, we label the rows of M_t with the elements of $\{-1,1\}^\ell$. Next, we label the columns with element of F_ℓ. The entries of M_t are define as follows: $M_t[x, f_{a,k}] = f_{a,k}(x)$. That is, let $z_1 = 1^\ell$, $z_2 = 1^{\ell-1} \cdot -1$, $\ldots$, $z_{2^\ell} = -1^\ell$, where $\cdot$ denotes concatenation, then

$$M_t = \left(\begin{array}{ccc|ccc|c|ccc} f_{z_1,1}(z_1) & \ldots & f_{z_1,\ell}(z_1) & f_{z_2,1}(z_1) & \ldots & f_{z_2,\ell-1}(z_1) & \ldots & f_{z_{2^{\ell-1}},1}(z_1) \\ f_{z_1,1}(z_2) & \ldots & f_{z_1,\ell}(z_2) & f_{z_2,1}(z_2) & \ldots & f_{z_2,\ell-1}(z_2) & \ldots & f_{z_{2^{\ell-1}},1}(z_2) \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ f_{z_1,1}(z_{2^\ell}) & \ldots & f_{z_1,\ell}(z_{2^\ell}) & f_{z_2,1}(z_{2^\ell}) & \ldots & f_{z_2,\ell-1}(z_{2^\ell}) & \ldots & f_{z_{2^{\ell-1}},1}(z_{2^\ell}) \end{array}\right).$$

Note that the columns of the matrix are the truth tables of the functions in F_ℓ.

The above matrix is an optimal size search matrix for the $\text{coin}_B(n)$ weighing problem in the noiseless case [5]. That is, suppose that $v \in \{0,1\}^n$ is a hidden vector, then there is an algorithm that reconstruct v given $M_n v$.

In [5] Bshouty also show a generalization for the above family of matrices. He gave a construction for similar search matrices for the following problem.

Lemma 11. *[5] Let $v \in [d_1]_0 \times [d_2]_0 \times \cdots \times [d_n]_0$ be a hidden vector. There is a matrix $M \in \{0,1\}^{k\times n}$ where*

$$k(\log k - 4) \le 2n \log \frac{\sum_i d_i}{n}.$$

such that given Mv there is a polynomial time algorithm that reconstructs the hidden vector v.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de