Reconstructing Graphs Using Edge Counting Queries

Hanna Mazzawi

Reconstructing Graphs Using Edge Counting Queries

Research Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Hanna Mazzawi

Submitted to the Senate of the
Technion - Israel Institute of Technology

Tishrei 5772 Haifa October 2011

The Research Thesis was Done Under the Supervision of Prof. Nader H. Bshouty in the Faculty of Computer Science.

I would like to express my deepest gratitude to my advisor, Prof. Nader H. Bshouty, for his academic parenting. I feel fortunate to have him as my advisor and could never thank him enough for his devoted support, encouragement and guidance throughout this research.

I thank my friends for the good times we had in the last few years. My friends have always been there for me in times of need and for that I am very grateful.

Last, but not least, I thank my family for their endless love, encouragement and support. This thesis is dedicated to them.

The generous financial help of the Technion is gratefully acknowledged.

Contents

List of Tables

List of Figures

Abstract

In this thesis we study three well known combinatorial search problems in various settings: The coin weighing problem, the problem of reconstructing graphs using additive queries and the problem of reconstructing hypergraphs using additive queries.

All of the three combinatorial search problems share a common structure. In each problem we have a set of objects called a *universe* or an *instance space*. From the instance space a unique object is selected, we call it the *hidden object*. To solve a combinatorial search problem an algorithm must uniquely identify the hidden object by asking minimal number of queries of a given type.

We distinguish between two types of algorithms to solve any combinatorial search problem. Adaptive algorithms are algorithms that take into account outcomes of previous queries while non-adaptive algorithms make all queries in advance, before any answer is known.

In the coin weighing problem, we have a hidden n-vector v with m non-zero real number entries. We are allowed to ask queries of the form

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

where $x \in \{0,1\}^n$. We show the existence of a non-adaptive algorithm for reconstructing the hidden vector with query complexity that matches the information theoretic lower bound. We also give a polynomial time adaptive algorithm for the same problem. This algorithm is optimal for smaller m's and almost optimal for all range of m. Moreover, we introduce few techniques that make our algorithms resistant to noise. That is, using these techniques our algorithms reconstruct correctly the hidden vector even if some of the answers to the queries are incorrect.

In the problem of reconstructing a hidden graph from queries, we have a hidden graph $G = (V, E, w)$, where $E \subseteq V \times V$ and $w \in \mathbb{R}^E$. The set of vertices V is known, and the set of edges E and their weights are unknown. We are allowed to ask queries of the form

$$Q_G(S) = \sum_{e \in E \cap (S \times S)} w(e),$$

for $S \subseteq V$. That is, the query returns the sum of weights of the edges in the subgraph induced by S. For the problem of reconstructing a hidden weighted graph, we show the existence of a non-adaptive algorithm with query complexity that matches the information theoretic lower bound. We also give the first optimal polynomial time adaptive algorithm for reconstructing unweighted graphs. We then extend this result by giving an almost optimal polynomial time adaptive algorithm for reconstructing weighted graphs.

Finally, we study the problem of reconstructing hidden hypergraphs. We have a hidden hypergraph $H = (V', E', w')$, where $E' \subset 2^{V'}$ and $w' \in \mathbb{R}^{E'}$. As in the previous problem, the set of vertices is the only set known, and one must reconstruct the hidden hypergraph using queries of the form

$$Q_G(S) = \sum_{e \in E \cap 2^S} w'(e),$$

for $S \subseteq V$. That is, the query returns the sum of weights in the subgraph induced by S. For this problem, we show the existence of a non-adaptive algorithm for reconstructing an unweighted hypergraph with constant rank with query complexity that matches the information theoretic lower bound. Here the rank of a hypergraph is the size of its largest edge. We also prove the existence of a non-adaptive algorithm for constant rank weighted graphs with query complexity that matches the information theoretic lower bound.

To achieve the above results, we use numerous techniques from theoretical computer science such as the probabilistic method, fourier coefficients analysis of functions, the divide and conquer approach, the guess and check approach, methods from coding theory and more.

Notation and Abbreviations

$[c]$	–	The set $\{1, 2, \ldots, c\}$.
$[c]_0$	–	The set $[c] \cup \{0\}$.
$\mathbb{R}$	–	The set of real numbers.
$\mathbb{R}^+$	–	The set of positive real numbers.
$\mathbb{R}_0^+$	–	The set of non-negative numbers.
$\mathbb{Z}$	–	The set of integers.
$\mathbb{Z}^+$	–	The set of positive integers.
$\mathbb{Z}_0^+$	–	The set of non-negative integers.
$\mathbb{Z}_p$	–	The field of integers modulo the prime p.
$wt(x)$	–	The Hamming weight of x.
$\psi(x)$	–	The sum of entries in x.
$\chi_a(x)$	–	The function $\prod_{i:a_i=1} x_i$.
$\hat{f}(a)$	–	The Fourier coefficient of χ_a in f.
$M^{[j]}$	–	The first j rows of M.
$x * y$	–	The component-wise product between the two vectors.
" $*$ "	–	Entry in a matrix that is unknown.
$\|\|q\|\|$	–	The number of ones in the vector q.
β	–	The constant $\frac{1}{2+\log 3} = 0.2789\cdots$.
$[n, k, d]$	–	Linear code of length n, dimension k and minimal distance d.
$X(S)$	–	The sum of entries in the set S.
$f\|_{x_i=\sigma}$	–	The function f when fixing x_i to σ.
$a\|_{(j_1,j_2,\ldots,j_t)}$	–	The t-vector where the ith entry equals a_{j_i}.

Chapter 1

Introduction

In this research we study several combinatorial search problems. In a combinatorial search problem we have a set of objects called a *universe* or an *instance space*. From the instance space a unique object is selected, we call it the *hidden object*. To solve a combinatorial search problem an algorithm must uniquely identify the hidden object by asking queries of a given type.

We distinguish between two types of algorithms to solve any combinatorial search problem. Adaptive algorithms are algorithms that take into account outcomes of previous queries while non-adaptive algorithms make all queries in advance, before any answer is known.

We explore three well known problems: The coin weighing problems, Section 1.1, the problems of reconstructing graphs using additive queries, Section 1.2, and the problems of reconstructing hypergraphs using additive queries, Section 1.3.

1.1 The Coin Weighing Problems

The coin weighing problems are also called the problems of reconstructing hidden vectors using queries. In these problems, we have a hidden vector v of size n. We are allowed to ask queries of the form

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

where $x \in \{0,1\}^n$. Our goal is to exactly reconstruct the hidden vector v using queries.

The first coin weighing problem that was studied in the literature is the $\mathbf{coin}(n)$ weighing problem. In this problem, the hidden vector v is a $(0,1)$-vector. The information theoretic lower bound for the query complexity of this problem is

$$\frac{n}{\log(n+1)}.$$

The $\mathbf{coin}(n)$ weighing problem was first studied in [15] by Cantor, in [45] by Soderberg and Shapiro, in [26] by Erdös and Rényi, in [34, 35, 36, 37] by Lindström and in [17] by Cantor and Mills. For this problem, Erdös and Rényi proved a lower bound of $2n/\log n$ queries. See also [35, 40, 37, 41, 38]. Later, Lindström [34] and independently Cantor and Mills [17] showed a non-adaptive polynomial time algorithm for the problem with query complexity that matches the lower bound[1]. Simplifications for Lindström's algorithm were introduced in [37, 1, 10].

The next problem is the $\mathbf{coin}(n,m)$ weighing problem. In this problem, the hidden vector v is a $(0,1)$-vector with at most m non-zero entries. The information theoretic lower bound for the query complexity of this problem is

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

Note that the $\mathbf{coin}(n)$ problem is the same as the $\mathbf{coin}(n,n)$ problem. In [27] Grebinski and Kucherov gave a non-constructive non-adaptive algorithm for the problem that matches this lower bound for all m. That is, they proved the existence of an algorithm with query complexity that matches the information theoretic lower bound without showing an explicit construction.

Several papers in the literature generalize the former results by showing algorithms for the $\mathbf{coin}^{+}(n,\sum v_i \leq k)$ problem. In this problem, the hidden vector v has entries that are non-negative *integers* and the sum of the entries (L_1-norm) is bounded by k. In [42] Pippenger gave a non-constructive algorithm in case $k=n$. This algorithm asks $O(n/\log n)$ queries. In [29] Grebinski extended this result by showing an optimal non-constructive non-adaptive algorithm for $n^2 \geq k \geq n$. In [44] Ruszinkó and Vanroose gave the first polynomial time adaptive algorithm for the case of $k=n$. This algorithm asks

[1]Lindström's algorithm works for the more general case of reconstructing a hidden vector from $[d]^n$. In this case, it uses $O(n/\log_d n)$ queries.

$O((n \log \log n)/\log n)$ queries. The algorithm runs in polynomial time, however, its query complexity is not optimal.

For many years, the only algorithms reaching the information theoretic lower bound for all the above mentioned problems were non-constructive (except for the $\mathbf{coin}(n)$ problem). It was only until recently that Bshouty [10] gave the first optimal polynomial time adaptive algorithm for the $\mathbf{coin}^{+}(n, \sum v_i \leq k)$ weighing problem. For $k \leq n$, this algorithm asks

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

queries. Obviously, this algorithm is also an optimal algorithm for the $\mathbf{coin}(n, m)$ weighing problem.

Finally, we have the more general $\mathbf{coin}_{\mathbb{R}}(n, m)$ and $\mathbf{coin}^{+}_{\mathbb{R}}(n, m)$ weighing problems. In these problems, the hidden vector v is a vector in $\mathbb{R}^n$ and $(\mathbb{R}^{+}_0)^n$, respectively, where $\mathbb{R}$ is the set of real numbers, and the hidden vector has at most m non-zeros entries. The information theoretic lower bound for the query complexity of the $\mathbf{coin}_{\mathbb{R}}(n, m)$ and the $\mathbf{coin}^{+}_{\mathbb{R}}(n, m)$ problems is

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

In [21] Choi and Han Kim showed an optimal non-constructive non-adaptive algorithm for reconstructing a hidden vector with at most m non-zero entries, where each non-zero entry holds a real number with magnitude bounded by n^{-a} and n^{b} for any constants a and b.

In our research, we first close the gap left in [21]. We show the following,

Theorem 1. *[12] For any prime p there exists a matrix $M \in \{0,1\}^{k \times n}$ such that*

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

and every m columns are linearly independent over $\mathbb{Z}_p$.

The existence of such matrix for a prime $p > m$ yields a non-constructive non-adaptive optimal algorithm for reconstructing *any* hidden *non-bounded real numbers* vector with at most m non-zero entries. Moreover, it yields an optimal algorithm for the $\mathbf{coin}_{\mathbb{Z}_p}(n, m)$ weighing problem where the hidden vector is in $\mathbb{Z}_p$ with at most m non-zero entries.

To prove the above theorem, we show that a random $(0,1)$-matrix of the mentioned size over $\mathbb{Z}_p$ satisfies the above theorem with a high probability. Our construction requires $n \cdot k$ random bits. To reduce the number of random bits, one can use n random variables that are m-wise independent. This gives a construction with $O((m^2 \log^2 n)/\log m)$ random bits. In our research, we show a new technique that gives for any $m = n^c$ where c is a constant, a construction that uses $O(m^{1+\epsilon})$ random bits for any constant ϵ. Each row in the constructed matrix is a tensor product of a (constant) d $(0,1)$-vectors of size $n^{1/d}$. This reduces the number of random bits for the construction of the corresponding coin weighing algorithm to $O(m^{1+\epsilon})$ as well.

Although there is a reduction in the number of random bits used, the above mentioned algorithm is still non-constructive. Unlike the restricted $\mathbf{coin}^+(n, \sum v_i \leq k)$ problem, for the $\mathbf{coin}^+_{\mathbb{R}}(n,m)$ and $\mathbf{coin}_{\mathbb{R}}(n,m)$ problems all algorithms that match the information theoretic lower bound are non-constructive and therefore proving an upper bound only. They prove that there exists a set of queries such that the answer to the queries uniquely identify the hidden vector; However, it is unknown how to deterministically construct this set of queries in polynomial time, and moreover, given the answers to such set of queries, it is unknown how to reconstruct the hidden vector in polynomial time. The best polynomial time algorithm for the $\mathbf{coin}^+_{\mathbb{R}}(n,m)$ problem is the trivial recursive search which asks $O(m \log \frac{n}{m})$ queries. This bound can also be achieved for the $\mathrm{coin}_{\mathbb{R}}(n,m)$ as well. See [30] and references within.

In our research, we show the following,

Theorem 2. *[13] There is a polynomial time adaptive algorithm for reconstructing a hidden vector $v \in (\mathbb{R}^+_0)^n$ with at most m non-zero entries that uses*

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

queries.

This is the first polynomial time algorithm that beats the trivial $O(m \log \frac{n}{m})$ recursive search. Our algorithm matches the information theoretic lower bound when $m \leq n^{c/\log \log n}$, where c is any constant. Also, our query complexity is within $\log \log m$ times the optimal complexity for all m.

Below is a table that summarize all known results.

Type	Hidden vector v	Restriction	Lower bound	Tight upper bound	Polynomial time
Non Adaptive	$\{0,1\}^n$	$m = n$	$\Omega\left(\frac{n}{\log n}\right)$	[34, 17]	[34, 17]
		–	$\Omega\left(\frac{m \log \frac{n}{m}}{\log m}\right)$	[27]	OPEN
	$\mathbb{Z}^n$	–	$\Omega\left(\frac{m \log n}{\log m}\right)$	Ours [12]	OPEN
	$\mathbb{R}^n$	$\forall i : v_i \in [n^{-a}, n^b]$	$\Omega\left(\frac{m \log n}{\log m}\right)$	[21]	OPEN
		–	$\Omega\left(\frac{m \log n}{\log m}\right)$	Ours [12]	OPEN
Adaptive	$\{0,1\}^n$	–	$\Omega\left(\frac{m \log \frac{n}{m}}{\log m}\right)$	[27]	[10]
	$(\mathbb{Z}_0^+)^n$	$\sum_i v_i = O(m)$	$\Omega\left(\frac{m \log \frac{n}{m}}{\log m}\right)$	[29]	[10]
		–	$\Omega\left(\frac{m \log n}{\log m}\right)$	Ours [12]	Ours [13]
	$(\mathbb{R}_0^+)^n$	–	$\Omega\left(\frac{m \log n}{\log m}\right)$	Ours [12]	Ours [13]

Table 1.1: Known results for the coin weighing problem.

1.1.1 Multiple Access Adder Channels

In this subsection we introduce one application, for the coin weighing problem. We introduce the signature coding problem for the multiple access adder channels [7].

Consider n stations or users that transmit information using a common channel. Each user i wants to transmit a bit of information at each iteration. To do so, he 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. Assume that codewords sent through the channel by the users are bit and block synchronized. Each user i wants to send one of the words $y_i \in C_i$. The output of the channel is a vector in $\mathbb{N}^k$ that is equal to,

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

In the *permanently active* model, a code $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ is *uniquely decipherable* (u.d.) if the messages of the users can be recovered from Y. Note that this definition is for channels without noise. In later sections, we discuss the signature coding problem with noise.

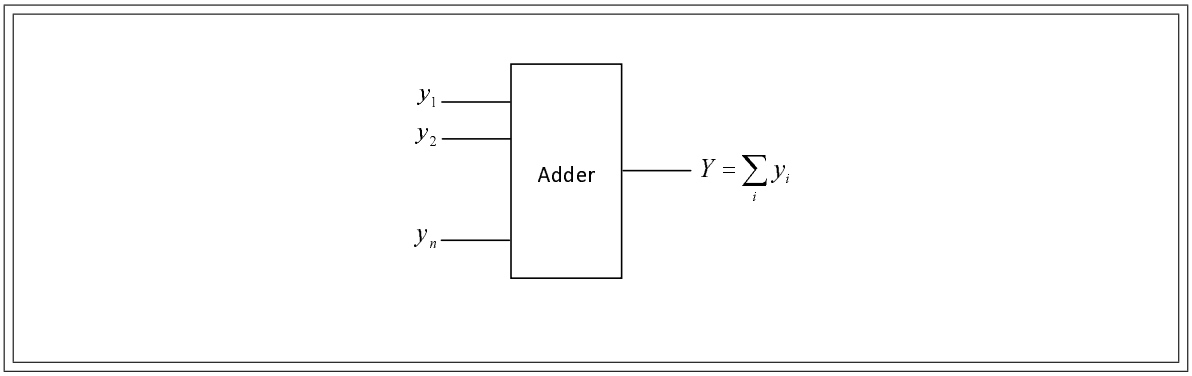

Figure 1.1: Multiple access adder channel

One use of u.d. signature codes is the following [7]. Consider an alarm system, where components or users that want to signal an alarm send their non-zero codeword. Other components which do not want to send an alarm signal turn off their transmitter. The goal of the receiver is to detect the alarming components.

There are many constructions for u.d. signature codes (both for the noiseless case and the noisy case) in this model in the literature, see [34, 17, 18, 31, 25, 23, 33]. A non-adaptive algorithm for the **coin**(n) weighing problem with query complexity k yields a u.d. signature code $\mathcal{C}$ of length k for n users. Therefore, Lindström [34] and Cantor and Mills [17] show how to construct optimal u.d. codes for the noiseless problem.

Now, for many real life examples, we have that only few of the users are active at any given time. For this scenario we have the *partially active* model. In this model, a code $\mathcal{C} = \{C_1 = \{0^k, x_1\}, \ldots, C_n = \{0^k, x_n\}\}$, where $x_1, \ldots, x_n \in \{0,1\}^k$, is *uniquely decipherable* if for any two subsets $S_1, S_2 \subseteq [n]$ where $S_1 \neq S_2$, $|S_1| \leq m$ and $|S_2| \leq m$, we have

$$\sum_{i \in S_1} x_i \neq \sum_{j \in S_2} x_j.$$

That is, the message of every user can be recovered given that at most m users are active.

For the partially active signature coding problem, Biglieri and Györfi [7], showed that there exists u.d. code of length

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

Theorem 1 above, generalizes this result by showing the existence of optimal length (up to

constant factor) codes when the addition is modulo any prime p. Moreover, if we assume that the channel is implemented so that the user i sends only a (0,1)-message, $b_i \in \{0,1\}$, and the adder sends

$$\sum_i b_i x_i,$$

then, Theorem 1 implies that for large primes p (primes that are greater than m^c for any constant c) there is a u.d. code of length

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

where a user can send a message $b_i \in \mathbb{Z}_p$ rather than a $(0,1)$-message. The length of this code matches the information theoretic lower bound.

1.2 Reconstructing Hidden Graphs using Queries

In this section, we introduce the problems of reconstructing a hidden graph using queries.

The weighted graph reconstructing problem, denoted by $\mathbf{graph}^+_{\mathbb{R}}$, is the following [28]: Let $G = (V, E, w)$ be a weighted hidden graph, where $E \subseteq V \times V$ and $w \in (\mathbb{R}^+)^E$ (here $\mathbb{R}^+$ denotes the set of positive real numbers). Suppose that the set of vertices is known. The goal is to exactly reconstruct the set of edges and their weights using additive queries of the form

$$Q_G(S) = \sum_{e \in (S \times S) \cap E} w(e)$$

where $S \subseteq V$. That is, $Q_G(S)$ returns the sum of weights of the edges in the subgraph induced by S.

The problems of reconstructing graphs using additive queries has been motivated by applications in bioinformatics [9, 28]. Assume that we have a set of labeled chemicals, and we are able to tell how many pairs react when mixing several of those chemicals together. We can represent the problem as a graph, where the chemicals are the vertices and two chemicals that react with each other are connected by an edge. The goal is to reconstruct this reaction graph using as few experiments as possible.

One concrete example for reconstructing a hidden graph is in genome sequencing. Obtaining the genome sequence is important for the study of organisms. To obtain the

sequence, one common approach is to obtain short *reads* from the genome sequence. These reads are assembled to *contigs*, which are contiguous fragments that cover the genome sequence with possible gaps. Given these contigs, the goal is to determine their relative place in the genome sequence. The process of ordering the contigs is done using the *multiplex PCR method*. This method, given a group of contigs determines the number of adjacent contigs in the original genome sequence. Assuming that the genome sequence is circular, the problem of ordering the contigs using the multiplex PCR method is equivalent to reconstructing a hidden Hamiltonian cycle using queries [9, 28].

The graph reconstructing problem has known significant progress lately [29, 28, 27, 9, 43, 21]. We start with exploring the unweighted case (in this case the answer to the query with the set of vertices S simply returns the number of edges in the subgraph induced by S). The information theoretic lower bound for the query complexity of reconstructing an unweighted graph is

$$\Omega\left(\frac{|E|\log\frac{|V|^2}{|E|}}{\log|E|}\right).$$

For the unweighted problem, Grebinski and Kucherov, [27], showed an optimal non-constructive non-adaptive algorithm for reconstructing a d-bounded degree graph. They also showed $O(|V|^2/\log|V|)$ polynomial algorithm for reconstructing an arbitrary graph. Later, Grebinski [29] extended the result by giving a non-constructive non-adaptive optimal algorithm for reconstructing d-degenerate graphs, that is, graphs that their edges can be changed to directed edges where the out-degree of each vertex is bounded by d. Finally, Choi and Han Kim [21], extended the later result by showing a non-constructive non-adaptive optimal algorithm for arbitrary graphs.

For the unweighted problem all algorithms that match the information theoretic lower bound are non-constructive. That is, they prove the existence of a set of queries such that the answers to these queries uniquely identify the hidden graph, however, it is unknown how to find such a set in deterministic polynomial time and moreover, given the answers to such set of queries, it is unknown how to reconstruct the hidden graph in polynomial time.

In our research, we solve this open problem [21]. We provide the first constructive adaptive polynomial time algorithm the matches the information theoretic lower bound

for the unweighted problem. We prove the following

Theorem 3. *[39] Let $G = (V, E, w)$ be a hidden graph with positive integer weights. Then, there exists an polynomial time adaptive algorithm that reconstructs G using*

$$O\left(\frac{k \log \frac{n^2}{k}}{\log k}\right)$$

additive queries. Here k is the sum of weights in G.

As for the weighted case where the weights are real numbers, the information theoretic lower bound for reconstructing a hidden weighted graph is

$$\Omega\left(\frac{|E| \log |V|}{\log |E|}\right).$$

Choi and Han Kim [21], showed a non-constructive non-adaptive optimal algorithm for reconstructing hidden graphs with many edges ($|E|$ must be at least $\log^{\alpha} |V|$ for a sufficiently large α) and weights that are bounded by $|V|^{-a}$ and $|V|^{b}$, where a and b are any constants.

In our research, we solve the open problems left in [21] by Choi and Kim. We show the following,

Theorem 4. *[11] There exists an optimal non-constructive non-adaptive algorithm for reconstructing any weighted hidden graph $G = (V, E, w)$ with any number of edges, where the weights on the edges are bounded between $|V|^{-a}$ and $|V|^{b}$ for any constants a and b.*

The next theorem seems somewhat unrelated, however, it implies that an optimal non-constructive non-adaptive algorithm exists even when removing the condition on the edges. That is, the edges can be arbitrary real numbers.

Theorem 5. *[12] For any prime p there exists a matrix $M \in \{0,1\}^{k \times n}$ such that*

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

every row is a tensor product of two equal size $(0,1)$-vectors and every m columns are linearly independent over $\mathbb{Z}_p$.

All the algorithms mentioned above for the weighted problem are non-constructive. The only polynomial time algorithm known is the trivial $O\left(|E|\log\frac{|V|^2}{|E|}\right)$ recursive search. To this end, we show the following:

Theorem 6. *[13] Let $G=(V,E,w)$ be any hidden graph, where the weights on the edges are arbitrary positive real numbers. There is a polynomial time algorithm that reconstruct the hidden graph in*

$$O\left(\frac{|E|\log|V|}{\log|E|}+|E|\log\log|E|\right)$$

queries.

The query complexity of the above algorithm matches the information theoretic lower bound when $|E|\leq|V|^{c/\log\log|V|}$, where c is any constant. Also, this is the first algorithm that beats the trivial

$$O\left(|E|\log\frac{|V|^2}{|E|}\right)$$

recursive search algorithm.

Below is a table that summarizes all known results.

Graph Type	Restriction	Lower bound	Tight upper bound	Polynomial time
Unweighted	constant degree d	$\Omega(d\|V\|)$	[27]	Ours [39]
	d-degenerate	$\Omega(d\|V\|)$	[29]	Ours [39]
	No restriction	$\Omega\left(\|E\|\log\frac{\|V\|^2}{\|E\|}/\log\|E\|\right)$	[21]	Ours [39]
Weighted	Bounded weights and many edges	$\Omega\left(\|E\|\log\frac{\|V\|^2}{\|E\|}/\log\|E\|\right)$	[21]	Ours [13]
	Bounded weights	$\Omega\left(\|E\|\log\frac{\|V\|^2}{\|E\|}/\log\|E\|\right)$	Ours [11]	Ours [13]
	No restriction	$\Omega\left(\|E\|\log\frac{\|V\|^2}{\|E\|}/\log\|E\|\right)$	Ours [12]	Ours [13]

Table 1.2: Known results for the problem of reconstructing hidden graphs.

1.3 Reconstructing Hidden Hypergraphs using Queries

In this section we explore the problem of reconstructing a hidden weighted Hypergraph using additive queries. Let $G=(V,E,w)$ be a weighted hidden hypergraph where $E\subset 2^V$,

$|e|$ is constant for all $e \in E$, $w : E \to \mathbb{R}$, and n is the number of vertices in V. Denote by m the size of E. Suppose that the set of vertices V is known and the set of edges E is unknown. Given a set of vertices $S \subseteq V$, an additive query, $Q_G(S)$, returns the sum of weights in the sub-hypergraph induced by S. That is,

$$Q_G(S) = \sum_{e \in E \cap 2^S} w(e).$$

Our goal is to exactly reconstruct the set of edges using additive queries.

For an unweighted hypergraph of rank d the information theoretic lower bound gives

$$\Omega\left(\frac{m \log \frac{\binom{n}{d}}{m}}{\log m}\right),$$

for the query complexity for any adaptive algorithm for this problem.

In this research we give a tight upper bound for all unweighted hypergraphs of constant rank. We show the following,

Theorem 7. *[14] There is a non-adaptive algorithm with query complexity*

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

for reconstructing any weighted hypergraphs over $\mathbb{R}$ of rank at most d, at most m edges and weights that are integers bounded by $w = poly(n^d/m)$.

Obviously, the algorithm is optimal even for weighted hypergraphs with integer weights $|w| = poly(n^d/m)$ where d is the rank of the hypergraph. This solves the open problems in [21, 22, 11, 12].

As for weighted hypergraphs with real number weights (non-integer and unbounded) we show the following,

Theorem 8. *[14] There is a non-adaptive algorithm for reconstructing weighted hypergraphs over $\mathbb{R}$ with m edges and constant rank that uses*

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

queries.

This algorithm is also optimal, as the information theoretic lower bound for the query complexity of reconstructing a weighted hypergraph of constant rank is

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

1.4 Noise

The combinatorial search problems we study are motivated by real-life problems. As such, it is important to study these problems in the presence of noise. In this section, we study some of the combinatorial search problems mentioned above with the presence of noise, that is, when some of the answers to the queries are incorrect or missing.

We start with coin weighing problem by formally defining the problem. 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 some of the answers received are incorrect (we call them *errors*), and 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.

Given a reconstructing algorithm, we say that it *tolerates* e incorrect answers (errors) and unknown answers (erasures) if the algorithm can reconstruct the hidden vector v correctly when the number of incorrect and erasures answers of the queries are bounded by e.

For the **coin**(n) weighing problem with noise, we show the following.

Theorem 9. *Let $v \in \{0,1\}^n$ be a hidden vector. There exists a non-adaptive reconstructing algorithm that runs in time $O(n^{3/2})$, tolerates $c(n/(\epsilon\log n))^{\frac{1-\epsilon}{4}}$ errors and erasures, for any constant $c < 1/2$, and reconstructs v using*

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

queries.

The proof of the above theorem relies on the properties of the fourier coefficients of function. Although the algorithm is 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. The proof of this theorem relies on tools from coding theory.

Theorem 10. *Let $v \in \{0,1\}^n$ be a hidden vector. There exists a non-adaptive polynomial time reconstructing algorithm that tolerates $\Omega(n^{1-\epsilon})$ errors and erasers and reconstructs v using*

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

queries.

Moreover, if the errors and erasures are consecutive, then, the algorithm works correctly even if their number is $\Omega(n/(\epsilon \log n))$.

An algorithm for the coin weighing problems with the presence of noise yields a u.d. signature codes for noisy channels. The existence of such codes was heavily studied in the literature [18, 25, 46, 19, 23]. In [18], Chang and Weldon first showed a technique to reconstruct u.d. codes for noisy channels. Using this technique, one can build a u.d. 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 that $O(n^{1-\epsilon})$. Here we use the notation from subsection 1.1.1, also $\tilde{Y}$ is the received codeword with noise, see Figure 1.2 below (note that unlike what is presented in Figure 1.2, in the literature the received codeword does not include erasures. Therefore, when talking about previous related work, always assume that the received work $\tilde{Y}$ 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 [46], Wilson showed a similar technique. As in the previous technique, using Wilson's technique, one can reconstruct a u.d. code 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

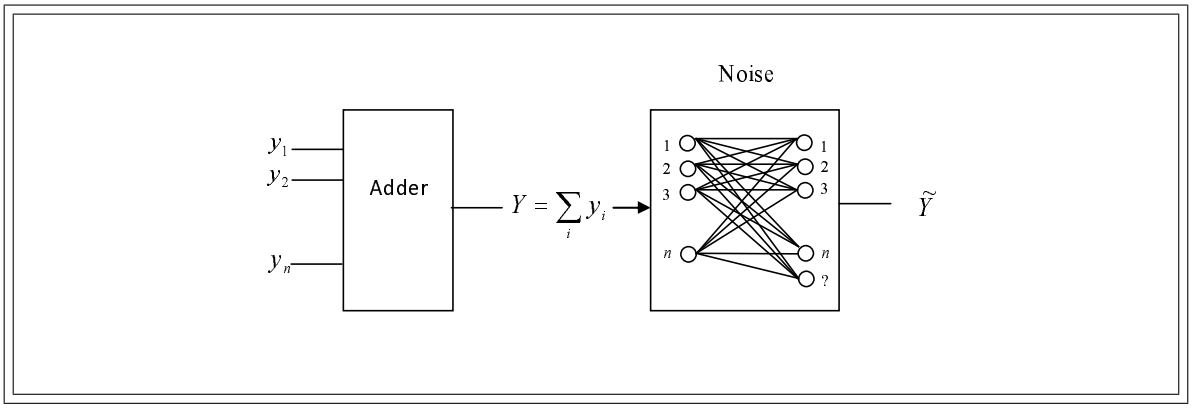

Figure 1.2: Multiple access adder channel with noise

$O(\frac{n^\epsilon}{\epsilon \log n})$ 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 [20, 19]. Finally, in [23], 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 if the error vector, $\tilde{Y} - Y$, has a bounded L_1 norm. Our theorem, Theorem 10, yields a u.d. 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.

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

Theorem 11. *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 $\mathbf{coin}_{\mathbb{R}}(n,m)$ weighing problem. This algorithm works correctly even if 11 percent of the answers received are erroneous. Moreover, the theorem yields a u.d. signature code for the partially active model where at most m user are active. The code is of length k and is decoded correctly even if the error vector is of Hamming weight smaller than $0.11k$.

Finally, we show a polynomial adaptive algorithm for the $\mathbf{coin}(n,m)$ problem. We show the following.

Theorem 12. *Let $v \in \{0,1\}^n$ be a hidden vector with at most m nonzero entries. There is a polynomial time adaptive algorithm that tolerates $O(m^{1-\epsilon})$ errors and erasures and reconstructs v using*

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

queries.

1.5 Outline of the Thesis

In Chapter 2 we present basic results that will be used in following chapters. In Chapter 3 we present all results for the problem of reconstructing a hidden vector. In Chapter 4 we present all results for reconstructing hidden graphs using additive queries. In Chapter 5 we show all results for the problem of reconstructing hidden weighted hypergraphs of constant rank. Finally, in Chapter 6 we present all results for the problem reconstructing vectors with the presence of noise.

Chapter 2

Preliminaries and Basic tools

In this chapter we present notation, preliminary results and some basic tools that will be used throughout this thesis.

2.1 Notation

Let $\mathbb{R}$ denote the set of real numbers. We denote by $\mathbb{R}^+$ the set of positive real numbers and by $\mathbb{R}_0^+$ the set $\mathbb{R}^+ \cup \{0\}$. Let r be a positive integer, we denote by $[r]$ the set $\{1, 2, \ldots, r\}$.

For a prime p and integers a and b we write $a =_p b$ for $a = b \mod p$. We will also allow $p = \infty$. In this case a and b can be any real numbers and $a =_\infty b$ will mean $a = b$ as real numbers.

For a t-vector w, we denote by w_i the ith entry of w. We denote by $|w|$ the t-vector where $|w|_i = |w_i|$ for all $i \in [t]$. For $w \in \{-1, 1\}^t$ we denote by $||w||$ the number of ones in w. For two vectors $w, q \in \{-1, 1\}^t$, we write $w \leq q$ if for every $i \in [t]$ we have $w_i \leq q_i$. We write $w < q$ if $w \leq q$ and $w \neq q$.

Let S_i be a set of real numbers where $i \in [t]$, we denote by $\prod_{i=1}^t S_i$ the Cartesian product of the sets, that is, the set of t-vectors $\{(w_1, w_2, \ldots, w_t) \,|\, \forall i : w_i \in S_i\}$.

For $S \subseteq [r]$ we define $x^S \in \{0, 1\}^r$ where $x_i^S = 1$ if and only if $i \in S$. The inverse operation is $S^x = \{i \mid x_i = 1\}$. We say that $x_1, \ldots, x_d \in \{0, 1\}^n$ are *pairwise disjoint* if for every $i \neq j$, we have $x_i * x_j = \mathbf{0}$ where $*$ is component-wise product of two vectors.

Let x be a vector or a matrix. We denote by $wt(x)$ the Hamming weight of x, that is,

the number of non-zero entries in x. We denote by $\psi(x)$ the sum of its entries.

2.2 Fourier Representation

In this section, we 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.

Next, we present results in probability theory that will be used in the following chapters.

2.3 Basic Probability

We first show the following,

Lemma 1. *Let $a \in \mathbb{Z}_p^n \backslash \{0^n\}$ and $M \in \mathbb{Z}_p^{n\times n} \backslash \{0^{n\times n}\}$. Then, for a uniformly randomly chosen vectors $x, y \in \{0,1\}^n$, we have*

$$\Pr_x[a^T x =_p 0] \leq 1/2 \quad \textit{and} \quad \Pr_{x,y}[x^T M y =_p 0] \leq 3/4.$$

Next we have,

Lemma 2. *[12] Let $a \in \mathbb{Z}_p^n \backslash \{0^n\}$ be a vector, where p is a prime number. Then, for a uniformly randomly chosen vector $x \in \{0,1\}^n$, we have*

$$\Pr_x[a^T x =_p 0] \leq \max\left(\frac{1}{wt(a)^\beta}, \frac{1}{p^{1/2}}\right),$$

where $\beta = \frac{1}{2+\log 3} = 0.278943\cdots$.

Proof. Let $S = \{a_i \,|\, i \in [n]\}$ and $\alpha = \frac{\log 3}{2+\log 3}$. We take two cases:

Case 1: The size of S is at most $wt(a)^\alpha$. Using the pigeon hole principle, there is an element $g \in \mathbb{Z}_p \backslash \{0\}$ that appears in a more than $wt(a)^{1-\alpha}$ times. Suppose w.l.o.g. that $a_1 = a_2 = \ldots = a_t = g$, where $t = \min(wt(a)^{1-\alpha}, p)$. For any fixed $x_{t+1}, x_{t+2}, \ldots, x_n \in \{0,1\}$ we have

$$a^T x =_p g(x_1 + x_2 + \ldots + x_t) + b_1,$$

where b_1 is a constant. Therefore, $a^T x = 0$ implies

$$x_1 + x_2 + \ldots + x_t =_p -b_1 g^{-1}.$$

Since $t \leq p$, all the solutions of $x_1 + x_2 + \ldots + x_t =_p -b_1 g^{-1}$ in $\{0,1\}^t$ have the same number of ones. Therefore, we have for $c = \sqrt{2/\pi} = 0.797885\cdots < 1$,

$$\begin{aligned} \Pr_x[a^T x =_p 0] &\leq \frac{\binom{t}{\lfloor t/2 \rfloor}}{2^t} \leq \frac{c}{\sqrt{t}} \\ &= \frac{c}{\min\left(wt(a)^{\frac{1-\alpha}{2}}, p^{1/2}\right)} \\ &\leq \max\left(\frac{1}{wt(a)^\beta}, \frac{1}{p^{1/2}}\right). \end{aligned}$$

Case 2: The size of S is at least $wt(a)^\alpha$. For a set of elements $Q = \{q_1, q_2, \ldots, q_r\} \subseteq \mathbb{Z}_p$ denote by

$$\psi(Q) = |\{(q_1 y_1 + q_2 y_2 + \cdots + q_r y_r) \mod p \,|\, y_1, \ldots, y_r \in \{0,1\}\}|,$$

and by $\mathcal{A}(Q)$ the set

$$\{(q_1 z_1 + q_2 z_2 + \cdots + q_r z_r) \bmod p \,|\, z_1, \ldots, z_r \in \{-1, 0, 1\}\}.$$

Since $|S| > wt(a)^\alpha$, we argue that there exists a set of entries $Q = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ such that $k \geq \log_3 wt(a)^\alpha$ and $\psi(Q) = 2^k$. We prove this claim by showing how to find such set of entries. The process of finding the entries is iterative. At every iteration j we have a set of entries $Q_j = \{a_{i_1}, a_{i_2}, \ldots, a_{i_j}\}$ of size j such that $\psi(Q_j) = 2^j$. It is easy to see that if $a_{i_{j+1}} \notin \mathcal{A}(Q_j)$ then $\psi(Q_j \cup \{a_{i_{j+1}}\}) = 2^{j+1}$. An element $a_{i_{j+1}} \in S$ can be added to Q_j as long as $|\mathcal{A}(Q_j)| \leq 3^j < |S|$. Therefore, we are able to find a set Q such that $|Q| \geq \log_3 |S|$ and $\psi(Q) = 2^{|Q|}$.

Now, let W denote the set $[n]\backslash\{i_1,\ldots,i_k\}$. For any fixed values for entries in W we have that $a^T x =_p a_{i_1}x_{i_1}+a_{i_2}x_{i_2}+\cdots+a_{i_k}x_{i_k}+b_2$, where b_2 is a constant. By the properties of Q, there is at most one $y\in\{0,1\}^k$ such that $a_{i_1}y_1+a_{i_2}y_2+\cdots+a_{i_k}y_k=-b_2$. Therefore,

$$\begin{aligned}\Pr_x[a^T x =_p 0] &\leq \frac{1}{2^k}\leq\frac{1}{2^{\log_3 wt(a)^\alpha}}\\ &= \frac{1}{wt(a)^{\alpha\log_3 2}}=\frac{1}{wt(a)^\beta}.\end{aligned}$$

□

2.4 Means

In this subsection we present some known inequalities that will help us analyze the complexity of some of the algorithm.

Let $r_1,r_2,\ldots,r_n$ be n real numbers. The *arithmetic mean* of $r_1,r_2,\ldots,r_n$ is

$$AM(r_1,r_2,\ldots,r_n)=\frac{r_1+r_2+\cdots+r_n}{n}.$$

The *geometric mean* is

$$GM(r_1,r_2,\ldots,r_n)=(r_1\cdot r_2\cdots r_n)^{\frac{1}{n}}.$$

The *harmonic mean* is

$$HM(r_1,r_2,\ldots,r_n)=n\left(\frac{1}{r_1}+\frac{1}{r_2}+\cdots+\frac{1}{r_n}\right)^{-1}.$$

When $r_i\geq 0$ for all $i\in[n]$ then, the means satisfy

$$HM(r_1,r_2,\ldots,r_n)\leq GM(r_1,r_2,\ldots,r_n)\leq AM(r_1,r_2,\ldots,r_n).$$

We now rely on the above fact for proving two useful lemmas.

Lemma 3. *Let $b_1,b_2,\ldots,b_t,s_1,s_2,\ldots,s_t$ be positive numbers. Let $B=b_1+b_2+\cdots+b_t$ and $S=s_1+s_2+\cdots+s_t$. Then, we have*

$$\prod_{i=1}^{t}\left(\frac{b_i}{s_i}+1\right)^{s_i}\leq\left(\frac{B}{S}+1\right)^S.$$

Proof. Define $r_{ij} = b_i/s_i + 1$ for all $i \in [t]$ and $j \in [s_i]$. Then, we have

$$\left(\prod_{i=1}^{t}\left(\frac{b_i}{s_i}+1\right)^{s_i}\right)^{1/S} = GM(r_{11},\ldots,r_{ts_t}) \leq AM(r_{11},\ldots,r_{ts_t}) = \left(\frac{B}{S}+1\right).$$

This implies the result. □

Lemma 4. *Let $k_1, k_2, \ldots, k_t$ be positive numbers. Let $K = k_1 + k_2 + \cdots + k_t$. Then we have*

$$\prod_{i=1}^{t}(k_i)^{k_i} \geq \left(\frac{K}{t}\right)^{K}.$$

Proof. Define $r_{ij} = k_i$ for all $i \in [t]$ and $j \in [k_i]$. Then we have

$$\left(\prod_{i=1}^{t}(k_i)^{k_i}\right)^{1/K} = GM(r_{11},\ldots,r_{tk_t}) \geq HM(r_{11},\ldots,r_{tk_t}) = \frac{K}{t}$$

This implies the result. □

2.5 Coding Theory Basics

In this section, we introduce the well known Hamming bound and some notation from coding theory we use in this thesis.

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. That is, the code $\mathcal{C}$ is a linear subspace of $\mathbb{F}^n$, its dimension is k and for every two different codewords $c_1, c_2 \in \mathcal{C}$ we have $wt(c_1 - c_2) \geq d$.

A *generating matrix* G for an $[n, k, d]$ linear code $\mathcal{C}$ over $\mathbb{F}$ is a $k \times n$ matrix over F. The rows of G form a basis for the linear subspace $\mathcal{C}$. That is, we have $c \in \mathcal{C}$ if and only if there is $q \in \mathbb{F}^k$ such that

$$c^T = q^T G.$$

A *parity check matrix* H for an $[n, k, d]$ linear code $\mathcal{C}$ over $\mathbb{F}$ is a $(n-k) \times n$ matrix over $\mathbb{F}$ such that for every $x \in \mathbb{F}^n$ we have $x \in \mathcal{C}$ if and only if

$$Hx = 0.$$

Since for every $x \in \mathbb{F}^n$ where $0 < wt(x) < d$ we have that $x \notin \mathcal{C}$, we get that $Hx \neq 0$. Therefore, every $d-1$ columns of H are linearly independent. In this thesis, we also call a parity check matrix $d-1$-independent matrix.

We now present the well known Hamming bound. We will use it in the sequel.

Lemma 5. *Let p be a prime number. For any $[n,k,d]$ linear code over $\mathbb{Z}_p$ we have*

$$p^k \leq \frac{p^n}{\sum_{i=0}^{t} \binom{n}{i}(p-1)^i}$$

where $t = \lfloor \frac{d-1}{2} \rfloor$.

Chapter 3

Reconstructing Hidden Vectors

In this chapter we present all results for the problem of reconstructing a hidden vector using additive queries. We present lower bounds, upper bounds and an almost tight polynomial time algorithm for the problem. Although we are mainly interested in solving the problem over the field of real numbers, we extend our discussion in the following two sections by giving a lower bound and a matching tight upper bound for solving the problem over finite fields as well.

In the following two section, we are interested only in the query complexity. Given these settings, we show some reductions or connections between different problems. The following lemma shows that the coin(n, m) weighing problem in the non-adaptive setting is equivalent to the problem of finding a $(0, 1)$-matrix M with minimal number of rows for which every $2m$ columns are linearly independent. That is, it is equivalent to finding a (0,1) parity check matrix with minimal number of rows for a linear code of minimal distance $2m + 1$.

Lemma 6. *There exists a non-adaptive algorithm that reconstructs a vector $v \in \mathbb{F}^n$ with at most m non-zeros (here $2m \leq n$) using k queries if and only if there exists a $(0, 1)$-matrix M of size $k \times n$ such that every $2m$ columns are linearly independent.*

Proof. Suppose there exists a non-adaptive algorithm for the problem. Let $(q_1, q_2, \ldots, q_k)$

be the set of queries that solves the problem. We argue that in the $(0,1)$-matrix

$$M = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_k \end{pmatrix},$$

every $2m$ columns are linearly independent. To see this, suppose on the contrary that there are $2m$ columns that are linearly dependent. Assume that

$$c_1 M_{i_1} + c_2 M_{i_2} + \ldots + c_{2m} M_{i_{2m}} = 0,$$

where M_i is the ith column of M and $c_1, \ldots, c_{2m}$ are constants that are not all zeros. Let v be the n-vector where the entry i_k equals c_k for all $k \in [m]$. Also, let u be the n-vector where the entry i_k equals $-c_k$ for all $k \in \{m+1, \ldots, 2m\}$. We have that $M(v-u) = 0$. Therefore, $Mv = Mu$. That is, the algorithm cannot distinguish between u and v since it receives the same vector of answers on the two. Contradiction.

On the other hand, let M be a $(0,1)$-matrix with k rows where every $2m$ columns are linearly independent. We argue that the rows of the matrix are queries that solve the problem of reconstructing a hidden vector with at most m non-zeros. For any vectors $u, v \in \mathbb{F}^n$ with at most m non-zeros, where $u \neq v$, we have that $0 < wt(u-v) \leq 2m$, and thus $M(v-u) \neq 0$. Therefore, $Mv \neq Mu$. That is, the suggested set of queries can distinguish between any two vectors u and v with at most m non-zero entries. □

In the next two section, we use the above lemma. In some cases (finite fields case), we show a lower bound and an upper bound on the number of rows of a $(0,1)$-matrix for which every $2m$ columns are linearly independent instead of showing a lower bound or an upper bound on the number of queries needed for solving the corresponding vector reconstructing problem.

Next, we have the following lemma that connects between the problem of reconstructing a vector in $(\mathbb{R}_0^+)^n$ and reconstructing a vector in $\mathbb{R}^n$.

Lemma 7. *A non-adaptive algorithm that solves the coin$^+(n, 2m)$ weighing problem also solves the coin(n, m) weighing problem.*

Proof. Let $(q_1, q_2, \ldots, q_k)$ be the set of queries that solves the $\text{coin}^+(n, 2m)$ problem. Let $M \in \{0,1\}^{k \times n}$ be the matrix for which the ith row equals q_i. We show that for any two different vectors $x, y \in \mathbb{R}^n$ such that $wt(x), wt(y) \leq m$ we have

$$Mx \neq My.$$

Assume on the contrary that this is not true. That is, $Mx = My$. Let $x^+ \in (\mathbb{R}_0^+)^n$ be the n-vector where $(x^+)_i = x_i$ if $x_i > 0$ and zero otherwise. Let $x^- \in (\mathbb{R}_0^+)^n$ be the vector where $(x^-)_i = -x_i$ if $x_i < 0$ and zero otherwise. Similarly, let $y^+ \in (\mathbb{R}_0^+)^n$ be the n-vector where $(y^+)_i = y_i$ if $y_i > 0$ and zero otherwise. Let $y^- \in (\mathbb{R}_0^+)^n$ be the vector where $(y^-)_i = -y_i$ if $y_i < 0$ and zero otherwise. We have

$$M(x^+ - x^-) = M(y^+ - y^-).$$

Therefore,

$$M(x^+ + y^-) = M(y^+ + x^-).$$

Since $x \neq y$ we have $(x^+ + y^-) \neq (y^+ + x^-)$. Also, $(x^+ + y^-), (y^+ + x^-) \in (\mathbb{R}_0^+)^n$ and $wt(x^+ + y^-), wt(y^+ + x^-) \leq 2m$. This is a contradiction, since M solves the $\text{coin}^+(n, 2m)$ problem. □

3.1 Lower Bounds

We start with the lower bound for the query complexity of reconstructing a hidden vector $v \in \{0,1\}^n$ with at most m non-zero entries.

Lemma 8. *Any algorithm (adaptive or non-adaptive) must ask at least*

$$\Omega\left(\frac{m \log \frac{n}{m}}{\log m}\right),$$

queries in order to reconstruct a hidden vector $v \in \{0,1\}^n$ with at most m non-zeros.

Proof. For every query, we have $m+1$ possible answers (which are: $0, 1, \ldots, m$). We also have $\sum_{i=0}^{m} \binom{n}{i}$ possible hidden vectors. Therefore, using a simple information theoretic argument, we must ask at least

$$\frac{\log \sum_{i=0}^{m} \binom{n}{i}}{\log m + 1} = \Omega\left(\frac{m \log \frac{n}{m}}{\log m}\right),$$

queries. □

Next, we present the lower bound of reconstructing a hidden vector $v \in \mathbb{R}^n$ with at most m non-zero entries.

Lemma 9. *Any algorithm (adaptive or non-adaptive) must ask at least*

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

queries in order to reconstruct a hidden vector $v \in \mathbb{R}^n$ with at most m non-zeros.

Proof. First, we argue that the algorithm must ask at least m queries. This is because the algorithm must be able to reconstruct hidden vectors with m non-zero independent entries. Also, From Lemma 8, the algorithm must ask at least $\frac{\log \sum_{i=0}^{m} \binom{n}{i}}{\log m+1}$ queries. Therefore, the algorithm must ask at least

$$max\left(\frac{\log \sum_{i=0}^{m} \binom{n}{i}}{\log m + 1}, m\right) = \Omega\left(\frac{m \log \frac{n}{m}}{\log m} + m\right) = \Omega\left(\frac{m \log n}{\log m}\right),$$

queries. □

Finally, we show the lower bound for the query complexity of the same problem over finite fields.

Theorem 13. *Let p be any prime number. A $(0,1)$-matrix $M \in \{0,1\}^{k \times n}$ such that every m columns in M are linearly independent over $\mathbb{Z}_p$ must have at least*

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

rows.

Proof. For $p < m$ and by the Hamming bound (Lemma 5) we have

$$p^k \geq \sum_{i=0}^{m/2} \binom{n}{i} (p-1)^i.$$

Therefore,

$$\begin{aligned} k &\geq \frac{\log \sum_{i=0}^{m/2} \binom{n}{i} (p-1)^i}{\log p} \\ &\geq \frac{\log(p-1)}{2 \log p} m + \frac{\log \binom{n}{m/2}}{\log p} = \Omega\left(m + \frac{m \log \frac{n}{m}}{\log p}\right). \end{aligned}$$

For $p > m$, notice that for every $v, u \in \{0, 1, \ldots, m\}^n$ of weight equal to $m/2$ we have $Mv \neq_p Mu$. Otherwise, $M(v-u) =_p 0^k$ and the columns that corresponds to the (at most m) entries that are not zero in $v - u$ are linearly dependent. Since for every $v \in \{0, 1, \ldots, m\}^n$ of weight at most $m/2$ we have $Mv \in \{0, 1, \ldots, m^2/2\}^k$ we must have

$$\left(\frac{m^2}{2} + 1\right)^k \geq \binom{n}{m/2}(m-1)^{m/2}.$$

Therefore,

$$k = \Omega\left(m + \frac{m \log \frac{n}{m}}{\log m}\right).$$

□

3.2 Tight Upper Bounds

In this section, we prove Theorem 1, that is, we present optimal non-constructive algorithm for the general problem of reconstructing a hidden non-bounded real numbers vector with at most m non-zero entries.

We start with the following lemmas. These lemmas follow from Lemma 2. Similar properties were proved for the field of real numbers in [32, 6, 8].

Lemma 10. *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 $1/2$.

Proof. Denote by M_i the ith column of M. Let $M_{i_1}, M_{i_2}, \ldots, M_{i_r}$ be any r linearly independent columns of M. Let M_j be any other column of the matrix, that is, $j \neq i_s$ for all $s \in [r]$. Then, there are unique constants $\alpha_1, \alpha_2, \ldots, \alpha_r$ such that

$$M_j =_p \alpha_1 M_{i_1} + \alpha_2 M_{i_2} + \ldots + \alpha_r M_{i_r}.$$

Therefore,

$$\alpha_1 M_{i_1} + \alpha_2 M_{i_2} + \ldots + \alpha_r M_{i_r} - M_j =_p 0.$$

Let a be the m-vector, where $a_j = -1$, $a_{i_s} = \alpha_{i_s}$ for all $s \in [r]$ and all other entries are zeros. Then,

$$\Pr[r(M') = r] \leq \Pr[a^T y =_p 0].$$

Now by Lemma 1 the result follows. □

Lemma 11. *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 $\max\left(\frac{1}{s^\beta}, \frac{1}{p^{1/2}}\right)$.

Proof. Using the same notation as in the proof of Lemma 10. Observe that $wt(a) \geq s$. Otherwise, there are s linearly dependent columns in M. Therefore, by Lemma 2,

$$\Pr[r(M') = r] \leq \Pr[a^T y =_p 0] \leq \max\left(\frac{1}{s^\beta}, \frac{1}{p^{1/2}}\right).$$

□

Now, after proving the above lemmas, we prove our main theorem,

Theorem 1. *For any prime p there exists a matrix $M \in \{0,1\}^{k \times n}$ such that*

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

and every m columns are linearly independent over $\mathbb{Z}_p$.

Proof. Let $t = m/\log^2 m$. We first prove the existence of a matrix $M^* \in \{0,1\}^{k_1 \times n}$, such that

$$k_1 = t + \log t + 2 \log \binom{n}{t},$$

where every t columns are linearly independent. We use probabilistic method. We randomly uniformly choose k_1 $(0,1)$-vectors of size n to be the rows of the matrix. Denote

by M_i the ith column of the matrix M^*. Now, let $M_{i_1}, M_{i_2}, \ldots, M_{i_t}$ be any t columns. Consider the matrix

$$M' = [M_{i_1}|M_{i_2}|\cdots|M_{i_t}]$$

and let $M'^{(j)}$ be the jth row of M' and $M'^{[j]}$ 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., $r(M'^{[j]}) = r(M'^{[j-1]}) + 1$. By Lemma 10,

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

Therefore, the probability that the rank of the matrix M' is smaller than t is bounded by

$$\begin{aligned}\Pr[X_1 + \cdots + X_{k_1} \leq t-1] &= \sum_{\substack{\xi_1+\cdots+\xi_{k_1}\leq t-1,\\ \xi_j\in\{0,1\}}} \Pr[X_1 = \xi_1, \ldots, X_{k_1} = \xi_{k_1}] \\ &\leq \frac{\sum_{i=0}^{t-1}\binom{k_1}{i}}{2^{k_1-t+1}} \leq t2^{t-1}\frac{\binom{k_1}{t}}{2^{k_1}} \\ &< \frac{\binom{n}{t}}{2\binom{n}{t}^2} = \frac{1}{2\binom{n}{t}}.\end{aligned}$$

Using union bound, the probability that there exists a set of t columns that are linearly dependent is less than $1/2$. This implies the existence of M^*.

Now, we have a matrix M^* that every $m/\log^2 m$ columns are linearly independent. We add k_2 uniformly randomly chosen rows to the matrix, where

$$k_2 = \frac{m\log q + \log\binom{n}{m}}{\log q - 1} \tag{3.1}$$

and $q = \min(t^\beta, p^{1/2})$. Again, given any m columns $M_{j_1}, M_{j_2}, \ldots, M_{j_m}$. By Lemma 11, the probability that the matrix

$$M'' = [M_{j_1}|M_{j_2}|\cdots|M_{j_m}]$$

has rank smaller than m is bounded by

$$\begin{aligned}\frac{\sum_{i=0}^{m-1}\binom{k_2}{i}}{q^{k_2-m+1}} &< \frac{2^{k_2}}{2q^{k_2-m}} \\ &= 2^{k_2(1-\log q)+m\log q-1} = \frac{1}{2\binom{n}{m}}.\end{aligned} \tag{3.2}$$

Therefore, the probability that there exists m columns of M that are linearly dependent is bounded by $1/2$. This, together with the fact that

$$k_1 + k_2 = O\left(m + \frac{m \log \frac{n}{m}}{\log \min(m, p)}\right),$$

implies the result. □

The following corollary solve the vector reconstructing problem over the field $\mathbb{R}$.

Corollary 1. *There exists a matrix $M \in \{0,1\}^{k \times n}$ where*

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

and for every two distinct vectors $x, y \in \mathbb{R}^n$ such that $wt(x) \leq m$ and $wt(y) \leq m$ we have $Mx \neq My$.

Proof. Choose a prime $2m < p$. By Theorem 1 there exists a $k \times n$ matrix M such that every $2m$ columns are linearly independent over $\mathbb{Z}_p$, and therefore, over $\mathbb{R}$. For any two vector $x, y \in \mathbb{R}^n$ such that $wt(x) \leq m$, $wt(y) \leq m$ and $x \neq y$ we have that $0 < wt(x - y) \leq 2m$. Therefore, $M(x - y) \neq 0^k$, and $Mx \neq My$. □

In Appendix A, we show a construction that reduces the number of random bits to $O(m^{1+\epsilon})$ in case $m = n^c$ for some constant c.

3.3 Polynomial Time Algorithm

In this section we present an almost tight polynomial time algorithm for the problem of reconstructing a hidden vector $v \in \mathbb{R}^n$ with at most m non-zero entries.

The algorithm we present is iterative. It uses the divide and conquer approach. The algorithm holds disjoint sets of entries of the hidden vector v. Each such set holds at least one non-zero entry. At the beginning of each iteration, the algorithm knows the sum of the entries in each set. It divides each set into two equal size set (up to ± 1) and its goal is to find the sum of the entries for each new set. The algorithm uses search matrices to obtain this goal (defined below). When using the search matrices, the algorithm makes some assumptions on the input of the subproblem. These assumptions are not always true, and

therefore, before advancing to the next iteration, the algorithm has an additional stage that checks its outcome and corrects it if needed.

3.3.1 Search Matrix

In this subsection we show,

Theorem 14. *Let $y \in (\mathbb{R}^+)^t$ be any vector such that $y_1 \geq y_2 \geq \ldots \geq y_t$. There exists a polynomial time non-adaptive algorithm for reconstructing a hidden vector $v \in \prod_{i=1}^{t}\{0, y_i\}$ that uses*

$$O\left(\frac{t}{\log t}\right)$$

queries of the form: $Q(w) = w^T v$ where $w \in \{0,1\}^t$.

Before proving this theorem, note that this problem is equivalent to finding in polynomial time a (0,1)-matrix M_t of size $O(t/\log t) \times t$ such that, $M_t v \neq M_t u$ for all $u, v \in \prod_{i=1}^{t}\{0, y_i\}$ where $u \neq v$ and given $M_t v$ one can reconstruct v in polynomial time. Such matrix M_t is called a *search matrix*.

We now show how to construct a search matrix for the problem. The matrix we present was first introduced in [10]. 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

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

Finally, define a partial order over F_ℓ in the following way: $f_{a,k_1} \leq f_{b,k_2}$ if and only if $a < b$ or ($a = b$ and $k_1 \geq k_2$).

The following are properties of F_ℓ and its functions [10].

- **P1.** $f_{a,k} \in \{0,1\}^{\{-1,1\}^\ell}$.
- **P2.** $|F_\ell| = \sum_i i\binom{\ell}{i} = \ell 2^{\ell-1}$.

- **P3.** The Fourier coefficient of χ_a in $f_{a,k}$ is 2^{-k}.
- **P4.** For any $a, b \in \{-1,1\}^\ell$, such that $a \neq b$ and $b \not> a$ the Fourier coefficient of χ_a in $f_{b,k}$ is equal to 0.

The following lemma can be derived immediately from **P3** and **P4**.

Lemma 12. *Let $a \in \{-1,1\}^\ell$ be vector. Let $I \subseteq [||a||]$ be any set. Let $A = \{(a,i)|i \in I\}$. Let $B = \{(b_1,i_1),\ldots,(b_{|B|},i_{|B|})\} \subseteq \{-1,1\}^\ell \times [\ell]$ such that for all j we have $i_j \in [||b_j||]$. If for all j we have $b_j \not\geq a$, then the Fourier coefficient of χ_a in*

$$f(x) = \sum_{(v,u)\in A\cup B} c_{u,v} f_{v,u}(x),$$

where $c_{u,v}$ are real numbers, is equal to $\sum_{(v,u)\in A} \frac{c_{v,u}}{2^u}$.

We now use F_ℓ to construct our search matrix. 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. This is not always true. In such case, we take the minimal ℓ such that $\ell 2^{\ell-1} > t$ create the matrix and take the first t columns): First, we label the rows of M_t with the elements of $\{-1,1\}^\ell$. Next, we label the columns with elements of F_ℓ in a descending order, that is, for every two indices i and j and their corresponding functions $f_{a,k}$ and $f_{b,s}$ (respectively), we have that if $i < j$ then $f_{b,s} \not\geq f_{a,k}$. Let $M_t[x, f_{a,k}] = f_{a,k}(x)$ be the search matrix that algorithm uses. 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

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

We will now give an algorithm showing that given $r = M_t v$ where $v \in \prod_i \{0, y_i\}$ one can exactly reconstruct v. Let $\sigma_t : [\ell 2^{\ell-1}] \to \{(a,k)|a \in \{-1,1\}^\ell$ and $k \in [||a||]\}$ defined as follows: $\sigma_t(j) = (a,k)$, where $f_{a,k}$ is the function corresponding to the jth column of the search matrix M_t.

Algorithm (Proof of Theorem 14). The algorithm **Reconstruct_Vector**, presented in Figure 3.1, is iterative. We run it for $start = 1$, and $r = M_t v$. The goal of the algorithm is to output a vector $h = v$. In each iteration the algorithm determines one entry of the hidden vector v. At iteration i, the entries $1, 2, \ldots, i-1$ are known to the algorithm (stored in $h_1, h_2, \ldots, h_{i-1}$), and the goal is to determine the ith entry. The vector r will be regarded as a function $r : \{-1, 1\}^\ell \to \mathbb{R}$. Therefore,

$$r(x) = \sum_{j=1}^{t} v_j f_{\sigma_t(j)}(x).$$

Algorithm **Reconstruct_Vector**(*start*, t, r, h, y)

1. **For** $i = start$ **to** t **do**
2. $\quad f(x) \leftarrow r(x) - \sum_{j=1}^{i-1} h_j f_{\sigma_t(j)}(x)$.
3. $\quad$ Let $(a, k) \leftarrow \sigma_t(i)$.
4. $\quad$ **if** $\hat{f}(a) < \frac{y_i}{2^k}$ **then**
5. $\quad\quad h_i = 0$.
6. $\quad$ **Else**
7. $\quad\quad h_i = y_i$.
8. $\quad$ **End if**.
9. **End for**.
10. **Return** h.

Comments:
1. The algorithm doesn't ask queries. All queries are asked and given to the algorithm as input in $r = M_t v$.
   The algorithm starts reconstructing from entry v_{start}, assuming that $h_1 = v_1, \ldots, h_{start-1} = v_{start-1}$.

Figure 3.1: Algorithm for reconstructing a vector in $\prod_i \{0, y_i\}$.

The algorithm first calculates the function

$$f(x) = r(x) - \sum_{j=1}^{i-1} h_j f_{\sigma_t(j)}(x).$$

Let (a,k) be equal to $\sigma_t(i)$. The algorithm calculates the Fourier coefficient of χ_a in f in polynomial time. If $\hat{f}(a) < \frac{y_i}{2^k}$ then, v_i equals 0. Otherwise, we have that $\hat{f}(a) \geq \frac{y_i}{2^k}$ and then, v_i equals y_i.

We now prove the correctness of the algorithm by induction.

Lemma 13. *(Algorithm's Correctness) Assuming correctness of the reconstruction of the first $i-1$ indices, that is, given that $h_1 = v_1, \ldots, h_{i-1} = v_{i-1}$, then, we have that $h_i = v_i$.*

Proof. Since $h_j = v_j$ for all $j \in [i-1]$, then we have

$$f(x) = r(x) - \sum_{j=1}^{i-1} h_j f_{\sigma_t(j)}(x) = r(x) - \sum_{j=1}^{i-1} v_j f_{\sigma_t(j)}(x) = \sum_{j=i}^{t} v_j f_{\sigma_t(j)}(x).$$

Let s be the index such that $\sigma_t(s) = (a, ||a||)$. Since for every $j > s$ and $(b,k') = \sigma_t(j)$, we have that $b \not\geq a$, by Lemma 12, the coefficient of χ_a in f is

$$\hat{f}(a) = \frac{v_i}{2^k} + \frac{v_{i+1}}{2^{k+1}} + \cdots + \frac{v_s}{2^{||a||}}.$$

Now, note that if $v_i = y_i$ then,

$$\hat{f}(a) = \frac{v_i}{2^k} + \frac{v_{i+1}}{2^{k+1}} + \cdots + \frac{v_s}{2^{||a||}} \geq \frac{y_i}{2^k}$$

and therefore $h_i = y_i$. Otherwise, $v_i = 0$, and then

$$\begin{aligned}
\hat{f}(a) &= \frac{v_i}{2^k} + \frac{v_{i+1}}{2^{k+1}} + \cdots + \frac{v_s}{2^{||a||}} \\
&= \frac{v_{i+1}}{2^{k+1}} + \frac{v_{i+2}}{2^{k+2}} + \cdots + \frac{v_s}{2^{||a||}} \\
&\leq \frac{y_{i+1}}{2^{k+1}} + \frac{y_{i+2}}{2^{k+2}} + \cdots + \frac{y_s}{2^{||a||}} \\
&\leq \frac{y_i}{2^{k+1}} + \frac{y_i}{2^{k+2}} + \cdots + \frac{y_i}{2^{||a||}} \\
&< \frac{y_i}{2^k},
\end{aligned}$$

and therefore, $h_i = 0$. □

Corollary 2. *Let $y \in (\mathbb{R}^+)^t$ be any positive real numbers. Then, there exists a polynomial time non-adaptive algorithm for reconstructing a hidden vector $v \in \prod_{i=1}^{t}\{0, y_i\}$ that asks*

$$O\left(\frac{t}{\log t}\right)$$

queries.

Proof. Let $P \in \{0,1\}^{t\times t}$ be a permutation matrix that sorts the vector y. Then, the matrix M_tP is a search matrix for the problem. Given M_tPv, we can reconstruct Pv and therefore reconstruct v. □

In the above proofs, we use a family of matrices introduced by Bshouty in [10]. This family is a part of larger family of matrices introduced by Bshouty. We present it below for convenience.

Lemma 14. *[10] 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.

3.3.2 The Main Algorithm

In this section we prove our main result.

Theorem 2. *There is a polynomial time algorithm for reconstructing a hidden vector $v \in (\mathbb{R}_0^+)^n$ with at most m non-zero entries that uses*

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

queries.

Proof. The algorithm we present is iterative. It searches for non-zero entries using the divide and conquer approach. At iteration τ, the algorithm holds at most m disjoint sets

$S_1, S_2, \ldots, S_q \subseteq [n]$ of indices. Each set S_i contains at least one index j for which $v_j \neq 0$. For a set $S \subseteq [n]$ we denote by $X(S)$ the sum

$$X(S) = \sum_{j \in S} v_j.$$

At the beginning of the iteration τ the algorithm knows $X(S_i)$ for all $i \in [q]$. As before, we may assume that for every $i \in [q-1]$ we have that $X(S_i) \geq X(S_{i+1})$. The algorithm divides each set S_i into two arbitrary equal size (up to ± 1) sets $S_{i,1}$ and $S_{i,2}$. Obviously, $X(S_i) = X(S_{i,1}) + X(S_{i,2})$. Now, the algorithm's goal is to find $X(S_{i,j})$ for all $i \in [q]$ and $j \in [2]$. Instead of asking q queries to achieve this, the algorithm uses the search matrix presented in the previous section in order to reconstruct the hidden vector,

$$u = (X(S_{1,1}), X(S_{1,2}), X(S_{2,1}), X(S_{2,2}), \ldots, X(S_{q,1}), X(S_{q,2})),$$

assuming that $u \in \prod_{i=1}^{2q} \{0, X(S_{\lceil i/2 \rceil})\}$. The algorithm simulates queries for u using the fact that

$$Q_u(x) = Q_v(y), \text{ where } y_k = \begin{cases} 1 & \exists i \in [q], \exists j \in [2] : k \in S_{i,j} \\ & \text{and } x_{2(i-1)+j} = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Here x is any $(0,1)$ vector. Denote by h the output of **Reconstruct_Vector** for reconstructing u. The assumption that $u \in \prod_{i=1}^{2q} \{0, X(S_{\lceil i/2 \rceil})\}$ is not always true. It may happen that for some index $j \in [q]$ we have $X(S_{j,1}) \neq 0$ and $X(S_{j,2}) \neq 0$, that is, $X(S_{j,1})$ and $X(S_{j,2})$ are not in $\{0, X(S_j)\}$. Such violation can occur at most m times throughout the running of the algorithm. In case where $X(S_{j,1}) \neq 0$ and $X(S_{j,2}) \neq 0$, we say that a *split* have occurred in the set S_j.

The next lemmas show how to find a split. Their proofs are similar to the proof of Lemma 13.

Lemma 15. *At iteration i of the running of **Reconstruct_Vector** for reconstructing*

$$u = (X(S_{1,1}), X(S_{1,2}), X(S_{2,1}), X(S_{2,2}), \ldots, X(S_{q,1}), X(S_{q,2})).$$

If

$$h_1 = u_1, h_2 = u_2, \ldots, h_{i-1} = u_{i-1},$$

and $u_i \in \{0, X(S_{\lceil i/2 \rceil})\}$, then, the algorithm reconstructs u_i correctly. That is, $h_i = u_i$.

Proof. Since $h_j = u_j$ for all $j \in [i-1]$, then we have

$$f(x) = r(x) - \sum_{j=1}^{i-1} h_j f_{\sigma_{2q}(j)}(x) = r(x) - \sum_{j=1}^{i-1} u_j f_{\sigma_{2q}(j)}(x) = \sum_{j=i}^{2q} u_j f_{\sigma_{2q}(j)}(x).$$

See definition of $f(x), r(x), \sigma$ in algorithm **Vector_Reconstruct**. Let $\sigma_{2q}(i)$ be (a,k). Let s be the index such that $\sigma_{2q}(s) = (a, ||a||)$. Since for every $j > s$ and $(b,k') = \sigma_{2q}(j)$, we have that $b \not\geq a$, by Lemma 12, the coefficient of χ_a in f is

$$\hat{f}(a) = \frac{u_i}{2^k} + \frac{u_{i+1}}{2^{k+1}} + \cdots + \frac{u_s}{2^{||a||}}.$$

Now, note that if $u_i = X(S_{\lceil i/2 \rceil})$ then,

$$\hat{f}(a) = \frac{u_i}{2^k} + \frac{u_{i+1}}{2^{k+1}} + \cdots + \frac{u_s}{2^{||a||}} \geq X(S_{\lceil i/2 \rceil})$$

and therefore $h_i = X(S_{\lceil i/2 \rceil})$. Otherwise, $u_i = 0$, and then

$$\begin{aligned}
\hat{f}(a) &= \frac{u_i}{2^k} + \frac{u_{i+1}}{2^{k+1}} + \cdots + \frac{u_s}{2^{||a||}} = \frac{u_{i+1}}{2^{k+1}} + \frac{u_{i+2}}{2^{k+2}} + \cdots + \frac{u_s}{2^{||a||}} \\
&\leq \frac{X(S_{\lceil (i+1)/2 \rceil})}{2^{k+1}} + \frac{X(S_{\lceil (i+2)/2 \rceil})}{2^{k+2}} + \cdots + \frac{X(S_{\lceil s/2 \rceil})}{2^{||a||}} \\
&\leq \frac{X(S_{\lceil i/2 \rceil})}{2^{k+1}} + \frac{X(S_{\lceil i/2 \rceil})}{2^{k+2}} + \cdots + \frac{X(S_{\lceil i/2 \rceil})}{2^{||a||}} < \frac{X(S_{\lceil i/2 \rceil})}{2^k},
\end{aligned}$$

and therefore, $h_i = 0$. □

Lemma 16. *Let h be the output of* ***Reconstruct_Vector*** *for reconstructing*

$$u = (X(S_{1,1}), X(S_{1,2}), X(S_{2,1}), X(S_{2,2}), \ldots, X(S_{q,1}), X(S_{q,2})).$$

Let S_j be the set with the minimal index in which a split has occurred. Then, for every $i \leq 2(j-1)$ we have that $h_i = u_i$. Moreover, either $h_{2j-1} = 0$ or $h_{2j} = 0$.

Proof. The first part of the lemma follows immediately from the proof of Lemma 15 above. As for the second part, we need to show that either $h_{2j-1} = 0$ or $h_{2j} = 0$. Since $h_\ell = u_\ell$ for all $\ell \in [2j-2]$, then at iteration $2j-1$ we have,

$$f(x) = \sum_{i=2j-1}^{2q} u_i f_{\sigma_{2q}(i)}(x).$$

See definition of $f(x), r(x), \sigma$ in algorithm **Vector_Reconstruct**. Let $\sigma_{2q}(2j-1)$ be (a_1, k_1) and $\sigma_{2q}(2j)$ be (a_2, k_2). Let s be the index such that $\sigma_{2q}(s) = (a_1, ||a_1||)$. Note that since a split occurred in S_j we have that $u_{2j-1} < X(S_j)$ and $u_{2j} < X(S_j)$. We divide into two cases:

- Case 1: $a_1 \neq a_2$.

  In this case, the coefficient of χ_{a_1} in f is

  $$\hat{f}(a_1) = \frac{u_{2j-1}}{2^{k_1}}.$$

  Since $u_{2j-1} < X(S_j)$, we get that $h_{2j-1} = 0$.

- Case 2: $a_1 = a_2$.

  First note that $k_2 = k_1 + 1$ in this case. Also, the coefficient of χ_{a_1} in f is

  $$\hat{f}(a_1) = \frac{u_{2j-1}}{2^{k_1}} + \frac{u_{2j}}{2^{k_2}} + \cdots + \frac{u_s}{2^{||a_1||}}.$$

  Suppose on the contrary that neither $h_{2j-1} = 0$ nor $h_{2j} = 0$. This means

  $$\hat{f}(a_1) = \frac{u_{2j-1}}{2^{k_1}} + \frac{u_{2j}}{2^{k_2}} + \cdots + \frac{u_s}{2^{||a_1||}} > \frac{X(S_j)}{2^{k_1}} + \frac{X(S_j)}{2^{k_2}} = \frac{3X(S_j)}{2^{k_2}}.$$

  Note that $u_{2j-1} + u_{2j} = X(S_j)$. That is, $\hat{f}(a_1)$ is strictly less than $\frac{3X(S_j)}{2^{k_2}}$. Contradiction.

□

We now show how subroutine **Fix_Output** presented below uses the lemmas above to find the splits and how it changes h as if no split had occurred. Assume, for the sake of simplicity, that $2q = \ell 2^{\ell-1}$ for some integer ℓ (in the previous section we showed how to create a search matrix for the case in which the number of columns is not of the form $\ell 2^{\ell-1}$). For $a \in \{-1, 1\}^\ell$, let B_a denote the set of all indices j such that $\sigma_{2q}(j) = (a, k)$ for some k. Also let H_0 denote the set of all indices j for which $h_j = 0$. Given a set $S \subseteq [n]$ we denote by 1_S the (0,1)-vector of size n where the ith entry equals one if and only if $i \in S$. For an integer j, denote $\overline{j} = j - 1$ if j is even and $\overline{j} = j + 1$, otherwise. Let

Subroutine **Fix_Output**

1. Set $z \leftarrow z_1$, $i \leftarrow 1$.
2. While $(Q_u(1_{H_0 \cap B_z}) = 0$ and $i \neq 2^\ell + 1)$ do $i \leftarrow i + 1$, $z \leftarrow z_i$.
3. If $i = 2^\ell + 1$, then halt.
4. Using a binary search, find the minimal index $j \in H_0 \cap B_z$ for which $u_j \neq 0$.
5. Ask a query to find u_j (that is, $Q_u(1_{\{j\}})$).

   Update $h_{\bar{j}} = X(S_{\lceil j/2 \rceil}) - u_j$ and $h_j = u_j$.
6. Use the algorithm **Reconstruct_Vector** with the new values $h_1, \ldots, h_{\max(j,\bar{j})}$ and reconstruct h starting from entry $\max(j, \bar{j}) + 1$.
7. Update H_0 using the new h and goto 2.

Figure 3.2: Subroutine - Fix_Output

$z_1 = 1^\ell$, $z_2 = 1^{\ell-1} \cdot -1$, $\ldots$, $z_{2^\ell} = -1^\ell$ (these are the vectors we used to label the rows of the search matrix, see Subsection 3.3.1). The subroutine performs the following steps:

The idea of the subroutine is the following: For each B_z where $z \in \{z_1, \ldots, z_{2^{\ell-1}}\}$, the subroutine searches for errors caused by a split (note that $B_{z_1}, B_{z_2}, \ldots, B_{z_{2^{\ell-1}}}$ are disjoint and that $B_{z_1} \cup B_{z_2} \cup \cdots \cup B_{z_{2^{\ell-1}}} = [2q]$). Lemma 16 implies that if $Q_u(1_{H_0 \cap B_z}) = 0$ then the algorithm has reconstructed the entries that correspond to the indices B_z correctly (note that the order of searching for errors is important, we must first search for errors in B_{z_1} then in B_{z_2}, etc.). On the other hand, if $Q_u(1_{H_0 \cap B_z}) \neq 0$, then there is an index $j \in B_z \cap H_0$ such that $u_j \neq 0$. Let $j \in B_z \cap H_0$ be the minimal index for which $u_j \neq 0$, Lemma 15 and Lemma 16 imply that a split have occurred in $S_{\lceil j/2 \rceil}$. Line 5 corrects the two entries in h that were effected by this split. In line 6, **Reconstruct_Vector** retraces its steps and cancel errors in following entries that were possibly caused by the erroneous values of h_j and $h_{\bar{j}}$. Now, after reconstructing with the correct h_j and $h_{\bar{j}}$ (again by Lemma 15 and Lemma 16) the next error is guaranteed to occur in the entries that correspond to the next split.

After running **Fix_Output** and finding $X(S_{i,j})$ for all $i \in [q]$ and $j \in [2]$, the algorithm throws out all sets $S_{i,j}$ such that $X(S_{i,j}) = 0$ and advances to iteration $\tau + 1$.

As for the complexity analysis, the algorithm runs $\log n$ iterations. At each iteration τ, let β_τ denote the number of splits in this iterations. Let q_τ denote the number of sets at the beginning of the iteration. At each iteration, the algorithm uses search matrices. The complexity of this phase is $O(2q_\tau / \log 2q_\tau)$. Next, the algorithm corrects h. In line 2 of **Fix_Output**, the algorithm asks at most $O(2q_\tau / \log 2q_\tau) + \beta_\tau$ queries. Line 4 asks at most $O(\beta_\tau \log\log 2q_\tau)$ queries and line 5 asks β_τ queries. It is easy to see that the algorithm do not need to ask queries in line 6. Therefore, using the fact that $q_\tau \leq m$ for every $\tau \in [\log n]$, the total query complexity is

$$\sum_{\tau=1}^{\log n} O\left(\frac{m}{\log m}\right) + O(\beta_\tau \log\log m).$$

Since $\sum_{\tau=1}^{\log n} \beta_\tau = m$, we get the result. □

Chapter 4

Reconstructing Hidden Graphs

In this chapter we present all results for the problem of reconstructing a hidden graph using additive queries. We present lower bounds, upper bounds and polynomial time algorithms for the problem.

We start by presenting an algebraic view of the problem.

4.1 Algebraic View of the Problem

Let $G = (V, E, w)$ be a undirected weighted graph where $V = \{v_1, v_2, \ldots, v_n\}$, $E \subseteq V \times V$ and $w : E \to \mathbb{R}^+$. Let $A_G = (a_{ij}) \in \mathbb{R}^{n \times n}$ be its adjacency matrix, that is, a_{ij} equals $w((i,j))$ if $(i,j) \in E$ and equals zero otherwise. Given a set of vertices $V' \subseteq V$ define the vector a where a_i equals "1" if $v_i \in V'$ and "0" otherwise. Then, we have

$$Q(V') = \frac{a^T A_G\, a}{2}.$$

Since A_G is symmetric, Grebinski and Kucherov, [29, 27], show that one can turn our oracle to the assignment $f_A(x,y) = x^T A y$ in 5 queries. Let $z = x * y = (x_i y_i) \in \{0,1\}^n$ and $x_1 = x - z$, $y_1 = y - z$. Since A_G is symmetric, we have

$$x^T A_G\, y = \frac{x^T A_G\, x}{2} + \frac{y^T A_G\, y}{2} + \frac{(x_1 + y_1)^T A_G (x_1 + y_1)}{2} - x_1^T A_G\, x_1 - y_1^T A_G\, y_1.$$

Thus, the problem of reconstructing a graph G using additive queries is equivalent to reconstructing its adjacency matrix A_G using queries of the form

$$f(x,y) = x^T A_G\, y,$$

where $x, y \in \{0,1\}^n$.

4.2 Lower Bounds

In this section we present the lower bounds for the problem with various settings.

We start with reconstructing unweighted graphs.

Lemma 17. *Any algorithm (adaptive or non-adaptive) must ask at least*

$$\Omega\left(\frac{m \log \frac{n^2}{m}}{\log m}\right)$$

queries in order to reconstruct an unweighted graph with n vertices and at most m edges.

Proof. For every query, we have $m+1$ possible answers (which are: $0, 1, \ldots, m$). We also have $\sum_{i=0}^{m} \binom{\binom{n}{2}}{i}$ possible hidden graphs. Therefore, using a simple information theoretic argument, we must ask at least

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

queries. □

Next, we have weighted graphs.

Lemma 18. *Any algorithm (adaptive or non-adaptive) must ask at least*

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

queries in order to reconstruct a weighted graph (with real number weights) with n vertices and at most m edges.

Proof. First, we argue that the algorithm must ask at least m queries. This is because the algorithm must be able to reconstruct the weights of m independent edges. Also, From Lemma 17, the algorithm must ask at least $\frac{\log \sum_{i=0}^{m} \binom{\binom{n}{2}}{i}}{\log m+1}$ queries. Therefore, the algorithm must ask at least

$$max\left(\frac{\log \sum_{i=0}^{m} \binom{\binom{n}{2}}{i}}{\log(m+1)}, m\right) = \Omega\left(\frac{m \log \frac{n^2}{m}}{\log m} + m\right) = \Omega\left(\frac{m \log n}{\log m}\right),$$

queries. □

4.3 Upper Bounds

In this section we prove Theorem 5. That is, we prove the existence of an m-independent column (over $\mathbb{Z}_p$) $t \times n$ $(0,1)$-matrix that its rows are tensor product of two $(0,1)$-vectors in $\{0,1\}^{\sqrt{n}}$ and

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

We also show how this result implies the existence of an optimal non-constructive algorithm for the problem of reconstructing weighted graphs using additive queries.

We start with the following theorem.

Theorem 15. *Let $p < n^\gamma$ be a prime number for some constant $\gamma > 1$. There exists a set $S = \{(x_1, y_1), (x_2, y_2) \ , \ldots, (x_k, y_k)\}$, where $x_i, y_i \in \{0,1\}^n$ and*

$$k = O\left(m + \frac{m \log \frac{n^2}{m}}{\log \min(m,p)}\right),$$

such that for any matrix $A \in \mathbb{Z}_p^{n \times n} \backslash \{0^{n \times n}\}$ with $wt(A) \leq m$, there exists an i such that $x_i^T A y_i \neq_p 0$.

Proof. First notice that when m is constant then

$$O\left(m + \frac{m \log \frac{n^2}{m}}{\log \min(m,p)}\right) = O\left(m \log \frac{n^2}{m}\right),$$

and the result follows immediately from Lemma 1. Therefore, we may assume that $m = \omega(1)$. Note also that we may assume that $m < n^2/2$. Otherwise, we can just take all the n^2 pairs (e_i, e_j), where $\{e_i\}_{i \in [n]}$ is the standard basis.

We divide the set of matrices,

$$\mathcal{A} = \{A \mid A \in \mathbb{Z}_p^{n \times n} \backslash \{0^{n \times n}\} \text{ and } wt(A) \leq m\},$$

into three (non-disjoint) sets:

- $\mathcal{A}_1$: The set of all non-zero matrices $A \in \mathbb{Z}_p^{n \times n}$ such that $wt(A) \leq m / \log m$.
- $\mathcal{A}_2$: The set of all non-zero matrices $A \in \mathbb{Z}_p^{n \times n}$ such that $m \geq wt(A) > m / \log m$ and there are at least $\sqrt{\frac{m}{\log m}}$ non-zero rows.

- $\mathcal{A}_3$: The set of all non-zero matrices $A \in \mathbb{Z}_p^{n\times n}$ such that $m \geq wt(A) > m/\log m$ and there are at least $\sqrt{\frac{m}{\log m}}$ non-zero columns.

Note that for any matrix A of weight $wt(A) > d = m/\log m$, either A has more than $\sqrt{d}$ non-zero rows or it has more than $\sqrt{d}$ non-zero columns. Therefore,

$$\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3.$$

Using the probabilistic method, we give three sets S_1, S_2 and S_3 of vector pairs, such that for every $j = 1, 2, 3$ and $A \in \mathcal{A}_j$ there exists a pair of vectors $(x, y) \in S_j$ such that $x^T Ay \neq_p 0$ and

$$|S_1| + |S_2| + |S_3| = O\left(m + \frac{m \log \frac{n^2}{m}}{\log \min(m, p)}\right).$$

Case 1: $A \in \mathcal{A}_1$. This case follows immediately from Lemma 1. Randomly choose $O(m \log n/\log m)$ vector pairs (x_i, y_i). The probability that there exists a matrix $A \in \mathcal{A}_1$ such that for all i we have $x_i^T Ay_i = 0$ is

$$\Pr\left[\exists A \in \mathcal{A}_1, \forall i : x_i^T Ay_i = 0\right] \leq |\mathcal{A}_1| \left(\frac{3}{4}\right)^{O(m \log n/\log m)}.$$

Using the fact that $|\mathcal{A}_1| < n^{O(m/\log m)}$, we get that this probability is less than 1. This implies the result.

Case 2: $A \in \mathcal{A}_2$. We start by proving the following two lemmas.

Lemma 19. *Let $U \subset \mathbb{Z}_p^n$ be the set of all non-zero vectors with weight smaller than $m^{3/4}$. For any constant $C > (1+\gamma)16/\log e$ and*

$$\begin{aligned} k_2 = C\left(m + \frac{m \log \frac{n^2}{m}}{\log \min(m, p)}\right) &> C\left(m + \frac{m \log \frac{n^2}{m}}{\log m}\right) \\ &= C\left(\frac{m \log n^2}{\log m}\right), \end{aligned} \tag{4.1}$$

there exists a multiset of (0,1)-vectors $Y = \{y_1, y_2, \ldots, y_{k_2}\}$ such that for every $u \in U$ the size of the multiset $Y_u = \{i \,|\, u^T y_i \neq_p 0\}$ is at least $k_2/4$.

Proof. By Lemma 1 for a randomly chosen vector $y \in \{0,1\}^n$ and any $u \in U$ we have $\Pr[u^T y =_p 0] \leq 1/2$. Therefore, if we randomly uniformly choose the vectors of Y, then the expected size of Y_u is greater than $k_2/2$ for any $u \in U$. Using Chernoff bound we have that $\Pr[|Y_u| < k_2/4] \leq e^{-\frac{k_2}{16}}$. Therefore, the probability that there exists $u \in U$ such that $|Y_u| < k_2/4$ is

$$\begin{aligned}
\Pr[\exists u \in U : |Y_u| < k_2/4] &\leq \frac{|U|}{e^{\frac{C}{16}\left(\frac{m \log n^2}{\log m}\right)}} \\
&\leq \frac{\sum_{i=0}^{m^{3/4}} \binom{n}{i}(p-1)^i}{n^{\frac{C \log e}{8}\left(\frac{m}{\log m}\right)}} \\
&\leq \frac{n^{m^{3/4}} n^{\gamma m^{3/4}}}{n^{\frac{C \log e}{8}\left(\frac{m}{\log m}\right)}} \\
&\leq n^{(1+\gamma)\left(m^{3/4} - 2\frac{m}{\log m}\right)} \\
&< 1.
\end{aligned}$$

This implies the result. □

Note that the constant C will be determined later in the proof. Now for the next lemma, define for non-negative integer r, $\iota(r) = \min(r, p)$ if $r > 0$ and $\iota(0) = 1$.

Lemma 20. *Let $m_1, m_2, \ldots, m_{k_2}$ be integers in $[m] \cup \{0\}$ such that $m_1 + m_2 + \cdots + m_{k_2} = \ell \geq k_2$. Then $\prod_{i=0}^{k_2} \iota(m_i) \geq \min(m, p)^{\lfloor (\ell - k_2)/(m-1) \rfloor}$.*

Proof. We first proof that when $1 < m_1 \leq m_2 < m$ then

$$\iota(m_1 - 1)\iota(m_2 + 1) \leq \iota(m_1)\iota(m_2). \tag{4.2}$$

We have four cases: When $p \leq m_1 - 1$ then (4.2) gives $p^2 \leq p^2$. When $p = m_1$ then (4.2) gives $(p-1)p \leq p^2$. When $m_1 < p \leq m_2$ then (4.2) gives $(m_1 - 1)p \leq m_1 p$. When $p \geq m_2 + 1$ then (4.2) gives $(m_1 - 1)(m_2 + 1) < m_1 m_2$. In all cases the inequality is true.

Also when $m_1 = 0$ and $1 < m_2 < m$ then $\iota(m_1 + 1)\iota(m_2 - 1) = \min(m_2 - 1, p) \leq \min(m_2, p) = \iota(m_1)\iota(m_2)$. Therefore the minimal value of $\iota(m_1)\iota(m_2)\cdots\iota(m_t)$ is obtained when for every $0 < i < j \leq k_2$ we either have $m_i \in \{1, m\}$ or $m_j \in \{1, m\}$. This is equivalent to: all $m_i \in \{1, m\}$ except at most one. This implies that at least $\lfloor (\ell - k_2)/(m-1) \rfloor$ of the m_is are equal to m. □

Now let U be the set of vectors defined in Lemma 19. Let $A \in \mathcal{A}_2$. Since $wt(A) \leq m$ there are at most $m^{1/4}$ rows in A with weight greater than $m^{3/4}$. Therefore, there are at least $q = \sqrt{m/\log m} - m^{1/4}$ rows in A that are in U (recall that $m = \omega(1)$, thus, $q = \omega(1)$). Let A_U be $q \times n$ matrix that its rows are any q rows in A that are in U. Let $Y = \{y_1, y_2, \ldots, y_{k_2}\}$ be the set we proved its existence in Lemma 19 (see (4.1)). Note that $\sum_i wt(A_U y_i) \geq \frac{qk_2}{4}$. Since $wt(A_U y_i) \leq q$ for all $i \in [k_2]$, by Lemma 20 we have

$$\begin{aligned}
\prod_i \iota(wt(A_U y_i)) &\geq (\min(q,p))^{\lfloor \frac{\frac{qk_2}{4} - k_2}{q-1} \rfloor} \\
&\geq (\min(q,p))^{c_1 k_2} \\
&\geq (\min(m,p))^{c_2 k_2},
\end{aligned}$$

where c_1 and c_2 are constants. If we randomly choose $x_1, x_2, \ldots, x_{k_2}$ then by Lemma 2, we have

$$\begin{aligned}
\Pr[\forall i \in [k_2] : x_i^T A y_i =_p 0] &\leq \prod_i \frac{1}{\min((wt(Ay_i))^\beta, p^{1/2})} \\
&\leq \prod_i \frac{1}{\iota(wt(Ay_i))^\beta} \\
&\leq \prod_i \frac{1}{\iota(wt(A_U y_i))^\beta} \\
&= \left(\frac{1}{\prod_i \iota(wt(A_U y_i))} \right)^\beta \\
&\leq \frac{1}{(\min(m,p))^{\beta c_2 k_2}} \\
&= (\min(m,p))^{-c_3 k_2},
\end{aligned}$$

where c_3 is a constant. Therefore, the probability that there exists a matrix $A \in \mathcal{A}_2$ such

that for all x_i, y_i we have $x_i^T A y_i =_p 0$ is

$$\begin{aligned}\Pr[\exists A \in \mathcal{A}_2, \forall i \in [k_2] : & x_i^T A y_i =_p 0] \\ & \leq \frac{|\mathcal{A}_2|}{(\min(m,p))^{c_3 k_2}} \\ & \leq \frac{\binom{n^2}{m} p^m}{(\min(m,p))^{c_3 k_2}} \\ & \leq \frac{\left(\frac{n^2}{m}\right)^m (ep)^m}{\left(\frac{n^2}{m}\right)^{c_3 C m} \min(m,p)^{c_3 C m}}.\end{aligned}$$

For $p < m$, since $m < n^2/2$, the above is less than 1 for $c_3 C > 3$. For $p \geq m$ we get

$$\begin{aligned}\frac{\left(\frac{n^2}{m}\right)^m (ep)^m}{\left(\frac{n^2}{m}\right)^{c_3 C m} \min(m,p)^{c_3 C m}} &= \frac{\left(\frac{n^2}{m}\right)^m (ep)^m}{n^{2 c_3 C m}} \\ &\leq \frac{n^{(2+2\gamma)m}}{n^{c_3 C m}} < 1,\end{aligned}$$

for $C > (2 + 2\gamma)/c_3$. Thus, the result follows.

Case 3: $A \in \mathcal{A}_3$. Similar to Case 2. This implies the result. □

Now after proving the above theorem, Theorem 5 follows immediately.

Theorem 5. *There exists a matrix $M \in \{0,1\}^{k \times n}$ where*

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

every row of M is a tensor product of two vectors $x, y \in \{0,1\}^{\sqrt{n}}$ and every m columns of M are linearly independent over $\mathbb{Z}_p$.

In particular the same matrix is $(0,1)$-matrix with m-independent columns over any field of characteristic p and over the real field $\mathbb{R}$.

Proof. Assume n is a perfect square. Let $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$ be the set we found in Theorem 15 with vectors in $\{0,1\}^{\sqrt{n}}$. Define the matrix M where the ith row is $x_i \otimes y_i$. We argue that every m columns of M are linearly independent over $\mathbb{Z}_p$, and therefore they are also linearly independent over $\mathbb{R}$. Suppose on the contrary that

there is a set of columns $M_{i_1}, M_{i_2}, \ldots, M_{i_m}$ that are linearly dependent. Then, there are constants $\alpha_1, \ldots, \alpha_m$ that are not all equal to 0 such that

$$\alpha_1 M_{i_1} + \alpha_2 M_{i_2} + \cdots + \alpha_m M_{i_m} = 0^k.$$

Define the following matrix $A \in \mathbb{Z}_p^{\sqrt{n}\times\sqrt{n}}$: For every column's index i_j let the entry (u, v) of the matrix be equal to α_j where $u = \lfloor (i_j - 1)/\sqrt{n} \rfloor + 1$ and $v = (i_j - 1 \mod \sqrt{n}) + 1$. All other entries are zero. It is easy to see that

$$x_i^T A y_i$$

equals the ith entry of the vector $\alpha_1 M_{i_1} + \alpha_2 M_{i_2} + \cdots + \alpha_m M_{i_m}$. Therefore we get that

$$x_i^T A y_i = 0,$$

for all $i \in [k]$. Since $A \neq 0^{\sqrt{n}\times\sqrt{n}}$ and $wt(A) \leq m$ we get a contradiction. □

Corollary 3. *There exists a set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$, where $x_i, y_i \in \{0,1\}^n$ and*

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

such that for any matrix $A \in \mathbb{R}^{n\times n}$, where $wt(A) \leq m$ and $A \neq 0^{n\times n}$, there exists an i such that $x_i^T A y_i \neq 0$.

Proof. Choose a prime $m < p < 2m$. We argue that the set S found in Theorem 15 is the desired set. Let A be a matrix, let $A^{(i)}$ denote the ith row. Define the n^2-vector

$$A^v = [A^{(1)}|A^{(2)}|\cdots|A^{(n)}].$$

Then, for any $x, y \in \{0,1\}^n$ we have

$$x^T A y = (x \otimes y)^T A^v.$$

Define the matrix M where the ith row is $x_i \otimes y_i$. In the previous corollary we showed that every m columns of M are linearly independent over $\mathbb{R}$. Now, suppose that there exists a matrix A such that $wt(A) \leq m$ and for all $i \in [k]$ we have

$$x_i^T A y_i = 0.$$

Since $x^T Ay = (x \otimes y)^T A^v$ and $x_i^T Ay_i = 0$ for all $i \in [k]$ we get that

$$MA^v = 0^k.$$

This is a contradiction since $wt(A^v) = wt(A) \leq m$ and every m columns of M are linearly independent. □

We now prove

Corollary 4. *There exists a non-adaptive algorithm that uses*

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

additive queries and reconstruct any weighted hidden graph with at most m edges.

Proof. From Corollary 3 it follows that there exists a set

$$S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\},$$

where $x_i, y_i \in \{0,1\}^n$ and

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

where for any matrix $A \in \mathbb{R}^{n \times n}$ such that $wt(A) \leq 4m$ and $A \neq 0^{n \times n}$, there exists an i such that $x_i^T Ay_i \neq 0$. Now we use (x_i, y_i) to find $z_i = x_i^T A_G y_i$. We claim that the answers $(z_i)_i$ uniquely determines A_G. Otherwise, there are two weighted graphs $G \neq G'$ with at most m edges such that for all i, $x_i^T A_G y_i = x_i^T A_{G'} y_i$. This implies that for every i, $x_i^T (A_G - A_{G'}) y_i = 0$. Since $1 \leq wt(A_G - A_{G'}) \leq 4m$, we get a contradiction. □

4.4 Polynomial Time Algorithms

In this section we present two polynomial time algorithms for reconstructing a hidden graph using additive queries. More precisely, we prove Theorem 3 and Theorem 6. The two algorithms we present share common structure. They search for non-zero entries in the adjacency matrix using the divide and conquer approach and repeatedly use vector reconstructing algorithms to minimize the query complexity.

We start with reconstructing a hidden graph with integer weights.

4.4.1 Integer Weights

In this subsection, we present the first algorithm. That is, we prove Theorem 3.

We start by presenting a non-adaptive algorithm for reconstructing a matrix given an upper bound on the value of every entry of the matrix.

Theorem 16. *Let $A = (a_{ij}) \in \mathbb{N}_0^{n\times n}$ be a hidden matrix. Let $B = (b_{ij}) \in \mathbb{N}_0^{n\times n}$ be any matrix such that, $a_{ij} \leq b_{ij}$ for every $i, j \in [n]$. Let $m = \psi(B)$, that is, m equals the sum of entries in B. Then, there exists a polynomial time non-adaptive algorithm that given B it can reconstruct A using*

$$O\left(\min\left\{\frac{wt(B)\log\frac{m}{wt(B)}}{\log wt(B)}, wt(B)\right\}\right)$$

queries of the form $f_A(x, y) = x^T Ay$, where $x, y \in \{0, 1\}^n$.

Before proving Theorem 16 we give some definitions and prove useful lemmas. The algorithm we will present is iterative. At each iteration, it chooses a large set of unknown entries and determine their values. The matrix A is partly reconstructed at each iteration, that is, some of its entries are known to the algorithm, others are unknown.

At a given iteration, let $M = (m_{ij}) \in (\mathbb{N}_0 \cup \{*\})^{n\times n}$ be a matrix that holds the known entries of A. That is, $m_{ij} = a_{ij}$ for all entries a_{ij} that are known, and $m_{ij} = *$ for all entries a_{ij} that are still unknown. For such matrix M, we say that M is a *partial copy* of A. Define $\Gamma(m_{ij})$ to be "1" if $m_{ij} = *$ and "0" otherwise. Notice that for any entry m_{ij} such that $b_{ij} = 0$ we have that $a_{ij} = m_{ij} = 0$ and therefore $\Gamma(m_{ij}) = 0$. We extend the definition of Γ in the following way: if x is a row, a column or a matrix then $\Gamma(x)$ equals the number of entries in x that are equal to "$*$".

Now, Let (i_1, j_1) and (i_2, j_2) where $i_1, i_2, j_1, j_2 \in [n]$ be two indices. We say that these indices are "*independent*" in M if $i_1 \neq i_2$, $j_1 \neq j_2$, $\Gamma(m_{i_1j_1}) = 1$, $\Gamma(m_{i_2j_2}) = 1$, $\Gamma(m_{i_1j_2}) = 0$ and $\Gamma(m_{i_2j_1}) = 0$. A set of indices S is called "*independent set of indices*" in M if for every pair of indices $u, v \in S$, u and v are independent in M.

We now give useful lemmas showing how to reconstruct various sets of unknown entries.

Lemma 21. *There exists an algorithm that reconstruct the ith row of A, $A^{(i)}$, (and therefore the ith column, since A is symmetric) using k queries where*

$$k(\log k - 4) \leq 2wt(B^{(i)}) \log \left(\frac{\psi(B^{(i)})}{wt(B^{(i)})} + 1 \right).$$

Proof. We can query the ith row $A^{(i)}$ by asking $e_i^T Ay$. Thus, using Lemma 14 the result follows. □

Next we have,

Lemma 22. *Let M be a partial copy of A. Let $S = \{(i_1, j_1), (i_2, j_2), \ldots, (i_{|S|}, j_{|S|})\}$ be an independent set of indices in M and let $b = \sum_{(i,j)\in S} b_{ij}$. Then, entries $a_{i_1 j_1}, a_{i_2 j_2}, \ldots, a_{i_{|S|} j_{|S|}}$ can be reconstructed using k queries where*

$$k(\log k - 4) \leq 2|S| \log \left(\frac{b}{|S|} + 1 \right).$$

Proof. Let $C = (c_{ij}) \in \mathbb{N}_0^{n \times n}$ be a matrix. Let

$$c_{ij} = (1 - \Gamma(m_{ij}))a_{ij}.$$

In other words, c_{ij} equals a_{ij} if a_{ij} is known and equals zero otherwise. The matrix C can be reconstructed without asking queries. Now, let us look at the matrix

$$D = A - C.$$

The matrix D has the following properties:

1. For every $r \in [|S|]$ we have $d_{i_r j_r} = a_{i_r j_r}$.
2. For every (i, j) such that $i = i_t$, $j = j_r$ where $r, t \in [|S|]$ and $r \neq t$, we have $d_{ij} = 0$.
3. For any two vectors $x, y \in \{0, 1\}^n$ we have $x^T Dy = x^T Ay - x^T Cy$.

Next, let $N = (a_{i_1 j_1}, a_{i_2 j_2}, \ldots, a_{i_{|S|} j_{|S|}})$ be a vector. We now show how to query the vector N, that is, simulate $g(z) = z^T N$ where $z \in \{0, 1\}^{|S|}$ using queries of the form $f(x, y) = x^T Ay$ where $x, y \in \{0, 1\}^n$.

Let $z \in \{0,1\}^{|S|}$ be a vector. Define $X(z) \in \{0,1\}^n$ to be "1" in every entry i_r such that $z_r = 1$ and zero otherwise (here $r \in [|S|]$). Also, let $Y(z) \in \{0,1\}^n$ be "1" in each entry j_r such that $z_r = 1$ and zero otherwise. From property (1) and (2) we have that

$$z^T N = X(z)^T D Y(z).$$

Therefore, by property (3) we get

$$z^T N = X(z)^T A Y(z) - X(z)^T C Y(z).$$

Since C can be reconstructed without asking queries, we can simulate $g(z)$ using $f(x,y)$. By Lemma 14 the result follows. □

Finally, we have

Lemma 23. *Let $\alpha < 1/3$ be any constant and let M be a partial copy of A. If $\Gamma(M) \geq wt(B)^{3\alpha}$ and for all $i \in [n]$, $\Gamma(M^{(i)}) < wt(B)^{\alpha}$ and $\Gamma(M_i) < wt(B)^{\alpha}$, then, there exists an independent set of indices in M of size at least $wt(B)^{\alpha}/2$.*

Proof. Choose any index (i,j) such that $m_{ij} = *$. Now, for every index (i,ℓ_1) such that $\Gamma(m_{i\ell_1}) = 1$ remove the column ℓ_1 from the matrix. Also, for every index (ℓ_2, j) such that $\Gamma(m_{\ell_2,j}) = 1$ remove the row ℓ_2 from the matrix. Since every row or column has at most $wt(B)^{\alpha}$ unknown entries, then we have removed at most $2wt(B)^{2\alpha} - 2wt(B)^{\alpha} + 1$ unknown entries. Note that the index (i,j) is independent with the remaining $\Gamma(M) - 2wt(B)^{2\alpha} + 2wt(B)^{\alpha} - 1$ indices that contains unknown entries.

The above can be repeated

$$\frac{\Gamma(M)}{2wt(B)^{2\alpha} - 2wt(B)^{\alpha} + 1} \geq \frac{wt(B)^{3\alpha}}{2wt(B)^{2\alpha}}$$

times. Thus, by repeating the above we are able to find a set of $(wt(B)^{\alpha})/2$ independent indices in M. □

Now, after proving the above lemmas we present our algorithm.

The algorithm. The algorithm, presented in Figure 4.1, first checks whether

$$wt(B) \leq \frac{wt(B) \log \frac{\psi(B)}{wt(B)}}{\log wt(B)}. \tag{4.3}$$

In case (4.3) is true, the algorithm asks a query to determine the value of every entry a_{ij} such that $b_{ij} > 0$ (the value a_{ij} equals $e_i^T A e_j$). Otherwise, the algorithm works in iterations. At each iteration, the goal is to find a sufficiently large set (that is, of size $\Omega(wt(B)^\alpha)$, for some constant $0 < \alpha < 1/3$) of unknown entries, and to determine their values using the known vector reconstructing algorithm. The algorithm first reconstruct all rows with at least $wt(B)^\alpha$ unknown entries (see Lemma 25). Then, at each iteration, it finds a large set of independent indices and determine the value of the entries in those indices (Lemma 22 and Lemma 23). The algorithm continues iterating until at most $wt(B)^{3\alpha}$ unknown entries are left in A. Then, it asks a query for each unknown entry.

This algorithm is non-adaptive. This is because the choice of a later query does not depend on an earlier query's answer. In each iteration i the algorithm chooses a set of entries S_i to reconstruct. The choice of this set depends on which entries are known and which are unknown; However, we do not rely on the values of the known entries. That is, the sets $S_1, S_2, \ldots, S_t$ can be determined before any query is asked. Thus, the set of queries the algorithm asks can be determined before receiving any answers. After receiving all answers, the construction of the entries must be sequential. That is, we need to know all the values of entries in $S_1, \ldots, S_{i-1}$ to reconstruct S_i.

Complexity Analysis. we now prove Theorem 16.

Proof. At every iteration the algorithm finds a set of entries of size $\Omega(wt(B)^\alpha)$ and determines their values. Let S_i be the set of entries the algorithm reconstructed at iteration i. Note that $S_i \cap S_j = \emptyset$ for all $i, j \in [t]$ where $i \neq j$ and t equals the number of iterations. let $b_i = \sum_{(u,v)\in S_i} b_{uv}$ and $s_i = |S_i|$. By Lemma 14 the number of queries used for reconstructing the ith set, denoted by k_i, satisfies

$$k_i(\log k_i - 4) \leq 2s_i \log\left(\frac{b_i}{s_i} + 1\right).$$

Thus,

$$\sum_i k_i(\log k_i - 4) \leq \sum_i 2s_i \log\left(\frac{b_i}{s_i} + 1\right).$$

Algorithm **Matrix Reconstruct**($B \in \mathbb{N}_0^{n \times n}$)

1. Define $M = (m_{ij}) \in (\mathbb{N}_0 \cup \{*\})^{n \times n}$.
2. **For** every $i, j \in [n]$ **do**
3. $\quad m_{ij} \leftarrow 0$.
4. $\quad$ **if** $b_{ij} > 0$ **then** $m_{ij} \leftarrow *$.
5. **End For**.

6. **if** $wt(B) \leq \frac{wt(B) \log \frac{\psi(B)}{wt(B)}}{\log wt(B)}$ **then**
7. $\quad$ Goto 18.
8. **End if**.

9. **While**($\Gamma(M) > wt(B)^{3\alpha}$)
10. $\quad$ **if** exists i such that $\Gamma(M^{(i)}) > wt(B)^{\alpha}$ **then**
11. $\quad\quad$ reconstruct $M^{(i)}$.
12. $\quad\quad$ update M_i (since A is symmetric).
13. $\quad\quad$ Goto 9.
14. $\quad$ **End if**.
15. $\quad$ Find an independent set of indices S in M such that $|S| > wt(B)^{\alpha}/2$.
16. $\quad$ Determine the values of the entries in S.
17. **End While**.

18. **For** every (i, j) such that $\Gamma(m_{ij}) = 1$ **do**
19. $\quad$ Ask $f(e_i, e_j)$.
20. **End For**.

Algorithm:	**Matrix Reconstruct**
Description:	A reconstruction algorithm for a matrix A.
Complexity:	$O\left(\min\left\{\frac{wt(B) \log \frac{\psi(B)}{wt(B)}}{\log wt(B)}, wt(B)\right\}\right)$

Figure 4.1: Algorithm for reconstructing a matrix given upper bounds on the entries

Let $S = s_1 + s_2 + \cdots + s_t$. We have

$$\begin{aligned}
\sum_i 2s_i \log\left(\frac{b_i}{s_i}+1\right) &= 2\log\prod_i\left(\frac{b_i}{s_i}+1\right)^{s_i} \\
&\leq 2\log\left(\frac{m}{S}+1\right)^{S} \qquad (4.4)\\
&= 2S\log\left(\frac{m}{S}+1\right) \\
&\leq 2wt(B)\log\left(\frac{m}{wt(B)}+1\right). \qquad (4.5)
\end{aligned}$$

In (4.4) we use Lemma 3. The inequality (4.5) follows from the fact that the function $f(x) = x\log(\frac{m}{x}+1)$ is monotone non-decreasing for $x \in [1,m]$ and the fact that $S \leq wt(B) \leq m$.

On the other hand, denote by $K = k_1 + k_2 + \cdots + k_t$. We have

$$\begin{aligned}
\sum_i k_i(\log k_i - 4) &= \sum_i k_i(\log k_i) - 4K \\
&= \log\prod_i (k_i)^{k_i} - 4K \\
&\geq K\log\frac{K}{t} - 4K \qquad (4.6)
\end{aligned}$$

In (4.6) we use Lemma 4. Therefore, the total number of queries K for reconstructing the sets satisfies

$$K\log\frac{K}{t} - 4K \leq 2wt(B)\log\left(\frac{m}{wt(B)}+1\right).$$

Since for all i we have $s_i = \Omega(wt(B)^{\alpha})$, we conclude that the number of sets is

$$t = O(wt(B)^{1-\alpha}).$$

Thus,

$$K = O\left(\frac{wt(B)\ \log\left(\frac{m}{wt(B)}+1\right)}{\log wt(B)}\right).$$

For more details see [10]. To conclude the total number of queries asked is at most

$$O\left(\frac{wt(B)\ \log\left(\frac{m}{wt(B)}+1\right)}{\log wt(B)}\right) + wt(B)^{3\alpha}.$$

Since $\alpha < 1/3$ the result follows. □

After proving Theorem 16 we are now ready to prove our main theorem, Theorem 3. In the previous algorithm we showed how to reconstruct a matrix given an upper bound on every entry of the matrix. Now, we show how to reconstruct the matrix without having those upper bounds.

Theorem 3. *Let $A = (a_{ij}) \in \mathbb{N}_0^{n\times n}$ be a hidden matrix such that the sum of the entries of A is equal to m. Then, there exists a polynomial time algorithm that can reconstruct A using*

$$O\left(\frac{m \log \frac{n^2}{m}}{\log m}\right)$$

queries of the form $f_A(x, y) = x^T Ay$, where $x, y \in \{0, 1\}^n$. This algorithm asks its queries in $\log n$ non-adaptive rounds.

Proof. First, assume without loss of generality that $\log n$ is an integer. Define the following matrices $A^{\{0\}}, A^{\{1\}}, \ldots, A^{\{\log n\}}$ where $A^{\{i\}} \in \mathbb{N}_0^{2^i\times 2^i}$. Denote by $a_{rj}^{\{i\}}$ the (r, j)th entry of $A^{\{i\}}$. Let

$$a_{rj}^{\{i\}} = \psi(A[\, h_1^{\{i\}}(r),\, h_2^{\{i\}}(r) \,|\, h_1^{\{i\}}(j),\, h_2^{\{i\}}(j)])$$

where $h_1^{\{i\}}(z) = (z-1)n/2^i + 1$ and $h_2^{\{i\}}(z) = zn/2^i$ for $z \in \mathbb{Z}$. In other words, if we divide A into 2^{2i} blocks A_{rj} of size $\frac{n}{2^i} \times \frac{n}{2^i}$, that is,

$$A = \left(\begin{array}{c|c|c|c} A_{1,1} & A_{1,2} & \ldots & A_{1,2^i} \\ \hline A_{2,1} & A_{2,2} & \ldots & A_{2,2^i} \\ \hline \vdots & \vdots & & \vdots \\ \hline A_{2^i,1} & A_{2^i,2} & \ldots & A_{2^i,2^i} \end{array}\right),$$

then, $a_{rj}^{\{i\}} = \psi(A_{rj})$ (recall that $\psi(x)$ equals the sum of entries in x). Note that, $A^{\{0\}}$ has one entry that is equal to m. Also note that $A^{\{\log n\}} = A$.

Lemma 24. *We can simulate queries for the matrices $A^{\{i\}}$ using queries for the matrix A.*

Proof. Simply note that $x^T A^{\{i\}} y = (x')^T Ay'$ where

$$x' = (x_0)^{\frac{n}{2^i}} \cdot (x_1)^{\frac{n}{2^i}} \cdots (x_{2^i})^{\frac{n}{2^i}} \quad \text{and} \quad y' = (y_0)^{\frac{n}{2^i}} \cdot (y_1)^{\frac{n}{2^i}} \cdots (y_{2^i})^{\frac{n}{2^i}},$$

and $\cdot$ denote concatenation. □

We now present our main algorithm. Our algorithm is iterative. At iteration i the algorithm knows $A^{\{i-1\}}$ and the goal is to reconstruct the matrix $A^{\{i\}}$.

Note that

$$a_{rj}^{\{i-1\}} = \sum_{d=2r-1}^{2r} \sum_{\ell=2j-1}^{2j} a_{d\ell}^{\{i\}}.$$

Thus, every entry $a_{rj}^{\{i\}}$ is bounded by $a_{\lceil \frac{r}{2} \rceil \lceil \frac{j}{2} \rceil}^{\{i-1\}}$.

Define $B^{\{i\}} = (b_{rj}^{\{i\}}) \in \mathbb{N}_0^{2^i \times 2^i}$ where

$$b_{rj}^{\{i\}} = a_{\lceil \frac{r}{2} \rceil \lceil \frac{j}{2} \rceil}^{\{i-1\}}.$$

Then, we have that $a_{rj}^{\{i\}} \leq b_{rj}^{\{i\}}$ and $\psi(B^{\{i\}}) = 4\psi(A^{\{i-1\}}) = 4m$. Thus, using Lemma 24 and Theorem 16, we are able to reconstruct $A^{\{i\}}$.

Now, since $A^{\{\log n\}}$ is equal to A, then after $\log n$ iteration the algorithm has successfully reconstructed the hidden matrix.

Complexity Analysis We now present a the proof of Theorem 3. Define

$$b_i = wt(B^{\{i\}})$$

Also define

$$K(i) = \min \left\{ \frac{b_i \left(\log \frac{4m}{b_i} + 1 \right)}{\log b_i}, b_i \right\}.$$

By Theorem 16, our total complexity is

$$O\left(\sum_{i=1}^{\log n} K(i) \right).$$

We will divide this sum to three parts $i \leq \frac{1}{4} \log m$, $\frac{1}{4} \log m < i \leq \frac{1}{2} \log m$ and $\frac{1}{2} \log m < i \leq \log n$ and estimate each part separately.

The first part

$$\sum_{i=1}^{\frac{1}{4} \log m} K(i) \leq \sum_{i=1}^{\frac{1}{4} \log m} wt(B^{\{i\}}) \leq \sum_{i=1}^{\frac{1}{4} \log m} 4^i = O\left(\sqrt{m} \right).$$

For the second part, we note that if $b_i \leq \sqrt{m}$ then $K(i) \leq b_i \leq \sqrt{m}$. Otherwise, $b_i > \sqrt{m}$ and then $K(i) \leq \frac{b_i \log \left(\frac{4m}{b_i} + 1 \right)}{\log b_i} \leq \frac{b_i \log \left(\frac{4m}{b_i} + 1 \right)}{\log \sqrt{m}}$. Therefore,

$$K(i) < \frac{b_i \log \left(\frac{4m}{b_i} + 1 \right)}{\log \sqrt{m}} + \sqrt{m}.$$

Now we have

$$\begin{aligned}
\sum_{i=\frac{1}{4}\log m+1}^{\frac{1}{2}\log m} K(i) &\leq \sum_{i=\frac{1}{4}\log m+1}^{\frac{1}{2}\log m} \left(\frac{b_i \log\left(\frac{4m}{b_i}+1\right)}{\log\sqrt{m}} + \sqrt{m} \right) \\
&= \frac{1}{4}\sqrt{m}\log m + \sum_{i=\frac{1}{4}\log m+1}^{\frac{1}{4}\log m} \frac{b_i \log\left(\frac{4m}{b_i}+1\right)}{\log\sqrt{m}} \\
&\leq \frac{1}{4}\sqrt{m}\log m + \sum_{i=\frac{1}{4}\log m+1}^{\frac{1}{2}\log m} \frac{4^i \log\left(\frac{4m}{4^i}+1\right)}{\log\sqrt{m}} \qquad (4.7) \\
&\leq \frac{1}{4}\sqrt{m}\log m + \sum_{i=\frac{1}{4}\log m+1}^{\frac{1}{2}\log m} \frac{4^i \log\left(8m - \log 4^i\right)}{\log\sqrt{m}} \\
&= \frac{1}{4}\sqrt{m}\log m + \frac{1}{\log\sqrt{m}} \sum_{i=\frac{1}{4}\log m+1}^{\frac{1}{2}\log m} \left(4^i \log 8m - 2i\, 4^i\right) \\
&= \frac{1}{4}\sqrt{m}\log m + \frac{(\frac{4m}{3} - \frac{4\sqrt{m}}{3})\log 8m - (\frac{4m}{3}\log m - O(m))}{\log\sqrt{m}} \\
&= O\left(\frac{m}{\log m}\right).
\end{aligned}$$

Inequality (4.7) follows from the fact that the function $f(x) = x\log(\frac{4m}{x}+1)$ is a monotone non-decreasing function for $x \in [1, 4m]$ and the fact that $b_i \leq 4^i \leq m$.

Finally, the last part, that is, $1/2\log m < i \leq \log n$, we have two cases: if $b_i < \sqrt{m}$ then, $K(i) \leq b_i \leq \sqrt{m}$, otherwise, we have $b_i \geq \sqrt{m}$, and then

$$K(i) \leq \frac{b_i \log\left(\frac{4m}{b_i}+1\right)}{\log b_i} \leq \frac{b_i \log\frac{8m}{b_i}}{\log\sqrt{m}} \leq \frac{b_i \frac{8m}{b_i}}{\log\sqrt{m}} = \frac{16m}{\log m}.$$

In both cases we have

$$K(i) = O\left(\frac{m}{\log m}\right).$$

Thus,

$$\sum_{i=\frac{1}{2}\log m+1}^{\log n} K(i) = O\left(\frac{m\log\frac{n}{\sqrt{m}}}{\log m}\right) = O\left(\frac{m\log\frac{n^2}{m}}{\log m}\right).$$

By adding all three part together, the result immediately follows. □

4.4.2 Real Numbers Weights

In this subsection, we present the second algorithm. We show the following,

Theorem 17. *There is a polynomial time algorithm for reconstructing a hidden matrix $A \in (\mathbb{R}_0^+)^{n\times n}$ with at most m non-zero entries that uses*

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

queries of the form $f_A(x,y) = x^T Ay$.

Proof. Assume for simplicity that $\log n$ is an integer. Again, the algorithm is iterative. It searches for non-zero entries using the divide and conquer approach. At iteration τ, the matrix A is partitioned into non-overlapping equal size submatrices $S_{i,j}$, where $i,j \in [2^\tau]$. That is,

$$A = \left(\begin{array}{c|c|c|c} S_{1,1} & S_{1,2} & \dots & S_{1,2^\tau} \\ \hline S_{2,1} & S_{2,2} & \dots & S_{2,2^\tau} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline S_{2^\tau,1} & S_{2^\tau,2} & \dots & S_{2^\tau,2^\tau} \end{array}\right).$$

The algorithm knows the sum of entries in each submatrix $S_{i,j}$, denoted by $\psi(S_{i,j})$. The algorithm partition each submatrix $S_{i,j}$ into four equal sizes submatrices, $S_{i,j,1},\dots,S_{i,j,4}$. The algorithm's goal is to find the sum of entries for all new submatrices $S_{i,j,k}$, for all $i,j \in [2^\tau]$ and $k \in [4]$. That is, the algorithm reconstruct the hidden matrix

$$U = \left(\begin{array}{cc|cc|c|cc} \psi(S_{1,1,1}) & \psi(S_{1,1,2}) & \psi(S_{1,2,1}) & \psi(S_{1,2,2}) & \dots & \psi(S_{1,2^\tau,1}) & \psi(S_{1,2^\tau,2}) \\ \psi(S_{1,1,3}) & \psi(S_{1,1,4}) & \psi(S_{1,2,3}) & \psi(S_{1,2,4}) & \dots & \psi(S_{1,2^\tau,3}) & \psi(S_{1,2^\tau,4}) \\ \hline \psi(S_{2,1,1}) & \psi(S_{2,1,2}) & \psi(S_{2,2,1}) & \psi(S_{2,2,2}) & \dots & \psi(S_{2,2^\tau,1}) & \psi(S_{2,2^\tau,2}) \\ \psi(S_{2,1,3}) & \psi(S_{2,1,4}) & \psi(S_{2,2,3}) & \psi(S_{2,2,4}) & \dots & \psi(S_{2,2^\tau,3}) & \psi(S_{2,2^\tau,4}) \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline \psi(S_{2^\tau,1,1}) & \psi(S_{2^\tau,1,2}) & \psi(S_{2^\tau,2,1}) & \psi(S_{2^\tau,2,2}) & \dots & \psi(S_{2^\tau,2^\tau,1}) & \psi(S_{2^\tau,2^\tau,2}) \\ \psi(S_{2^\tau,1,3}) & \psi(S_{2^\tau,1,4}) & \psi(S_{2^\tau,2,3}) & \psi(S_{2^\tau,2,4}) & \dots & \psi(S_{2^\tau,2^\tau,3}) & \psi(S_{2^\tau,2^\tau,4}) \end{array}\right).$$

Again, Lemma 24, shows how one can simulate an oracle for the matrix U. Now, instead of reconstructing each entry using an upper bound as in the integer case, the algorithm

applies algorithm **Reconstruct_Vector** with the assumption that $\psi(S_{i,j,k}) \in \{0, \psi(S_{i,j})\}$ for all $i, j \in [2^\tau]$ and $k \in [4]$; and then run **Fix_Output** repeatedly to reconstruct the hidden entries $\psi(S_{i,j,k})$.

Formally speaking, define a split in $S_{i,j}$ as the case in which we have that $\psi(S_{i,j,k}) \neq 0$ for more than one $k \in [4]$. We call entries in U that correspond to $S_{i,j,k}$ in which $\psi(S_{i,j,k}) \notin \{0, \psi(S_{i,j})\}$ violating entries. Also define *independent set* of entries and Γ as defined in the previous subsection. The following three lemmas formally show how to find and reconstruct the set of entries in the hidden matrix U.

Lemma 25. *Any row U_s of the matrix U can be constructed using*

$$O\left(\frac{\Gamma(U_s)}{\log \Gamma(U_s)} + \beta \log\log \Gamma(U_s)\right)$$

queries of the form $f(x, y) = x^T U y$, where β is the number violating entries in U_s.

Proof. Note that, for any $x, y \in 2^{\tau+1}$, we have $x^T U_s = e_s U x$. That is, one can simulate queries to U_s of the form $Q(x) = x^T U_s$. Now, using algorithm **Reconstruct_Vector** with the assumption that $\psi(S_{i,j,k}) \in \{0, \psi(S_{i,j})\}$ for all $i, j \in [2^\tau]$ and $k \in [4]$; and then running **Fix_Output** gives the result. □

Lemma 26. *Any row $U^{(s)}$ of the matrix U can be constructed using*

$$O\left(\frac{\Gamma(U^{(s)})}{\log \Gamma(U^{(s)})} + \beta \log\log \Gamma(U^{(s)})\right)$$

queries of the form $f(x, y) = x^T U y$, where β is the number of violating entries in $U^{(s)}$.

Proof. Note that, for any $x, y \in 2^{\tau+1}$, we have $x^T U^{(s)} = x^T U e_s$. Therefore, as in Lemma 25, the result follows. □

Lemma 27. *Let $S = \{u_{i_1,j_1}, u_{i_2,j_2}, \ldots, u_{i_{|S|},j_{|S|}}\}$ be a set of independent entries. Then, these entries can be reconstructed using*

$$O\left(\frac{|S|}{\log |S|} + \beta \log\log |S|\right)$$

queries of the from $f(x, y) = x^T U y$, where β is the number violating entries in S.

Proof. Let v be the following hidden vector $(u_{i_1,j_1}, u_{i_2,j_2}, \ldots, u_{i_{|S|},j_{|S|}})$. We now show how to simulate an oracle to v of the form $Q(x) = x^T v$. Let C be a matrix where $c_{i_s,j_t} = u_{i_s,j_t}$ for all $s,t \in [|S|]$ such that $s \neq t$, and all other entries are equal to zero. Since S is an independent set, all entries of C are known to the algorithm (that is, C is reconstructed without asking queries). Now, for any $x \in \{0,1\}^{|S|}$, let $X(x)$ be the n-vectors, where the ith entry equals

$$X(x)_i = \begin{cases} 1 & i = i_s \text{ for } s \in [|S|] \text{ and } x_s = 1 \\ 0 & \text{otherwise} \end{cases}$$

That is, $X(x)$ equals "1" in every entry i_s such that $s \in [|S|]$ and $x_s = 1$; and equals zero otherwise. Similarly, define $Y(x)$ to be "1" in every entry j_s for which $s \in [|S|]$ and $x_s = 1$ and zero otherwise. Now, note that

$$x^T v = X(x)^T U Y(x) - X(x)^T C Y(x).$$

Therefore, one can simulate queries to v. Now, as before, using algorithm **Reconstruct_Vector** for reconstructing v with the assumption that $X(S_{i,j,k}) \in \{0, X(S_{i,j})\}$ for all $i,j \in [2^\tau]$ and $k \in [4]$; and then running **Fix_Output** gives the result. □

The subroutine presented below is similar to the algorithm **Reconstruct_Matrix**. It uses all the above lemmas to reconstruct the hidden matrix U. It preforms the following steps,

1. **while** $(\Gamma(U) > m^{0.9})$
2. if there is a row/colum, $U_s/U^{(s)}$, with more than $m^{0.3}$ unknown entries, then reconstruct it and goto 1.
3. find independent set of size at least $m^{0.3}/2$ and reconstruct it.
4. **end of while**.
5. Ask a query $f(e_s, e_t)$ to determine every remaining unknown entry $u_{s,t}$.

After running subroutine and finding all entries in U, the algorithm advances to iteration $\tau + 1$.

As for the complexity analysis, the algorithm runs $\log n$ iterations. At each iteration τ, let β_τ denote the number of splits in this iterations. The subroutine above asks at most

$$O\left(\frac{m}{\log m^{0.3}} + m^{0.9} + \beta_\tau \log\log m\right) = O\left(\frac{m}{\log m} + \beta_\tau \log\log m\right).$$

Therefore, the total query complexity is

$$\sum_{\tau=1}^{\log n} O\left(\frac{m}{\log m}\right) + O(\beta_\tau \log\log m).$$

Since $\sum_{\tau=1}^{\log n} \beta_\tau = m$, we get the result. □

Chapter 5

Reconstructing Hidden Hypergraphs

In this section, we present all the results for reconstructing constant rank weighted hypergraphs using queries. We start by introducing d-dimensional matrices (Tensors) as we use them to represent the hypergraphs.

5.1 d-Dimensional Matrices

A *d-dimensional matrix* A of *size* $n_1 \times \cdots \times n_d$ over a field $\mathbb{F}$ is a map $A : \prod_{i=1}^{d}[n_i] \to \mathbb{F}$. We denote by $\mathbb{F}^{n_1 \times \cdots \times n_d}$ the set of all d-dimensional matrices A of size $n_1 \times \cdots \times n_d$. We write $A_{i_1,\ldots,i_d}$ for $A(i_1, \ldots, i_d)$.

The zero map is denoted by $0^{n_1 \times \cdots \times n_d}$. The matrix $B = (A_{i_1,i_2,\ldots,i_d})_{i_1 \in I_1, i_2 \in I_2, \ldots, i_d \in I_d}$ where $I_j \subseteq [n_j]$, is the $|I_1| \times \cdots \times |I_d|$ matrix where $B_{j_1,\ldots,j_d} = A_{\ell_1,\ldots,\ell_d}$ and ℓ_i is the j_ith smallest number in I_i. When $I_j = [n_j]$ we just write j and when $I_j = \{\ell\}$ we just write $j = \ell$. For example, $(A_{i_1,i_2,\ldots,i_d})_{i_1,i_2=\ell,i_3 \in I_2,\ldots,i_d \in I_d} = (A_{i_1,i_2,\ldots,i_d})_{i_1 \in [n_1], i_2 \in \{\ell\}, i_3 \in I_2, \ldots, i_d \in I_d}$.

When $n_1 = n_2 = \cdots = n_d = n$ then we denote $\mathbb{F}^{n_1 \times \cdots \times n_d}$ by $\mathbb{F}^{\times_d n}$ and $0^{n_1 \times \cdots \times n_d}$ by $0^{\times_d n}$.

We say that the entry $A_{i_1,i_2,\ldots,i_d}$ is of *dimension* r if $|\{i_1, \ldots, i_d\}| = r$. For d-dimensional matrix A we denote by $wt(A)$ the number of points in $\prod_{i=1}^{d}[n_i]$ that are mapped to non-zero elements in $\mathbb{F}$. We denote by $wt_r(A)$ the number of points in $\prod_{i=1}^{d}[n_i]$ of dimension r that are mapped to non-zero elements in $\mathbb{F}$. Therefore, $wt(A) = wt_1(A) + wt_2(A) + \cdots + wt_d(A)$. We denote by $\mathcal{A}_{d,m}$ the set of d-dimensional matrices $A \in \mathbb{F}^{\times_d n}$ where $wt_d(A) \leq m$

and $\mathcal{A}^{\star}_{d,m}$ the set of d-dimensional matrices $A \in \mathbb{F}^{\times_d n}$ where $1 \leq wt_d(A) \leq m$.

For d-dimensional matrix A of size $n_1 \times \cdots \times n_d$ and $x_i \in \mathbb{F}^{n_i}$ we define

$$A(x_1, \ldots, x_d) = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} A_{i_1,i_2,\ldots,i_d} x_{1i_1} \cdots x_{di_d}.$$

The vector $v = A(\cdot, x_2, \ldots, x_d)$ is n_1-dimensional vector that its ith entry is

$$\sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} A_{i,i_2,\ldots,i_d} x_{2i_2} \cdots x_{di_d}.$$

For a set of d-dimensional matrices $\mathcal{B}$, a set $S \subseteq (\{0,1\}^n)^d$ is called a *zero test set* for $\mathcal{B}$ if for every $A \in \mathcal{B}$, $A \neq 0$, there is $x \in S$ such that $A(x) \neq 0$.

A d-dimensional matrix is called *symmetric* if for every $i = (i_1, \ldots, i_d) \in [n]^d$ and any permutation ϕ on $[d]$, we have $A_i = A_{\phi i}$, where $\phi i = (i_{\phi(1)}, \ldots, i_{\phi(d)})$. Notice that for a symmetric d-dimensional matrix $A \in \mathbb{F}^{\times_d n}$, $x_i \in \{0,1\}^n$ and any permutation ϕ on $[d]$, we have $A(x_1, \ldots, x_d) = A(x_{\phi(1)}, \ldots, x_{\phi(d)})$.

We will be interested mainly in the fields $\mathbb{F} = \mathbb{R}$, the field of real numbers, and $\mathbb{F} = \mathbb{Z}_p$, the field of integers modulo p, and in matrices of constant $d = O(1)$ dimension. Also $p > d!$. Although it seems that we are restricting the parameters, the final result has no restriction on the parameters except for $d = O(1)$. We will also abuse the notations $\mathbb{Z}_p$ and $=_p$ and allow $p = \infty$ (so in this thesis ∞ is also prime number). In that case $\mathbb{Z}_\infty = \mathbb{R}$ and $=_\infty$ is equality in the filed of real numbers.

5.2 Hypergraphs

A *hypergraph* G is a pair $G = (V, E)$ where $V = [n]$ is a set of elements, called nodes or vertices, and E is a set of non-empty subsets of 2^V called *hyperedges* or *edges*. The *rank* $r(G)$ of a hypergraph G is the maximum cardinality of any of the edges in the hypergraph. A hypergraph is called *d-uniform* if all of its edges are of size d.

A *weighted hypergraph* $G = (V, E, w)$ over $\mathbb{Z}_p$ is a hypergraph (V, E) with a weight function $w : E \to \mathbb{Z}_p$. For two weighted hypergraph $G_1 = (V, E_1, w_1)$ and $G_2 = (V, E_2, w_2)$ we define the weighted hypergraph $G_1 - G_2 = (V, E, w)$ where $E = \{e \in E_1 \cup E_2 \mid w_1(e) \neq$

$w_2(e)\}$, and for every $e \in E$, $w(e) = w_1(e) - w_2(e)$. Obviously, $G_1 = G_2$ if and only if $G_1 - G_2$ is an independent set, i.e., $E = \emptyset$.

We denote by $\mathcal{G}_d$ the set of all weighted hypergraphs over $\mathbb{Z}_p$ of rank at most d, $\mathcal{G}_{d,m}$ the set of all weighted hypergraphs over $\mathbb{Z}_p$ of rank at most d and at most m edges and $\mathcal{G}^\star_{d,m}$ the set of all weighted hypergraphs over $\mathbb{Z}_p$ of rank d and at most m edges.

Let $w^\star : 2^V \to \mathbb{Z}_p$ be w extended to all possible edges where for $e \in E$, $w^\star(e) = w(e)$ and for $e \notin E$, $w^\star(e) = 0$.

An *adjacency d-dimensional matrix of a weighted hypergraph* G is a d-dimensional matrix A^G_d where $d \geq r(G)$ such that for every set $e = \{i_1, i_2, \ldots, i_\ell\}$ of size at most d we have $A^G_{d(j_1,\ldots,j_d)} =_p w^\star(e)/N(d,\ell)$ for all $j_1, \ldots, j_d$ such that $\{j_1, j_2, \ldots, j_d\} = \{i_1, \ldots, i_\ell\}$ where

$$N(d,\ell) = \sum_{i=0}^{\ell} (-1)^i \binom{\ell}{i} (\ell - i)^d.$$

That is, $N(d,\ell)$ is the number of possible sequences $(j_1, \ldots, j_d)$ such that $\{j_1, \ldots, j_d\} = \{i_1, \ldots, i_\ell\}$. Note that $N(d,\ell) \leq d! < p$ and therefore $N(d,\ell) \neq_p 0$ and A^G_d is well defined.

It is easy to see that the adjacency matrix of a weighted hypergraph is a symmetric matrix and $r(G) = r$ if and only if the adjacency matrix of G has an non-zero entry of dimension r and all entries of dimension greater than r are zero.

5.3 Additive Model

Recall that in the Additive Model the goal is to exactly learn a hidden hypergraph with minimal number of additive queries. Given a set of vertices $S \subseteq V$, an *additive query*, $Q_G(S)$, returns the sum of weights in the subgraph induces by S. That is, $Q_G(S) =_p \sum_{e \in E \cap 2^S} w(e)$. Our goal is to exactly reconstruct the set of edges and find their weights using additive queries. See the many applications of this problem in [9, 21, 22].

We say that the set $\mathcal{S} = \{S_1, S_2, \cdots, S_k\} \subseteq 2^V$ is a *detecting set* for $\mathcal{G}_{d,m}$ if for any hypergraph $G \in \mathcal{G}_{d,m}$ there is S_i such that $Q_G(S_i) \neq 0$. We say that the set $\mathcal{S} = \{S_1, S_2, \cdots, S_k\} \subseteq 2^V$ is a *search set* for $\mathcal{G}_{d,m}$ if for any two distinct hypergraphs $G_1, G_2 \in \mathcal{G}_{d,m}$ there is S_i such that $Q_{G_1}(S_i) \neq Q_{G_2}(S_i)$. That is, given $(Q_G(S_i))_i$ one can uniquely determines G. We now prove the following,

Lemma 28. *If $\mathcal{S} = \{S_1, S_2, \cdots, S_k\} \subseteq 2^V$ is a detecting set for $\mathcal{G}_{d,2m}$ then it is a search set for $\mathcal{G}_{d,m}$.*

Proof. Let $G_1, G_2 \in \mathcal{G}_{d,m}$ be two distinct weighted hypergraphs. Let $G = G_1 - G_2$. Since $G \in \mathcal{G}_{d,2m}$ there must be $S_i \in \mathcal{S}$ such that $Q_G(S_i) \neq 0$. Since $Q_G(S_i) = Q_{G_1}(S_i) - Q_{G_2}(S_i)$ we have $Q_{G_1}(S_i) \neq Q_{G_2}(S_i)$. □

5.4 Algebraic View of the Model

It is easy to show that for any hypergraph G of rank r the adjacency d-dimensional matrix of G, A_d^G, for $d \geq r$, is symmetric, contains a nonzero entry of dimension r and

$$Q_G(S) =_p A_d^G(x^S, x^S, \overset{d}{\ldots}, x^S) \triangleq B_d^G(x^S).$$

Recall that for $S \subseteq [|V|]$, $x^S \in \{0,1\}^{|V|}$ is a vector where $x_i^S = 1$ if and only if $i \in S$.

For a symmetric d-dimensional matrix A let $B(x) =_p A(x, x, \overset{d}{\ldots}, x)$ where $x \in \{0,1\}^n$. When $x_1, \ldots, x_d \in \{0,1\}^n$ are pairwise disjoint the following lemma shows that $A(x_1, \ldots, x_d)$ can be found by 2^d values of B.

Lemma 29. *If $x_1, \ldots, x_d \in \{0,1\}^n$ are pairwise disjoint then*

$$A(x_1, \ldots, x_d) =_p \frac{1}{d!} \sum_{I \in 2^{[d]}} (-1)^{d-|I|} B\left(\sum_{i \in I} x_i\right).$$

Proof. Since

$$A(x_1 + x_1', x_2, \ldots, x_d) =_p A(x_1, x_2, \ldots, x_d) + A(x_1', x_2, \ldots, x_d)$$

and

$$A(x_1, x_2, \ldots, x_d) =_p A(x_{\phi(1)}, x_{\phi(2)}, \ldots, x_{\phi(d)})$$

for any permutation ϕ on $[d]$, the result is analogous to the fact that

$$y_1 y_2 \cdots y_d =_p \frac{1}{d!} \sum_{I \in 2^{[d]}} (-1)^{d-|I|} \left(\sum_{i \in I} y_i\right)^d, \tag{5.1}$$

for formal variables $y_1, \ldots, y_d$. Now notice that

$$\left(\sum_{i\in I} y_i\right)^d =_p \sum_{q_1+\cdots+q_d=d} \chi\left[\{i|q_i \neq 0\} \subseteq I\right] \binom{d}{q_1\ q_2\ \cdots q_d} y_1^{q_1}\cdots y_d^{q_d},$$

where $\chi[L] = 1$ if the statement L is true and 0 otherwise. Therefore, the coefficient of $y_1^{q_1}\cdots y_d^{q_d}$ in the right hand side of (5.1) is

$$\sum_{I\in 2^{[d]}} (-1)^{d-|I|}\chi\left[\{i|q_i \neq 0\} \subseteq I\right] \binom{d}{q_1\ q_2\ \cdots q_d}$$

$$=_p \binom{d}{q_1\ q_2\ \cdots q_d} \sum_{I\in 2^{[d]}} (-1)^{d-|I|}\chi\left[\{i|q_i \neq 0\} \subseteq I\right].$$

Now if $\ell = |\{i|q_i \neq 0\}| < d$ then

$$\sum_{I\in 2^{[d]}} (-1)^{d-|I|}\chi\left[\{i|q_i \neq 0\} \subseteq I\right] =_p \sum_{i=\ell}^{d} (-1)^{d-i}\binom{d-\ell}{i-\ell} =_p \sum_{i=0}^{d-\ell} (-1)^{d-\ell-i}\binom{d-\ell}{i} = 0.$$

If $\ell = |\{i|q_i \neq 0\}| = d$ then $q_1 = q_2 = \cdots = q_d = 1$ and

$$\sum_{I\in 2^{[d]}} (-1)^{d-|I|}\chi\left[\{i|q_i \neq 0\} \subseteq I\right] =_p 1.$$

This implies the result. □

Let G be a hypergraph of rank d and $G^{(i)}$, $i \leq d$, be the sub-hypergraph of G that contains all the edges in G of size i then

Lemma 30. *If $x_1, \ldots, x_d \in \{0,1\}^n$ are pairwise disjoint then, we have that $A_d^G(x_1, \ldots, x_d) = A_d^{G^{(d)}}(x_1, \ldots, x_d)$. In particular, if $r(G) < d$ then $A_d^G(x_1, \ldots, x_d) = 0$.*

Proof. Since $x_1, \ldots, x_d \in \{0,1\}^n$ are pairwise disjoint we have

$$\begin{aligned} A_d^G(x_1, \ldots, x_d) &= \sum_{i_1=1}^{n_1}\cdots\sum_{i_d=1}^{n_d} \frac{w^\star(\{i_1, i_2, \ldots, i_d\})}{N(d, |\{i_1, i_2, \ldots, i_d\}|)} x_{1i_1}\cdots x_{di_d} \\ &= \sum_{|\{i_1,\ldots,i_d\}|=d} \frac{w^\star(\{i_1, i_2, \ldots, i_d\})}{N(d,d)} x_{1i_1}\cdots x_{di_d} \\ &= A_d^{G^{(d)}}(x_1, \ldots, x_d). \end{aligned}$$

Now when $r(G) < d$ then $G^{(d)}$ is an independent set (has no edges) and $A_d^{G^{(d)}} = 0$. Then $A_d^G(x) = A_d^{G^{(d)}}(x) = 0$. □

We now prove

Lemma 31. *Let $\Phi_d = \{z_1^{(d)}, \ldots, z_{k_d}^{(d)}\} \subset (\{0,1\}^n)^d$ where for every i the vectors $z_{i,1}^{(d)}, \ldots, z_{i,d}^{(d)}$ are pairwise disjoint. If Φ_d is a zero test set for $\mathcal{A}^\star_{d,(d!)m}$ then*

$$S^{\Phi_d} \triangleq \left\{ S^{y_J} \;\middle|\; y_J = \sum_{j \in J} z_{i,j}^{(d)},\ J \subseteq [d] \right\}$$

is a detecting set for $\mathcal{G}^\star_{d,m}$.

Proof. Let Φ_d be a zero test set for $\mathcal{A}^\star_{d,(d!)m}$. Let $G \in \mathcal{G}^\star_{d,m}$. Then $A_d^G \neq 0$ and $A_d^G \in \mathcal{A}^\star_{d,(d!)m}$. Therefore, for every $G \in \mathcal{G}^\star_{d,m}$ there is $z_i^{(d)}$ such that $A_d^G(z_i^{(d)}) \neq 0$. By Lemma 29,

$$A_d^G(z_i^{(d)}) =_p \frac{1}{d!} \sum_{J \in 2^{[d]}} (-1)^{d-|J|} B_d^G \left(\sum_{j \in J} z_{i,j}^{(d)} \right) \neq 0,$$

and therefore for some $J_0 \subseteq [d]$,

$$B_d^G \left(\sum_{j \in J_0} z_{i,j}^{(d)} \right) \neq 0,$$

which implies that $Q_G\left(S^{y_{J_0}}\right) \neq 0$ for $y_{J_0} = \sum_{j \in J_0} z_{i,j}^{(d)}$. □

We now show

Lemma 32. *A detecting set for $\mathcal{G}_{d,m}$ over $\mathbb{Z}_p$ is a detecting set for $\mathcal{G}_{d,m}$ over $\mathbb{R}$.*

Proof. Consider a detecting set $\mathcal{S} = \{S_1, S_2, \cdots, S_k\} \subseteq 2^V$ for $\mathcal{G}_{d,m}$ over $\mathbb{Z}_p$. Consider a $k \times q$ matrix M where

$$q = \sum_{i=0}^{d} \binom{n}{i}$$

that its columns are labelled with sets in $2^{[n]}$ of size at most d and for every $S \subset [n]$ of size at most d we have $M[i,S] = 1$ if $S \subseteq S_i$ and 0 otherwise. Consider for every graph $G \in \mathcal{G}_{d,m}$ a q-vector v_G that its entries are labelled with subsets of $[n]$ of size at most d and $v_G[S] = w^\star(S)$. The labels in v_G are in the same order as the labels of the columns of M. Then it is easy to see that

$$M v_G =_p (Q_G(S_1), \ldots, Q_G(S_k))^T.$$

Since $Mv_G \neq_p 0$ for every $v_G \in \mathbb{Z}_p^q$ of weight at least one and at most m, every m columns in M are linearly independent over $\mathbb{Z}_p$. Since the entries of M are zeros and ones every m columns in M are linearly independent over $\mathbb{R}$. Therefore,

$$Mv_G = (Q_G(S_1), \ldots, Q_G(S_k))^T \neq 0,$$

for every $v_G \in \mathbb{R}^q$ of weight at least 1 and at most m. □

5.5 Distributions

In this section we give a distribution that will be used in this chapter.

The *uniform disjoint distribution* $\Omega_{d,n}(x)$ over $(\{0,1\}^n)^d$ is defined as

$$\Omega_{d,n}(x) = \begin{cases} \frac{1}{(d+1)^n} & x_1, \ldots, x_d \text{ is pairwise disjoint.} \\ 0 & \text{otherwise.} \end{cases}$$

In order to choose a random vector x according to the uniform disjoint distribution, one can randomly independently uniformly choose n elements $w_1, w_2, \ldots, w_n$ where $w_i \in [d+1]$ and define the following vector $x = (x_1, x_2, \ldots, x_d) \in (\{0,1\}^n)^d$:

$$x_{ji} = \begin{cases} 1 & j = w_i \text{ and } w_i \in [d] \\ 0 & \text{otherwise.} \end{cases}$$

We call any index $k \in [n]$ such that $x_{jk} = 0$ for all $j \in [d]$ a *free index*. Let $\Gamma_{d,n} \subset (\{0,1\}^n)^d$ be the set of all pairwise disjoint d-tuple.

5.6 Preliminary Results

In this section we prove,

Lemma 33. *Let $A \in \mathbb{F}^{\times_d n} \setminus \{0^{\times_d n}\}$ be an adjacency d-dimensional matrix of a hypergraph G of rank d. Let $x = (x_1, x_2, \ldots, x_d) \in (\{0,1\}^n)^d$ be a randomly chosen d-tuple, that is chosen according to the distribution $\Omega_{d,n}$. Then*

$$\Pr_{x \in \Omega_{d,n}}[A(x) = 0] \leq 1 - \frac{1}{(d+1)^d}.$$

Proof. Let $e = \{i_1, \ldots, i_d\}$ be an edge of size $|e| = d$ and let $x'_j = (x_{j,i_1}, \ldots, x_{j,i_d})$. Consider $\phi(x'_1, \ldots, x'_d)$ that is equal to $A(x)$ with some fixed $x_{j,i} = \xi_{j,i} \in \{0,1\}$ for $i \notin e$. Since $A(x)$ contains the monomial $M = x_{1,i_1} x_{2,i_2} \cdots x_{d,i_d}$ and no other monomial in $A(x)$ contains it, ϕ contains monomial M and therefore $\phi(x'_1, \ldots, x'_d) \not\equiv 0$. If we substitute $x_{j_1,i_{j_2}} = 0$ in ϕ for all $j_1 \neq j_2$ we still get a nonzero function $\phi'(x_{1,i_1}, x_{2,i_2}, \cdots, x_{d,i_d})$ that contains M. Therefore, there is $\xi = (\xi_{1i_1}, \xi_{2i_2}, \cdots, \xi_{di_d}) \in \{0,1\}^d$ such that $\phi'(\xi) \neq 0$. The probability that $(x_{1,i_1}, x_{2,i_2}, \cdots, x_{d,i_d}) = \xi$ and $x_{j_1,i_{j_2}} = 0$ for all $j_1 \neq j_2$ is $(1/d+1)^d$. This implies the result. □

5.7 Reconstructing Hypergraphs

In this section we prove our main theorems, Theorem 7 and Theorem 8. We restate them more formally.

Theorem 7. *There is a search set for ${\mathcal{G}'}_{d,m}$ over $\mathbb{R}$ of size $k = O\left(\frac{m \log \frac{n^d}{m}}{\log m}\right)$, where ${\mathcal{G}'}_{d,m}$ denotes the set of all weighted hypergraphs over $\mathbb{R}$ of rank at most d, at most m edges and weights that are integers bounded by $w = poly(n^d/m)$.*

Theorem 8. *There is a search set for $\mathcal{G}_{d,m}$ over $\mathbb{R}$ of size $k = O\left(\frac{m \log n}{\log m}\right)$.*

Proof. We give the proof of Theorem 8. The proof of Theorem 7 is similar.

Let $m < p < 2m$ be a prime number. Suppose there is a zero test set from $\Gamma_{d,n}$ for $\mathcal{A}^\star_{d,m}$ over $\mathbb{Z}_p$ of size $T(n,m,d)$. By Lemma 31, there is a detecting set for $\mathcal{G}^\star_{d,m}$ over $\mathbb{Z}_p$ of size $2^d T(n,(d!)m,d)$. Therefore, by Lemma 30, there is a detecting set for $\mathcal{G}_{d,m}$ over $\mathbb{Z}_p$ of size $T'(n,m,d) = \sum_{\ell=1}^{d} 2^\ell T(n,(\ell!)m,\ell)$. By Lemma 32, there is a detecting set for $\mathcal{G}_{d,m}$ over $\mathbb{R}$ of size $T'(n,m,d)$. Finally, by Lemma 28, there is a search set for $\mathcal{G}_{d,m}$ over $\mathbb{R}$ of size $T'(n,2m,d)$. Now for constant d, if

$$T(n,m,d) = O\left(\frac{m \log n}{\log m}\right), \tag{5.2}$$

then $T'(n,2m,d) = O(T(n,m,d))$. Therefore it is enough to prove the following.

Lemma 34. *Let p be a prime number such that $m < p < 2m$. There exists a set $S = \{x_1, x_2, \ldots, x_k\} \subseteq (\{0,1\}^n)^d$ where $x_i = (x_{i,1}, \ldots, x_{i,d}) \in \Gamma_{d,n}$ for $i \in [k]$ and*

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

such that: for every d-dimensional matrix $A \in \mathbb{Z}_p^{\times_d n} \setminus \{0^{\times_d n}\}$ with $1 \leq wt_d(A) \leq m$ there exists an i such that $A(x_i) \neq_p 0$.

Proof. Since $wt_d(A) > 1$ the matrix A has at least one nonzero entry of dimension d. We will assume that all the entries of dimension less than d are zero, that is, $wt(A) = wt_d(A)$. This is because, by Lemma 30, the entries of dimension less than d have no effect when the vectors $x_i \in \Gamma_{d,n}$.

We divide the set of such matrices $\mathcal{A} = \{A \mid A \in \mathbb{Z}_p^{\times_d n} \setminus \{0^{\times_d n}\}$ and $wt(A) \leq m\}$ into $d+1$ (non-disjoint) sets:

- $\mathcal{A}_0$: The set of all non-zero matrices $A \in \mathbb{Z}_p^{\times_d n}$ such that $wt(A) \leq m/\log m$.
- $\mathcal{A}_j$ for $j = 1, \ldots, d$: The set of all non-zero matrices $A \in \mathbb{Z}_p^{\times_d n}$ such that $m \geq wt(A) > m/\log m$ and there are at least

  $$\left(\frac{m}{\log m}\right)^{1/d}$$

  non-zero elements in $I_j = \{i_j | \exists (i_1, i_2, \ldots, i_{j-1}, i_{j+1}, \ldots, i_d) : A_{i_1, i_2, \ldots, i_d} \neq 0\}$.

Note that $I = \{(i_1, i_2, \ldots, i_d) | A_{i_1,i_2,\ldots,i_d} \neq 0\} \subseteq I_1 \times I_2 \times \cdots \times I_d$ and therefore either $I = wt(A) \leq m/\log m$ or there is j such that $|I_j| > (m/\log m)^{1/d}$. Therefore, $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_d$.

Using the probabilistic method, we give $d+1$ sets of pairwise disjoint tuples of vectors $S_0, S_1, \ldots, S_d$ such that for every $j \in \{0\} \cup [d]$ and $A \in \mathcal{A}_j$ there exists a d-tuple x in S_j such that $A(x) \neq 0$ and

$$|S_0| + |S_1| + \cdots + |S_d| = O\left(\frac{m \log n}{\log m}\right).$$

Case 1: $A \in \mathcal{A}_0$: For a random d-tuple x, chosen according to the distribution $\Omega_{d,n}$ we have that

$$\Pr_x[A(x) =_p 0] \leq 1 - \frac{1}{(d+1)^d}.$$

If we randomly choose

$$k_1 = \frac{cm \log n}{\log m}$$

d-tuples, $x_1, \ldots, x_{k_1}$, according to the distribution $\Omega_{d,n}$, then the probability that $A(x_i) = 0$ for all $i \in [k_1]$ is

$$\Pr[\forall i \in [k_1] : A(x_i) =_p 0] \leq \left(1 - \frac{1}{(d+1)^d}\right)^{k_1}.$$

Therefore, by union bound, the probability that there exists a matrix $A \in \mathcal{A}_0$ such that $A(x_i) = 0$ for all $i \in [k_1]$ is

$$\begin{aligned} \Pr[\exists A \in \mathcal{A}_0, \forall i \in [k_1] : A(x_i) =_p 0] &\leq \binom{n^d}{\frac{m}{\log m}} p^{\frac{m}{\log m}} \left(1 - \frac{1}{(d+1)^d}\right)^{\frac{cm \log n}{\log m}} \\ &< n^{d\frac{m}{\log m}} n^{\frac{m}{\log m}} n^{-\frac{c'cm}{\log m}} < 1, \end{aligned}$$

for some constant c. This implies the result.

Case2: $A \in \mathcal{A}_j$ where $j = 1, \ldots, d$: We will assume w.l.o.g that $j = 1$. We first prove the following lemma

Lemma 35. *Let $U \subseteq \mathbb{Z}_p^{\times d-1 n}$ be the set of all $d-1$-dimensional matrices with weight smaller than $m^{d/(d+1)}$. For $A \in U$ let $\Upsilon(A) \subseteq [n]$ be following set*

$$\Upsilon(A) = \{j \,|\, \exists A_{i_1,i_2,\ldots,i_{d-1}} \neq 0 \text{ and } j \notin \{i_1, i_2, \ldots, i_{d-1}\}\}.$$

Define $Q = \{(A, j) \,|\, A \in U \text{ and } j \in \Upsilon(A)\}$. Then, there is a constant c_0 such that for every $C > c_0$ and

$$k_2 = C\frac{m \log n}{\log m}$$

there exists a multi-set of $d-1$-tuples of (0,1)-vectors $Z = \{z_1, z_2, \ldots, z_{k_2}\} \subseteq (\{0,1\}^n)^{d-1}$ such that for every $(A, j) \in Q$ the size of the set

$$Z_{(A,j)} = \{i \,|\, A(z_i) \neq 0 \text{ and } j \text{ is a free index}\}$$

is at least $\frac{k_2}{2d^d}$.

Proof. Let $z_i = (z_{i,1}, z_{i,2}, \ldots, z_{i,d-1}) \in (\{0,1\}^n)^{d-1}$ be random $d-1$-tuple of $(0,1)$-vector chosen according to the distribution $\Omega_{d-1,n}$. For $(A,j) \in Q$, and by Lemma 33, we have

$$\Pr_{z_i \in \Omega_{d-1,n}}[A(z_i) \neq 0 \text{ and } j \text{ is a free}] = \Pr[j \text{ is free}] \Pr[A(z_i) \neq 0 | j \text{ is free}] \geq \frac{1}{d} \cdot \frac{1}{d^{d-1}} = \frac{1}{d^d}.$$

Therefore, the expected size of $Z_{(A,j)}$ is greater than $\frac{k_2}{d^d}$. By Chernoff bound, if we randomly choose all z_i, $i \in [k_2]$ according to the distribution $\Omega_{d-1,n}$, then, we have

$$\Pr\left[|Z_{(A,j)}| \leq \frac{k_2}{2d^d}\right] \leq e^{\frac{-k_2}{8d^d}}.$$

Thus, the probability that there exists $(A,j) \in Q$ such that $|Z_{(A,j)}| \leq \frac{k_2}{2d^d}$ is

$$\begin{aligned}\Pr\left[\exists (A,j) \in Q : |Z_{(A,j)}| \leq \frac{k_2}{2d^d}\right] &\leq \frac{|Q|}{e^{\frac{-k_2}{8d^d}}} \leq \frac{|U \times [n]|}{e^{\frac{-k_2}{8d^d}}} \leq \frac{n\binom{n^{d-1}}{m^{d/(d+1)}} p^{m^{d/(d+1)}}}{e^{\frac{Cm \log n}{8d^d \log m}}} \\ &\leq \frac{n\binom{n^{d-1}}{m^{d/(d+1)}} n^{m^{d/(d+1)}}}{n^{\frac{C(\log e)m}{8d^d \log m}}} \leq \frac{n^{O\left(m^{d/(d+1)}\right)}}{n^{\frac{Cc'm}{\log m}}} < 1,\end{aligned}$$

for large enough C. This implies the result. □

Now, Let U and Q be the sets we defined in Lemma 35. Let $A \in \mathcal{A}_1$. Since $wt(A) \leq m$ there are at most $m^{1/(d+1)}$ $d-1$-dimensional matrices $(A_{i_1,i_2,\ldots,i_d})_{i_1=j,i_2,\ldots,i_d}$ with weight greater than $m^{d/(d+1)}$. Therefore, there is at least

$$q = \left(\frac{m}{\log m}\right)^{1/d} - m^{1/(d+1)}$$

indices j such that $(A_{i_1,i_2,\ldots,i_d})_{i_1=j,i_2,\ldots,i_d} \in U$. Let U' contain any q indices such that $(A_{i_1,i_2,\ldots,i_d})_{i_1=j,i_2,\ldots,i_d} \in U$. Let A_U be the matrix

$$(A_{i_1,i_2,\ldots,i_d})_{i_1 \in U', i_2,\ldots,i_d}.$$

Let $z_1, z_2, \ldots, z_{k_2} \in (\{0,1\}^n)^{d-1}$ be the set we proved its existence in Lemma 35. We now choose $x_i \in \{0,1\}^n$, $i \in [k_2]$ in the following way: Take z_i. For every free index j, choose x_{ij} to be "1" with probability 1/2 and "0" with probability 1/2 (independently for every j). All other entries in x_i are zero, that is, all entries that correspond to non-free index j in z_i are zero. Let $u \in \{0,1\}^n$ be a vector where $u_j = 1$ if $j \in U'$ and zero

otherwise. Also, for a $d-1$-tuple z_i let $v_i \in \{0,1\}^n$ be the vector where $v_{ij} = 1$ if j is a free index in z_i and $v_{ij} = 0$ otherwise. By Lemma 2 we have that

$$\Pr_x[A(x_i, z_i) =_p 0] \leq \prod_i \frac{1}{\iota(wt(v_i * A(\cdot, z_i)))^\beta} \leq \prod_i \frac{1}{\iota(wt(v_i * (u * A(\cdot, z_i))))^\beta}. \tag{5.3}$$

Here ι is the function defined in Lemma 20.

Note that, A is a hypergraph, thus, for every j such that $(A_{i_1,i_2,\ldots,i_d})_{i_1=j,i_2,\ldots,i_d} \in U$, we have that $((A_{i_1,i_2,\ldots,i_d})_{i_1=j,i_2,\ldots,i_d}, j) \in Q$. Therefore,

$$\sum_i wt(v_i * (u * A(\cdot, z_i))) \geq \frac{qk_2}{2d^d}.$$

Using Lemma 20 we have

$$\prod_i \iota(wt(v_i * (u * A(\cdot, z_i)))) \geq q^{\lfloor \frac{\frac{qk_2}{2d^d} - k_2}{q-1} \rfloor} = m^{c_1 k_2}.$$

Therefore, using (5.3), $\Pr_x[A(x_i, z_i) =_p 0] \leq \frac{1}{m^{c_1 \beta k_2}}$. Thus, the probability that there exists a matrix $A \in \mathcal{A}_1$ such that for all $i \in [k_2]$ we have $A(x_i, z_i) = 0$ is

$$\Pr_x[A(x_i, z_i) =_p 0] \leq \frac{|\mathcal{A}_1|}{m^{c_1 \beta k_2}} \leq \frac{\binom{n^d}{m} p^m}{m^{c_1 \beta k_2}} \leq \frac{n^{dm} n^m}{m^{c_1 \beta k_2}} < 1,$$

for large enough constant. This implies Lemma 34. □

This completes the proof of Theorem 8. □

Chapter 6

Noise

In this chapter, we give results for the coin weighing problem with the presence of noise. In the following sections we assume that $m = \omega(1)$.

6.1 Non-constructive Algorithm

We start this chapter by proving Theorem 11.

Theorem 11. *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 $\mathbb{Z}_p$.

Proof. 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 is a positive constant that is close to "1", say 0.9999. By Lemma 10,

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

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

$$\begin{aligned}
\Pr[r(M'^{[k_1]}) < t] &= \Pr[X_1 + \cdots + X_{k_1} \leq t-1] \\
&= \sum_{\substack{\xi_1+\cdots+\xi_{k_1} \leq t-1, \\ \xi_j \in \{0,1\}}} \Pr[X_1 = \xi_1, \ldots, X_{k_1} = \xi_{k_1}] \\
&\leq \frac{\sum_{i=0}^{t-1} \binom{k_1}{i}}{2^{k_1-t+1}} \leq t 2^{t-1} \frac{\binom{k_1}{t}}{2^{k_1}} \qquad (6.1) \\
&< \frac{t 2^{t-1} \left(\frac{k_1 e}{t}\right)^t}{2^{k_1}} < \frac{n^t}{2^{k_1}} = \frac{2^{\frac{m \log n}{\log^2 m}}}{2^{k_1}} = \frac{1}{2^{(1-o(1))k_1}}. \qquad (6.2)
\end{aligned}$$

Equation (6.1) 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 (6.2) and the union bound, we have

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

Note that for $\epsilon < 0.22$ and c_2 close enough to "1", we have that

$$\Pr[\mathcal{A}] \leq \frac{\binom{n}{t}}{2^{(1-o(1))k_1}} = \frac{1}{2^{(1-o(1))k_1}} = \frac{1}{2^{(1-o(1))(1-\epsilon)k}} < \frac{1}{2 \cdot 2^{H(\epsilon)k}} < \frac{1}{2\binom{k}{k^*}}. \qquad (6.3)$$

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 $\mathcal{B}$.

$$\begin{aligned}
\Pr[\mathcal{B}] &= \Pr[\mathcal{B}|\mathcal{A}] \Pr[\mathcal{A}] + \Pr[\mathcal{B}|\neg\mathcal{A}](1 - \Pr[\mathcal{A}]) \\
&\leq \Pr[\mathcal{A}] + \Pr[\mathcal{B}|\neg\mathcal{A}]. \qquad (6.4)
\end{aligned}$$

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 2. Using Lemma 11, the probability that M'' is of rank smaller than m given $\neg\mathcal{A}$ is,

$$\begin{aligned}
\Pr[r(M'') < m|\neg\mathcal{A}] &\leq \Pr[Y_{k_1} + Y_{k_1+1} + \dots + Y_{k^*} \leq m-1|\neg\mathcal{A}] \\
&\leq \frac{\sum_{i=0}^{m-1}\binom{k^*-k_1}{i}}{q^{k^*-k_1-m+1}} < \frac{2^{k^*-k_1}}{q^{k^*-k_1-m+1}} \qquad (6.5)\\
&= \frac{1}{q^{(k^*-k_1)(1-o(1))-m+1}} \qquad (6.6)\\
&= \frac{1}{q^{(1-\epsilon)(1-c_2)(1-o(1))k-m+1}}.
\end{aligned}$$

Again, in (6.5) we use Lemma 11. In (6.6) 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

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

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

$$\Pr[\mathcal{B}|\neg\mathcal{A}] \leq \frac{1}{q^{O\left(m+\frac{m\log\frac{n}{m}}{\log\min(m,p)}\right)}} < \frac{1}{2^k} < \frac{1}{2\binom{k}{k^*}} \qquad (6.7)$$

Finally, from (6.3), (6.4) and (6.7), we get that the probability that there exists m columns that are linearly independent in M^* is

$$\begin{aligned}
\Pr[\mathcal{B}] &\leq \Pr[\mathcal{A}] + \Pr[\mathcal{B}|\neg\mathcal{A}] \\
&< \frac{1}{2\binom{k}{k^*}} + \frac{1}{2\binom{k}{k^*}} = \frac{1}{\binom{k}{k^*}} \qquad (6.8)
\end{aligned}$$

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

Corollary 5. *Let $\epsilon < 0.22$ be a constant, there exists a non-adaptive algorithm for reconstructing a hidden vector $v \in \mathbb{R}^n$, $wt(v) \leq m$, 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, $e_1 + e_2 < \frac{\epsilon}{2}k$.

6.2 Polynomial Time Algorithm

In this section, we show polynomial time algorithms for reconstructing hidden (0,1)-vectors in the presence of noise. We start with a non-adaptive algorithm for reconstructing a hidden $v \in \{0,1\}^n$.

6.2.1 Fast Algorithm for Reconstructing (0,1)-vectors with Noise

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

Theorem 9. *Let $v \in \{0,1\}^n$ be a hidden vector. There exists a non-adaptive reconstructing algorithm that runs in time $O(n^{3/2})$, tolerates $c(n/(\epsilon \log n))^{\frac{1-\epsilon}{4}}$ errors and erasures, for any constant $c < 1/2$, and reconstructs v using*

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

queries.

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

Lemma 36. *Let $f \in \mathbb{R}^{\{-1,1\}^n}$. If f has s non-zero values, then, there is $q \in \{-1,1\}^n$ where $||q|| \leq \lceil \log s \rceil$ 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,

$$\hat{f}(q) = E[f \cdot \chi_q] = \lambda_1 \chi_q(a_1) + \lambda_2 \chi_q(a_2) + \ldots + \lambda_s \chi_q(a_s) = 0. \tag{6.9}$$

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 (6.9), 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 6. *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 36, this is a contradiction. □

Lemma 37. *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 6.1. 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

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

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

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

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 6.1: Subroutine for finding non-zero values.

Therefore, by Lemma 36, 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 6.2. 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, **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

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

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

□

Now after proving the above lemmas, we are ready to prove our main result, that is, prove Theorem 9.

Our problem is equivalent to constructing a $(0,1)$-matrix M such that given $Mv + e$ where the rows of M are the query vectors, $v \in \{0,1\}^n$ is the hidden vector and e is the error vector, one can reconstruct v. The idea of the construction is the following. We construct a large set of functions $M_i : \{-1,+1\}^\ell \to \{0,1\}$ that their low degree fourier coefficients are zero. Those functions represent the columns of the search matrix M. When the algorithm receives $f = Mv + e$, it regards it as a function $f : \{-1,1\}^\ell \to [n]$. Since the columns M_i of M are functions with zero low degree fourier coefficient, the low degree coefficients of Mv are zero as well. Therefore by finding the low degree fourier coefficients of $f = Mv + e$ we find the low degree fourier coefficients of e. Using the above results, the error e can then be found. Then $f - e = Mv$ can be found and the problem is reduced to the non-noisy case.

We now give a formal proof.

Proof. Consider the matrix M_t defined in Subsection 3.3.1, with $t = \ell 2^{\ell-1}$ where ℓ will be determined later. 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 $||a|| < \ell/2$. Also remove all columns that corresponds to functions $f_{a,k}$ where k is grater than $\epsilon||a||$. Denote the resulting matrix by M'. The number of columns of M' is

$$N_\ell = \sum_{i=\frac{\ell}{2}}^{\ell} \epsilon i \binom{\ell}{i} \geq \epsilon \ell 2^{\ell-2}.$$

We now choose ℓ such that $N_{\ell-1} < n \leq N_\ell$. We now remove arbitrary $N_\ell - n$ columns from M' and get a matrix M'_n with n columns. The number of rows in M'_n is $2^\ell = O(n/\epsilon \log n)$. It is easy to see that if $f_{a,k}$ corresponds to some column in M'_n then all fourier coefficient $\widehat{f_{a,k}}(q)$ of $f_{a,k}$ where $||q|| < (1-\epsilon)||a|| \leq (1-\epsilon)\ell/2$ are equal to zero with the exception of $\widehat{f_{a,k}}(-1^\ell)$ that is equal to $1/2$.

Now, consider the following matrix

$$M^* = \begin{pmatrix} M'_n \\ \hline 1^{n^{1/4} \times n} \end{pmatrix}.$$

where $1^{n^{1/4}\times n}$ is the all-one $n^{1/4} \times n$ matrix. 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 $\frac{v^T 1^n}{2}$. 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 37, 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 Subsection 3.3.1, we are able to reconstruct the hidden vector v. □

6.2.2 Reconstructing (0,1)-vectors with 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 10. *Let $v \in \{0,1\}^n$ be a hidden vector. There exists a non-adaptive polynomial time reconstructing algorithm that tolerates $\Omega(n^{1-\epsilon})$ errors and erasers and reconstructs v using*

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

queries.

Moreover, if the errors and erasures are consecutive, then, the algorithm works correctly even if their number is $\Omega(n/(\epsilon \log n))$.

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 38. *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 Subsection 3.3.1. 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,

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

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

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

Then, using the algorithm in Subsection 3.3.1, we reconstruct the vectors $v_1, v_2, \dots, v_{n^{1-\epsilon}}$ 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

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

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

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

Note that,

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

By Lemma 38, 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 38, we are able from $Bv + e$ to reconstruct all $x^{(i)}$'s. This completes the proof. □

6.2.3 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 12.

Theorem 12. *Let $v \in \{0,1\}^n$ be a hidden vector with at most m nonzero entries. There is a polynomial time adaptive algorithm that tolerates $O(m^{1-\epsilon})$ errors and erasures and reconstructs v using*

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

queries.

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 18. *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 [10]. As mentioned in Subsection 3.3.1, 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 [10] for reconstructing the vector

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

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

$$B = \begin{pmatrix} g_{1,1}M_1 & g_{2,1}M_2 & \ldots & g_{n^{1-\epsilon},1}M_{n^{1-\epsilon}} \\ g_{1,2}M_1 & g_{2,2}M_2 & \ldots & g_{n^{1-\epsilon},2}M_{n^{1-\epsilon}} \\ \vdots & \vdots & \ddots & \vdots \\ g_{1,c_1 n^{1-\epsilon}}M_1 & g_{2,c_1 n^{1-\epsilon}}M_2 & \ldots & g_{n^{1-\epsilon},c_1 n^{1-\epsilon}}M_{n^{1-\epsilon}} \end{pmatrix}.$$

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

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

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 38, 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 7. *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 12.

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 6.3

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 38 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 6.3: Reconstructing with noise. Algorithm's first stage.

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 38, 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 7 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 $\log n$ iterations, 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 7) asks

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

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

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

This completes the proof. □

Bibliography

[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] N. Alon, R. Beigel, S. Kasif, S. Rudich and B. Sudakov. Learning a Hidden Matching. *SIAM J. Comput.* 33, 2, 487–501, 2004.

[4] D. Angluin. and J. Chen. Learning a Hidden Graph Using $O(\log n)$ Queries per Edge. *Conference on Learning Theory*, 210–223, 2004.

[5] D. Angluin amd J. Chen. Learning a Hidden Hypergraph. *Journal of Machine Learning Research*, 7, 2215–2236, 2006.

[6] B. Bollobás, Random Graphs, second ed., Cambridge Stud. Adv. Math., vol. 73, Cambridge Univ. Press, Cambridge, 2001.

[7] 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.

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

[9] M. Bouvel, V. Grebinski, G. Kucherov: Combinatorial Search on Graphs Motivated by Bioinformatics Applications: A Brief Survey. *WG*, 16–27, 2005.

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

[11] N. H. Bshouty and H. Mazzawi. Reconstructing Weighted Graphs with Minimal Query Complexity. *ALT*. LNCS (LNAI) pp. 97–109, 2009. Journal version appears in *Theoretical Computer Science*, Volume 412, Issue 19, April, 2011.

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

[13] N. H. Bshouty and H. Mazzawi. Toward a Deterministic Polynomial Time Algorithm with Optimal Additive Query Complexity. *MFCS*, 2010. Journal version will appear in *Theoretical Computer Science*.

[14] N. H. Bshouty and H. Mazzawi. Optimal Query Complexity for Reconstructing Hypergraphs. *STACS*, 2010.

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

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

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

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

[19] 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.

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

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

[22] S. Choi, K. Jung, J. H. Kim. Almost Tight Upper Bound for Finding Fourier Coefficients of Bounded Pseudo- Boolean Functions. COLT 2008, 123-134, 2008.

[23] 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.

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

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

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

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

[28] 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.

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

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

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

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

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

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

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

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

[37] 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.

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

[39] H. Mazzawi. Optimally Reconstructing Weighted Graphs Using Queries. *Symposium on Discrete Algorithms*, 608–615, 2010.

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

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

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

[43] L. Reyzin and N. Srivastava. Learning and Verifying Graphs using Queries with a Focus on Edge Counting. *ALT*. LNAI 4754, pp. 277?-289, 2007.

[44] M. Ruszinkó, P. Vanroose. How an Erdős-Rényi-type search approach gives an explicit code construction of rate 1 for random access with multiplicity feedback. *IEEE Transactions on Information Theory*. 43(1). 368–372, 1997.

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

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

Appendix A: $(0,1)$-Matrices with Rows that are Tensor Product of Vectors

In this section we show that there is an m-independent column $t \times n$ $(0,1)$-matrix that its rows are tensor product of d $(0,1)$-vectors in $\{0,1\}^{n^{1/d}}$ and

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

We start with some notation. Some of the notation and definitions below were already presented in Chapter 5. We presented them again for the reader's convenience.

A d-dimensional matrix A of size $n_1 \times \cdots \times n_d$ over a field F is a map $A : \prod_{i=1}^{d}[n_i] \to F$. We denote by $F^{n_1 \times \cdots \times n_d}$ the set of all d-dimensional matrices A of size $n_1 \times \cdots \times n_d$. We write $A_{i_1,\ldots,i_d}$ for $A(i_1,\ldots,i_d)$. The zero map is denoted by $0^{n_1 \times \cdots \times n_d}$. For $I_j \subseteq [n_j]$, the matrix $B = (A_{i_1,i_2,\ldots,i_d})_{i_1 \in I_1, i_2 \in I_2, \ldots, i_d \in I_d}$ is the $|I_1| \times \cdots \times |I_d|$ matrix where $B_{j_1,\ldots,j_d} = A_{\ell_1,\ldots,\ell_d}$ where ℓ_i is the j_ith smallest number in I_i. When $I_j = [n_j]$ we just write j and when $I_j = \{\ell\}$ we just write $j = \ell$. For example $(A_{i_1,i_2,\ldots,i_d})_{i_1,i_2=\ell,i_3 \in I_2,\ldots,i_d \in I_d} = (A_{i_1,i_2,\ldots,i_d})_{i_1 \in [n_1], i_2 \in \{\ell\}, i_3 \in I_2,\ldots,i_d \in I_d}$

When $n_1 = n_2 = \cdots = n_d = n$ then we denote $F^{n_1 \times \cdots \times n_d}$ by $F^{\times_d n}$ and $0^{n_1 \times \cdots \times n_d}$ by $0^{\times_d n}$. For d-dimensional matrix A we denote by $wt(A)$ the number of points in $\prod_{i=1}^{d}[n_i]$ that are mapped to non-zero elements in F. For d-dimensional matrix A of size $n_1 \times \cdots \times n_d$ and $x_i \in F^{n_i}$ we define

$$A(x_1,\ldots,x_d) = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} A_{i_1,i_2,\ldots,i_d} x_{1i_1} \cdots x_{di_d}.$$

The vector $v = A(\cdot, x_2,\ldots,x_d)$ is n_1-dimensional vector that its i_1 entry is

$$\sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} A_{i_1,i_2,\ldots,i_d} x_{2i_2} \cdots x_{di_d}.$$

We first prove the following:

Theorem 19. *Let $p < n^\gamma$ be a prime number for some constant γ. There exists a set $S = \{(x_{11}, \ldots, x_{1d}), (x_{21}, \ldots, x_{2d}), \ldots, (x_{k1}, \ldots, x_{kd})\}$ where $x_{ij} \in \{0,1\}^n$ and*

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

such that: for any d-dimensional matrix $A \in \mathbb{Z}_p^{\times_d n} \backslash \{0^{\times_d n}\}$ with $wt(A) \leq m$, there exists an i such that

$$A(x_{i1}, \ldots, x_{id}) \neq_p 0.$$

Proof. We divide the set of matrices

$$\mathcal{A} = \{A \mid A \in \mathbb{Z}_p^{\times_d n} \backslash \{0^{\times_d n}\} \text{ and } wt(A) \leq m\}$$

into $d+1$ (non-disjoint) sets:

- $\mathcal{A}_0$: The set of all matrices $A \in \mathbb{Z}_p^{\times_d n} \backslash \{0^{\times_d n}\}$ such that $wt(A) \leq m/\log m$.
- $\mathcal{A}_j, j = 1, \ldots, d$: The set of all matrices $A \in \mathbb{Z}_p^{\times_d n}$ such that $m \geq wt(A) > m/\log m$ and there are at least

  $$\left(\frac{m}{\log m}\right)^{1/d}$$

  non-zero elements in

  $$I_j = \{i_j \mid \exists (i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_d)\ A_{i_1, i_2, \ldots, i_d} \neq 0\}.$$

Note that $I = \{(i_1, \ldots, i_j) \mid A_{i_1, \ldots, i_j} \neq 0\} \subseteq I_1 \times I_2 \times \cdots \times I_d$ and therefore either $|I| = wt(A) \leq m/\log m$ or there is j such that $|I_j| > (m/\log m)^{1/d}$. Therefore, $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_d$.

Using the probabilistic method, we give $d+1$ sets of d-tuples of vectors $S_0, S_1, \ldots, S_d$ such that for every $j \in \{0\} \cup [d]$ and $A \in \mathcal{A}_j$ there exists a d-tuple of vectors $(x_1, \ldots, x_d) \in S_j$ such that $A(x_1, \ldots, x_d) \neq_p 0$ and

$$|S_0| + |S_1| + \cdots + |S_d| = O\left(m + \frac{m \log \frac{n^d}{m}}{\log \min(m, p)}\right).$$

Case 1: $A \in \mathcal{A}_0$.

As in the proof of Lemma 1 it can be shown that for randomly chosen vectors $x_{i1}, \ldots, x_{id} \in \{0,1\}^n$ and $A \in \mathcal{A}_0$ we have

$$\Pr[A(x_{i1}, x_{i2}, \ldots, x_{id}) =_p 0] \leq \frac{2^d - 1}{2^d}.$$

Randomly uniformly choose

$$k_1 = c\left(m + \frac{m \log \frac{n^d}{m}}{\log \min(m, p)}\right)$$

d-tuples of $(0,1)$-vectors $x_i = (x_{i1}, \ldots, x_{id}) \in (\{0,1\}^n)^d$ where c is a constant. The probability that for all x_i we have $A(x_i) =_p 0$ is bounded by

$$\Pr[\forall i \in [k_1] : A(x_i) =_p 0] \leq \left(\frac{2^d - 1}{2^d}\right)^{k_1}.$$

Therefore, by union bound, the probability that there exists a matrix A of weight smaller than $m/\log m$ such that $A(x_i) =_p 0$ for all $i \in [k_1]$ is

$$\begin{aligned}\Pr[\exists A \in \mathcal{A}_0, \forall i \in &[k_1] : A(x_i) =_p 0] \\ &\leq \binom{n^d}{\frac{m}{\log m}} p^{\frac{m}{\log m}} \left(\frac{2^d - 1}{2^d}\right)^{k_1} < 1,\end{aligned}$$

for some constant c. This implies the result.

Case 2: $A \in \mathcal{A}_1$.

We start by proving the following two lemmas,

Lemma 39. *Let $U \subset \mathbb{Z}_p^{\times d-1 n}$ be the set of all non-zero $d-1$-dimensional matrices with weight smaller than $m^{d/(d+1)}$. Then there is a constant c_0 such that for any constant $C > c_0$ and*

$$k_2 = C\left(m + \frac{m \log \frac{n^d}{m}}{\log \min(m, p)}\right)$$

there exists a multiset of $d-1$-tuple of (0,1)-vectors $Y = \{y_1, y_2, \ldots, y_{k_2}\} \subseteq (\{0,1\}^n)^{d-1}$ such that for every $A \in U$ the size of the multiset

$$Y_A = \{i \mid A(y_i) \neq_p 0\}$$

is at least $\frac{k_2}{2^d}$.

Proof. As above for a randomly chosen vector $y \in (\{0,1\}^n)^{d-1}$ and any $A \in U$ we have

$$\Pr[A(y) =_p 0] \leq \frac{2^{d-1}-1}{2^{d-1}}.$$

Therefore, if we randomly uniformly choose the vectors of Y, then the expected size of Y_u is greater than $k_2/2^{d-1}$ for any $A \in U$. Using Chernoff bound we have that

$$\Pr[|Y_A| < k_2/2^d] \leq e^{\frac{k_2}{2^{d+2}}}.$$

Therefore, the probability that there exists $A \in U$ such that $|Y_A| < k_2/2^d$ is

$$\begin{aligned}\Pr[\exists A \in U : |Y_A| < k_2/2^d] &\leq \frac{|U|}{e^{\frac{k_2}{2^d}}} \\ &\leq \frac{\sum_{i=0}^{m^{d/(d+1)}} \binom{n^{d-1}}{i}(p-1)^i}{e^{\frac{k_2}{2^d}}} \\ &< 1,\end{aligned}$$

for some constant c_0 and all $C > c_0$. This implies the result. □

Now let U be the set of $d-1$-dimensional matrices defined in Lemma 39. Let $A \in \mathcal{A}_1$. Since $wt(A) \leq m$ there are at most $m^{1/(d+1)}$ $d-1$-dimensional matrices $(A_{i_1,i_2,\ldots,i_d})_{i_1=j,i_2,\ldots,i_d}$ with weight greater than $m^{d/(d+1)}$. Therefore, there are at least

$$q = \left(\frac{m}{\log m}\right)^{1/d} - m^{1/(d+1)}$$

indices j such that $(A_{i_1,i_2,\ldots,i_d})_{i_1=j,i_2,\ldots,i_d} \in U$. Let U' contains q indices j such that $(A_{i_1,i_2,\ldots,i_d})_{i_1=j,i_2,\ldots,i_d} \in U$. Let A_U be the matrix $(A_{i_1,i_2,\ldots,i_d})_{i_1 \in U',i_2,\ldots,i_d}$. Let $Y = \{y_1, y_2, \ldots, y_{k_2}\}$ be the set we proved its existence in Lemma 39. Note that

$$\sum_i wt(A_U(\cdot, y_i)) \geq \frac{qk_2}{2^d}.$$

Since $wt(A_U(\cdot, y_i)) \leq q$ for all $i \in [k_2]$ and by Lemma 20 we have

$$\begin{aligned}\prod_i \iota(wt(A_U(\cdot, y_i))) &\geq (\min(q,p))^{\left\lfloor \frac{\frac{qk_2}{2^d} - k_2}{q-1} \right\rfloor} \\ &= (\min(q,p))^{c_1 k_2} \\ &= (\min(m,p))^{c_2 k_2},\end{aligned}$$

where c_1 and c_2 are constants. If we randomly choose $x_1, x_2, \ldots, x_{k_2}$ then by Lemma 2, we have

$$\begin{aligned}
\Pr[\forall i \in [k_2] : A(x_i, y_i) \neq 0] &\leq \prod_i \frac{1}{\iota(wt(A(\cdot, y_i)))^\beta} \\
&\leq \prod_i \frac{1}{\iota(wt(A_U(\cdot, y_i)))^\beta} \\
&= \left(\frac{1}{\prod_i \iota(wt(A_U(\cdot, y_i)))}\right)^\beta \\
&\leq \frac{1}{(\min(m,p))^{\beta c_2 k_2}} \\
&= (\min(m,p))^{-c_3 k_2}
\end{aligned}$$

where c_3 is a constant. Therefore, the probability that there exists a matrix $A \in \mathcal{A}_1$ such that for all x_i, y_i we have $A(x_i, y_i) =_p 0$ is

$$\begin{aligned}
\Pr[\exists A \in \mathcal{A}_1, \forall i \in [k_2] : A(x_i, y_i) =_p 0] &\leq \frac{|\mathcal{A}_1|}{(\min(m,p))^{c_3 k_2}} \\
&\leq \frac{\binom{n^d}{m} p^m}{(\min(m,p))^{c_3 k_2}} \\
&< 1,
\end{aligned}$$

for constant some constant C. Thus, the results follows.

□

Now we get our main result

Corollary 8. *There exists a matrix $M \in \{0,1\}^{k \times n}$ where*

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

every row of M is a tensor product of d vectors $x_1, \ldots, x_d \in \{0,1\}^{n^{1/d}}$ and every m columns of M are linearly independent over $\mathbb{Z}_p$.

כאשר $r(G')$ הינו קבוע. בדומה לתוצאה הקודמת, האלגוריתם תואם את החסם התחתון לסיבוכיות השאילתות גם כן.

על מנת להשיג את התוצאות המוזכרות לעיל, אנו משתמשים במספר טכניקות ושיטות מהתיאוריה של מדעי המחשב. שיטות כגון: השיטה ההסתברותית, ניתוח מקדמי פורייה של פונקציות, שיטת ניסוי ותהייה, שיטות מתורת הצפינה ועוד.

$$. O\left(\frac{|E| \log \frac{|V|^2}{|E|}}{\log |E|} \right)$$

עבור בעיית שחזור הגרפים משאילתות, אנו מראים קיום אלגוריתם לא אדפטיבי אשר משחזר גרפים ממשקלים. האלגוריתם משיג את החסם התחתון של סיבוכיות שאילת שאילתות. בנוסף, אנו מראים אלגוריתם ראשון אשר משחזר גרפים לא ממושקלים, משיג את החסם התחתון לסיבוכיות השאילתות ורץ בזמן פולינומי. את התוצאה הזו אנו מרחיבים בכך שאנו מראים אלגוריתם שרץ בזמן פולינומי, משחזר גרפים ממושקלים עם סיבוכיות שאילתות כמעט הדוקה. סיבוכיות השאילתות תואמת את החסם התחתון עבור גרפים עם מעט קשתות, ורחוקה פקטור

$$\log \log |E|$$

מהחסם התחתון לכל גרף.

לבעיית שחזור הגרפים משאילתות אדיטיביות אפליקציות רבות מתחום הביו-אינפורמטיקה. הניחו כי בידינו קבוצה של חומרים כימים. הניחו כי אנו מסוגלים לקבוע כמה זוגות מגיבים אחד בסביבת השני ברגע שאנו מערבבים תת קבוצה של החומרים הכימיים שבידינו. אפשר להסתכל על הבעיה כעל בעיית שחזור גרף כאשר החומרים הכימיים הם הצמתים. כל זוג חומרים שמגיב אחד לשני מחובר בקשת. והמטרה לשחזר את גרף הריאקציות הנ"ל על ידי ניסויים כאשר בכל ניסוי, אנו מערבבים תת-קובצה של החומרים הכימיים ביחד.

דוגמא קונקרטית לשחזור גרף ריאקציות ניתן למצא מתחום ריצוף הגינום. השגת רצף הגינום המלא הינו דבר חיוני עבור חקר אורגניזמים. בגלל מגבלות טכנולוגיות והגדלים של הגנום, אחת השיטות למציאת רצף הגינום היא על ידי קריאת פענוח חלקים קצרים של הגנום וחיבורם ביחד. בחלק מתהליך החיבור יש בידינו קטעים רציפים המכסים את כל הגנום, והמטרה היא לדעת את מיקומם היחסי של הקטעים ברצף הגינום. לפתרון בעיה מציאת המיקום משתמשים במתודה אשר מקבלת תת-קבוצה של הקטעים המוזכרים ומחזירה את מספר הזוגות של קטעים שכנים בתת קבוצה. בהינתן שהגנום מעגלי, הבעיה של מציאת מיקומם היחסי של הקטעים ברצף הגינום דרך המתודה המוזכרת שקול לשחזור גרף המילטוני משאילתות אדיטיביות.

לבסוף, אנו חוקרים את בעיית שחזור היפר-גרפים משאילתות אדיטיביות. בבעיה זו ישנו היפר-גרף חבוי $G' = (V', E', w')$, כאשר $E' \subseteq 2^{V'}$ ו- $w' \in R^{E'}$. בדומה לבעיה הקודמת, קבוצת הצמתים ידועה לכל ולעומתה קבוצת הקשתות ומשקולותיהם אינה ידועה. עלינו לשחזר את קבוצת הקשתות ומשקולותיהם על ידי שאילת שאילתות מהצורה

$$Q_{G'} = \sum_{e \in E \cap 2^S} w'(e)$$

כאשר S היא תת-קבוצה של V'. זאת אומרת, השאילתה מחזירה את סכום המשקולות בתת גרף של S.

נסמן ב-$r(G')$ את גודל הקשת הגדולה ביותר ב-G'. עבור בעיית שחזור היפר-גרפים משאילתות, אנו מראים קיום אלגוריתם לא אדפטיבי אשר משחזר היפר גרף לא ממושקל כלשהו G' בתנאי ש- $r(G')$ הינו קבוע (זאת אומרת $r(G') = O(1)$). האלגוריתם תואם את החסם התחתון של סיבוכיות השאילתות לבעיה. בנוסף, אנו מראים קיום אלגוריתם לא אדפטיבי אשר משחזר היפר-גרף ממושקל כלשהו G'

שאילתות. הוא רץ בזמן פולינומי ומשחזר נכונה את הוקטור החבוי גם אם $O\left(n^{1-\varepsilon}\right)$ מהתשובות שקיבל שגויות. בנוסף, אנו מראים כיצד ליצור אלגוריתם אשר שואל

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

שאילתות ומשחזר וקטור של אפסים ואחדים בגודל n עם לכל היותר m אחדים. האלגוריתם רץ בזמן פולינומי ומשחזר נכונה את הוקטור החבוי גם אם $O\left(m^{1-\varepsilon}\right)$ מהתשובות שקיבל שגויות.

לבעיית המטבעות המזויפים אפליקציות רבות בתעשיה, אחת מהן היא בעיית העברת מסרים דרך ערוץ חיבורי. נניח ישנן n תחנות. לכל תחנה i ישנו קוד $C_i = \{0^k, x_i\} \subset \{0,1\}^k$. בכל איטרציה, כל תחנה מנסה להעביר מסר $y_i \in C_i$ דרך הערוץ. הניחו כי קיים מנגנון סינכרון בין התחנות. ידוע גם כי בכל איטרציה לכל היותר m תחנות שולחות מסר ששונה מאפס (זאת אומרת, 0^k). המסרים מועברים דרך מחבר (מעל שדה F) אשר מעביר דרך הערוץ את חיבור המסרים שקיבל מהתחנות. זאת אומרת,

$$Y = \sum_i y_i .$$

ראה תמונה 1.1. על המפענח לשחזר נכונה את כל המסרים אותם שלחו התחנות מהמילה Y. בעיה זו שקולה לבעיית המטבעות המזויפים בגרסה הלא אדפטיבית מעל שדה F. קיום של אלגוריתם לא אדפיטיבי לפתרון בעיית המטבעות המזויפים עם סיבוכיות שאילתות k שקול לקיום קוד באורך k $C = \{C_1, C_2, ..., C_n\}$ אשר פותר את בעיית העברת המסרים דרך ערוץ חיבורי (הכוונה באורך k היא ש- $C_1, C_2, ..., C_n \subset \{0,1\}^k$). זאת ועוד, אם האלגוריתם לבעיית המטבעות המזויפים מתגבר על q תשובות שגויות, אזי הדבר גורר קוד C באורך k אשר ניתן לפענוח גם אם q מהכניסות של Y (המילה שהתקבלה דרך הערוץ) שגויות.

באשר לבעיית שחזור הגרפים משאילתות אדיטיביות, בבעיה זו ישנו גרף חבוי $G = (V, E, w)$, כאשר E היא תת-קבוצה של VxV ו- w הינה פונקציה מקבוצת הקשתות E לקבוצת המספרים הממשיים R. קבוצת הצמתים בגרף החבוי ידועה לכול. המטרה הינה לשחזר את קבוצת הקשתות ומשקולותיהן על ידי שאילת שאילתות מהצורה

$$Q_G(S) = \sum_{e \in E \cap (S \times S)} w(e),$$

כאשר S הינה תת-קבוצה של קבוצת הצמתים V. במילים אחרות, השאילתה מחזירה את סכום המשקולות בתת גרף של S (בגרפים לא ממושקלים, השאילתה מחזירה את מספר הקשתות בתת גרף של S). החסם התחתון לסיבוכיות השאילתות של שחזור גרף ממושקל הינה

$$O\left(\frac{|E|\log|V|}{\log|E|}\right).$$

לעומת גרפים ממושקלים, החסם התחתון של סיבוכיות השאילתות עבור שחזור גרפים שאינם ממושקלים הוא

תקציר

בתזה זו אנו חוקרים שלוש בעיות חיפוש קומבינטורי. בעיית המטבעות המזויפים הידועה גם בכינויה בעיית שחזור ווקטורים משאילתות אדיטיביות, בעיית שחזור גרפים משאילתות אדיטיביות, ובעיית שחזור היפר-גרפים משאילתות אדיטיביות.

לשלושת הבעיות המוזכרות לעיל מבנה משותף. בכולן יש "עולם" או "מרחב עצמים". מתוך מרחב העצמים, נבחר עצם ייחודי לו קוראים ה"עצם החבוי". על מנת לפתור בעיית חיפוש קומבינטורי, עלינו לשחזר או למצוא את העצם החבוי על ידי שאילת שאילתות מצורה ידועה מראש.

בתזה זו אנו מבדילים בין שני סוגי אלגוריתמים לפתירת הבעיות. אלגוריתמים לא אדפטיביים שהם אלגוריתמים ששואלים את כל השאילתות מראש עוד לפני שקיבלו תשובות. ולעומתם אלגוריתמים אדפטיביים לוקחים בחשבון את התשובות לשאילתות קודמות בזמן שאילת שאילתה.

בבעיית המטבעות המזויפים, ישנו ווקטור v מעל שדה F בגודל n עם לכל היותר m כניסות ששונות מאפס. מותר לשאול שאילתות מהצורה

$$Q_v(x) = x^{\mathrm{T}} v$$

כאשר x הוא ווקטור של אפסים ואחדים בגודל n. המטרה היא לשחזר את הוקטור החבוי על ידי שאילת שאילתות. החסם התחתון לסיבוכיות השאילתות של פתירת בעיה זו עבור שדה הממשיים R הוא

$$O(m \log n/ \log m).$$

לעומת זאת החסם התחתון לבעיה מעל השדה הסופי Z_p (שדה השלמים מודולו הראשוני p) הוא

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

אנו מראים כי קיים אלגוריתם לא אדפטיבי לפתרון הבעיה עם סיבוכיות שאילתות שתואמת את החסם התחתון עבור השדה R והשדות Z_p לכל ראשוני p. בנוסף, אנו מראים אלגוריתם עם זמן ריצה פולינומי לבעיה מעל השדה R. אלגוריתם זה תואם את החסם התחתון עבור m-ים קטנים, הוא כמעט תואם את החסם עבור כל m (רחוק פקטור $\log \log m$ מהחסם).

בנוסף, אנו מראים טכניקות אשר הופכות את האלגוריתמים שלנו לפתרון בעיית המטבעות המזויפים לעמידים בפני רעש. זאת אומרת, האלגוריתמים משחזרים נכונה את הווקטור החבוי גם אם חלק מהתשובות שקיבלו הינו שגוי. אנו מראים כיצד ליצור אלגוריתם לא אדפטיבי אשר משחזר וקטור חבוי של אפסים ואחדים בגודל n (מעל R). האלגוריתם שואל

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

המחקר נעשה בהנחיית פרופ' נאדר בשותי מהפקולטה למדעי המחשב.

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

למידת גרפים משאילתות

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר

דוקטור לפילוסופיה

חנא מזאוי

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

תמוז – תשע''א חיפה יולי 2011

למידת גרפים משאילתות

חנא מזאוי