A Framework for Fast and Efficient Algorithms for Sparse Recovery Problems

CAI, Sheng

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Information Engineering

The Chinese University of Hong Kong
May 2015

Abstract of thesis entitled:

A Framework for Fast and Efficient Algorithms for Sparse Recovery Problems

Submitted by CAI, Sheng

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in May 2015

The sparse recovery problem aims to reconstruct a high-dimensional sparse signal from its low-dimensional measurements given a carefully designed measuring process. This thesis presents a framework for graphical-model-based sparse recovery algorithms. Differing measurement processes lead to specific problems. The sparse recovery problems studied in this thesis include compressive sensing, network tomography, group testing and compressive phase retrieval. For compressive sensing and network tomography, the measurement processes are linear (freely chosen, and topology-constrained measurements respectively). For group testing and compressive phase retrieval, the processes are non-linear (disjunctive, and intensity measurements respectively). For all the problems in this thesis, we present algorithms whose measurement structures are based on bipartite graphs. By studying the properties of bipartite graphs and designing novel measuring process and corresponding decoding algorithms, the number of measurements and computational decoding complexities of all the algorithms are information-theoretically either order-optimal or nearly order-optimal.

摘要

稀疏還原問題旨在通過精心設計的低維度度量重建高維度稀疏信號。這篇論文提出了一個基於圖模型的稀疏還原演算法的框架。研究的稀疏還原問題包括了壓縮感知，網路斷層掃描，組測試和壓縮相位恢復。對於壓縮感知和網路斷層掃描，度量過程是線性的（分別是無約束的度量和拓撲結構約束的度量）。對於組測試和壓縮相位恢復，度量過程是非線性的（分別是邏輯度量和強度度量）。對於提到的問題，這篇論文提出的演算法的度量結構基於二部圖。通過學習二部圖的性質，我們提出了新穎的度量方法和相對應的解碼演算法。對於這些演算法，它們的度量維度和解碼演算法的運算複雜度都是（或接近於）資訊理論最優解。

Acknowledgement

First, I would like to express my sincere gratitude to my supervisors Prof. Sidharth Jaggi, Prof. Mayank Bakshi and Prof. Minghua Chen for the support of my PhD study and research, for their consideration, patience, motivation, enthusiasm, and immense knowledge. Without their help and brilliant ideas, I could not have published several research papers, presented our works in academic conferences, visited California Institute of Technology, and finished this thesis.

Besides my supervisors, I would like to thank the co-authors of my publications: Mohammad Jahangoshahi, Chun Lam Chan and Prof. Vankatesh Saligrama for their insightful discussions on our works and great efforts to improve the quality of this thesis.

Next, I would like to thank the committee members of my thesis defense: Prof. Anthony Man-Cho So, Prof. Pascal Vontobel, Prof. Kenneth Shum and Prof. Babak Hassibi for attending my oral examination and providing useful comments on my thesis.

Also, I would like to thank my labmates in The Chinese University of Hong Kong. To name a few: Qiwen Wang, Pak Hou Che, Chun Lam Chan, as well as former members Ziyu Shao, Shaoquan Zhang, Jihang Ye, Zhe Zhu, Xiangwen Chen, Tan Lu for all the time we have spent together in the years.

Last but not least, I would like to thank my wife, parents and parents-

in-law. I am sorry for not observing filial piety as I have been pursuing PhD degree in Hong Kong and leaving them in Shanghai for more than 4 years. My family always support and encourage me to pursue my dreams.

This work is dedicated to my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we study problems where we wish to estimate a high-dimensional sparse signal from its low-dimensional measurements. Suppose $\mathbf{x}$ is any length-n input vector. We say $\mathbf{x}$ is *k-sparse* if and only if there are exactly k ($k < n$) entries of $\mathbf{x}$ are non-zero. The function $A(\cdot)$ is a measurement process mapping a length-n vector to a length-m (m is typically chosen to satisfy $k < m < n$) output vector. The measurement process may be linear or non-linear for specific problems. For compressive sensing and network tomography, the measurement processes are linear. For group testing and compressive phase retrieval, the measurement processes are non-linear. The measurement process may be noiseless or noisy. In this thesis, we study two sources of noise – source tail and measurement noise. The source tail, $\mathbf{z}$, is a length-n vector containing small components outside the support of $\mathbf{x}$. The measurement noise, $\mathbf{e}$, is a length-m vector. In general, the measurement process may be adaptive (the measurement design may depend on the outcome of a previous measurement) or non-adaptive (each measurement is performed independent of the outcome of other measurements). The performance of adaptive algorithms is in general better than that of non-adaptive algorithms, since the decoder has more information. How-

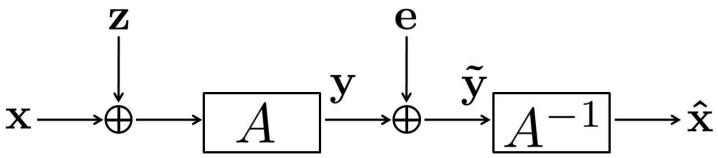


Figure 1.1: Block diagram for sparse recovery problems. $\mathbf{x}$ is a length-n k-sparse input vector. $\mathbf{y}$ is a length-m measurement vector. $\mathbf{z}$ is a length-n source tail. $\mathbf{e}$ is a length-m measurement noise. Here, A is the measurement process and A^{-1} is the corresponding reconstruction algorithm taking noisy measurement $\tilde{\mathbf{y}}$ as input and outputting $\hat{\mathbf{x}}$ as the estimate of $\mathbf{x}$.

ever, non-adaptive algorithms are often preferred to adaptive algorithms in applications, since they allow for parallelizable implementation and/or the usage of off-the-shelf hardware. The goal of sparse recovery problems is first to design the measurement process, A, based on constraints for the specific measurement model and second, given measurement ($\mathbf{y} = A(x)$ in the noiseless case and $\tilde{\mathbf{y}} = A(\mathbf{x} + \mathbf{z}) + \mathbf{e}$ in the noisy case) to produce a reconstruction $\hat{\mathbf{x}}$ of $\mathbf{x}$. Here, "+" is not restricted to be arithmetic addition, especially for the group testing problem introduced below. See Figure 1.1 for a block diagram for sparse recovery problems.

In the rest of this chapter, we introduce four sparse recovery problems discussed in this thesis. See Table 1.1 for short summary.

1.1 Compressive Sensing – SHO-FA

Suppose $\mathbf{x}$ is any exactly k-sparse vector in $\mathbb{R}^n$. We present a class of "sparse" matrices A, and a corresponding algorithm that we call SHO-FA (for Short and Fast[1]) that, with high probability over A, can reconstruct $\mathbf{x}$ from $A\mathbf{x}$. The SHO-FA algorithm is related to the Invertible Bloom Lookup Tables (IBLTs) recently introduced by Goodrich *et al.*, with two important

[1] Also, SHO-FA sho good! In fact, it is all $\mathcal{O}(k)$.

Problem	Operation	Algorithm	$\mathbf{x}$	$\mathbf{y}$	$\mathbf{z}$	$\mathbf{e}$
Compressive Sensing	Arithmetic	SHO-FA	$\mathbb{R}^n$	$\mathbb{C}^m$	Gaussian	Gaussian
Network Tomography	Arithmetic	FRANTIC	$\mathbb{R}^n_{\geq 0}$	$\mathbb{R}^m_{\geq 0}$	No	No
Group Testing	Logic	GROTESQUE	$\{0,1\}^n$	$\{0,1\}^m$	No	Bernoulli
Phase Retrieval	Intensity	SUPER	$\mathbb{C}^n$	$\mathbb{R}^m_{\geq 0}$	No	No

Table 1.1: **Problems considered in this thesis:** For the compressive sensing problem, although the input signal is a length-n vector over real numbers, the measurements designed by SHO-FA are over complex numbers. For the network tomography problem, the linear measurements are constrained by network topology. Further, each measurement is a weighted sum of non-negative real values (represented by $\mathbb{R}_{\geq 0}$) with positive integer weights. For the phase retrieval problem, measurements have non-negative values even if the input signal is over complex numbers since we only measure the intensity of the linear measurements. In the compressive sensing problem, we analyze the effects of both the source tail and measurement noise. In the group testing problem, we only consider the effect of measurement noise. For the other two problems, while we do not consider either source tail, or measurement noise, techniques from our analysis of compressive sensing and group testing can in general be used here.

distinctions – SHO-FA relies on linear measurements, and is robust to noise. The SHO-FA algorithm is the first to simultaneously have the following properties: (a) it requires only $\mathcal{O}(k)$ measurements, (b) the bit-precision of each measurement and each arithmetic operation is $\mathcal{O}\left(\log(n) + P\right)$ (here 2^{-P} corresponds to the desired relative error in the reconstruction of $\mathbf{x}$), (c) the computational complexity of decoding is $\mathcal{O}(k)$ arithmetic operations and that of encoding is $\mathcal{O}(n)$ arithmetic operations, and (d) if the reconstruction goal is simply to recover a single component of $\mathbf{x}$ instead of all of $\mathbf{x}$, with significant probability over A this can be done in constant time. All constants above are independent of all problem parameters other than the desired probability of success. For a wide range of parameters these properties are information-theoretically order-optimal. In addition, our SHO-FA algorithm works over fairly general ensembles of "sparse random matrices",

is robust to random noise, and (random) approximate sparsity for a large range of k. In particular, suppose the measured vector equals $A(\mathbf{x}+\mathbf{z})+\mathbf{e}$, where $\mathbf{z}$ and $\mathbf{e}$ correspond respectively to the *source tail* and *measurement noise*. Under reasonable statistical assumptions on $\mathbf{z}$ and $\mathbf{e}$ our decoding algorithm reconstructs $\mathbf{x}$ with an estimation error of $\mathcal{O}(||\mathbf{z}||_2 + ||\mathbf{e}||_2)$. The SHO-FA algorithm works with high probability over A, $\mathbf{z}$, and $\mathbf{e}$, and still requires only $\mathcal{O}(k)$ steps and $\mathcal{O}(k)$ measurements over $\mathcal{O}(\log(n))$-bit numbers. This is in contrast to most existing algorithms which focus on the "worst-case" $\mathbf{z}$ model, where it is known $\Omega(k\log(n/k))$ measurements over $\mathcal{O}(\log(n))$-bit numbers are necessary. Our algorithm has good empirical performance, as validated by simulations.[2]

1.2 Network Tomography – FRANTIC

We study the problem of link and node delay estimation in undirected networks when at most k out of n links or nodes in the network are congested. Our approach relies on end-to-end measurements of path delays across prespecified paths in the network. We present a class of algorithms that we call FRANTIC .[3] The FRANTIC algorithms are motivated by compressive sensing; however, unlike traditional compressive sensing, the measurement design here is constrained by the network topology, and the matrix entries are constrained to be positive integers. A key component of our design is a new compressive sensing algorithm SHO-FA-INT that is related to the SHO-FA algorithm [10] for compressive sensing, but unlike SHO-FA, the ma-

[2] A preliminary version of this work was presented in [10]. The journal version of this work has been accepted for publication in the IEEE Transactions on Information Theory. In parallel and independently of this work, an algorithm with very similar design and performance was proposed and presented in [112].

[3] FRANTIC stands for **F**ast **R**eference-based **A**lgorithm for **N**etwork **T**omography v**I**a **C**ompressive sensing.

trix entries here are drawn from the set of integers $\{0, 1, \ldots, M\}$. We show that $\mathcal{O}(k \log(n)/\log(M))$ measurements suffice both for SHO-FA-INT and FRANTIC . Further, we show that the computational complexity of decoding is also $\mathcal{O}(k \log(n)/\log(M))$ for each of these algorithms. Finally, we look at efficient constructions of the measurement operations through Steiner Trees.[4]

1.3 Group Testing – GROTESQUE

Group-testing refers to the problem of identifying (with high probability) a (small) subset of k defectives from a (large) set of n items via a "small" number of "pooled" tests (*i.e.*, tests that have a positive outcome if at least one of the items being tested in the pool is defective, else have a negative outcome). For ease of presentation in this work we focus on the regime when $k = \mathcal{O}\left(n^{1-\delta}\right)$ for some $\delta > 0$. The tests may be *noiseless* or *noisy*, and the testing procedure may be adaptive (the pool defining a test may depend on the outcome of a previous test), or non-adaptive (each test is performed independent of the outcome of other tests). A rich body of literature demonstrates that $\Theta(k \log(n))$ tests are information-theoretically necessary and sufficient for the group-testing problem, and provides algorithms that achieve this performance. However, it is only recently that reconstruction algorithms with computational complexities that are sublinear in n have started being investigated (recent work by [72, 77, 105] gave some of the first such algorithms). In the scenario with adaptive tests with noisy outcomes, we present the first scheme that is simultaneously order-optimal (up to small constant factors) in *both* the number of tests and the decoding complexity ($\mathcal{O}(k \log(n))$ in both the performance metrics). The total number of *stages* of our adaptive algorithm is "small"

[4] This work has been published in [23]. The journal version of this work can be found in [22].

($\mathcal{O}(\log(k))$). Similarly, in the scenario with non-adaptive tests with noisy outcomes, we present the first scheme that is simultaneously near-optimal in both the number of tests and the decoding complexity (via an algorithm that requires $\mathcal{O}(k\log(k)\log(n))$ tests and has a decoding complexity of $\mathcal{O}(k(\log(n)+\log^2(k)))$. Finally, we present an adaptive algorithm that only requires 2 stages, and for which both the number of tests and the decoding complexity scale as $\mathcal{O}(k(\log(n)+\log^2(k)))$. For all three settings the probability of error of our algorithms scales as $\mathcal{O}\left(1/(\text{poly}(k)\right)$. For each of the statements above about the order of the number of measurements, decoding complexity, and probability of error, we provide explicitly computed "small" universal factors in our theorem statements.[5]

1.4 Compressive Phase Retrieval – SUPER

Compressive phase retrieval algorithms attempt to reconstruct a "sparse high-dimensional vector" from its "low-dimensional intensity measurements". Suppose $\mathbf{x}$ is any length-n *input vector* over $\mathbb{C}$ with exactly k non-zero entries, and A is an $m \times n$ ($k < m \ll n$) *phase measurement matrix* over $\mathbb{C}$. The decoder is handed m "intensity measurements" $(|A_1\mathbf{x}|, \ldots, |A_m\mathbf{x}|)$ (corresponding to component-wise absolute values of the linear measurement $A\mathbf{x}$) – here A_i's correspond to the rows of the measurement matrix A. In this work, we present a class of measurement matrices A, and a corresponding decoding algorithm that we call SUPER, which can reconstruct $\mathbf{x}$ up to a global phase from intensity measurements. The SUPER algorithm is the first to simultaneously have the following properties: (a) it requires only $\mathcal{O}(k)$ (order-optimal) measurements, (b) the computational complex-

[5] A preliminary version of this work has been published in [26]. The journal version of this work has been submitted to the IEEE Transactions on Information Theory and can be found in [27].

ity of decoding is $\mathcal{O}(k \log k)$ (near order-optimal) arithmetic operations, (c) it succeeds with high probability over the design of A. Our results hold for all $k \in \{1, 2, \ldots, n\}$.[6]

□ **End of chapter.**

[6] A preliminary version of this work has been published in [24]. The journal version of this work has been submitted to the IEEE Transactions on Information Theory and can be found in [25]. In parallel and independently of this work, an algorithm with very similar design and performance was proposed and presented in [114].

Chapter 2

Technical Background

2.1 Representation of measurement process

2.1.1 Measurement matrix

For all the problems considered in this thesis, the measurement process can be represented by an $m \times n$ measurement matrix, A. Each row of A corresponds to a measurement, and each column of A corresponds to a entry of $\mathbf{x}$. To simplify our discussion, we only consider the noiseless case.

For compressive sensing, the measurement $\mathbf{y}$ equals $A\mathbf{x}$ which corresponds to the matrix multiplication of A and $\mathbf{x}$. Similarly for network tomography, we use matrix multiplication to get the measurement. For phase retrieval, the measurement model implies that the measurement output is the result of taking the element-wise magnitude of $A\mathbf{x}$.

In the group testing problem, the entries of the measurement matrix A are binary such that if the entry of i-th row and j-th column is 1, then the j-th item is included in the i-th test/measurement. At the risk of potential confusion, we follow notation convention in the group testing literature to *also* denote entries in the input vector $\mathbf{x}$ and the measurement vector $\mathbf{y}$ by 0's and 1's. The physical meanings of these 0's and 1's is as

follows: 0's and 1's, in $\mathbf{x}$, represent non-defective items and defective items respectively, and in $\mathbf{y}$, represent negative test outcomes and positive test outcomes respectively. As per the group testing measurement model, the test outcome is positive if and only if there exists at least one defective item in that test. In this model, $A(\mathbf{x})$ therefore represents the composition of m disjunctive measurements.

2.1.2 Bipartite graph

The structure of a measurement process can also be represented by a bipartite graph. Put n nodes on the left and m' nodes on the right. Each left node represents an element of $\mathbf{x}$ and each right node represents a set of measurement (in general $m' < m$).[1] If there is an edge between the j-th left node and the i-th right node, the j-th entry is involved in the i-th set of measurements.

Hence, there is a mapping between the structure of measurement matrix and the bipartite graph. The structure of A is the *biadjacency matrix* of the bipartite graph.

2.1.3 Picking/Peeling process

There are three types of right/measurement nodes.

Zeroton/Zero node: A right node is called a zeroton (or a zero node) if it connects to only left nodes which correspond to zero components of $\mathbf{x}$. Although such zerotons may provide useful information about the zero components of $\mathbf{x}$, for reasons pertaining to computation complexity, we will not use zerotons in our decoding algorithms.

[1] For reasons that will become clear later, in most of our algorithms we clump together some "small" constant number of measurements together into a single measurement node. Hence, in general m/m' will equal an integer.

Singleton/Leaf node: A right node is called a singleton (or a leaf node) if it connects to exactly one left node which corresponds to non-zero components of $\mathbf{x}$. Singletons are the "most important" nodes in our algorithms, since they allow us to "bootstrap" our decoding algorithms. For SHO-FA, GROTESQUE and FRANTIC, we only use leaf nodes for decoding. For SUPER, leaf nodes are used to reconstruct the magnitudes of non-zero entries.

Multiton/Non-leaf node: A right node is called a multiton (or a non-leaf node) if it connects to more than one left node which corresponds to non-zero components of $\mathbf{x}$. Multitons are particularly useful in some types of non-linear measurement models, for instance, in the phase recovery model, and our SUPER algorithm exploits multitions critically.

See Algorithm 1 for the meta-algorithm for sparse recovery algorithms. To guarantee the existence of "useful" right/measurement nodes, we study different types of bipartite graphs. For the detailed properties used, please refer to Chapters 3, 4, 5, and 6. However, there are several challenges. One lies in identifying which type of node (zeroton, singleton, or multition) a right node belongs to. The second lies in identifying which non-zero component of $\mathbf{x}$ is measured by the useful node. The third is to be able to do all this blindingly fast (in time linear in the sparsity k, rather than in the ambient signal dimension n). One "trick" we use is that we use "structured" measurements, with potentially several measurements per measurement node. The actual structure of these structured measurements, and the corresponding decoding algorithms depends critically on the measurement model under consideration, and is explicated in greater detail in Chapters 3, 4, 5, and 6.

Algorithm 1 Meta Algorithm for Sparse Recovery Problems

1: Input: A, $\mathbf{y}$

2: Output: $\hat{\mathbf{x}}$

3: **while** not all non-zeros of $\mathbf{x}$ are recovered **do**

4: 1. Picking: identify one set of "useful" measurement node, U

5: 2. Decode: $\hat{\mathbf{x}} = \mathcal{D}(\mathbf{y}_U, \hat{\mathbf{x}})$ {$\mathcal{D}$ denotes the decoding procedure}

6: 3. Peeling: update $\mathbf{y}$

7: **end while**

2.2 Error-correcting codes

We can use error-correcting codes to design the robust algorithms. The idea of using a measurement matrix whose columns are codewords of a linear code is not new, especially in the context of compressive sensing [28, 45]. However, we only use a linear code as a "sub-routine" to do the estimate if the right node is leaf node (or singleton).

In this thesis, we use expander codes to design robust and efficient group testing algorithms in the scenario of noisy tests. Although in this thesis this idea is only applied to group testing problem, as future work it can be further explored for the other three problems.

2.3 Big-Oh Notation

We use Big Oh notation as is standard in computer science. Specifically, $f(n)$ is said to be $\mathcal{O}(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$, $o(g(n))$, and $\omega(g(n))$ if $g(n)$ is *asymptotically an upper bound, a lower bound, a tight bound, a tight upper bound, and a tight lower bound* for $f(n)$ respectively. For details, please refer to Chapter 3 of [42]. Specifically, $\mathcal{O}(\text{poly}(n))$ is defined as $\mathcal{O}\left(n^c\right)$ for $c > 0$.

2.4 Chernoff Bound and McDiamid's Inequality

In this work, exp corresponds to the exponential function base e but logarithms are base 2.

Theorem 1. *(Chernoff Bound) Let $X_1, \ldots, X_n$ be independent but not necessarily identically distributed random variables. Assume that $0 \leq X_i \leq 1$ for each $i \in [n]$. Let $X = X_1 + \cdots + X_n$. $\mu = \mathbf{E}[X] = \mathbf{E}[X_1] + \cdots + \mathbf{E}[X_n]$. Then for any $0 \leq \epsilon \leq 1$,*

$$\Pr[X \leq (1-\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2}{2}\mu\right)$$

and

$$\Pr[X \geq (1+\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2}{2+\epsilon}\mu\right);$$

$$\Pr[X \leq \mu - \epsilon n] \leq \exp\left(-2n\epsilon^2\right)$$

and

$$\Pr[X \geq \mu + \epsilon n] \leq \exp\left(-2n\epsilon^2\right).$$

Theorem 2. *[98](McDiarmid's Inequality) Let $X_1, \ldots, X_n$ be independent but not necessarily identically distributed random variables all taking values in the set $\mathcal{X}$. Further, let $f : \mathcal{X}^n \to \mathbb{R}$ be a function of $X_1, \ldots, X_n$ that satisfies $\forall i, \forall x_1, \ldots, x_n, x_i' \in \mathcal{X}$,*

$$|f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, x_i', \ldots, x_n)| \leq c_i.$$

Then for all $\epsilon > 0$,

$$\Pr(f(X_1, \ldots, X_n) - \mathbf{E}[f(X_1, \ldots, X_n)] \geq \epsilon) \leq \exp\left(\frac{-2\epsilon^2}{\Sigma_{i=1}^m c_i^2}\right)$$

and

$$\Pr(f(X_1, \ldots, X_n) - \mathbf{E}[f(X_1, \ldots, X_n)] \leq -\epsilon) \quad \leq \quad \exp\left(\frac{-2\epsilon^2}{\Sigma_{i=1}^{m} c_i^2}\right).$$

□ **End of chapter.**

Chapter 3

Compressive Sensing – SHO-FA

3.1 Introduction

In recent years, spurred by the seminal work on *compressive sensing* of [31, 51], much attention has focused on the problem of reconstructing a length-n "compressible" vector $\mathbf{x}$ over $\mathbb{R}$ with fewer than n linear measurements. In particular, it is known (*e.g.* [11,34]) that with $m = \mathcal{O}(k \log(n/k))$ linear measurements one can computationally efficiently[1] obtain a vector $\hat{\mathbf{x}}$ such that the *reconstruction error* $\|\mathbf{x} - \hat{\mathbf{x}}\|_1$ is $\mathcal{O}(\|\mathbf{x} - \mathbf{x}_k{}^*\|_1)$,[2] where $\mathbf{x}_k{}^*$ is the best possible k-sparse approximation to $\mathbf{x}$ (specifically, the k non-zero terms of $\mathbf{x}_k{}^*$ correspond to the k largest components of $\mathbf{x}$ in magnitude, hence $\mathbf{x} - \mathbf{x}_k{}^*$ corresponds to the "tail" of $\mathbf{x}$).[3] A number of different classes of algorithms are able to give such performance guarantees, such as those based

[1] The caveat is that the reconstruction techniques require one to solve an LP. Though polynomial-time algorithms to solve LPs are known, they are generally considered to be impractical for large problem instances.

[2] In fact this is the so-called $\ell_1 < C\ell_1$ guarantee. One can also prove stronger $\ell_2 < C\ell_1/\sqrt{k}$ reconstruction guarantees for algorithms with similar computational performance, and it is known that a $\ell_2 < C\ell_2$ reconstruction guarantee is not possible if the algorithm is required to be zero-error [41], but is possible if some (small) probability of error is allowed [65, 119].

[3] All the notations for the model of compressive sensing are listed in Table 3.1.

on ℓ_1-optimization (*e.g.* [31, 51]), and those based on iterative "matching pursuit" (*e.g.* [53,137]). Similar results, with an additional additive term in the reconstruction error hold even if the linear measurements themselves also have noise added to them (*e.g.* [11, 34]). The fastest of these algorithms use ideas from the theory of expander graphs, and have running time $\mathcal{O}(n \log(n/k))$ [16, 17, 64].

The class of results summarized above are indeed very strong – they hold for *all* $\mathbf{x}$ vectors, including those with "worst-case tails", *i.e.* even vectors where the components of $\mathbf{x}$ smaller than the k largest coefficients (which can be thought of as "source tail") are chosen in a maximally worst-case manner. In fact [9] proves that to obtain a reconstruction error that scales linearly with the ℓ_1-norm of $\mathbf{z}$ (the tail of $\mathbf{x}$) requires $\Omega(k \log(n/k))$ linear measurements.

Number of measurements: However, depending on the application, such a lower bound based on "worst-case $\mathbf{z}$" may be unduly pessimistic. For instance, it is known that if $\mathbf{x}$ is exactly k-sparse (has exactly exactly k non-zero components, and hence $\mathbf{z} = \mathbf{0}$), then based on Reed-Solomon codes [121] one can efficiently reconstruct $\mathbf{x}$ with $\mathcal{O}(k)$ noiseless measurements (*e.g.* [111]) via algorithms with decoding time-complexity $\mathcal{O}(n \log(n))$, or via codes such as in [89, 101] with $\mathcal{O}(k)$ noiseless measurements with decoding time-complexity $\mathcal{O}(n)$.[4] In the regime where $k = \Theta(n)$ [80] uses the "sparse-matrix" techniques of [16, 17, 64] to demonstrate that $\mathcal{O}(k) = \mathcal{O}(n)$ measurements suffice to reconstruct $\mathbf{x}$.

Noise: Even if the source is not exactly k-sparse, a spate of recent work has taken a more information-theoretic view than the coding-theoretic/worst-case point-of-view espoused by much of the compressive sensing work thus

[4] In general the linear systems produced by Reed-Solomon codes are ill-conditioned over $\mathbb{R}$ and $\mathbb{C}$ [121], which causes problems for large n.

far. Specifically, suppose the length-n source vector is the sum of *any* exactly k-sparse vector $\mathbf{x}$ and a "random" source noise vector $\mathbf{z}$ (and possibly the linear measurement vector $A(\mathbf{x}+\mathbf{z})$ also has a "random" measurement noise vector $\mathbf{e}$ added to it). Then as long as the noise variances are not "too much larger" than the signal power, the work of [4] demonstrates that $\mathcal{O}(k)$ measurements suffice (though the proofs in [4] are information-theoretic and existential – the corresponding "typical-set decoding" algorithms require time exponential in n). Indeed, even the work of [9], whose primary focus was to prove that $\Omega(k\log(n/k))$ linear measurements are necessary to reconstruct $\mathbf{x}$ in the worst case, also notes as an aside that if $\mathbf{x}$ corresponds to an exactly k-sparse vector plus random noise, then in fact $\mathcal{O}(k)$ measurements suffice. The work in [143, 144] examines this phenomenon information-theoretically by drawing a nice connection with the *Rényi information dimension* $\bar{d}(X)$ of the signal/noise. The heuristic algorithms in [88] indicate that *approximate message passing* algorithms achieve this performance computationally efficiently (in time $\mathcal{O}(n\log(n))$), and [52] proves this rigorously. Corresponding lower bounds showing $\Omega(k\log(n/k))$ samples are required in the higher noise regime are provided in [63, 139].

Number of measurement bits: However, most of the works above focus on minimizing the number of linear measurements in $A\mathbf{x}$, rather than the more information-theoretic view of trying to minimize the number of bits in $A\mathbf{x}$ over all measurements. Some recent work attempts to fill this gap – notably "Counting Braids" [91, 92] (this work uses "multi-layered non-linear measurements"), and "one-bit compressive sensing" [79, 115] (the corresponding decoding complexity is somewhat high (though still polynomial-time in k and n) since it involves solving an LP).

Decoding time-complexity: The emphasis of the discussion thus far has been on the number of linear measurements/bits required to reconstruct

$\mathbf{x}$. The decoding algorithms in most of the works above have decoding time-complexities[5] that scale at least linearly with n. In regimes where k is significantly smaller than n, it is natural to wonder whether one can do better. Indeed, algorithms based on iterative techniques answer this in the affirmative. These include Chaining Pursuit [66], group-testing based algorithms [44], and Sudocodes [125] – each of these have decoding time-complexity that can be sub-linear in n (but at least $\Omega(k\log(k)\log(n))$), but each requires at least $\Omega(k\log(n))$ linear measurements.

Database query: Finally, we consider a *database query* property that is not often of primary concern in the compressive sensing literature. That is, suppose one is given a compressive sensing algorithm that is capable of reconstructing $\mathbf{x}$ with the desired reconstruction guarantee. Now suppose that one instead wishes to reconstruct, with reasonably high probability, just "a few" (constant number) *specific* components of $\mathbf{x}$, rather than all of it. Is it possible to do so even faster (say in constant time) – for instance, if the measurements are in a database, and one wishes to query it in a computationally efficient manner? If the matrix A is "dense" (most of its entries are non-zero) then one can directly see that this is impossible. However, several compressive sensing algorithms (for instance [80]) are based on "sparse" matrices A, and it can be shown that in fact these algorithms do indeed have this property "for free" (as indeed does our algorithm), even though the authors do not analyze this. As can be inferred from the name, this database query property is more often considered in the database community, for instance in the work on IBLTs [70].

[5]For ease of presentation, in accordance with common practice in the literature, in this discussion we assume that the time-complexity of performing a single arithmetic operation is constant. Explicitly taking the complexity of performing finite-precision arithmetic into account adds a multiplicative factor (corresponding to the precision with which arithmetic operations are performed) in the time-complexity of most of the works, including ours.

Notation	Definition
$\mathbf{x}$	Length-n signal over $\mathbb{R}$ with sparsity k
A	Dimension-$n \times m$ measurement matrix over $\mathbb{C}$
P	Bits of precision
$\mathbf{z}$	Source tail distributed as Gaussian with mean 0 and variance σ_z^2
$\mathbf{e}$	Measurement noise distributed as complex Gaussian with mean 0 and variance σ_e^2
$\mathbf{x}_k^*$	Best k-sparse approximation to $\mathbf{x}$
$\hat{\mathbf{x}}$	Estimate of $\mathbf{x}$
$\mathbf{y}$	Length-m measurement over $\mathbb{C}$

Table 3.1: Table of notation for the compressive sensing model

Locally decodable codes (LDCs) and locally recoverable codes (LRCs) also look at similar questions. Both LDCs and LRCs are special classes of error-correcting codes aiming to recover any individual symbol from accessing other r codeword symbols. These codes provide a solution to ensure reliability against storage node failures in distributed and cloud storage systems. The main difference between LDCs and LRCs is that LDCs aim to recover any message symbol, while LRCs aim to recover any codeword symbol. We refer the readers interested in this area to [134,148] for details.

3.1.1 Our contributions

Conceptually, the "iterative decoding" technique we use is not new. Similar ideas have been used in various settings in, for instance [70, 89, 118, 132]. However, to the best of our knowledge, no prior work has the same performance as our work – namely – information-theoretically order-optimal number of measurements, bits in those measurements, and time-complexity, for the problem of reconstructing a sparse signal (or sparse signal with a noisy tail and noisy measurements) via linear measurements (along with the

database query property).The key to this performance is our novel design of "sparse random" linear measurements, as described in Section 3.2.

Exactly k-sparse x and noiseless measurements

First, our results for the case that $\mathbf{x}$ is exactly k-sparse and measurements are noiseless are stated below. We use P as a "precision parameter", denoting the requirement that the relative error between the reconstructed $\hat{\mathbf{x}}$ and the original $\mathbf{x}$ be at most 2^{-P}, as defined in property i) of the Theorem below.

Theorem 1. *Let $k \leq n$. There exists a reconstruction algorithm SHO-FA for $A \in \mathbb{C}^{m \times n}$ with the following properties:*

i) For every k-sparse $\mathbf{x} \in \mathbb{R}^n$, with probability $1 - \mathcal{O}(1/k)$ over the choice of A, SHO-FA produces a reconstruction $\hat{\mathbf{x}}$ such that $\|\mathbf{x} - \hat{\mathbf{x}}\|_1 / \|\mathbf{x}\|_1 \leq 2^{-P}$,

ii) The number of measurements $m = 2ck$ for some $c > 0$,

iii) The number of steps required by SHO-FA is $\mathcal{O}(k)$, and

iv) The number of bitwise arithmetic operations required by SHO-FA is $\mathcal{O}(k(\log(n) + P))$.

Theorem 2. *Let $k \leq n$. There exists a reconstruction algorithm SHO-FA* for $A \in \mathbb{C}^{m \times n}$ with the following properties:*

i) For every k-sparse $\mathbf{x} \in \mathbb{R}^n$, with probability $1 - \mathcal{O}(1/k)$ over the choice of A, SHO-FA* produces a reconstruction $\hat{\mathbf{x}}$ such that $\|\mathbf{x} - \hat{\mathbf{x}}\|_1 / \|\mathbf{x}\|_1 \leq 2^{-P}$,

ii) The number of measurements $m = 2ck$, $\forall\ c > 1.22$,

iii) The number of steps required by SHO-FA is at most $4ck + 14k$, and

iv) The number of bitwise arithmetic operations required by SHO-FA is $\mathcal{O}(k(\log(n) + P))$.

SHO-FA* has better reconstruction performance than SHO-FA. The probabilistic construction of *measurement matrix* A is the same. The difference in SHO-FA is that the analysis is self-contained which requires that the *left degree* d of the underlying bipartite graph for the construction to be at least 13 (See Section 3.2.2 for details). The analysis of Theorem 2 mainly depends on the existence of 2-cores in d-uniform hypergraphs (for d being at least 3) after exploring its relationship with SHO-FA algorithm. Due to expositional complexity of the analysis, we omit the proofs of [70, Theorem 1] and Theorem 5 (see Section 3.2.7 for some remarks) and refer the reader to the original works for the proof.

Approximate reconstruction in the presence of noise

For approximate reconstruction, our results are stated below. We assume that both the source tail and measurement noise are distributed according to independent Gaussian distributions.

Theorem 3. *Let* $k = \mathcal{O}(n^{1-\Delta})$ *for some* $\Delta > 0$. *There exists a reconstruction algorithm SHO-FA-NO for* $A \in \mathbb{C}^{m \times n}$ *such that*

i) $m = ck$,

ii) SHO-FA consists of at most $4k$ *iterations, each involving a constant number of arithmetic operations with a precision of* $\mathcal{O}(\log(n))$ *bits, and*

iii) With probability $1 - o(1/k)$ *over the design of* A *and randomness in* $\mathbf{e}$ *and* $\mathbf{z}$,

$$\|\hat{\mathbf{x}} - \mathbf{x}\|_1 \leq C \left(\|\mathbf{z}\|_1 + \sqrt{\log(k)} \|\mathbf{e}\|_1 \right)$$

for some $C = C(\sigma_z, \sigma_e) > 0$.

Theorem 4. *Let $k = \mathcal{O}(n^{1-\Delta})$ for some $\Delta > 0$ and let $\epsilon > 0$. There exists a reconstruction algorithm SHO-FA-NO* for $A \in \mathbb{C}^{m \times n}$ such that*

i) $m = ck$,

ii) SHO-FA-NO* consists of at most $5k$ iterations, each involving a constant number of arithmetic operations with a precision of $\mathcal{O}(\log(n))$ bits, and

iii) With probability $1 - o(1/k)$ over the design of A and randomness in $\mathbf{e}$ and $\mathbf{z}$, and for each $l \in \{1, 2\}$, $\|\hat{\mathbf{x}} - \mathbf{x}\|_l \leq (1 + \epsilon)\left(\|\mathbf{z}\|_l + \|\mathbf{e}\|_l\right)$.

Theorem 4 has better reconstruction performance than Theorem 3. This is achieved by first running the algorithm from Theorem 3 to determine the support of $\mathbf{x}$ and then employing the Set-Query Algorithm of [118] to reduce the reconstruction error. (See Section 3.3.4 for details.)

To summarize, the desirable properties of SHO-FA are that with high probability:[6]

- **Number of measurements:** For every k-sparse $\mathbf{x}$, with high probability over A, $\mathcal{O}(k)$ linear measurements suffice to reconstruct $\mathbf{x}$. This is information-theoretically order-optimal.
- **Number of measurement bits:** The total number of bits in $A\mathbf{x}$ required to reconstruct $\mathbf{x}$ to a relative error of 2^{-P} is $\mathcal{O}(k(\log(n)+P))$. This is information-theoretically order-optimal for any $k = \mathcal{O}(n^{1-\Delta})$ (for any $\Delta > 0$).

[6] For most of the properties, we show that this probability can be made as large as $1 - 1/k^c$ for any $c > 0$, at the cost of increasing both the number of measurements and the decoding complexities by multiplication factors $f_1(c)$ and $f_2(c)$, respectively. For ease of exposition, we explicitly prove only $1 - \mathcal{O}(1/k)$.

- **Decoding time-complexity:** The total number of arithmetic operations required is $\mathcal{O}(k)$. This is information-theoretically order-optimal.

- **"Database-type queries":** With constant probability $1-\epsilon$ any single "database-type query" can be answered in constant time. That is, the value of a single component of $\mathbf{x}$ can be reconstructed in constant time with constant probability.[7]

- **Encoding/update complexity:** The computational complexity of generating $A\mathbf{x}$ from A and $\mathbf{x}$ is $\mathcal{O}(n)$, and if $\mathbf{x}$ changes to some $\mathbf{x}'$ in $\mathcal{O}(1)$ locations, the computational complexity of updating $A\mathbf{x}$ to $A\mathbf{x}'$ is $\mathcal{O}(1)$. Both of these are information-theoretically order-optimal.

- **Noise:** Suppose $\mathbf{z}$ and $\mathbf{e}$ have i.i.d. components[8] drawn respectively from $\mathcal{N}(0,\sigma_z^2)$ and $\mathcal{N}(0,\sigma_e^2)$. For every $\epsilon' > 0$ and for $k = \mathcal{O}(n^{1-\Delta})$ for any $\Delta > 0$, we present in Section 3.3 a modified version of SHO-FA called SHO-FA-NO that with high probability reconstructs $\mathbf{x}$ with an estimation error of $(1+\epsilon')(\|\mathbf{z}\|_2 + \|\mathbf{e}\|_2)$.[9]

[7] The constant ϵ can be made arbitrarily close to zero, at the cost of multiplicative factors $\mathcal{O}(1/\epsilon)$ in the number of measurements and decoding complexities required. In fact, if we allow the number of measurements and decoding complexity to scale as $\mathcal{O}(k\log(k))$, we can support any number of database queries, each in constant time, with probability of every one being answered correctly at least $1-\epsilon$.

[8] Even if the statistical distribution of the components of $\mathbf{z}$ and $\mathbf{e}$ are not i.i.d. Gaussian, statements with a similar flavor can be made. For instance, pertaining to the effect of the distribution of $\mathbf{z}$, it turns out that our analysis is sensitive only on the distribution of the *sum* of components of $\mathbf{z}$, rather then the components themselves. Hence, for example, if the components of $\mathbf{z}$ are i.i.d. non-Gaussian, it turns out that via the Berry-Esseen theorem [129] one can derive similar results to the ones derived in this work. In another direction, if the components of $\mathbf{z}$ are not i.i.d. but do satisfy some "regularity constraints", then using Bernstein's inequality [19] one can again derive analogous results. However, these arguments are more sensitive and outside the scope of this work, where the focus is on simpler models.

[9] As noted in Footnote 2, this $\ell_2 < \ell_2$ reconstruction guarantee implies the weaker $\ell_1 < \ell_1$

- **Practicality:** As validated by simulations (shown in Appendix 3.4), most of the explicitly calculable constant factors mentioned above are not large.

- **Different bases:** As is common in the compressive sensing literature, our techniques generalize directly to the setting wherein $\mathbf{x}$ is sparse in an alternative basis represented by the $n \times n$ matrix B (say, for example, in a wavelet basis). In this case, our measurement matrix is then AB^{-1}.

- **Universality:** While we present a specific ensemble of matrices over which SHO-FA operates, we argue that in fact similar algorithms work over fairly general ensembles of "sparse random matrices" (see Section 4.4), and further that such matrices can occur in applications, for instance in wireless MIMO systems [71] (Figure 3.10 gives such an example) and Network Tomography [146].

---

reconstruction guarantee $\|\mathbf{x} - \hat{\mathbf{x}}\|_1 < (1 + \epsilon')(\|\mathbf{z}\|_1 + \|\mathbf{e}\|_1)$

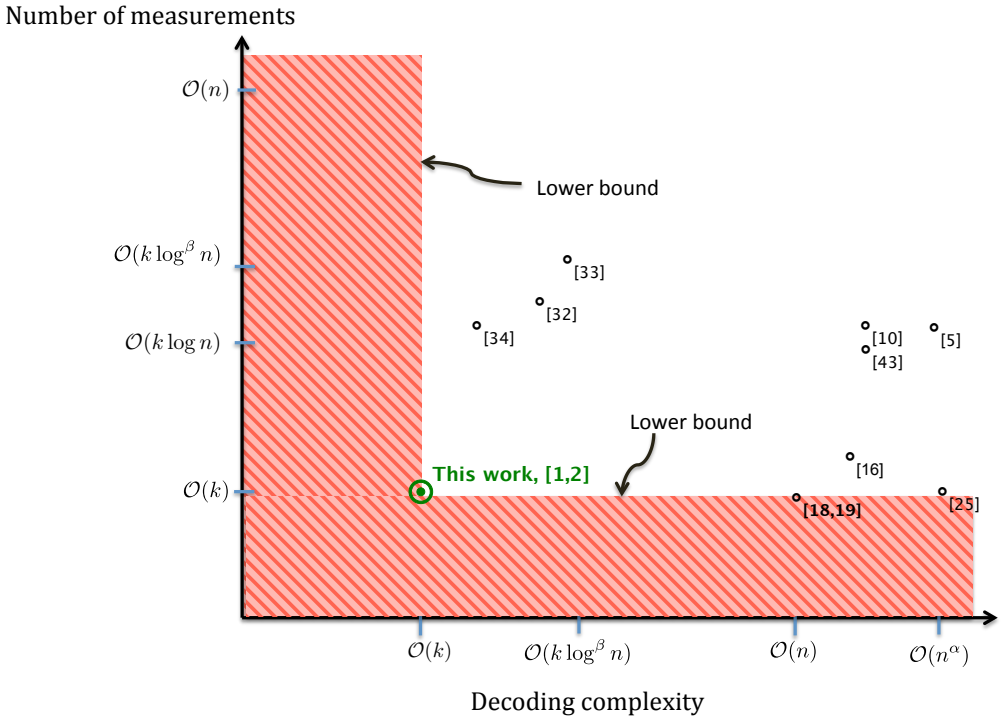


Figure 3.1: A comparison of prior work with this work (and that of [112]) in two parameters – decoding complexity, and number of measurements. Information-theoretic bounds show that the shaded region is infeasible. Only some of the many prior results in the field of Compressive Sensing are plotted here (some other works are referenced in the following table). Since k and n do not scale together in a fixed manner, the position of points in this graph should be interpreted in a relative rather than absolute manner. The constants α and β are intended to represent the degrees of low-degree polynomials.

Reference	A	$\mathbf{x}$	$\mathbf{z}$	$\mathbf{e}$	Reconstruction Goal	$\mathbf{P_e}$	# Measurements	# Decoding steps	Precision
Reed-Solomon [121]	D	D	0	0	Exact	0	$2k+1$	$\mathcal{O}(n\log(n))$ [6]	–
Singleton [130]	D/R	D	0	0	Exact	0	$\geq 2k$	–	–
Mitzenmacher-Varghese [101]	R	D	0	0	Exact	$\mathcal{O}(1)$	$\mathcal{O}(k)$	$\mathcal{O}(n)$	–
Kudekar-Pfister [89]	R	D	0	0	Exact	$o(1)$	$\mathcal{O}(k)$	$\mathcal{O}(n)$	–
Tropp-Gilbert [137]	G	D	0	0	Exact	$o(1)$	$\mathcal{O}(k\log(n))$	$\mathcal{O}(k^2 n\log(n))$	–
Wu-Verdú '10 [143]	R	R	R	0	Exact	$o(1)$	$n\bar{d}(\mathbf{x}+\mathbf{z})+o(n)$	$\mathcal{O}(\exp(n))$	–
Donoho *et al.* [52]	R	R	R	0 R	Exact $\ell_2 < C\ell_2$	o(1)	$n\bar{d}(\mathbf{x}+\mathbf{z})+o(n)$	$\mathcal{O}(n^3)$	–
Cormode-Muthukrishnan [44]	R	D	0	0	$\ell_2 < C\ell_2$	$o(1)$	$\mathcal{O}(k\ \mathrm{poly}(\log(n)))$	$\mathcal{O}(k\ \mathrm{poly}(\log(n)))$	–
Cohen *et al.* [41]	D	D	D	0	$\ell_2 < C\ell_2$	0	$\Omega(n)$	–	–
Price-Woodruff [119]	D	D	D	0	$\ell_2 < C\ell_2$	$o(1)$	$\theta(k\log(n/k))$	–	–
Ba *et al.* [9]	D/R	D	D	0	$\ell_1 < C\ell_1$	$\mathcal{O}(1)$	$\Omega(k\log(n/k))$	–	$\mathcal{O}(\log(n))$
Ba *et al.* [9]	R	D	R	0	$\ell_2 < C\ell_2$	$o(1)$	$\mathcal{O}(k)$	$\mathcal{O}(\exp(n))$	
Candès [34], Baraniuk *et al.* [11]	R	D	D	D	$\ell_2 < \frac{C}{\sqrt{k}}\ell_1$	$o(1)$	$\mathcal{O}(k\log(n/k))$	LP	–
Indyk *et al.* [78]	D	D	D	D	$\ell_1 < (1+\epsilon)\ell_1$	0	$\mathcal{O}(k\log(n/k))$	$\mathcal{O}(n\log(n/k))$	–
Akçakaya *et al.* [3]	R	D	0	R	$\ell_2 < C\ell_2$/ Sup. Rec.	0	$\mathcal{O}(k)$ Cond. on x_{min}	$\mathcal{O}(\exp(n))$	–
Wu-Verdú '11 [144]	R	R	R	R	$\ell_2 < C\ell_2$	$\mathcal{O}(1)$	$\bar{d}(\mathbf{x}+\mathbf{z})$	$\mathcal{O}(\exp(n))$	–
Wainwright [139]	$\mathcal{N}$	D	0	R	Sup. Rec.	$\mathcal{O}(1)$	$\Omega(k\log(n/k))$	–	–
Fletcher *et al.* [63]	$\mathcal{N}$	D	0	R	Sup. Rec.	$o(1)$	$\mathcal{O}(k\log(n-k))$	–	–
Aeron *et al.* [1]	$\mathcal{N}$	D	0	R	Sup. Rec.	$\mathcal{O}(1)$	$\Omega(k\log(n/k))$		–
Plan-Vershynin [115]	R	D	0	sgn	$\ell_2 < f(\mathbf{x},\mathbf{x}_k)$	$\mathcal{O}(1)$	$k^2\log(n/k)$	LP	1
Jacques *et al.* [79]	R	D	0	sgn	$\ell_2 < f'(n,k)$	$\mathcal{O}(1)$	$k\log(n)$	$\exp(n)$	1
Sarvotham *et al.* [125]	R	D	0	0	Exact	$\mathcal{O}(1)$	$k\log(n)$	$k\log(k)\log(n)$	–
Gilbert *et al.* [66]	R	P.L.	P.L.	0	$\ell_1 < [1+C\log(n)]\ell_1$	$\mathcal{O}(1)$	$k\log^2(n)$	$k\log^2(n)\log^2(k)$	–
This work/Pawar *et al.* [112]	R	D	0	0	Exact	$o(1)$	$\mathcal{O}(k)$	$\mathcal{O}(k)$	$\mathcal{O}(\log(n)+P)$
	R	D	R	R	$\ell_1 < C\ell_1$	$o(1)$	$\mathcal{O}(k)$	$\mathcal{O}(k)$	$\mathcal{O}(\log(n))$

Explanations and discussion of the reference table: At the risk

of missing much of the literature, and also perhaps oversimplifying nuanced results, we summarize in this table many of the strands of work preceding this paper and related to it – not all results from each work are represented in this table. The second to the fifth columns respectively reference whether the measurement matrix A, source k-sparse vector $\mathbf{x}$, source noise $\mathbf{z}$, and measurement noise $\mathbf{e}$ are random (R) or deterministic (D) – a 0 in a column corresponding to noise indicates that that work did not consider that type of noise. An entry "P.L." stands for "Power Law" decay in columns corresponding to $\mathbf{x}$ and $\mathbf{z}$. For achievability schemes, in general D-type results are stronger than R-type results, which in turn are stronger than 0-type results. This is because a D-type result for the measurement matrix indicates that there is an explicit construction of a matrix that satisfies the required goals, whereas the R-type results generally indicate that the result is true with high probability over measurement matrices. Analogously, a D in the columns corresponding to $\mathbf{x}$, $\mathbf{z}$ or $\mathbf{e}$ indicates that the scheme is true for all vectors, whereas an R indicates that it is true for random vectors from some suitable ensemble. For converse results, the opposite is true, *i.e.*, 0 results are stronger than R-type results, which are stronger than D-type results. An entry $\mathcal{N}$ indicates the normal distribution – the results of [139] and [63] are converses for matrices with i.i.d. Gaussian entries. An entry "sgn" denotes (in the case of works dealing with one-bit measurements) that the errors are sign errors. The sixth column corresponds to what the desired goal is. The strongest possible goal is to have exact reconstruction of $\mathbf{x}$ (up to quantization error due to finite-precision arithmetic), but this is not always possible, especially in the presence of noise. Other possible goals include "Sup. Rec. " (short for support recovery) of $\mathbf{x}$, or that the reconstruction $\hat{\mathbf{x}}$ of $\mathbf{x}$ differs from $\mathbf{x}$ as a "small" function of $\mathbf{z}$. It is known that if a deterministic reconstruction algorithm is desired to work for all $\mathbf{x}$

and $\mathbf{z}$, then $\|\hat{\mathbf{x}}-\mathbf{x}\|_2 < \mathcal{O}(\|\mathbf{z}\|_2)$ is not possible with less than $\Omega(n)$ measurements [41], and that $\|\hat{\mathbf{x}}-\mathbf{x}\|_2 < \mathcal{O}(\|\mathbf{z}\|_1/\sqrt{k})$ implies $\|\hat{\mathbf{x}}-\mathbf{x}\|_1 < \mathcal{O}(\|\mathbf{z}\|_1)$. The reconstruction guarantees in [79, 115] unfortunately do not fall neatly in these categories. The seventh column indicates what the probability of error is – *i.e.* the probability over any randomness in A, $\mathbf{x}$, $\mathbf{z}$ and $\mathbf{e}$ that the reconstruction goal in the sixth column is not met. In the eighth column, some entries are marked $\bar{d}(\mathbf{x}+\mathbf{z})$ – this denotes the (upper) Rényi dimension of $\mathbf{x}+\mathbf{z}$ – in the case of exactly k-sparse vectors this equals k, but for non-zero $\mathbf{z}$ it depends on the distribution of $\mathbf{z}$. The ninth column considers the computational complexity of the algorithms – the entry "LP" denotes the computational complexity of solving a linear program.[10] The final column notes whether the particular work referenced considers the precision of arithmetic operations, and if so, to what level. See Figure 3.1 for a comparison of prior works with this work in number of measurements and decoding complexity.

3.1.2 Special acknowledgements

While writing the paper [10], we became aware of parallel and independent work by Pawar and Ramchandran [112] that relies on ideas similar to our work and achieves similar performance guarantees. Both the work of [112] and the preliminary version of this work [10] were presented at the same venue.

In particular, the bounds on the minimum number of measurements required for "worst-case" recovery and the corresponding discussion on recovery of signals with "random tails" in [9] led us to consider this problem

[10] Of course, if the LP has structure, then general bounds on time complexity might be too pessimistic. We refer the readers interested in this topic to [15] where the measurement matrix A is binary and sparse.

in the first place. Equally, the class of compressive sensing codes in [80], which in turn build upon the constructions of expander codes in [132], have been influential in leading us to this work. While the model in [118] differs from the one in this work, the techniques therein are of significant interest in our work. The analysis in [118] of the number of disjoint components in certain classes of random graphs, and also the analysis of how noise propagates in iterative decoding is potentially useful sharpening our results. We elaborate on these in Section 3.3.

The work that is conceptually the closest to SHO-FA is that of the Invertible Bloom Lookup Tables (IBLTs) introduced by Goodrich and Mitzenmacher [70] (though our results were derived independently, and hence much of our analysis follows a different line of reasoning). The data structures and iterative decoding procedure (called "peeling" in [70]) used are structurally very similar to the ones used in this work. However the "measurements" in IBLTs are fundamentally non-linear in nature – specifically, each measurement includes within it a "counter" variable – it is not obvious how to implement this in a linear manner. Therefore, though the underlying graphical structure of our algorithms is similar, the details of our implementation require new non-trivial ideas. Also, IBLTs as described are not robust to either signal tails or measurement noise. Nonetheless, the ideas in [70] have been influential in this work. In particular, the notion that an individual component of $\mathbf{x}$ could be recovered in constant time, a common feature of Bloom filters, came to our notice due to this work.

The rest of this chapter is organized as follows. We first present SHO-FA algorithm for the case that $\mathbf{x}$ is exactly k-sparse and measurements are noiseless in Section 3.2. In Section 3.3, we introduce SHO-FA-NO algorithm for the case that $\mathbf{x}$ is approximately sparse and measurements are noisy. In Section 3.4, we validate the performance of our algorithms via numerical

Notation	Definition
$\mathcal{G}$	A bipartite graph with n nodes on the left and m' nodes on the right
d	Left regularity of $\mathcal{G}$
$\mathcal{S}(\mathbf{x})$	Set of left nodes of $\mathcal{G}$ of size less than k
$\mathcal{S}'(\mathbf{x})$	Subset of $\mathcal{S}(\mathbf{x})$
$\mathcal{S}(\mathbf{x}) - leaf$ node	A right node of $\mathcal{G}$ with exactly one neighbor in $\mathcal{S}(\mathbf{x})$
$\mathcal{S}(\mathbf{x}) - non - leaf$ node	A right node of $\mathcal{G}$ with more than one neighbor in $\mathcal{S}(\mathbf{x})$
$\mathcal{S}(\mathbf{x}) - zero$ node	A right node of $\mathcal{G}$ with no neighbor in $\mathcal{S}(\mathbf{x})$
$a_{i,j}^{(I)}$	The j-th entry of the $(2i-1)$-th rows of A for $j \in [n]$ and $i \in m'$
$a_{i,j}^{(V)}$	The j-th entry of the $(2i)$-th rows of A for $j \in [n]$ and $i \in m'$
$\mathbf{y}^{(I)}$	Length-m' identification measurement over $\mathbb{C}$
$\mathbf{y}^{(V)}$	Length-m' verification measurement over $\mathbb{C}$
$\mathcal{L}(t)$	Leaf node list in the t-th iteration
$\hat{\mathbf{x}}(t)$	Estimate of $\mathbf{x}$ in the t-th iteration
$\tilde{\mathbf{y}}^{(I)}(t)$	Residual identification measurement in the t-th iteration
$\tilde{\mathbf{y}}^{(V)}(t)$	Residual verification measurement in the t-th iteration

Table 3.2: Table of notation for SHO-FA algorithm

results. Section 3.6 concludes this work.

3.2 Exactly k-sparse $\mathbf{x}$ and noiseless measurements

We first consider the simpler case when the source signal is exactly k-sparse and the measurements are noiseless, *i.e.*, $\mathbf{y} = A\mathbf{x}$, and both $\mathbf{z}$ and $\mathbf{e}$ are all-zero vectors. The intuition presented here carries over to the scenario wherein both $\mathbf{z}$ and $\mathbf{e}$ are non-zero, considered separately in Section 3.3.

For k-sparse input vectors $\mathbf{x} \in \mathbb{R}^n$ let the set $\mathcal{S}(\mathbf{x})$ denote its *support*, *i.e.*, its set of nonzero values $\{j : x_j \neq 0\}$. Recall that in our setup, for some m, a *measurement matrix* $A \in \mathbb{C}^{m \times n}$ is chosen probabilistically. This matrix operates on $\mathbf{x}$ to yield the *measurement vector* $\mathbf{y} \in \mathbb{C}^m$ as $\mathbf{y} = A\mathbf{x}$. The decoder takes the vector $\mathbf{y}$ as input and outputs the reconstruction $\hat{\mathbf{x}} \in \mathbb{R}^n$ – it is desired that $\hat{\mathbf{x}}$ equal $\mathbf{x}$ (with up to P bits of precision) with high probability over the choice of measurement matrices .

In this section, we describe a probabilistic construction of the measurement matrix A and a reconstruction algorithm SHO-FA that achieves the guarantees stated in Theorem 3.

We present a "simple" and self-contained proof of the Theorem 3 in Sections 3.2.1 to 3.2.6. In Section 3.2.7 I direct the reader to an alternative, more technically challenging, analysis (based on the work of [70]) that leads to a tighter characterization of the constant factors in the parameters of Theorem 3.

3.2.1 High-level intuition

If $m = \Theta(n)$, the task of reconstructing $\mathbf{x}$ from $\mathbf{y} = A\mathbf{x}$ appears similar to that of *syndrome decoding* of a channel code of rate n/m [124]. It is well-known [74] that channel codes based on *bipartite expander graphs*, *i.e.*, bipartite graphs with good expansion guarantees for all sets of size less than or equal to k, allow for decoding in a number of steps that is linear in the size of $\mathbf{x}$. In particular, given such a bipartite expander graph with n nodes on the left and m nodes on the right, choosing the parity-check matrix A as an $m \times n$ binary matrix with non-zero values in the locations where the corresponding pair of nodes in the graph has an edge is known to result in codes with constant rate and minimum distance that is linear in n.

Motivated by this [80] explores a measurement design that is derived from expander graphs and shows that $\mathcal{O}(k \log(n/k))$ measurements suffice, and $\mathcal{O}(k)$ iterations with overall decoding complexity of $\mathcal{O}(n \log(n/k))$.[11]

It is tempting to think that perhaps an optimized application of ex-

[11] The work of [17] is related – it also relies on bipartite expander graphs, and has similar performance for exactly k-sparse vectors. But [17] can also handle a significantly larger class of approximately k-sparse vectors than [80]. However, our algorithms are closer in spirit to those of [80], and hence we focus on this work.

pander graphs could result in a design that require only $\mathcal{O}(k)$ number of measurements. However, we show that in the compressive sensing setting, where, typically $k = o(n)$, it is not possible to satisfy the desired expansion properties. In particular, if one tries to mimic the approach of [80], one would need bipartite expanders such that *all* sets of size k on one side of the graph "expand" – we show in Lemma 2 that this is not possible. As such, this result may be of independent interest for other work that require similar graphical constructions (for instance the "magical graph" constructions of [40], or the expander code constructions of [132] in the high-rate regime).

Instead, one of our key ideas is that we do not really need "true" expansion. Instead, we rely on a notion of approximate expansion that guarantees expansion for most k-sized sets (and their subsets) of nodes on the left of our bipartite graph. We do so by showing that any set of size at most k, with high probability over suitably chosen measurement matrices, expands to the desired amount. Probabilistic constructions turn out to exist for our desired property.[12] Such a construction is shown in Lemma 1.

Our second key idea is that in order to be able to recover all the k non-zero components of $\mathbf{x}$ with at most $\mathcal{O}(k)$ steps in the decoding algorithm, it is necessary (and sufficient) that on average, the decoder reconstructs one previously undecoded non-zero component of $\mathbf{x}$, say x_j, in $\mathcal{O}(1)$ steps in the decoding algorithm. For $k = o(n)$ the algorithm does not even have enough time to write out all of $\mathbf{x}$, but only its non-zero values. To achieve such efficient identification of x_j, we go beyond the 0/1 matrices used in almost all prior work on compressive sensing based on expander graphs.[13] Instead,

[12] In fact similar properties have been considered before in the literature – for instance [40] constructed so-called "magical graphs" with similar properties. Our contribution is the way we use this property for our needs.

[13] It can be argued that such a choice is a historical artifact, since error-correcting codes based

we use distinct values in each row for the non-zero values in A, so that if only one non-zero x_j is involved in the linear measurement involving a particular y_i (a situation that we demonstrate happens in a constant fraction of y_i), one can identify which x_j it must be in $\mathcal{O}(1)$ time. Our decoding then proceeds iteratively, by identifying such x_j and canceling their effects on y_i, and terminates after $\mathcal{O}(k)$ steps after all non-zero x_j and their locations have been identified (since we require our algorithm to work with high probability for all $\mathbf{x}$, we also add "verification" measurements – this only increases the total number of measurements by a constant factor).[14] Our calculations are precise to $\mathcal{O}(\log(n) + P)$ bits – the first term in this comes from requirements necessary for computationally efficient identification of non-zero x_j, and the last term from the requirement that we require that the reconstructed vector be correct up to P bits of precision. Hence the total number of bits over all measurements is $\mathcal{O}(k((\log(n) + P))$. Note that this is information-theoretically order-optimal, since even specifying k locations in a length-n vector requires $\Omega(k(\log(n/k))$ bits, and specifying the value of the non-zero locations so that the relative reconstruction error is $\mathcal{O}(2^{-P})$ requires $\Omega(kP)$ bits.

We now present our SHO-FA algorithm in two stages. We first use our first key idea (of "approximate" expansion) in Section 3.2.2 to describe some properties of bipartite expander graphs with certain parameters. We then show in Section 3.2.3 how these properties, via our second key idea

on expanders were originally designed to work over the binary field $\mathbb{F}_2$ (Of course, there also exists similar code over the non-binary field [50].) There is no reason to stick to this convention when, as now, computations are done over $\mathbb{R}$ or $\mathbb{C}$.

[14]The verification-based decoding algorithm in [151] built on LDPC codes has the similar flavor to ours. The decoding complexity of a check-node operation is linearly with the degree of check node which scales with n. Therefore, the overall decoding complexity also scales linearly with n. However, in our SHO-FA algorithm, we implement identification and verification steps both in constant time.

(of efficient identification) can be used by SHO-FA to obtain desirable performance.

3.2.2 "Approximate Expander" Graph $\mathcal{G}$

We first construct a bipartite graph $\mathcal{G}$ (see Example 1 in the following) with some desirable properties outlined below. We then show in Lemmas 1 and 3 that such graphs exist (Lemma 2 shows the non-existence of graphs with even stronger properties). In Section 3.2.3 we then use these graph properties in the SHO-FA algorithm. To simplify notation in what follows (unless otherwise specified) we omit rounding numbers resulting from taking ratios or logarithms, with the understanding that the corresponding inaccuracy introduced is negligible compared to the result of the computation. Also, for ease of exposition, we fix various internal parameters to "reasonable" values rather than optimizing them to obtain "slightly" better performance at the cost of obfuscating the explanations – whenever this happens we shall point it out parenthetically. Lastly, let ϵ be any "small" positive number, corresponding to the probability of a certain "bad event".

Properties of $\mathcal{G}$:

1. *Construction of a left-regular bipartite graph:* The graph $\mathcal{G}$ is chosen uniformly at random from the set of bipartite graphs with n nodes on the left and m' nodes on the right, such that each node on the left has degree $d \geq 13$.[15] In particular, m' is chosen to equal ck for some design parameter c to be specified later as part of code design.

2. *Edge weights for "identifiability":* For each node on the right, the weights of the edges attached to it are required to be distinct. In particular, each edge weight is chosen as a complex number of unit

[15] For ease of analysis we now consider the case when $d \geq 13$ – our tighter result in Theorem 2 relaxes this, and works for any $d \geq 3$.

magnitude, and phase between 0 and $\pi/2$. Since there are a total of dn edges in $\mathcal{G}$, choosing distinct phases for each edge attached to a node on the right requires at most $\log(dn)$ bits of precision (though on average there are about dn/m' edges attached to a node on the right, and hence on average one needs about $\log(dn/m')$ bits of precision).

3. $\mathcal{S}(\mathbf{x})$-*expansion:* With high probability over $\mathcal{G}$ defined in Property 1 above, for any set $\mathcal{S}(\mathbf{x})$ of at most k nodes on the left, the number of nodes neighbouring those in any $\mathcal{S}'(\mathbf{x}) \subseteq \mathcal{S}(\mathbf{x})$ is required to be at least $2d/3$ times the size of $\mathcal{S}'(\mathbf{x})$.[16] The proof of this statement is the subject of Lemma 1.

4. *"Many"* $\mathcal{S}(\mathbf{x})$-*leaf nodes:* For any set $\mathcal{S}(\mathbf{x})$ of at most k nodes on the left of $\mathcal{G}$, we call any node on the right of $\mathcal{G}$ an $\mathcal{S}(\mathbf{x})$-*leaf node* if it has exactly one neighbor in $\mathcal{S}(\mathbf{x})$, and we call it a $\mathcal{S}(\mathbf{x})$-*non-leaf node* if it has two or more neighbours in $\mathcal{S}(\mathbf{x})$. (If the node on the right has no neighbours in $\mathcal{S}(\mathbf{x})$, we call it a $\mathcal{S}(\mathbf{x})$-*zero node.*) Assuming $\mathcal{S}(\mathbf{x})$ satisfies the expansion condition in Property 3 above, it can be shown that at least a fraction $1/2$ of the nodes that are neighbours of any $\mathcal{S}'(\mathbf{x}) \subseteq \mathcal{S}(\mathbf{x})$ are $\mathcal{S}'(\mathbf{x})$-leaf nodes.[17] The proof of this statement is the subject of Lemma 3.

Example 1*:* We now demonstrate via the following toy example in Figures 3.2 and 3.3 a graph $\mathcal{G}$ (where d is chosen to be 3) satisfying Proper-

[16] The *expansion factor* $2d/3$ is somewhat arbitrary. In our proofs, this can be replaced with any number strictly between half the degree and the degree of the left nodes, and indeed one can carefully optimize over such choices so as to improve the constant in front of the expected time-complexity/number of measurements of SHO-FA. Again, we omit this optimization since this can only improve the performance of SHO-FA by a constant factor.

[17] Yet again, this choice of $1/2$ is a function of the choices made for the degree of the left nodes in Property 1 and the expansion factor 2 in Property 3. Again, we resist the temptation to optimize these constants.

ties 1-4.

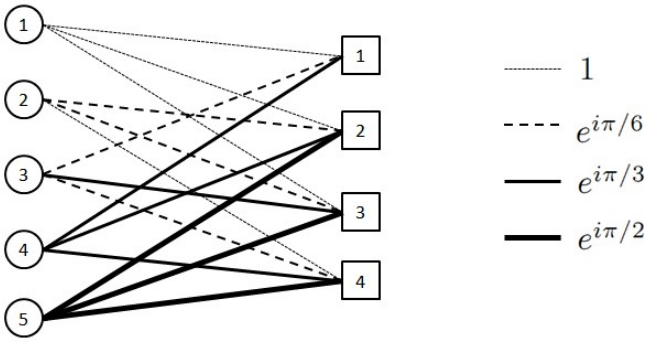


Figure 3.2: *Property 1*: Bipartite "approximate expander" graph with $n = 5$ nodes on the left, and $m' = 4$ nodes on the right. Each node on the left has degree 3. *Property 2*: The thickness of the edges represents the weights assigned to the edges. In particular, it is required that for each node on the right, the edges incoming have distinct weights. In this example, the thinnest edges are assigned a weight of 1, the next thickest edges have a weight $e^{\iota\pi/6}$, the next thickest edges have weight $e^{\iota 2\pi/6} = e^{\iota\pi/3}$, and the thickest edges have weight $e^{\iota 3\pi/6} = e^{\iota\pi/2}$.

□

We now state the Lemmas needed to make our arguments precise. First, we formalize the $\mathcal{S}'(\mathbf{x})$-expansion property defined in Property 3.

Lemma 1. ***(Property 3 ($\mathcal{S}(\mathbf{x})$-expansion)):*** *Let $k < n \in \mathbb{N}$ be arbitrary, and let $c \in \mathbb{N}$ be fixed. Let $\mathcal{G}$ be chosen uniformly at random from the set of all bipartite graphs with n left nodes (each of degree d) and $m' = ck$ right nodes. Then for any fixed subset $\mathcal{S}(\mathbf{x})$ of the left nodes having size at most k, with probability $1 - o(1/k)$ (over the random choice $\mathcal{G}$), every $\mathcal{S}'(\mathbf{x}) \subseteq \mathcal{S}(\mathbf{x})$ has at least $(2d/3)|\mathcal{S}'(\mathbf{x})|$ neighbors.*

Proof. Follows from a standard probabilistic method argument. Given for completeness in Appendix A.1.1. ■

Note here that, in contrast to the "usual" definition of "vertex expansion" [74] (wherein the expansion property is desired *"for all"* subsets of

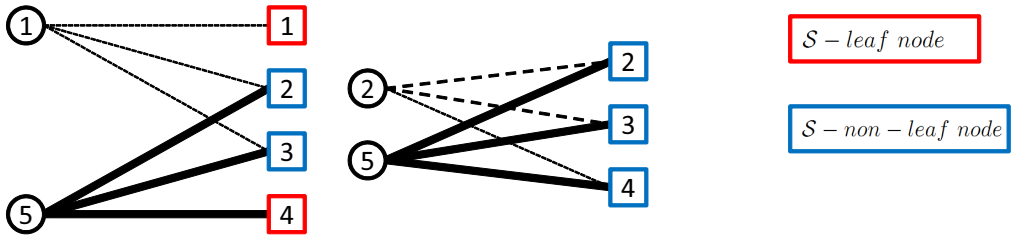


Figure 3.3: *Property 3*: We require that *most* sets $\mathcal{S}'(\mathbf{x})$ of at most $|\mathcal{S}(\mathbf{x})| = k = 2$ nodes on the left in the graph $\mathcal{G}$ in Figure 3.2 have at least $2|\mathcal{S}'(\mathbf{x})|$ neighbors on the right.) In the graph in Figure 3.2 it can be manually verified that most sets of size $\mathcal{S}'(\mathbf{x})$ at most 2 have at least $2|\mathcal{S}'(\mathbf{x})|$ neighbors. For example, Figure 3.2(a) focuses on the subset $\mathcal{S}'(\mathbf{x}) = \{1, 5\}$ of nodes on the left side of $\mathcal{G}$ in Figure 3.2. This particular $\mathcal{S}'(\mathbf{x})$ has 4 neighbours, and all its single-node subsets have 3 neighbours. The only $\mathcal{S}'(\mathbf{x})$ set of two or fewer nodes that does not satisfy Property 3 is $\{2, 5\}$, as shown in Figure 3.2(b), since it has only $3 < 2 \times 2$ neighbours. *Property 4*: For sets $\mathcal{S}'(\mathbf{x})$ that satisfy Property 3 it can be manually verified that "many" of their neighbours are $\mathcal{S}'(\mathbf{x})$-leaf nodes. For example, for $\mathcal{S}'(\mathbf{x}) = \{1, 5\}$, two out of its four neighbours (*i.e.*, a fraction 1/2) are $\mathcal{S}'(\mathbf{x})$-leaf nodes – which satisfies the constraint that at least a fraction 1/2 of its neighbours be $\mathcal{S}'(\mathbf{x})$-leaf nodes. On the other hand, for $\mathcal{S}'(\mathbf{x}) = \{2, 5\}$ (which does *not* satisfy Property 3), *none* of its neighbours are $\mathcal{S}'(\mathbf{x})$-leaf nodes.

left nodes up to a certain size) Lemma 1 above only gives a probabilistic expansion guarantee for any subset of $\mathcal{S}(\mathbf{x})$ of size k. In fact, Lemma 2 below shows that for the parameters of interest, "for all"-type expanders cannot exist.

Lemma 2. *Let $k = o(n)$, and $d > 0$ be an arbitrary constant. Let $\mathcal{G}$ be an arbitrary bipartite graph with n nodes (each of degree d) on the left and m' nodes on the right. Then for all sufficiently large n, suppose each set of size k of $\mathcal{S}(\mathbf{x})$ nodes on the left of $\mathcal{G}$ has strictly more than $d/2$ times as many nodes neighbouring those in $\mathcal{S}(\mathbf{x})$, as there are in $\mathcal{S}(\mathbf{x})$. Then*

$m' = \Omega(k \log(n/k))$.

Proof. Follows from the Hamming bound in coding theory [124] and standard arguments about expander codes [132]. Proof in Appendix A.1.2. ■

Another way of thinking about Lemma 2 is that it indicates that if one wants a "for all" guarantee on expansion, then one has to return to the regime of $m' = \mathcal{O}(k \log(n/k))$ measurements, as in "usual" compressive sensing.

Next, we formalize the "many $\mathcal{S}(\mathbf{x})$-leaf nodes" property defined in Property 4. Recall that for any set $\mathcal{S}(\mathbf{x})$ of at most k nodes on the left of $\mathcal{G}$, we call any node on the right of $\mathcal{G}$ an $\mathcal{S}(\mathbf{x})$-*leaf node* if it has exactly one neighbor in $\mathcal{S}(\mathbf{x})$.

Lemma 3. *Let $\mathcal{S}(\mathbf{x})$ be a set of k nodes on the left of $\mathcal{G}$ such that the number of nodes neighbouring those in any $\mathcal{S}'(\mathbf{x}) \subseteq \mathcal{S}(\mathbf{x})$ is at least $2d/3$ times the size of $\mathcal{S}'(\mathbf{x})$. Then at least a fraction $1/2$ of the nodes that are neighbours of any $\mathcal{S}'(\mathbf{x}) \subseteq \mathcal{S}(\mathbf{x})$ are $\mathcal{S}'(\mathbf{x})$-leaf nodes.*

Proof. Based on Lemma 1. Follows from a counting argument similar to those used in expander codes [132]. Proof in Appendix A.1.3. ■

Given a graph $\mathcal{G}$ satisfying properties 1-4, we now describe our encoding and decoding procedure.

3.2.3 Measurement design

Matrix structure and entries: The encoder's *measurement matrix* A is chosen based on the structure of $\mathcal{G}$ (recall that $\mathcal{G}$ has n nodes on the left and m' nodes on the right). To begin with, the matrix A has $m = 2m'$ rows, and its non-zero values are unit-norm complex numbers.

Remark 1. This choice of using complex numbers rather than real numbers in A is for notational convenience only. One can equally well choose a matrix A' with $m = 4m'$ rows, and replace each row of A with two consecutive rows in A' comprising respectively of the real and imaginary parts of rows of A. Since the components of $\mathbf{x}$ are real numbers, hence there is a bijection between $A\mathbf{x}$ and $A'\mathbf{x}$ – indeed, consecutive pairs of elements in $A'\mathbf{x}$ are respectively the real and imaginary parts of the complex components of $A\mathbf{x}$. Also, as we shall see (in Section 3.2.7), the choice of unit-norm complex numbers ensures that "noise" due to finite precision arithmetic does not get "amplified". In Section 3.2.7, we argue that this property enables us to apply SHO-FA to other settings such as wireless systems that naturally generate an ensemble of matrices that resemble SHO-FA.

Based on the idea in Remark 1, we get the following corollary.

Corollary 1. *Let $k \leq n$. There exists a reconstruction algorithm SHO-FA for $A \in \mathbb{R}^{m \times n}$ with the following properties:*

i) For every k-sparse $\mathbf{x} \in \mathbb{R}^n$, with probability $1 - \mathcal{O}(1/k)$ over the choice of A, SHO-FA produces a reconstruction $\hat{\mathbf{x}}$ such that $\|\mathbf{x} - \hat{\mathbf{x}}\|_1 / \|\mathbf{x}\|_1 \leq 2^{-P}$

ii) The number of measurements $m = 4ck$ for some $c > 0$

iii) The number of steps required by SHO-FA is $\mathcal{O}(k)$

iv) The number of bitwise arithmetic operations required by SHO-FA is $\mathcal{O}(k(\log(n) + P))$.

Note that Corollary 1 is different from Theorem 3 in that the entries of measurement matrix A are over real numbers instead of complex numbers. Therefore, the number of measurements required doubles. The remaining

of the analysis in this section still focuses on the measurement matrix design over complex numbers. In particular, corresponding to node i on the right-hand side of $\mathcal{G}$, the matrix A has two rows. The j^{th} entries of the $(2i-1)^{th}$ and $2i^{th}$ rows of A are respectively denoted $a_{i,j}^{(I)}$ and $a_{i,j}^{(V)}$. (The superscripts (I) and (V) respectively stand for *Identification* and *Verification*, for reasons that shall become clearer when we discuss the process to reconstruct $\mathbf{x}$.)

Identification entries: If $\mathcal{G}$ has no edge connecting node j on the left with i on the right, then the *identification entry* $a_{i,j}^{(I)}$ is set to equal 0. Else, if there is such an edge, $a_{i,j}^{(I)}$ is set to equal

$$a_{i,j}^{(I)} = e^{\iota j\pi/(2n)}. \tag{3.1}$$

(Here ι denotes the imaginary unit.) This entry $a_{i,j}^{(I)}$ can also be thought of as the weight of the edge in $\mathcal{G}$ connecting j on the left with i on the right. In particular, the *phase* $j\pi/(2n)$ of $a_{i,j}^{(I)} = e^{\iota j\pi/(2n)}$ will be critical for our algorithm. As in Property 2 in Section 3.2.2, our choice above guarantees distinct weights for all edges connected to a node i on the right.

Verification entries: Whenever the identification entry $a_{i,j}^{(I)}$ equals 0, we choose to set the corresponding *verification entry* $a_{i,j}^{(V)}$ also to be zero. On the other hand, whenever $a_{i,j}^{(I)} \neq 0$, then we set $a_{i,j}^{(V)}$ to equal $e^{\iota\theta_{i,j}^{(V)}}$ for $\theta_{i,j}^{(V)}$ chosen uniformly at random from $[0, \pi/2]$ (with $\mathcal{O}(\log(k))$ bits of precision).[18]

Example 2*:* The matrix A corresponding to the graph $\mathcal{G}$ in Example 1 is shown in Figure 3.4.

[18] This choice of precision for the verification entries contributes one term to our expression for the precision of arithmetic required. As we argue later in Section 3.2.7, this choice of precision guarantees that if a single identification step returns a value for x_j, this is indeed correct with probability $1 - o(1/k)$. Taking a union bound over $\mathcal{O}(k)$ indices corresponding to non-zero x_j

$$A = \left[\begin{array}{ccccc}
e^{\iota\pi\cdot 0} & 0 & e^{\iota\pi/6} & e^{\iota\pi/3} & 0 \\
e^{\iota\theta_{1,1}} & 0 & e^{\iota\theta_{1,3}} & e^{\iota\theta_{1,4}} & 0 \\
\hdashline
e^{\iota\pi\cdot 0} & e^{\iota\pi/6} & 0 & e^{\iota\pi/3} & e^{\iota\pi/2} \\
e^{\iota\theta_{2,1}} & e^{\iota\theta_{2,2}} & 0 & e^{\iota\theta_{2,4}} & e^{\iota\theta_{2,5}} \\
\hdashline
e^{\iota\pi\cdot 0} & e^{\iota\pi/6} & e^{\iota\pi/3} & 0 & e^{\iota\pi/2} \\
e^{\iota\theta_{3,1}} & e^{\iota\theta_{3,2}} & e^{\iota\theta_{3,3}} & 0 & e^{\iota\theta_{3,5}} \\
\hdashline
0 & e^{\iota\pi\cdot 0} & e^{\iota\pi/6} & e^{\iota\pi/3} & e^{\iota\pi/2} \\
0 & e^{\iota\theta_{4,2}} & e^{\iota\theta_{4,3}} & e^{\iota\theta_{4,4}} & e^{\iota\theta_{4,5}}
\end{array}\right]$$

Figure 3.4: This 8×5 matrix denotes the A corresponding to the graph $\mathcal{G}$. Note that its primary purpose is expository – clearly, 8 measurements (or indeed, 16 measurements over $\mathbb{R}$) to reconstruct a 2-sparse vector of length 5 is too many! Nonetheless, this is just an artifact of the fact that n in this example is small. In fact, according to our proofs, even as n scales to infinity, the number of measurements required to reconstruct a 2-sparse vector (or in general a k-sparse vector for constant k) remains constant! Also, note that we do not use the assignment for the identification entries $a_{i,j}^{(I)}$ specified in (3.2), since doing so would result in ugly and not very illuminating calculations in Example 3 below. However, as noted in Remark 2, this is not critical – it is sufficient that distinct entries in the identification rows of the matrix be distinct.

3.2.4 Reconstruction

Overview

We now provide some high-level intuition on the decoding process.

Since the measurement matrix A has interspersed identification and verification rows, this induces corresponding interspersed *identification observations* $y_i^{(I)}$ and *verifications observations* $y_i^{(V)}$ in the *observation vector* $\mathbf{y} = A\mathbf{x}$. Let $\mathbf{y}^{(I)} = \{y_i^{(I)}\}$ denote the length-m' *identification vector* over $\mathbb{C}$, and $\mathbf{y}^{(V)} = \{y_i^{(V)}\}$ denote the length-m' *verification vector* over $\mathbb{C}$.

gives us an overall $1 - o(1)$ probability of success.

Given the measurement matrix A and the observed $(\mathbf{y}^{(I)}, \mathbf{y}^{(V)})$ identification and verification vectors, the decoder's task is to find *any "consistent"* k-sparse vector $\hat{\mathbf{x}}$ such that $A\hat{\mathbf{x}}$ results in the corresponding identification and verification vectors. We shall argue below that if we succeed, then with high probability over A (specifically, over the verification entries of A), this $\hat{\mathbf{x}}$ must equal $\mathbf{x}$.

To find such a consistent $\hat{\mathbf{x}}$, we design an iterative decoding scheme. This scheme starts by setting the initial guess for the reconstruction vector $\hat{\mathbf{x}}$ to the all-zero vector. It then initializes, in the manner described in the next paragraph, a $\mathcal{S}(\mathbf{x})$-leaf-node list denoted $\mathcal{L}(1)$, corresponding to a set of indices of $\mathcal{S}(\mathbf{x})$-leaf nodes.

The decoder checks to see whether i is a $\mathcal{S}(\mathbf{x})$-leaf node in the following way. First, it looks at the entry $y_i^{(I)}$ and "estimates" which node j on the left of the graph $\mathcal{G}$ "could have generated the identification observation $y_i^{(I)}$". It then uses the verification entry $a_{i,j}^{(V)}$ and the verification observation $y_i^{(V)}$ to verify its estimate. After sequentially examining each entry $y_i^{(I)}$, the list of *all* $\mathcal{S}(\mathbf{x})$-leaf nodes is denoted $\mathcal{L}(1)$.

In the t^{th} iteration of the decoding process, the decoder picks a leaf node in $i \in \mathcal{L}(t)$. Using this, it then reconstructs the non-zero component x_j of $\mathbf{x}$ that "generated" $y_i^{(I)}$. If this reconstructed value x_j is successfully "verified" using the verification entry $a_{i,j}^{(V)}$ and the verification observation $y_i^{(V)}$),[19] then the algorithm performs the following steps in this iteration:

- It updates the observation vectors by subtracting the "contribution" of the coordinate x_j to the measurements it influences (there are exactly 13 of them since the degree of the nodes on the left side of $\mathcal{G}$ is 13),
- It updates the $\mathcal{S}(\mathbf{x})$-leaf-node list $\mathcal{L}(t)$ by removing i from $\mathcal{L}(t)$ and checking the change of status (zero, leaf, or non-leaf) of other indices

[19] As Ronald W. Reagan liked to remind us, "*doveryai, no proveryai*".

influenced by x_j (there at most 12),

- Finally the algorithm picks a new index i from the updated list $\mathcal{L}(t+1)$ for the next iteration.

The decoder performs the above operations repeatedly until $\hat{\mathbf{x}}$ has been completely recovered. We also show that (with high probability over A) in at most k steps this process does indeed terminate.

Example 3*:* Figures 3.5–3.9 show a sample decoding process for the matrix A as in Example 2, and the observed vector $\mathbf{y}$ shown in the figures. The example also demonstrates each of several possible scenarios the algorithm can find itself in, and how it deals with them.

Formal description of SHO-FA's reconstruction process

Our algorithm proceeds iteratively, and has at most k overall number of iterations, with t being the variable indexing the iteration number.

1. *Initialization*: We initialize the algorithm by setting the *signal estimate vector* $\hat{\mathbf{x}}(1)$ to the all-zeros vector 0^n, and the *residual measurement identification/verification vectors* $\tilde{\mathbf{y}}^{(I)}(1)$ and $\tilde{\mathbf{y}}^{(V)}(1)$ to the decoder's observations $\mathbf{y}^{(I)}$ and $\mathbf{y}^{(V)}$.

2. *Leaf-Node List*: Let $\mathcal{L}(1)$, the initial $\mathcal{S}(\mathbf{x})$-leaf node set, be the set of indices i which are $\mathcal{S}(\mathbf{x})$-leaf nodes. We generate the list via the following steps:

   (a) *Compute angles $\theta^{(I)}(i)$ and $\theta^{(V)}(i)$:* Let the *identification and verification angles* be defined respectively as the phases of the identification and verification entries being considered for index i (starting from 1), as follows:

$$\theta^{(I)}(i) \triangleq \angle\left(\tilde{y}_i^{(I)}(1)\right),$$

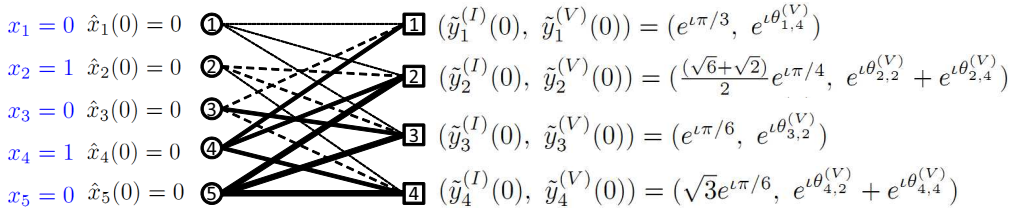


Figure 3.5: **Initialization**: The (true) $\mathbf{x}$ equals $(0, 1, 0, 1, 0)$ (and hence $\mathcal{S}(\mathbf{x}) = \{2, 4\}$). Also note that nodes 1 and 3 on the right of $\mathcal{G}$ are $\mathcal{S}(\mathbf{x})$-leaf nodes, as defined in Property 4. However, all of this is unknown to the decoder *a priori*. The decoder sets the (starting) estimate $\hat{\mathbf{x}}(0)$ of the reconstruction vector $\hat{\mathbf{x}}$ to the all-zeros vector. The (starting) *gap vector* $\tilde{\mathbf{y}}$ is set to equal $\mathbf{y}$, which in turn equals the corresponding 4 pairs of identification and verification observations on the right-hand side of $\mathcal{G}$. The specific values of $\theta_{i,j}^{(V)}$ in the verification observations do not matter currently – all that matters is that given $\mathbf{x}$, each of the four verification observations are non-zero (with high probability over the choices of $\theta_{i,j}^{(V)}$). Hence the (starting) value of the *neighbourly* set equals $\{1, 2, 3, 4\}$. This step takes $\mathcal{O}(k)$ number of steps, just to initialize the neighbourly set. By the end of the decoding algorithm (if it runs successfully), the tables will be turned – all the entries on the right of $\mathcal{G}$ will equal zero, and (at most) k entries on the left of $\mathcal{G}$ will be non-zero.

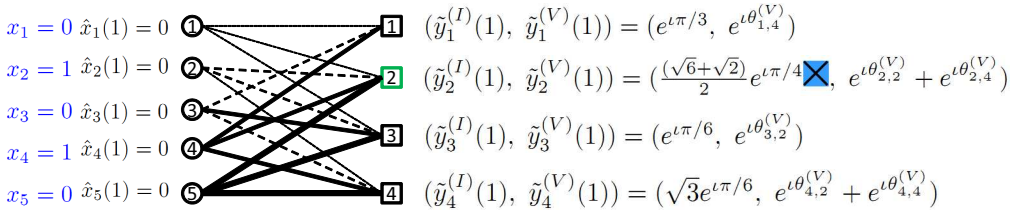


Figure 3.6: **Leaf-Node List 1 (Failed identification)**: The decoder picks the index $i = 2$ from the neighbourly set $\{1, 2, 3, 4\}$, and checks the phase of the corresponding gap vector identification observation $\tilde{y}_2^{(I)}$. Since this equals $\pi/4$, which is *not* in the set of possible phases in the 2^{nd} identification row of A (which are all multiples of $\pi/6$), the decoder declares $i = 2$ is not a leaf node. This entire process takes a constant number of steps.

$$\theta^{(V)}(i) \triangleq \angle\left(\tilde{y}_i^{(V)}(1)\right).$$

Here $\angle(.)$ computes the phase of a complex number up to $\mathcal{O}(\max\{\log(n/k), \log(k)\})$ bits of precision.[20]

(b) *Check if the current identification and verification angles correspond to a valid and unique x_j:* For this, we check at most two things (both calculations are done up to the precision specified in the previous step).

i. First, we check if $j \triangleq \theta^{(I)}(i)(2n/\pi)$ is an integer, and the corresponding j^{th} element of the i^{th} row is non-zero. If so, we have "tentatively identified" that the i^{th} component of $\tilde{\mathbf{y}}$ is a leaf-node of the currently unidentified non-zero components of $\mathbf{x}$, and in particular is connected to the j^{th} node on the left, and the algorithm proceeds to the next step below. If not, we simply increment i by 1 and return to Step (2a).

ii. Next, we verify our estimate from the previous step. If

[20] Roughly, the former term guarantees that the identification angle is calculated precisely enough, and the latter that the verification angle is calculated precisely enough.

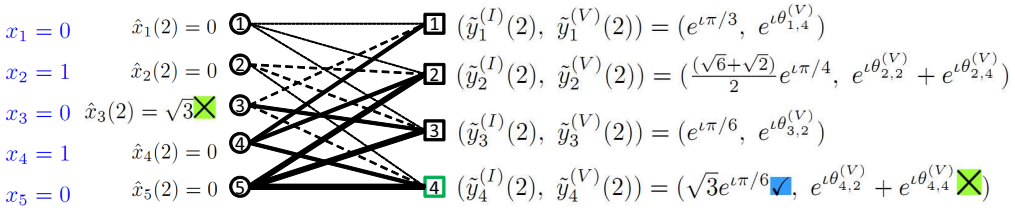


Figure 3.7: **Leaf-Node List 2 (Passed identification, failed verification)**: In this step, a potentially more serious failure could happen. In particular, suppose the decoder picks the index $i = 4$ from the neighbourly set $\{1, 2, 3, 4\}$ (note that 4 is *also* not a $\mathcal{S}(\mathbf{x})$-leaf node), and checks the phase of the corresponding gap vector identification observation $\tilde{y}_4^{(I)}$, it just so happens that the value of $\mathbf{x}$ is such that this corresponds to a phase of $\pi/6$. But as can be seen from the matrix in Figure 3.4, for $i = 4$ this corresponds to $a_{i,j}^{(I)}$ for $j = 3$. Hence the decoder would make a "false identification" of $j = 3$, and estimate that $\hat{x}_3$ equals the magnitude of $\tilde{y}_4^{(I)}$, which would equal $\sqrt{3}$. This is where the verification entries and verification observations save the day. Recall that the phase of each verification entry is chosen uniformly at random (with sufficient bit precision) from $[0, \pi/2)$, independently of both $\mathbf{x}$ and the other entries of A. Hence the probability that $\sqrt{3}$ (the miscalculated value of $\hat{x}_3$) times the corresponding verification entry $a_{4,3}^{(V)}$ equals $\tilde{y}_4^{(I)}$ is "small". Hence the decoder in this case too declares $i = 4$ is not a leaf node. This entire process takes a constant number of steps.

$a_{i,j}^{(V)}\tilde{y}_i^{(I)}/a_{i,j}^{(I)} = \tilde{y}_i^{(V)}$, the verification test passes, and include i in $\mathcal{L}(1)$. If not, we simply increment i by 1 and return to Step (2a).

3. *Operations in t^{th} iteration:* The t^{th} decoding iteration accepts as its input the t^{th} signal estimate vector $\hat{\mathbf{x}}(t)$, the t^{th} leaf node set $\mathcal{L}(t)$, and the t^{th} residual measurement identification/verification vectors $(\tilde{\mathbf{y}}^{(I)}(t), \tilde{\mathbf{y}}^{(V)}(t))$. In $\mathcal{O}(1)$ steps it outputs the $(t+1)^{th}$ signal estimate vector $\hat{\mathbf{x}}(t+1)$, the $(t+1)^{th}$ leaf node set $\mathcal{L}(t+1)$, and

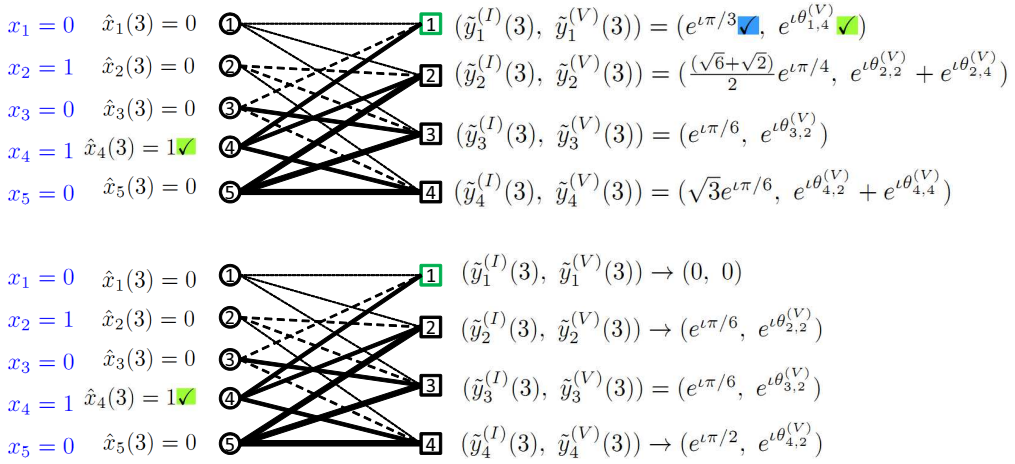


Figure 3.8: **Leaf-Node List 3 (Passed identification, passed verification) and the first iteration**: Now, suppose the decoder randomly picks the index $i = 1$ from the neighbourly set $\{1, 2, 3, 4\}$ (note that 1 *is* a $\mathcal{S}(\mathbf{x})$-leaf node). In this case, the phase of the corresponding gap vector identification observation $\tilde{y}_1^{(I)}$ equals $\pi/3$. As can be seen from the matrix in Figure 3.4, for $i = 1$ this corresponds to $a_{i,j}^{(I)}$ for $j = 4$. Hence the decoder makes a "correct identification" of $j = 4$, and estimates (also correctly) that $\hat{x}_4$ equals the magnitude of $\tilde{y}_1^{(I)}$, which equals 1. On checking with the verification entry, the decoder observes also that 1 (the detected value of $\hat{x}_4$) times the corresponding verification entry $a_{1,4}^{(V)}$ equals $\tilde{y}_1^{(V)}$. Hence it declares that $i = 1$ is a leaf node. Similarly, we know that 3 is a leaf node too. Therefore, the leaf node set equals $\{1, 3\}$. The entire process of making a list of leaf nodes takes $\mathcal{O}(k)$ number of steps. Suppose in the first iteration, the decoder picks $i = 1$. Hence it updates the value of $\hat{x}_4$ to 1, the neighbourly set to $\{2, 3, 4\}$, the leaf node set to $\{2, 3, 4\}$ and $\tilde{\mathbf{y}}$ to the values shown (only the three indices 1, 3 and 4 on the right need to be changed). At this point, note that $\mathcal{S}'(\mathbf{x})$ also changes from $\{2, 4\}$ to the singleton set $\{4\}$. This entire iteration takes a constant number of steps.

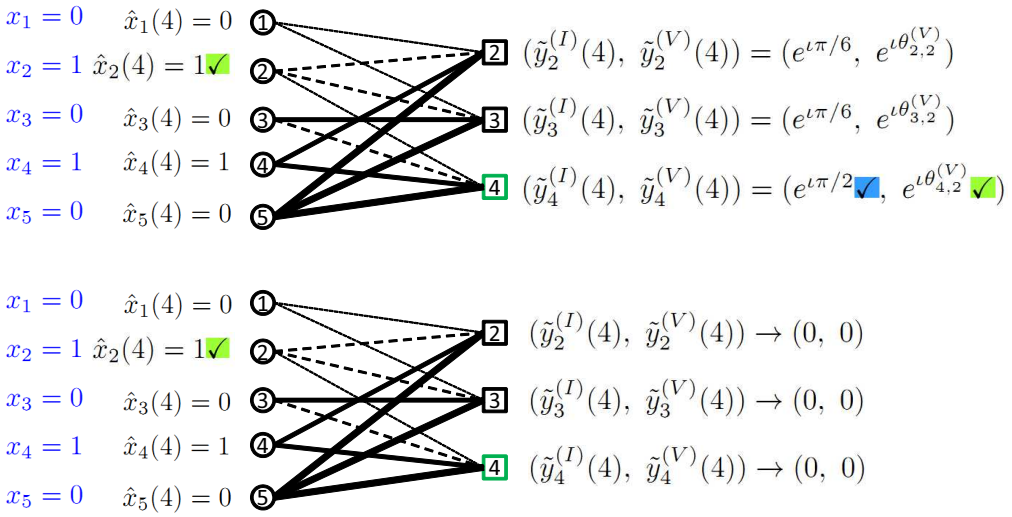


Figure 3.9: **Second iteration and Termination**: In the second iteration, the decoder randomly picks $i = 4$ from the leaf node set $\{2, 3, 4\}$. Recall that in the first iteration this choice of i did not aid in decoding. However, now that node 4 on the right of $\mathcal{G}$ has been "cleaned up", it is now a leaf node for $\mathcal{S}'(\mathbf{x})$. This demonstrates the importance of not "throwing away" information which seems useless at some point in time. Hence, analogously to the process in Figure 3.8, the decoder estimates the value of $\hat{x}_2$ to 1, updates the leaf node set to the empty set, and $\tilde{\mathbf{y}}$ to the all-zero vector (all in a constant number of steps). Since the gap vector is zero, this indicates to the decoder that it should output $\hat{\mathbf{x}}$ as its estimate of $\mathbf{x}$, and terminate.

the $(t+1)^{th}$ residual measurement identification/verification vectors $(\tilde{\mathbf{y}}^{(I)}(t+1), \tilde{\mathbf{y}}^{(V)}(t+1))$ after the performing the following steps sequentially (each of which takes at most a constant number of atomic steps):

(a) *Pick a random $i(t) \in \mathcal{L}(t)$:* The decoder picks an element $i(t)$ uniformly at random from the t^{th} leaf-node list $\mathcal{L}(t)$.

(b) *Compute angles $\theta^{(I)}(t)$ and $\theta^{(V)}(t)$:* Let the *current identification and verification angles* be defined respectively as the phases of the residual identification and verification entries being considered in that step, as follows:

$$\begin{aligned} \theta^{(I)}(t) &\triangleq \angle\left(\tilde{y}^{(I)}_{i(t)}(t)\right), \\ \theta^{(V)}(t) &\triangleq \angle\left(\tilde{y}^{(V)}_{i(t)}(t)\right). \end{aligned}$$

(c) *Locate non-zero entry j and derive the value of $\hat{x}_{j(t)}(t)$:* For this, we do at most two things (both calculations are done up to the precision specified in the previous step).

i. First, we calculate $j(t) \triangleq \theta^{(I)}(t)(2n/\pi)$. We have identified that the i^{th} component of $\tilde{\mathbf{y}}$ is a leaf-node of the currently unidentified non-zero components of $\mathbf{x}$, and in particular is connected to the $j(t)^{th}$ node on the left, and the algorithm proceeds to the next step below.

ii. Next, we assign the value, $\tilde{y}^{(I)}_{i(t)}(t)/a^{(I)}_{i(t),j(t)} = \tilde{y}^{(V)}_{i(t)}(t)/a^{(V)}_{i(t),j(t)}$, to $\hat{x}_{j(t)}(t)$ and proceeds the algorithm to the next step below.

(d) *Update $\hat{\mathbf{x}}(t+1)$, $\mathcal{L}(t+1)$,$\tilde{\mathbf{y}}^{(I)}(t+1)$, and $\tilde{\mathbf{y}}^{(V)}(t+1)$:* In particular, at most 13 components of each of these vectors need to be updated. Specifically, $\hat{x}_{j(t)}(t+1)$ equals $\tilde{y}^{(I)}_{i(t)}(t)/a^{(I)}_{i(t),j(t)}$. $i(t)$ is removed from the leaf node set $\mathcal{L}(t)$ and check whether the

(at most six) neighbours of $\hat{x}_{j(t)}(t)$ become leaf nodes to get the leaf-node list $\mathcal{L}(t+1)$. And finally (at most) seven values each of $\tilde{\mathbf{y}}^{(I)}(t+1)$ and $\tilde{\mathbf{y}}^{(V)}(t+1)$ are updated from those of $\tilde{\mathbf{y}}^{(I)}(t)$ and $\tilde{\mathbf{y}}^{(V)}(t)$ (those corresponding to the neighbours of $\hat{x}_{j(t)}(t)$) by subtracting out $\hat{x}_{j(t)}(t)$ multiplied by the appropriate coefficients of A.

4. Termination: The algorithm stops when the leaf node set is empty, and outputs the last $\hat{\mathbf{x}}(t)$.

3.2.5 Decoding complexity

We start by generating $\mathcal{L}(1)$, the initial list of leaf nodes. For each node i, we calculate the identification and verification angles (which takes 2 operations), and then check if the identification and verification angles correspond to a valid and unique x_j (which takes 2 operations). Therefore generating the initial list of leaf nodes takes $\mathcal{O}(k)$ (to be precise $4ck$) operations .

In iteration t, we decode a new non-zero entry x_j of $\mathbf{x}$ by picking a leaf node from $\mathcal{L}(t)$, identifying the corresponding index j and value x_j (via 2 arithmetic operations corresponding to the identification and verification steps, respectively), and updating $\mathcal{L}(t+1)$ (since x_j is connected to d nodes on the right, out of which one has already been decoded, this takes at most $2(d-1)$ operations – $(d-1)$ for identification and $(d-1)$ for verification), $\tilde{\mathbf{y}}^{(I)}(t+1)$, and $\tilde{\mathbf{y}}^{(V)}(t+1)$ (similarly, this takes at most $4(d-1)$ operations – $2(d-1)$ additions and $2(d-1)$ multiplications).

Next we note that each iteration results in recovering a new non-zero coordinate of $\mathbf{x}$ (assuming no decoding errors, which is true with high probability as demonstrated in the next section). Hence the total number of iterations is at most k.

Hence the overall number of operations over all iterations is $\mathcal{O}(k)$ (to be precise, at most $4ck + (6d - 4)k$).

3.2.6 Correctness

Next, we show that $\hat{\mathbf{x}} = \mathbf{x}$ with high probability over A. To show this, it suffices to show that each non-zero update to the estimate $\hat{\mathbf{x}}(t)$ sets a previously zero coordinate to the correct value with sufficiently high probability.

Note that if $i(t)$ is a leaf node for $\mathcal{S}(t)$, and if all non-zero coordinates of $\hat{\mathbf{x}}(t)$ are equal to the corresponding coordinates in $\mathbf{x}$, then the decoder correctly identifies the parent node $j(t) \in \mathcal{S}(t)$ for the leaf node $i(t)$ as the unique coordinate that passes the phase identification and verification checks.

Thus, the t^{th} iteration ends with an erroneous update only if

$$\angle \left(\sum_{p \in N(\{i(t)\})} x_p e^{\iota \theta^{(I)}_{i(t),p}} \right) = \theta^{(I)}_{i(t),j(t)}$$

for some j such that there are more than one non-zero terms in the summation on the left. Next, consider

$$\angle \left(\sum_{p \in N(\{i(t)\})} x_p e^{\iota \theta^{(V)}_{i(t),p}} \right) = \theta^{(V)}_{i(t),j(t)}.$$

Since $\theta^{(V)}_{i(t),j(t)}$ is drawn uniformly at random from $[1, 2, \ldots, \pi/2]$ (with $\Omega(\log(n) + P) = \mathcal{O}(\log(k))$ bits of precision), the probability that the second equality holds with more than one non-zero term in the summation on the left is at most $o(1/\operatorname{poly}(k))$. The above analysis gives an upper bound on the probability of an incorrect update for a single iteration to be $o(1/\operatorname{poly}(k))$. Finally, as the total number of updates is at most k,

by applying a union bound over all updates, the probability of incorrect decoding is bounded from above by $o(1/\operatorname{poly}(k))$.

3.2.7 Remarks on the Reconstruction process for exactly k-sparse signals

We elaborate on these choices of entries of A in the remarks below, which also gives intuition about the reconstruction process outlined in Section 3.2.4.

Remark 2. In fact, it is not critical that (3.2) be used to assign the identification entries. As long as j can be "quickly" (computationally efficiently) identified from the phases of $a_{i,j}^{(I)}$ (as outlined in Remark 3 below, and specified in more detail in Section 3.2.4), this suffices for our purpose. This is the primary reason we call these entries identification entries.

Remark 3. The reason for the choice of phases specified in (3.2) is as follows. Suppose $\mathcal{S}(\mathbf{x})$ corresponds to the support (set of non-zero values) of $\mathbf{x}$. Suppose y_i corresponds to a $\mathcal{S}(\mathbf{x})$-leaf node, then by definition $y_i^{(I)}$ equals $a_{i,j}^{(I)} x_j$ for some j in $\{1, \ldots, n\}$ (if y_i corresponds to a $\mathcal{S}(\mathbf{x})$-*non*-leaf node, then in general $y_i^{(I)}$ depends on two or more x_j). But x_j is a real number. Hence examining the phase of y_i enables one to efficiently compute $j\pi/(2n)$, and hence j. It *also* allows one to recover the magnitude of x_j, simply by computing the magnitude of y_i.

Remark 4. The choice of phases specified in (3.2) divides the set of allowed phases (the interval $[0, \pi/2]$) into n distinct values. Two things are worth noting about this choice.

1. We consider the interval $[0, \pi/2]$ rather than the full range $[0, 2\pi)$ of possible phases since we wish to use the phase measurements to *also* recover the sign of x_js. If the phase of y_i falls within the interval $[0, \pi/2]$, then (still assuming that y_i corresponds to a $\mathcal{S}(\mathbf{x})$-leaf node)

x_j must have been positive. On the other hand, if the phase of y_i falls within the interval $[\pi, 3\pi/2]$, then x_j must have been negative. (It can be directly verified that the phase of a $\mathcal{S}(\mathbf{x})$-leaf node y_i can never be outside these two intervals – this wastes roughly half of the set of possible phases we could have used for identification purposes, but it makes notation easier.

2. The choice in (3.2) divides the interval $[0, \pi/2]$ into n distinct values. However, in expectation over $\mathcal{G}$ the actual number of non-zero entries in a row of A is $\mathcal{O}(n/k)$, so on average one only needs to choose $\mathcal{O}(n/k)$ distinct phases in (3.2), rather than the worst case n number of values. This has the advantage that one only needs $\mathcal{O}(\log(n/k))$ bits of precision to specify distinct phase values (and in fact we claim that this is the level of precision required by our algorithm). However, since we analyze only left-regular $\mathcal{G}$, the degrees of nodes on the right will in general vary stochastically around this expected value. If k is "somewhat large" (for instance $k = \Omega(n)$), then the degrees will not be very tightly concentrated around their mean. One way around this is to choose $\mathcal{G}$ uniformly at random from the set of bipartite graphs with n nodes (each of degree d) on the left and m nodes (each of degree dn/m) on the right. This would require a more intricate proof of the $\mathcal{S}'(\mathbf{x})$-expansion property defined in Property 3 and proved in Lemma 1. For the sake of brevity, we omit this proof here.

Remark 5. We can further reduce the number of measurements by combining the identification and verification measurements in the following way.

1. Measurement design: If $\mathcal{G}$ has no edge connecting node j on the left with i on the right, then the *identification entry* $a_{i,j}^{(I)}$ is set to equal 0.

Else, if there is indeed such an edge, $a_{i,j}^{(I)}$ is set to equal

$$a_{i,j} = e^{\iota(j\pi/(2n)+\theta_{i,j})}. \tag{3.2}$$

Here, $\theta_{i,j}$ is chosen uniformly at random from $[-\pi/(4n), \pi/(4n)]$ (with $\mathcal{O}(\log(k))$ bits of precision).

2. Reconstruction: The reconstruction algorithm proceeds similarly to the one described earlier. The only modifications are in the identification and verification steps for each iteration. We describe below the modified identification and verification steps for the first iteration of the algorithm. The identification and verification steps for the other iterations proceed similarly by replacing the vector $\mathbf{y}$ by the corresponding residual measurement vector $\tilde{\mathbf{y}}$ for that iteration. Let $\lfloor\cdot\rceil$ be used to denote the nearest integer function. In the identification step, the decoder first identifies $j = \lfloor \angle(y_i)/(\pi/(2n))\rceil$. Next, the decoder verifies that i is indeed a leaf node corresponding to the right node j by verifying if $\theta_{i,j}$ equals $\angle(y_i) - \lfloor \angle(y_i)/(\pi/(2n))\rceil(\pi/(2n))$. If yes, the node i is declared to be a leaf node corresponding to the left node j and the value of x_j is decoded.

This more careful design of measurement design leads to the following corollary.

Corollary 2. *Let $k \leq n$. There exists a reconstruction algorithm SHO-FA for $A \in \mathbb{C}^{m\times n}$ with the following properties:*

i) For every k-sparse $\mathbf{x} \in \mathbb{R}^n$, with probability $1 - \mathcal{O}(1/k)$ over the choice of A, SHO-FA produces a reconstruction $\hat{\mathbf{x}}$ such that $\|\mathbf{x} - \hat{\mathbf{x}}\|_1/\|\mathbf{x}\|_1 \leq 2^{-P}$,

ii) The number of measurements $m = ck$ for some $c > 0$,

iii) The number of steps required by SHO-FA is $\mathcal{O}(k)$, and

iv) The number of bitwise arithmetic operations required by SHO-FA is $\mathcal{O}(k(\log(n) + P))$.

Remark 6. In fact, the recent work of [70] demonstrates an alternative analytical technique (bypassing the expansion arguments outlined in this work), involving analysis of properties of the "2-core" of random hypergraphs, that allows for a tight characterization of the number of measurements required by SHO-FA to reconstruct $\mathbf{x}$ from $\mathbf{y}$ and A, rather than the somewhat loose (though order-optimal) bounds presented in this work. Since our focus in this work is a simple proof of order-optimality (rather than the somewhat more intricate analysis required for the tight characterization) we again omit this proof here.[21]

3.2.8 SHO-FA v.s. "2-core" of random hyper-graphs

We reprise some concepts pertaining to the analysis of random hypergraphs (from [102]), which are relevant to our work.

2-cores of d-uniform hypergraphs: A *d-uniform hypergraph with m nodes and k hyperedges* is defined over a set of m nodes, with each d-uniform hyperedge corresponding to a subset of the nodes of size exactly d. The 2-*core* is defined as the largest *sub-hypergraph* (subset of nodes, and hyperedges defined only on this subset) such that each node in this subset is contained in at least 2 hyperedges on this subset.

A standard "peeling process" that computationally efficiently finds the 2-core is as follows: while there exists a node with degree 1 (connected to just one hyperedge), delete it and the hyperedges containing it.

[21] We thank the anonymous reviewers who examined a draft version of [10] for pointing out the extremely relevant techniques of [70] and [118] (though the problems considered in those works were somewhat different).

The relationship between 2-cores in d-uniform hypergraphs and SHO-FA: As in [70] and other works, there exists a bijection between d-uniform hypergraphs and d-left-regular bipartite graphs, which can be constructed as follows:

(a) Each hyperedge in the hypergraph is mapped to a left node in the bipartite graph,

(b) Each node in the hypergraph is mapped to a right node in the bipartite graph, and

(c) The edges leaving a left-node in the bipartite graph correspond to the nodes contained in the corresponding hyperedge of the hypergraph.

Suppose the d-uniform hypergraph does not contain a 2-core. This means that, in each iteration of "peeling process", we can find a vertex with degree 1, delete it and the corresponding hyperedges, and continue the iterations until all the hyperedges are deleted. Correspondingly, in the bipartite graph, we can find a leaf node in each iteration, delete it and the corresponding left node and continue the iterations until all left nodes are deleted. We note that the SHO-FA algorithm follows essentially the same process. This implies that SHO-FA succeeds if and only if the d-uniform hypergraph contains a 2-core.

Existence of 2-cores in d-uniform hypergraphs and SHO-FA: We now reprise a nice result due to [102] that helps us tighten the results of Theorem 3.

Theorem 5. *([102]) Let $d + l > 4$ and G be a d-uniform random hypergraph with k hyperedges and m nodes. Then there exists a number $c^*_{d,l}$ (independent of k and m) that is the threshold for the appearance of an l-core in G. That is, for constant c and $m \to \infty$: If $k/m = c < c^*_{d,l}$, then*

G has an empty l-core with probability $1 - o(1)$*;If* $k/m = c > c^*_{d,l}$*, then G has an l-core of linear size with probability* $1 - o(1)$.

Specifically, the results in [70, Theorem 1] (which explicitly calculates some of the $c^*_{d,l}$) give that for $l = 2$ and $d = 3$, $c^*_{3,2} = 1/1.22$ with probability $1 - \mathcal{O}(1/k)$. This leads to Theorem 2, that has tighter parameters than Theorem 3.

Remark 7. We note that by carefully choosing a *degree distribution* (for instance the "enhanced harmonic degree distribution" [93]) rather than constant degree d for left nodes in the bipartite graph, the constant c can be made to approach to 1, while still ensuring that 2-cores do not occur with sufficiently high probability. This can further reduce the number of measurements m. However, this can come at a cost in terms of computational complexity, since the complexity of SHO-FA depends on the average degree of nodes on the left, and this is not constant for the enhanced harmonic degree distribution.

Remark 8. The results in Theorem 5 also indicate a "phase transition" on the emergence of l-cores. These results explain our simulation results, presented in Section 3.4.

3.2.9 Other properties of SHO-FA

Database queries

A useful property of our construction of the matrix A is that any desired signal component x_j can be reconstructed in constant time with a constant probability from measurement vector $\mathbf{y} = A\mathbf{x}$. The following Lemma makes this precise. The proof follows from a simple probabilistic argument and is included in Appendix A.1.4.

Lemma 4. *Let* $\mathbf{x}$ *be exactly* k*-sparse. Let* $j \in \{1, 2, \ldots, n\}$ *and let* $A \in \mathbb{C}^{ck \times n}$ *be randomly drawn according to SHO-FA. Then, (with probability at least* $(1-(d/c)^d)$*) there exists an algorithm* $\mathcal{A}$ *such that given inputs* $(j, \mathbf{y})$*,* $\mathcal{A}$ *produces an output* $\hat{x}_j$ *such that* $\hat{x}_j = x_j$ *with probability* $1-o(1/\operatorname{poly}(k))$*.*

SHO-FA for sparse vectors in different bases

In the setting of SHO-FA we consider k-sparse input vectors $\mathbf{x}$. In fact, we also can deal with the case that $\mathbf{x}$ is sparse in a certain basis that is known *a priori* to the decoder,[22] say Ψ, which means that $\mathbf{x} = \Psi\mathbf{w}$ where $\mathbf{w}$ is a k-sparse vector. Specifically, in this case we write the measurement vector as $\mathbf{y} = B\mathbf{x}$, where $B = A\Psi^{-1}$. Then, $\mathbf{y} = A\Psi^{-1}\Psi\mathbf{w} = A\mathbf{w}$, where A is chosen based on the structure of the $\mathcal{G}$ and $\mathbf{w}$ is a k-sparse vector. We can then apply SHO-FA to reconstruct $\mathbf{w}$ and consequently $\mathbf{x} = \Psi\mathbf{w}$. What has been discussed here covers the case where $\mathbf{x}$ is sparse itself, for which we can simply take $\Psi = I$ and $\mathbf{x} = \mathbf{w}$.

Information-theoretically order-optimal encoding/update complexity

The sparse structure of A also ensures ("for free") order-optimal encoding and update complexity of the measurement process.

We first note that for any measurement matrix A that has a "high probability" (over A) of correctly reconstructing arbitrary $\mathbf{x}$, there is a lower bound of $\Omega(n)$ on the computational complexity of computing $A\mathbf{x}$. This is because if the matrix does not have at least $\Omega(n)$ non-zero entries, then with probability $\Omega(1)$ (over A) at least one non-zero entry of $\mathbf{x}$ will "not be measured" by A, and hence cannot be reconstructed. In the SHO-FA algorithm, since the degree of each left-node in $\mathcal{G}$ is a constant (d),

[22] For example, "smooth" signals are sparse in the Fourier basis and "piecewise smooth" signals are sparse in wavelet bases.

the encoding complexity of our measurement process corresponds to dn multiplications and additions.

Analogously, the complexity of updating $\mathbf{y}$ if a single entry of $\mathbf{x}$ changes is at most d for SHO-FA, which matches the natural lower bound of 1 up to a constant (d) factor.

Information-theoretically optimal number of bits

We recall that the reconstruction goal for SHO-FA is to reconstruct $\mathbf{x}$ up to relative error 2^{-P}. That is,

$$\|\mathbf{x} - \hat{\mathbf{x}}\|_1 / \|\mathbf{x}\|_1 \leq 2^{-P}.$$

We first present a sketch of an information-theoretic lower bound of $\Omega\left(k\left(P + \log(n)\right)\right)$ bits which holds for any algorithm that outputs a k-sparse vector that achieves this goal with high probability.

To see that this is true, consider the case where the locations of k non-zero entries in $\mathbf{x}$ are chosen uniformly at random among all the n, entries and the value of each non-zero entry is chosen uniformly at random from the set $\{1, \ldots, 2^P\}$. Then recovering even the support requires at least $\log\left(2^{kP}\binom{n}{k}\right)$ bits, which is $\Omega(kP + k\log(n/k))$.[23] Also, at least a constant fraction of the k non-zero entries of $\mathbf{x}$ must be be correctly estimated to guarantee the desired relative error. Hence $\Omega\left(k\left(P + \log(n)\right)\right)$ is a lower bound on the measurement bit-complexity.

The following arguments show that the total number of bits used in our algorithm is information-theoretically order-optimal for any $k = \mathcal{O}(n^{1-\Delta})$ (for any $\Delta > 0$). First, to represent each non-zero entry of $\mathbf{x}$, we need to use arithmetic of $\Omega(P + \log(k))$ bit precision. Here the P term is so as to attain the required relative error of reconstruction, and the $\log(k)$

[23] Stirling's approximation(c.f. [94, Chapter 1]) is used in bounding from below the combinatorial term $\binom{n}{k}$.

term is to take into account the error induced by finite-precision arithmetic (say, for instance, by floating point numbers) in $\mathcal{O}(k)$ iterations (each involving a constant number of finite-precision additions and unit-magnitude multiplications). Second, for each identification step, we need to use $\Omega(\log(n) + \log(k))$ bit-precision arithmetic. Here the $\log(n)$ term is so that the identification measurements can uniquely specify the locations of non-zero entries of $\mathbf{x}$. The $\log(k)$ term is again to take into account the error induced in $\mathcal{O}(k)$ iterations. Third, for each verification step, the number of bits we use is (say) $3\log(k)$. Here, by the Schwartz-Zippel Lemma [127, 154], $2\log(k)$ bit-precision arithmetic guarantees that each verification step is valid with probability at least $1 - 1/k^2$ – a union bound over all $\mathcal{O}(k)$ verification steps guarantees that all verification steps are correct with probability at least $1 - \mathcal{O}(1/k)$ (this probability of success can be directly amplified by using higher precision arithmetic). Therefore, the total number of bits needed by SHO-FA is $\mathcal{O}(k(\log(n) + P))$. As claimed, this matches, up to a constant factor, the lower bound sketched above.

Universality

While the ensemble of matrices $\{A\}$ we present above has carefully chosen identification entries, and all the non-zero verification entries have unit magnitude, as noted in Remark 1, the implicit ideas underlying SHO-FA work for significantly more general ensembles of matrices. In particular, Property 1 only requires that the graph $\mathcal{G}$ underling A be "sparse", with a constant number of non-zero entries per column. Property 2 only requires that each non-zero entry in each row be distinct – which is guaranteed with high probability, for instance, if each entry is chosen $i.i.d$ from any distribution with sufficiently large support. An example of such a scenario is shown in Figure 3.10. This naturally motivates the application of SHO-FA

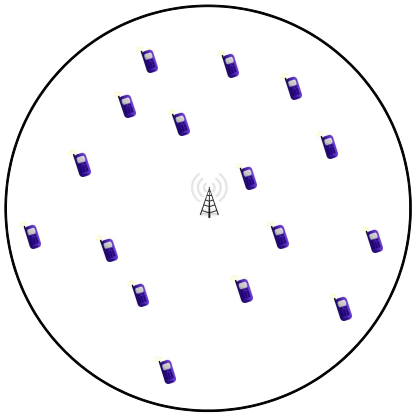


Figure 3.10: **An example of a physical system that "naturally" generates ensembles of sparse A that SHO-FA can use**: Suppose there are k cellphones (out of a set of n possible different cellphones in the whole world) in a certain neighbourhood that has a base-station. The goal is for the j-th cellphone to communicate its information (x_j) to the base-station at least once per *frame* of ck consecutive time-slots. The challenge is to do so in a distributed manner, since multiple cellphones transmitting at the same time i would result in a linear combination $y_i = \sum_j a_{ij} x_j$ of their transmissions reaching the base-station, where a_{ij} corresponds to the channel gain from the cellphone j to the base-station during time-slot i. With high probability, such a_{ij} satisfying the properties we require for our algorithm to work – "sparsity" (relatively few transmitters transmit at any given time) and "distinctness" (with high probability the channel gains from different transmitters are different). Each cellphone transmits x_j to the base-station a constant (d) number of times in each frame – the set of d time-slots in each frame that cellphone j transmits in is chosen by j uniformly at random from the set of all $\binom{ck}{d}$ sets of slots.

to a variety of scenarios, for *e.g.*, neighbor discovery in wireless communication [71].

3.3 Approximate reconstruction in the presence of noise

A prominent aspect of the design presented in the previous section is that it relies on exact determination of all the phases as well as magnitudes of the measurement vector $A\mathbf{x}$. In practice, however, we often desire that the measurement and reconstruction be robust to corruption both before and and during measurements. In this section, we show that our design may be modified slightly such that with a suitable decoding procedure, the reconstruction is robust to such "noise" (See Figure 3.12 for the effect of noise on a measurement).

We consider the following setup. Let $\mathbf{x} \in \mathbb{R}^n$ be a k-sparse signal with support $\mathcal{S}(\mathbf{x}) = \{j : x_j \neq 0\}$. Let $\mathbf{z} \in \mathbb{R}^n$ have support $\{1, 2, \ldots, n\} \setminus \mathcal{S}(\mathbf{x})$ with each z_j distributed according to a Gaussian distribution with mean 0 and variance σ_z^2. Denote the measurement matrix by $A \in \mathbb{C}^{m \times n}$ and the measurement vector by $\mathbf{y} \in \mathbb{C}^m$. Let $\mathbf{e} \in \mathbb{C}^m$ be the measurement noise with e_i distributed as a complex Gaussian with mean 0 and variance σ_e^2 along each axis. The vector $\mathbf{y}$ is related to the signal as

$$\mathbf{y} = A(\mathbf{x} + \mathbf{z}) + \mathbf{e}.$$

We first propose a design procedure for A satisfying the properties stated in Theorem 3. We present a "simple" proof of Theorem 3 in Appendix. In Theorem 4, we outline an analysis (based on the work of [118]) that leads to a tighter characterization of the constant factors in the parameters of Theorem 3.

Recall that in the exactly k-sparse case, the decoding step in t-th iteration relies on first finding an $\mathcal{S}(t)$-leaf node, then decoding the correspond-

ing signal coordinate and updating the undecoded measurements. In this procedure, it is critical that each iteration operates with low reconstruction errors as an error in an earlier iteration can propagate and cause potentially catastrophic errors. In general, one of the following events may result in any iteration ending with a decoded signal value that is far from the true signal value:

(a) The decoder picks an index outside the set $\{i : (A\mathbf{x})_i \neq 0\}$, but in the set $\{i : (A(\mathbf{x}+\mathbf{z})+\mathbf{e})_i \neq 0\}\}$.

(b) The decoder picks an index within the set $\{i : (A\mathbf{x})_i \neq 0\}$ that is also a leaf for $\mathcal{S}$ with parent node j, but the presence of noise results in the decoder identifying (and verifying) a node $j' \neq j$ as the parent and, subsequently, incorrectly decoding the signal at j'.

(c) The decoder picks an index within the set $\{i : (A\mathbf{x})_i \neq 0\}$ that is not a leaf for $\mathcal{S}$, but the presence of noise results in the decoder identifying (and verifying) a node j as the parent and, subsequently, incorrectly decoding the signal at j.

(d) The decoder picks an index within the set $\{i : (A\mathbf{x})_i \neq 0\}$ that is a leaf for $\mathcal{S}$ with parent node j, which it also identifies (and verifies) correctly, but the presence of noise introduces a small error in decoding the signal value. This error may also propagate to the next iteration and act as "noise" for the next iteration.

To overcome these hurdles, our design takes the noise statistics into account to ensure that each iteration is resilient to noise with a high probability. This is achieved through several new ideas that are presented in the following. Next, we perform a careful analysis of the corresponding decoding algorithm and show that under certain regularity conditions, the

overall failure probability can be made arbitrarily small to output a reconstruction that is robust to noise. Key to this analysis is bounding the effect of propagation of estimation error as the decoder steps through the iterations.[24]

3.3.1 Key ideas

Truncated reconstruction

We observe that in the presence of noise, it is unlikely that signal values whose magnitudes are comparable to that of the noise values can be successfully recovered. Thus, it is futile for the decoder to try to reconstruct these values as long as the overall penalty in ℓ_1-norm is not high. The following argument shows that this is indeed the case. Let

$$\mathcal{S}_\delta(\mathbf{x}) = \{j : |x_j| < \delta/k\}, \tag{3.3}$$

and let $\mathbf{x}_{\mathcal{S}_\delta}$ be the vector defined as

$$(x_{\mathcal{S}_\delta})_j = \begin{cases} 0, & j \notin \mathcal{S}_\delta(\mathbf{x}), \\ x_j, & j \in \mathcal{S}_\delta(\mathbf{x}). \end{cases}$$

Similarly, define $\mathbf{x}_{\mathcal{S}_\delta^c}$ as the projection of $\mathbf{x}$ which has non-zero entries only within the set $\mathcal{S}(\mathbf{x}) \setminus \mathcal{S}_\delta(\mathbf{x})$ (See Figure 3.11 for illustration). The following sequence of inequalities shows that the total ℓ_1 norm of $\mathbf{x}_{S_\delta}$ is small:

$$\|\mathbf{x}_{\mathcal{S}_\delta}\|_1 \quad = \quad \sum_{j \in \mathcal{S}_\delta(\mathbf{x})} |x_j|$$

[24] For simplicity, the analysis presented here relies only on an upper bound on the length of the path through which the estimation error introduced in any iteration can propagate. This bound follows from known results on size of largest components in sparse hypergraphs [85]. We note, however, that a tighter analysis that relies on a finer characterization of the interaction between the size of these components and the contribution to total estimation error may lead to better bounds on the overall estimation error. Indeed, as shown in [118], such an analysis enables us to achieve a tighter reconstruction guarantee of the form $\|\mathbf{x}-\hat{\mathbf{x}}\|_1 = \mathcal{O}(\|\mathbf{z}\|_1 + \|\mathbf{e}\|_1)$.

$$\begin{aligned} &\leq |\mathcal{S}_\delta(\mathbf{x})|\frac{\delta}{k} \\ &\leq |\mathcal{S}(\mathbf{x})|\frac{\delta}{k} \\ &= \delta. \end{aligned} \tag{3.4}$$

Further, as an application of triangle inequality and the bound in (3.4), it follows that

$$\begin{aligned} \|\hat{\mathbf{x}} - \mathbf{x}\|_1 &= \|\hat{\mathbf{x}} - \mathbf{x}_{\mathcal{S}_\delta^c} - \mathbf{x}_{\mathcal{S}_\delta}\|_1 \\ &\leq \|\hat{\mathbf{x}} - \mathbf{x}_{\mathcal{S}_\delta^c}\|_1 + \|\mathbf{x}_{\mathcal{S}_\delta}\|_1 \\ &\leq \|\hat{\mathbf{x}} - \mathbf{x}_{\mathcal{S}_\delta^c}\|_1 + \delta. \end{aligned} \tag{3.5}$$

Keeping the above in mind, we rephrase our reconstruction objective to satisfy the following criterion with a high probability:

$$\|\hat{\mathbf{x}} - \mathbf{x}_{\mathcal{S}_\delta^c}\|_1 \leq C_1(\|\mathbf{z}\|_1 + \sqrt{\log(k)}\|\mathbf{e}\|_1), \tag{3.6}$$

while simultaneously ensuring that our choice of parameter δ satisfies

$$\delta < C_2\|\mathbf{z}\|_1 \tag{3.7}$$

for some C_2, with high probability.

Phase quantization

In the noisy setting, even when i is a leaf node for $\mathcal{S}(\mathbf{x})$, the phase of y_i may differ from the phase assigned by the measurement. This is geometrically shown in Figure 3.12a for a measurement matrix A'. To overcome this, we modify our decoding algorithm to work with "quantized" phases, rather than the actual received phases. The idea behind this is that if i is a leaf node for $\mathcal{S}(\mathbf{x})$, then quantizing the phase to one of the values allowed by the measurement identifies the correct phase with a high probability. The following lemma facilitates this simplification.

Lemma 5 (*Almost bounded phase noise*). *Let* $\mathbf{x}, \mathbf{z} \in \mathbb{R}^n$ *with* $|x_j| > \delta/k$ *for each* j. *Let* $A' \in \mathbb{C}^{m' \times n}$ *be a complex valued measurement matrix with the underlying graph* $\mathcal{G}$. *Let* i *be a leaf node for* $\mathcal{S}(\mathbf{x})$. *Let* $\Delta\theta_i = |\angle y_i - \angle(A'\mathbf{x})_i|$. *Then, for every* $\alpha > 0$,

$$E_{\mathbf{z},\mathbf{e}}(\Delta\theta_i) \leq \sqrt{\frac{2\pi k^2(dn\sigma_z^2/ck + \sigma_e^2)}{\delta^2}}$$

and

$$\Pr_{\mathbf{z},\mathbf{e}}\left(\Delta\theta_i > \alpha E_{\mathbf{z},\mathbf{e}}(\Delta\theta_i)\right) < \frac{1}{2}e^{-(\alpha^2/2\pi)}.$$

Proof. See Appendix A.1.5. ■

For a desired error probability ϵ', the above lemma stipulates that it suffices to let $\alpha = \sqrt{2\pi \log(1/2\epsilon')}$. We examine the effect of phase noise in more detail in Appendix A.1.6.

Repeated measurements

Our algorithm works by performing a series of $\Gamma \geq 1$ identification and verification measurements in each iteration instead of a single measurement of each type as done in the exactly k-sparse case. The idea behind this is that, in the presence of noise, even though a single set of identification and verification measurements cannot exactly identify the coordinate j from the observed y_i, it helps us narrows down the set of coordinates j that can possibly contribute to give the observed phase. Performing measurements repeatedly, each time with a different measurement matrix, helps us identify a single j with high probability.

We implement the above idea by first mapping each $j \in \{1, 2, \ldots, n\}$ to its Γ-digit representation in base $\mathbb{G} = \{0, 1, \ldots \lceil n^{1/\Gamma} - 1 \rceil\}$. For each $j \in \{1, 2, \ldots, n\}$, let $g(j) = (g_1(j), g_2(j), \ldots, g_\Gamma(j))$ be the Γ-digit representation of j. Next, perform one pair of identification and verification measurements (and corresponding phase reconstructions), each of which is

intended to distinguish exactly one of the digits. In our construction, we only need a constant number of such phase measurements per iteration. See Figure 3.13 for an illustrating example.

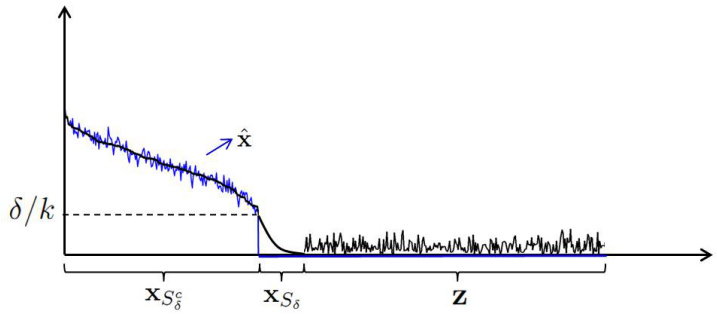
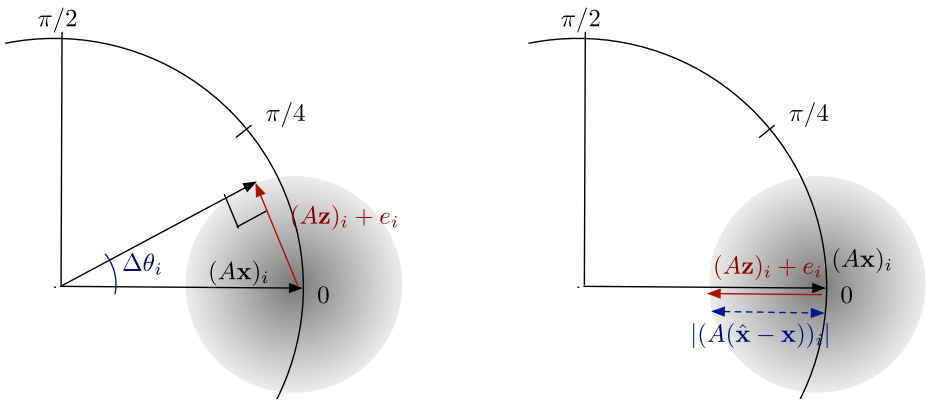


Figure 3.11: The black curve corresponds to the magnitudes of $\mathbf{x}+\mathbf{z}$ (for ease of visual presentation, the components of $\mathbf{x}$ have been sorted in decreasing order of magnitude and placed in the first k components of the signal, but the components of $\mathbf{z}$ are unsorted. The blue curve corresponds to our reconstruction $\hat{\mathbf{x}}$ of $\mathbf{x}$. Note that we only attempt to reconstruct components of $\mathbf{x}$ that are "sufficiently large" (that is, we make no guarantees about correct reconstruction of components of $\mathbf{x}$ in $\mathcal{S}_\delta(\mathbf{x})$, , *i.e*, those components of $\mathbf{x}$ that are smaller than some "threshold" δ/k. Here δ is a parameter of code-design to be specified later. As shown in Section 3.3.1, as long as δ is not "too large", this relaxation does not violate our relaxed reconstruction criteria (3.6).

3.3.2 Measurement Design

As in the exactly k-sparse case, we start with a randomly drawn left regular bipartite graph $\mathcal{G}$ with n nodes on the left and m' nodes on the right.

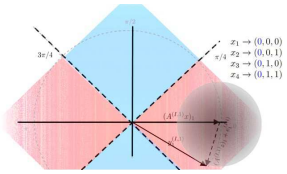
Measurement matrix: The measurement matrix $A \in \mathbb{C}^{2m'\Gamma \times n}$ is chosen based on the graph $\mathcal{G}$. The rows of A are partitioned into m' groups, with each group consisting of 2Γ consecutive rows. The j-th entries of the rows



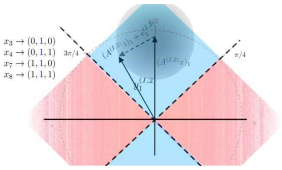
(a) Maximum phase displacement occurs when the contribution due to noise, i.e., $(A\mathbf{z})_i + e_i$ is orthogonal to the measurement $\mathbf{y}_i$

(b) Maximum magnitude displacement takes place when the contribution due to noise is aligned with $(A\mathbf{x})_i$

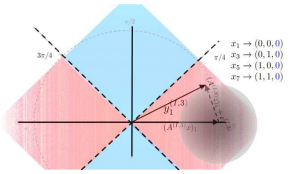
Figure 3.12: The effect of noise on a measurement output.



(a) The decoder "randomly" picks y_1. Since the phase of $y_1^{(I,1)}$ is between $-\pi/4$ and $\pi/4$, the decoder can distinguish that the first bit of non-zero location is 0 since the decoder can tolerate at most $\pi/4$ phase displacement for y_1. So, the non-zero entry is one of x_1, x_2, x_3, or x_4.



(b) The decoder "randomly" picks y_1 again. Since the phase of $y_1^{(I,2)}$ is between $3\pi/4$ and $5\pi/4$, the decoder can distinguish that the second bit of non-zero location is 1 since the decoder can tolerate at most $\pi/4$ phase displacement for y_1. So, the non-zero entry is one of x_3, x_4, x_7, x_8. Combing the output in the first phase measurement, we conclude that the non-zero entry is one of x_3 and x_4.



(c) The decoder "randomly" picks y_1 again. Since the phase of $y_1^{(I,3)}$ is between $-\pi/4$ and $\pi/4$, the decoder can distinguish that the third bit of non-zero location is 0 since the decoder can tolerate at most $\pi/4$ phase displacement for y_1. So, the non-zero entry is one of x_1, x_3, x_5, x_7. Combing the outputs in the first and second phase measurement, we conclude that the non-zero entry is x_3.

Figure 3.13: If we were to distinguish each j from 1 to 8 by a different phase, the decoder can tolerate at most $\pi/14$ phase displacement for any output y_i. Instead, we first represent each $j = 1, 2, \ldots, 8$ by a length-three binary vector. Next, we perform three sets of phase assignments – one for each digit. It is easily seen that by allowing multiple measurements, the noise tolerance for the decoder increases.

$2(i-1)\Gamma+1, (i-1)\Gamma+2, \ldots, 2i\Gamma$ are denoted by

$$a_{i,j}^{(I,1)}, a_{i,j}^{(I,2)}, \ldots, a_{i,j}^{(I,\Gamma)}, a_{i,j}^{(V,1)}, a_{i,j}^{(V,2)}, \ldots, a_{i,j}^{(V,\Gamma)}$$

respectively. In the above notation, I and V are used to refer to identification and verification measurements.

For ease of notation, for each $\gamma = 1, 2, \ldots, \Gamma$, we use $A^{(I,\gamma)}$ (respectively $A^{(V,\gamma)}$) to denote the sub-matrix of A whose (i,j)-th entry is $a_{i,j}^{(I,\gamma)}$ (respectively $a_{i,j}^{(V,\gamma)}$).

We define the γ-th identification matrix $A^{(I,\gamma)}$ as follows. For each (i,j), if the graph $\mathcal{G}$ does not have an edge connecting i on the right to j on the left, then $a_{i,j}^{(I,\gamma)} = 0$. Otherwise, we set $a_{i,j}^{(I,\gamma)}$ to be the unit-norm complex number

$$a_{i,j}^{(I,\gamma)} = e^{\iota g_\gamma(j)\pi/2(|\mathbb{G}|-1)}.$$

Note here that the construction for the exactly k-sparse case can be recovered by setting $\Gamma = 1$, which results in $\mathbb{G} = \{1, 2, \ldots, n\}$ and $g_\gamma(j) = j$.

Next, we define the γ-th verification matrix $A^{(V,\gamma)}$ in a way similar to how we defined the verification entries in the exactly k-sparse case. For each (i,j), if the graph $\mathcal{G}$ does not have an edge connecting i on the right to j on the left, then $a_{i,j}^{(V,\gamma)} = 0$. Otherwise, we set

$$a_{i,j}^{(V,\gamma)} = e^{\iota\theta_{ij}^{(V,\gamma)}},$$

where $\theta_{i,j}^{(V,\gamma)}$ is drawn uniformly at random from $\{0, \pi/2(|\mathbb{G}|-1), \pi/(|\mathbb{G}|-1), 3\pi/2(|\mathbb{G}|-1) \ldots, \pi/2\}$.

Given a signal vector $\mathbf{x}$, signal noise $\mathbf{z}$, and measurement noise $\mathbf{e}$, the measurement operation produces a measurement vector $\mathbf{y} = A(\mathbf{x}+\mathbf{e})$. Since A can be partitioned into Γ identification and Γ verification rows, we think of the measurement vector $\mathbf{y}$ as a collection of outcomes from Γ successive measurement operations such that

$$\mathbf{y}^{(I,\gamma)} = A^{(I,\gamma)}(\mathbf{x}+\mathbf{z}) + \mathbf{e}^{(I,\gamma)}$$

and

$$\mathbf{y}^{(V,\gamma)} = A^{(V,\gamma)}(\mathbf{x} + \mathbf{z}) + \mathbf{e}^{(V,\gamma)}$$

are the outcomes from the γ-th measurement and $\mathbf{y} = ((\mathbf{y}^{(I,\gamma)}, \mathbf{y}^{(V,\gamma)}) : 1 \leq \gamma \leq \Gamma)$.

3.3.3 Reconstruction

The decoding algorithm for this case extends the decoding algorithm presented earlier for the exactly k-sparse case by including the ideas presented in Section 3.3.1. The total number of iterations for our algorithm is upper bounded by $4k$.

1. We initialize by setting the *signal estimate vector* $\hat{\mathbf{x}}(1)$ to the all-zeros vector 0^n, and for each $\gamma = 1, 2, \ldots, \Gamma$, we set the *residual measurement identification/verification vectors* $\tilde{\mathbf{y}}^{(I,\gamma)}(1)$ and $\tilde{\mathbf{y}}^{(V,\gamma)}(1)$ to the decoder's observations $\mathbf{y}^{(I,\gamma)}$ and $\mathbf{y}^{(V,\gamma)}$.

   Let $\mathcal{B}(1)$, the initial neighborly set, be the set of indices i for which, at which the magnitude corresponding to all verification and identification vectors is greater than δ/k, *i.e.*,

   $$\mathcal{B}(1) = \bigcap_{\gamma=1}^{\Gamma} \left\{ i : |y_i^{(I,\gamma)}| > \frac{\delta}{k},\ |y_i^{(V,\gamma)}| > \frac{\delta}{k} \right\}.$$

   This step takes $\mathcal{O}(k)$ steps, since merely reading $\mathbf{y}$ to check for the zero locations of $\mathbf{y}^{(V)}$ takes that long.

2. The t^{th} decoding iteration accepts as its input the t^{th} signal estimate vector $\hat{\mathbf{x}}^{(t)}$, the t^{th} neighbourly set $\mathcal{B}(t)$, and the t^{th} residual measurement identification/verification vectors

   $$\left((\tilde{\mathbf{y}}^{(I,\gamma)}(t), \tilde{\mathbf{y}}^{(V,\gamma)}(t)) : \gamma = 1, 2, \ldots, \Gamma\right).$$

In $\mathcal{O}(1)$ steps it outputs the $(t+1)^{th}$ signal estimate vector $\hat{\mathbf{x}}^{(t+1)}$, the $(t+1)^{th}$ neighbourly set $\mathcal{B}(t+1)$, and the $(t+1)^{th}$ residual measurement identification/verification vectors

$$\left((\tilde{\mathbf{y}}^{(I,\gamma)}(t+1), \tilde{\mathbf{y}}^{(V,\gamma)}(t+1)) : \gamma = 1, 2, \ldots, \Gamma\right)$$

after the performing the following steps sequentially (each of which takes at most a constant number of atomic steps).

3. *Pick a random $i(t)$:* The decoder picks $i(t)$ uniformly at random from $\mathcal{B}(t)$

4. *Compute quantized phases:* For each $\gamma = 1, 2, \ldots, \Gamma$, compute the *current identification angles*, $\hat{\theta}_t^{(I,\gamma)}$, and *current identification angles*, $\hat{\theta}_t^{(I,\gamma)}$ defined as follows:

$$\begin{aligned}
\hat{\theta}_t^{(I,\gamma)} &= \left[\frac{2(|\mathbb{G}|-1|)\left(\angle y_{i(t)}^{(I,\gamma)} (\text{mod } \pi)\right)}{\pi}\right] \frac{\pi}{2(|\mathbb{G}|-1|)}, \\
\hat{\theta}_t^{(V,\gamma)} &= \left[\frac{2(|\mathbb{G}|-1|)\left(\angle y_{i(t)}^{(V,\gamma)} (\text{mod } \pi)\right)}{\pi}\right] \frac{\pi}{2(|\mathbb{G}|-1|)}.
\end{aligned}$$

In the above, $[\cdot]$ denotes the closest integer function. Since there are $\Theta(n)$ different phase vectors, to perform this computation, $\mathcal{O}(\log(n))$ bits of precision and $\mathcal{O}(1)$ computation steps suffice.

For each $\gamma = 1, 2, \ldots, \Gamma$, let $\hat{g}_\gamma^{(t)} = 2(|\mathbb{G}|-1|)\hat{\theta}^{(I,\gamma)}/\pi$ be the *current estimate of γ-th digit* and let $j(t)$ be the number whose representation in $\mathbb{G}$ is $(\hat{g}_1^{(t)}, \hat{g}_2^{(t)}, \ldots, \hat{g}_\Gamma^{(t)})$.

5. *Check if the current identification and verification angles correspond to a valid and unique j:* This step determines if $i(t)$ is a leaf node for $\mathcal{S}_\delta(\mathbf{x} - \hat{\mathbf{x}}(t))$. This operation is similar to the corresponding exactly k-sparse case. The main difference here is that we perform verification

operations on each of the Γ measurements separately and declare $i(t)$ as a leaf node only if it passes all the verification tests. The verification step for the γ-th measurement is given by the test:

$$\hat{\theta}_t^{(V,\gamma)} \stackrel{?}{=} \theta_{i(t),j(t)}^{(V,\gamma)}.$$

If the above test succeeds for every $\gamma = 1, 2, \ldots, \Gamma$, we set $\Delta x(t)$ to $|\tilde{y}_{i(t)}^{(I,\gamma)}(t)|$ if $\angle y_{i(t)}^{(I,\gamma)} \in (-\pi/4, 3\pi/4]$, and $-|\tilde{y}_{i(t)}^{(I,\gamma)}(t)|$ if $\angle y_{i(t)}^{(I,\gamma)} \in (3\pi/4, 7\pi/4]$. Otherwise, we set $\Delta x(t) = 0$. This step requires at most Γ verification steps and therefore, can be completed in $\mathcal{O}(1)$ steps.

6. *Update* $\hat{\mathbf{x}}(t+1)$, $\tilde{\mathbf{y}}(t+1)$, *and* $\mathcal{B}(t+1)$*:* If the verification tests in the previous steps failed, there are no updates to be done, *i.e.*, set $\hat{\mathbf{x}}(t+1) = \hat{\mathbf{x}}(t)$, $\tilde{\mathbf{y}}(t+1) = \tilde{\mathbf{y}}(t)$, and $\mathcal{B}(t+1) = \mathcal{B}(t)$.

   Otherwise, we first update the current signal estimate to $\hat{\mathbf{x}}(t+1)$ by setting the $j(t)$-th coordinate to $\Delta x(t)$. Next, let i_1, i_2, i_3 be the possible neighbours of $j(t)$. We compute the *residual identification/verification vectors* $\tilde{\mathbf{y}}(t+1)$ at i_1, i_2, i_3 by subtracting the weight due to $\Delta x(t)$ at each of them. Finally, we update the neighbourly set by removing i_1, i_2, and i_3 from $\mathcal{B}(t)$ to obtain $\mathcal{B}(t+1)$.

The decoding algorithm terminates after the T-th iteration, where $T = \min\{4k, \{t : \mathcal{B}(t+1) = \phi\}\}$.

3.3.4 Improving performance guarantees of SHO-FA via Set-Query Algorithm of [118]

In [118], Price considers a related problem called the *Set-Query problem*. In this setup, we are given an unknown signal vector $\mathbf{x}$ and the objective is to design a measurement matrix A such that given $\mathbf{y} = A\mathbf{x} + \mathbf{e}$ (here, $\mathbf{e}$

is an arbitrary "noise" vector), and a desired *query set* $\mathcal{S} \subseteq \{1, 2, \ldots, n\}$, the decoder outputs a reconstruction $\hat{\mathbf{x}}$ with having support $\mathcal{S}$ such that $\hat{\mathbf{x}} - \mathbf{x}_{\mathcal{S}}$ is "small". The following Theorem from [118] states the performance guarantees for a randomized construction of A.

Theorem 6 (Theorem 3.1 of [118]). *For every $\epsilon > 0$, there is a randomized sparse binary sketch matrix A and recovery algorithm $\mathcal{A}$, such that for any $\mathbf{x} \in \mathbb{R}^n$, $\mathcal{S} \subseteq \{1, 2, \ldots, n\}$ with $|\mathcal{S}| = k$, $\hat{\mathbf{x}} = \mathcal{A}(A\mathbf{x} + \mathbf{e}, \mathcal{S}) \in \mathbb{R}^n$ has support $\mathcal{S}(\hat{\mathbf{x}}) \subseteq \mathcal{S}$ and*

$$\|\hat{\mathbf{x}} - \mathbf{x}_{\mathcal{S}}\|_l \leq (1 + \epsilon)(\|\mathbf{x} - \mathbf{x}_{\mathcal{S}}\|_l + \|\mathbf{e}\|_l)$$

for each $l \in \{1, 2\}$ with probability at least $1 - 1/k$. A has $\mathcal{O}(k)$ rows and $\mathcal{A}$ runs in $\mathcal{O}(k)$ time.

We argue that the above design may be used in conjunction with our SHO-FA algorithm from Theorem 3 to give stronger reconstruction guarantee than Theorem 3. In fact, this allows us to even prove a stronger reconstruction guarantee of the $\ell_2 < \ell_2$ form. Theorem 4 makes this precise.

Proof of Theorem 4: We first note that the measurement matrix proposed in [118] is independent of the query set $\mathcal{S}$ and depends only on the size k of the set $\mathcal{S}$. We design our measurement matrix $A \in \mathbb{C}^{m \times n}$ by combining the measurement matrices from Theorem 3 and Theorem 6 as follows. Let $A_1 \in \mathbb{C}^{m_1 \times n}$ be drawn according to Theorem 3 and $A_2 \in \mathbb{C}^{m_2 \times n}$ be drawn according to Theorem 6 for some m_1 and m_2 scaling as $\mathcal{O}(k)$ so as to achieve an error probability $\mathcal{O}(1/k)$. Let $A \in \mathbb{C}^{m \times n}$ with $m = m_1 + m_2$ with the upper m_1 rows consisting of all rows of A_1 and the lower m_2 rows consisting of all the row of A_2.

To perform the decoding, the decoder first produces a coarse reconstruction $\hat{\hat{\mathbf{x}}}$ by passing the first m_1 rows of the measurement output $\mathbf{y}$ through

the SHO-FA decoding algorithm. Let δ be the truncation threshold for the decoder. Next, the decoder computes $\mathcal{S}_\delta(\mathbf{x})$ to be the support of $\hat{\hat{\mathbf{x}}}$. Finally, the decoder applies the set query algorithm from Theorem 6 with inputs $\left([y_{m_1+2}, \ldots, y_{m_2}]^T, \mathcal{S}_\delta(\mathbf{x})\right)$ to obtain a reconstruction $\hat{\mathbf{x}}$ that satisfies the desired reconstruction criteria. ■

3.4 Simulation Results

This section describes simulations that use synthetic data. The k-sparse signals used here are generated by randomly choosing k locations for non-zero values and setting the non-zero values to 1 for Figures 3.14, 3.15 and 3.16, and to standard uniform random variable for Figure 3.17. The contours in each plot show the probability of successful reconstruction (the lighter the color, the higher the probability of reconstruction). The probability of error at each data point in the plots was obtained by running multiple simulations (400 in Figure 3.14 and Figure 3.15, and 200 in Figure 3.16 and Figure 3.17) and noting the fraction of simulations which resulted in successful reconstruction.

3.5 Acknowledgement

We would like to thank Prof. Dongning Guo for pointing out the compressive neighboring identification problem [152] as a motivation. We thank Prof. Babak Hassibi and Prof. Piotr Indyk for useful discussions – our work is partially inspired by their prior works. We thank the anonymous reviewers who pointed out the connection between the first version of this work and [70] and [118]. We thank Prof. Pascal Vontobel for providing insightful comments on this chapter.

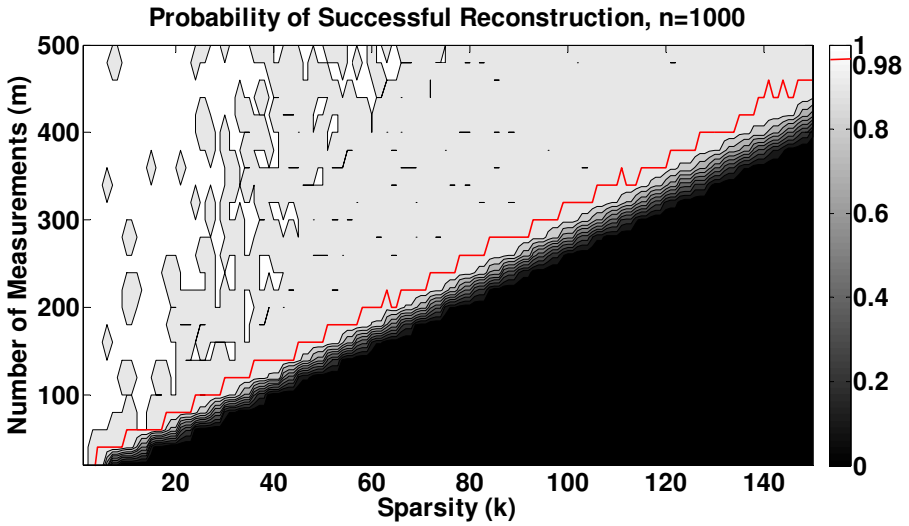


Figure 3.14: **Exactly sparse signal and noiseless measurements – reconstruction performance for fixed signal length n:** The y-axis denotes the number of measurements m, and the x-axis denotes the sparsity k, for fixed signal length $n = 1000$. The simulation results show that the number of measurements m grows roughly proportionally to the sparsity k for a fixed probability of reconstruction error. Also note that there is a sharp transition in reconstruction performance once the number of measurements exceeds a linear multiple of k. The red line denotes the curve where the probability of successful reconstruction equals 0.98. For $k = 150$, the probability of success equals 0.98 when $m = 450$ and $c = m/k = 3$.

3.6 Conclusion

In this chapter we present a suite of algorithms (that we call SHO-FA) for compressive sensing that require an information-theoretically order-optimal number of measurements, bits over all measurements, and encoding, update, and decoding time-complexities. As a bonus, with non-zero probability it can also handle "data-base queries". The algorithms are robust to noisy signal tails and noisy measurements. The algorithms are practical (all constant factors involved are small), as validated by both our analysis, and simulations. Our algorithms can reconstruct signals that are sparse in any basis that is known *a priori* to both the encoder and decoder, and

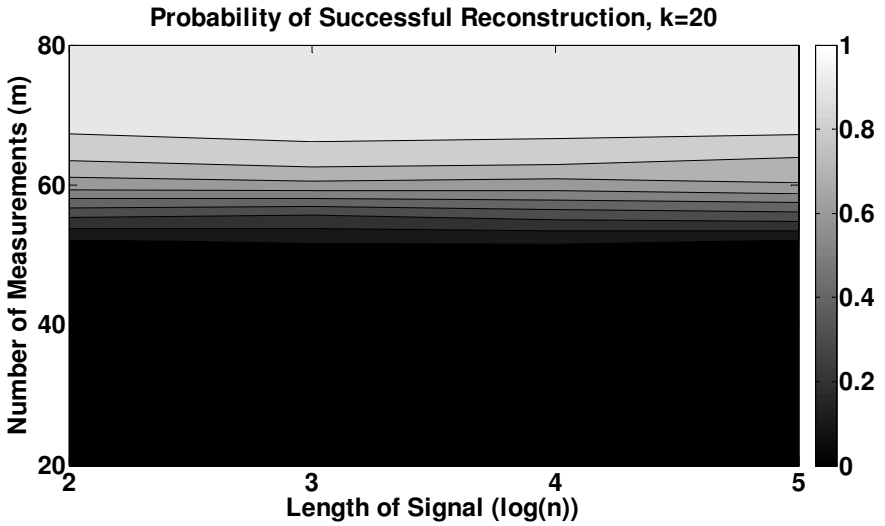


Figure 3.15: **Exactly sparse signal and noiseless measurements – reconstruction performance for fixed sparsity k:** The number of measurements m are plotted on the y-axis, plotted against $\log(n)$ on the x-axis – the sparsity k is fixed to be 20. Note that there is *no* scaling of m with n, as guaranteed by our theoretical bounds.

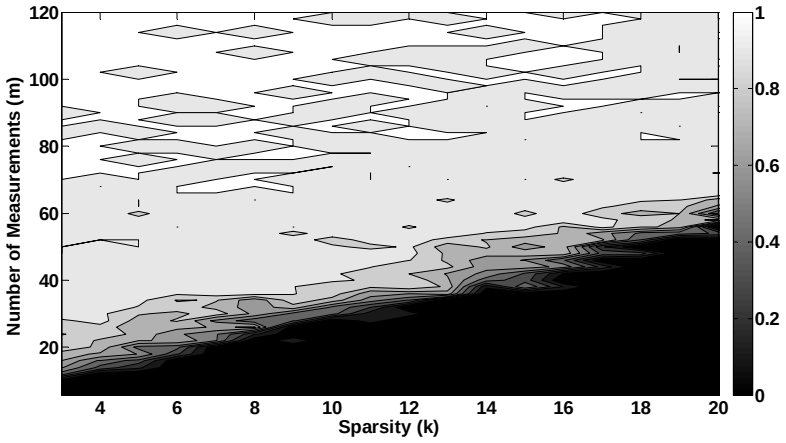


Figure 3.16: **Approximately sparse signal and noisy measurements – reconstruction performance for fixed signal-length n:** As in Fig 3.14, the y-axis denotes the number of measurements m, and the x-axis denotes the sparsity k, for fixed signal length $n = 1000$. In this case, we set $\sigma_z = 0.03$, and allowed relative reconstruction error of at most 0.3.

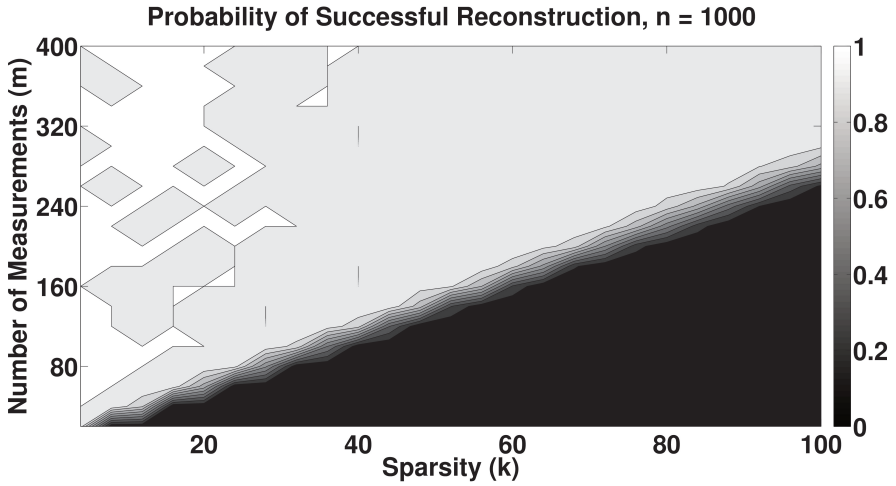


Figure 3.17: **Exactly sparse signal (non-zero entries follow standard uniform distribution) and noiseless measurements – reconstruction performance for fixed signal length** n: As in Fig 3.14, the y-axis denotes the number of measurements m, and the x-axis denotes the sparsity k, for fixed signal length $n = 1000$. Even if the non-zero values are set to be standard uniform random variables, the performance of reconstruction is similar to the case that non-zero values are set to be 1.

work for many ensembles of sparse measurement matrices.

□ **End of chapter.**

Chapter 4

Network Tomography – FRANTIC

4.1 Introduction

Monitoring performance characteristics of individual links is important for diagnosing and optimizing network performance. Making direct measurements for each link, however, is impractical in large-scale networks because (i) nodes inside the networks may not be available to carry out measurements due to physical or protocol constraints, and (ii) measuring each link *separately* incurs excessive control-traffic overhead and is not scalable.

A viable alternative approach is network tomography [138]. It aims to infer the performance characteristics of internal links by *path measurements* between controllable nodes, where a path measurement is function of the characteristics of the links on the path. It does not require access to all the nodes and, more importantly, it allows clever solutions to leverage the network structure (*e.g.*, topology and graph properties) to *jointly* infer the performance characteristics of multiple links via path measurements. Many existing works have explored such insights to design excellent solutions that

are able to infer the congested links with far fewer measurements than the direct link measurement approach [21,37,87,107,153]. See [35] for a survey.

Recently, Xu *et al.* [146] further argue that usually only a small fraction of network links, say k out of total $|\mathcal{E}|$ links ($k \ll |\mathcal{E}|$), are congested (*i.e.*, experiencing large congestion delay or high packet loss rate). They interpret each path measurement as a linear combination of the delays or loss rates of the k congested links. With this understanding, the problem of network tomography can be viewed as recovering a k-sparse link vector from a set of linear measurements.

Exploiting the "sparse congestion structure" insight, Xu *et al.* [146] propose a compressive sensing based scheme that can identify any k congested links using $\mathcal{O}(T_{\mathcal{N}} k \log(|\mathcal{E}|))$ path measurements over any sufficiently-connected graph. Here, each of the path measurements is a random walk on the graph, and $T_{\mathcal{N}}$ is the mixing time of the random walk. Further, they show that one can actually *recover* the performance characteristics of any k congested links with again $\mathcal{O}(T_{\mathcal{N}} k \log(|\mathcal{E}|))$ path measurements using an ℓ_1-minimization decoder. Similar results are also obtained by [39, 62, 141]. Given all these exciting results, a natural question is whether one can do better, and if so, how?

4.1.1 Our contribution

Summary

In this work, we build upon our recently developed compressive sensing algorithm named SHO-FA [10] to design a new network tomography solution that we call FRANTIC . FRANTIC achieves the following performance:

- FRANTIC can identify any k congested links (or nodes) out of n and recover the corresponding link (or node) performance characteristics

using $\mathcal{O}(\rho k \log(n)/\log(M))$ path measurements with a high probability. Here, $M \in \mathbb{N}$ and $\rho \in \Omega(1) \cap o(n/k)$ are design parameters. See Section 4.3.2 for a discussion.

- The FRANTIC decoding algorithm can recover the link (or node) performance characteristics in $\mathcal{O}(\rho k \log(n)/\log(M))$ steps.

As compared to the solution in [146], our solution improves both the number of measurements and the number of recovery steps from $\mathcal{O}(T_{\mathcal{N}} k \log(n))$ to $\mathcal{O}(k)$ (obtainable by setting $M = \mathcal{O}(n)$).

Techniques and results

The main techniques that lead to these improvements are as follows. First, in Section 4.4, we develop an efficient compressive sensing algorithm SHO-FA-INT when the entries of the measurement matrix are constrained to be positive integers. Our algorithm borrows key ideas from a prior work [10] that studies compressive sensing in the unconstrained setting. A key technique here is to group together measurements and choose the "weights" of the measurement matrix as co-prime vectors. A set of vectors are called co-prime if any vector is not a multiple of another vector in the set. This ensures that each network link has a distinct signature in the measurement output, which allows us to decode the delay values for congested links in an iterative manner. Theorem 7 states the performance guarantees of our algorithm. Next, we propose a design for measuring the delay on congested links in a network in Section 4.5.1. An important insight in our design is that by using local loops at individual edges, end-to-end delay measurements can be designed to assign different integer weights to delays for different edges. We start with a compressive sensing matrix given by SHO-FA-INT and emulate the output of the matrix by first designing two

correlated network paths, and then cancelling out the contribution of unwanted links by subtracting one from another. Theorems 8-10 state the performance guarantees of the FRANTIC algorithms. We also note that the path lengths required for FRANTIC can be suitably optimized by using Steiner Trees and network decomposition. Theorem 10 and subsequent discussions point this out.

Reference	Type	# Measurements	Decoding Complexity	Path Length	Network Topology
[141]	Node	$R\mathcal{O}(k\log(\lvert\mathcal{V}\rvert/k))+R+1$	CS with $0-1$ matrix	–	General graph, R is the radius of the graph
	Node	$\mathcal{O}(rk\log(\lvert\mathcal{V}\rvert/k))+r$	CS with $0-1$ matrix	–	If G has an r-partition
	Node	$\mathcal{O}(2k\log(\lvert\mathcal{V}\rvert/2k))+r$	CS with $0-1$ matrix	–	Erdos-Renyi random graph $G(\lvert\mathcal{V}\rvert,p)$, with $p=\beta\log(\lvert\mathcal{V}\rvert)/\lvert\mathcal{V}\rvert$ and $\beta\geq 2$
[146]	Edge	$\mathcal{O}(T_{\mathcal{N}}k\log(\lvert\mathcal{E}\rvert))$	l_1 minimization	$\mathcal{O}(\lvert E\rvert/k)$	G is a (D,c)-uniform graph, $D\geq D_0=\mathcal{O}(c^2kT_{\mathcal{N}}^2)$.
[62]	Edge	$\mathcal{O}(k\log(\lvert\mathcal{E}\rvert/k))$	l_1 minimization	–	Network is 1-identifiable
[39]	Node	$\mathcal{O}\left(c^4k^2T_{\mathcal{N}}^2\log(\lvert\mathcal{V}\rvert/d)\right)$	Disjunct matrix	$\mathcal{O}(\lvert\mathcal{V}\rvert/(c^3kT_{\mathcal{N}}))$	G is a (D,c)-uniform graph. $D\geq D_0=\mathcal{O}(c^2kT_{\mathcal{N}}^2)$.
	Edge	$\mathcal{O}(c^4k^2T_{\mathcal{N}}^2\log(\lvert\mathcal{E}\rvert/d))$	Disjunct matrix	$\mathcal{O}(\lvert\mathcal{V}\rvert D/(c^3kT_{\mathcal{N}}))$	
	Node	$\mathcal{O}(c^8k^3T_{\mathcal{N}}^4\log(\lvert\mathcal{V}\rvert/d))$	Disjunct matrix	unbounded (sink node)	
	Edge	$\mathcal{O}(c^9k^3DT_{\mathcal{N}}^4\log(\lvert\mathcal{E}\rvert/d))$	Disjunct matrix	unbounded (sink node)	
	Node	$\mathcal{O}(k^2(\log^3(\lvert\mathcal{V}\rvert)))/(1-p)^2$	Disjunct matrix	$\mathcal{O}(\lvert\mathcal{V}\rvert/(c^3kT_{\mathcal{N}}))$	G is D-regular expander graph or Erdos-Renyi random graph, $G(\lvert\mathcal{V}\rvert,D/\lvert\mathcal{V}\rvert)$, with $D\geq D_0=\Omega(k\log^2(\lvert\mathcal{V}\rvert))$.
	Edge	$\mathcal{O}(k^2(\log^3(\lvert\mathcal{E}\rvert)))/(1-p)^2$	Disjunct matrix	$\mathcal{O}(\lvert\mathcal{V}\rvert D/(c^3kT_{\mathcal{N}}))$	
	Node	$\mathcal{O}(k^3(\log^5(\lvert\mathcal{V}\rvert)))/(1-p)^2$	Disjunct matrix	unbounded (sink node)	
	Edge	$\mathcal{O}(k^3D(\log^5(\lvert\mathcal{E}\rvert)))/(1-p)^2$	Disjunct matrix	unbounded (sink node)	
This paper	Node	$\mathcal{O}(k\log(\lvert\mathcal{V}\rvert)/\log(M))$	$\mathcal{O}(k\log(\lvert\mathcal{V}\rvert)/\log(M))$	$\mathcal{O}(D\lvert\mathcal{V}\rvert/k)$	General Graph D is the diameter of the graph
	Edge	$\mathcal{O}(k\log(\lvert\mathcal{E}\rvert)/\log(M))$	$\mathcal{O}(k\log(\lvert\mathcal{E}\rvert)/\log(M))$	$\mathcal{O}(D\lvert\mathcal{E}\rvert/k)$	

Partial literature review of reference table: The work of [141] considers the node delay estimation problem where a set of nodes can be measured together in one measurement if and only if the induced sub-

graph is connected and each measurement is an additive sum of values at the corresponding nodes. The generated sensing matrix is a $0-1$ matrix, therefore the decoding complexity mainly depends on which binary compressive sensing algorithm is used. General graphs and also some special graphs are studied. The idea of a binary compressive sensing algorithm is borrowed by [62], in which a single edge delay estimation problem is studied, and estimation is done using l_1 minimization. In [146], a random-walk based approach is proposed to solve the k-edge delay estimation problem. $T_{\mathcal{N}}$ is the $\frac{1}{(2c|\mathcal{V}|)^2}$-mixing time of $\mathcal{N}$. Networks with assumptions of bounded node degrees are studied. Similar to [146], [39] uses random-walk measurements to solve both node and edge failure localization problem where group testing (non-linear version of compressive sensing) algorithms are used. The goal is to generate disjunct matrices which are suitable for group testing. The starting points of measurements can be chosen within a fixed set of designated vertices, or, chosen randomly among all vertices of the graph. Separately, the problem of edge failure localization has also been studied in the optical networking literature [2, 73, 147]. In [73], which considers the problem of single edge failure localization, which has the same flavor as [141]. Binary-search type algorithms are proposed for some special graphs. For general graphs, the upper bound on the number of measurements required for single edge failure localization is $\mathcal{O}(D(\mathcal{N})+\log^2(|\mathcal{V}|))$ where $D(\mathcal{N})$ is the diameter of the graph. In [147], the problem of multi-link failure localization is considered. For small networks, a tree-decomposition based method has the upper bound on the number of trials is $\min(\mathcal{O}(D(\mathcal{N})\log(|\mathcal{V}|)), \mathcal{O}(D(\mathcal{N}) + \log^2(|\mathcal{V}|)))$. For large-scale networks, a random-walk based method similar to [39] is proposed. They also consider "practical" constraints such as the number of failed links cannot be known beforehand. In [2], the solution proposed

requires $(k+2)$-edge-connected networks so as to guarantee k-link-failure localization.

4.2 Model and problem formulation

Network and delay model: Let $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ be a undirected network with node set $\mathcal{V}$ and link set $\mathcal{E}$. In this work, we consider the reference-based tomography problem where the topology of $\mathcal{N}$ is known. We assume that $\mathcal{N}$ is connected. We say that a node $v \in \mathcal{V}$ has delay x_v if every packet that passes through v is delayed by x_v. Similarly, a link $e \in \mathcal{E}$ has delay x_e if every packet passing through e in any direction is delayed by x_e. We say a node or link is *congested* if the delay associated with it is non-zero. A congested node is called *isolated* if there exists one of its neighbours which is not congested. Let $\mathbf{x}_{\mathcal{V}} = (x_v : v \in \mathcal{V})$ and $\mathbf{x}_{\mathcal{E}} = (x_e : e \in \mathcal{E})$. Both $\mathbf{x}_{\mathcal{V}}$ and $\mathbf{x}_{\mathcal{E}}$ are unknown but have at most k non-zero coordinates.

Measurement model: Each measurement is performed by sending test packets over pre-determined paths[1] and measuring the end-to-end time taken for its transmission. Some nodes (resp. links) may be visited more than once in a given path. As a result, each measurement output y_i, $i = 1, 2, \ldots, m$, is a weighted sum of delays involving nodes and links that lie in the measurement path, where the weight of a given node or link is the number of times it is visited by the measurement path. In this work, we consider two kinds of measurements – *node measurements* and *link measurements*. In the node (resp. link) measurements, we associate each node (resp. link) with a real-valued delay and the objective is to reconstruct the node (resp. link) delay vector $\mathbf{x}_{\mathcal{V}}$ (resp. $\mathbf{x}_{\mathcal{E}}$) given the collection of measurement outputs.

[1] In present-day networks, this may be accomplished by employing source-based routing (c.f. [84]) for test packets.

1. *Node measurements:* In the node measurement model, we associate each node with a real-valued delay (see [141], for example). Let $\mathcal{S} \subseteq \mathcal{V}$ denote a subset of nodes in $\mathcal{N}$. Let $\mathcal{E}_{\mathcal{S}}$ denote the subset of links with both ends in $\mathcal{S}$, then $\mathcal{N}_{\mathcal{S}} = (\mathcal{S}, \mathcal{E}_{\mathcal{S}})$ is the induced subgraph of $\mathcal{N}$. A set $\mathcal{S}$ of nodes can be measured together in one measurement if and only if $\mathcal{N}_{\mathcal{S}}$ is connected.
2. *Link measurements:* In the link measurement setup, we associate each link with a real-valued delay. Let $\mathcal{T} \subseteq \mathcal{E}$ denote a subset of links in $\mathcal{N}$. A set $\mathcal{T}$ of links can be measured together in one measurement if and only if there exists a path that traverses each link in $\mathcal{T}$.

For each of these models, we express the measurement output as a vector $\mathbf{y} \in \mathbb{R}^m$ that is related to the delay vector through a measurement matrix A through matrix multiplication.

n	Total number of links (or nodes) in the network
k	Number of congested links (or nodes) in the network
M	The maximum number of times a test packet may travel over any edge
D	The diameter of $\mathcal{N}$
$T_{\mathcal{N}}$	The mixing time of the random walk over graph $\mathcal{N}$
ρ	A design parameter that controls the tradeoff between the path lengths and the number of the measurements
$\mathcal{N}$	$\mathcal{N} = (\mathcal{V}, \mathcal{E})$, an undirected network with node set $\mathcal{V}$ and link set $\mathcal{E}$
x_v	Time taken by a test packet to pass through node $v \in \mathcal{V}$
$\mathbf{x}_{\mathcal{V}}$	Node delay vector of length $\lvert\mathcal{V}\rvert$
x_e	Time taken by a test packet to pass through link $e \in \mathcal{E}$ in any direction
$\mathbf{x}_{\mathcal{E}}$	Link delay vector of length $\lvert\mathcal{E}\rvert$

Table 4.1: Table of notation for network parameters

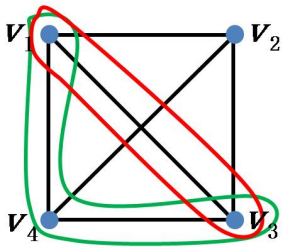
R	$R \in \mathbb{N}^+$ such that $M^R/\zeta(R) \geq 3n$ where $\zeta(\cdot)$ be the Riemann zeta function
$\mathbf{y}$	Measurement output of length $m = R\mu$
A	Measurement matrix of dimension $R\mu \times n$
$a_{ij}^{(r)}$	The r-th row entry in the j-th column of the i-th group of A for $r = 1, 2, \ldots, R$, $i = 1, 2, \ldots, \mu$ and $j = 1, 2, \ldots, n$
$\mathcal{G}_{n,m'}$	A bipartite graph with left vertex set $\{1, 2, \ldots, n\}$ and right vertex set $\{1, 2, \ldots m'\}$
$N(\mathcal{S})$	The set of right neighbours of a subset of left nodes $\mathcal{S}$ in $\mathcal{G}_{n,\mu}$
$\boldsymbol{P}$	A path of length T over the network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$, *i.e.*, a sequence $(e_1, e_2, \ldots, e_T)$ of links from $\mathcal{E}$
$W(\boldsymbol{P}, e)$	The multiplicity of a link $e \in \mathcal{E}$ given a path $\boldsymbol{P}$, *i.e.*, the number of times $\boldsymbol{P}$ visits e
$\Delta(\boldsymbol{P})$	The end-to-end delay for a path $\boldsymbol{P}$

Table 4.2: Table of notation for Design Variables

4.3 High-level Intuition and Main Results

4.3.1 Key ideas

In this section, we present some key observations and challenges that this work focuses on. We begin with the observation that there is a high-level connection between the compressive sensing and the network tomography problem. As noted in the previous section, network tomography can be treated as a problem of solving a system of linear equations. Under the assumption that the underlying unknown vector is sparse, it is natural to think of it as a compressive sensing problem [31, 33, 51, 67, 136]. Building on this intuition, network tomography can be formulated as the following compressive sensing problem: i) design a matrix A, ii) obtain delay measurements $\mathbf{y} = A\mathbf{x}_{\mathcal{V}}$, and iii) reconstruct $\mathbf{x}_{\mathcal{V}}$ from $\mathbf{y}$. Fig. 4.1 illustrates this connection. Since each subset of nodes in a complete graph induces a connected subgraph, we can freely choose the locations of non-zero entries in each row of A. Then, any compressive sensing algorithm with a 0-1 measurement matrix [15, 145] can be applied to recover the vector $\mathbf{x}_{\mathcal{V}}$. However, when we go beyond complete graphs and node measurements, it



$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \mathbf{x}_{\mathcal{V}}$$

Figure 4.1: Node Delay Estimation: For a complete graph with four vertices. We can get any measurements we want since each subgraph of a complete graph is connected. For example, the subgraphs induced by $\{v_1, v_3\}$ (covered by red cycle) and $\{v_1, v_3, v_4\}$ (covered by green cycle) are connected, therefore we get the measurements $[1\ 0\ 1\ 0]\mathbf{x}_{\mathcal{V}}$ and $[1\ 0\ 1\ 1]\mathbf{x}_{\mathcal{V}}$, respectively.

is not straightforward to apply compressive sensing directly. The network topology may impose constraints on implementable measurement matrices (See Figs. 4.2, 4.3, 4.5, and 4.6).

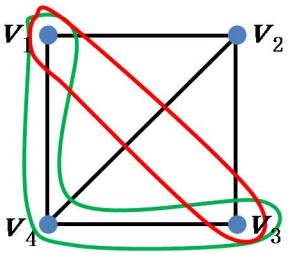


Figure 4.2: General Networks: If the link (v_1, v_3) is removed from the original complete graph, we cannot get the measurement $[1\ 0\ 1\ 0]\mathbf{x}_{\mathcal{V}}$ any longer.

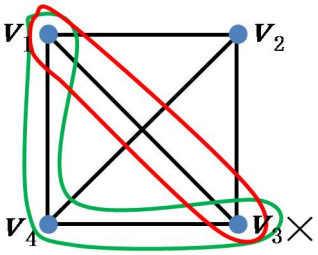


Figure 4.3: Inaccessible Nodes: If there is some constraint such that we can not access to v_3 directly, then the measurements in Fig. 4.1 are not implementable.

Xu *et al.* [146] get around some of these problems by using random walks. One drawback of their approach is that it involves a factor of mixing time $T_{\mathcal{N}}$ for both the number of measurements and the path length. For networks without sufficient connectivity, the mixing time may be very large,

e.g., for cycle graph, $T_{\mathcal{N}} = \Omega(|\mathcal{E}|^2)$. In the following, we propose two news ideas that enable us to circumvent the above problem.

Idea 1: *Cancellation enables selecting disconnected subsets of links and nodes.* The idea here is similar to that used in [141] where they use structures called "hubs" to get arbitrary measurement matrix. However, they only consider the node delay model, and special graphs which have r-partitions. In this work, we expand this approach to both link delay and node delay models. By considering correlated measurements, we can cancel out the contribution of undesired links and nodes in a given measurement. Using this approach, we can mimic arbitrary measurements on general graphs. See Fig. 4.4 as an illustration. One drawback of the cancellation based approach is that if the selected measurement has too many disjoint components, then the number of measurements required is very large. In Fig. 4.4, the number of cancellations is 2.

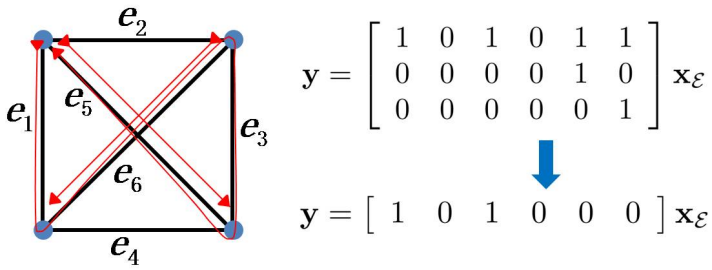


Figure 4.4: Cancellation: There are three paths in this graph:$\{e_1e_6e_3e_5\}$, $\{e_5\}$ and $\{e_6\}$. Triangles indicate the source and destination of a path. Correspondingly, we can derive three measurements $[1\ 0\ 1\ 0\ 1\ 1]\mathbf{x}_{\mathcal{E}}$, $[0\ 0\ 0\ 0\ 1\ 0]\mathbf{x}_{\mathcal{E}}$, and $[0\ 0\ 0\ 0\ 0\ 1]\mathbf{x}_{\mathcal{E}}$. Subtracting the second and the third measurements from the first measurement, we get the measurement $[1\ 0\ 1\ 0\ 0\ 0]\mathbf{x}_{\mathcal{E}}$ which cannot be obtained by just one path.

Idea 2: *Weighted measurements reduce the number of cancellations required and allow us to implement arbitrary integer valued matrices.* The

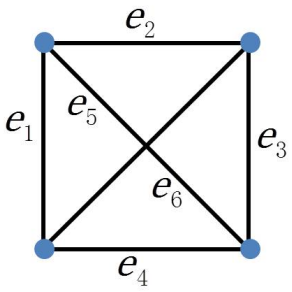


Figure 4.5: Edge Delay Estimation: We know that we can not get arbitrary measurements by one path even if the graph is complete. (*e.g.*, the measurement [0 0 0 0 1 1]$\mathbf{x}_{\mathcal{E}}$ cannot be obtained since there is no path just visiting e_5 and e_6.)

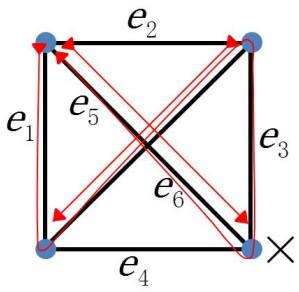


Figure 4.6: Inaccessible Nodes: The second measurement in Fig. 4.4 cannot be obtained since the v_3 is not accessible.

insight here is that if we have two paths along the same set of links, we can assign different weights for each link (or node) on these paths by performing local loops. Specifically, for a given set of weights on a subset of links (or nodes), we construct two measurements - a spanning measurement, and a weighted measurement. The spanning measurement is constructed by finding any path that visits through all the links (or nodes) in the desired subset at least once. The weighted measurement, then follows the same set of edges as the spanning measurement, but visits each link (or node) an additional number of times in accordance with the desired weight for that link (or node). Finally, we subtract the end-to-end delay for the weighted path from that of the spanning path to get an output that is proportional to the output of the corresponding compressive sensing problem (See Fig. 4.7). These ideas enable us to reduce the network tomography problem to a compressive sensing problem with integer valued matrices. In the Section 4.4, we present an efficient compressive sensing algorithm with integer

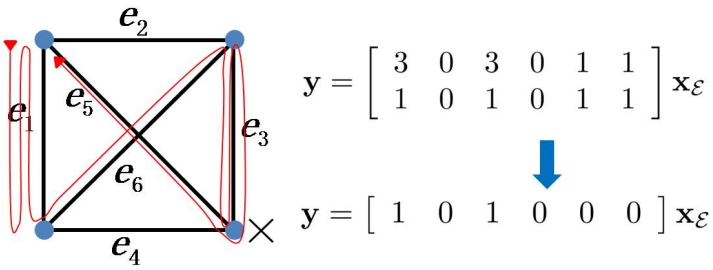


Figure 4.7: Cancellation using weighted measurements: To get the measurement [1 0 1 0 0 0]$\mathbf{x}_{\mathcal{E}}$, we design the paths as follows. First, we just follow the path $\{e_1e_6e_3e_5\}$, we get the measurement [1 0 1 0 1 1]$\mathbf{x}_{\mathcal{E}}$. Second, when visiting e_1 and e_3 for the first time, the probe does one more local loop for both links to get the measurement [3 0 3 0 1 1]$\mathbf{x}_{\mathcal{E}}$. Finally, we take the difference of these two measurements and divide the result by 2. Note that 1) Only one cancellation is required. 2) Even if v_3 is inaccessible, we still can obtain the two target measurements. 3) One additional local loop at e_1 in the second step (so that e_1 is visited 5 times), allows us to obtain the measurement [2 0 1 0 0 0]$\mathbf{x}_{\mathcal{E}}$. Thus, controlling the number of local loops allows us to implement other ensembles of measurement matrices.

entries.

4.3.2 Main Theorems

In this section, we state the main results of this work. Let $\rho \in \Omega(1) \cap o(|\mathcal{E}|/k)$ be a design parameter.

Theorem 7 (Compressive sensing via integer matrices). *Let $M \in \mathbb{Z}^+$. There exists a constant c such that whenever $m > ck\lceil \log(n)/\log(M) \rceil$, the ensemble of $\mathbb{Z}_M$-valued matrices $\{A_{m\times n}\}$ designed in Section 4.4 and the SHO-FA-INT reconstruction algorithm has the following properties:*

i) Given $(A_{m\times n}, A_{m\times n}\mathbf{x})$ as input, where $\mathbf{x}$ is an arbitrary k-sparse vector in $\mathbb{R}^n$, $SHO-FA-INT$ outputs a vector $\hat{\mathbf{x}} \in \mathbb{R}^n$ that equals $\mathbf{x}$ with probability at least $1-\mathcal{O}(1/k)$ under the distribution of $A_{m\times n}$ over the ensemble $\{A_{m\times n}\}$,

ii) Given $A_{m\times n}\mathbf{x}$, $\hat{\mathbf{x}}$ is reconstructed in $\mathcal{O}(k\lceil \log(n)/\log(M) \rceil)$ arithmetic operations, and

iii) Each row of $A_{m\times n}$ has $\mathcal{O}(n/k)$ non-zeros in expectation.

Theorem 8 (Network tomography for link congestion). *Let $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ be an undirected network of diameter D such that at most k links have unknown non-zero link delays. Let $M \in \mathbb{Z}^+$ Then, the FRANTIC algorithm has the following properties:*

i) FRANTIC requires $\mathcal{O}(\rho k\lceil \log(|\mathcal{E}|)/\log(M) \rceil)$ measurements,

ii) For every edge delay vector $\mathbf{x}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|}$,FRANTIC outputs $\hat{\mathbf{x}}_{\mathcal{E}}$ that equals $\mathbf{x}_{\mathcal{E}}$ with probability $1-\mathcal{O}(1/\rho k)$,

iii) The FRANTIC reconstruction algorithm requires $\mathcal{O}(\rho k\lceil \log(|\mathcal{E}|)/\log(M) \rceil)$ arithmetic operations, and

iv) The number of links of $\mathcal{N}$ traversed by each test measurement packet in FRANTIC is $\mathcal{O}(D|\mathcal{E}|/\rho k)$ and the total number of hops for each packet is $\mathcal{O}(DM|\mathcal{E}|/\rho k)$.

Definition 1 (Isolated congested node)**.** *A congested node is called isolated if there exists at least one neighbour which is not congested.*

Theorem 9 (Network tomography for node congestion)**.** *Let $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ be an undirected network of diameter D such that at most k have unknown non-zero node delays and all congested nodes are isolated. Let $M \in \mathbb{Z}^+$ Then, the FRANTIC algorithm has the following properties:*

i) FRANTIC requires $\mathcal{O}(\rho k \lceil \log(|\mathcal{V}|)/\log(M) \rceil)$ measurements,

ii) For every edge delay vector $\mathbf{x}_\mathcal{V} \in \mathbb{R}^{|\mathcal{V}|}$, FRANTIC outputs $\hat{\mathbf{x}}_\mathcal{V}$ that equals $\mathbf{x}_\mathcal{V}$ with probability $1 - \mathcal{O}(1/\rho k)$,

iii) The FRANTIC reconstruction algorithm requires $\mathcal{O}(\rho k \lceil \log(|\mathcal{V}|)/\log(M) \rceil)$ arithmetic operations, and

iv) The number of links of $\mathcal{N}$ traversed by each test measurement packet in FRANTIC is $\mathcal{O}(D|\mathcal{V}|/\rho k)$ and the total number of hops for each packet is $\mathcal{O}(DM|\mathcal{V}|/\rho k)$.

Explanation of design parameters

The parameter M denotes the maximum number of times a test packet may travel over any edge. In many present-day networks, the value of M is usually fixed to be a small constant. In this setting, our algorithm requires $\mathcal{O}(k \log(n))$ measurements and decoding steps. Additionally, if M is allowed to increase with the network size (possibly, in future generation networks), the number of measurements and decoding complexity our algorithms may be decreased to $\mathcal{O}(k)$.

The parameter ρ is a design parameter that controls the tradeoff between the measurement path lengths and the number of measurements required. When $\rho = 1$, we require $\mathcal{O}(k \log(n)/\log(M))$ measurements with path lengths $\mathcal{O}(nD/k)$. On the other extreme, if ρ is set to be $n/(k\omega(1))$, we require up to $o(n)$ measurements but with as little as $\omega(D)$ path-length. In our exposition, we prove the correctness of our schemes for the case when $\rho = 1$. The results for other values of ρ follow from this analysis by pretending that the network has ρk congested nodes instead of k.

4.4 SHO-FA-INT algorithm for Compressive Sensing

We begin by describing a new compressive sensing algorithm SHO-FA-INT which has several properties that are desirable for our application.

In the measurement designs presented in Sections 3.2 and 3.3, a key requirement is that the entries of the measurement matrix may be chosen to be arbitrary real or complex numbers of magnitude (up to $\mathcal{O}(\log(n))$ bits of precision). However, in several scenarios of interest, the entries of the measurement matrix are constrained by the measurement process.

- In network tomography [146], one attempts to infer individual link delays by sending test packets along pre-assigned paths. In this case, the overall end-to-end path delay is the sum of the individual path delays along that path, corresponding to a measurement matrix with only 0s and 1s (or in general, with positive integers, if loops are allowed).
- If transmitters in a wireless setting are constrained to transmit symbols from a fixed constellation, then the entries of the measurement matrix can only be chosen from a finite ensemble.

In both these examples, the entries of the measurement matrix are tightly constrained.

In this section, we discuss how key ideas from Sections 3.2 and 3.3 may be applied in compressive sensing problems where the entries of measurement matrix is constrained to take values form a discrete set. For simplicity, we assume that the entries of the matrix A can take values in the set $\{1, 2, \ldots, M\}$ for some integer $M \in \mathbb{N}^+$. For simplicity, we consider only the exact k-sparse problem, noting that extensions to the approximate k-sparse case follow from techniques similar to those used in Section 3.3.

Let $\{\mathcal{G}_{n,m'}\}_{n,m' \in \mathbb{N}}$ be an ensemble of left-regular bipartite graphs, where each $\mathcal{G}_{n,m'}$ is a bipartite graph with left vertex set $\{1, 2, \ldots, n\}$ and right vertex set $\{1, 2, \ldots, m'\}$. For each left vertex $j \in \{1, 2, \ldots, n\}$, we pick three distinct vertices uniformly at random from the set of right vertices $\{1, 2, \ldots, m'\}$.

Measurement Design: Let $\zeta(\cdot)$ be the Riemann zeta function. Let $R \in \mathbb{N}^+$ such that $M^R/\zeta(R) \geq 3n$ and let $[M]$ denote the set $\{1, 2, \ldots, M\}$. Given the graph $\mathcal{G}_{n,m'}$, we design a $Rm' \times n$ measurement matrix $A(= A_{Rm' \times n})$ as follows. First, we partition the rows of A into m' groups of rows, each consisting of R consecutive rows as follows.

$$A = \left[\begin{array}{c:c:c:c} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{11}^{(R)} & a_{12}^{(R)} & \ldots & a_{2n}^{(R)} \\ \hdashline a_{21}^{(1)} & a_{22}^{(1)} & \ldots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{21}^{(R)} & a_{22}^{(R)} & \ldots & a_{2n}^{(R)} \\ \hdashline \vdots & \vdots & \ddots & \vdots \end{array}\right]$$

Let $a_{ij}^{(r)}$ be the r-th row entry in the j-th column of the i-th group and let $\boldsymbol{a}_{ij} = [a_{ij}^{(1)} a_{ij}^{(2)} \ldots a_{ij}^{(R)}]^T$. First, for each (i, j) not in $\mathcal{G}_{n,m'}$, we set $\boldsymbol{a}_{ij} = 0^R$. Next, we randomly choose $3n$ distinct values from the set

$$\mathcal{C} \triangleq \left\{[c_1, c_2, \ldots, c_R]^T \in (\mathbb{Z}_M)^R : \gcd(c_1, c_2, \ldots, c_R) = 1\right\}$$

and use these to set the values of $\boldsymbol{a}_{ij}$ for each edge (i, j) in $\mathcal{G}_{n,m'}$. Note that vectors in the set $\mathcal{C}$ are co-prime. The assumption that $M^R/\zeta(R) \geq 3n$ ensures that such a sampling is possible. We skip the proof of Lemma 6 here and refer the reader to [10] for the proof.

Lemma 6. *For M large enough, $M^R/\zeta(R) \leq |\mathcal{C}| \leq M^R$.*

Proof. The upper bound on $|\mathcal{C}|$ is trivial, since each element of $\mathcal{C}$ is an R length vector whose each coordinate takes values from the set $[M]$. To prove the lower bound, note that a sufficient condition for $\gcd(c_1, c_2, \ldots, c_R)$ to be true is that for each prime number $p \in \mathbb{N}$, there exists at least one index $r \in [R]$ such that p does not divide c_r. Since the number of vectors $(c_1, c_2, \ldots, c_R) \in [M]^R$ such that for each r, c_r is divisible by p is at most $(M/p)^R$, the number of vectors in $[M]^R$ such that at least one component is not divisible by p is at least $M^R(1 - p^{-R})$. Denoting the set of prime numbers by $\mathbb{P}$ and extending the above argument to exclude all vectors that are divisible by some prime number greater than or equal to two, we obtain, for M large enough,

$$|\mathcal{C}| \geq M^R \prod_{p:p\in\mathbb{P}} (1 - 1/p^{-R}) = M^R/\zeta(R).$$

In the above, the second equality follows from Euler's product formula for the Riemann zeta function. ■

The output of the measurement is a Rm'-length vector $\mathbf{y} = A\mathbf{x}$. Again, we partition $\mathbf{y}$ into m' groups of R consecutive rows each, and denote the i-th sub-vector as $\mathbf{y}_i$. Note each $\mathbf{y}_i \in \mathbb{R}^R$ follows the relation $\mathbf{y}_i = [a_{i1}a_{i2}\ldots a_{in}]\mathbf{x}$.

Reconstruction algorithm: The decoding process is essentially equivalent to the "peeling process" to find 2-cores of uniform hypergraphs [70, 102]. The decoding takes place over $\mathcal{O}(k)$ iterations. The decoding algorithm is very similar to that of SHO-FA. In each iteration, we find one non-zero

undecoded x_j with a constant probability after locating a right node that is connected to exactly one non-zero left node. After decoding the non-zero x_j for the current iteration, we cancel out the contribution of x_j from all measurements and proceed iteratively. To describe the peeling process, we define leaf nodes as follows.

Definition 2 ($\mathcal{S}$-leaf node)**.** *A right node i is a leaf node for set of left nodes $\mathcal{S}$ if i is connected to exactly one left node $j \in \mathcal{S}$ in the graph $\mathcal{G}_{n,m'}$.*

First, the algorithm initializes the *reconstruction vector* $\hat{\mathbf{x}}(1)$ to the all zeros vector 0^n, the *residual measurement vector* $\tilde{\mathbf{y}}(1)$ to $\mathbf{y}$, and the *neighbourly set* $\mathcal{B}(1)$ to be the set of all right nodes for which y_i does not equal 0^R. In each iteration t, the decoder picks a right node $i(t)$ from the current neighbourly set $\mathcal{B}(t)$ and checks if only one left node contributes to the value of $(\tilde{y}(t))_{i(t)}$. If so, it identifies $i(t)$ as a *leaf* node, decodes delay value at the corresponding parent node, and updates $\mathcal{B}(t+1), \tilde{\mathbf{y}}(t+1)$, and $\hat{\mathbf{x}}(t+1)$ for the next iteration. The decoder terminates when the residual measurement vector becomes zero.

Next, we prove the performance guarantees of SHO-FA-INT as claimed in Theorem 7. Let $k = k(n)$ grow as a function of n. We show that the algorithm presented above correctly reconstructs the vector $\hat{\mathbf{x}}$ with a high probability over the ensemble of matrices $\{A_{Rm' \times n}\}$. To this end, we first note that if $m' = \Omega(k)$, the ensemble of graphs $\{\mathcal{G}_{n,m'}\}$ satisfies the following "many leaf nodes" as shown in the following lemma.

Lemma 7 (Many leaf nodes)**.** *Let $\mathcal{S}$ be a subset of the left nodes of the $\mathcal{G}_{n,m'}$ and let $N(\mathcal{S}')$ be the set of right neighbours of a set S'. If $|\mathcal{S}| \leq k$ then with probability $1 - \mathcal{O}(1/k)$, for every $\mathcal{S}' \subseteq \mathcal{S}$, $N(\mathcal{S}')$ contains at least $|N(\mathcal{S}')|/2$ $\mathcal{S}'$-leaf nodes.*

Proof of Theorem 7: Let $\mathcal{S}(\mathbf{x}) = \{j \in \{1, 2, \ldots, n\} : x_j \neq 0\}$. By Lemma 7, with probability $(1 - \mathcal{O}(1/k))$, all its subsets $\mathcal{S}'$ of $\mathcal{S}(\mathbf{x})$ have at least twice as many leaf neighbours as the the number of elements in the $\mathcal{S}'$. Therefore, in each iteration, the probability of picking a leaf node is at least half. Next, we note that in each iteration that we pick a leaf node, the probability of identifying as one and finding its left neighbour correctly is 1. This is true because the weight vectors $\boldsymbol{a}_{ij}$'s corresponding different neighbours of a given right node i are different, and for a leaf node i with the sole non-zero neighbour j, the output value $\mathbf{y}_i$ exactly equals $x_j \boldsymbol{a}_{ij}$.

Next, we argue that if i is not a leaf node, then the probability of it being declared a leaf node in any iteration is $\mathcal{O}(1/n)$. Note that for this error event to occur for a right node i, it must be true that $\sum_{j' \in N(i)} x_{e'} a_{ij'} = x'' a_{ij''}$ for some $x'' \in \mathbb{R}$ and j'' connected to i. Since all the measurement weights are chosen randomly, by the Schwartz-Zippel lemma [127, 154], the probability of this event is $\mathcal{O}(1/n)$, which is $o(1/k)$.

Since the probability of picking a leaf node at any iteration is at least $1/2$, the expected number of iterations before a new leaf node is picked is upper bounded by 2. Since there are at most k non-zero x_j's, in expectation, the algorithm terminates in $\mathcal{O}(k)$ steps. Further, since the probability of finding a leaf in each iteration is independent across iterations, by applying standard concentration arguments, the total number of iterations required is upper bounded by $2k$ in probability.

Finally, to compute the decoding complexity, note that each iteration requires a constant number of arithmetic operations over vectors in $[M]^R$, which in turn can be decomposed into $\mathcal{O}(R)$ arithmetic operations over integers. Therefore, the total number of integer operations required is $\mathcal{O}(Rk) = \mathcal{O}(k\lceil \log(n)/\log(M) \rceil)$. Finally, we note that since each left node in $\mathcal{G}_{n,m'}$ has exactly 3 right neighbours which are picked uniformly among

all right nodes and independently across different left nodes, with high probability, each right node has no more than $4n/m'$ left neighbours. This can be proved by first computing the expected number of left neighbours for a right node and then applying the Chernoff bound to concentrate it to close to its expectation. This shows that, with high probability, the number of non-zero entries in each row of A is $\mathcal{O}(n/k)$. ■

4.5 The FRANTIC algorithm

4.5.1 Link Delay Estimation

We define a *path* $\boldsymbol{P}$ of length T over the network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ as a sequence $(e_1, e_2, \ldots, e_T) = ((v_1, v_2), (v_2, v_3), \ldots (v_T, v_{T+1}))$ such that $e_t \in \mathcal{E}$ for $t = 1, 2, \ldots, T$. For a given path $\boldsymbol{P}$, we define the multiplicity $W(\boldsymbol{P}, e)$ of a link $e \in \mathcal{E}$ as the number of times $\boldsymbol{P}$ visits e. Let $\Delta(\boldsymbol{P})$ be the end-to-end delay for a path $\boldsymbol{P}$.

Definition 3 ($\boldsymbol{w}$-spanning measurement)**.** *Given a measurement weight vector* $\boldsymbol{w} = [w_1 w_2 \ldots w_{|\mathcal{E}|}]$*, a* $\boldsymbol{w}$-spanning measurement *is a path* $\boldsymbol{P} = (e_1, e_2., \ldots e_T)$ *in the network* $\mathcal{N}$ *such that* $\boldsymbol{P}$ *visits each* e *in* $\{e : w_e \neq 0\}$ *at least once.*

Definition 4 ($(\boldsymbol{w}, \boldsymbol{P})$-weighted measurement)**.** *Given a measurement weight vector* $\boldsymbol{w}$ *and a* $\boldsymbol{w}$*-spanning measurement* $\boldsymbol{P} = (e_1, e_2, \ldots e_T)$*, a* $(\boldsymbol{w}, \boldsymbol{P})$-weighted measurement *is a path* $\boldsymbol{Q} = (e'_1, e'_2, \ldots e'_H)$ *in the network* $\mathcal{N}$ *such that* $W(\boldsymbol{Q}, e) = W(\boldsymbol{P}, e) + 2w_e$ *for each link* e*.*

Observe that the end-to-end delay for a $\boldsymbol{w}$-spanning measurement $\boldsymbol{P}$ is equal to $\Delta(\boldsymbol{P}) = \sum_{e \in \mathcal{E}} W(\boldsymbol{P}, e) x_e$, and that for a $(\boldsymbol{w}, \boldsymbol{P})$-weighted measurement is equal to

$$\Delta(\boldsymbol{Q}) = \sum_{e \in \mathcal{E}} W(\boldsymbol{Q}, e) x_e = \Delta(\boldsymbol{P}) + 2 \sum_{e \in \mathcal{E}} w_e x_e. \tag{4.1}$$

Proof of Theorem 8: To prove Theorem 8, we start with a measurement matrix A drawn according to the SHO-FA-INT construction for Theorem 7. For each row of the measurement matrix, we construct two paths in the network - a spanning measurement and a weighted measurement. Next, we subtract the end-to-end delay for the spanning measurement from the weighted measurement to get an output that is exactly twice the measurement output corresponding to the compressive sensing measurement using measurement matrix A. Thus, we can apply the SHO-FA-INT reconstruction algorithm from Section 4.4 to reconstruct the delay vector $\mathbf{x}_{\mathcal{E}}$. More precisely, let A be a $Rm' \times n$ matrix drawn from the ensemble of Section 4.4, where $R = \mathcal{O}(\lceil \log(n)/\log(M) \rceil)$ and $n = |\mathcal{E}|$.

Measurement Design: Let $\boldsymbol{a}(i) = [a_{i1} a_{i2} \dots a_{in}]$ be the i-th row of A. Consider network measurements $\boldsymbol{P}(i)$ and $\boldsymbol{Q}(i)$ defined as follows. Let $\boldsymbol{P}(i)$ be an $\boldsymbol{a}(i)$-*spanning measurement* obtained by picking the links in $\{e : \boldsymbol{a}(i) \neq 0\}$ one-by-one and finding a path from one link to another. By the definition of the diameter of the graph, there exists a path of length at most D between any pair of links. Therefore, there exists a path $\boldsymbol{P}(i) = ((v_1, v_2), (v_2, v_3), \dots, (v_T, v_{T+1}))$ of length $T = \mathcal{O}(Dn/k)$ that covers all the $\mathcal{O}(n/k)$ vertices that have non-zero components in $\boldsymbol{a}(i)$.

Next, let $\boldsymbol{Q}(i) = (e'_1, e'_2 \dots, e'_{T'})$ be a $(\boldsymbol{P}(i), \boldsymbol{a}(i))$-*weighted measurement* of length $T' = T + 2\sum_{e \in \mathcal{E}} a_e(i)$ as follows. Let $e'_1 = (v_1, v_2)$. If $a_{(v_1,v_2)}(i) \neq 0$, we traverse the edge (v_1, v_2) an additional $2a_{(v_1,v_2)}(i)$ times by going in the forward direction, *i.e.* on (v_1, v_2), and the reverse direction, *i.e.* on (v_2, v_1), an additional $a_{(v_1,v_2)}(i)$ times each. Thus, for $\tau = 1, 3, 5, \dots, 2a_{(v_1,v_2)}(i)+1$, we set $e'_\tau = (v_1, v_2)$ and for $\tau = 2, 4, \dots, 2a_{(v_1,v_2)}(i)$, we set $e'_\tau = (v_2, v_1)$. Next, if $v_3 = v_1$, *i.e.,* we have already visited e_2, we traverse the link e_2 once more, else we traverse it $a_{(v_2,v_3)}(i) + 1$ times in the forward direction and $a_{(v_2,v_3)}(i)$ times in the reverse direction. That is, for $\tau = 2a_{(v_1,v_2)}(i) +$

$2, 2a_{(v_1,v_2)}(i)+4, \ldots, 2a_{(v_1,v_2)}(i)+2a_{(v_2,v_3)}(i)+2$, we set $e'_\tau = (v_2, v_3)$ and for $\tau = 2a_{(v_1,v_2)}(i)+3, 2a_{(v_1,v_2)}(i)+5, \ldots, 2a_{(v_1,v_2)}(i)+2a_{(v_2,v_3)}(i)+1$, we set $e'_\tau = (v_3, v_2)$. We continue this process for each link (v_t, v_{t+1}) in the path $\boldsymbol{P}(i)$. That is, if (v_t, v_{t+1}) has been visited already in either the forward or reverse direction by $\boldsymbol{Q}(i)$, we add it to $\boldsymbol{P}(i)$ only once, else, we traverse it an additional $a_{(v_t,v_{t+1})}(i)$ times in each direction. Therefore, $\boldsymbol{Q}(i)$ visits every edge $e \in \mathcal{E}$ a total of $2a_e(i)$ times more than $\boldsymbol{P}(i)$ does.

Reconstructing $\mathbf{x}_{\mathcal{E}}$*:* Next, we measure the end-to-end delays for the paths $\boldsymbol{P}(i)$ and $\boldsymbol{Q}(i)$ for each $i = 1, 2, \ldots, Rm'$ and let $y_i = (\Delta(\boldsymbol{Q}(i)) - \Delta(\boldsymbol{P}(i)))/2$. From equation (4.1), it follows that $y_i = \sum_{e\in\mathcal{E}} a_{ie} d_e$. Note that this exactly equals the output of a compressive sensing measurement with $\mathbf{x}$ as the input vector, A as the measurement matrix, and $\mathbf{y}$ and the measurement output vector. Using this observation, we input the vector $\mathbf{y}$ to the SHO-FA-INT algorithm to correctly reconstruct $\mathbf{x}$ with probability $1 - \mathcal{O}(1/k)$. The guarantees on the decoding complexity follow from the decoding complexity of the SHO-FA-INT algorithm and that on the total number of hops follows by noting that each link in a measurement path may be visited at most $2M$ times. ■

4.5.2 Node Delay Estimation

The measurement design and the decoding algorithm for node delay estimation proceeds in a similar way to the link delay estimation algorithm of Section 4.5.1. The difference here is that instead of assigning weights to links in a path, our design assigns weights to nodes in a path by visiting each node repeatedly. We skip the proof of Theorem 9 here as it essentially follows from the technique used in the proof of Theorem 8. The only difference is that for node delay estimation we add the isolation assumption. If there exists one congested node, $v \in \mathcal{V}$, whose neighbors are all congested

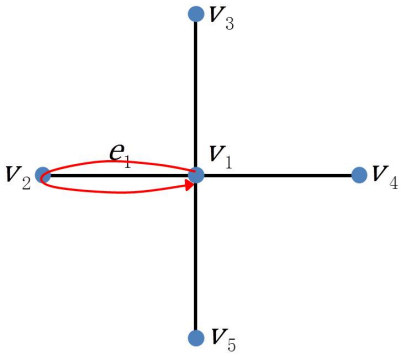


Figure 4.8: Isolated Node: For a subgraph with 5 vertices and 4 links, all vertices are congested. The node v_1 is not isolated since all of its neighbors are congested. Suppose there is a local loop involving v_1, v_2, and e_1. For link measurement, only the delay of e_1 is added to the weighted measurement. However, for the node measurement, the delays of v_1 and v_2 are both added to the weighted measurement. The delay of v_2 will not be canceled by the corresponding spanning measurement.

nodes, then we are not able to generate the measurement involving v by subtracting the weighted measurement from the spanning measurement. The reason is that each local loop involving v adds one more delay corresponding to one of its congested neighbor. However, this problem doesn't happen in edge delay measurements. (See Fig. 4.8.)

4.5.3 Extension of the FRANTIC algorithm

In fact, the ideas of cancellation and weighted measurements of the FRANTIC algorithm can be applied in any measurement matrix with integer entries and corresponding compressive sensing algorithm [15, 78, 145, 146] to generate new algorithms for network tomography problem. The reason why we stick to our SHO-FA-INT algorithm is that both the number of measurements and decoding complexity are order-optimal if there is no constraint on the number of times a packet may travel over any edge.

4.6 Exploiting network structure

4.6.1 Reducing Path Lengths through Steiner Trees:

One drawback of the approaches presented in the previous section is that even though on an average, each row of A contains only $\mathcal{O}(n/\rho k)$ non-zero entries, our upper bound on the path-length relies on worst-case pairwise paths for each pair of successive edges to be measured. In this subsection, we propose a Steiner-Tree-based approach to design the measurement paths given a measurement matrix A.

Definition 5 (Steiner Tree)**.** *Let $\mathcal{U} \subseteq \mathcal{V}$. We say that $\mathcal{T} \subseteq \mathcal{E}$ is a Steiner Tree for $\mathcal{U}$ if $\mathcal{T}$ has the least number of edges among all subsets of $\mathcal{E}$ that form a connected graph that is incident on every $v \in \mathcal{U}$. Let $L(\mathcal{U})$ be the length of a Steiner Tree for $\mathcal{U}$.*

For every $u \in \mathbb{Z}^+$, let

$$L^*(u) \triangleq \max_{\substack{\mathcal{U}:\mathcal{U}\subseteq\mathcal{V} \\ |\mathcal{U}|\leq s}} L(\mathcal{U}).$$

Note that, in general, $L^*(u) \leq Du$. Further, in many graphs of practical interest, $L^*(u) \ll Du$. For example, in a line graph with n vertices, $L^*(u)$ is at most n, while Du maybe as large as $\mathcal{O}(nu)$. Using this observation, we may further improve the performance guarantee of our algorithm. We note that it suffices to find a Steiner Tree that passes through all links specified by a given row of the measurement matrix A. Also, we already know that, with high probability, the number of non-zero entries in each row of A is $\mathcal{O}(n/\rho k)$. Thus, in general, the number of links traversed by each link (or node) delay measurement is $\mathcal{O}(L^*(u))$ where $s = \mathcal{O}(|\mathcal{E}|/\rho k)$ (or $\mathcal{O}(|\mathcal{V}|/\rho k)$, respectively) is the number of non-zero entries in the measurement. This proves the following assertion.

Theorem 10 (Network tomography for link/node congestion using Steiner Trees)**.** *For the setting of Theorem 8, the number of links of $\mathcal{N}$ traversed by each measurement of FRANTIC is at most $\mathcal{O}(L^*(u))$ where $s = \mathcal{O}(|\mathcal{E}|/\rho k)$ is the number of non-zero entries in the measurement and the total number of hops for each measurement is $\mathcal{O}(ML^*(u))$.*

Remark: There exist polynomial-time approximation schemes with a performance ratio decreased from 2 to 1.55 by a series of works [18, 75, 86, 120, 123, 133, 149, 150].

4.6.2 Average length of Steiner Trees:

In Theorem 10, we analyzed the length of measurement paths in terms of the worst-case length of Steiner trees that contain an arbitrary subset of s links (respectively nodes). However, on average, however, this may be too conservative an estimate.

Definition 6 (Average length of Steiner tree)**.** *For every $u \in \mathbb{N}^+$, let*

$$\overline{L}(u) \triangleq \frac{\sum_{\mathcal{U}:\mathcal{U}\subseteq\mathcal{V}} L(\mathcal{U})}{|\{\mathcal{U} \subseteq \mathcal{V} : |\mathcal{U}| = u\}|}$$

denote the average length of Steiner tree.

In the example shown in Fig. 4.9, we argue that, with high probability, the length of paths required is upper bounded by $\overline{L}(u)$ which may be significantly smaller than $L^*(u)$.

4.6.3 Network decomposition:

Since we already know the topology of the network, exploring the structure of the topology may help us to reduce the path length of each measurement. In Fig. 4.10, we illustrate how to reduce the length of Steiner tree by network decomposition.

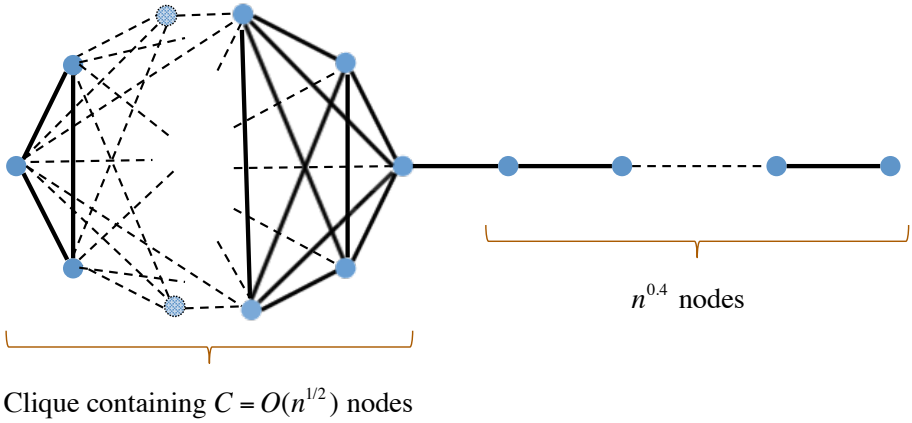


Figure 4.9: Worst-case vs Average length of Steiner trees: Consider a network of n links. The network has two parts - a clique consisting of $C = (1 + \sqrt{1 + 8(n - n^{0.4})})/2$ fully connected nodes and a line consisting of $n^{0.4}$ nodes. Let the number of congested links in the network be $k = n^{0.95}$. Thus, each measurement path has to cover a set of links of size $\mathcal{O}(n^{0.05})$ specified by SHO-FA-INT . In the worst case, such a set can include the two ends of the linear subnetwork. Thus, in the worst case, the length of the Steiner tree can exceed $n^{0.4}$. However, we note that the SHO-FA-INT algorithm picks the measurement nodes uniformly at random. Thus, the probability of picking even one edge from the linear part of the network is $\mathcal{O}(n^{0.05} \times n^{0.4}/n^{0.5}) = \mathcal{O}(n^{-0.05})$ by the union bound. Therefore, on an average, the length of Steiner tree is at most $\mathcal{O}(n^{-0.05} \times n^{0.4}) = \mathcal{O}(n^{0.35})$, which is lower than the worst-case length of a Steiner tree covering $\mathcal{O}(n^{0.05})$ links in the network.

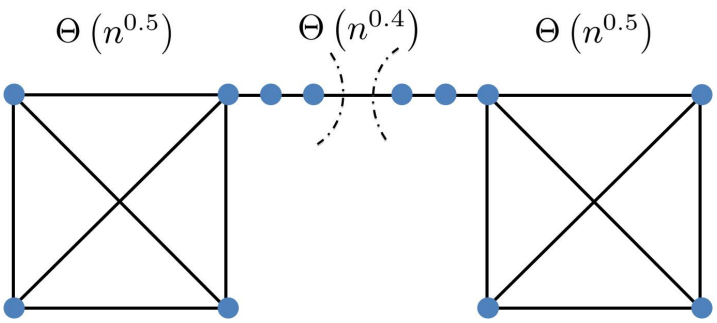


Figure 4.10: Network Decomposition: The network consists of three parts – two complete graphs with $\Theta\left(n^{0.5}\right)$ vertices and a line graph with $\Theta\left(n^{0.4}\right)$ vertices. It follows that there are $\Theta(n)$ links in each of the two complete graphs and $\Theta\left(n^{0.4}\right)$ links in the line graph. For each link measurement, with high probability, two links involved locate in each of two complete graphs. Therefore, the average length of Steiner tree is at least $\Theta\left(n^{0.4}\right)$. If we decompose the original network into two subgraphs as shown in the figure and do the link delay estimation on them separately, the average length of Steiner tree becomes at most $\Theta\left(n^{0.35}\right)$ (shown in Fig. 4.9) which is smaller than $\Theta\left(n^{0.4}\right)$.

4.7 Acknowledgements

This work was partially supported by a grant from University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08), a grant from the Microsoft-CUHK Joint Laboratory for Human-centric Computing and Interface Technologies, and an SHIAE grant.

□ **End of chapter.**

Chapter 5

Group Testing – GROTESQUE

5.1 Introduction

Suppose a "large" number (denoted by n) of items contains a "small" number (denoted by k, where k is assumed to be "much smaller" than n) of "defective" items. The problem of *Group Testing* is to identify the defective items by using as few "group tests" as possible. This problem was first considered by Dorfman in 1943 [54] as a means of identifying a small number of diseased individuals from a large population via "pooled tests". In this scenario, blood from a subset of individuals is pooled together and tested in one go. If the test outcome was "negative", then the subset of individuals tested in that test does not contain a diseased individual, otherwise it contains at least one diseased individual. In the ensuing decades a rich literature pertaining to the problem has built up and group testing has found many applications in different areas such as multiple access communication [69, 142], DNA Library Screening [38, 106, 126] – a good survey of some of the algorithms, bounds and applications can be found in the books by Du and Hwang [55, 56].

Number of tests: A natural information-theoretic lower bound on the

number of tests required to identify the set of defectives is $\Omega(k\log(n/k))$.[1] One way of deriving this follows by noting that the number of bits required to even describe a subset of size k from a set of size n equals $\log\left(\binom{n}{k}\right) = k\log(n/k)(1+o(1))$, hence at least this many tests (each of which have binary outcomes) are needed. In this work, for ease of presentation of our results, we focus on the setting where $k = \mathcal{O}\left(n^{1-\Delta}\right)$ for some $\Delta > 0$ (though in principle our results hold in greater generality). All asymptotic notation will be relative to the asymptote n. In this regime, the number of required tests scales as $\Omega(k\log(n))$. Note that this argument also demonstrates that the decoding complexity of any group-testing algorithm scales as $\Omega(k\log(n))$.[2]

Group-testing problem comes in different flavours. There are at least three different design choices for group testing problems depending on the applications – whether the tests are noiseless or not, adaptive or not, and the algorithm is required to be zero-error or not.[3]

- *Noiseless vs. noisy tests:* If test outcomes are always positive when tests contain at least one defective item, and always negative otherwise, then they are said to be *noiseless* tests. In some settings, however, the test outcomes are "noisy" – a common model for this noise (*e.g.,* [8, 36, 96]) is when test outcomes are flipped i.i.d.[4] via

[1] We use standard computer science Big Oh notation in this work. Details are in the Section 2.3.

[2] A slightly more involved information-theoretic argument is required if the group-testing procedure is allowed to fail with "small" probability. However, even in this setting, essentially the same lower bounds can be proven (up to small constant multiplicative factors that may depend on the allowed probability of error) – *e.g.*, [36, 95].

[3] Another flavour we do not focus on in this work corresponds to whether the design of the testing procedure is deterministic or randomized. Recent works – for instance [97, 117] – provide computationally efficient deterministic designs. In this work, however, we focus on "Monte Carlo" type randomized design algorithms.

[4] A "worst-case" noise model wherein *arbitrary* (rather than *random*) errors, up to a constant

Bernoulli(q) noise.[5] It is known in several settings (*e.g.,* [8, 36, 96]) that the number of noisy tests required to reconstruct the set of defectives is at most a constant factor greater than the number of noiseless tests required (both requiring $\mathcal{O}(k\log(n))$ tests). This constant factor depends on q proportionally with $1/(1-H(q))$, where $H(q)$ is the binary entropy function. In this work we focus on the general setting with noisy measurements.

- *Adaptive vs. Non-adaptive tests:* Adaptive group testing refers to a setting where the design of tests of any stage is allowed to depend on the test outcomes from prior stages. In contrast, in a "non-adaptive" algorithm, all tests are designed a priori. Whether the tests are noiseless and noisy tests, as long as a "small" probability of error is allowed for the reconstruction algorithm (Monte-Carlo algorithms), it turns out the number of tests required by even non-adaptive algorithms meets (up to a constant factor that may depend on q) the information-theoretic lower-bound of $\mathcal{O}(k\log(n))$ (*e.g.,* [36, 38, 48, 49, 99]). Given this, non-adaptive algorithms are often preferred to adaptive algorithms in applications, since they allow for parallelizable implementation and/or the usage of off-the-shelf hardware. Even among the class of adaptive algorithms, it is preferable to have *as few* adaptive stages as possible. In this work we focus on both adaptive and non-adaptive algorithms. In the case of adaptive algorithms, we further also consider the case of adaptive algorithms with just two stages (*i.e.,* only one round of feedback).

fraction of tests, has also been considered in the literature (*e.g.,* [8, 105]).

[5]Other types of noise have also been considered in the literature – another well-analyzed type called "dilution" noise (*e.g.,* [76]) corresponds to tests with fewer defectives having a higher chance of resulting in a false positive outcome.

- *Zero-error vs. "small-error" algorithms:* Potential goals for group testing algorithms (in the setting with noiseless tests) are to either *always* identify the set of defective items correctly, or to output it correctly "with high probability". With adaptive tests, it turns out that both these settings require $\Theta(k\log(n))$ tests (*e.g.,* [38] for the former setting and [48] for the latter). With non-adaptive tests, however, it turns out that requiring zero-error reconstruction implies that at least $\Omega(k^2\log(n)/\log(k))$ tests must be performed [57, 58]. Given this potentially large gap between the number of tests required in the two setting, in this work we focus on the "small-error" setting. In our work, our algorithms are said to fail if not all defectives are recovered – our goal is to design testing procedures that do *not* fail with high probability (over the randomness in the set of items that instantiate as defectives). As is common in information theory achievability arguments, we instead prove that for any specific set of k defective items, with high probability over testing procedures, our algorithms work as claimed. This implies via standard averaging arguments that "most" testing procedures also work as claimed.

Decoding complexity:[6] The discussion above focused exclusively on the number of tests required, with no regard to the computational complexity of the corresponding reconstruction algorithms. While many of the algorithms reprised above are reasonably computationally efficient, the decoding complexity of most still scales at least linearly in n. Some notable exceptions are the results in [72, 77, 105]. The most recent work in this line [105] with decoding complexity that is sub-linear in n culminated in a group-testing algorithm with $m = \mathcal{O}\left(k^2\log(n)\right)$ tests, and decoding com-

[6] The decoding complexity is defined as number of arithmetic operations (see Section 5.3 for details of what comprises a basic arithmetic operation).

plexity that scales as poly(m).[7] As noted in [77], such algorithms can find applications in data-stream algorithms for problems such as the "heavy hitter" problem [43] or in cryptographic applications such as the "digital forensics" problem [68]. Also as noted in [105], *"[the group-testing] primitive has found many applications as stand alone objects and as building blocks in the construction of other combinatorial objects."* We refer the reader interested in these and other applications to the excellent expositions in [77, 105], and focus henceforth simply on the purely combinatorial problem of group-testing.

Our starting point is to note that since the sub-linear time algorithms in [77, 105] require zero-error reconstruction, the penalty paid in terms of the number of tests is heavy ($\Omega(k^2 \log(n)/\log(k))$ as opposed to $\mathcal{O}(k \log(n))$). Further, the decoding complexity in [77, 105] is a low-degree polynomial in $m = \mathcal{O}(k^2 \log(n))$, leaving a significant gap vis-a-vis the information-theoretic lower-bound of $\Omega(k \log(n))$ decoding steps (since any algorithm must examine at least $\mathcal{O}(k \log(n))$ test outcomes to have a "reasonable" probability of success). Hence in this work we choose to improve on both these parameters with respect to [77, 105] – number of measurements and decoding complexity.

5.1.1 Our contributions

In our work, we consider both the adaptive and non-adaptive group-testing settings, with noisy or noiseless tests, and decoding error scaling as $\mathcal{O}(1/\operatorname{poly}(k))$ in both settings.[8] We present our theorem statements including internal code-design parameters that may be somewhat freely chosen. Let $g(q) = \frac{32}{(1-2q)^2} + \frac{256}{(1-H(q))^3}$, where $H(\cdot)$ is the binary entropy function.

[7] An algorithm with $\mathcal{O}(k \log(n/k))$ tests was also presented in the regime where $k = \Theta(n)$.

[8] The term $\mathcal{O}(1/\operatorname{poly}(k))$ is defined in Section 2.3.

***Multi-stage adaptive algorithm*:** In the adaptive setting, the given algorithm is the first to be simultaneously order-optimal (up to a small constant factor that depends on the noise parameter q) in the number of tests required as well as in the decoding complexity and achieves $\Theta(k\log(n))$ for both measures. Our adaptive algorithm also does not need "much" adaptivity. In particular, our algorithm has $\mathcal{O}(\log(k))$ stages, where the tests within each stage are non-adaptive – it is only across stages that adaptivity is required. More precisely, the adaptive group testing algorithm achieves the following guarantees.

Theorem 11. *For any set of defectives $\mathcal{S}\subseteq\mathcal{I}$, where $|\mathcal{S}| = k$ and $|\mathcal{I}| = n$ are large enough, the Multi-stage Adaptive Group Testing algorithm with Bernoulli(q) noise produces a reconstruction $\hat{\mathcal{S}}$ of $\mathcal{S}$ such that $\hat{\mathcal{S}} = \mathcal{S}$. Let $C \geq \frac{(1-H(q))^2 g(q)}{128}$, $c_{\text{adp,rn}} \geq 0$, $c_{\text{adp,deg}} \geq 0$ be constants. The algorithm has the following properties:*

1) The number of tests m is at most

$$4e^{1/2}Ck\log(n) + c_{\text{adp,rn}}C(\log(k))^2\log(\log(k))\log(n),$$

2) The number of stages is at most $1+\log\left(k/\log(k)\right)/[-\log\left(1-e^{-1/2}/2\right)]$,

3) The decoding complexity is $\left[4e^{1/2}k + c_{\text{adp,rn}}(\log(k))^2\log(\log(k))\right]\left[\frac{32C}{(1-2q)^2g(q)}\log(n) + 2\log\left(\frac{32C}{(1-2q)^2g(q)}\log(n)\right) + t_{\text{loc}}\frac{256C}{(1-H(q))^3g(q)}\log(n)\right]$ *for some positive constant t_{loc}, and*

4) The error probability is at most $\frac{\log\left(D/\log(D)\right)}{-\log\left(1-e^{-1/2}/2\right)}k^{-1/(4e)} + 12e^{1/2}kn^{-\frac{C}{g(q)}} + (\log(k))^{1-\left(c_{\text{adp,rn}}P_{\text{leaf}}\right)\log(k)}$ *over the internal randomness in the algorithm. Here, $P_{\text{leaf}} \geq c_{\text{adp,deg}}e^{-c_{\text{adp,deg}}}$ is a constant.*

***Non-adaptive algorithm*:** Analogously, in the non-adaptive setting we present the first algorithm that is simultaneously near-optimal in both number of measurements and decoding complexity (requiring $\mathcal{O}(k\log(k)\log(n))$

tests and having a decoding complexity of $\mathcal{O}(k(\log(n) + \log^2(k)))$. More precisely, the non-adaptive algorithm achieves the following guarantees.

Theorem 12. *For any set of defectives $\mathcal{S} \subseteq \mathcal{I}$, where $|\mathcal{S}| = k$ and $|\mathcal{I}| = n$ are large enough, the Non-Adaptive Group Testing algorithm with Bernoulli(q) noise produces a reconstruction $\hat{\mathcal{S}}$ of $\mathcal{S}$. Let $c_{\text{non,bpt}}$, $c_{\text{non,rn}}$, c_{mul}, and $c_{\text{loc}} \geq \frac{2}{1-H(q)}$ be positive constants. The algorithm has the following properties:*

1) The number of tests m equals

$$c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k)(c_{\text{mul}} \log(k) + c_{\text{loc}} \log(n)),$$

2) The decoding complexity is

$[c_{\text{mul}} \log(k) + 2\log(c_{\text{mul}} \log(k))]\,(c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k))$
$+(c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k)) \log(c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k)) + k[t_{\text{loc}} c_{\text{loc}} \log(n) +$
$2\log(c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k))(c_{\text{non,bpt}} \log(k))]$ *for some positive constant t_{loc}, and*

3) The error probability is at most $k^{1-\exp(-1/c_{\text{non,rn}})c_{\text{non,bpt}}/8} +$
$2c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k) \exp\left((-c_{\text{mul}}(1-2q)^2/32)\log(k)\right) +$
$k \exp((-c_{\text{loc}}(1-H(q))^3/256)\log(n))$ *over the internal randomness in the algorithm.*

***Two-stage adaptive algorithm*:** Finally, combining ideas from the adaptive and non-adaptive algorithms, we present the first 2-stage algorithm that is simultaneously near-optimal in both number of measurements and decoding complexity, with both the number of tests and the decoding complexity scaling as $\mathcal{O}(k(\log(n) + \log^2(k)))$. More precisely, the two-stage adaptive algorithm achieves the following guarantees.

Theorem 13. *For any set of defectives $\mathcal{S}$ from $\mathcal{I}$ ($|\mathcal{S}| = k$ and $|\mathcal{I}| = n$ are large enough), the Two-stage Adaptive Group Testing algorithm with*

Bernoulli(q) noise produces a reconstruction of the collection $\hat{\mathcal{S}}$ *of* $\mathcal{S}$*. Let* $c_{\text{non,bpt}}$, $c_{\text{non,rn}}$, c_{mul}, $c_{\text{loc}} \geq \frac{2}{1-H(q)}$ *be positive constants and let* $s = k^{\gamma}$ *with* $\gamma > 3$,

1) The number of tests m *equals* $c_{\text{non,bpt}}c_{\text{non,rn}}(c_{\text{mul}} + \gamma c_{\text{loc}})k(\log(k))^2 + c_{\text{loc}}k\log(n)$,

2) The number of stages is 2,

3) The decoding complexity is $[c_{\text{mul}}\log(k) + 2\log(c_{\text{mul}}\log(k))](c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) + (c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) + k[t_{\text{loc}}c_{\text{loc}}\gamma\log(k) + 2\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))(c_{\text{non,bpt}}\log(k))] + t_{\text{loc}}c_{\text{loc}}k\log(n)$ *for some positive constant* t_{loc}*, and*

4) The error probability is at most $k^{1-\exp(-1/c_{\text{non,rn}})c_{\text{non,bpt}}/8} + 2c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)\exp\left((-c_{\text{mul}}(1-2q)^2/32)\log(k)\right) + k\exp((-c_{\text{loc}}(1-H(q))^3/256)\log(S)) + k^{3-\gamma} + k\exp(-c_{\text{loc}}(1-H(q))^3/256)\log(n))$ *over the internal randomness in the algorithm.*

Remark 9. While in the above theorem, and throughout this work, we assume that we know the value of k exactly, in reality our algorithms can be shown to work (with a constant factor loss in performance) even if we only know the value of k up to a constant factor. Indeed, estimating k up to a constant factor is the subject of several works on competitive group testing (CGT) such as [47,131], which are able to generate such an estimate (with $\mathcal{O}(\log(n))$ tests in that extra stage).

One could even envisage models wherein only an upper bound k'' on k is known, and measure performance against $k''\log(n)$ (in terms of number of tests and decoding complexity). Indeed, this is the paradigm presented in, for example [36]. We claim (without presenting details) that with sufficient care the algorithms in this work can also be suitably modified to fit this paradigm. However, for ease of presentation of already complex

results, we focus on the "clean" case in which k is known precisely. The challenge in the "full" analysis is primarily in showing suitable concentration of the probability of error even when the "true" number of defective is much smaller than the outer bound k - while this can be handled, it would make the presentation much messier.

The rest of this chapter is organized as follows. We first present the high-level overview of GROTESQUE tests (which is the main tool for our algorithm designs) and three group testing algorithms in Section 5.2. Sections 5.4–5.7 contain detailed descriptions and analysis of GROTESQUE tests and our group testing algorithms. Section 5.9 concludes this work.

5.2 High-level overview

We now preview the key ideas used in designing our algorithms.

We begin by noting that our multi-round adaptive algorithm has decoding complexity that is information-theoretically order-optimal (and in some parameter ranges, for instance when $k = \mathcal{O}(\text{poly}(\log(n)))$, the 2-round adaptive algorithm does too). If our algorithms are to indeed be as blindingly fast as claimed above, it'd be very nice to have a "black-box" that has the following property – with probability $1 - \mathcal{O}(1/\text{poly}(k))$, given $\mathcal{O}(\log(n))$ (noisy) non-adaptive tests on a subset of items that contain exactly one defective item that has not yet been identified, in $\mathcal{O}(\log(n))$ time the black-box outputs the index number of this defective item. Our multi-stage adaptive group testing algorithm then gives to this black-box subsets of items that, with constant probability, contain exactly one unidentified defective item. Our non-adaptive group testing algorithm, on the other hand, gives to this black-box subsets of items that, with probability $\Omega(1/\log(k))$, contain exactly one unidentified defective item. These choices

lead to the claimed performance of our algorithms.

5.2.1 GROTESQUE Tests

We first intuitively describe a non-adaptive testing and decoding procedure (which we call GROTESQUE[9] testing) that implements the "black-box" described above. Details, with explicit characterizations of constants, are in Section 5.4.

GROTESQUE first performs *multiplicity testing* – it takes as inputs a set of n' items (where n' may in general be smaller than n), and "quickly" (in time $\mathcal{O}(\log(k))$) first estimates (with "high" probability[10]) whether these n' items contain 0, 1, or more than one defectives. If the n' items contain 0 or more than 1 defectives, GROTESQUE outputs this information and terminates at this point. However, if the n' items contain exactly 1 defective item, it then performs *localization* – it "quickly" (in time $\mathcal{O}(\log(n'))$) estimates (with "high" probability) the index number of this item. Both these processes (multiplicity testing, and localization) are non-adaptive.

- ***Multiplicity tests*:** The idea behind multiplicity testing is straightforward – GROTESQUE simply performs $\Theta(\log(k))$ *random* tests, in which each of the n' items is present in each of the $\Theta(\log(k))$ tests with probability 1/2 (hence these tests are non-adaptive). As Table 5.4 demonstrates, if the set of n' items being tested has exactly one defective item, then in expectation about half the $\Theta(\log(k))$ multiplicity tests should have positive outcomes, otherwise the number of tests with positive outcomes should be strictly bounded away from 1/2 (even if the tests are noisy). In fact, the probability of error in

[9] GROTESQUE is short for **GRO**up **TES**ting **QU**ick and **E**fficient.

[10] In this work, the term "with high probability" means that the error probability approaches 0 as k *or* n grows without bound. The precise error probabilities are carefully calculated in the affiliated analysis.

the Multiplicity testing stage can be concentrated to be lower than $\exp(-\Theta(\log(k))) = \mathcal{O}(1/(\text{poly}(k)))$.

- ***Localization tests*:** The idea behind *localization* is somewhat more involved. For this sub-procedure, GROTESQUE (non-adaptively) designs *a priori* a sequence of binary $\Theta(\log(n)) \times n'$ matrices.[11] In particular, the columns of each such matrix correspond to the collection of codewords of a *constant-rate expander code* [132] with block-length $\Theta(\log(n))$. In brief, these are error-correcting codes whose redundancy is a constant fraction of the block-length that can correct a constant fraction of bit-flips with "high probability" (for instance, Barg and Zémor [13] analyze their performance against the "probability q bit-flip noise" and demonstrate that the probability of error decays exponentially in the block-length). Further, expander codes have the desirable property that their decoding complexity scales linearly in the block-length. Conditioning on the event that the multiplicity of defectives in the n' items being tested equals exactly 1 (say the i-th item is defective), this means that in the *noiseless* setting, the binary vector of $\Theta(\log(n))$ outcomes of the localization tests performed by GROTESQUE correspond exactly to the i-th codeword of the expander code. Even in the *noisy* setting, the vector of test outcomes corresponds to the i-th codeword being corrupted by Bernoulli(q) bit-flips. In both of these settings, by the guarantees provided in [13, 132], the GROTESQUE localization procedure outputs the incorrect index (corresponding to the defective item) with probability $\exp(-\mathcal{O}(\log(n))) = \mathcal{O}(1/(\text{poly}(n))) = o(1/(\text{poly}(k)))$.

[11] As mentioned before, n' could generally be much smaller than n. However, in our algorithms, n' could be as large as n in the worst case. Therefore, we choose the block-length of expander code to be $\log(n)$.

We now present the ideas behind our three algorithms, highlighting the use of GROTESQUE tests in each.

5.2.2 Adaptive Group Testing

For the adaptive group-testing problem, we now use a few "classical" combinatorial primitives–"balls and bins problem", McDiarmid's concentration inequality, "coupon collector's problem"– combined carefully with the GROTESQUE testing procedure. We present the intuition here and the detailed calculations in Section 5.5. Our algorithm works in two phases:

- ***Random binning***: We first note that if we randomly partition the set of all N items into say $\Theta(k)$ disjoint pools (each with $n/\Theta(k)$) items, then with "high" probability (via McDiarmid's inequality [98]) a constant fraction of the pools contain exactly one defective item. Hence GROTESQUE can be used on each disjoint pools as inputs with $n' = n/(\Theta(k))$. Thus, in a single stage of $\Theta(k)$ pools and corresponding $\Theta(k) \cdot \mathcal{O}(\log(n)) = \mathcal{O}(k \log(n/k))$ non-adaptive tests, we can identify a constant fraction of the D defective items (with probability at least $1 - \exp(-\Theta(k))$). In the subsequent $\mathcal{O}(\log(k))$ stages, since the number of unidentified defectives decays geometrically, the number of pools per stage can be chosen to decay geometrically for comparable performance. Since the number of tests decay geometrically, the overall number of GROTESQUE tests sum up to $\mathcal{O}(k)$. However, each GROTESQUE test requires at most $\log(n)$ tests with corresponding time-complexity $\mathcal{O}(\log(n))$. Hence the overall number of tests, and time-complexity, of these random binning stages is $\mathcal{O}(k \log(n))$.

  However, by the time we're at the $\mathcal{O}(\log(k))$-th stage, the number of remaining unidentified defective items is "small" (at most $\log(k)$).

Hence concentration inequalities may not provide the desired decay in the probability of error of that stage (corresponding to the event that the stage correctly recovers less than a certain constant fraction of the defective items remaining from the previous stage). The overall probability of error of all the random-binning stages is dominated by the probability of error of the last random-binning stage.

- ***Coupon collection***: To compensate for this "problem of small numbers", in the last stage we segue to an alternative primitive, that of *coupon collection* [103]. We choose parameters so that at the beginning of this coupon-collection stage, there are less than $\log(k)$ unidentified defectives remaining. Rather than *partitioning* the set of items into pools as in the previous stages, in this stage we *independently* choose $\mathcal{O}(\log^2(k)\log(\log(k)))$ pools (corresponding to the "coupons" in the coupon-collector's problem) – note that $\mathcal{O}(\log^2(k)\log(\log(k))) = o(k\log(n))$, hence this coupon-collection stage does not change the overall number of tests required by more than a constant factor. Each pool is chosen to be of size so that with constant probability it contains one of the remaining $\mathcal{O}(\log(k))$ unidentified defectives. Each pool/coupon is given as an input to GROTESQUE. By standard concentration inequalities on the coupon collection process, after $\mathcal{O}(\log^2(k)\cdot\log(\log(k)))$ coupons have been collected, with probability $1-\mathcal{O}(1/\operatorname{poly}(k))$ all the defectives are decoded.

To summarize the discussion on the number of tests and decoding complexity which form the primary thrust of this work.

- Random binning:
  - In the random binning phase there are $\mathcal{O}(\log(k))$ stages.

  - In the first stage GROTESQUE is called $\mathcal{O}(k)$ times. Hence the number of tests and decoding complexity are both $\mathcal{O}(k\log(n))$.
  - In i-th stage, the number of GROTESQUE calls is a constant factor c smaller than in the previous stage. Therefore, the number of tests and decoding complexity decay geometrically in proportion to c^i.
  - Summing the above geometric series, the total number of GROTESQUE calls is $\mathcal{O}(k)$. Hence, the number of tests and decoding complexity are both $\mathcal{O}(k\log(n))$.

- Coupon collection:
  - The number of GROTESQUE calls is $\mathcal{O}(\log^2(k)\log(\log(k)))$. Therefore, the number of tests and decoding complexity are both

$$\mathcal{O}(\log^2(k)\log(\log(k))\log(n)) = o(k\log(n)).$$

5.2.3 Non-adaptive Group Testing

We present the intuition here and the detailed calculations in Section 5.6. The critical difference between adaptive and non-adaptive group testing is that defective items that have already been identified cannot be excluded from future tests. This means that if we naïvely use the adaptive procedure outlined above (after suitably optimizting the parameters) we get an algorithm with $\mathcal{O}(k\log(k)\log(n))$ tests and $\mathcal{O}(k\log(k)\log(n))$ decoding complexity. Instead, we redesign our testing procedure to speed up the decoding complexity to $\mathcal{O}(k(\log^2(k)+\log(n)))$ (though we still need $\mathcal{O}k\log(k)\log(n)$ tests). In particular, we first non-adaptively choose a set of $\mathcal{O}(\log(k))$ random graphs with the following properties – each graph $\mathcal{G}_g$ is bipartite, has n nodes on the left of $\mathcal{G}_g$ and is left-regular with left-degree 1, and has $\mathcal{O}(k)$ nodes on the right. Each left node of $\mathcal{G}_g$ corresponds to an

item. Each right node corresponds to a group of $\mathcal{O}(\log(n))$ non-adaptive (GROTESQUE) tests, for a total of $\mathcal{O}(k\log(k)\log(n))$ non-adaptive tests. In the rest of this work, we refer to a left node as an item node and to a right node as a testing node. We show that, with a total of $k\log(k)$ testing nodes, our iterative decoding algorithm identifies exactly one defective item in each iteration, and all defectives are identified in k iterations.

A node on the right of $\mathcal{G}_g$ is said to be a "leaf node with respect to $\mathcal{G}_g$" if the item nodes connected to it contain exactly one defective item (or in other words, GROTESQUE's multiplicity test, run on the items connected to such a node, would with high probability return a value of 1). It can be shown via standard concentration inequalities that for each defective item (on the left of each bipartite graph $\mathcal{G}_g$), a constant fraction of its $\mathcal{O}(\log(k))$ right neighbours (over all $\mathcal{O}(\log(k))$ graphs) are such that they are "leaf nodes with respect to $\mathcal{G}_g$". For each $\mathcal{G}_g$, the items/left-nodes connected to its testing/right nodes may now be given as an input to GROTESQUE (with $n' = \mathcal{O}(\log(n))$ tests (however, in our actual algorithm, not all testing nodes of all $\mathcal{G}_g$s are necessarily chosen as inputs to GROTESQUE – the speedup in decoding complexity arises crucially from a more careful procedure in deciding which testing nodes to use to give inputs to the GROTESQUE testing procedure).

Decoding proceeds by iteratively following the steps below:

1. *Initialization of leaf-nodes:* We initialize a *leaf-node list* that contains all leaf nodes. We do this by picking testing nodes and sequentially feeding the corresponding item nodes to GROTESQUE's multiplicity testing procedure (*not* its localization procedure, at least yet).

2. *Localization of a single defective item:* We pick a testing node in the leaf node list, and use GROTESQUE's localization testing procedure on this node to identify the corresponding defective item.

3. *Updating the leaf-node list:* We remove all the right neighbours of the defective item identified in the previous stage from each of the $\mathcal{O}(\log(k))$ graphs, and update the leaf node list. Finally we return to Step 2, until all D defectives have been found.

It can be verified that the first and third steps of this algorithm both take $\mathcal{O}\left(k\log^2(k)\right)$ steps, and the second step takes $\mathcal{O}(k\log(n))$ steps, thus giving us the overall desired computational complexity.

To summarize the discussion on the number of tests and decoding complexity:

- The number of testing nodes is $\mathcal{O}(k\log(k))$. Therefore, the number of tests are $\mathcal{O}(k\log(k)\log(n))$.
- The decoding complexity for initializing a leaf-node list, localizing defectives via "peeling process", and updating leaf-node list iteratively are $\mathcal{O}(k\log^2(k))$, $\mathcal{O}(k\log(n))$, and $\mathcal{O}(k\log^2(k))$, respectively. Therefore, the decoding complexity is $\mathcal{O}(k\log(n)+k\log^2(k))$.

5.2.4 Two-stage Adaptive Group Testing

We now merge ideas from our previous algorithms to present an adaptive group testing algorithm with "minimal adaptivity" (just two stages). We also use in our algorithm a key primitive suggested in Theorem 1 of [48], specifically "birthday paradox hashing". We present intuition here, with detailed calculations in Section 5.7.

The main difference between our algorithm and the one presented in Theorem 1 of [48] is that our algorithm is robust to Bernoulli(q) noise, has decoding complexity scaling as $\mathcal{O}(k(\log(n)+\log^2(k)))$, and number of tests scaling as $\mathcal{O}(k(\log(n)+\log^2(k)))$. In contrast, the algorithm in [48]

requires fewer tests ($\mathcal{O}(k\log(n))$), but significantly higher decoding complexity ($\mathcal{O}(\exp(n))$), and is not robust to noise in the measurement process.

The high-level intuition behind the algorithm in Theorem 1 of [48] is to first partition the n items into at least k^2 groups. The "Birthday Paradox" is a simple calculation that demonstrate that if k balls are thrown uniformly at random into more than k^2 bins, then the probability of a "collision" (there being a bin with more than one ball in it) is small.

Using this primitive, it follows that with high probability each group contains at most one defective item. In the first stage, $\mathcal{O}\left(k\log(k^2)\right)$ non-adaptive group tests are performed to identify the k groups (out of k^2) that contain exactly one defective.

In the second stage (that depends adaptively on the outcomes of the first stage), $\mathcal{O}\left(\log(n/k^2)\right)$ non-adaptive group tests are performed on the n/k^2 items of each group that has been identified as containing a defective in the first stage. Thus, the total number of tests required for the second stage is $\mathcal{O}\left(k\log((n/k^2))\right)$.

However, the high decoding complexity of the algorithm in Theorem 1 of [48] arises from the fact that the non-adaptive group testing algorithm used has high decoding complexity. We hence substitute the non-adaptive group test used in their scheme with the one presented in Section 5.6 resulting in a drastic decrease in the decoding complexity at the cost of a potential slight increase (an additive factor of $\mathcal{O}(k\log^2(k))$) in the number of tests required. Another relatively minor difference in our algorithm is that to get the probability of error to decay as $\mathcal{O}(1/\operatorname{poly}(k))$ as desired for all our algorithms, we use $\operatorname{poly}(k)$ groups (where the polynomial is of degree at least 3) in the first stage instead of the $\Omega(k^2)$ groups used in the first stage of [48].

To summarize the discussion on the number of tests and decoding com-

plexity:

- Birthday paradox:
  - Since we apply our non-adaptive algorithm in this stage, the number of tests and decoding complexity are both $\mathcal{O}(k \log^2(k))$.
- Localization:
  - The number of GROTESQUE calls is exactly k. Hence the number of tests and decoding complexity are both $\mathcal{O}(k \log(n))$.

Group Testing	
n	The total number of items
k	The total number of defective items
Δ	$k = \mathcal{O}\left(n^{1-\Delta}\right)$
q	The pre-specified probability that the result of a test differs from the true result
$\mathcal{S}$	The set of all k defective items
$\mathcal{I}$	The set of all n items
m	The total number of tests required to identify the set of defective items

Table 5.1: Table of notation used for the general group testing problem

5.3 Basic Arithmetic Operations

Since in this work we claim computational complexity that is essentially information-theoretically order-optimal, we have to be careful about our computation model. We thus discuss the basic arithmetic operations used in this work and their corresponding time complexities. The usage of Red-Black trees, a type of binary tree, is specifically for our non-adaptive group testing algorithm to manipulate the leaf-node list in an efficient manner. See [14] for the details of Red-Black trees and Section 5.6 for how we use them.

GROTESQUE TESTS	
$\mathcal{I}'$	The set of items being tested in GROTESQUE TESTS.
n'	The number of items being tested in GROTESQUE TESTS, $n' = \lvert\mathcal{S}\rvert$
k'	The total number of defectives of GROTESQUE TESTS input
m_{mul}	The number of Multiplicity tests
m_{loc}	The number of Localization tests
K	The number of positive results of Multiplicity tests
$\mathcal{C}$	Expander code
$\mathbf{y}^{(M)}$	The length-m_{mul} binary vector $\left(y_1^{(M)}, y_2^{(M)}, \ldots, y_{m_{\text{mul}}}^{(M)}\right)^T$ denoting the outcomes of the Multiplicity Encoder in the absence of noise.
$\mathbf{y}^{(L)}$	The length-m_{loc} binary vector $\left(y_1^{(L)}, y_2^{(L)}, \ldots, y_{m_{\text{loc}}}^{(L)}\right)^T$ denoting the outcomes of the Localization Encoder in the absence of noise.
$\hat{\mathbf{y}}^{(M)}$	The length-m_{mul} binary vector denoting the actually observed noisy outcomes of the Multiplicity Encoder.
$\hat{\mathbf{y}}^{(L)}$	The length-m_{loc} binary vector denoting the actually observed noisy outcomes of the Localization Encoder.

Table 5.2: Table of notation used in GROTESQUE tests

5.4 GROTESQUE Tests

A key component of the algorithms that we present in this work are GROTESQUE Tests (short for **GRO**up **TES**ting **QU**ick and **E**fficient). Given a set $\mathcal{I}' = \{j_1, j_2, \ldots, j_{n'}\} \subseteq \mathcal{I}$ containing n' items, and out of which an unknown number k' of k are defectives, the GROTESQUE tests tell us, with high probability, the number of defective items k' and the location if there is just one. One way of thinking about the GROTESQUE module is that it attempts to solve a "reduced" version of the original "k defectives out of n items" problem, instead solving a "k' defectives out of n' items" problem, where k' and n' are generally both much smaller than k and n, respectively. The input to GROTESQUE tests is a length-

Name of operation	Time complexity
Addition of two bits	1
Comparison of two bits	1
Searching/Insertion/Deletion element in Red-Black trees	log(number of elements in the tree)

Table 5.3: Basic operations and corresponding time complexities

n' vector, $(x_j : j \in \mathcal{I}')$, where x_j is 1 if j is defective, and 0 otherwise. The test outputs are $y_1, y_2, \ldots, y_{m'}$. While test outcomes are positive or negative, for notational convenience we represent the corresponding vector as a length-m binary vector with 1 representing a positive test outcome, and 0 representing a negative test outcome. Each of which are flipped independently by a Binary Symmetric Channel with transition probability q to obtain noisy tests $\hat{y}_1, \ldots \hat{y}_{m'}$. The noisy tests are then processed by the GROTESQUE decoder to output one of the following possibilities:

1. $k' = 0$, *i.e.*, there is no defective.
2. $k' = 1$. In this case, the decoder also outputs the location of the defective in the set $\mathcal{S}$.
3. $k' > 1$, *i.e.*, there are at least two defective items.

GROTESQUE consists of two kinds of tests - *multiplicity tests* and *localization tests*. Multiplicity tests tell us which of the above three possibilities it is, and the localization tests tell us which item is defective if there exists exactly one defective item in the n' items. See Figure 5.1 for the block diagram.

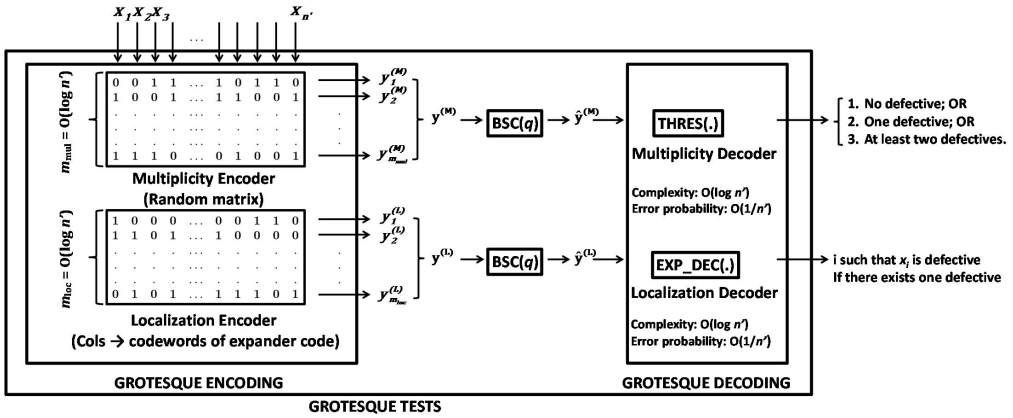


Figure 5.1: The input of a GROTESQUE tests is a set of n' items with unknown number of defectives. There are three possible outputs: there exists no defective item, or there exists exactly one defective item and the corresponding index number, or there exist at least two defective items (but GROTESQUE does not output the corresponding index numbers). There are two parts to GROTESQUE Encoding: the Multiplicity Encoder and the Localization Encoder. For the Multiplicity Encoder, we generate m_{mul} tests. Each item is included in each test uniformly at random. In the Localization Encoder, first an expander code with n' codewords is chosen. Next, the j-th item is included in the i-th test if and only if the i-th bit of the j-th codeword is 1. The inputs to GROTESQUE Decoding are the results of outputs of GROTESQUE Encoding passing through $BSC(q)$. Again, GROTESQUE Decoding is divided into two parts: the Multiplicity Decoder and the Localization Decoder. In the Decoder, we count the number of positive outputs of the Multiplicity Encoder, compare this number with the expected numbers for three cases, and decide on the multiplicity according to the rule given in Equation 5.1. We call the above process $THRES(.)$ (short for *threshold detector*). To implement the Localization Decoder, we use EXP_DEC (short for *expander code decoder*) to do the decoding. If there exists exactly one defective item, the output should be one of the codewords of the expander code. This tells us which item is defective. For example, let $m_{\text{mul}} = \mathcal{O}(\log(n'))$ and $m_{\text{loc}} = \mathcal{O}(\log(n'))$. The overall time complexity and error probability for the GROTESQUE tests are, respectively, $\mathcal{O}(\log(n'))$ and $\mathcal{O}(1/\operatorname{poly}(n'))$.

5.4.1 Multiplicity testing

We generate m_{mul} tests in this part. In each test, the j-th item is included with probability 1/2. If we represent the tests as an $m_{\text{mul}} \times n'$ matrix, $A^{(M)}$, then each entry of the $A^{(M)}$ is a Bernoulli random variable[12] with parameter 1/2. If the entry in the i-th row and the j-th column is 1, it means that the j-th item is included in the i-th test. We count the number of positive outcomes in a multiplicity test and denote it by K. We then use Equation (5.1) to estimate the multiplicity value of the test. The use of Equation (5.1) is justified by the values of Table 5.4, which computes the expected value of the number of positive test outcomes K for both noisy as well as noiseless settings.

k'	Expected value of K (Noiseless tests)	Expected value of K (Noisy tests)
0	0	$q \times m_{\text{mul}}$
1	$1/2 \times m_{\text{mul}}$	$1/2 \times m_{\text{mul}}$
≥ 2	$\geq 3/4 \times m_{\text{mul}}$	$\geq (3/4 - q/2) \times m_{\text{mul}}$

Table 5.4: Expected number of positive test outcomes in the multiplicity tests.

When the tests are noisy (as mentioned before, we assume that the noise follows the output of a BSC(q)), GROTESQUE's multiplicity decoder uses the following rule to estimate the multiplicity (when the tests are noiseless, the same equation with q set to zero may be used):

$$\hat{k'} = \begin{cases} 0, & \text{if } K \in \left[0, \left(\frac{1}{4} + \frac{q}{2}\right) m_{\text{mul}}\right) \\ 1, & \text{if } K \in \left[\left(\frac{1}{4} + \frac{q}{2}\right) m_{\text{mul}}, \left(\frac{5}{8} - \frac{q}{4}\right) m_{\text{mul}}\right) \\ \geq 2, & \text{if } K \in \left[\left(\frac{5}{8} - \frac{q}{4}\right) m_{\text{mul}}, \infty\right). \end{cases} \tag{5.1}$$

[12]One could argue that the idea of using Bernoulli matrices to estimate the number of defective items in a group is very similar to the competitive group testing literature, for instance the work of Damaschke et al [47]. Our need here is for a very crude version of these tests.

5.4.2 Localization

If the multiplicity test estimates k' to be 1, we then use the results of the m_{loc} localization tests (which have been non-adaptively designed *a priori*) to localize the defective item. We represent the tests as an $m_{\text{loc}} \times n'$ matrix, $A^{(L)}$. The difference between $A^{(L)}$ and $A^{(M)}$ is that the columns of $A^{(L)}$ correspond to distinct codewords of an expander code,[13] $\mathcal{C}$ (while the entries of $A^{(M)}$ are chosen uniformly at random). Different columns correspond to different codewords. The j-th item is included in the i-th test if and only if the i-th bit of the j-th codeword is 1. If there is exactly one defective item, then the output of the localization tests should be one of the codewords of $\mathcal{C}$ in the scenario with noiseless tests, or the result of one of the codewords of $\mathcal{C}$ XOR'd with a vector whose entries are i.i.d. Bernoulli(q) random variables in the scenario with noisy tests. By Theorem 14, the Localization step is correct with error probability $\leq 2^{-fm_{\text{loc}}}$ (where f is a constant for the code $\mathcal{C}$) and decoding complexity $\mathcal{O}(m_{\text{loc}})$.

The following theorem about error-exponents of expander codes is useful in our construction.

Theorem 14 ([12, 13]). *For a given rate R, large enough m_{loc}, and any $\varepsilon > 0$, there exists a polynomial-time constructible code $\mathcal{C}$ of length m_{loc} such that $P_e(\mathcal{C}, q) \leq 2^{-m_{\text{loc}} f(R,q)/2}$, where*

$$f(R,\ q) = \max_{R \leq R_0 \leq 1-H(q)} E(R_0, q)(R_0 - R)/2 - \varepsilon.$$

Here $E(R_0, q)$ is the "random coding exponent" and $H(\cdot)$ is the binary entropy function. The decoding complexity[14] *of a sequential implementation*

[13] The idea of using a measurement matrix whose columns are codewords of a linear code is now new, especially in the context of compressive sensing [28, 45]. However, we only use a linear code as a sub-module to identify exactly one defective when multiplicity testing outputs 1.

[14] For details, please refer to [13]. Instead of using big O notation to describe the decoding

of this decoding is $t_{\text{loc}} m_{\text{loc}}$ *for some* $t_{\text{loc}} > 0$ *and* $f(R, q)$ *is positive for all* $0 < q < 1/2$.

5.4.3 Performance Analysis

In this section, we analyze the error probability and time complexity of GROTESQUE tests in terms of m_{mul} and m_{loc}. The actual choices of these parameters are made in Sections 5.5, 5.6 and 5.7, corresponding, respectively, to adaptive, non-adaptive, and two-stage algorithms.

a) *Error probability:*

We now bound from above the probability of error of the multiplicity and localization sub-routines of GROTESQUE testing. In Lemma 8 below, we explicitly derive the dependence of the probability of error of GROTESQUE tests on the value of q. This dependence can be directly translated into a dependence on the probability of error in each of our algorithms, but for ease of presentation we omit this dependence on q outside this lemma (and focus only on the dependence on k and n).

Lemma 8. *The error probability of GROTESQUE multiplicity testing is at most* $2\exp\left(-m_{\text{mul}}(1-2q)^2/32\right)$. *Conditioned on* k' *being correctly identified as* 1, *the error probability of GROTESQUE localization testing is at most* $\exp(-m_{\text{loc}}(1-H(q))^3/256)$ *for large enough* N *and* $m_{\text{loc}} \geq \frac{2\log(n')}{1-H(q)}$.

Proof. Probability of error for multiplicity testing: The outputs of each individual test in the multiplicity testing are i.i.d. Bernoulli random variables, with mean depending on the value of k' and q. There are three possible error events for multiplicity testing.

1. The true value of k' equals 0, but GROTESQUE estimates it to be $\geq$ 1. In this scenario the expected number of positive outcomes is qm_{mul}.

complexity, we say that there exists a positive constant t_{loc} such that the decoding complexity of expander code is $t_{\text{loc}} m_{\text{loc}}$.

To decide on the value of k' (as either 0 or 1) the threshold that the multiplicity tester (given in Equation 5.1 is $\frac{1}{2}\left(q+\frac{1}{2}\right)m_{\text{mul}}$. Hence the multiplicity tester makes an error if the true number of positive outcomes exceeds the expected number by $z_1 = \frac{1}{2}\left(q+\frac{1}{2}\right)m_{\text{mul}} - qm_{\text{mul}} = \frac{m_{\text{mul}}}{2}\left(\frac{1}{2}-q\right)$.

2. The true value of k' equals 1, but GROTESQUE estimates it to be either 0 or at least 2. In this scenario the expected number of positive outcomes is $m_{\text{mul}}/2$. To decide on the value of k' (as either 1 or not) the closest threshold that the multiplicity tester (given in (5.1) is $\left(\frac{5}{8}-\frac{q}{4}\right)m_{\text{mul}}$. Hence the multiplicity tester may make an error if the true number of positive outcomes differs from the expected number by $z_2 = \left(\frac{5}{8}-\frac{q}{4}\right)m_{\text{mul}} - \frac{m_{\text{mul}}}{2} = \frac{m_{\text{mul}}}{4}\left(\frac{1}{2}-q\right)$.

3. The true value of k' is greater than or equal to 2, but GROTESQUE estimates it to be either 0 or 1. In this scenario the expected number of positive outcomes is at least $\left(\frac{3}{4}-\frac{q}{2}\right)m_{\text{mul}}$. To decide on the value of k' (as either ≥ 2 or not) the closest threshold that the multiplicity tester (given in (5.1) is $\left(\frac{5}{8}-\frac{q}{4}\right)m_{\text{mul}}$. Hence the multiplicity tester may make an error if the true number of positive outcomes differs from the expected number by $z_3 = \left(\frac{3}{4}-\frac{q}{2}\right)m_{\text{mul}} - \left(\frac{5}{8}-\frac{q}{4}\right)m_{\text{mul}} = \frac{m_{\text{mul}}}{4}\left(\frac{1}{2}-q\right)$.

We now use the additive form of the Chernoff bound (see Theorem 1 in Appendix), which states that the probability of m_{mul} i.i.d. copies of a binary random variable differing its expected value by more than z is at most $2e^{-2z^2/m_{\text{mul}}}$. Noting that $z_1 > z_2 = z_3$ in the three cases analyzed, we have the desired bound on the probability of error.

Probability of error for localization testing: This error event corresponds to the scenario when there is exactly one defective item and we claim that $k' = 1$, but we locate the wrong defective item. Here we use the exponentially

small upper bound on the probability of error of expander codes, as shown in Theorem 14, to bound our probability of error. The probability of error of the expander code for $m_{\text{loc}} \geq \frac{2\log(n')}{1-H(q)}$ is bounded as

$$\begin{aligned} P_e(\mathcal{C}, q) &\leq 2^{-m_{\text{loc}} f(R,q)/2} \\ &= 2^{-m_{\text{loc}} \max_{R \leq R_0 \leq 1-H(q)} E(R_0,q)(R_0-R)/4-\epsilon} \\ &\leq 2^{-m_{\text{loc}}(1-H(q)-R_0)^2(1-H(q)-R)/8} \\ &= 2^{-m_{\text{loc}}(1-H(q)-R)^3/32} \\ &\leq 2^{-m_{\text{loc}}(1-H(q))^3/256}, \end{aligned} \tag{5.2}$$

where for the middle inequality we choose $R_0 = (R + 1 - H(q))/2$, and the last inequality follows by choosing the rate $R(= \frac{\log(n')}{m_{\text{loc}}})$ of our expander code to be smaller than $(1 - H(q))/2$.

Based on the above analysis, setting $m_{\text{mul}} = c_{\text{mul}} \log(k)$ and $m_{\text{loc}} = c_{\text{loc}} \log(n)$ (recall that k is the total number defective items and n is the total number of items) where c_{mul} and c_{loc} are constants, we can guarantee that the error probability of GROTESQUE test is upper bounded by

$$2\exp\left(\left(-c_{\text{mul}}(1-2q)^2/32\right)\log(k)\right) + \exp\left(\left(-c_{\text{loc}}(1-H(q))^3/256\right)\log(n)\right) \tag{5.3}$$

for $c_{\text{loc}} \geq \frac{2}{1-H(q)}$. Specifically, in the setting $k = \mathcal{O}(n^{1-\Delta})$ for $\Delta > 0$, which is of primary interest to us, if $m_{\text{mul}} + m_{\text{loc}} = C\log(n)$, then the error probability, P_e^{GR}, is at most $3n^{-\frac{C}{g(q)}}$ by setting $m_{\text{mul}} = C\frac{32}{(1-2q)^2 g(q)}\log(n)$ and $m_{\text{loc}} = C\frac{256}{(1-H(q))^3 g(q)}\log(n)$. Here, $C \geq \frac{(1-H(q))^2 g(q)}{128}$ and $g(q) = \frac{32}{(1-2q)^2} + \frac{256}{(1-H(q))^3}$. ■

b) Decoding complexity:

Multiplicity testing involves counting the total number of positives from m_{mul} tests. The complexity of counting is m_{mul}. We also need to compare

the total number of positives to $(\frac{1}{4}+\frac{q}{2})m_{\text{mul}}$ and $(\frac{5}{8}-\frac{q}{4})m_{\text{mul}}$ to estimate the value of k'. The complexity of these two comparisons is $2\log(m_{\text{mul}})$.

Localization testing involves decoding an expander code of block-length m_{loc}, which is $t_{\text{loc}}m_{\text{loc}}$ by Theorem 14.

Algorithm 2 Grotesque Decoding ($\hat{\mathbf{y}}^{(M)}$,$\hat{\mathbf{y}}^{(L)}$)

do Multiplicity Decoding
if $\hat{k'} == 1$ **then**
    do Localization Decoding
else
    return $\varnothing$
end if

Algorithm 3 Multiplicity Decoding ($\hat{\mathbf{y}}^{(M)}$)

count the number of 1's in $\hat{\mathbf{y}}^{(M)}$, K
if $0 \leq K < \left(\frac{1}{4}+\frac{q}{2}\right) m_{\text{mul}}$ **then**
    $\hat{k'} \leftarrow 0$
else if $K > \left(\frac{5}{8}-\frac{q}{4}\right) m_{\text{mul}}$ **then**
    $\hat{k'} \leftarrow 2$ // Could be 2 or more than 2 defectives in this pool. However, it does not matter since in this event we never use these pools further - the value 2 is merely a placeholder.
else
    $\hat{k'} \leftarrow 1$
end if
return $\hat{k'}$

5.5 Adaptive Group Testing

In this section, we consider the *adaptive group testing* problem. The objective here is to determine an unknown set $\mathcal{S}$ of k defective items from

Algorithm 4 Localization Decoding ($\hat{\mathbf{y}}^{(L)}$, $A^{(L)}$)

1: $\hat{\mathcal{C}} \leftarrow$expander code decoding given $\hat{\mathbf{y}}^{(L)}$ as an input

2: $j \leftarrow$ index of the column of $A^{(L)}$ such that the column $= \hat{\mathcal{C}}$

3: **return** j

Adaptive Algorithm	
$\mathcal{I}'$	Left subset, $\mathcal{I}' \subseteq \mathcal{I}$
$\deg_{\mathcal{I}'}(i)$	The number of neighbors in left subsets $\mathcal{I}'$
$\mathcal{S}$-zero nodes	$\{$right nodes i: $\deg_{\mathcal{S}}(i) = 0\}$
$\mathcal{S}$-leaf nodes	$\{$right nodes i: $\deg_{\mathcal{S}}(i) = 1\}$
$\mathcal{S}$-non-leaf nodes	$\{$right nodes i: $\deg_{\mathcal{S}}(i) \geq 2\}$
T	The total number of stages
$\Lambda^{(t)}$	The set of unrecovered defectives before the t-th stage tests are performed, $\Lambda^{(1)} = \mathcal{S}$, $t \in \{1, \ldots, T\}$
$k^{(t)}$	The number of unrecovered defectives before the t-th stage tests are performed, $k^{(t)} = \lvert\Lambda^{(t)}\rvert$, $t \in \{1, \ldots, T\}$
$\mathcal{G}^{(t)}$	The bipartite graph for the t-th stage, $t \in \{1, \ldots, T\}$
$r^{(t)}$	The number of defectives recovered in t-th stage, $r^{(t)} = k^{(t)} - k^{(t+1)}$, $t \in \{1, \ldots, T-1\}$
$c_{\text{adp,rn}}$	a constant related to the number of right nodes for the bipartite graph in the coupon collection stage
$c_{\text{adp,deg}}$	a constant related to the degree of right nodes for the bipartite graph in the coupon collection stage

Table 5.5: Table of notation used in our adaptive algorithm

a collection $\mathcal{I}$ of size n. In this setting, we are allowed to perform tests sequentially in an adaptive manner, *i.e.*, the subset tested in each test may depend on the outcome of previous tests. As stated earlier, we assume that each test outcome may be incorrect independently with probability q.

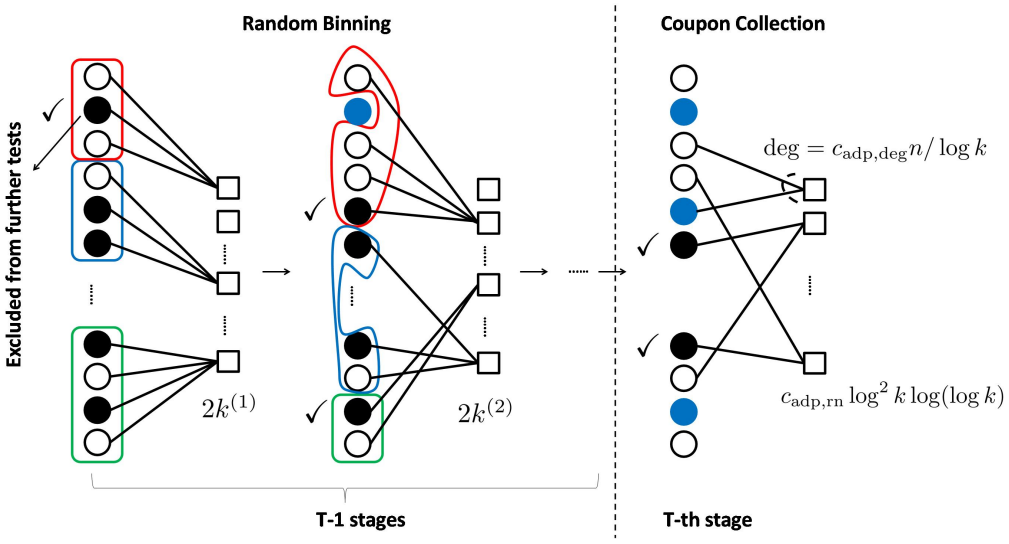


Figure 5.2: In each t-th stage ($t \leq T-1$), we generate a bipartite graph with n nodes on the left representing n items and $2k^{(t)}$ nodes on the right. The black circular nodes represent defective items and the white ones represent non-defective items. Each bipartite graph is left-regular with left-degree equal to 1. The item/left nodes of each such graph are partitioned randomly – different coloured collections show different "pools" within a partition. Nodes in the same pool connect to the same testing/right node. Each testing node passes the items connected to it to the GROTESQUE tests. If there exists only one defective in one pool, then GROTESQUE locates the defective item with high probability. For example, the second node in the red partition of first graph will be detected by the GROTESQUE tests at the first testing node with high probability. In the next iteration, we exclude decoded defective items (colored blue) and use a similar decoding process. Finally, in the T-th stage, we generate a bipartite graph with $c_{\text{adp,rn}}(\log^2(k))(\log(\log(k)))$ right nodes. For each testing node, we pick its $c_{\text{adp,deg}} n/\log(k)$ neighbouring item nodes by choosing uniformly from all item nodes with replacement. By the coupon collection argument, with high probability, we can decode the remaining undecoded defective items.

5.5.1 Overview

Our algorithm has $\mathcal{O}(\log(k))$ adaptive stages. In all except the last stage, we design our tests so that with a high probability, in each stage, we recover a constant fraction of the remaining defectives. In each of these stages, the number of tests is roughly proportional to the number of remaining defectives. Thus, the total number of tests is $\mathcal{O}(k\log(n))$. In each of these stages, the tests are designed by first removing the defectives recovered so far, partitioning the set of remaining items into twice as many sets as the

number of remaining defectives, performing GROTESQUE tests on each set from the partition. In our analysis, to guarantee that in each stage we recover a constant fraction of the remaining defectives with a high enough probability, we apply concentration inequalities. This works only when the number of unrecovered defectives be at least $\Omega(\log(k))$. Thus, we move on to the last stage when all but $\log(k)$ defectives have been recovered.

In the last stage, we use the coupon collection problem [103] as a primitive, to identify the remaining defectives. First we remove the set of defectives already recovered from the set of items being tested. Next, we design $\mathcal{O}(\log(k)\log(\log(k)))$ non-adaptive tests by picking random subsets of an appropriate size and performing GROTESQUE tests for that subset. We view the collection of outcomes from these sets of GROTESQUE tests as a random process that generates one independent "coupon" with constant probability each time. Thus, $\mathcal{O}(\log(k)\log(\log(k)))$ such tests suffice by standard coupon collector arguments. See Figure 5.2 for illustration.

Overall, our algorithm requires $\mathcal{O}(k\log(n))$ tests and runs in $\mathcal{O}(k\log(n))$ steps.

5.5.2 Formal Description

Let $\Lambda^{(t)}$, and $k^{(t)}$, respectively, be the set, and the number of unrecovered defectives before the t-th stage tests are performed. Note that $\mathcal{S} = \Lambda^{(1)} \supseteq \Lambda^{(2)} \ldots$ and $k = k^{(1)} \geq k^{(2)} \geq \ldots$ since we recover some defectives in each stage. Let $T = \inf\{t : k^{(t)} \leq \log(D)\}$ denote the number of stages after which (with high probability) the number of unrecovered defectives is no larger than $\log(k)$. For any testing node i, and subset of item nodes $\mathcal{S} \subseteq \mathcal{I}$ we define $\deg_{\mathcal{S}}(i)$ as the number of item nodes in $\mathcal{S}$ that neighbor the testing node i. There are three types of testing nodes: *$\mathcal{S}$-leaf nodes*, *$\mathcal{S}$-zero nodes* and *$\mathcal{S}$-non-leaf nodes*. A $\mathcal{S}$-leaf node is a testing node i with

$\deg_{\mathcal{S}}(i) = 1$ (it is connected to a single defective item), a $\mathcal{S}$-zero node is a testing node i with $\deg_{\mathcal{S}}(i) = 0$ (it is connected to no defective items), and a $\mathcal{S}$-non-leaf node is a testing node i with $\deg_{\mathcal{S}}(i) \geq 2$ (it is connected to multiple defective items). Specifically, $\mathcal{S}$-leaf nodes are very helpful in our quick decoding process since our GROTESQUE black-box shall with high probability correctly output the location of a defective item if there exists exactly one defective item among its neighbours.

Note: Even though in all our algorithms, both $\mathcal{S}$-zero nodes and $\mathcal{S}$-non-leaf nodes contain "potentially useful" information, we do not use them for decoding. This is because we require our algorithms to work with computational complexity that is comparable (up to constant factors for the adaptive algorithm, and at most a logarithmic factor in the other algorithms) to the size of the output of our algorithm (that is, $\mathcal{O}(k \log(n))$). To run this "blindingly fast", we need our algorithms to output "something interesting" in "very little" time. So, for instance, while the $\mathcal{S}$-zero nodes tell us which inputs are non-defective, using this information takes "too long" (since it essentially tells us which items are *not interesting* rather than those which *are* interesting, but there are many more non-interesting items than interesting ones).

a) Test design:

Conceptually, we design the first $T-1$ stages of our adaptive algorithm using the idea of "random binning", and the T-th stage using that of "coupon collection".

- ***Random binning***: For each stage $t = 1, \ldots, T-1$, we consider a random left regular bipartite graph $\mathcal{G}^{(t)}$ with $2k^{(t)}$ testing nodes and $n - (k - k^{(t)})$ item nodes corresponding to the set $(\mathcal{I} \setminus S) \cup \Lambda^{(t)}$, *i.e.,* all items except those already recovered in the previous $t-1$ stages. We let each item node of $\mathcal{G}^{(t)}$ be of degree one. We choose this

graph uniformly at random from all possible bipartite graphs having mentioned properties above. Next, for each testing node of $\mathcal{G}^{(t)}$, we use its set of left neighbours as the input for a GROTESQUE test. In each stage, for each testing node of $\mathcal{G}^{(t)}$, if GROTESQUE detects it has multiplicity one, it decodes the corresponding defective item. Else if GROTESQUE identifies a right-node as having multiplicity greater than one or zero, it does not further use these test outcomes.

- ***Coupon collection***: In the final stage, we consider a left regular bipartite graph $\mathcal{G}^{(T)}$ with item node set $(\mathcal{I} \setminus \mathcal{S}) \cup \Lambda^{(T)}$ and $c_{\text{adp,rn}}(\log(k))^2 \log(\log(k))$ testing nodes. For each testing node, we choose its $c_{\text{adp,deg}} n / \log(k)$ neighbours independently and uniformly at random with replacement. Next, we design $\mathcal{O}(\log(n))$ GROTESQUE tests at each testing node to test for defectives among its $\mathcal{O}(n)$ left neighbours.

b) Decoding algorithm:

The decoding for each stage corresponds to detection of leaf nodes in that stage and corresponding localization via GROTESQUE tests. Specifically, in each of the t-th stages for $t \leq T$, we sequentially pass the outputs of each of the testing nodes of $\mathcal{G}^{(t)}$ to GROTESQUE, which identifies leaf nodes and localized the corresponding defective items. Note that the structure of $\mathcal{G}^{(T)}$ at the T-stage differs from the structure of $\mathcal{G}^{(t)}$ for $t < T$, since they are chosen according to different processes (coupon collection versus random binning). Nonetheless, the same decoding procedure (leaf detection and corresponding localization of defectives) is performed in both the first $T-1$ stages and the T-th stage. The algorithm makes an error if not all defective items have been localized by the T-th stage (one can test for this by passing the set of remaining items to GROTESQUE's multiplicity

testing subroutine).

Algorithm 5 Adaptive Group Testing (k, n)

$t \leftarrow 1$
$\hat{\mathcal{S}} \leftarrow \varnothing$ // Initial list of defectives
$k^{(t)} \leftarrow k$ // Number of defectives still to be identified
// Random binning stages
while $k^{(t)} > \log(k)$ **do**
  for each testing node i of $\mathcal{G}^{(t)}$ **do**
    do Grotesque Decoding // which returns j (if possible) as the item corresponding to the testing node i
    $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \bigcup \{j\}$ // Add j to list of defective items
  end for
  $t \leftarrow t + 1$
end while
// Coupon collection stage
for each testing node i of $\mathcal{G}^{(T)}$ **do**
  do Grotesque Decoding // which returns j (if possible) as the item corresponding to the testing node i
  $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \bigcup \{j\}$ // Add j to list of defective items
end for
return $\hat{\mathcal{S}}$

5.5.3 Performance Analysis

To analyze the performance of the first part of the algorithm, we require the following lemma.

Lemma 9. *Let $\mathcal{G}$ be a random bipartite graph with n' nodes on the left, and $2k'$ nodes on the right side. Let Λ be a subset of the item nodes of size k'. Also, let each node on the left side of $\mathcal{G}$ have degree one. Then, for any*

$\epsilon > 0$, with probability at least $1 - \exp(-\epsilon^2 k'/2)$, at least $(e^{-1/2} - \epsilon)k'$ nodes on the left are connected to Λ-leaf nodes.

Proof. We first note that the probability that a node $j \in \Lambda$ on the left of the bipartite graph is connected to a Λ-leaf node is:

$$\begin{aligned} \Pr(j \text{ is connected to a } \Lambda\text{-leaf}) &= \left(\frac{2k'-1}{2k'}\right)^{k'-1} \\ &> \left(1 - \frac{1}{2k'}\right)^{(2k'-1)/2} > e^{-1/2}, \end{aligned}$$

where the last inequality comes from the well-known inequality: $\left(1 - \frac{1}{x}\right)^{x-1} > e^{-1}$. Therefore the expected number of nodes from Λ that are connected to Λ-leaf nodes is at least $e^{-1/2}k'$. Next, we show a concentration result for this number by applying McDiarmid's inequality [98] the statement of which is reprised in Appendix A as follows. First, we label nodes on the right with numbers from 1 to $2k'$ and for each $i \in 1, 2, \ldots, k'$, let

$\mathbf{X}_i \triangleq$ label of the node on the right which is connected to the i-th node in Λ.

Also, define $f : \prod_{i=1}^{k'} \{1, 2, \ldots, 2k'\} \to \mathbb{Z}$ as the number of item nodes connected to a Λ-leaf node *i.e.*,

$$f(x_1, x_2, \cdots, x_{k'}) \triangleq |\{x_1, x_2, \cdots, x_{k'}\}|. \tag{5.4}$$

It is observed that for any fixed $x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_{k'}$ and any $x_i, x_i' \in \mathbf{X}_i$,

$$|f(x_1, \cdots, x_i, \cdots, x_{k'}) - f(x_1, \cdots, x_i', \cdots, x_{k'})| \leq 2.$$

For example, $\exists i, j(\neq i)$, $x_i = x_j$ and $x_i' \neq x_i$. It is possible that x_j-th and x_i'-th testing nodes which initially are not Λ-leaf nodes become Λ-leaf nodes. Then, $f(x_1, \cdots, x_i, \cdots, x_{k'}) - f(x_1, \cdots, x_i', \cdots, x_{k'}) = -2$. In fact, we can numerate all possible cases to conclude out result.

It is clear that $\mathbf{X}_i$'s are independent. Therefore, by McDiarmid's inequality [98] we have:

$$\Pr\left(f(\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_{k'}) - \mathbf{E}f(\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_{k'}) < -\epsilon k'\right) \leq \exp\left(-\epsilon^2 k'/2\right).$$

Hence,

$$\begin{aligned}
&\Pr\left(f(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{k'}) - e^{-1/2}k' < -\epsilon k'\right) \\
&\leq \Pr\left(f(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{k'}) - \mathbf{E}f(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{k'}) < -\epsilon k'\right) \\
&\leq \exp\left(-\epsilon^2 k'/2\right).
\end{aligned}$$

■

Corollary 3. *With probability at least* $1 - \left[\frac{\log(k/\log(k))}{-\log(1-e^{-1/2}/2)} k^{-1/(4e)} + 4e^{1/2} k P_e^{GR}\right]$, *in each of the first* $T-1$ *stages, our decoding algorithm recovers no smaller than a* $e^{-1/2}/2$ *fraction of defectives that have not been decoded up to that stage.*

Proof. The event that we recover fewer than $e^{-1/2}/2$ fraction of defectives in t-th stage is a subset of the union of the following two events:

(1) Less than a $e^{-1/2}/2$ fraction of defectives are connected to a $\Lambda^{(t)}$-leaf node.

(2) The outcome of the GROTESQUE tests is incorrect for any of the $2k^{(t)}$ testing nodes.

By Lemma 9, with $\Lambda = \Lambda^{(t)}$, $n' = n - k^{(t)}$, and $\epsilon = e^{-1/2}/2$, the probability of event (1) is at most $\exp(-k^{(t)}/4e)$. Further, by Lemma 8 and Union bound, with $n' = n - k^{(t)}$, the probability of event (2) is upper bounded by $2k^{(t)}P_e^{GR}$. For each t, let $r^{(t)} = k^{(t)} - k^{(t+1)}$. Therefore, the probability that in one of the $T-1$ stages, fewer than a $e^{-1/2}/2$ fraction of defectives are correctly decoded is bounded from above as

$$\Pr\left(\bigcup_{t=1}^{T-1}\{r^{(t)} < e^{-1/2}k^{(t)}/2\}\right) = \sum_{t=1}^{T-1}\Pr\left(r^{(t)} < \frac{e^{-\frac{1}{2}}k^{(t)}}{2}\,\middle|\,\bigcap_{\tau=1}^{t-1}\left\{r^{(\tau)} > \frac{e^{-\frac{1}{2}}k^{(\tau)}}{2}\right\}\right).$$

Under the conditioning event for the t-th term, we may bound $d^{(t)}$ from above by $k(1-e^{-1/2}/2)^{t-1}$. Further, by the definition of T, there are at most $-\log\left(k/\log(k)\right)/\log\left(1-e^{-1/2}/2\right)$ terms. The chain of inequalities is further simplified as

$$\begin{aligned}
&\Pr\left(\bigcup_{t=1}^{T-1}\{r^{(t)} < e^{-1/2}k^{(t)}/2\}\right)\\
&\leq \sum_{t=1}^{-\log\left(k/\log(k)\right)/\log\left(1-e^{-1/2}/2\right)}\left[\exp\left(-k\left(1-e^{-1/2}/2\right)^{t-1}/4e\right) + 2k^{(t)}P_e^{\mathrm{GR}}\right]\\
&\leq \frac{\log\left(k/\log(k)\right)}{-\log\left(1-e^{-1/2}/2\right)}\exp\left(-\frac{k}{4e}\left(1-e^{-1/2}/2\right)^{\frac{\log\left(k/\log(k)\right)}{-\log\left(1-e^{-1/2}/2\right)}}\right)\\
&\quad + \sum_{t=1}^{\infty}2k(1-e^{-1/2}/2)^{t-1}P_e^{\mathrm{GR}}\\
&= \frac{\log\left(k/\log(k)\right)}{-\log\left(1-e^{-1/2}/2\right)}k^{-1/(4e)} + 4e^{1/2}kP_e^{\mathrm{GR}}.
\end{aligned}$$

■

a) Number of tests:

In the Random Binning part of the algorithm, there are $2k^{(t)}$ testing nodes in the t-th stage, each of which requires $2Ck^{(t)}\log(n)$ tests for some constant $C = C(q)$ (as determined in Section 5.4). Hence the total number of tests for the Random Binning part of the algorithm is $\sum_{t=1}^{T-1}2Ck^{(t)}\log(n)$. By Corollary 3, with high probability, in each stage the algorithms recovers a constant fraction of the undecoded defectives. Thus, $k^{(t)} \leq (1-e^{-1/2}/2)^{t-1}k$ for $t \in [0, T-1]$. Therefore, the total number of tests required in the first $T-1$ stages is at most $4e^{1/2}Ck\log(n)$.

For the Coupon Collection stage, the number of testing nodes is $c_{\mathrm{adp,rn}}(\log(k))^2\log(\log(k))$. Since we perform GROTESQUE tests for each

testing node, the total number of tests required is $c_{\text{adp,rn}}C(\log(k))^2\log(\log(k))\log(n)$, which is less than the $\mathcal{O}(k\log(n))$ tests required in the Random Binning part of the algorithm.

Thus our proposed scheme requires at most $4e^{1/2}Ck\log n + c_{\text{adp,rn}}C(\log(k))^2\log(\log(k))\log(n)$ tests overall.

b) Decoding complexity:

Since in each stage our decoding algorithm has to step through all testing nodes to decode the corresponding GROTESQUE tests, the total number of testing nodes the algorithm needs to consider is $\sum_{t=1}^{T-1} 2k^{(t)} + c_{\text{adp,rn}}(\log(k))^2\log(\log(k)) = 4e^{1/2}k + c_{\text{adp,rn}}(\log(k))^2\log(\log(k))$. By the analysis from Section 5.4.3, decoding each collection of GROTESQUE tests at a testing node takes

$$\frac{32C}{(1-2q)^2g(q)}\log(n)+2\log\left[\frac{32C}{(1-2q)^2g(q)}\log(n)\right]+t_{\text{loc}}\frac{256C}{(1-H(q))^3g(q)}\log(n)$$

time. Therefore, our algorithm runs in $[4e^{1/2}k+c_{\text{adp,rn}}(\log(k))^2\log(\log(k))]\left[\frac{32C}{(1-2q)^2g(q)}\log(n)+2\log\left(\frac{32C}{(1-2q)^2g(q)}\log(n)\right)+t_{\text{loc}}\frac{256C}{(1-H(q))^3g(q)}\log(n)\right]$ time.

c) Error probability:

We show that the above algorithm succeeds with high probability in decoding all the defectives in the claimed number of stages.

By the analysis of error events (1) and (2) in Corollary 3, in the first part of the algorithm (random binning) we recover at least $k-\log(k)$ defectives with probability $1-\left[\frac{\log(k/\log(k))}{-\log(1-e^{-1/2}/2)}k^{-1/(4e)}+4e^{1/2}kP_e^{\text{GR}}\right]$.

To analyze the last (coupon collection) stage of tests, note that an error may occur only if one of the following events occur,

(3) In the coupon collection process, fewer than $\log(k)$ distinct coupons are collected.

(4) At least one of the collected coupons was incorrectly decoded.

To bound the probability of event (3), we apply standard concentration bounds on the coupon collector's problem [103]. Towards this end, we first note that the probability that our algorithm identifies a testing node i as a leaf node, P_{leaf}, may be written as

$$
\begin{aligned}
&\Pr(i \text{ decoded as a } \Lambda^{(T)}\text{-leaf node}) \\
&\quad \geq \Pr(i \text{ decoded as a } \Lambda^{(T)}\text{-leaf node}, i \text{ is a } \Lambda^{(T)}\text{-leaf node}) \\
&\quad = \Pr(i \text{ decoded as a } \Lambda^{(T)}\text{-leaf node} \mid i \text{ is a } \Lambda^{(T)}\text{-leaf node}) \times \\
&\qquad \Pr(i \text{ is a } \Lambda^{(T)}\text{-leaf node}) \\
&\quad \geq (1 - P_e^{\text{GR}}) \times \Pr(i \text{ is a } \Lambda^{(T)}\text{-leaf node}) \\
&\quad = (1 - P_e^{\text{GR}}) \times \frac{c_{\text{adp,deg}} n}{\log(k)} \frac{\log(k)}{n - k + \log(k)} \left(1 - \frac{\log(k)}{n - k + \log(k)}\right)^{\frac{c_{\text{adp,deg}} n}{\log(k)}} \\
&\quad \to c_{\text{adp,deg}} e^{-c_{\text{adp,deg}}}, as\ n \to \infty.
\end{aligned}
$$

Since each decoded leaf node is independently chosen and the probability of picking a coupon is constant, by tail bounds on the coupon collection process [103], the probability that at least one coupon has not been collected in $c_{\text{adp,rn}}(\log(k))^2 \log(\log(k))$ steps is $(\log(k))^{1 - c_{\text{adp,rn}} P_{\text{leaf}} \log(k)}$.

Thus, the overall error probability decays is at most $\frac{\log(k/\log(k))}{-\log(1 - e^{-1/2}/2)} k^{-1/(4e)} + 4e^{1/2} k P_e^{\text{GR}} + (\log(k))^{1 - c_{\text{adp,rn}} P_{\text{leaf}} \log(k)}$.

5.6 Non-adaptive Group Testing

We consider non-adaptive group testing in this section. In non-adaptive group testing, the set of items being tested in each test is required to be independent of the outcome of every other test [56].

The objective here is to determine an unknown set $\mathcal{S}$ of k defective items from a collection $\mathcal{I}$ of size n. We assume that each test outcome may be incorrect independently with probability q.

Non-adaptive Algorithm	
$\mathcal{G}_g$	g-th sub bipartite graph, $g \in \{1, \ldots, c_{\text{non,bpt}} \log(D)\}$
$\hat{\mathbf{y}}_i$	The length-$\mathcal{O}(\log(N))$ binary vector denoting the actually observed noisy outcomes of the i-th GROTESQUE Encoder
$c_{\text{non,bpt}}$	A constant related to the number of sub bipartite graphs
$c_{\text{non,rn}}$	A constant related to the number of testing/right nodes for each bipartite graph
P_0	The probability that the neighbor of a defective item is a $\mathcal{S}$-leaf node
$\mathcal{L}(\mathcal{S})$	Leaf node list which contains all the $\mathcal{S}$-leaf nodes
$\mathcal{L}(t)$	Leaf node list for the t-th iteration.

Table 5.6: Table of notation used in our Non-adaptive algorithm

5.6.1 Overview

The structure of group-testing tests is based on left-regular bipartite graphs $\{\mathcal{G}_g\}$. We put n items on the left-hand side and $c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k)$ nodes on the right-hand side of a bipartite graph. Each node on the right-hand side of a bipartite graph is called a *group-testing* node. Group tests corresponding to the multiplicity and localization tests of GROTESQUE are performed as part of the encoding process, but the results are not necessarily used to decode. This is because our decoding algorithm can cherry-pick the group-testing nodes to decode only the "useful" ones.

The set of $\mathcal{S}$-leaf nodes (see the definition in Section 5.5.2) are helpful for our decoding process since GROTESQUE tests performed on a testing node output the location of a defective item only if there exists exactly one defective item among its neighbours. In our iterative algorithm, we claim that there exists at least one $\mathcal{S}$-leaf node in each iteration, so that we can decode one defective item.

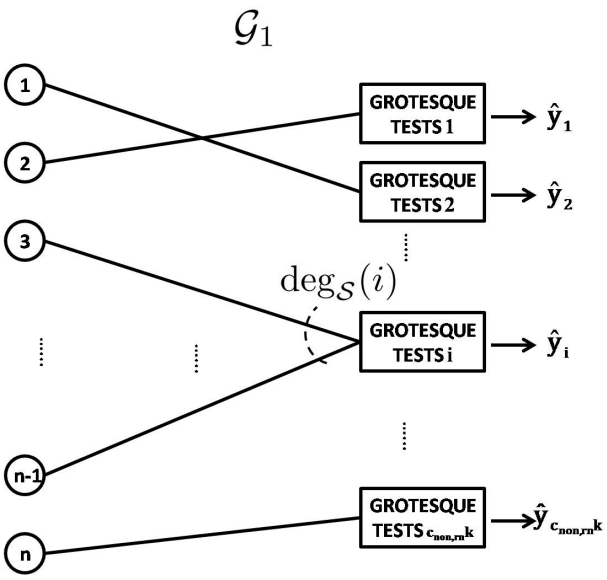


Figure 5.3: We generate $c_{\text{non,bpt}} \log(k)$ bipartite graphs with n nodes on the left (representing n items) and $c_{\text{non,rn}}k$ nodes on the right. The total number of multiplicity and localization tests done for each testing node is $\mathcal{O}(\log(n))$. Take $\mathcal{G}_1$ as an example. For each testing node i, we generate $\mathcal{O}(\log(n))$ tests, $\hat{\mathbf{y}}_\mathbf{i}$, by GROTESQUE tests. The input of the i-th GROTESQUE TESTS are the items connected to the testing node i. The size of outputs for each GROTESQUE encoding is $\mathcal{O}(\log(n))$. Based on the properties of GROTESQUE TESTS, we can estimate whether there exists exactly one defective item and if so, we can estimate its location with high probability.

5.6.2 Formal Description

In this section, we describe a probabilistic construction of the tests and an iterative Non-Adaptive Group Testing algorithm.

a) Description of graph properties:

We first construct a bipartite graph $\mathcal{G}$ (Figure 5.4) with some desirable properties outlined below. We then show that such the random graphs we choose satisfy such properties with high probability. In Section V-Bb), we then use these graph properties in the Non-adaptive Group Testing algorithm.

Properties of $\mathcal{G}$:

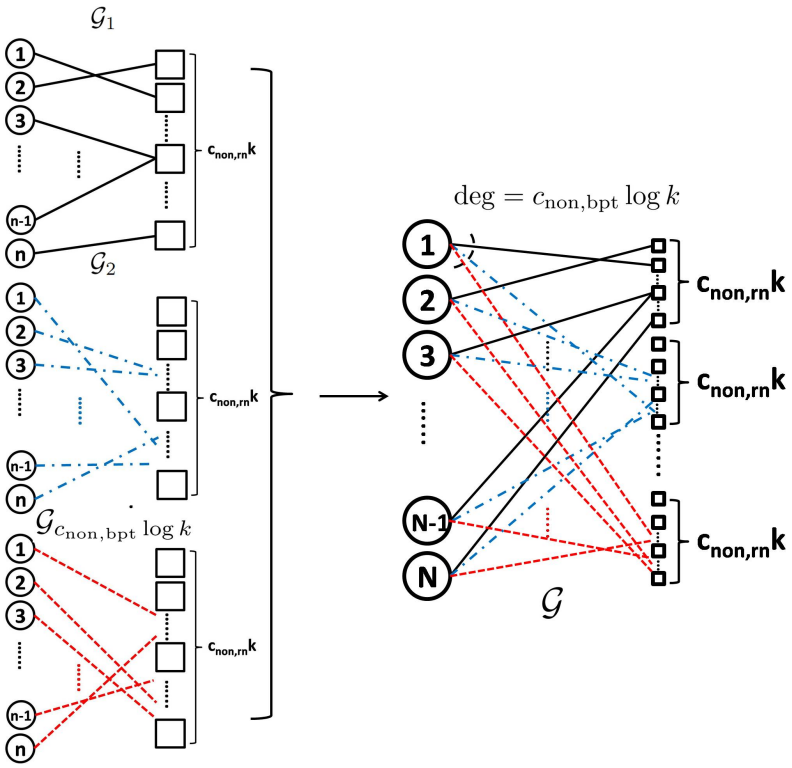


Figure 5.4: After generating $c_{\text{non,bpt}} \log(k)$ bipartite graphs as in Figure 5.3, we combine them to obtain the overall graph $\mathcal{G}$ for our non-adaptive group testing algorithm. There are n nodes on the left representing n items. Collect the testing nodes of all the $c_{\text{non,bpt}} \log(k)$ bipartite graphs on the right side of $\mathcal{G}$ and maintain the connectivity between two sides. Different colors show the connectivity between N items and different right node sets of equal size $c_{\text{non,rn}}k$. Finally, we get a bipartite graph $\mathcal{G}$ with left regularity equal to $c_{\text{non,bpt}} \log(k)$. We can guarantee that, with high probability, each defective item connects to a $\mathcal{S}$-leaf node.

1. *Construction of a left-regular bipartite graph:* As in Figure 5.4, we generate $\mathcal{G}$ by combining $c_{\text{non,bpt}} \log(D)$ left-regular graphs $\mathcal{G}_g$, for $g = 1, \ldots, c_{\text{non,bpt}} \log(k)$. For each $\mathcal{G}_g$, there are n items on the left and $c_{\text{non,rn}}k$ group-testing nodes on the right. The graph $\mathcal{G}_g$ has left-regularity equal to 1 and each edge connects to a testing node uniformly at random. After constructing all the $c_{\text{non,bpt}} \log(k)$ graphs $\mathcal{G}_g$, we combine them to form $\mathcal{G}$ in the following way. Keep n items on the left, and collect the testing nodes. Therefore, $\mathcal{G}$ has the properties that it has n nodes on the left with left-regularity equal to

$c_{\text{non,bpt}} \log(k)$, and $c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k)$ nodes on the right.

2. *"Many" $\mathcal{S}$-leaf nodes:* For any set $\mathcal{S}$ of size k on the left of $\mathcal{G}$, none of the nodes in $\mathcal{S}$ has fewer than a constant fraction of $\mathcal{S}$-leaf nodes. The proof of this statement is the subject of Lemma 10.

Lemma 10. *With probability* $1 - k^{1-\exp(-1/c_{\text{non,rn}})c_{\text{non,bpt}}/8}$, *the fraction*[15] *of* $c_{\text{non,bpt}} \log(k)$ *neighbors of each defective item that are* $\mathcal{S}$*-leaf nodes is at least* $\frac{1}{2} e^{-\frac{1}{c_{\text{non,rn}}}}$.

Proof. Define $\mathbf{W}_{i,j}$ as the random variable representing whether the neighbor of a defective item x_j is a $\mathcal{S}$-leaf node for the i-th graph $\mathcal{G}_i$.

$$\mathbf{W}_{i,j} = \begin{cases} 1, & \text{if the neighbor item node } j \text{ is a } \mathcal{S}\text{-leaf node} \\ 0, & \text{otherwise.} \end{cases}$$

Then, the total number of $\mathcal{S}$-leaf nodes of x_j is $\sum_{i=1}^{c_{\text{non,bpt}} \log(k)} \mathbf{W}_{i,j}$.

$$\begin{aligned} P_0 &\triangleq \Pr(\text{the neighbor of a defective} \\ &\quad \text{item } x_j \text{ is a } \mathcal{S}\text{-leaf node}) \\ &= \left(1 - \frac{1}{c_{\text{non,rn}} k}\right)^{k-1} \\ &\to \exp\left(-\frac{1}{c_{\text{non,rn}}}\right), as\ k \to \infty, \end{aligned}$$

and $\mathbf{W}_{1,j}, \ldots, \mathbf{W}_{c_{\text{non,bpt}} \log(k),j}$ are i.i.d. Bernoulli random variables with parameter P_0.

Therefore, by the Chernoff bound, we have

$$\begin{aligned} &\Pr\left(\sum_{i=1}^{c_{\text{non,bpt}} \log(k)} \mathbf{W}_{i,j} - c_{\text{non,bpt}} P_0 \log(k) \le -\frac{1}{2} c_{\text{non,bpt}} P_0 \log(k)\right) \\ \le\ &\exp\left(-\frac{1}{8} c_{\text{non,bpt}} P_0 \log(k)\right) \end{aligned}$$

[15] In fact, one neighbor is $\mathcal{S}$-leaf node is sufficient. Here, we prove a stronger result.

Hence, the probability for each defective item that it has at least a constant fraction of $\mathcal{S}$-leaf nodes is $1 - k^{-\exp(-1/c_{\text{non,rn}})c_{\text{non,bpt}}/8}$.

Then, by the union bound, all defective items have at least constant fraction of $\mathcal{S}$-leaf nodes with probability at least $1-k^{1-\exp(-1/c_{\text{non,rn}})c_{\text{non,bpt}}/8}$.

■

In the following two sections, we describe how to use the properties of $\mathcal{G}$ to perform the encoding and decoding.

b) Test design:

For each node i on the right of $\mathcal{G}$, we design GROTESQUE tests with $\deg_{\mathcal{I}}(i)$ inputs which are the items connected to testing node i. We choose the number of multiplicity tests, m_{mul}, to be $c_{\text{mul}} \log(k)$ and the number of localization tests, m_{loc} to be $c_{\text{loc}} \log(n)$. Therefore, the overall number of tests $c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k)(c_{\text{mul}} \log(k) + c_{\text{loc}} \log(n))$.

c) Decoding algorithm:

Before the iterative decoding process, we make a *leaf node list*, $\mathcal{L}(\mathcal{S})$, which contains all the $\mathcal{S}$-leaf nodes based on the multiplicity testing part of GROTESQUE tests. Based on the properties of graph $\mathcal{G}$, we know that each defective item has at least a constant fraction of $\mathcal{S}$-leaf nodes. Denote $\mathcal{L}(t)$ as the leaf node list in t-th iteration, $t = 1, 2, \ldots, k, k+1$. $\mathcal{L}(1) = \mathcal{L}(\mathcal{S})$, $\mathcal{L}(k+1) = \varnothing$ and $\mathcal{L}(t) \neq \varnothing$, for $t = 1, 2, \ldots, k$. In the t-th iteration, we pick a testing node $i \in \mathcal{L}(t)$ and decode a defective item using the localization part of GROTESQUE tests of i to locate the corresponding defective item. After that, we cancel the defective item, its corresponding edges and its neighbors. If the removed neighbor is a $\mathcal{S}$-leaf node, it is also removed from $\mathcal{L}(t)$. We update the $\mathcal{L}(t)$ to $\mathcal{L}(t+1)$. In the $(t+1)$-th iteration, we pick another $\mathcal{S}$-leaf node in $\mathcal{L}(t+1)$. The formal description of the non-adaptive group testing algorithm is as follows:

1. *Initialization:* Go through all the testing nodes and use *only* the mul-

tiplicity testing part of GROTESQUE-test to initialize $\mathcal{L}(1) = \mathcal{L}(\mathcal{S})$.

2. *Operations in the t-th iteration:*

   i) Pick any testing node j in $\mathcal{L}(t)$;

   ii) Use localization part of GROTESQUE-test to decode the corresponding defective;

   iii) Remove the decoded defective item, all the edges connected to it, and all its neighbours;

   iv) Update $\mathcal{L}(t)$ to $\mathcal{L}(t+1)$ by removing the leaf nodes removed in step iii) and return to step i).

3. *Termination:* The algorithm stops when the leaf node list becomes empty, and outputs the defective set $\hat{\mathcal{S}}$.

5.6.3 Performance Analysis

a) Number of iterations:

If each defective item is attached to at least one $\mathcal{S}$-leaf node, then in each iteration the algorithm identifies and removes one defective item and the number of iterations is exactly k.

b) Decoding complexity:

For each testing node, checking the multiplicity testing part of GROTESQUE-test costs $c_{\text{mul}} \log(k) + 2\log(c_{\text{mul}} \log(k))$ steps. Therefore, the total computational cost is $[c_{\text{mul}} \log(k) + 2\log(c_{\text{mul}} \log(k))]\,(c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k))$ in the initialization step. We generate the leaf node list using a Red-Black tree and inserting the $\mathcal{S}$-leaf nodes into the Red-Black Tree sequentially. The computational cost is at most $(c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k)) \log(c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k))$.

In each iteration, the cost of localization is $t_{\text{loc}} c_{\text{loc}} \log(n)$ steps. For the total $c_{\text{non,bpt}} \log(k)$ neighbors of the decoded defective item, searching

Algorithm 6 Non-Adaptive Group Testing (k, n)

// Initialization
$\hat{\mathcal{S}} \leftarrow \varnothing$ // Initial list of defective
$\mathcal{L}(\mathcal{S}) \leftarrow \varnothing$ // Initial list of leaf nodes
for each testing node i of $\mathcal{G}$ **do**
    do Multiplicity Decoding
    if $\hat{k'} == 1$ **then**
        $\mathcal{L}(\mathcal{S}) \leftarrow \mathcal{L}(\mathcal{S}) \bigcup \{i\}$ // Found a new leaf node
    end if
end for
$t \leftarrow 1$
$\mathcal{L}(t) \leftarrow \mathcal{L}(\mathcal{S})$
// Operations in t-th iteration
while $t \leq D$ **do**
    Pick a testing node i in $\mathcal{L}(t)$ uniformly at random
    do Localization Decoding // which returns j as the item corresponding to testing node i
    $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \bigcup \{j\}$ // Add j to list of defective items
    $\mathcal{L}(t) \leftarrow \mathcal{L}(t) \backslash$Neighbours of j // Update lisft of leaf nodes
    $t \leftarrow t + 1$
end while
return $\hat{\mathcal{S}}$

whether they are in the leaf node list and removing them from the leaf node list takes at most $2\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))(c_{\text{non,bpt}}\log(k))$ operations, where $\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))$ is the cost of addressing one neighbour of the decoded defective item. Therefore, the time complexity for the iterative decoding process is $t_{\text{loc}}c_{\text{loc}}k\log(n)$ $+2\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))$ $(c_{\text{non,bpt}}k\log(k))$.

Hence, we can conclude that the overall time complexity is at most $[c_{\text{mul}}\log(k) + 2\log(c_{\text{mul}}\log(k))]\,(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))+(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))\cdot$ $\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) + k[t_{\text{loc}}c_{\text{loc}}\log(n) + 2\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))$

$(c_{\text{non,bpt}} \log(k))]$ based on the analysis above.

c) Error probability:

Finally, we show that $\hat{\mathcal{S}} = \mathcal{S}$ with a high probability by choosing the parameters c_{mul} and c_{loc} large enough.

The probability of incorrect multiplicity decoding for each node is at most

$$2 \exp\left(\left(-c_{\text{mul}}(1-2q)^2/32\right) \log(k)\right). \tag{5.5}$$

Then by the union bound, the probability of incorrect multiplicity decoding is bounded from above by

$$2c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k) \exp\left(\left(-c_{\text{mul}}(1-2q)^2/32\right) \log(k)\right). \tag{5.6}$$

The probability of incorrect localization in each iteration is at most $\exp\left((-c_{\text{loc}}(1-H(q))^3/256) \log(n)\right)$. Finally, by applying the union bound over D iteration, the probability of incorrect decoding is bounded from above by

$$k \exp\left(\left(-c_{\text{loc}}(1-H(q))^3/256\right) \log(n)\right). \tag{5.7}$$

Therefore, the overall error probability of our algorithm is at most

$$\begin{aligned} & k^{1-\exp(-1/c_{\text{non,rn}})c_{\text{non,bpt}}/8} + 2c_{\text{non,bpt}} c_{\text{non,rn}} k \log(k) \\ & \exp\left(\left(-c_{\text{mul}}(1-2q)^2/32\right) \log(k)\right) + \\ & k \exp\left(\left(-c_{\text{loc}}(1-H(q))^3/256\right) \log(n)\right). \end{aligned} \tag{5.8}$$

5.7 Two-stage Group Testing

In this section, we present a 2-stage adaptive group testing problem with both decoding complexity and number of tests that is nearly order-optimal (up to a multiplicative factor that is at most $\mathcal{O}(\log(n))$). Again, the objective is to determine an unknown set $\mathcal{S}$ of k defective items from a collection $\mathcal{I}$ of size n. In both stages, we perform tests in a non-adaptive manner,

though the tests of the second stage depend on the outcomes of tests in the first stage. As earlier, we assume that each test outcome may be incorrect independently with probability q.

5.7.1 Overview

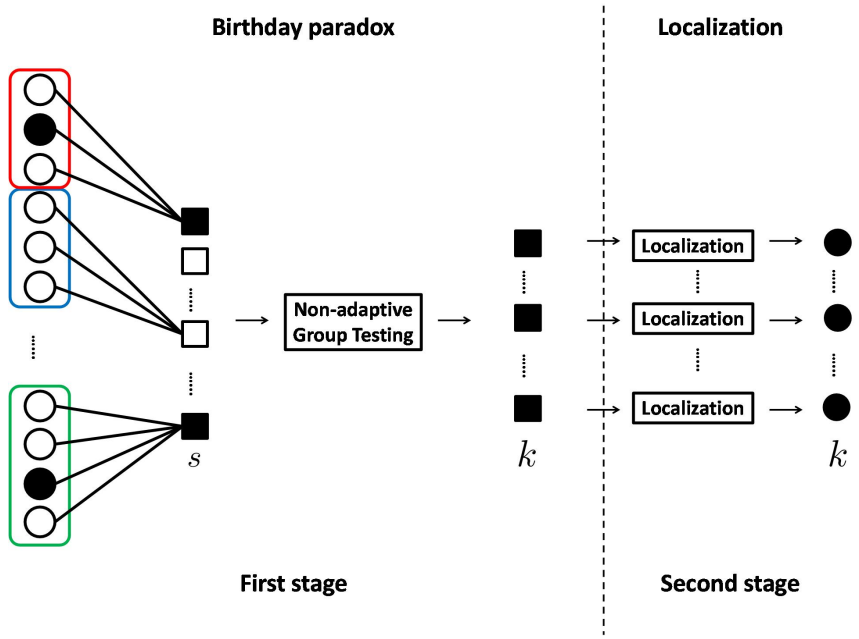


Figure 5.5: In the first stage, we generate a bipartite graph with n nodes on the left representing n items and s nodes on the right. The black circular nodes represent defective items and the white ones represent non-defective items. Each bipartite graph is left-regular with left-degree equal to 1. The item nodes of such graph are partitioned randomly – different coloured collections have different "birthdays". Nodes in the same partition have the same "birthday". With high probability, each testing node is either $\mathcal{S}$-leaf node (black testing node) or $\mathcal{S}$-zero node (white testing node) according to our choice of s. Applying our non-adaptive algorithm, we identify k leaf nodes. In the second stage, applying localization testing on each $\mathcal{S}$-leaf node, we identify the corresponding defective items.

Two-Stage Adaptive Algorithm	
$\mathcal{G}$	A bipartite graph used in the first stage
s	The number of nodes on the right of $\mathcal{G}$

Table 5.7: Table of notation used in our 2-stage adaptive algorithm

Our algorithm has 2 adaptive stages.

In the first stage, we use the birthday paradox problem as a primitive to construct a bipartite graph $\mathcal{G}$. $\mathcal{G}$ has the following properties - the graph is bipartite, has n nodes on the left representing n items (n "people"), is left-regular with regularity equals 1, and s (= poly(k) with degree larger than 3) nodes on the right (s choices of "birthdays"). We show that with high probability $(1 - \mathcal{O}(1/\operatorname{poly}(k)))$, each testing node is either a $\mathcal{S}$-leaf node or a $\mathcal{S}$-zero node (*i.e.*, no pair of them have the same birthday). We use the non-adaptive algorithm discussed in Section 5.6 on the s testing nodes to identify the k $\mathcal{S}$-leaf nodes. In the first stage the total number of tests is $\mathcal{O}(k\log(k)\log(s)) = \mathcal{O}(k\log^2(k))$ and the decoding complexity is $\mathcal{O}(k(\log(s) + \log^2(k))) = \mathcal{O}(k\log^2(k))$.

In the second stage, we use the localization procedure of GROTESQUE with $n' = \mathcal{O}(n/\operatorname{poly}(k))$, on all $\mathcal{S}$-leaf nodes identified in the first stage. Note that with high probability there are exactly k testing nodes that are $\mathcal{S}$-leaf nodes, out of poly(k) testing nodes in total – the fact that we test only k of them is what gives us potentially significant savings in the number of tests and decoding complexity. In the second stage the total number of tests is $\mathcal{O}(k\log(n))$ and the decoding complexity is $\mathcal{O}(k\log n)$. See Figure 5.5 for illustration.

Over both stages, our 2-stage adaptive algorithm hence requires $\mathcal{O}(k(\log(n) + \log^2(k)))$ tests and runs in $\mathcal{O}(k(\log(n) + \log^2(k)))$ steps.

5.7.2 Formal Description

a) Test design and decoding algorithm:

- ***Birthday paradox hashing***: In the first stage, we consider a random left regular bipartite graph $\mathcal{G}$ with s testing nodes and n item nodes. We set each item node of $\mathcal{G}$ to be of degree one. We choose this

Algorithm 7 Two-Stage Group Testing (D, N)

$\hat{\mathcal{S}} \leftarrow \varnothing$ // Initial list of defective
// Birthday paradox: getting D $\mathcal{S}$-leaf nodes
Randomly pool items into S pools, each with N/S items
do Non-Adaptive Group Testing (D, S)
// Localization
for each positive pool i **do**
  Localization Decoding // Given input as pool i, returns item j
  $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \bigcup \{j\}$ // Add j to list of defective items
end for
return $\hat{\mathcal{S}}$

graph uniformly at random. The property we required is that, with high probability, each testing node is either a $\mathcal{S}$-leaf node or a $\mathcal{S}$-zero node (see the definitions in Section 5.5.2). By the "standard birthday paradox argument" [59], the failure probability scales as $\mathcal{O}(1/\operatorname{poly}(k))$ if we choose $s = \mathcal{O}(\operatorname{poly}(k))$ with degree larger than 3 (see Lemma 11 below). To identify all the $\mathcal{S}$-leaf nodes is equivalent to the group testing problem of finding k defectives from s items. We apply our non-adaptive algorithm to all testing nodes. Here if a testing node i is (respectively is not) included in a test, then all the neighbors of i are (respectively are not) included in that test. The outcomes of the first stage are all $\mathcal{S}$-leaf nodes.

- ***Localization*:** In the second stage, we use the GROTESQUE localization procedure (with $n' = \mathcal{O}(n/\operatorname{poly}(k))$) on each $\mathcal{S}$-leaf node identified in the first stage, to decode the corresponding defective item.

5.7.3 Performance Analysis

The analyze the performance of the first part of the algorithm, we require the following lemma.

Lemma 11. *The probability that no defective items have the same neighbor (right node) is at least* $1 - k^{2-\gamma}$*, if we choose* $s = k^\gamma$ *for any* $\gamma \geq 3$*.*

Proof. There are at least two ways to prove the correctness of this lemma.

First, using an argument similar to that in the birthday paradox problem,

$$\begin{aligned}
&\Pr(\text{No defective items have the same neighbor}) \\
&= \frac{\binom{s}{k} k!}{s^k} \\
&= \left(1 - \frac{1}{s}\right)\left(1 - \frac{2}{s}\right) \ldots \left(1 - \frac{k-1}{s}\right) \\
&= \prod_{i=1}^{k-1} \left(1 - \frac{i}{s}\right).
\end{aligned}$$

Using Stirling's approximation,

$$\begin{aligned}
\prod_{i=1}^{k-1} \left(1 - \frac{i}{s}\right) &= \frac{(s-1)!}{s^{k-1}(s-k)!} \\
&\to \frac{\sqrt{2\pi}(s-1)^{s-1+1/2}e^{-s+1}}{s^{k-1}\sqrt{2\pi}(s-k)^{s-k+1/2}e^{-s+k}}, \text{ as } k \to \infty \\
&= e^{1-k} \cdot \frac{(s-1)^{s-1+1/2}}{s^{k-1}(s-k)^{s-k+1/2}} \\
&= e^{1-k} \cdot \frac{(s-1)^{s-1}}{s^{s-1}} \cdot \frac{s^{s-k}}{(s-k)^{s-k}} \cdot \frac{(s-1)^{1/2}}{(s-k)^{1/2}} \\
&\geq e^{1-k} \cdot e^{-1} \cdot e^{k-k^2/s} \cdot e^{-(k-1)/(2s-2)} \\
&= e^{-k^2/s-(k-1)/(2s-2)} \\
&\to e^{-k^2/s} \\
&\geq 1 - k^2/s.
\end{aligned}$$

Therefore, if we choose $s = k^{\gamma}$ with γ larger than 3, the probability that each testing node has no more than two defective items is at least $1 - k^{2-\gamma}$.

For an alternative proof, we consider the probability that the event considered in the statement of this lemma does not happen.

The probability Pr(Any two defective items have the same neighbor) equals $\frac{1}{s}$. Then, by the union bound, the probability that there exist two defective items that have the same neighbor is at most $\frac{\binom{k}{2}}{s} < \frac{k^2}{s}$. Again, we choose $s = k^{\gamma}$ with γ larger than 3 to complete the proof. ■

a) Number of tests:

The number of tests in the first stage is $c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)(c_{\text{mul}}\log(k)+ c_{\text{loc}}\log(s)) = c_{\text{non,bpt}}c_{\text{non,rn}}(c_{\text{mul}} + \gamma c_{\text{loc}})k(\log(k))^2$ and the number of tests in the second stage is $c_{\text{loc}}k\log(n)$. Overall, the number of tests required is $c_{\text{non,bpt}}c_{\text{non,rn}}(c_{\text{mul}} + \gamma c_{\text{loc}})k(\log(k))^2 + c_{\text{loc}}k\log(n)$.

b) Decoding complexity:

The decoding complexity in the first stage is $(c_{\text{non,bpt}}c_{\text{non,rn}}k\log k)\cdot \log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) + k[t_{\text{loc}}c_{\text{loc}}\gamma\log(k) + 2\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) (c_{\text{non,bpt}}\log(k))]$ and the decoding complexity in the second stage is $kt_{\text{loc}}c_{\text{loc}}\log(n)$. Overall, the decoding complexity is $(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) \log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) + k[t_{\text{loc}}c_{\text{loc}}\gamma\log(k) + 2\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) (c_{\text{non,bpt}}\log(k))] + t_{\text{loc}}c_{\text{loc}}k\log(n)$.

c) Error probability:

There are three events we need to consider.

First, the error probability of constructing bipartite graph $\mathcal{G}$ with the properties we need is $k \times k^{2-\gamma} = k^{3-\gamma}$ by union bound over k defective items and Lemma 11.

Second, the error probability of non-adaptive group testing algorithm is

$$k^{1-\exp(-1/c_{\text{non,rn}})c_{\text{non,bpt}}/8} + 2c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)$$

$$\exp\left(\left(-c_{\text{mul}}(1-2q)^2/32\right)\log(k)\right)$$
$$+k\exp\left(\left(-c_{\text{loc}}(1-H(q))^3/256\right)\log(s)\right).$$

Third, in the second stage, the error probability of each localization testing is $\exp\left((-c_{\text{loc}}(1-H(q))^3/256)\log(n)\right)$. By applying union bound over total k $\mathcal{S}$-leaf nodes, the probability of incorrect decoding is at most

$$k\exp\left(\left(-c_{\text{loc}}(1-H(q))^3/256\right)\log(n)\right). \tag{5.9}$$

Therefore, the overall error probability of incorrect decoding scales as $\mathcal{O}(1/\operatorname{poly}(k))$.

5.8 Numerical Results for Noiseless Case

In what follows we assume k and n is an integer power of 2; otherwise small "quantization" correction terms are needed that we do not track in our analysis.

5.8.1 Deterministic grotesque testing with noiseless tests

Multiplicity testing

In Section 5.4, we design the multiplicity testing by picking each item in each test with probability 1/2. Actually, we can derandomize the tests by choosing the matrix $A^{(M)}$ such that each column contains exactly $\frac{m_{\text{mul}}}{2}$ 1's, and requiring all the columns are distinct. Again, we estimate the value of $\hat{k}'$ by counting the number of positives in the multiplicity tests. The value of k' is 1 if and only if the fraction of positives of multiplicity tests is *exactly* 1/2. Since all the columns are different, the fraction must be strictly larger than half when there is more than 1 defectives involved. To guarantee that all the columns are different, we choose m_{mul} to be $2\log(N)$. This choice works since $\binom{a}{b} \geq (\frac{a}{b})^b$, and setting $a = m_{\text{mul}}$, $b = \frac{m_{\text{mul}}}{2}$ we get

that the number of columns that can thereby be obtained is at least N. The decision rule given in Equation (5.1) becomes the following:

$$\hat{k}' = \begin{cases} 0, & \text{if } K = 0 \\ 1, & \text{if } K = \frac{m_{\text{mul}}}{2} \\ \geq 2, & \text{if } K > \frac{m_{\text{mul}}}{2}. \end{cases} \tag{5.10}$$

After counting the number of defectives, we need to compare the number with $m_{\text{mul}}/2$. Therefore, the decoding complexity is $2\log(n)+\log(2\log(n))$. Note that our deterministic multiplicity testing with noiseless tests has the added advantage that it is error-free.

Localization testing

With noiseless tests, we no longer need to use expander codes for localization decoding. Different columns of $A^{(L)}$ correspond to the indices of different items in binary form. We set m_{loc} to be exactly $\log(n)$ so that each item has a distinct signature. If the length of binary number is less than $\log(n)$, 0's are added before the number. If the outcome of multiplicity decoding, $\hat{k}'$, is 1, then we identify the index of the defective item by treating the $\log(N)$-bit test outcome vector (treating positive test outcomes as 1's, and negative test outcomes as 0's) as the binary expansion of the index of the defective item.

For example, suppose there is exactly one defective item among 8 items. The columns of $A^{(L)}$ are $[0,0,1]^T$ through $[1,1,1]^T$ If the outcome of localization decoding is $[0,1,0]^T$, we know that the second item is defective.

Given that the multiplicity decoding is correct, the localization decoding is correct with probability 1. Also, the decoding complexity of localization depends on reading the output which costs $\log(n)$ computational steps.

Based on the modified design described above of grotesque testing for

noiseless tests, we have the following corollaries,

Corollary 4. *For any set of defectives $\mathcal{S}$ from $\mathcal{I}$ ($|\mathcal{S}| = k$ and $|\mathcal{I}| = n$ are large enough), the Multi-stage Adaptive Group Testing algorithm with noiseless tests produces a reconstruction of the collection $\hat{\mathcal{S}}$ of $\mathcal{S}$ such that $\hat{\mathcal{S}} = \mathcal{S}$ for any positive constants $c_{\text{adp,rn}}$, $c_{\text{adp,deg}}$ and $P'_{\text{leaf}} \geq c_{\text{adp,deg}} e^{-c_{\text{adp,deg}}}$. The algorithm has the following properties:*

1) The number of tests m is at most

$$12e^{1/2}k\log(n) + 3c_{\text{adp,rn}}(\log(k))^2\log(\log(k))\log(n),$$

2) The number of stages is at most $1+\log\left(k/\log(k)\right)/[-\log\left(1-e^{-1/2}/2\right)]$,

3) The decoding complexity is

$$\left[4e^{1/2}k + c_{\text{adp,rn}}(\log(k))^2\log(\log(k))\right]\left[3\log(n) + \log(2\log(n))\right], \text{and}$$

4) The error probability is at most

$$\frac{\log\left(k/\log(k)\right)}{-\log\left(1-e^{-1/2}/2\right)}k^{-1/(4e)} + (\log(k))^{1-c_{\text{adp,rn}}P'_{\text{leaf}}\log(k)}$$

over the internal randomness in the algorithm.

Corollary 5. *For any set of defectives $\mathcal{S}$ from $\mathcal{I}$ ($|\mathcal{S}| = k$ and $|\mathcal{I}| = n$ are large enough), the Non-Adaptive Group Testing algorithm with noiseless tests produces a reconstruction of the collection $\hat{\mathcal{S}}$ of $\mathcal{S}$ such that $\hat{\mathcal{S}} = \mathcal{S}$ for any positive constants $c_{\text{non,bpt}}$, $c_{\text{non,rn}}$. The algorithm has the following properties:*

1) The number of tests m equals $3c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)\log(n)$,

2) The decoding complexity is

$$\left[2\log(n) + \log(2\log(n))\right]\left(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)\right)+$$

$$\left(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)\right)\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))+$$

$$k[\log(n) + 2\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))(c_{\text{non,bpt}}\log(k))], \text{and}$$

3)The error probability is at most $k^{1-\exp(-1/c_{\text{non,rn}})c_{\text{non,bpt}}/8}$ *over the internal randomness in the algorithm.*

Corollary 6. *For any set of defectives* $\mathcal{S}$ *from* $\mathcal{I}$ *(*$|\mathcal{S}| = k$ *and* $|\mathcal{I}| = n$ *are large enough), the Two-stage Adaptive Group Testing algorithm with noiseless tests produces a reconstruction of the collection* $\hat{\mathcal{S}}$ *of* $\mathcal{S}$ *for any positive constants,* $c_{\text{non,bpt}}$*,* $c_{\text{non,rn}}$*, and* $s = k^{\gamma}$ *with* $\gamma > 3$*. The algorithm has the following properties:*

1) The number of tests m *equals* $3c_{\text{non,bpt}}c_{\text{non,rn}}\gamma k(\log(k))^2 + k\log n$*,*

2) The number of stages is 2*,*

3) The decoding complexity is

$$[2\gamma\log(k) + \log(2\gamma\log(k))]\,(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) +$$

$$(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k)) +$$

$$k[\gamma\log(k) + 2\log(c_{\text{non,bpt}}c_{\text{non,rn}}k\log(k))(c_{\text{non,bpt}}\log(k))] + k\log(n), \textit{and}$$

4) The error probability is at most $k^{1-\exp(-1/c_{\text{non,rn}})c_{\text{non,bpt}}/8} + k^{3-\gamma}$ *over the internal randomness in the algorithm.*

Remark 10. Note that for the multi-stage adaptive algorithm, the number of tests therefore scales as $\mathcal{O}(k\log(n))$. The decoding complexity scales as $\mathcal{O}(k\log(n))$. The number of stages scales as $\mathcal{O}(\log(k))$. The error probability scales as $\mathcal{O}(1/\operatorname{poly}(k))$. For non-adaptive algorithm, the number of tests therefore scales as $\mathcal{O}(k\log(k)\log(n))$. The decoding complexity scales as $\mathcal{O}(k\log(k)\log(n))$. The error probability scales as $\mathcal{O}(1/\operatorname{poly}(k))$. For two-stage adaptive algorithm, the number of tests therefore scales as $\mathcal{O}(k\log(n) + k\log^2(k))$. The decoding complexity scales as $\mathcal{O}(k\log(n) + k\log^2(k))$. The error probability scales as $\mathcal{O}(1/\operatorname{poly}(k))$.

In the next section, we run simulations for multi-stage adaptive and non-adaptive algorithms in noiseless case.

5.8.2 Simulation Results

We present simulation results to demonstrate a "proof of concept" for our design principles. We focus on the scenario with noiseless tests; in principle similar experiments could also be done for the scenario with noisy tests. However, the size of k and n required to demonstrate meaningful performance gains would be somewhat intricate, involving careful choice of error-correcting codes with good parameters. Since our two-stage adaptive algorithm is based on our non-adaptive algorithm, we only focus on the multi-stage adaptive algorithm and the non-adaptive algorithm in this section.

Adaptive Algorithm

The parameters values for our "base scenario" are listed in Table 5.8. For each set of parameters, we run simulations 100 times and interpret the percentage of times we successfully decode as our success rate. The decoding is successful if and only if $\hat{\mathcal{S}} = \mathcal{S}$. We set the number of items n to be $100,000$. Therefore, we set the number of tests per multiplicity testing, m_{mul}, as 20 since $\left(\binom{20}{10} = 184,756 > 100,000\right)$ and setting the number of tests per localization testing, m_{loc}, as 17 since $(2^{17} = 131,072 > 100,000)$. Recall that $c_{\text{adp,rn}}$ and $c_{\text{adp,deg}}$ are constants related to the number of testing nodes and the degree of testing nodes for bipartite graph in coupon collection stage respectively. For the first two sets of simulations, the results guide us on how to choose the internal parameters $c_{\text{adp,rn}}$ and $c_{\text{adp,deg}}$. Then, we see how the probability of successful decoding changes as the number of tests varies. Last, we record the running time of our decoding algorithm for different values of number of defectives.

a) Varying $c_{\text{adp,rn}}$

We keep all the parameters the same as in the base scenario except that

n	$100,000$
k	50
m_{mul}	20
m_{loc}	17
$c_{\text{adp,rn}}$	4
$c_{\text{adp,deg}}$	1

Table 5.8: Base scenario for multi-stage adaptive algorithm

we change the value of $c_{\text{adp,rn}}$ from 0.2 to 4. The simulation result is shown in Fig. 5.6. We observe that when $c_{\text{adp,rn}} \geq 2.4$ our adaptive group testing algorithm is successful with probability close to 1.

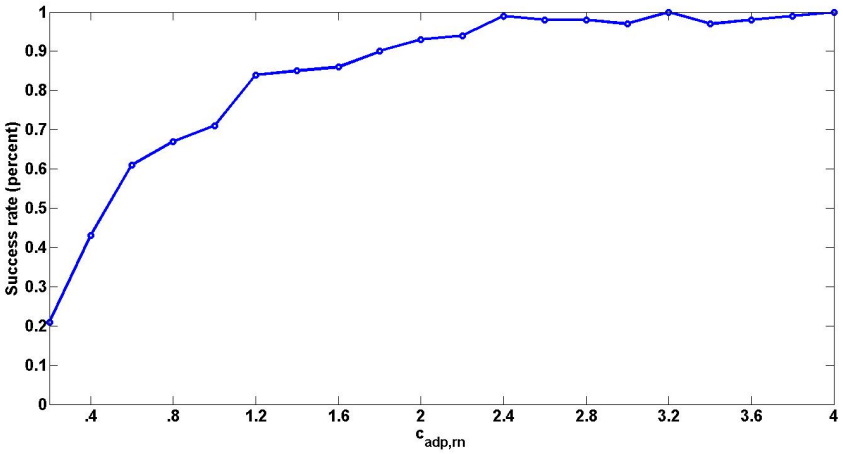


Figure 5.6: Adaptive algorithm with Noiseless tests - reconstruction performance for varying $c_{\text{adp,rn}}$.

b) Varying $c_{\text{adp,deg}}$

We keep all the parameters the same as in the base scenario except that we change the value of $c_{\text{adp,deg}}$ from 0.1 to 1. The simulation result is shown in Fig. 5.7. We observe that when $c_{\text{adp,deg}} \geq 0.5$ our adaptive group testing algorithm is successful with probability close to 1.

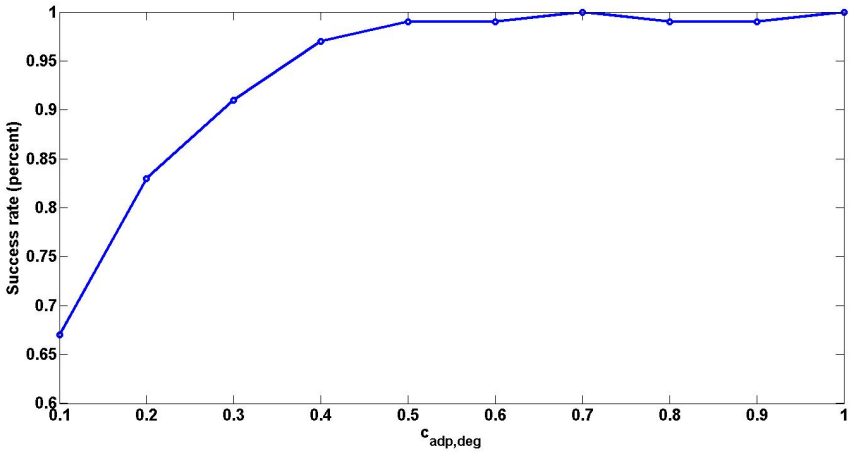


Figure 5.7: Adaptive algorithm with Noiseless tests - reconstruction performance for varying $c_{\text{adp,deg}}$.

c) Number of tests for varying k and n

We keep all the parameters the same as in the base scenario except that we change the number of defective items from 10 to 100. For multi-stage adaptive algorithm, it is impossible to get the exact number of tests. In this set of simulations, for a certain ratio of $m/k\log(n)$ (see Fig. 5.8), our decoding algorithm is successful if and only if $\hat{\mathcal{S}} = \mathcal{S}$ and the number of tests over $k\log(n)$ is less than the ratio. The vertical line at $m/(k\log(n)) = 12e^{1/2} \approx 20$ corresponds to the theoretical upper bound.

d) Running time of decoding algorithm for varying k and n

We keep all the parameters the same as in the base scenario except that we change the number of defective items from 10 to 100 and the number of items from $1,000$ to $100,000$. For each set of parameters, we estimate the running time of our algorithm by averaging the running time over 100 simulations. The exact running time can be much improved considering that the simulator is written in Matlab and not fully optimized, and we run the algorithm on a laptop.[16] Note that the running time is linear in k

[16]We use a laptop with 1.7GHz Intel Core i5 and 4 GB memory.

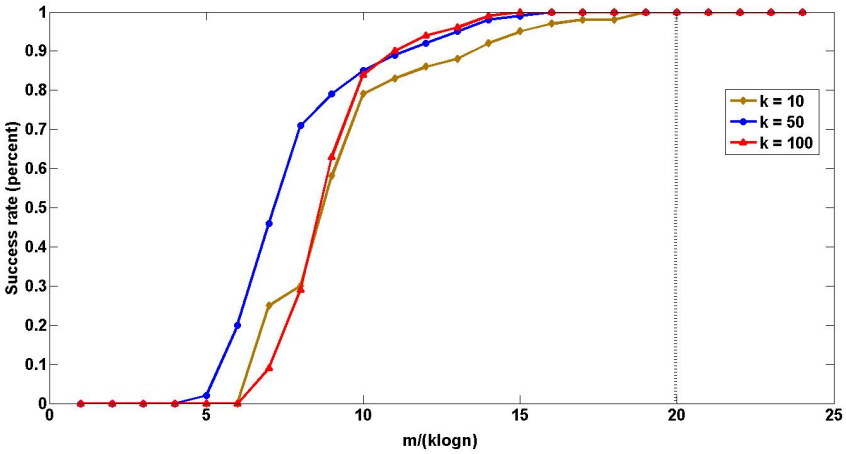


Figure 5.8: Adaptive algorithm with Noiseless tests - number of tests required for varying k and m.

and running time increases by a small constant factor for larger n.

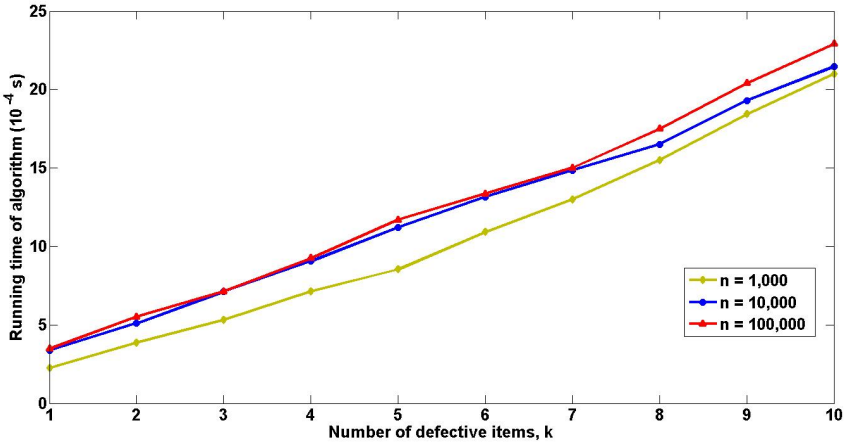


Figure 5.9: Adaptive algorithm with Noiseless tests - running time for varying k and n.

Non-adaptive Algorithm

The parameters' values for base scenario are listed in Table 5.8. For each set of parameters, we run simulation 100 times and use the percentage

of successful decoding as success rate. The decoding is successful if and only if $\hat{\mathcal{S}} = \mathcal{S}$. We set the number of items n to be $100,000$. Therefore, we set the number of tests per multiplicity testing, m_{mul}, as 20 since $\left(\binom{20}{10} = 184,756 > 100,000\right)$ and setting the number of tests per localization testing, m_{loc}, as 17 since $(2^{17} = 131,072 > 100,000)$. Recall that $c_{\text{non,rn}}$ and $c_{\text{non,bpt}}$ are constants related to the number of bipartite graphs and the number of testing nodes for each bipartite graph, respectively. For the first two sets of simulations, the results guide us on how to choose the internal parameters $c_{\text{non,rn}}$ and $c_{\text{non,bpt}}$. Then, we see how the probability of successful decoding changes as the number of tests varies. Last, we record the running time of our decoding algorithm for different values of number of defectives.

n	$100,000$
k	50
m_{mul}	20
m_{loc}	17
$c_{\text{non,rn}}$	1.2
$c_{\text{non,bpt}}$	2

Table 5.9: Base scenario for non-adaptive algorithm

a) Varying $c_{\text{non,rn}}$

We keep all the parameters the same as in the base scenario except that we change the value of $c_{\text{non,rn}}$ from 0.1 to 2. The simulation result is shown in Fig. 5.10. We observe that when $c_{\text{non,rn}} \geq 1.2$ our adaptive group testing algorithm is successful with probability close to 1.

b) Varying $c_{\text{adp,deg}}$

We keep all the parameters the same as in the base scenario except that we change the value of $c_{\text{non,bpt}}$ from 0.4 to 3.2. The simulation result

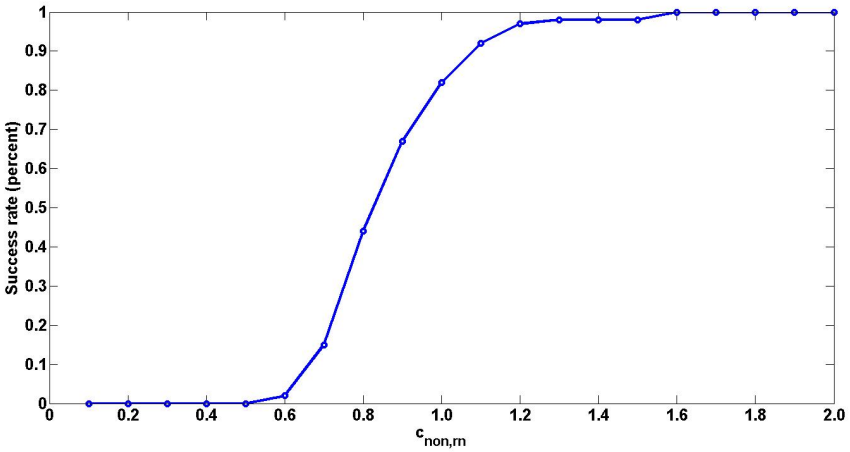


Figure 5.10: Non-adaptive algorithm with Noiseless tests - reconstruction performance for varying $c_{\text{non,rn}}$.

is shown in Fig. 5.11. We observe that when $c_{\text{non,bpt}} \geq 1.6$ our adaptive group testing algorithm is successful with probability close to 1.

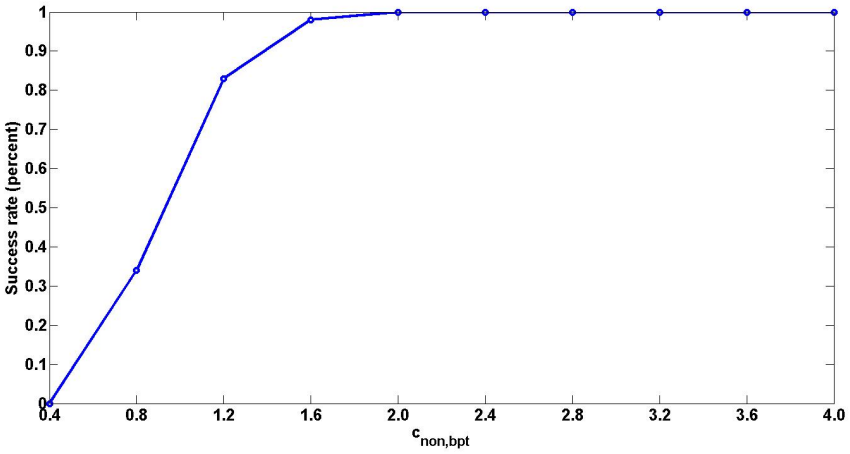


Figure 5.11: Non-adaptive algorithm with Noiseless tests - reconstruction performance for varying $c_{\text{non,bpt}}$.

c) Number of tests for varying k and n

We keep all the parameters the same as in the base scenario except that we change the number of defective items from 10 to 100. Note that the

actual number of tests required to get reasonable decay in probability of err is significantly smaller than would be implied by Corollary 5.

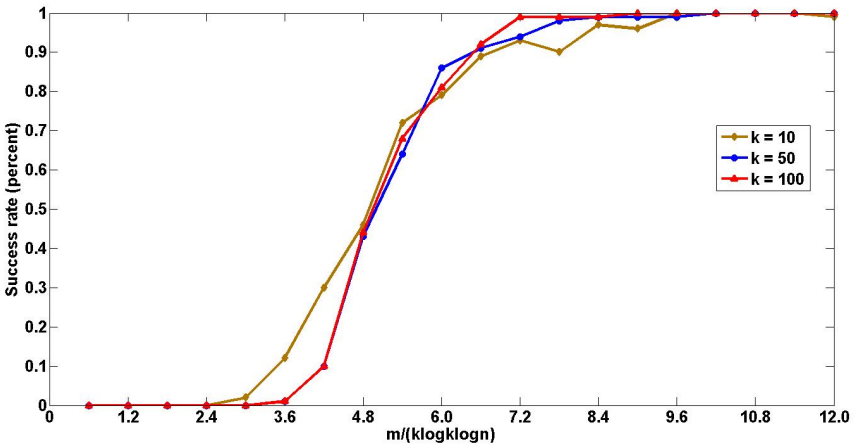


Figure 5.12: Non-adaptive algorithm with Noiseless tests - number of tests required for varying k and m.

d) Running time of decoding algorithm for varying k and n

We keep all the parameters the same as in the base scenario except that we change the number of defective items from 10 to 100 and the number of items from $10,000$ to $100,000$. For each set of parameters, we estimate the running time of our algorithm by averaging the running time over 100 simulations.

5.9 Conclusion

In this work we consider three group testing algorithms, specifically for adaptive, nonadaptive, and two-stage adaptive scenarios. In each of these scenarios, we present the first algorithms whose computational complexity is nearly information-theoretically order-optimal. The number of tests required in our algorithms is also nearly information-theoretically order-optimal (by the same factor). Our "near optimality" is up to either uni-

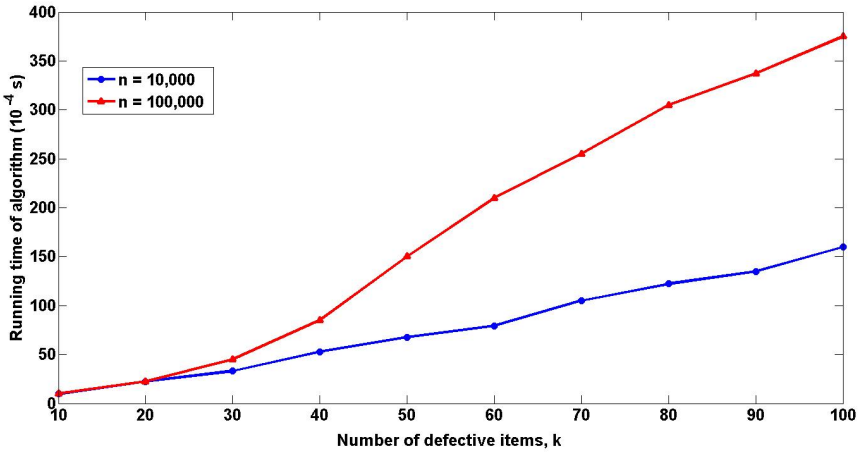


Figure 5.13: Non-adaptive algorithm with Noiseless tests - running time for varying k and n.

versal constant factors or in some cases factors that are poly-logarithmic in problem parameters. We present the framework that can indeed lead to practical, fast algorithms. We do not further explore optimized codes in this work (for instance, it is likely that a variant of density evolution [122] may well lead to better constant factor performance. This is a topic of ongoing research).

□ **End of chapter.**

Chapter 6

Compressive Phase Retrieval – SUPER

6.1 Introduction

Phase Retrieval: In many applications of practical interest, it is difficult to measure the phase information of the signal directly. Instead, we first perform intensity measurements based on the underlying signal and then reconstruct the signal from the measurements. For instance, in X-ray crystallography, optics [100] and image reconstruction for astronomy [46], the signal/image is reconstructed from the intensity measurements of its Fourier transform.

Let $A \in \mathbb{C}^{m \times n}$ be used to denote the *phase measurement matrix*, and $\mathbf{x} \in \mathbb{C}^n$ be used to denote the unknown underlying signal. Instead of *linear* measurements of the form $A\mathbf{x}$ as in the *compressive sensing* literature (for instance, see [31]) in the *phase retrieval problem* we have m *non-linear intensity measurements* of the form $y_i = |A_i\mathbf{x}|$. Here the index i is an integer in $\{1, \ldots, m\}$ (or [m] for short), A_i is the i-th row of phase measurement matrix A and $|\cdot|$ is the absolute value.

Problems of this kind have been studied extensively over the last decades. A good survey of some of the algorithms via non-convex methods can be found in [60,61]. Recently, two convex optimization methods, PhaseLift [32] and PhaseCut [140], have been proposed by Candès *et al.* and Waldspurger *et al.* PhaseLift is inspired by finding low-rank matrices via minimizing the *trace norm* of matrices [30]. Specifically, for the phase retrieval problem, the rank of matrices is one. PhaseLift is able to reconstruct $\mathbf{x}$ with $\mathcal{O}(n\log(n))$ intensity measurements by solving a semidefinite programming (SDP) problem with high probability. The A_i's are independently sampled on the unit sphere of $\mathbb{C}^n$. Subsequently, the number of intensity measurements can be improved to $\mathcal{O}(n)$ by choosing A_i's independently and identically from the uniform distribution on the sphere of radius $\sqrt{n}$, or the complex normal distribution [29]. PhaseCut is inspired by solving max-cut problem via SDP. The decoding complexity for both PhaseLift and PhaseCut is $\mathcal{O}\left(n^3\right)$, which is still computationally costly when n is large.

Recently, besides the SDP-based approach, more computationally efficient algorithms are proposed such as in [7], [104] and [116]. For instance, in [104], the number of intensity measurements required is $\mathcal{O}\left(n\log^3(n)\right)$. However, the decoding complexity is $\mathcal{O}\left(n^2\log^3(n)\right)$ which is less than that of the SDP-based approach. In [116], the number of measurements required and decoding complexity of the algebraic reconstrucction algorithm are $4n-4$ and $\mathcal{O}(n)$, respectively. The algorithm works for all signals except a specific 1-dimensional subspace, or a union of subspaces of $\mathbb{C}^n$, respectively.

Compressive Phase Retrieval:

Suppose $\mathbf{x}$ is "sparse", *i.e.*, the number of non-zero components of $\mathbf{x}$ is at most k, which is much less than the length n of $\mathbf{x}$. This assumption is not uncommon in many applications like X-ray crystallography. Then,

given A and $\mathbf{y}$, the goal of *compressive phrase retrieval* is to reconstruct $\mathbf{x}$ as $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ equals $\mathbf{x}$ up to a global phase. That is, $\hat{\mathbf{x}} = \mathbf{x}e^{\iota\Theta}$ for some arbitrary fixed $\Theta \in [0, 2\pi)$. Here ι denotes the imaginary unit. The reason we allow this degeneracy in $\hat{\mathbf{x}}$, up to a global phase factor, is that all such $\hat{\mathbf{x}}$'s result in the same measurement vector under intensity measurements. If $\hat{\mathbf{x}}$ does indeed equal $\mathbf{x}$ up to a global phase, then we denote this "equality" as $\hat{\mathbf{x}} \hat{=} \mathbf{x}$.

It is shown that $4k - 1$ intensity measurements suffice to uniquely reconstruct $\mathbf{x}$ in [108] (for $\mathbf{x} \in \mathbb{R}^n$) and [5] (for $\mathbf{x} \in \mathbb{C}^n$). However, no efficient algorithm is given. The ℓ_1-regularized PhaseLift method is introduced in the compressive phase retrieval problem in [109]. In [90], it is shown that if the number of Gaussian intensity measurements is $\mathcal{O}\left(k^2 \log(n)\right)$, $\mathbf{x}$ can be correctly reconstructed via ℓ_1-regularized PhaseLift.

The works in [128] and the works by Jaganathan *et al.* [81–83] study the case when the phase measurement matrix is a Fourier transform matrix. [110] shows that SDP-based methods can reconstruct $\mathbf{x}$ with sparsity up to $o\left(\sqrt{n}\right)$. In [83], the algorithm based on reweighted ℓ_1-minimization with $\mathcal{O}\left(k^2 \log(n)\right)$ phaseless Fourier measurements is proposed to go beyond this bottleneck. When the phase measurement matrix is allowed to be designed, a matrix ensemble and a corresponding combinatorial algorithm is proposed in [83] such that $\mathbf{x}$ is correctly reconstructed with $\mathcal{O}(k \log(n))$ intensity measurements in $\mathcal{O}(kn \log n)$ time. The Unicolor algorithm in [114] builds on our work [24] and is able to reconstruct a constant fraction of non-zero components of $\mathbf{x}$ with $\mathcal{O}(k)$ measurements in $\mathcal{O}(k)$ time. See Section 6.3 for a detailed comparison.

To the best of our knowledge, in the literature, there is no prior construction of an ensemble of measurement matrices A and a corresponding reconstruction algorithm that correctly reconstructs $\mathbf{x}$ with an order-optimal

number of measurements and with near-optimal decoding complexity simultaneously.

Notation	Definition
$\mathbf{x}$	Length-n signal over $\mathbb{C}$ with sparsity k
A	Dimension-$n \times m$ phase measurement matrix over $\mathbb{C}$.
$\mathbf{y}$	Length-m Intensity measurement vector over $\mathbb{R}_0^+$.
A_i	The i-th row of phase measurement matrix A for all $\forall i \in [m]$.
y_i	$y_i = \lvert\langle A_i,\ \mathbf{x}\rangle\rvert$, the i-th intensity measurement $\forall i \in [m]$.
k	$k = \lVert\mathbf{x}\rVert_0$, the number of non-zero components (sparsity) of $\mathbf{x}$.

Table 6.1: Table of notation for the model

6.1.1 Our Contribution

In this work,[1] we focus on compressive phase retrieval problem with noiseless intensity measurements. We propose SUPER,[2] which consists of a randomized design of the measurement matrix and a corresponding decoding algorithm that achieve the following guarantees:

Theorem 3. *(Main theorem) There exists a measurement ensemble $\{A\}$ and a corresponding decoding algorithm for compressive phase retrieval with the following performance:*

1. For every $\mathbf{x} \in \mathbb{C}^n$, with probability $1-o(1)$ over the randomized design of A, the algorithm exactly reconstructs $\mathbf{x}$ up to a global phase
2. The number of measurements $m = \mathcal{O}(k)$, and

[1] While in this work we focus on the "sparse regime", $k = o(n)$, our techniques work for all $k \in \{1, 2, \ldots, n\}$. If $k = \omega(1)$, our algorithm has the same performance stated in Theorem 3. If k is a constant and error probability of our algorithm is P_e, then the number of measurements required is $f(P_e)k$ for some function f.

[2] SUPER stands for Sparse signals with Unknown Phases Efficiently Recovered.

3. The decoding complexity is $\mathcal{O}(k\log(k))$.

The rest of this chapter is organized as follows. We first present the high-level overview of our algorithm in Section 6.2. In Section 6.4, we introduce the graphs used to induce the measurement structure. Sections 6.5 and Section 6.7 contain actual measurement design. Section 6.6 and Section 6.8 discuss the reconstruction algorithm and its performance. Section 6.9 provides concluding remarks.

6.2 Overview/High-level Intuition

Our SUPER algorithm is non-adaptive. There are three phases[3] in our decoding algorithm. In the first phase (called seeding phase), we are able to recover the magnitudes and relative phases of a constant fraction of non-zero components of $\mathbf{x}$. In the second phase (called geometric-decay phase), there are $\mathcal{O}(\log(\log(k)))$ stages. In each stage, we recover the magnitudes and relative phases of a constant fraction of unresolved non-zero components of $\mathbf{x}$. In the third phase (called cleaning-up phase), the remaining $\mathcal{O}(k/\log(k))$ unresolved non-zero components are decoded.

6.2.1 Pieces of the puzzle

We first define some useful terminology.

Singletons:

If a measurement y_i involves only a single non-zero component of $\mathbf{x}$, then we say that such a measurement is a *singleton*.[4] Singletons are important

[3] All the measurements are designed before the decoding process, so it is non-adaptive.

[4] Singletons and multitons in this chapter are interchangeable with leaf-nodes and non-leaf-nodes in previous chapters. However, a special kind of non-leaf-nodes, doubletons, are exploited in this chapter. To differentiate the different non-leaf-nodes, we stick to using singletons, doubletons, and multitons in this chapter.

since they can be used to pin down the magnitude (though not the phase) of components of $\mathbf{x}$. There are several challenges, however. One lies in even identifying whether a measurement is a singleton or not. The second lies in identifying which of the $\mathbf{x}$ components being measured in y_i corresponds to the singleton. The third is to be able to do all this blindingly fast, in fact in *constant* time (independent of n and k). Each of these challenges can be handled by using ideas from our prior work on compressive sensing [10]. For details, see Sections 6.5 and 6.6 below.

Doubletons:

Similarly, if a measurement y_i involves exactly two non-zero components of $\mathbf{x}$, then we say that such a measurement is a *doubleton*. Doubletons, especially doubletons measuring two non-zero components of $\mathbf{x}$ which have already been measured by singletons (we call such doubletons *resolvable doubletons*), are useful since they can be used to deduce the relative phases of the two non-zero components of $\mathbf{x}$. For example, if one is given the magnitudes $|x_i|$, $|x_j|$, and $|x_i + x_j|$, then one can determine the angle θ between the phases of the complex numbers x_i and x_j (up to degeneracy of sign of θ). In fact, even this degeneracy can be resolved by an additional judiciously chosen measurement. Similar challenges to those mentioned above vis-a-vis singletons (identifying whether or not a measurement is a doubleton/resolvable doubleton, identifying which components of $\mathbf{x}$ it corresponds to, and doing so in constant time) also hold for doubletons. See Sections 6.5 and 6.6 for details.

Mutual resolvability:

We say our decoding algorithm has thus far *mutually resolved* two non-zero components x_i and x_j of $\mathbf{x}$ if the magnitudes of both x_i and x_j have been deduced, and also the relative phase between x_i and x_j has been deduced (for instance via resolvable doubleton measurements roughly de-

scribed above). Note that mutual resolvability is an equivalence relation – it is reflexive, symmetric and transitive. Note therefore that if x_i and $x_{i'}$ have been mutually resolved, it is not necessary that they even are involved in the same measurement; it is sufficient that x_i and $x_{i'}$ are part of a chain of non-zero components of $\mathbf{x}$ that are pairwise mutually resolved.[5] Finally, we note that as our decoding algorithm progresses, if it is successful, in fact *all* the non-zero components of $\mathbf{x}$ are eventually mutually resolved. Hence this property of mutual resolvability is perhaps most interesting in the intermediate stages of our decoding algorithm.

Giant component:

We say that a subset of the non-zero components of $\mathbf{x}$ form a giant component if it is the largest subset satisfying the two properties:

- The subset is of size linear in k.
- Any pair of components in the subset have been mutually resolved (thus far) by the decoding algorithm.

Non-zero components of $\mathbf{x}$ that have not (yet) been mutually resolved with respect to an element of the giant component by the decoding algorithm are said to be unresolved.

Essentially, our algorithm proceeds by iteratively enlarging the giant component until it engorges all the non-zero components of $\mathbf{x}$.

Resolvable multiton:

We say that a measurement y_i is a resolvable multiton if it is the case that exactly one (say x_j) of the non-zero components of $\mathbf{x}$ involved in the measurement y_i is outside the giant component, and at least one of non-zero components of $\mathbf{x}$ is inside the giant component. Such measurements are useful since, in the latter parts of our algorithm, there are not enough

[5] Mutual resolvability is a symmetric and transitive property.

resolvable doubletons. By carefully choosing the parameters of the algorithm, one can guarantee that a constant fraction of measurements are resolvable mutitons.

Judiciously designed measurements (see Section 6.5) enable one to mutually resolve the component x_i that is outside the giant component, with the components of $\mathbf{x}$ inside the giant component, by solving a quadratic equation. Care is indeed required in choosing the measurements since the amplitude measurement process is inherently non-linear, and there may not be a "clean" manner to mutually resolve x_i via arbitrary measurements – indeed the design of such a measurement process is also one of the intellectual contributions we wish to highlight in this work. We call this process "cancelling out" the already resolved components of $\mathbf{x}$.

6.2.2 Putting the pieces together

Seeding phase:

In the first phase, called the *seeding phase*, there are $\mathcal{O}(k)$ "sparse" measurements (each measurement involves, in expectation, $\mathcal{O}(n/k)$ components of $\mathbf{x}$). We demonstrate that by first examining the measurements corresponding to this phase, the decoding algorithm is already able to decode a constant fraction (say half)[6] of the components of $\mathbf{x}$ up to a global phase. The algorithm is able to do this since we are able to show that a "significant" fraction of measurements are singletons and resolvable doubletons. Standard results in percolation theory [20] then lead one to conclude that the number of non-zero nodes that are mutually resolvable is linear in k, *i.e.*, that there is a giant component. Hence this phase is called the

[6] Here, 1/2 is arbitrarily chosen to simplify the presentation of intuition. The actual fraction of resolved non-zero components in the seeding phase is different from 1/2. See Section 6.8 for details. Here, the parameter 1/2 for the geometric-decay phase in this section is due to the same reason.

"seeding" phase, since the giant component forms the nucleus on which the remainder of the algorithm builds upon.

Prior work [83] close to our work here comprises essentially only of the seeding phase, but with $\mathcal{O}(k\log(k))$ measurements.[7] The reason that prior work needs this many measurements is essentially due to what happens at the tail end of a "coupon collection" process [103] (wherein one has to collect at least one copy of each of k coupons by sampling with replacement) – when most of the coupons have already been collected/the giant component is of size close to k, then the growth rate slows down. Specifically, this is because the fraction of resolvable doubletons decays slowly to zero, and an additional multiplicative factor of $\log(k)$ measurements is required so as to ensure the giant component subsumes all non-zero components of $\mathbf{x}$.

The key technique used in our work, then, is to segue to a different sampling process outlined below, and using resolvable multitons rather than doubletons. The challenge is to make the numbers work – unlike [83], not only do we require only $\mathcal{O}(k)$ measurements, but we also require our decoding complexity to be $\mathcal{O}(k\log(k))$.[8]

Geometric-decay phase:

This phase itself comprises of $\mathcal{O}(\log(\log(k)))$ separate stages. Each stage has half the number of measurements compared to the previous stage, but measurements in each stage are twice as "dense" as the measurements in the previous stage. So, for instance, if in the first stage of the geometric-decay phase, there are say $ck/2$ measurements, with each measurement involving $2n/k$ components of $\mathbf{x}$, then in the second stage of the geometric-decay

[7] The combinatorial algorithm in [83] can be modified to have $\mathcal{O}(k\log(k))$ measurement with error probability $\mathcal{O}(1/\operatorname{poly}(k))$ instead of $1/n$ in the paper. Also, based on our reconstruction algorithm, the decoding complexity can be reduced to $\mathcal{O}(k\log(k))$.

[8] In this section, we focus on the number of measurements and decoding complexity. For the error probability, please refer to Section 6.8.

phase, there are $ck/4$ measurements, but each measurement involves $4n/k$ components of $\mathbf{x}$.

There are two reasons for this choice of parameters. Firstly, with such a geometric decay in the number of measurements in each stage, the overall number of measurements in the geometric-decay phase is still $\mathcal{O}(k)$. Secondly, we show that with the geometric increase in the density of measurements, a significant fraction of measurements in each stage lead to resolvable multitons, and use this to show that the number of unresolved components decays geometrically.

The reason we run the geometric-decay phase for only $\mathcal{O}(\log(\log(k)))$ stages is also two-fold. Firstly, after that many stages, with the number of unresolved components halving at every stage, the number of unresolved components of $\mathbf{x}$ is, in expectation, $\mathcal{O}(k/\log(k))$. Hence the concentration inequalities (which depend on the number of unresolved components) we use to control the probabilities of error get progressively weaker (though they still result in good concentration at the last stage of the geometric-decay phase). Secondly, and more importantly, the number of non-zero components in each resolvable multiton increases geometrically as the number of stages increases. This has implications for the time-complexity of the decoding algorithm, since the time-complexity depends directly on the number of non-zero components in each measurement that need to be "cancelled out". By terminating the geometric-decay phase after $\mathcal{O}(\log(\log(k)))$ stages ensures that, in expectation, the number of such "cancellations" is at most $\mathcal{O}(\log(k))$, and hence the overall time-complexity of the algorithm scales as $\mathcal{O}(k\log(k))$.

Cleaning-up phase:

Finally, we segue to what we call the "cleaning-up" phase. As noted above, after the geometric-decay phase the number of unresolved compo-

nents of $\mathbf{x}$ is, in expectation, $k' \triangleq \mathcal{O}(k/\log(k))$. To fit our budget of $\mathcal{O}(k)$ measurements, and $\mathcal{O}(k\log(k))$ decoding time, we now segue to using "coupon collection" as a primitive. This may be viewed as restarting the seeding (first) phase, but with different parameters. In particular, the problem dimension has now been significantly reduced (since there are now only k' unresolved components of $\mathbf{x}$). Therefore we can now afford to pay the coupon collection penalty that we avoided in the seeding phase by moving to the geometric-decay phase.

Specifically, in this cleaning-up phase we take $\mathcal{O}(k'\log(k'))$ measurements so as to resolve the remaining k' unresolved components of $\mathbf{x}$. Note that $\mathcal{O}(k'\log(k'))$ scales as $\mathcal{O}(k)$. Each measurement we take has the same density as the measurements in the last stage of the geometric decay phase, and hence the time-complexity of resolving measurements also scales in the same manner. However, since there are many more measurements than in the last stage of the geometric-decay phase, by standard arguments corresponding to the coupon collection problem we are able to argue that for each unresolved component of $\mathbf{x}$ there is at least one resolvable multiton that helps resolve it.

6.2.3 Summary of the overview

As the above discussion outlines, to make the numbers work (*i.e.*, to ensure $\mathcal{O}(k)$ number of measurements and $\mathcal{O}(k\log(k))$ time-complexity), one has to delicately choose the parameters of the measurement ensemble. Our analysis indicates that having a phase in which the sparsity actually geometrically increases, at least for a while, significantly improves performance. To take advantage of this, however, we have to carefully design the measurements, so that one can resolve unresolved components of $\mathbf{x}$ via judiciously designed non-linear measurements. In this work we have not

attempted to optimize the constant factors – we expect further constant-factor improvements are possible via further careful tuning; indeed, this is one of the focuses of the work in [114].

6.3 Highly related work

Chronologically, and also logically, the first work to focus on phase recovery via a peeling process was that by [83]. Follow-up work by the authors of this work in [24] (the conference version of this work) improved on the parameter space as discussed below. Finally, the work by [114] builds further on [24] and expands the parameter regime considered.

At highest level, it could be argued that each of the works has the following two similarities. First, the measurement matrices in all the works can be generated using the structure of bipartite graphs, even if this is not explicitly mentioned in [83]. Second, all the algorithms use a peeling process to decode. Unlike the peeling process using singletons in the compressive sensing problem [10, 112], singleton peeling process recovers only the magnitudes of non-zero components in the phase retrieval problem. Therefore for the phase retrieval problem, a modified peeling process using doubletons or multitons helps to recover the relative phases between non-zero components – this observation was also used in each of [24, 83] and [114].

Next, we focus on the difference between these works. Differences in reconstructions goals, measurement ensembles (in terms of both sparsity structure of the measurement matrices, and entries of non-zeroes) and corresponding subtle changes can lead to large differences in provable guarantees of algorithms such as the number of measurements and decoding complexity.

- **Differing reconstruction goals**: In [83] and [24], the goal is to resolve all the non-zero components with high probability. However, the Unicolor algorithm in [114] resolves a constant (close to 1) fraction of non-zero components. If any missing non-zero component has large magnitude the Unicolor algorithm leads to catastrophic errors.

- **Differing measurement ensembles (sparsity structure) leading to differing number of measurements**: In [83], the appearance of each edge in the bipartite graph is i.i.d.. As mentioned earlier, $\mathcal{O}(k\log(k))$ measurements are required due to what happens at the tail end of a "coupon collection" process. Our work refines by tweaking tail measurements (in three separate phases – see Section 6.2 for intuition and Section 6.4 for details) to get information-theoretically optimal $\mathcal{O}(k)$ measurements, at slight cost of decoding complexity (see the next point). The algorithm in [114] chooses not to tweak tail measurements, hence gets $\mathcal{O}(k)$ measurements and pays a penalty of potentially catastrophic errors (see the previous point). The bipartite graph used in [114] is left-regular and density evolution techniques are used to optimize constant factors in the number of measurements over [24].

- **Differing measurement matrix ensembles (entries) leading to differing decoding complexities**: In [83], decoding complexity is dominated by the support-recovery module, which isn't state-of-the-art. The algorithms in [24] and [114][9] have more nuanced choice of measurements ensembles (involving carefully chosen specific measurements that speed up the recovery modules as in [10], and allow the

[9]The number of measurements in [114] decreases by a factor of more than 5/4 compared to that in [24], due to a more careful choice of structured measurements and density evolution to carefully choose the corresponding degree distribution of nodes.

non-linear measurements models to be "linearized" in [24] (see Section 6.6.2 for details)) leading to significant savings in decoding complexity from $\mathcal{O}(nk\log(k))$ in [83] to $\mathcal{O}(k\log(k))$ in [24]. Further reductions in decoding complexity to $\mathcal{O}(k)$ in [114] results from the fact that they have differing reconstruction goals from [24, 83]. Having seen the results of [114] (which was published after [24], but before this manuscript), we note that if we also relaxed our reconstruction goal to resolving most (but not all) of the non-zero components, we too could get both decoding complexity and a number of measurements scaling as $\mathcal{O}(k)$ (see Remark 11 in Section 6.7 for that).

- **Utility for applications**: One nice thing in [114] is that the authors also apply the techniques in [113] to generalize their algorithms in the scenario where measurements are "Fourier-friendly" – see [114] for details. Therefore, the algorithms are more practical for applications such as X-ray crystallography and optics [100] than the ones we propose.

6.4 Graph properties

We construct a series of bipartite graphs with some desirable properties outlined in this section. We then use the structure of the bipartite graphs to generate our measurement matrix A in Section 6.5 and design the corresponding reconstruction algorithm in Section 6.6. Each left node of a bipartite graph represents a component of $\mathbf{x}$, and each right node represents a set of intensity measurements.

6.4.1 Seeding Phase

The properties of the bipartite graph, $\mathcal{G}_I$, in the first phase are as follows:

1. There are n left nodes and ck right nodes, where c is a constant.

2. Each edge in $\mathcal{G}_I$ appears with probability $1/k$. For each right node, the degree, in expectation, is n/k.

3. Each edge in $\mathcal{G}_I$ is assigned different weights, as discussed in the measurement design (see Section 6.5).

4. Many singleton nodes: Singleton nodes are right nodes which involves exactly one non-zero component of $\mathbf{x}$. Singleton nodes help to recover the magnitude of non-zero component. See Section 6.8 for details.

5. Many resolvable doubleton nodes: Doubleton nodes are right nodes which involve exactly two non-zero components of $\mathbf{x}$. Resolvable doubletons are the doubletons which involve exactly two non-zero components whose magnitudes are recovered by singleton nodes. See Section 6.8 for details.

We also consider a graph $\mathcal{H}$, which is implied by $\mathcal{G}_I$ and is defined as follows. Each vertex in $\mathcal{H}$ represents a non-zero component of $\mathbf{x}$ and there is an edge in $\mathcal{H}$ if and only if two left nodes involved are mutually resolved by a resolvable doubleton node. We prove that the $\mathcal{H}$ satisfies the following property. $\mathcal{H}$ has a "giant" connected component $\mathcal{H}'$ that contains a constant fraction of nodes from $\mathcal{H}$. This property is formally stated in Section 6.8. See Figures 6.1 and 6.2 for illustration.

6.4.2 Geometric-decay phase

There are $\mathcal{O}(\log(\log(k)))$ separate bipartite graphs/stages in this phase.

The properties of the l-th bipartite graph, $\mathcal{G}_{II,l}$ ($l = 1, 2, \ldots, L = \mathcal{O}(\log(\log(k)))$), are as follows:

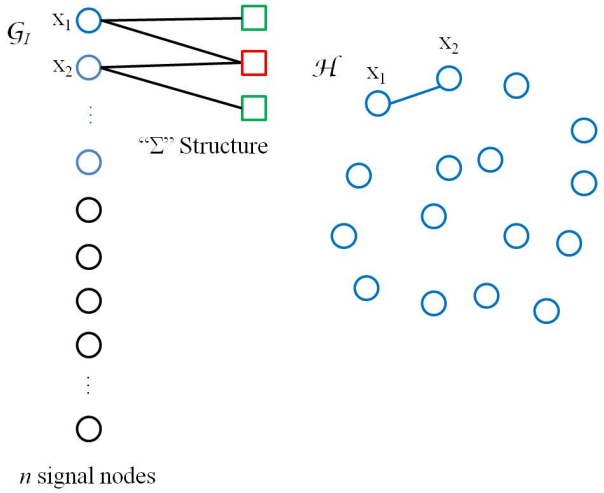


Figure 6.1: Implied Graph: $\mathcal{G}_I$ is a bipartite graph with n left nodes. Blue nodes and black nodes represent k (16 in this example) non-zero components and $n-k$ zero components, respectively. Green nodes and red nodes on the right represent singletons and doubletons, respectively. A graph $\mathcal{H}$ is implied by $\mathcal{G}_I$. Each vertex in $\mathcal{H}$ corresponds to a non-zero component of $\mathbf{x}$. There exists an edge in $\mathcal{H}$ if and only if the corresponding two non-zero components of $\mathbf{x}$ are mutually resolved. For example, in graph $\mathcal{G}_I$, x_1 and x_2 are connected by two singletons and one doubleton. The magnitudes of x_1 and x_2 are recovered by these two singletons. Therefore, the doubleton becomes a resolvable doubleton and x_1 and x_2 are mutually resolved. Accordingly, in graph $\mathcal{H}$, x_1 and x_2 are connected.

1. There are n left nodes and $cf_{II,l-1}k$ right nodes, where $f_{II,l-1}$ is the expected fraction of unresolved non-zero components of $\mathbf{x}$ after the $(l-1)$-th stage of decoding process in the second phase. $f_{II,0} = f_I$ is the expected fraction of unresolved non-zero components after seeding phase. The 0-th stage of geometric-decay phase is called the seeding-phase. The value of $f_{II,l}$ is discussed in Section 6.7.

2. Each edge in $\mathcal{G}_{II,l}$ appears with probability $1/\left(f_{II,l-1}k\right)$.

3. Each edge in $\mathcal{G}_{II,l}$ is assigned different weights, as discussed in the

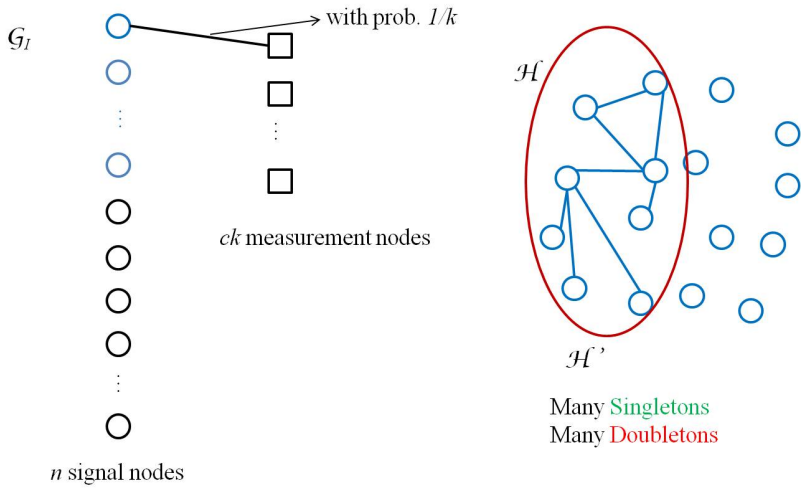


Figure 6.2: Seeding Phase: $\mathcal{G}_I$ is the bipartite graph with n left nodes and ck right nodes where c is a constant. Each edge in $\mathcal{G}_I$ appears with probability $1/k$. We prove that, in Section 6.8, there are many singleton and doubleton nodes in $\mathcal{G}_I$ and there is a "giant" connected component $\mathcal{H}'$ in $\mathcal{H}$. $\mathcal{H}'$ contains a constant fraction (1/2 in this example) of nodes from $\mathcal{H}$.

measurement design (see Section 6.5).

4. Many resolvable multiton nodes: Resolvable multiton nodes are right nodes which involve exactly one unresolved non-zero component of $\mathbf{x}$ and at least one of the resolved non-zero components. Each resolvable multiton node helps to recover both the magnitude and the relative phase of the corresponding unresolved non-zero component via "Cancelling out" process (see Section 6.6).

For a newly resolved non-zero component, the corresponding node in $\mathcal{H}$ is appended to the giant connected component, $\mathcal{H}'$. In expectation, there are $(f_{II,l-1} - f_{II,l})\,k$ non-zero components decoded in the l-th stage of decoding. We show in Section 6.8 that we are able to reconstruct a constant fraction of undecoded non-zero components with high probability at each stage. After $\mathcal{O}(\log(\log(k)))$ stages, there are $\mathcal{O}(k/\log(k))$ unresolved non-

zero components of $\mathbf{x}$ left. See Figure 6.3 for illustration.

6.4.3 Cleaning-up phase

The properties of the bipartite graph, $\mathcal{G}_{III}$, in the last phase are as follows:

1. There are n left nodes and $c\,(k/\log(k))\log\,(k/\log(k)) = \mathcal{O}(k)$ right nodes.
2. Each edge in $\mathcal{G}_{III}$ appears with probability $\log(k)/k$.
3. Each edge in $\mathcal{G}_{III}$ is assigned different weights, as discussed in the measurement design (see Section 6.5).
4. Many resolvable multiton nodes.

In this stage, all the resolved non-zero components of size $\mathcal{O}(k/\log(k))$ are finally recovered using resolvable multiton nodes by the "Cancelling out" process and a Coupon Collection argument. See Figure 6.4 for illustration.

6.5 Measurement Design

For each of the bipartite graphs described in the previous section ($\mathcal{G}_I$, $\mathcal{G}_{II,l}$'s and $\mathcal{G}_{III}$), we design a corresponding measurement matrix. In the following, we describe the design of the measurement matrix $A(\mathcal{G})$ for an arbitrary bipartite graph $\mathcal{G}$ with n right nodes and $m'_{\mathcal{G}}$ left nodes. This design is then used to generate measurement matrices for each of the graphs $\mathcal{G}_I$, $\mathcal{G}_{II,l}$'s and $\mathcal{G}_{III}$. Let $A'(\mathcal{G})$ be the dimension-$m'_{\mathcal{G}} \times n$ adjacency matrix of $\mathcal{G}$ where the entry at i-th row and j-th column equals to 1 if and only if i-th right node connects to the j-th left node for $j \in [n]$ and $i \in \left[m'_{\mathcal{G}}\right]$. The dimension-$m_{\mathcal{G}} \times n$ phase measurement matrix $A(\mathcal{G})$ is designed based on

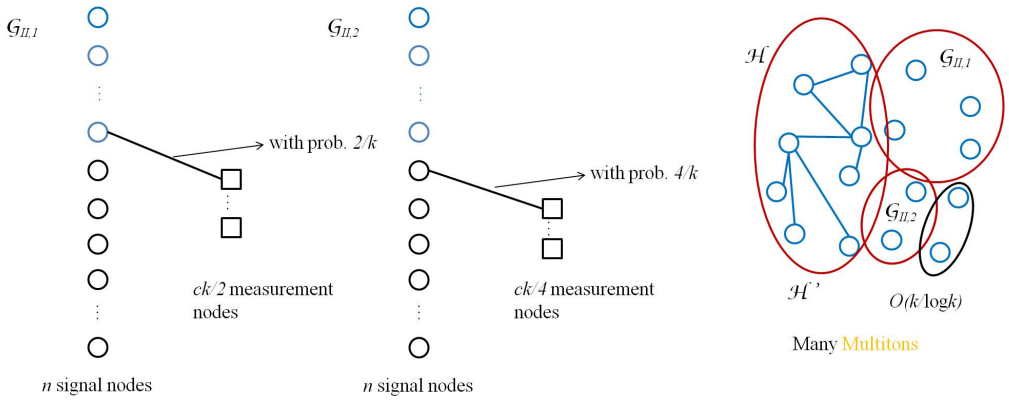


Figure 6.3: Geometric-decay phase: There are $\mathcal{O}(\log(\log(k)))$ (2 in this example) separate bipartite graphs in this phase. The number of right nodes decreases geometrically (from $ck/2$ to $ck/4$) and the density of graphs increases geometrically (from $2/k$ to $4/k$). We prove, in Section 6.8, that there are a significant fraction of measurements in each stage leading to resolvable multitons and the number of unresolved components decays geometrically. At the end of geometric-decay phase, the number of unresolved components is $\mathcal{O}(k/\log(k))$.

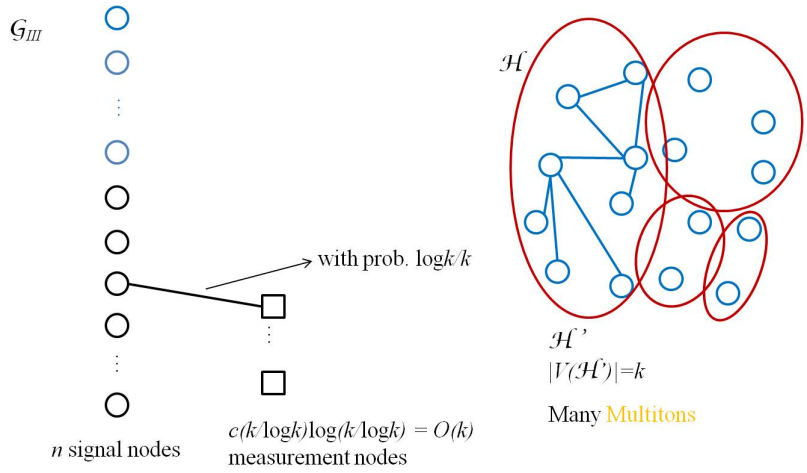


Figure 6.4: Cleaning-up Phase: $\mathcal{G}_{III}$ is a bipartite graph with n left nodes and $\mathcal{O}(k)$ right nodes. Each edge appears with probability $\log(k)/k$. We show, in Section 6.8, that all the remaining unresolved components (black part of $\mathcal{H}$ in Fig. 6.3) can be resolved using coupon collection argument.

Notation	Definition
$\mathcal{G}_I$	The bipartite graph used in the seeding phase with n left nodes and ck right nodes. Each edge appears with probability $1/k$.
$\mathcal{H}$	Implied graph by $\mathcal{G}_I$.
$\mathcal{H}'$	Giant connected component of $\mathcal{H}$.
$\mathcal{G}_{II,l}$	The l-th bipartite graph used in the l-th stage in geometric-decay phase with n left nodes and $cf_{II,l-1}k$ right nodes for $l \in [L]$. Each edge appears with probability $1/f_{II,l-1}k$.
f_I	The expected fraction of unresolved non-zero components of $\mathbf{x}$ after the seeding phase.
$f_{II,l}$	The expected fraction of unresolved non-zero components of $\mathbf{x}$ after the l-th stage of the geometric-decay phase. Let $f_{II,0} = f_I$.
$\mathcal{G}_{III}$	The bipartite graph used in the cleaning-up phase with n left nodes and $c\left(k/\log(k)\right)\log\left(k/\log(k)\right)$ right nodes. Each edge appears with probability $\log(k)/k$.

Table 6.2: Table of notation used in the design of bipartite graphs in this chapter.

$A'(\mathcal{G})$ where $m_{\mathcal{G}} = 5m'_{\mathcal{G}}$. By appending all the matrices $A(\mathcal{G})$ sequentially, we get the actual $m \times n$ measurement matrix A where $m = \Sigma_{\mathcal{G}} m_{\mathcal{G}}$. For i-th row $A'(\mathcal{G})_i$ of $A'(\mathcal{G})$, a set of rows (of size 5) of $A(\mathcal{G})$ are designed for $i \in \left[m'_{\mathcal{G}}\right]$. If the j-th entry of $A'(\mathcal{G})_i$ is zero, then corresponding set of entries of $A(\mathcal{G})$ are all zero for all $j \in [n]$. In the following measurement matrix design, we design the entries corresponding to non-zero entries in $A'(\mathcal{G})$. See Section 6.6 for how these measurements are used for decoding.

1. Trigonometric entries: The j-th entries of the $(5i-4)$-th and $(5i-3)$-th rows of $A(\mathcal{G})$ are denoted by $a_{i,j}^{(\mathcal{G},1)}$ and $a_{i,j}^{(\mathcal{G},2)}$. The values are set as follows:

$$a_{i,j}^{(\mathcal{G},1)} = \cos\left(\frac{j\pi}{2n}\right), and$$

$$a_{i,j}^{(\mathcal{G},2)} = \iota \sin\left(\frac{j\pi}{2n}\right),$$

where ι denotes the positive square root of -1 and $\pi/2n$ can be treated as the unit phase of the entry design. In particular, the phase $j\pi/2n$ will be critical for our algorithm. The first two entries are used in singleton node identification and "cancelling out" process of resolvable multiton nodes respectively.

2. Structured unit complex entries: The j-th entry of the $(5i-2)$-th row of $A(\mathcal{G})$ is denoted by $a_{i,j}^{(\mathcal{G},3)}$. The value is set as follows:

$$a_{i,j}^{(\mathcal{G},3)} = \exp\left(\iota\frac{j\pi}{2n}\right).$$

This type of measurement will be used only in "cancelling out" process of resolvable multiton nodes.

3. Unit entries: The j-th entry of the $(5i-1)$-th row of A is denoted by $a_{i,j}^{(\mathcal{G},4)}$. The value is set to be 1. This measurement is used in resolvable doubleton identification and "cancelling out" process of resolvable multiton nodes.

4. Random unit complex entries: The j-th entry of the $5i$-th row of A is denoted by $a_{i,j}^{(\mathcal{G},5)}$ used as verification. The value is set as follows:

$$a_{i,(j)}^{(\mathcal{G},5)} = \exp(\iota\phi_{i,j}),$$

where $\phi_{i,j}$ is chosen uniformly at random from $[0,\ \pi/2]$.[10] This measurement is used to resolve potential degeneracies arising from the

[10] If $\phi_{i,j}$ is chosen with $\Omega(\log(k))$ bits of precision, the error probability of verification and resolving degeneracy (see relative phase recovery part in Section 6.6) in a single step is at most $\mathcal{O}(1/\operatorname{poly}(k))$. Since the total number of times one needs to verify and resolve degeneracy is $\mathcal{O}(k)$, by applying the union bound over the decoding process, the probability of incorrect decoding is upper bounded by $\mathcal{O}(1/\operatorname{poly}(k))$.

solution of quadratic equations when resolvable multitons and resolvable doubletons are used for decoding. Also, it helps to verify our other measurements such as identification measurements, and the estimation of magnitude and relative phases.

Notation	Definition
$m'_{\mathcal{G}}$	The number of right nodes for the bipartite graph $\mathcal{G}$. $\mathcal{G}$ is one of $\mathcal{G}_I$, $\mathcal{G}_{II,l}$ for $l \in [L]$, and $\mathcal{G}_{III}$.
$A'(\mathcal{G})$	The dimension-$m'_{\mathcal{G}} \times n$ adjacent matrix of $\mathcal{G}$.
$A'(\mathcal{G})_i$	The i-th row of matrix $A(\mathcal{G})'$ for $i \in [m'_{\mathcal{G}}]$.
$A(\mathcal{G})$	The dimension-$m_{\mathcal{G}} \times n$ measurement matrix generated by $A(\mathcal{G})'$. Here $m_{\mathcal{G}} = 5m'_{\mathcal{G}}$.
A	The dimension-$m \times n$ phase measurement matrix generated by all $A(\mathcal{G})$'s. Here, $m = \Sigma_{\mathcal{G}} m_{\mathcal{G}}$.
$a_{i,j}^{(\mathcal{G},q)}$	The j-th entry of the $[5(i-1)+q]$-th the rows of $A(\mathcal{G})$. Here, $i \in [m_{\mathcal{G}}]$, $j \in [n]$, and $q \in [5]$.

Table 6.3: Table of notation for measurements design

6.6 Reconstruction Algorithm

Let $y_i^{(\mathcal{G},q)}$ denote the $[5(i-1)+q]$-th measurement generated by $A(\mathcal{G})$. Here, $\mathcal{G}$ is one of the $\mathcal{G}_I$, $\mathcal{G}_{II,l}$'s and $\mathcal{G}_{III}$, $i \in [m_{\mathcal{G}}]$, and $q \in [5]$.

6.6.1 Seeding phase

Overview

1. Preprocessing: Each right node is attached to a list to record its neighbours (left nodes) in the decoding process.

2. Singleton Identification and Magnitude Recovery: Check every right node to see whether it is a singleton node or not. If so, we locate the corresponding non-zero component and measure its magnitude.

3. Doubleton Identification: After decoding the non-zero components (only their magnitudes till this step), each list of their neighbours' (right nodes') is inserted the indices of the decoded non-zero components.[11] The right nodes with the length-2 lists are identified as potential resolvable doubletons. Later, we use verification measurements to figure out actual resolvable doubletons. The reason why we need the verification step is that the potential resolvable doubletons may involve other non-zero components which have not been resolved yet.

   So far, we decode the magnitudes of a constant fraction of all the non-zero components, and also identify the locations of these non-zero components. We also identify potential resolvable doubletons by checking whether the list affiliated with right nodes are of size exactly 2, and the actual resolvable doubletons (by using verification measurements).

4. Relative Phase Recovery: Each resolvable doubleton is used to resolve the phase between the two non-zero components whose locations lie in the neighbour list of the resolvable doubleton.

   Breadth First Search (BFS) or Depth First Search (DFS) [135] algorithm allow us to explore a spanning forest for graph $\mathcal{H}$ computationally efficiently, in time $\mathcal{O}(k)$. We only care about the largest

[11] We do the insertion only if the length of the list is no larger than one. For the list whose length is 3 after insertion, it will be discarded and won't be considered in the following iteration since it definitely is not a doubleton.

connected component, $\mathcal{H}'$. After this step, any pair of nodes in $\mathcal{H}'$ are mutually resolved.

The formal description of reconstruction algorithm

1. Initialization: We initialize by setting the signal estimate vector $\hat{\mathbf{x}}$ to all-zeros vector $\mathbf{0^n}$. Each right node $i \in \left[m'_{\mathcal{G}_I}\right]$ is attached to an empty neighbour list $\mathcal{M}(i)$. Let $\mathcal{D}$ denote a list of the resolvable doubletons. Initially, $\mathcal{D}$ is empty. Set $i = 1$.

2. Singleton Identification and Magnitude Recovery:

   Compute the ratio of Trigonometric measurements:

$$s_i = \frac{\arctan\left(\frac{y_i^{(\mathcal{G}_I,2)}}{y_i^{(\mathcal{G}_I,1)}}\right)}{\frac{\pi}{2n}}.$$

   If s_i is not an integer, increment i by 1 and start a new iteration. If s_i is an integer, we do the following steps:

   (a) Singleton Identification: We tentatively identify that i is a singleton.

   (b) Magnitude Estimation: Assume that the s_i-th entry of $\mathbf{x}$ is non-zero and

$$|\hat{x}_{s_i}| = \begin{cases} \frac{y_i^{(\mathcal{G}_I,1)}}{a_{i,s_i}^{(\mathcal{G}_I,1)}} & \text{if } a_{i,s_i}^{(\mathcal{G}_I,1)} \neq 0 \\ \iota\frac{y_i^{(\mathcal{G}_I,2)}}{a_{i,s_i}^{(\mathcal{G}_I,2)}} & \text{if } a_{i,s_i}^{(\mathcal{G}_I,2)} \neq 0. \end{cases}$$

   (c) Verification: If $|\hat{x}_{s_i}| \neq \left|y_i^{(\mathcal{G}_I,5)}\right|$, the verification fails. We increment i by 1 and go back to step a) to start a new iteration. If verification passes, we do the following steps:

      i. Updating neighbour List: s_i is appended to the neighbour lists of all its neighbours. For $i \in \left[m'_{\mathcal{G}_I}\right]$, it is no longer

considered in the later process if $|\mathcal{M}(i)| \geq 3$ since in the next step we only care about doubleton whose neighbour list size equals 2.

ii. Increment i by 1 and go back to step a) to start a new iteration.

3. Doubleton Identification: For each i whose neighbour list is of size 2, it is appended to the resolvable doubleton list $\mathcal{D}$ where $\mathcal{M}(i)[1]$ and $\mathcal{M}(i)[2]$ are the two indices of non-zero components whose magnitudes have been recovered.

4. Relative Phase Recovery:

   To compute connected component of $\mathcal{H}$, Breadth first search or depth first search for adjacent list representation of $\mathcal{H}$ is applied in this step. For each $i \in \mathcal{D}$, the elements in $\mathcal{M}(i)$ tell which two vertices in $\mathcal{H}$ are connected. BFS or DFS outputs connected components of graph $\mathcal{H}$. We run the BFS or DFS, for each edge in $\mathcal{H}$, with additional steps stated below:

   (a) Relative Phase Estimation: Suppose i's two neighbours are denoted by $\mathcal{M}(i)[1]$ and $\mathcal{M}(i)[2]$. The fourth measurement is used to derive the phase between $\mathcal{M}(i)[1]$-th and $\mathcal{M}(i)[2]$-th components of $\mathbf{x}$, $\theta = \left|\theta_{\mathcal{M}(i)[1]} - \theta_{\mathcal{M}(i)[2]}\right|$, by Law of Cosine.[12]

   (b) Resolving Degeneracy and Verification: The verification measurement helps to resolve the degeneracy of sign of θ (*i.e.*, whether θ or $-\theta$ is the actual phase difference we are interested in.) by

[12] Given the lengths of two complex number A and B, we can deduce the phase between A and B, Δ, by Law of Cosine if we also know the length of $A + B$. To be more explicit, $-\cos\Delta = \frac{|A|^2+|B|^2-|A+B|^2}{2|A||B|}$.

checking whether

$$\left|\left|\hat{\mathbf{x}}_{\mathcal{M}(i)[1]}\right| \exp\left(\iota\phi_{i,\mathcal{M}(i)[1]}\right) + \left|\hat{\mathbf{x}}_{\mathcal{M}(i)[2]}\right| \exp\left(\iota\phi_{i,\mathcal{M}(i)[2]} + \iota\theta\right)\right| = \left|y_i^{(\mathcal{G}_I,5)}\right|$$

or

$$\left|\left|\hat{\mathbf{x}}_{\mathcal{M}(i)[1]}\right| \exp\left(\iota\phi_{i,\mathcal{M}(i)[1]}\right) + \left|\hat{\mathbf{x}}_{\mathcal{M}(i)[2]}\right| \exp\left(\iota\phi_{i,\mathcal{M}(i)[2]} - \iota\theta\right)\right| = \left|y_i^{(\mathcal{G}_I,5)}\right|.$$

If neither of the above equations holds, then i is not a resolvable doubleton. Namely, there is no edge between $\mathcal{M}(i)[1]$-th and $\mathcal{M}(i)[2]$-th components of $\mathbf{x}$.

For the first node in a connected component, its phase is set to be zero. When the BFS or DFS terminates, we can find the largest connected component of $\mathcal{H}$, $\mathcal{H}'$. For all the node pairs in $\mathcal{H}'$, they are mutually resolved.

6.6.2 Geometric-decay and Cleaning-up phases

Claim 1. *("Cancelling out" Process) For a bipartite graph $\mathcal{G}$ in geometric-decay phase or cleaning-up phase, if a right node i is a resolvable multiton node, it involves exactly one (unknown) undecoded non-zero component, $\mathbf{x}_j$, and at least one (known) resolved non-zero components. Then, we are able to find the location of $\mathbf{x}_j$, j, and resolve $\mathbf{x}_j$ (both magnitude and relative phase).*

Proof. Please refer to Appendix A.2.1. ■

Note that if "cancelling out" fails (*i.e.,* none of the pairs of j and $\mathbf{x}_j$ satisfies the last equation in the proof), then i is not a resolvable multiton. In each stage at geometric-decay phase and cleaning-up phase, we go through

all the right nodes, find resolvable multitons and use them to recover unresolved non-zero components by the "cancelling out" process. For a newly resolved component of $\mathbf{x}$, the corresponding node in $\mathcal{H}$ is appended to $\mathcal{H}'$. In the end, the size of the node set of $\mathcal{H}'$ should be k.

Notation	Definition
$y_i^{(\mathcal{G},q)}$	The $[5(i-1)+q]$-th intensity measurement generated by measurement matrix $A(\mathcal{G})$. Here, $i \in [m_{\mathcal{G}}]$, and $q \in [5]$.
$\mathcal{D}$	Resolvable doubleton list used in the seeding phase.
$\mathcal{M}(i)$	The neighbour list for i-th node in $\mathcal{G}_I$ for $i \in [m_{\mathcal{G}_I}]$.

Table 6.4: Table of notation for measurements design

6.7 Choice of Parameters

All the parameters designed in this section are calculated based on expectation. The actual performance of our algorithm will be discussed in Section 6.8.

6.7.1 Seeding phase

Magnitude Recovery by singletons

- The probability of a right node being a singleton node is given by

$$\begin{aligned} P_S &= \binom{k}{1}\frac{1}{k}\left(1-\frac{1}{k}\right)^{k-1} \\ &= \left(1-\frac{1}{k}\right)^{k-1} \\ &\doteq e^{-1}. \end{aligned}$$

- The expected number of singletons is equal to $ck \times P_S \doteq e^{-1}ck$.

- The expected number of different non-zero components whose magnitudes are recovered is given by the following lemma.

Lemma 12. *(Generalized coupon collection) Given V different coupons and $V\log\left(\frac{V}{V-U}\right)$ picks with repetition ($U < V$), the expected number of different coupons picked is U for $V \to +\infty$. With probability at least $1 - 2\exp\left(-\frac{2\epsilon^2(V-U)U}{V}\right)$, the number of different coupons picked is between $(1-\epsilon)U$ and $(1+\epsilon)U$ for any $\epsilon > 0$.*

Proof. Please refer to Appendix A.2.2. ■

By Lemma 12 (let $V = k$ and $V\log\left(\frac{V}{V-U}\right) = ck \times P_S$), we know that the expected number of non-zero components of x whose magnitudes are recovered is $k\left(1 - e^{-cP_S}\right)$.

Relative Phase Recovery by resolvable doubletons

- The probability of a right node being doubleton is given by

$$\begin{aligned} P_D &= \binom{k}{2}\left(\frac{1}{k}\right)^2\left(1-\frac{1}{k}\right)^{k-2} \\ &= \frac{1}{2}\left(1-\frac{1}{k}\right)^{k-1} \\ &\doteq \frac{e^{-1}}{2}. \end{aligned}$$

- The expected number of doubletons is equal to $ck \times P_D \doteq e^{-1}ck/2$.

- The expected number of resolvable doubletons is given by the following statement.

Note that only the doubleton which involves two non-zero components whose magnitudes have been recovered is useful to recover the relative phase.

$$\begin{aligned} \#\text{ resolvable doubletons} &= \frac{\binom{k\left(1-e^{-cP_S}\right)}{2}}{\binom{k}{2}} \times ckP_D \\ &\doteq \frac{\left(1-e^{-cP_S}\right)^2 cke^{-1}}{2}. \end{aligned}$$

- The expected number of different pairs of components whose relative phase is recovered by resolvable doubletons is given by Lemma 12.

Given $k\left(1-e^{-cP_S}\right)$ nodes and $\left(1-e^{-cP_S}\right)^2 ckP_D$ edges with repetition in $\mathcal{H}$, there are

$$\left(1+\mathcal{O}(1/k)\right)\left(1-e^{-cP_S}\right)^2 ckP_D$$

distinct edges.

The giant connected component

Theorem 4. *[20] For a random graph $\mathcal{G}_{N,M}$ with N nodes and M edges chosen at random among the $\binom{N}{2}$ possible edges. Let $\mathcal{Z}_{N,M}$ denote the size of the greatest component of $\mathcal{G}_{N,M}$. If $r = 2M/N > 1$, we have for any $\epsilon > 0$*

$$\Pr\left(\left|\frac{\mathcal{Z}_{N,M}}{N} - \beta\right| < \epsilon\right) = 1 - \mathcal{O}\left(\frac{1}{\epsilon^2 N}\right),$$

where β is the unique solution to $\beta + \exp(-\beta r) = 1$.

We need to find the size of giant connected component of a random graph with $k\left(1-e^{-cP_S}\right)$ nodes and $\left(1-e^{-cP_S}\right)^2 ckP_D$ edges (with repetition) and therefore $\left(1+\mathcal{O}(1/k)\right)\left(1-e^{-cP_S}\right)^2 ckP_D$ distinct edges (implied by Lemma 12). Let's say the size is $(1-f_I)\,k$ where f_I is the function of c.

By Theorem 4, when $2\left(1-e^{-cP_S}\right)cP_D > 1$, the giant connected component exists (this inequality holds when constant c is large enough) and

the size of the giant component is $(1-f_I)\,k = \beta_c k\left(1-e^{-cP_S}\right)$ where β_c is the unique solution to $\beta + \exp\left[-\beta \cdot 2\left(1-e^{-cP_S}\right)cP_D\right] = 1$.

6.7.2 Geometric-decay phase:

Let $f_{II,l}$ denote the expected fraction of unresolved non-zero components after the l-th stages in this phase. Let $f_I = f_{II,0}$.

Stage $l+1$ $(0 \le l \le L-1)$

- The probability that a right node being a resolvable multiton is given by

$$\begin{aligned} & P_M^{(II,l+1)} \\ = \; & \binom{f_{II,l}k}{1}\frac{1}{f_{II,l}k}\left(1-\frac{1}{f_{II,l}k}\right)^{f_{II,l}k-1}\left[1-\left(1-\frac{1}{f_{II,l}k}\right)^{\left(1-f_{II,l}\right)k}\right] \\ = \; & \left(1-\frac{1}{f_{II,l}k}\right)^{f_{II,l}k-1} - \left(1-\frac{1}{f_{II,l}k}\right)^{k-1} \\ \doteq \; & e^{-1} - e^{-\frac{1}{f_{II,l}}}. \end{aligned}$$

- The expected number of resolvable multitons is equal to $cf_{II,l}kP_M^{(II,l+1)}$.

- The expected number of non-zero components which are resolved (both magnitude and phase) is given by Lemma 12.

Let

$$f_{II,l}k\log\left(\frac{f_{II,l}k}{f_{II}k-(f_{II,l}-f_{II,l+1})k}\right) = cf_{II,l}kP_M^{(II,l+1)},$$

we know that $f_{II,l} - f_{II,l+1} = f_{II,l}\left(1-e^{-cP_M^{(II,l+1)}}\right)$.

Therefore, $f_{II,l+1} = e^{-cP_M^{(II,l+1)}}f_{II,l}$. We can compute the value of $f_{II,l}$ recursively.

Note that $P_M^{(II,l)}$ increases as l increases. So, $P_M^{(II,l)}$ is bounded by $e^{-1}-e^{-\frac{1}{f_I}}$ $(l=0)$ and e^{-1} $(l=+\infty)$.

End of this phase

There are $\mathcal{O}(\log(\log(k)))$ stages in the geometric-decay phase. We already show that in each step we expect to recover constant fraction of remaining unresolved non-zero components. In the end of this phase, the number of unresolved non-zero components is $\mathcal{O}(k/\log(k))$.

Remark 11. If we release our goal to resolve only a constant fraction of non-zero components, we may only introduce constant number of steps in the geometric-decay phase and remove the following clear-up phase. In this way, we are able to mutually resolve constant fraction of non-zero components and the number of measurements required remains $\mathcal{O}(k)$ but decoding complexity reduces from $\mathcal{O}(k\log(k))$ to $\mathcal{O}(k)$.

6.7.3 Cleaning-up phase

Recall that, in this phase, each edges appears with probability $\log(k)/k$ and there are $c\left(k/\log(k)\right)\log\left(k/\log(k)\right) = \mathcal{O}(k)$ right nodes in $\mathcal{G}_{III}$.

6.8 Performance of the algorithm (Proof of the Main Theorem)

Number of measurements:

In Section 6.7, we showed that $f_{II,l+1} = e^{-cP_M^{(II,l+1)}} f_{II,l}$. And $P_M^{(II,l)}$ increases as l increases. Therefore,

$$\begin{aligned} f_{II,l+1} &= \exp\left[-cP_M^{(II,l+1)}\right] f_{II,l} \\ &= \exp\left[-c\Sigma_{t=1}^{l+1} P_M^{(II,t)}\right] f_{II,0} \\ &= \exp\left[-c\Sigma_{t=1}^{l+1} P_M^{(II,t)}\right] f_I \\ &\leq \exp\left[-c(l+1)P_M^{(II,1)}\right] f_I. \end{aligned}$$

Using the above, we can bound the total number of measurements in the three phases as

$$\begin{aligned}
&\underbrace{ck}_{Seeding} + \underbrace{\Sigma_{l=1}^{L} cf_{II,l-1}k}_{geometric-decay} + \underbrace{c(k/\log(k))\log(k/\log(k))}_{cleaning-up} \\
&= \mathcal{O}(k) + \left(\Sigma_{l=1}^{L} cf_{II,l-1}\right) k + \mathcal{O}(k) \\
&\leq \mathcal{O}(k) + \left(\Sigma_{l=1}^{L} \exp\left[-clP_M^{(II,1)}\right]\right) f_I ck \\
&= \mathcal{O}(k).
\end{aligned}$$

Decoding complexity:

Almost all the operations take constant time except for DFS in the seeding phase and "Cancelling out" process in the geometric-decay and cleaning-up phases.

For DFS, the time complexity is linear in the size of node set and edge set. Since there are k nodes and $\mathcal{O}(k)$ edges involved in the seeding phase, the time complexity is $\mathcal{O}(k)$.

For "Cancelling out" process, the time complexity depends on the number of resolved non-zero components which corresponds to the resolvable multiton (See Appendix A.2.1).

In the later stage/phase, more non-zero components are associated with a measurement. Since the number of measurements is $\mathcal{O}(k)$, it suffices to show that each measurement involves at most $\mathcal{O}(\log(k))$ non-zero components (even if they are unresolved) in the cleaning-up phase with probability at least $1 - o(1/k)$.

Let NZ be the number of non-zero components involved in a measurement in cleaning-up phase. By Chernoff bound (Theorem 1, Appendix C), for any $\epsilon_{NZ} \geq 0$, we have

$$\Pr\left[NZ \geq (1+\epsilon_{NZ})\, k \cdot \log(k)/k\right] \leq \exp\left(-\frac{\epsilon_{NZ}^2}{2+\epsilon_{NZ}^2} k \cdot \log(k)/k\right).$$

Therefore,

$$\Pr\left[NZ = \mathcal{O}(\log(k))\right] \geq 1 - \mathcal{O}(1/\operatorname{poly}(k)).$$

Thus, we know that the decoding complexity is at most $\mathcal{O}(k \cdot NZ) = \mathcal{O}(k \log(k))$ with probability at least $1 - k \cdot \mathcal{O}(1/\operatorname{poly}(k)) = 1 - o(1)$ by the union bound.

Correctness:

While we design our measurement matrices by considering the expected values of unresolved non-zero components, actual number of unresolved non-zero components may differ slightly. Here, "slightly" means that the actual number of resolved non-zero components in each phase/stage deviates from the expected value but it can be concentrated around expectation with high probability. In the following, we show that this deviation leads to only a small probability of failure.

Let g_I denote the actual fraction of unresolved non-zero components after seeding phase. Let $g_{II,l}$ denote the actual fraction of unresolved non-zero components after the l-th stage in geometric-decay phase. Let $g_{II,0} = g_I$.

Recall the following properties of the bipartite graphs for measurement design. In the seeding phase, each edge appears with probability $1/k$ and there are ck right nodes. In the geometric-decay phase, each edge appears with probability $1/f_{II,l}k$ and there are $cf_{II,l}k$ right nodes in $(l+1)$-th step for $l \geq 0$. In the cleaning-up phase, each edge appears with probability $\log(k)/k$ and there are $c(k/\log(k))\log(k/\log(k))$ right nodes.

6.8.1 Seeding Phase

Magnitude recovery

- By Chernoff bound, the probability that the number of singletons is larger than $(1+\epsilon_S)\,ck \times P_S$ or smaller than $(1-\epsilon_S)\,ck \times P_S$ is less than $2e^{-(\epsilon_S)^2 ckP_S/2} = \mathcal{O}\left(\exp\left(-\epsilon_S^2 k\right)\right)$ for any $\epsilon_S > 0$.

- By Lemma 12 and the union bound, we know that, for any $\epsilon_{DS} > 0$, the number of different non-zero components whose magnitudes are recovered is between

$$(1-\epsilon_{DS})\left[1-e^{-(1-\epsilon_S)cP_S}\right]k$$

and

$$(1+\epsilon_{DS})\left[1-e^{-(1+\epsilon_S)cP_S}\right]k$$

with probability

$$1-\mathcal{O}\left[\exp\left(-\epsilon_{DS}^2 k\right)+\exp\left(-\epsilon_S^2 k\right)\right].$$

Note that $(1+\beta_S)\left[1-e^{-(1+\epsilon_S)cP_S}\right]k$ and $(1-\beta_S)\left[1-e^{-(1-\epsilon_S)cP_S}\right]k$ scale as

$$\left(1-e^{-cP_S}\right)\left[1+(\epsilon_{DS}+\epsilon_S)+o(\epsilon_{DS}+\epsilon_S)\right]k$$

and

$$\left(1-e^{-cP_S}\right)\left[1-(\epsilon_{DS}+\epsilon_S)-o(\epsilon_{DS}+\epsilon_S)\right]k,$$

respectively.

Relative phase recovery

- By Chernoff bound, the probability that the number of doubletons is larger than $(1+\epsilon_D)\,ck \times P_D$ or smaller than $(1-\epsilon_D)\,ck \times P_D$ is less than $2e^{-(\epsilon_D)^2 ckP_D/2} = \mathcal{O}\left(\exp\left(-\epsilon_D^2 k\right)\right)$ for any $\epsilon_D > 0$.

- The resolvable doubletons

$$\begin{aligned} \#\text{ resolvable doubletons } &\leq \frac{\binom{\left(1+\epsilon_{DS}\right)\left[1-e^{-\left(1+\epsilon_S\right)cP_S}\right]k}{2}}{\binom{k}{2}} \\ &\quad \times \left(1+\epsilon_D\right)\left(1+\epsilon_{RD}\right)ckP_D \end{aligned}$$

and

$$\begin{aligned} \#\text{ resolvable doubletons } &\geq \frac{\binom{\left(1-\epsilon_{DS}\right)\left[1-e^{-\left(1-\epsilon_S\right)cP_S}\right]k}{2}}{\binom{k}{2}} \\ &\quad \times \left(1-\epsilon_D\right)\left(1-\epsilon_{RD}\right)ckP_D \end{aligned}$$

with probability

$$1-\mathcal{O}\left[\exp\left(-\epsilon_{DS}^2k\right)+\exp\left(-\epsilon_S^2k\right)+\exp\left(-\epsilon_D^2k\right)+\exp\left(-\epsilon_{RD}^2k\right)\right],$$

for any $\epsilon_{RD} > 0$, by Chernoff bound and the union bound. Again, the upper bound and the lower bound on the number of resolvable doubletons scale as

$$\left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}\right)\right]\left(1-e^{-cP_S}\right)^2ckP_D$$

and

$$\left[1-\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}\right)\right]\left(1-e^{-cP_S}\right)^2ckP_D,$$

respectively.

- Number of distinct edges in the giant component: By Lemma 12 and the union bound, for any $\epsilon_{DRD} > 0$, with probability

$$\begin{aligned} 1-\mathcal{O}\left[\exp\left(-\epsilon_{DS}^2k\right)+\exp\left(-\epsilon_S^2k\right)+\exp\left(-\epsilon_D^2k\right)+\exp\left(-\epsilon_{RD}^2k\right)\right. \\ \left.+\exp\left(-\epsilon_{DRD}^2k\right)\right], \end{aligned}$$

the number of pairs of relative phase resolved by all the resolvable doubletons will be bounded from above by

$$(1+\epsilon_{DRD})\left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}\right)/k\right]\cdot$$
$$\left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}\right)\right]\left(1-e^{-cP_S}\right)^2 ckP_D,$$

and bounded from below by

$$(1-\epsilon_{DRD})\left[1-\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}\right)/k\right]\cdot$$
$$\left[1-\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}\right)\right]\left(1-e^{-cP_S}\right)^2 ckP_D,$$

which scale as

$$\left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}\right)\right]\left(1-e^{-cP_S}\right)^2 ckP_D$$

and

$$\left[1-\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}\right)\right]\left(1-e^{-cP_S}\right)^2 ckP_D.$$

The giant connected component

Let $N^+$and N^- be the upper and lower bounds respectively on the number of nodes in the giant component.

Let M^+ and M^- be the upper bound and lower bound on the number of edges in giant component. Then, $r^+ = 2M^+/N^-$ is the upper bound on twice the size of edges over size of nodes and the $r^- = 2M^-/N^+$ is the lower bound. β_c^+ and β_c^- are the solution to the equation $\beta + \exp\left(-\beta r^+\right) = 1$ and $\beta + \exp\left(-\beta r^-\right) = 1$.

We know that

$$\begin{aligned} r^+ &= 2M^+/N^- \\ &= 2\left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}\right)\right]\left(1-e^{-cP_S}\right)^2 ckP_D \\ &\quad /\left(1-e^{-cP_S}\right)\left[1-\mathcal{O}(\epsilon_{DS}+\epsilon_S)\right]k \\ &= 2\left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}\right)\right]\left(1-e^{-cP_S}\right)cP_D \end{aligned}$$

and

$$\begin{aligned} r^- &= 2M^-/N^+ \\ &= 2\left[1-\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}\right)\right]\left(1-e^{-cP_S}\right)^2 ckP_D \\ &\quad /\left(1-e^{-cP_S}\right)\left[1+\mathcal{O}(\epsilon_{DS}+\epsilon_S)\right]k \\ &= 2\left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}\right)\right]\left(1-e^{-cP_S}\right)cP_D. \end{aligned}$$

Since $r=-\log(1-\beta)/\beta$, $dr/d\beta=(-\log(1-\beta)/\beta)'_{\beta=\beta_c}$ is a constant. By Theorem 4,

$$\Pr\left(\left|\frac{\mathcal{Z}_{N,M}}{N}-\beta_c\right|\le\epsilon_{GC}\right) = \mathcal{O}\left(\frac{1}{\epsilon_{GC}^2 k}\right),$$

for any $\epsilon_{GC}>0$.

Therefore,

$$\beta_c^+=\left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}\right)\right](\beta_c+\epsilon_{GC})$$

and

$$\beta_c^-=\left[1-\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}\right)\right](\beta_c-\epsilon_{GC}).$$

The upper bound on the size of giant component is

$$\begin{aligned} N^+\beta_c^+ &= \left(1-e^{-cP_S}\right)\left[1+\mathcal{O}(\epsilon_{DS}+\epsilon_S)\right]k\cdot \\ &\quad \left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}\right)\right](\beta_c+\epsilon_{GC}) \\ &= \left[1+\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}+\epsilon_{GC}\right)\right]\beta_c\left(1-e^{-cP_S}\right)k \end{aligned}$$

and the lower bound on the size of giant component is

$$N^-\beta_c^- = \left[1-\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}+\epsilon_{GC}\right)\right]\beta_c\left(1-e^{-cP_S}\right)k,$$

with probability

$$1-\mathcal{O}\left[\exp\left(-\epsilon_{DS}^2k\right)+\exp\left(-\epsilon_S^2k\right)+\exp\left(-\epsilon_D^2k\right)+\exp\left(-\epsilon_{RD}^2k\right)\right.$$

$$+\exp\left(-\epsilon_{DRD}^2 k\right)+\mathcal{O}\left(1/\epsilon_{GC}^2 k\right)].$$

Recall that $(1-f_I)\,k=\beta_c\left(1-e^{-cP_S}\right)k$, we conclude that, with probability

$$1-\mathcal{O}\left[\exp\left(-\epsilon_{DS}^2 k\right)+\exp\left(-\epsilon_S^2 k\right)+\exp\left(-\epsilon_D^2 k\right)+\exp\left(-\epsilon_{RD}^2 k\right)\right.$$
$$\left.+\exp\left(-\epsilon_{DRD}^2 k\right)+\mathcal{O}\left(1/\epsilon_{GC}^2 k\right)\right],$$

there exists ϵ_I such that

$$(1-\epsilon_I)\,f_I\le g_I\le(1+\epsilon_I)\,f_I.$$

Here, ϵ_I scales as $\mathcal{O}\left(\epsilon_{DS}+\epsilon_S+\epsilon_D+\epsilon_{RD}+\epsilon_{DRD}+\epsilon_{GC}\right)$. Choose all the ϵ's to be $k^{-1/3}$. Then, ϵ_I scales as $\mathcal{O}\left(k^{-1/3}\right)$ with probability $1-\mathcal{O}\left(k^{-1/3}\right)$.

6.8.2 Geometric-decay Phase

Stage $l+1$ $(0\le l\le L-1)$

- The probability that a right node being a resolvable multiton:

$$\begin{aligned}
&Q_M^{(II,l+1)}\\
&\quad= \binom{g_{II,l}k}{1}\frac{1}{f_{II,l}k}\left(1-\frac{1}{f_{II,l}k}\right)^{g_{II,l}k-1}\left[1-\left(1-\frac{1}{f_{II,l}k}\right)^{(1-g_{II,l})k}\right]\\
&\quad= \frac{g_{II,l}}{f_{II,l}}\left[\left(1-\frac{1}{f_{II,l}k}\right)^{g_{II,l}k-1}-\left(1-\frac{1}{f_{II,l}k}\right)^{k-1}\right]\\
&\doteq \frac{g_{II,l}}{f_{II,l}}\left(e^{-\frac{g_{II,l}}{f_{II,l}}}-e^{-\frac{1}{f_{II,l}}}\right).
\end{aligned}$$

- The number of resolvable multitons is bounded by

$$\left(1+\epsilon_M^{(II,l+1)}\right)cf_{II,l}kQ_M^{(II,l+1)}$$

and

$$\left(1-\epsilon_M^{(II,l+1)}\right)cf_{II,l}kQ_M^{(II,l+1)}$$

with probability $1-\mathcal{O}\left(\exp\left(-\left[\epsilon_M^{(II,l+1)}\right]^2 f_{II,l}k\right)\right)$ for any $\epsilon_M^{(II,l+1)} > 0$.

- The number of non-zero components which are recovered (both magnitude and phase)

Let $\epsilon_{II,0} = \epsilon_I$. By Lemma 12, we know that

$$\begin{aligned} g_{II,l+1} - g_{II,l} &\leq \left(1+\epsilon_{DM}^{(II,l+1)}\right)\left[1-e^{-\left(1+\epsilon_M^{(II,l+1)}\right)cf_{II,l}Q_M^{(II,l+1)}/g_{II,l}}\right]g_{II,l} \\ &\doteq \left(1+\epsilon_{DM}^{(II,l+1)}\right)\left[1-e^{-\left(1+\epsilon_M^{(II,l+1)}\right)c\left(e^{-\frac{g_{II,l}}{f_{II,l}}}-e^{-\frac{1}{f_{II,l}}}\right)}\right]g_{II,l} \\ &\leq \left(1+\epsilon_{DM}^{(II,l+1)}\right)\left[1-e^{-\left(1+\epsilon_M^{(II,l+1)}\right)c\left(e^{-\left(1-\epsilon_{II,l}\right)}-e^{-\frac{1}{f_{II,l}}}\right)}\right]g_{II,l} \end{aligned}$$

and

$$\begin{aligned} g_{II,l+1} - g_{II,l} &\geq \left(1-\epsilon_{DM}^{(II,l+1)}\right)\left[1-e^{-\left(1-\epsilon_M^{(II,l+1)}\right)cf_{II,l}Q_M^{(II,l+1)}/g_{II,l}}\right]g_{II,l} \\ &\doteq \left(1-\epsilon_{DM}^{(II,l+1)}\right)\left[1-e^{-\left(1-\epsilon_M^{(II,l+1)}\right)c\left(e^{-\frac{g_{II,l}}{f_{II,l}}}-e^{-\frac{1}{f_{II,l}}}\right)}\right]g_{II,l} \\ &\geq \left(1-\epsilon_{DM}^{(II,l+1)}\right)\left[1-e^{-\left(1-\epsilon_M^{(II,l+1)}\right)c\left(e^{-\left(1-\epsilon_{II,l}\right)}-e^{-\frac{1}{f_{II,l}}}\right)}\right]g_{II,l}, \end{aligned}$$

with probability

$$1-\mathcal{O}\left(\exp\left(-\left[\epsilon_M^{(II,l+1)}\right]^2 f_{II,l}k\right)+\exp\left(-\left[\epsilon_{DM}^{(II,l+1)}\right]^2 g_{II,l}k\right)+k^{-1/3}\right)$$

by Lemma 12 and the union bound for any $\epsilon_{DM}^{(II,l+1)} > 0$.

We conclude there exists $\epsilon'_{II,l+1}$ such that

$$e^{-cP_M^{(II,l+1)}}\left(1-\epsilon'_{II,l+1}\right)g_{II,l} \leq g_{II,l+1}$$

$$\leq \quad e^{-cP_M^{(II,l+1)}} \left(1+\epsilon'_{II,l+1}\right) g_{II,l}.$$

Here $\epsilon'_{II,l+1}$ scales as $\mathcal{O}\left[\epsilon_{DM}^{(II,l+1)} + \epsilon_{M}^{(II,l+1)} + \epsilon_{II,l}\right]$.

Since $(1-\epsilon_{II,l})\, f_{II,l} \leq g_{II,l} = (1+\epsilon_{II,l})\, f_{II,l}$, we have

$$\begin{aligned} e^{-cP_M^{(II,l+1)}} \left(1-\epsilon'_{II,l+1}\right)(1-\epsilon_{II,l})\, f_{II,l} &\leq g_{II,l+1} \\ &\leq e^{-cP_M^{(II,l+1)}} \left(1+\epsilon'_{II,l+1}\right) \\ &\quad \cdot (1+\epsilon_{II,i})\, f_{II,l}. \end{aligned}$$

Since $f_{II,l+1} = e^{-cP_M^{(II,l+1)}} f_{II,l}$, we have

$$\begin{aligned} \left(1-\epsilon'_{II,l+1}\right)(1-\epsilon_{II,l})\, f_{II,l+1} &\leq g_{II,l+1} \\ &\leq \left(1+\epsilon'_{II,l+1}\right)(1+\epsilon_{II,l})\, f_{II,l+1}. \end{aligned}$$

Thus, we get that

$$\begin{aligned} (1-\epsilon_{II,l+1})\, f_{II,l+1} &\leq g_{II,l+1} \\ &\leq (1+\epsilon_{II,l+1})\, f_{II,l+1}. \end{aligned}$$

Here, $\epsilon_{II,l+1}$ scales as $\mathcal{O}\left[\epsilon_{DM}^{(II,l+1)} + \epsilon_{M}^{(II,l+1)} + 2\epsilon_{II,l}\right]$.

Choose $\epsilon_{DM}^{(II,l+1)}$ and $\epsilon_{M}^{(II,l+1)}$ to be $k^{-1/3}$ for all l. The error probability in each stage is $\mathcal{O}\left(k^{-1/3}\right)$ (which is the dominant term).

After L stages ($L = \mathcal{O}(\log(\log(k)))$),

$$(1-\epsilon_{II,L})\, f_{II,L} \leq g_{II,L} \leq (1+\epsilon_{II,L})\, f_{II,L} \tag{6.1}$$

holds with probability $1 - \mathcal{O}\left(\log(\log(k)) \cdot k^{-1/3}\right)$ by the union bound. Here $\epsilon_{II,L}$ scales as $\mathcal{O}\left(\log(k) \cdot k^{-1/3}\right)$ (since each stage $\epsilon_{II,l}$ doubles).

6.8.3 Cleaning-up phase

Theorem 5. *[Folklore](Coupon Collection) Let the random variable X denote the minimum number of trials for collecting each of the V types of coupons. Then, we have* $\Pr[X > \eta V \log(V)] \leq V^{-\eta+1}$ *for any* $\eta > 0$.

Proof. Please refer to Appendix A.2.3. ■

- The probability that a right node being a resolvable multiton:

$$Q_M^{III} \doteq \frac{g_{II,L}}{f_{II,L}} \left(e^{-\frac{g_{II,L}}{f_{II,L}}} - e^{-\frac{1}{f_{II,L}}} \right).$$

- $f_{III} = 1/\log(k)$. The number of resolvable multitons is lower bounded by

$$\left(1 - \epsilon_M^{III}\right) c f_{III} k \log\left(f_{III} k\right) Q_M^{III},$$

with probability $1 - \mathcal{O}\left(\exp\left(-\left[\epsilon_M^{III}\right]^2 k/\log(k)\right)\right)$ given Equation (6.1) holds for any $\epsilon_M^{(III)} > 0$.

- The number of unresolved components in this phase is $\mathcal{O}(k/\log(k))$. By Theorem 5, all the components are resolved with probability $1 - \mathcal{O}(\log(k)/k)$ for large enough c given Equation (6.1) holds.

By the union bound, the overall error probability is $o(1)$.

6.9 Conclusion

In this chapter, we present the first algorithm for compressive phase retrieval problem whose number of measurements is order-optimal and computational complexity is nearly order-optimal.

□ **End of chapter.**

Chapter 7

Conclusion

In this thesis, we present a framework to unify efficient algorithms for compressive sensing, network tomography, group testing and phase retrieval problems. The number of measurements and decoding complexity of all the algorithms introduced are (or nearly) information-theoretically order-optimal.

For future work, we aim to design robust and practical algorithms for these problems and to explore more applications of sparse recovery.

- Robust and practical algorithms:
  - Number of measurements: Choosing the node-degree distribution appropriately via density evolution proves to reduce the number of measurements required (even if by a constant factor).
  - Robust to noise: Instead of using repeated measurements in our SHO-FA algorithm, sophisticated and computationally efficient error-correcting codes can be implemented for leaf nodes to make the algorithm more robust to noise.
- More applications:

- Sparse recovery over finite fields: Many ideas from error-correcting codes over finite fields are borrowed to design novel sparse recovery algorithms over real numbers and complex numbers. In the reverse direction, the idea from our framework may also be applied to design different classes of efficient error-correcting codes over finite fields.
- Other sparse recovery problems: Sparse recovery problems come in many flavours and are not restricted to the four problems discussed in this thesis. Some examples of important sparse recovery problems that we have not considered are sparse fast fourier transform (sFFT), low-rank matrix completion, and sparse principal component analysis (sPCA). We may tailor the framework to these problems.
- Other applications: The sparse recovery algorithms can be applied to many real applications where sparsity can be exploited such as image processing (like face recognition), wireless sensor networks, storage for big data. It would be interesting to see how our framework may improve the efficiency in these areas.

□ **End of chapter.**

Appendix A

Proofs

A.1 SHO-FA

A.1.1 Proof of Lemma 1

Proof. It suffices to prove the desired property for all $\mathcal{S}(\mathbf{x})$ of size exactly k. Let $\mathcal{S}'(\mathbf{x}) \subseteq \mathcal{S}(\mathbf{x})$. Let $\{(s_1, t_1), (s_2, t_2), \ldots, (s_{d|\mathcal{S}'(\mathbf{x})|}, t_{d|\mathcal{S}'(\mathbf{x})|})\}$ be the set of outgoing edges from $\mathcal{S}'(\mathbf{x})$. Without loss of generality, we assume these edges are drawn in the following manner.

For each $i = 1, 2, \ldots, d|\mathcal{S}'(\mathbf{x})|$, the probability that the edge (s_i, t_i) reaches an "old" true node (on the right) that is already reached by those edges generated ahead of (s_i, t_i) is upper bounded as

$$\begin{aligned} \Pr_{\mathcal{G}}(t_i \in \{t_1, \ldots t_{i-1}\}) &\leq \frac{(i-1)}{ck} \\ &\leq \frac{d|\mathcal{S}'(\mathbf{x})|}{ck}. \end{aligned}$$

Let $N(\mathcal{S}'(\mathbf{x}))$ be the set of all neighboring nodes of the nodes in $\mathcal{S}'(\mathbf{x})$. The size of $N(\mathcal{S}'(\mathbf{x}))$ is no more than $2d|\mathcal{S}'(\mathbf{x})|/3$ if and only if out of $d|\mathcal{S}'(\mathbf{x})|$ edges, there exists a set of at least $d|\mathcal{S}'(\mathbf{x})|/3$ edges fail to reach "new" nodes (on the right). Exploiting this observation, we have

$$\Pr_{\mathcal{G}}\left(|N(\mathcal{S}'(\mathbf{x}))| \leq 2d|\mathcal{S}'(\mathbf{x})|/3\right)$$

$$
\begin{aligned}
&= \Pr_{\mathcal{G}}\left(\bigcup_{\substack{\sigma\subseteq\{1,\ldots,d|\mathcal{S}'(\mathbf{x})|\}\\ |\sigma|\geq|d\mathcal{S}'(\mathbf{x})/3|}}\bigcap_{i\in\sigma}\{t_i\in\{t_1,\ldots t_{i-1}\}\}\right)\\
&= \Pr_{\mathcal{G}}\left(\bigcup_{\substack{\sigma\subseteq\{1,\ldots,d|\mathcal{S}'(\mathbf{x})|\}\\ |\sigma|=d|\mathcal{S}'(\mathbf{x})|/3}}\bigcup_{\sigma'\supseteq\sigma}\bigcap_{i\in\sigma'}\{t_i\in\{t_1,\ldots t_{i-1}\}\}\right)\\
&\leq \binom{d|\mathcal{S}'(\mathbf{x})|}{d|\mathcal{S}'(\mathbf{x})|/3}\Big(\frac{d|\mathcal{S}'(\mathbf{x})|}{ck}\Big)^{d|\mathcal{S}'(\mathbf{x})|/3}.
\end{aligned}
$$

Consequently, the probability that there exists one $\mathcal{S}'(\mathbf{x})\subseteq\mathcal{S}(\mathbf{x})$ so that $|N(\mathcal{S}'(\mathbf{x}))|\leq 2d|\mathcal{S}'(\mathbf{x})|/3$ can be bounded by

$$
\begin{aligned}
&\Pr_{\mathcal{G}}(\cup_{\mathcal{S}'(\mathbf{x})\subseteq\mathcal{S}(\mathbf{x})}\{|N(\mathcal{S}'(\mathbf{x}))|\leq 2|\mathcal{S}'(\mathbf{x})|\})\\
&\leq \sum_{\mathcal{S}'(\mathbf{x})\subseteq\mathcal{S}(\mathbf{x})}\binom{d|\mathcal{S}'(\mathbf{x})|}{d|\mathcal{S}'(\mathbf{x})|/3}\Big(\frac{d|\mathcal{S}'(\mathbf{x})|}{ck}\Big)^{d|\mathcal{S}'(\mathbf{x})|/3}\\
&= \sum_{j=1}^{k}\binom{k}{j}\binom{dj}{dj/3}\Big(\frac{dj}{ck}\Big)^{dj/3}\\
&\leq \sum_{j=1}^{k}\Big(\frac{ke}{j}\Big)^{j}(3e)^{dj/3}\Big(\frac{dj}{ck}\Big)^{dj/3} \qquad \text{(A.1)}\\
&= \sum_{j=1}^{k}\left(\frac{ke}{j}\left(\frac{3dje}{ck}\right)^{d/3}\right)^{j}\\
&\leq \sum_{j=1}^{\lceil\sqrt{k}\rceil}\left(\frac{ke}{j}\left(\frac{3dje}{ck}\right)^{d/3}\right)^{j}+\sum_{j=\lfloor\sqrt{k}\rfloor}^{k}\left(\frac{ke}{j}\left(\frac{3dje}{ck}\right)^{d/3}\right)^{j}\\
&\leq \sqrt{k}\left(\frac{ke}{\sqrt{k}}\left(\frac{3d\sqrt{k}e}{ck}\right)^{d/3}\right)+\sum_{j=\lfloor\sqrt{k}\rfloor}^{\infty}\left(e\left(\frac{3de}{c}\right)^{d/3}\right)^{j} \qquad \text{(A.2)}\\
&\leq \left(\frac{3de}{c}\right)^{d/3}k^{1-d/6}e+\exp(-\Theta(\sqrt{k})) \qquad \text{(A.3)}\\
&= \mathcal{O}(k^{1-d/6}). \qquad \text{(A.4)}
\end{aligned}
$$

In the above, the inequality in (A.1) follows from Stirling's approximation; the upper bound in (A.2) is derived by noting that the first term in the sum takes its maximum when $j=\lfloor\sqrt{k}\rfloor$ and the second term is maximum

when $j = k$; (A.3) is obtained by noting that the second term is a geometric progression.

Finally, we plug in the choice of $d = 13$ to complete the proof. ■

A.1.2 Proof of Lemma 2

Suppose each set of of size k of $\mathcal{S}(\mathbf{x})$ nodes on the left of $\mathcal{G}$ has strictly more than $d/2$ times as many nodes neighbouring those in $\mathcal{S}(\mathbf{x})$, as there are in $\mathcal{S}(\mathbf{x})$. Then by standard arguments in the construction of expander codes [132], this implies the existence of a linear code of rate at least $1 - m/n$, and with relative minimum distance at least k/n.[1] But by the Hamming bound [124], it is known that codes of minimum distance δ can have rate at most $1-H(\delta)$, where $H(.)$ denotes the binary entropy function. Since $k = o(n)$, $\delta = k/n \to 0$. But in this regime $1-H(\delta) \to 1-\delta\log(1/\delta)$. Note that m/m' equals an integer. Comparing $(k/n)\log(n/k)$ with m/n gives the required result. ■

A.1.3 Proof of Lemma 3

For any set of nodes S in the graph $\mathcal{G}$, we define $N(S)$ as the set of neighboring nodes of the nodes in S. For any set $\mathcal{S}'(\mathbf{x}) \subseteq \mathcal{S}(\mathbf{x})$, we define β as

[1] For the sake of completeness we sketch such an argument here. Given such an expander graph $\mathcal{G}$, one can construct a $n \times k$ binary matrix A with 1s in precisely those (i, j) locations where the ith node on the left is connected with the jth node on the right. Treating this matrix A as the parity check matrix of a code over a block-length n implies that the rate of the code is at least k/n, since the parity-check matrix imposes at most k constraints on the n bits of the codewords. Also, the minimum distance is at least k. Suppose not, *i.e.* there exists a codeword in this linear code of weight less than k. Let the support of this codeword be denoted $\mathcal{S}(\mathbf{x})$. Then by the expansion property of $\mathcal{G}$, there are strictly more than $|\mathcal{S}(\mathbf{x})|d/2$ neighbours of $\mathcal{S}(\mathbf{x})$. But this implies that there is at least one node, say v, neighboring $\mathcal{S}(\mathbf{x})$ which has exactly one neighbor in $\mathcal{S}(\mathbf{x})$. But then the constraint corresponding to v cannot be satisfied, leading to a contradiction.

the portion of the nodes in $N(\mathcal{S}'(\mathbf{x}))$ that are $\mathcal{S}'(\mathbf{x})$-leaf nodes.

First, each node $v \in N(\mathcal{S}'(\mathbf{x}))$ is of one of the following two types:

1. It has only one neighboring node in $\mathcal{S}'(\mathbf{x})$, on the left of $\mathcal{G}$. By the definition of β, the number of nodes in $N(\mathcal{S}'(\mathbf{x}))$ of this type is $\beta|N(\mathcal{S}'(\mathbf{x}))|$.

2. It has at least two neighboring nodes in $\mathcal{S}'(\mathbf{x})$, on the left of $\mathcal{G}$. The number of nodes in $N(\mathcal{S}'(\mathbf{x}))$ of this type is $(1-\beta)|N(\mathcal{S}'(\mathbf{x}))|$.

We have two observations. First, since the degree of each node in $\mathcal{S}'(\mathbf{x})$ is d, the total number of edges from $\mathcal{S}'(\mathbf{x})$ to $N(\mathcal{S}'(\mathbf{x}))$ is at most $d|\mathcal{S}'(\mathbf{x})|$ and the number of nodes in $N(\mathcal{S}'(\mathbf{x}))$ is at most $d|\mathcal{S}'(\mathbf{x})|$.

Second, the total number of edges entering $N(\mathcal{S}'(\mathbf{x}))$ from $\mathcal{S}'(\mathbf{x})$ is at least

$$\beta|N(\mathcal{S}'(\mathbf{x}))| + 2(1-\beta)|N(\mathcal{S}'(\mathbf{x}))| = (2-\beta)|N(\mathcal{S}'(\mathbf{x}))|,$$

as the number of neighboring nodes for the nodes of Type 1 is one and of Type 2 is at least two.

Combining the above two observations, we can get the following inequality:

$$(2-\beta)|N(\mathcal{S}'(\mathbf{x}))| \leq d|\mathcal{S}'(\mathbf{x})|.$$

According to the setting of the Lemma, we also have $|N(\mathcal{S}'(\mathbf{x}))| \geq 2d/|\mathcal{S}'(\mathbf{x})|3$. Therefore, it follows that

$$2(2-\beta)d|\mathcal{S}'(\mathbf{x})|/3 \leq d|\mathcal{S}'(\mathbf{x})|,$$

and consequently $\beta \geq 1/2$. ■

A.1.4 Proof of Lemma 4

Consider the algorithm $\mathcal{A}$ that proceeds as follows. First, among the set of all right nodes that neighbour j, check if there exists a node i such that

$y_i^{(I)} = y_i^{(V)} = 0$. If there exists such a node, then output $\hat{x}_j = 0$. Otherwise, check if there exists a $\mathcal{S}(\mathbf{x})$-leaf node among the neighbours of j. This check can be performed by using verification and identification observations as described for the SHO-FA reconstruction algorithm. If there exists a leaf node, say i, then output $\hat{x}_j = |y_i|$. Else, the algorithm terminates without producing any output.

Two see that the above algorithm satisfies the claimed properties, consider the following two cases.

Case 1: $x_j = 0$. In this case, $\hat{x}_j = 0$ is output if at least one neighbour of j lies outside $N(\mathcal{S}(\mathbf{x}))$. Since $N(\mathcal{S}(\mathbf{x}))$ has at most dk elements, the probability that a neighbour of j lies inside $N(\mathcal{S}(\mathbf{x}))$ is at most $dk/ck = d/c$. Thus, the probability that none of the neighbours of j lie outside $N(\mathcal{S}(\mathbf{x}))$ is at least $(1 - (d/c)^d)$. The algorithm incorrectly reconstructs x_j if all neighbours of j lie within $N(\mathcal{S}(\mathbf{x}))$ and SHO-FA incorrectly identifies one of these nodes as a leaf node. By the analysis of SHO-FA, this event occurs with probability $o(1/k)$.

Case 2: $x_j \neq 0$. For $\mathcal{A}$ to produce the correct output, it has to identify one of the neighbours of j as a leaf. The probability that there exists a leaf among the neighbours of j is at least $(1 - (d/c)^d)$ by an argument similar to the previous case. Similarly, the proabability of erroneous identification is $o(1/k)$.

■

A.1.5 Phase noise

Proof of Lemma 5: First, we find an upper bound on the maximum possible phase displacement in y_i due to fixed noise vectors $\mathbf{z}$ and $\mathbf{e}$. Let $\Delta\theta_i$ be the difference in phase between the "noiseless" output $(A'\mathbf{x})_i$ and the actual

output $y_i = (A'(\mathbf{x}+\mathbf{z})+\mathbf{e})_i$. Figure 3.12a shows this geometrically. By a straightforward geometric argument, for fixed $\mathbf{z}$ and $\mathbf{e}$, the phase displacement $\Delta\theta_i$ is upper bounded by $\pi|(A'\mathbf{z})_i + e_i|/|(A'\mathbf{x})_i|$. Since i is a leaf node for $\mathcal{S}(\mathbf{x})$, $|(A'\mathbf{x})_i| \geq |\delta/k|$. Therefore,

$$\Delta\theta_i \leq \pi|(A'\mathbf{z})_i + e_i|k/\delta.$$

Since each z_j is a Gaussian with zero mean and variance σ_z^2, $(A'\mathbf{z})_i$ is a Complex Gaussian with zero mean and variance at most $n\sigma_z^2$. Further, each row of A' has at most dn/ck non-zero entries. Therefore, $(A'\mathbf{z})_i + e_i$ is a zero mean complex Gaussian with variance at most $(dn/ck)\sigma_z^2 + \sigma_e^2$.

The expected value of $\Delta\theta_i$ is bounded as follows:

$$\begin{aligned}
&E_{\mathbf{z},\mathbf{e}}(\Delta\theta_i) \\
&\leq E_{\mathbf{z},\mathbf{e}}(\pi|(A'\mathbf{z})_i + e_i|k/\delta) \\
&\leq \frac{\pi k}{\delta}\int_0^\infty \sqrt{\frac{2}{\pi(dn\sigma_z^2/ck)+\sigma_e^2}}\, l e^{-l^2/2(dn\sigma_z^2/ck+\sigma_e^2)} dl \\
&= \sqrt{\frac{2\pi k^2(dn\sigma_z^2/ck+\sigma_e^2)}{\delta^2}}.
\end{aligned}$$

Next, note that

$$\begin{aligned}
\Pr_{\mathbf{z},\mathbf{e}}\left(\Delta\theta_i > \alpha E_{\mathbf{z},\mathbf{e}}(\Delta\theta_i)\right) &\leq \Pr_{\mathbf{z},\mathbf{e}}\left(|(A'\mathbf{z})_i + e_i|k/\delta > \alpha E_{\mathbf{z},\mathbf{e}}(\Delta\theta_i)\right) \\
&= \Pr_{\mathbf{z},\mathbf{e}}\left(|(A'\mathbf{z})_i + e_i| > \alpha E_{\mathbf{z},\mathbf{e}}(\Delta\theta_i)\delta/\pi k\right) \\
&= Pr_{\mathbf{z},\mathbf{e}}\left(|(A'\mathbf{z})_i + e_i| > \alpha\sqrt{\frac{2(dn\sigma_z^2/ck+\sigma_e^2)}{\pi}}\right).
\end{aligned}$$

Finally, applying standard bounds on the tail probabilities of Gaussian random variables, the required probability is upper bounded by $e^{-(\alpha^2/2\pi)}/2$.

■

A.1.6 Probability of error

An error occurs only if one of the following take place:

1. The underlying graph $\mathcal{G}$ is not an $\mathcal{S}(\mathbf{x})$-expander. This probability can be made $o(1/k)$ by choosing $m = ck$, where the constant c is determined by Lemma 1

2. The phase noise in $\tilde{y}_{i(t)}(t)$ leads to an incorrect decoding of $\hat{\theta}_t^{(I,\gamma)}$ or $\hat{\theta}_t^{(V,\gamma)}$ for some γ and t.

   Note that the phase noise in $\tilde{y}_{i(t)}(t)$ consists:

   (a) The contribution due to noise vectors $\mathbf{z}$ and $\mathbf{e}$, and

   (b) The contribution due to the noise propagated while computing each $\tilde{y}_{i(t)}(\tau)$ from $\tilde{y}_{i(t)}(\tau - 1)$ for $\tau \leq t$.

   The contribution due to the first term is bounded by Lemma 5. Thus, for a target error probability ϵ', we choose $\alpha = \sqrt{2\pi \log 1/2\epsilon'}$, giving a contribution to the phase noise of at most

$$2\pi\sqrt{\frac{\log(1/2\epsilon')k^2(dn\sigma_z^2/ck + \sigma_e^2)}{\delta^2}}.$$

   To bound the contribution due to the second term, we note a few facts about the random graph $\mathcal{G}$. Let $\mathcal{G}_{\mathbf{x}}$ be the restriction of $\mathcal{G}$ to $\mathcal{S}(\mathbf{x})$ and its neighbours. Denote the smallest disjoint components of $\mathcal{G}_{\mathbf{x}}$ by $\mathcal{C}_{\mathbf{x}}(1), \mathcal{C}_{\mathbf{x}}(2), \ldots, \mathcal{C}_{\mathbf{x}}(M)$ and let the number of right nodes in component $\mathcal{C}_{\mathbf{x}}(p)$ be $D_{\mathbf{x}}(p)$. The following properties of the random sparse graph $\mathcal{G}_{\mathbf{x}}$ and its components follow from [85, 118].

   Lemma 13 ([85, 118]). *The random graph $\mathcal{G}_{\mathbf{x}}$ satisfies the following properties:*

   A. For a large enough choice of c, with probability $1 - o(1/k)$, $\mathcal{G}_{\mathbf{x}}$ consists almost entirely of hypertrees and unicyclic components.

   B. $\max_p D_{\mathbf{x}}(p) = \mathcal{O}(\log k)$ with probability $1 - o(1/k)$.

   C. $E_{\mathcal{G}}\left((D_{\mathbf{x}}(p))^2\right) = O(1)$.

Now, we observe that at each iteration t, any error in reconstruction of $\hat{x}_{j(t)}$ potentially adds to reconstruction error in all future iterations t' for which there is a path from $j(t)$ to $j(t')$. Thus, if $j(t)$ lies in the component $\mathcal{C}_{\mathbf{x}}(p_t)$, then from Property A above, the magnitude error in reconstruction of $\hat{x}_{j(t)}$ due to noisy reconstructions in previous iterations is upper bounded by

$$(D_{\mathbf{x}}(p_t))^2\sqrt{2\pi\log(1/2\epsilon')(n\sigma_z^2/k+\sigma_e^2)} \tag{A.5}$$

with probability at least $1-D_{\mathbf{x}}(p_t)\epsilon'$. Thus, the phase displacement in each $y_i^{(I,\gamma)}$ and $y_i^{(V,\gamma)}$ is at most

$$2\pi(D_{\mathbf{x}}(p_t))^2\sqrt{\frac{\log\left(1/2\epsilon'\right)k^2(n\sigma_z^2/k+\sigma_e^2)}{\delta^2}}.$$

Next, applying Property B, as long as

$$(\log k)^2\sqrt{\frac{2\pi\log\left(1/2\epsilon'\right)k^2(n\sigma_z^2/k+\sigma_e^2)}{\delta^2}}=o\left(n^{-1/\Gamma}\right), \tag{A.6}$$

the probability of any single phase being incorrectly detected is upper bounded by ϵ'. Since we there are a total of $8\Gamma k$ possible phase measurements, we choose $\epsilon'=1/\Gamma k^2$ to achieve an overall target error probability $1/k$.

3. The verification step passes for each measurement in the t-th measurement, even though $i(t)$ is not a leaf node for $\mathcal{S}_\delta^c(\mathbf{x})$.

4. $\mathcal{D}(T)\neq A'$, i.e., the algorithm terminates without recovering all x_j's. Note that similar to the exact k-sparse case, in each iteration t, by Lemma 3, the probability that $i(t)$ is a leaf node for $\mathcal{S}_\delta(\mathbf{x}-\hat{\mathbf{x}}(t))$ at least $1/2$. However, due to noise, there is a non-zero probability that even when $i(t)$ is a leaf node, it does not pass the verification tests. We know from the analysis for the previous case that this probability is $\mathcal{O}(1/k)$ for each $i(t)$. Therefore, the probability that a randomly

picked $i(t)$ passes the verification test is $1/2 - \mathcal{O}(1/k)$. Thus, in expectation, the number of iterations required by the algorithm is $2k/(1 - \mathcal{O}(1/k))$. By concentration arguments, it follows that the probability that the algorithm does not terminate in $4k$ iterations is $o(1/k)$ as k grows without bound.

A.1.7 Estimation error

Next, we bound the error in estimating $\hat{\mathbf{x}}$. We first find an upper bound on $||\hat{\mathbf{x}} - \mathbf{x}_{\mathcal{S}_\delta^c}||_1$ that holds with a high probability. Applying the bound in (A.5), for each $t = 1, 2, \ldots, T$,

$$|x_{j(t)} - \hat{x}_{j(t)}| = \mathcal{O}\left((D_{\mathbf{x}}(p_t))^2 \sqrt{2\pi \log(1/2\epsilon')(n\sigma_z^2/k + \sigma_e^2)}\right)$$

with probability $1 - \mathcal{O}(1/k)$. Therefore, with probability $1 - \mathcal{O}(1/k)$,

$$\begin{aligned} ||\hat{\mathbf{x}} - \mathbf{x}_{\mathcal{S}_\delta^c}||_1 &= \sum_{\substack{1 \le t \le T \\ t:j(t) \notin \mathcal{S}_\delta}} |\hat{x}_j - x_j| + \sum_{\substack{1 \le t \le T \\ t:j(t) \in \mathcal{S}_\delta}} |\hat{x}_j| \\ &\le \sum_{j \notin \mathcal{S}_\delta} |\hat{x}_j - x_j| + \sum_{j \in \mathcal{S}_\delta} |\hat{x}_j - x_j| + \sum_{j \in \mathcal{S}_\delta} |x_j| \\ &= \mathcal{O}\left(\sum_{p=1}^{P} \sum_{j \in \mathcal{C}(p)} (D_{\mathbf{x}}(p_t))^2 \sqrt{2\pi \log(1/2\epsilon')(n\sigma_z^2/k + \sigma_e^2)}\right) \quad \text{(A.7)} \end{aligned}$$

Next, note that $||\mathbf{z}||_1 = \sum_{j=1}^{n} |z_j|$ and $||\mathbf{e}||_1 = \sum_{i=1}^{m} |e_i|$. Since each z_j is a Gaussian random variable with variance σ_z^2, The expected value of $|z_j|$ is $\sigma_z\sqrt{2/\pi}$. Therefore, for every $\epsilon' > 0$, for n large enough,

$$\Pr(||\mathbf{z}||_1 < (1/2)n\sigma_z\sqrt{2/\pi}) < \epsilon'. \quad \text{(A.8)}$$

Similarly, for m large enough,

$$\Pr(||\mathbf{e}||_1 < (1/2)ck\sigma_e\sqrt{2/\pi}) < \epsilon'. \quad \text{(A.9)}$$

Combining inequalities (A.7)- (A.9) and Property C of Lemma 13, we have, with a high probability,

$$E\left(||\hat{\mathbf{x}} - \mathbf{x}_{\mathcal{S}_\delta^c}||_1\right) = \mathcal{O}\left(k\sqrt{\log(1/\epsilon')}\left(\frac{||\mathbf{z}||_1}{\sqrt{nk}} + \frac{||\mathbf{e}||_1}{k}\right)\right) + \delta$$

$$= \mathcal{O}\left(\sqrt{\frac{k}{n}}\sqrt{\log(1/\epsilon')}||\mathbf{z}||_1 + \sqrt{\log(1/\epsilon')}||\mathbf{e}||_1\right) \quad \text{(A.10)}$$

Next, applying the bound in (3.4), we obtain

$$\begin{aligned} E\left(||\hat{\mathbf{x}} - \mathbf{x}||_1\right) &= \mathcal{O}\left(\sqrt{\frac{k}{n}}\sqrt{\log(1/\epsilon')}||\mathbf{z}||_1 + \sqrt{\log(1/\epsilon')}||\mathbf{e}||_1\right) + 2\delta \\ &= \mathcal{O}(\sqrt{\frac{k\log k}{n}}||\mathbf{z}||_1 + \sqrt{\log k}||\mathbf{e}||_1) \quad \text{(A.11)} \end{aligned}$$

with a high probability.

A.1.8 Proof of Theorem 3

Finally, to complete the proof of Theorem 3, we let $\delta = \min\{\mathcal{O}(n\sigma_z), o(1)\}$. By (A.8) with a high probability, $\delta = \mathcal{O}(||\mathbf{z}||)$. Finally, recall the assumption that $k = \mathcal{O}(n^{1-\Delta})$. Applying these to the bound obtained in (A.11), we get

$$||\hat{\mathbf{x}} - \mathbf{x}||_1 \leq C\left(||\mathbf{z}||_1 + \sqrt{\log k}||\mathbf{e}||_1\right)$$

for an appropriate constant $C = C(\sigma_z, \sigma + e)$.

A.2 SUPER

A.2.1 Proof of Claim 1

Proof. We will use four measurements in the "cancelling out" process,

$$\begin{aligned} y_i^{(\mathcal{G},1)} &= \left|A + \mathbf{x}_j \cos\left(\frac{j\pi}{2n}\right)\right| \\ y_i^{(\mathcal{G},2)} &= \left|B + \mathbf{x}_j \iota \sin\left(\frac{j\pi}{2n}\right)\right| \\ y_i^{(\mathcal{G},3)} &= \left|C + \mathbf{x}_j \exp\left(\iota\frac{j\pi}{2n}\right)\right| \\ y_i^{(\mathcal{G},5)} &= \left|D + \mathbf{x}_j \exp\left(\iota\phi_{i,j}\right)\right|, \end{aligned}$$

where A, B, C, and D are calculated from the decoded non-zero components which connect to right node i in $\mathcal{G}$.

We find that by the measurements design

$$A + B = C$$

and

$$\mathbf{x}_j \cos\left(\frac{j\pi}{2n}\right) + \mathbf{x}_j \iota \sin\left(\frac{j\pi}{2n}\right) = \mathbf{x}_j \exp\left(\iota \frac{j\pi}{2n}\right).$$

Let

$$\begin{aligned} \frac{j\pi}{2n} &= \alpha \\ A + \mathbf{x}_j \cos\left(\frac{j\pi}{2n}\right) &= U \\ B + \mathbf{x}_j \iota \sin\left(\frac{j\pi}{2n}\right) &= V \\ C + \mathbf{x}_j \exp\left(\iota \frac{j\pi}{2n}\right) &= W, \end{aligned}$$

we have

$$\begin{aligned} y_i^{(\mathcal{G},1)} &= |U| \\ y_i^{(\mathcal{G},2)} &= |V| \\ y_i^{(\mathcal{G},3)} &= |W| \\ &= |U + V|. \end{aligned}$$

Finding the relation between U and V:

We know that

$$U = V \times \frac{y_i^{(\mathcal{G},1)}}{y_i^{(\mathcal{G},2)}} \exp(\iota\psi),$$

or

$$U = V \times \frac{y_i^{(\mathcal{G},1)}}{y_i^{(\mathcal{G},2)}} \exp(-\iota\psi),$$

where ψ is the phase between U and V and $\cos\psi = \frac{|U|^2+|V|^2-|U+V|^2}{2|U||V|}$.

Finding the relation between $\mathbf{x}$ and α:

For simplicity, we only consider the case that

$$\begin{aligned} U &= V \times \frac{y_i^{(\mathcal{G},1)}}{y_i^{(\mathcal{G},2)}} \exp(\iota\psi) \\ &\triangleq V \times M. \end{aligned}$$

So,

$$A + \mathbf{x}_j \cos\alpha = [B + \mathbf{x}_j \iota \sin\alpha] M.$$

We have

$$\mathbf{x}_j = \frac{BM - A}{\cos\alpha - \iota M \sin\alpha}.$$

Solving $\cos^2\alpha$ by quadratic equation:

Replacing $\mathbf{x}_j$in

$$y_i^{(\mathcal{G},1)} = |U|,$$

we know that

$$\begin{aligned} y_i^{(\mathcal{G},1)} &= \left| A + \frac{BM - A}{\cos\alpha - \iota M \sin\alpha} \cos\alpha \right| \\ &= \left| \frac{BM\cos\alpha - \iota AM\sin\alpha}{\cos\alpha - \iota M\sin\alpha} \right| \\ &= \left| \frac{B\cos\alpha - \iota A\sin\alpha}{\cos\alpha - \iota M\sin\alpha} \right| |M| \\ &= \left| \frac{B\cos\alpha - \iota A\sin\alpha}{\cos\alpha - \iota M\sin\alpha} \right| \frac{y_i^{(\mathcal{G},1)}}{y_i^{(\mathcal{G},2)}}. \end{aligned}$$

So,

$$y_i^{(\mathcal{G},2)} \left|\cos\alpha - \iota M \sin\alpha\right| = \left|B\cos\alpha - \iota A\sin\alpha\right|.$$

Let

$$\begin{aligned} A &= A_1 + \iota A_2 \\ B &= B_1 + \iota B_2 \\ M &= M_1 + \iota M_2, \end{aligned}$$

where A_1, A_2, B_1, B_2, M_1, and M_2 are real numbers. We have

$$\begin{aligned} & y_i^{(\mathcal{G},2)} \left|\cos\alpha - \iota\left(M_1 + \iota M_2\right)\sin\alpha\right| \\ &= \left|\left(B_1 + \iota B_2\right)\cos\alpha - \iota\left(A_1 + \iota A_2\right)\sin\alpha\right|. \end{aligned}$$

Squaring both sides, we get

$$\begin{aligned} & \left[y_i^{(\mathcal{G},2)}\right]^2 \left[\left(\cos\alpha + M_2\sin\alpha\right)^2 + \left(M_1\sin\alpha\right)^2\right] \\ &= \left(B_1\cos\alpha + A_2\sin\alpha\right)^2 + \left(B_2\cos\alpha - A_1\sin\alpha\right)^2. \end{aligned}$$

After reorganizing the above equation, we have

$$\begin{aligned} & \left(\left[y_i^{(\mathcal{G},2)}\right]^2 - |B|^2\right)\cos^2\alpha + \left(\left[y_i^{(\mathcal{G},1)}\right]^2 - |A|^2\right)\sin^2\alpha \\ &= 2\cos\alpha\sin\alpha\left(A_2B_1 - A_1B_2 - 2\left[y_i^{(\mathcal{G},2)}\right]^2 M_2\right). \end{aligned}$$

Let

$$\begin{aligned} P &= \left[y_i^{(\mathcal{G},2)}\right]^2 - |B|^2 \\ Q &= \left[y_i^{(\mathcal{G},2)}\right]^2 - |A|^2 \\ R &= A_2B_1 - A_1B_2 - 2\left[y_i^{(\mathcal{G},2)}\right]^2 M_2 \\ S &= \cos^2\alpha \end{aligned}$$

and square both sides, we have

$$[PS + Q(1-S)]^2 \quad = \quad 4R^2 S(1-S).$$

After reorganizing the above equation, we get

$$\begin{aligned} &\left(P^2 + Q^2 - 2PQ + 4R^2\right) S^2 \\ &+ \left(2PQ - 2Q^2 - 4R^2\right) S + Q^2 \quad = \quad 0. \end{aligned}$$

We are able to solve S (quadratic equation) in constant time and similarly for the case that $U = V \times \frac{y_i^{(\mathcal{G},1)}}{y_i^{(\mathcal{G},2)}} \exp(-\iota\psi)$.

Resolving the degeneracy via random unit complex measurements:

After deriving the value of $S = \cos^2 \alpha$, we can get the constant (4) possible value of j and $\mathbf{x}_j$ (both magnitude and the relative phase in $\mathcal{H}'$) pairs.

Last, we check which pairs of solution that satisfies the following equation to resolve the degeneracy

$$y_i^{(\mathcal{G},5)} \quad = \quad \left| D + \mathbf{x}_j \exp(\iota\phi_{i,j}) \right|.$$

■

A.2.2 Proof of Lemma 12

Proof. Let Y_i be the indicator random variable which represents whether i-th coupon is picked in M trials. We know that $Y_i's$ are dependent and

$$Y_i = \begin{cases} 1 & \text{with probability } 1 - \left(1 - \frac{1}{V}\right)^M \\ 0 & \text{with probability } \left(1 - \frac{1}{V}\right)^M. \end{cases}$$

Then, $Y = Y_1 + \cdots + Y_V$ is the total number of different types of coupons picked in M trials.

By the linearity of expectation, we have

$$\begin{aligned} E[Y] &= \Sigma_{i=1}^{V} E[Y_i] \\ &= V\left[1-\left(1-\frac{1}{V}\right)^{M}\right] \\ &\doteq V\left(1-e^{-\frac{M}{V}}\right) \\ &= V\left(1-\frac{V-U}{V}\right) \\ &= U. \end{aligned}$$

Let $Z_1, \ldots, Z_M$ be independent random variables all taking values in $[V]$ uniformly at random representing each pick for V types of coupon.

Let $f(Z_1, \ldots, Z_M)$ be the number of different types of coupons picked. Then, $\mathbf{E}[f] = \mathbf{E}[Y] \doteq U$.

Also, $\forall i \in [M]$,

$$|f(Z_1, \ldots, Z_i, \ldots, Z_M) - f(Z_1, \ldots, Z_i', \ldots, Z_M)| \leq 1.$$

.

For all $\beta > 0$, by McDiarmid's Inequality (Theorem 2), we have

$$\Pr(f - \mathbf{E}[f] \leq -\beta) \leq \exp\left(-\frac{2\beta^2}{M}\right)$$

and

$$\Pr(f - \mathbf{E}[f] \geq \beta) \leq \exp\left(-\frac{2\beta^2}{M}\right).$$

Thus,

$$\Pr(f \leq V(1-e^{-M/V}) - \beta) \leq \exp\left(-\frac{2\beta^2}{M}\right).$$

Let $M = V \log \frac{V}{V-U}$ and $\beta = \epsilon U$, we know that

$$\begin{aligned}\Pr(f \leq (1-\epsilon)U) &\leq \exp\left(-\frac{2(\epsilon U)^2}{V\log\frac{V}{V-U}}\right)\\ &\leq \exp -\frac{2(\epsilon U)^2}{V\frac{U}{V-U}}\\ &= \exp\left(-\frac{2\epsilon^2 U(V-U)}{V}\right)\end{aligned}$$

and

$$\Pr(f \geq (1+\epsilon)U) \leq \exp\left(-\frac{2\epsilon^2 U(V-U)}{V}\right).$$

Therefore,

$$\Pr((1-\epsilon)U \leq f \leq (1+\epsilon)U) \geq 1 - 2\exp\left(-\frac{2\epsilon^2 U(V-U)}{V}\right).$$

■

A.2.3 Proof of Theorem 5

Proof. Let Z_i be the event that i-th coupon has not yet picked in M trials. We know that

$$\begin{aligned}\Pr(Z_i) &= \left(1-\frac{1}{V}\right)^M\\ &\leq \exp(-M/V).\end{aligned}$$

Then,

$$\begin{aligned}\Pr(X > M) &= \Pr(\cup_{i=1}^{v} Z_i)\\ &\leq \sum_{i=1}^{v}\Pr(Z_i)\\ &\leq V\exp(-M/V).\end{aligned}$$

Let $M = \eta V \log V$, we get

$$\Pr\left(X > \eta V \log V\right) \quad \leq \quad V^{-\eta+1}.$$

■

□ **End of chapter.**

Bibliography

[1] S. Aeron, V. Saligrama, and M. Zhao. Information theoretic bounds for compressed sensing. *IEEE Transactions on Information Theory*, 56(10):5111 –5130, Oct. 2010.

[2] S. Ahuja, S. Ramasubramanian, and M. Krunz. Srlg failure localization in optical networks. *IEEE/ACM Transactions on Networking*, 19(4):989 –999, Aug. 2011.

[3] M. Akçakaya and V. Tarokh. A frame construction and a universal distortion bound for sparse representations. *IEEE Transactions on Signal Processing*, 56(6):2443–2450, June 2008.

[4] M. Akçakaya and V. Tarokh. Shannon-theoretic limits on noisy compressive sampling. *IEEE Transactions on Information Theory*, 56(1):492–504, Jan. 2010.

[5] M. Akçakaya and V. Tarokh. New conditions for sparse phase retrieval. *e-prints, arXiv:1310.1351[cs.IT]*, 2013.

[6] M. Alekhnovich. Linear diophantine equations over polynomials and soft decoding of reed-solomon codes. *IEEE Transactions on Information Theory*, 51(7):2257–2265, July 2005.

[7] B. Alexeev, A. S. Bandeira, M. Fickus, and D. G. Mixon. Phase retrieval with polarization. *e-prints, arXiv:1210.7752[cs.IT]*, 2012.

[8] G. Atia and V. Saligrama. Boolean compressed sensing and noisy group testing. *IEEE Transactions on Information Theory*, 58:1880 – 1901, March 2012.

[9] K. D. Ba, P. Indyk, E. Price, and D. P. Woodruff. Lower bounds for sparse recovery. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1190–1197, 2010.

[10] M. Bakshi, S. Jaggi, S. Cai, and M. Chen. SHO-FA: Robust compressive sensing with order-optimal complexity, measurements, and bits. In *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 786–793, Oct 2012.

[11] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, December 2008.

[12] A. Barg and G. Zemor. Error exponents of expander codes. *IEEE Transactions on Information Theory*, 48(6):1725 –1729, June 2002.

[13] A. Barg and G. Zémor. Error exponents of expander codes under linear-complexity decoding. *SIAM Journal on Discrete Mathematics*, 17(3):426–445, 2004.

[14] R. Bayer. Symmetric binary b-trees: Data structure and maintenance algorithms. *Acta Informatica*, 1(4):290–306, 1972.

[15] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, and M. Strauss. Combining geometry and combinatorics: A unified approach to sparse signal recovery. In *Proceedings of the 46th Annual Allerton Confer-*

ence on Communication, Control, and Computing (Allerton), pages 798 –805, Sept. 2008.

[16] R. Berinde and P. Indyk. Sequential sparse matching pursuit. In *Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 36–43, Sept 2009.

[17] R. Berinde, P. Indyk, and M. Ruzic. Practical near-optimal sparse recovery in the l1 norm. In *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 198–205, Sept 2008.

[18] P. Berman and V. Ramaiyer. Improved Approximations for the Steiner Tree Problem. *Journal of Algorithms*, 17:381–408, 1994.

[19] S. N. Bernstein. On certain modifications of chebyshev's inequality. *Doklady Akademii Nauk SSSR*, 17(6):275–277, 1937.

[20] B. Bollobas. Random Graphs. *Cambridge University Press*, 2001.

[21] T. Bu, N. Duffield, F. Presti, and D. Towsley. Network tomography on general topologies. *SIGMETRICS Performance Evaluation Review*, 30(1):21–30, Jun 2002.

[22] S. Cai, M. Bakshi, S. Jaggi, and M. Chen. FRANTIC: A fast reference-based algorithm for network tomography via compressive sensing. *e-prints, arXiv:1312.0825 [cs.NI]*, 2013.

[23] S. Cai, M. Bakshi, S. Jaggi, and M. Chen. FRANTIC: A fast reference-based algorithm for network tomography via compressive sensing. In *Proceedings of the Sixth International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–7, Jan 2014.

[24] S. Cai, M. Bakshi, S. Jaggi, and M. Chen. SUPER: Sparse signals with unknown phases efficiently recovered. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 2007–2011, June 2014.

[25] S. Cai, M. Bakshi, S. Jaggi, and M. Chen. SUPER: Sparse signals with unknown phases efficiently recovered. *e-prints, arXiv:1401.4269[cs.IT]*, 2014.

[26] S. Cai, M. Jahangoshahi, M. Bakshi, and S. Jaggi. GROTESQUE: Noisy group testing (quick and efficient). In *Proceedings of the 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1234–1241, Oct 2013.

[27] S. Cai, M. Jahangoshahi, M. Bakshi, and S. Jaggi. GROTESQUE: Noisy group testing (quick and efficient). *e-prints, arXiv:1307.2811[cs.IT]*, 2013.

[28] R. Calderbank, S. Howard, and S. Jafarpour. Sparse reconstruction via the reed-muller sieve. *e-prints, arXiv:1004.2926[cs.IT]*, 2010.

[29] E. Candès and X. Li. Solving quadratic equations via phaselift when there are about as many equations as unknowns. *Foundations of Computational Mathematics*, 14(5):1–10, Oct 2014.

[30] E. Candès and B. Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, June 2012.

[31] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489 – 509, Feb. 2006.

[32] E. Candès, T. Strohmer, and V. Voroninski. Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming. *Communications on Pure and Applied Mathematics*, 66(8):1241–1274, 2013.

[33] E. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406 –5425, Dec. 2006.

[34] E. J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592, 2008.

[35] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network tomography: recent developments. *Statistical Science*, 19:499–517, 2004.

[36] C. L. Chan, P. H. Che, S. Jaggi, and V. Saligrama. Non-adaptive probabilistic group testing with noisy measurements: Near-optimal bounds with efficient algorithms. In *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1832–1839, Sept 2011.

[37] Y. Chen, D. Bindel, H. Song, and R. Katz. Algebra-based scalable overlay network monitoring: algorithms, evaluation, and applications. *IEEE/ACM Transactions on Networking*, 15(5):1084–1097, 2007.

[38] Y. Cheng and D. Du. New constructions of one- and two-stage pooling designs. *Journal of Computational Biology*, 15(2):195–205, 2008.

[39] M. Cheraghchi, A. Karbasi, S. Mohajer, and V. Saligrama. Graph-constrained group testing. *IEEE Transactions on Information Theory*, 58(1):248 –262, Jan. 2012.

[40] H. Y. Cheung, T. C. Kwok, and L. C. Lau. Fast matrix rank algorithms and applications. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing (STOC)*, pages 549–562, New York, NY, USA, 2012. ACM.

[41] A. Cohen, R. DeVore, and W. Dahmen. Compressed sensing and best k-term approximation. *Journal of the AMS*, 22:211–231, 2009.

[42] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[43] G. Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM Transactions Database Systems (TODS)*, 30(1):249–278, March 2005.

[44] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In *Proceedings of the 40th Annual Conference on Information Sciences and Systems*, pages 198 –201, March 2006.

[45] W. Dai, O. Milenkovic, and H. V. Pham. Structured sublinear compressive sensing via belief propagation. *e-prints, arXiv:1101.3348[cs.IT]*, 2011.

[46] J. Dainty and J. Fienup. Phase Retrieval and Image Reconstruction for Astronomy. *Chapter 7 In H. Stark, ed., Image Recovery: Theory and Application, Academic Press*, pages 231–275, 1987.

[47] P. Damaschke and A. S. Muhammad. Competitive group testing and learning hidden vertex covers with minimum adaptivity. In *Proceed-*

ings of the 17th international conference on Fundamentals of computation theory (FCT), pages 84–95, 2009.

[48] P. Damaschke and A. S. Muhammad. Randomized group testing both query-optimal and minimal adaptive. In *Proceedings of the 38th international conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 214–225, 2012.

[49] A. De Bonis, L. Gasieniec, and U. Vaccaro. Optimal two-stage algorithms for group testing problems. *SIAM Journal on Computing*, 34(5):1253–1270, 2005.

[50] D. Declercq and M. Fossorier. Decoding Algorithms for Nonbinary LDPC Codes Over GF (q). *IEEE Transactions on Communications*, 55(4):633–643, April 2007.

[51] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, April 2006.

[52] D. L. Donoho, A. Javanmard, and A. Montanari. Information-theoretically optimal compressed sensing via spatial coupling and approximate message passing. *e-prints, arXiv:1112.0708 [cs.IT]*, 2011.

[53] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 58(2):1094–1121, Feb 2012.

[54] R. Dorfman. The detection of defective members of large populations. *Annals of Mathematical Statistics*, 14:436–411, 1943.

[55] D. Z. Du and F. K. Hwang. Combinatorial group testing and its applications. *World Scientific Series on Applied Mathematics*, 12, 1999.

[56] D. Z. Du and F. K. Hwang. *Combinatorial Group Testing and Its Applications*. World Scientific Publishing Company, 2nd edition, 2000.

[57] A. G. Dyachkov and V. V. Rykov. Bounds on the length of disjunctive codes. *Probl. Peredachi Inf.*, 18:7–13, 1982.

[58] A. G. Dyachkov, V. V. Rykov, and A. M. Rashad. Superimposed distance codes. *Problems Control Inform. Theory*, 18:237 – 250, 1989.

[59] W. Feller. *An Introduction to Probability Theory and Its Applications. Vol. I, 2nd ed.* Wiley, New York, 1958.

[60] J. Fienup. Phase retrieval algorithms: a comparison. *Appl. Opt.*, 21:2758–2769, 1982.

[61] J. Fienup. Phase retrieval algorithms: a personal tour [invited]. *Appl. Opt.*, 52:45–56, 2013.

[62] M. Firooz and S. Roy. Network tomography via compressed sensing. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, Dec 2010.

[63] A. K. Fletcher, S. Rangan, and V. K. Goyal. Necessary and sufficient conditions for sparsity pattern recovery. *IEEE Transactions on Information Theory*, 55(12):5758 – 5772, Dec. 2009.

[64] A. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of IEEE*, 98(6):937–947, 2010.

[65] A. C. Gilbert, Y. Li, E. Porat, and M. J. Strauss. Approximate sparse recovery: optimizing time and measurements. In *Proceedings of the 42nd ACM symposium on Theory of computing (STOC)*, pages 475–484, 2010.

[66] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. Algorithmic linear dimension reduction in the l1 norm for sparse vectors. In *Proceedings of the 44th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2006.

[67] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. One sketch for all: Fast algorithms for compressed sensing. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing (STOC)*, pages 237–246, 2007.

[68] M. T. Goodrich, M. J. Atallah, and R. Tamassia. Indexing information for data forensics. In *Applied Cryptography and Network Security Lecture Notes in Computer Science*, volume 3531, pages 206–221.

[69] M. T. Goodrich and D. S. Hirschberg. Improved adaptive group testing algorithms with applications to multiple access channels and dead sensor diagnosis. *Journal of Combinatorial Optimization*, 15:95 – 121, Jan 2008.

[70] M. T. Goodrich and M. Mitzenmacher. Invertible bloom lookup tables. *e-prints, arXiv:1101.2245 [cs.DS]*, 2011.

[71] D. Guo, J. Luo, L. Zhang, and K. Shen. Compressed neighbor discovery for wireless networks. *e-prints, arXiv:1012.1007 [cs.NI]*, 2010.

[72] V. Guruswami and P. Indyk. Linear-time list decoding in error-free settings: (extended abstract). *Automata, Languages and Programming Lecture Notes in Computer Science*, 3142:695–707, 2004.

[73] N. Harvey, M. Patrascu, Y. Wen, S. Yekhanin, and V. Chan. Non-adaptive fault diagnosis for all-optical networks via combinatorial group testing on graphs. In *Proceedings of IEEE International Con-*

ference on Computer Communications (INFOCOM), pages 697 –705, May 2007.

[74] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin (New series) of the American Mathematical Society*, 43:439–561, 2006.

[75] S. Hougardy and H. J. Prömel. A 1.598 approximation algorithm for the Steiner problem in graphs. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 448–453, Jan 1999.

[76] F. K. Hwang. Group testing with a dilution effect. *Biometrika*, 63(3):pp. 671–673, 1976.

[77] P. Indyk, H. Ngo, and A. Rudra. Efficiently decodable non-adaptive group testing. *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1126–1142, Jan 2010.

[78] P. Indyk and M. Ruzic. Near-optimal sparse recovery in the l1 norm. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 199–207, Oct 2008.

[79] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *e-prints, arXiv:1104.3160 [cs.IT]*, 2011.

[80] S. Jafarpour, W. Xu, B. Hassibi, and R. Calderbank. Efficient and robust compressed sensing using optimized expander graphs. *IEEE Transactions on Information Theory*, 55(9):4299–4308, 2009.

[81] K. Jaganathan, S. Oymak, and B. Hassibi. Phase retrieval for sparse signals using rank minimization. In *Proceedings of IEEE In-*

ternational Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3449–3452, March 2012.

[82] K. Jaganathan, S. Oymak, and B. Hassibi. Recovery of sparse 1-d signals from the magnitudes of their fourier transform. In *Proceedings of IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 1473–1477, July 2012.

[83] K. Jaganathan, S. Oymak, and B. Hassibi. Sparse phase retrieval: Convex algorithms and limitations. In *Proceedings of IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 1022–1026, July 2013.

[84] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, 353:153–181, 1996.

[85] M. Karoński and T. Łuczak. The phase transition in a random hypergraph. *J. Comput. Appl. Math.*, 142(1):125–135, May 2002.

[86] M. Karpinski and A. Zelikovsky. New Approximation Algorithms for the Steiner Tree Problem. *Journal of Combinatorial Optimizaion*, 1:47–65, 1997.

[87] J. Kleinberg. Detecting a network failure. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 231–239, Nov 2000.

[88] F. Krzakala, M. Mézard, F. Sausset, Y. Sun, and L. Zdeborová. Statistical-physics-based reconstruction in compressed sensing. *Physical Review X*, 2(2):021005, 2012.

[89] S. Kudekar and H. Pfister. The effect of spatial coupling on compressive sensing. In *Proceedings of the 48th Annual Allerton Conference*

on Communication, Control, and Computing (Allerton), pages 347–353, Sept 2010.

[90] X. Li and V. Voroninski. Sparse signal recovery from quadratic measurements via convex programming. *e-prints, arXiv:1209.4785[cs.IT]*, 2012.

[91] Y. Lu, A. Montanari, and B. Prabhakar. Counter braids: Asymptotic optimality of the message passing decoding algorithm. In *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 209 –216, Sept. 2008.

[92] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani. Counter braids: a novel counter architecture for per-flow measurement. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 121–132, June 2008.

[93] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2006.

[94] D. J. C. Mackay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, June 2007.

[95] M. B. Malyutov. Separating property of random matrices. *Mat. Zametki*, 23:155–167, 1978.

[96] M. B. Malyutov and H. Sadaka. Jaynes principle in testing active variables of linear model. *Random Operators and Stochastic Equations*, 6:311–330, 1998.

[97] A. Mazumdar. On almost disjunct matrices for group testing. *Algorithms and Computation Lecture Notes in Computer Science*, 7676:649–658, 2012.

[98] C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, 141(1):148–188, 1989.

[99] M. Mèdzard and C. Toninelli. Group testing with random pools: Optimal two-stage algorithms. *IEEE Transactions on Information Theory*, 57(3):1736 –1745, March 2011.

[100] R. Millane. Phase retrieval in crystallography and optics. *J. Opt. Soc. Am. A*, 7:394–411, 1990.

[101] M. Mitzenmacher and G. Varghese. Biff (bloom filter) codes: Fast error correction for large data sets. *2012 IEEE International Symposium onInformation Theory Proceedings (ISIT)*, pages 483–487, July 2012.

[102] M. Molloy. Cores in random hypergraphs and boolean formulas. *Random Struct. Algorithms*, 27(1):124–135, Aug. 2005.

[103] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[104] P. Netrapalli, P. Jain, and S. Sanghavi. Phase retrieval using alternating minimization. *e-prints, arXiv:1306.0160v1 [stat.ML]*, 2013.

[105] H. Ngo, E. Porat, and A. Rudra. Efficiently decodable error-correcting list disjunct matrices and applications. *Automata, Languages and Programming Lecture Notes in Computer Science*, 6755:557–568, 2011.

[106] H. Q. Ngo and D.-Z. Du. A Survey on Combinatorial Group Testing Algorithms with Applications to DNA Library Screening. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 55:171 – 182, 2000.

[107] H. Nguyen and P. Thiran. Using end-to-end data to infer lossy links in sensor networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–12, April 2006.

[108] H. Ohlsson and Y. C. Eldar. On conditions for uniqueness in sparse phase retrieval. *e-prints, arXiv:1308.5447[cs.IT]*, 2013.

[109] H. Ohlsson, A. Yang, R. Dong, and S. Sastry. Compressive phase retrieval from squared output measurements via semidefinite programming. *e-prints, arXiv:1111.6323v3 [math.ST]*, 2011.

[110] S. Oymak, A. Jalali, M. Fazel, Y. C. Eldar, and B. Hassibi. Simultaneously structured models with application to sparse and low-rank matrices. *e-prints, arXiv:1212.3753[cs.IT]*, 2013.

[111] F. Parvaresh and B. Hassibi. Explicit measurements with almost optimal thresholds for compressed sensing. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3853–3856, March 2008.

[112] S. Pawar and K. Ramchandran. A hybrid DFT-LDPC framework for fast, efficient and robust compressive sensing. In *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1943–1950, Oct 2012.

[113] S. Pawar and K. Ramchandran. Computing a k-sparse n-length discrete fourier transform using at most 4k samples and o(k log k) complexity. *e-prints, arXiv:1305.0870[cs.IT]*, 2013.

[114] R. Pedarsani, K. Lee, and K. Ramchandran. PhaseCode: Fast and efficient compressive phase retrieval based on sparse-graph-codes. *e-prints, arXiv:1408.0034[cs.IT]*, 2014.

[115] Y. Plan and R. Vershynin. One-bit compressed sensing by linear programming. *e-prints, arXiv:1109.4299 [cs.IT]*, 2011.

[116] V. Pohl, F. Yang, and H. Boche. Phase retrieval from low rate samples. *e-prints, arXiv:1311.7045 [cs.IT]*, 2013.

[117] E. Porat and A. Rothschild. Explicit non-adaptive combinatorial group testing schemes. *Automata, Languages and Programming Lecture Notes in Computer Science*, 5125:748–759, 2008.

[118] E. Price. Efficient sketches for the set query problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 41–56, Jan 2011.

[119] E. Price and D. P. Woodruff. (1 + eps)-approximate sparse recovery. In *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 295–304, Oct 2011.

[120] H. J. Prömmel and A. Steger. RNC-approximation algorithms for the Steiner problem. In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science*, volume 1200, pages 559–570, 1997.

[121] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300 – 304, Jun 1960.

[122] T. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, Feb 2001.

[123] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 770–779, Jan 2000.

[124] R. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

[125] S. Sarvotham, D. Baron, and R. Baraniuk. Sudocodes - fast measurement and reconstruction of sparse signals. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 2804 –2808, July 2006.

[126] A. Schliep, D. Torney, and S. Rahmann. Group testing with dna chips: generating designs and decoding experiments. In *Proceedings of IEEE Bioinformatics Conference (CSB)*, pages 84 – 91, Aug. 2003.

[127] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

[128] Y. Shechtman, A. Beck, and Y. C. Eldar. Gespar: Efficient phase retrieval of sparse signals. *e-prints, arXiv:1301.1018[cs.IT]*, 2013.

[129] I. G. Shevtsova. An Improvement of Convergence Rate Estimates in the Lyapunov Theorem. *Doklady Mathematics*, 82(3):862–864, 2010.

[130] R. C. Singleton. Maximum distance q-nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, 1964.

[131] M. Sobel and R. M. Elashoff. Group testing with a new goal, estimation. *Biometrika*, 62(1):181–193, 1975.

[132] D. A. Spielman. Linear-time encodable and decodable error-correcting codes. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (SOTC)*, pages 388–397, May 1995.

[133] H. Takahashi and A. Matsuyama. An Approximate Solution for the Steiner Problem in Graphs. *Math. Jap.*, 24:537–577, 1980.

[134] I. Tamo and A. Barg. A family of optimal locally recoverable codes. *IEEE Transactions on Information Theory*, 60(8):4661–4676, Aug 2014.

[135] R. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[136] J. Tropp and A. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, Dec 2007.

[137] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, pages 4655–4666, 2007.

[138] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, pages 365–377, 1996.

[139] M. J. Wainwright. Information-theoretic limitations on sparsity recovery in the high-dimensional and noisy setting. *IEEE Transactions on Information Theory*, 55(12):5728 – 5741, Dec. 2009.

[140] I. Waldspurger, A. d'Aspremont, and S. Mallat. Phase recovery, maxcut and complex semidefinite programming. *Mathematical Programming*, 149(1-2):47–81, 2015.

[141] M. Wang, W. Xu, E. Mallada, and A. Tang. Sparse recovery with graph constraints: Fundamental limits and measurement construction. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM)*, pages 1871 –1879, March 2012.

[142] J. Wolf. Born again group testing: Multiaccess communications. *IEEE Transactions on Information Theory*, 31(2):185 – 191, Mar 1985.

[143] Y. Wu and S. Verdú. Rényi information dimension: Fundamental limits of almost lossless analog compression. *IEEE Transaction on Information Theory*, 56(8):3721–3748, 2010.

[144] Y. Wu and S. Verdú. Optimal phase transitions in compressed sensing. *e-prints, arXiv:1111.6822 [cs.IT]*, 2011.

[145] W. Xu and B. Hassibi. Efficient compressive sensing with deterministic guarantees using expander graphs. In *Proceedings of IEEE Information Theory Workshop (ITW)*, pages 414 –419, Sept. 2007.

[146] W. Xu, E. Mallada, and A. Tang. Compressive sensing over graphs. In *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM)*, pages 2087 –2095, April 2011.

[147] Y. Xuan, Y. Shen, N. Nguyen, and M. Thai. Efficient multi-link failure localization schemes in all-optical networks. *IEEE Transactions on Communications*, 61(3):1144–1151, March 2013.

[148] S. Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012.

[149] A. Zelikovsky. An 11/6-Approximation Algorithm for the Network Steiner Problem. *Algorithmica*, 9:463–470, 1993.

[150] A. Zelikovsky. Better Approximation Bounds for the Network and Euclidean Steiner Tree Problem. *Technical report CS-96-06, University of Virginia*, 1993.

[151] F. Zhang and H. Pfister. Verification Decoding of High-Rate LDPC Codes With Applications in Compressed Sensing. *IEEE Transactions on Information Theory*, 58(8):5042–5058, Aug 2012.

[152] L. Zhang, J. Luo, and D. Guo. Neighbor discovery for wireless networks via compressed sensing. *Performance Evaluation*, 70:475 –477, July 2013.

[153] Y. Zhao, Y. Chen, and D. Bindel. Towards unbiased end-to-end network diagnosis. *IEEE/ACM Transactions on Networking*, 17(6):1724–1737, 2009.

[154] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposiumon on Symbolic and Algebraic Computation*, pages 216–226, 1979.