Non-adaptive group testing: theory and design

Dissertation by

BUI VAN THACH

submitted to the Department of Informatics

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

SOKENDAI (The Graduate University for Advanced Studies)

September 2019

Abstract

The goal of group testing is to identify a few specific items satisfying a specific property in a large population of items at the lowest cost in terms of time and money. These specific items are referred to as defective items, and the other items are referred to as negative items. To achieve this goal, the population of items is grouped into (overlapping) subsets. A test is performed on each subset is to determine whether it satisfies the property. A test outcome is positive if the property holds and negative otherwise.

Defective items in a subset tend to make the outcome of the test positive. Recent advances in the definition of group testing have added a new type of item: inhibitors. An item in a subset is considered to be an inhibitor if it interferes with the identification of defective items; i.e., it tends to make the outcome of the test negative.

In this thesis, we consider the non-adaptive design approach to group testing in which all tests are independent and designed in advance. The central contributions of this thesis are in two tracks: efficient testing designs for minimizing the number of tests and low-complexity decoding schemes for efficiently identifying specific items.

Contents

List of Tables

List of Figures

Acknowledgments

Much time has passed since I began my PhD program, and things have dramatically changed. I've had the privilege to meet, talk, and exchange ideas with many people, both directly and indirectly, all of which has made me a better person. A few paragraphs are not enough for me to express all the gratitude and emotions I am feeling, but I will try my best to put them into words.

My foremost gratitude goes to my mentor – Isao Echizen. In retrospect, it was a wonderful privilege to be accepted as an intern in his lab six years ago (2013). He was kind, generous, and responsive, even though I was just an intern. As the end of my internship was coming, I asked to join his lab as a PhD student, a request that I have never regretted. Although he had suffered my poor presentation skills and broken English, he granted my request. I wish he could have envisioned how much trouble I would cause him over the coming five years before making his decision. It would have been better for him, but I was lucky because he did not get that vision. As a consequence, I became his student.

During my study, I did not know how to present my work in a simple way. It was really difficult for even experts in my field to understand my presentations. Although his workload with research and administrative tasks was heavy, he always spared time to discuss and revise my presentations till the last minute before my defense and also in regular meetings. As a result, my presentation skills improved significantly which may be the biggest improvement I made during my PhD years.

Perhaps the biggest headache I have to Echizen was my abrupt decision to switch from a non-theory person to a theory person. As a result, the papers I subsequently submitted to journals and conferences were not accepted for a year. It was a dark and scary time because I would have no scholarship if my study

was extended. Nonetheless, Echizen continued to support me by allowing and encouraging me to follow my passion. I remember the night he congratulated me on my paper being accepted even before I had read the notification. That lifted a big weight from my shoulders as I had met the university's requirements and could continue following my dreaming. We finally made it through a hard time. Although I am a stutterer but can manage to say right this minute is, thank you, Echizen, for everything you have done for me.

I would also like to thank Yumiko Seino, the lab secretary, for all the help she has given me over the years. She ensured that various university procedures in regard to my PhD program and business trips always went smoothly. She organized our lab meeting and other activities remarkably well and quickly smoothed out any problems. Her smiles and hospitality make me feel good every time I see her.

I am thankful to Minoru Kuribayashi who has been my co-supervisor since 2016. He is always friendly and supportive. Every time he visits Tokyo, he prioritizes sparing time to meet with me and Echizen to discuss our work. Even when he found serious mistakes in my proposed scheme or results during intensive meetings, he continued to encourage and have faith in me, especially when my research was not going well. His patience and support truly cultivated me.

It would have been impossible for me to enter the theoretical computer science (TCS) community if Mahdi Cheraghchi did not accept me as an intern. It was my luck to have a chance to work with him. He had recently moved to Imperial College London and was new to the school's procedures. However, he welcomed me to join his lab as a visiting student although I had nothing but my passion to show him at that time. We spent hours during our meetings discussing various aspects of our work. My knowledge broadened quickly during my internship. Even after I returned to Japan, Mahdi remotely taught me how to write a mathematical paper. With his continued guidance, I finally published a theoretical paper that helped me become a member of the TCS community. Your generosity and kindness reflect just how much of a beautiful kind soul you are, Mahdi.

Special thanks to Testuya Kojima for helping me take my first step into theory. He patiently checked my drafts and corrected them in details despite his busy workload. I also thank Roghayyeh Haghvirdinezhad for spending time with me discussing group testing and probability. Her questions usually revealed one of the

many gaps in my knowledge. She would then gently pointed out why my solution would not work.

I am indebted to Nguyen Dinh Thuc, who was my supervisor when I was an undergraduate student and who is my colleague now. Due to his tremendous help, support and guidance, I was able to start my research after my second year as an undergraduate student. Attending a seminar led by Ngo Quang Hung, who he had invited to present his work on group testing, was a turning point in my life. It made me realize that I wanted to do in future with my background which became my dream. I want to express my sincere gratitude for what you, thay Thuc, have done for me over the past 11 years. You have earned my admiration and respect and once again, I thank you.

Thanks in particular to my colleagues for their invaluable support: Hai Van, Hong Nhung, Quoc Binh, Thanh Binh, Xanh, Thy Thy, Sang, Thien, Quy, Hong Huy, Minh Duc, Hoang Quoc, and Bao Thu. My life outside academia would have been less colorful and joyful if not for my wonderful friends: Nam Hai, Dieu Van, Hoang Long, Nhu Ngoc, Cam Liem, Hoang Chuong, Anh Phap, Bich Tien, Dieu Van, Tuyet Le, Xuan Hoa, Thao Uyen, Diem, Luu, Tu Uyen, Nithin Varma, and Kass. Memories may fade as time passes, but at least we were toghether for a while. I have had so much fun playing with you guys!

To all the players in the Dota community, thank you so much for spending time with me. I have been playing this game for 11 years so far. While some players are skillful and clever, others are "noobs." We spent time together, making our opponents and noob teammates feel bad when we won or were the MVP. We shared and lived every moment in our games.

I am indebted to Zen Master Thich Nhat Hanh for his wonderful books which have enlightened my perspective. No rush, no panic. No matter what happens to me, I should live every moment of my life. "The present moment is filled with joy and happiness. If I am attentive, I will see it."

I would like to thank Mother Nature for her beauty and the source of all my creative ideas. It is always thrilling and overwhelming to catch natural phenomenons and then interpret them in my own way.

Above all, thanks to my family for their constant love and support of all my endeavors. Finally, the best thing about coming to Japan was that I met my wife

here. Words cannot express how grateful I am for you being a part of my life, Hong Ngoc.

“Everyone has to start somewhere.”
– Dota 2’s quote.

1 Introduction

1.1 Basics of Group Testing

1.1.1 Overview

Infectious diseases have been a lethal threat to communities throughout history. Robert Dorfman, an economist, solved the problem of identifying which draftees had syphilis in World War II [1]. The initial assumption was that the number of infected draftees was much smaller than the number of disinfected draftees. This was thus the problem of identifying a small group of items satisfying *specific properties* in a large group of items. Such items are usually referred to as *defective items*, and the other items are usually referred to as *negative items.* In the infectious disease scenario, defective items were infected draftees, and negative items were disinfected draftees.

Testing items one by one is costly in terms of time and money. It is possible to reduce this Herculean task by combining items into groups and then testing the groups, i.e., *group testing* (GT). Testing design is diverse and depends on the classification of defective items. There are two approaches to designing such tests. The first is *adaptive design* in which there are several testing stages, and the design of each stage depends on the designs of the previous stages [2–4]. This design is time-consuming in practice. The second is *non-adaptive design* in which all tests are designed in advance, and the tests are performed in parallel [5]. Because the non-adaptive design saves time and is efficient with a parallel architecture, the focus of this thesis is on non-adaptive design.

Suppose that there are n items indexed from 1 to n and that defective set $D \subset [n] = \{1, \ldots, n\}$, where $|D| \leq d \ll n$. A test on a subset of n items is designed to determine whether the subset satisfies the specific properties. If the properties are satisfied, the test outcome is *positive.* Otherwise, the test outcome is negative. In general, the manner in which D is determined to be a defective set and the manner in which the members of D are presented in a test define the test outcome. The main goal of group testing is to efficiently classify all items.

The paradigm of group testing is illustrated in Figure 1.1 in terms of the theoretical and practical models. The group testing paradigm encompasses two procedures: encoding and decoding. In the theoretical model, encoding is used to

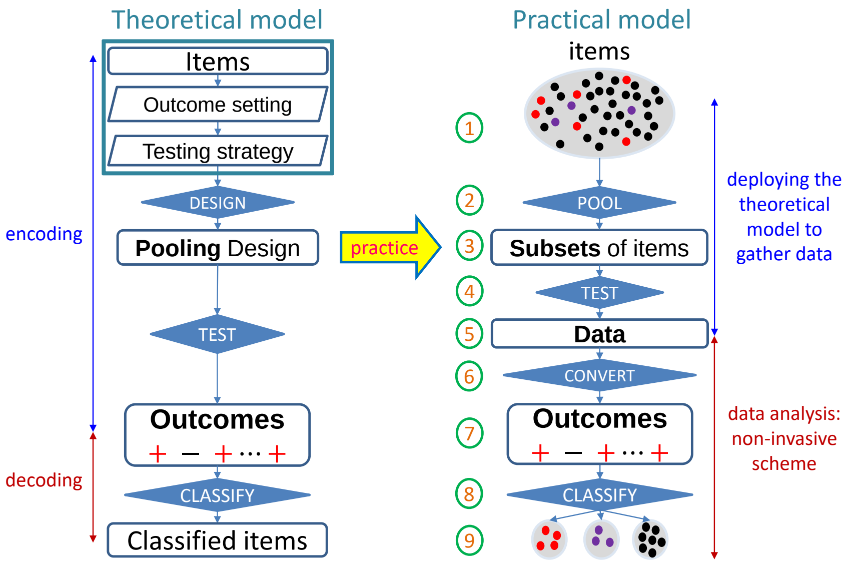


Figure 1.1: Group testing paradigm.

create a pooling design and then use it to perform tests to get outcomes. Decoding is used to classify the items from the outcomes. There are nine steps in the practical model, with steps 4 to 6 corresponding to one step in the theoretical model.

Feasibility of group testing: In the practical model, a set of items might consist of various types of items. The first step in implementing the theoretical model in practice is to determine what the *items* are and what the *specific items* are. Next, Steps 1 to 3 in the theoretical model are mapped to those steps in the practical model. In Steps 4 to 7 in the practical model, the items are divided into (overlapping) subsets so that the data gathered from the tests textitcan be converted into a binary vector. Finally, after the outcomes are obtained, all items can be classified by using the decoding algorithm in the theoretical model (Steps 7 to 9). The most important issue in this testing is deciding the *items* and the *specific items*. The next most important one is deciding how to design the tests so that the data gathered can be converted into a binary vector. This issue is the *basic* difference between the theoretical model and the practical one, and it naturally leads to the realization that the overall cost largely depends on the cost of *processing each item*. Therefore, minimizing *the number of items in each test* is

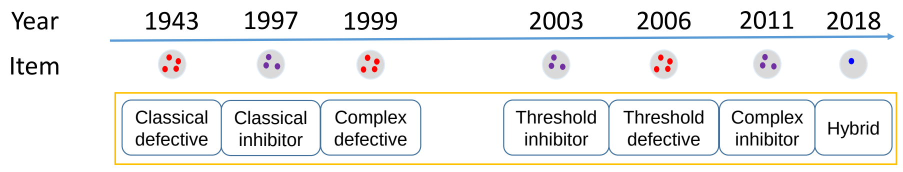


Figure 1.2: History of group testing.

important.

There are various models of group testing, and they heavily depend on the types of items. Note that types of items and their relationships to the test outcome are intertwined. A broad view of group testing models is shown in Figure 1.2. Four types of items have been defined so far: defectives, inhibitors, hybrids, and negatives. Negatives always make the outcome of a test negative. Defectives tend to make the outcome of a test positive. Inhibitors tend to make the outcome of a test negative. Hybrids combine the properties of both defectives and inhibitors. In addition, each item type has sub-types. The history of group testing based on item classification is illustrated in Figure 1.2.

The genesis of group testing which was sparked by the introduction of *classical defectives* was described by Dorfman [1] in 1943. More than a half century later, in 1997, new types of items began to proliferate. The *classical inhibitor* item was introduced [6] in 1997. Two years later, Torney [7] presented the concept of *complex defectives*, a generalization of classical defectives. Threshold inhibitors [8], threshold defectives [9], and complex inhibitors [10] were defined to solve specific problems in biology and computer networking. Recently, Bui et al. [11] introduced hybrid items for matching the model in neuroscience. Since this thesis focuses on classical defectives, classical inhibitors, and threshold defectives, we define them precisely and illustrate their relationships in Figure 1.3.

In classical group testing (CGT), the outcome of a test on a subset of items is positive if the subset has at least one item in D, and negative otherwise. CGT has been intensively studied since its inception [1] and has a wide range of its applications such as DNA library screening [12–16], data forensics [17], data streaming [18], high speed computer networks [19], multiaccess channels [8, 20–23], multilabel classification [24], compressed sensing [25, 26], similarity searching [27], traitor tracing [28], pay-tv [29], cryptography [30], sparse recovery [31], database

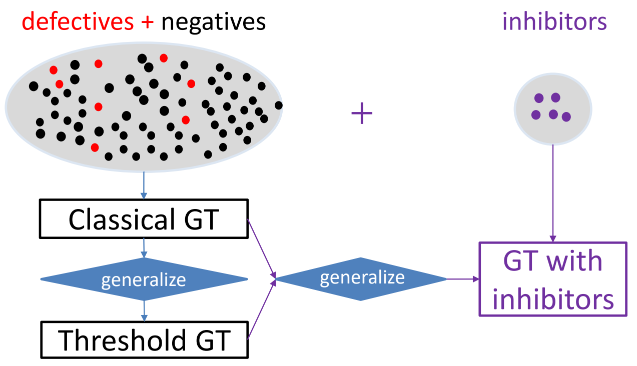


Figure 1.3: Group testing classification.

system [32], and cyber security [33–35].

In threshold group testing (TGT), given two integer parameters $0 \leq \ell < u \leq d$, the outcome of a test on a subset of items is negative if the subset has up to ℓ items in D, is positive if the subset has at least u items in D, and arbitrary otherwise. TGT reduces to CGT when $u = 1$. There is a sparsity of work and applications related to TGT [9, 36–39]. Learning a hidden graph is a promising application of TGT [40–44].

With the development of molecular biology [6], inhibitor items were adaptively introduced. An item is considered to be an inhibitor if it interferes with the identification of defective items in a test [45–50]. In this case, group testing is called *GT with inhibitors* (GTI).

Because non-adaptive design is exclusively considered here, each model of group testing described above is treated under the non-adaptive design regime. Specifically, non-adaptive CGT (NACGT), non-adaptive TGT (NATGT), and non-adaptive GTI (NAGTI) are the focus throughout this thesis.

1.1.2 Main challenges

There are two main challenges in group testing: designing tests and creating efficient decoding schemes for classifying items. Since CGT has had solid foundation since its inception, these challenges have mostly been overcome for CGT. However, nonrandom design has not been investigated well, though it plays a crucial role in practice. It is thus important to improve nonrandom design.

In contrast to the situation with CGT, these challenges have not been addressed well for the other group testing models, i.e., non-classical group testing, such as TGT and GTI. Prior to our work, the decoding time for non-classical group testing is linear or exponential wrt the number of items. Therefore, non-adaptive group testing is impractical. As a consequence, test designs in the encoding procedure are needed that are practical for application and decoding schemes are needed for efficiently classifying items.

1.2 Contributions

The central contributions of this thesis are in two tracks: efficient testing designs for the non-adaptive approach and low-complexity decoding schemes.

1.2.1 Non-adaptive classical group testing

If there are t tests, they can be represented as a $t \times n$ measurement matrix. We have answered the question of whether there exists a scheme such that a larger measurement matrix, built from a given $t \times n$ measurement matrix, can be used to identify up to d defective items in time $O(t \log_2 n)$. In the meantime, a $t \times n$ nonrandom measurement matrix with $t = O\left(\frac{d^2 \log_2^2 n}{(\log_2(d \log_2 n) - \log_2 \log_2(d \log_2 n))^2}\right)$ can be obtained to identify up to d defective items in time $\mathsf{poly}(t)$ (polynomial of t). This is much better than the best well-known bound, $t = O\left(d^2 \log_2^2 n\right)$. For the special case $d = 2$, there exists an efficient nonrandom construction in which at most two defective items can be identified in time $4 \log_2^2 n$ using $t = 4 \log_2^2 n$ tests. Numerical results show that our proposed scheme is more practical than existing ones, and experimental results confirm our theoretical analysis. In particular, up to $2^7 = 128$ defective items can be identified in less than 16s even for $n = 2^{100}$.

This work was done jointly with Isao Echizen, Minoru Kuribayashi, Tetsuya Kojima, and Roghayyeh Haghvirdinezhad, and was published at *Journal of Information Processing* [51].

1.2.2 Non-adaptive threshold group testing

We consider non-adaptive threshold group testing for identification of up to d defective items in a set of n items, where a test is positive if it contains at least $2 \leq u \leq d$ defective items, and negative otherwise. The defective items can be identified using $t = O\left(\left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} \left(u \log \frac{d}{u} + \log \frac{1}{\epsilon}\right) \cdot d^2 \log n\right)$ tests with probability at least $1 - \epsilon$ for any $\epsilon > 0$ or $t = O\left(\left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} d^3 \log n \cdot \log \frac{n}{d}\right)$ tests with probability 1, where log is the logarithm of base 2. The decoding time is $t \times \mathsf{poly}(d^2 \log n)$. This result significantly improves the best known results for decoding non-adaptive threshold group testing: $O(n \log n + n \log \frac{1}{\epsilon})$ for probabilistic decoding [38], where $\epsilon > 0$, and $O(n^u \log n)$ for deterministic decoding [39].

This work was done jointly with Isao Echizen, Minoru Kuribayashi, and Mahdi Cheraghchi, and was published at *2018 IEEE International Symposium on Information Theory (ISIT)* as well as in *IEEE Transactions on Information Theory* [52].

1.2.3 Non-adaptive group testing with inhibitors

Identification of up to d defective items and up to h inhibitors in a set of n items is the main task of non-adaptive group testing with inhibitors. A test outcome on a subset of items is positive if the subset contains at least one defective item and no inhibitors, and negative otherwise. We present two decoding schemes for efficiently identifying the defective items and the inhibitors in the presence of e erroneous outcomes in time $\mathsf{poly}(d, h, e, \log_2 n)$, which is sublinear to the number of items. This decoding complexity significantly improves the state-of-the-art schemes in which the decoding time is linear to the number of items, i.e., $\mathsf{poly}(d, h, e, n)$. Moreover, each column of the measurement matrices associated with the proposed schemes can be nonrandomly generated in polynomial order of the number of rows. As a result, one can save space for storing them. Simulation results confirm our theoretical analysis. When the number of items is sufficiently large, the decoding time in our proposed scheme is smallest in comparison with existing work. In addition, when some erroneous outcomes are allowed, the number of tests in the proposed scheme is often smaller than the number of tests in existing work.

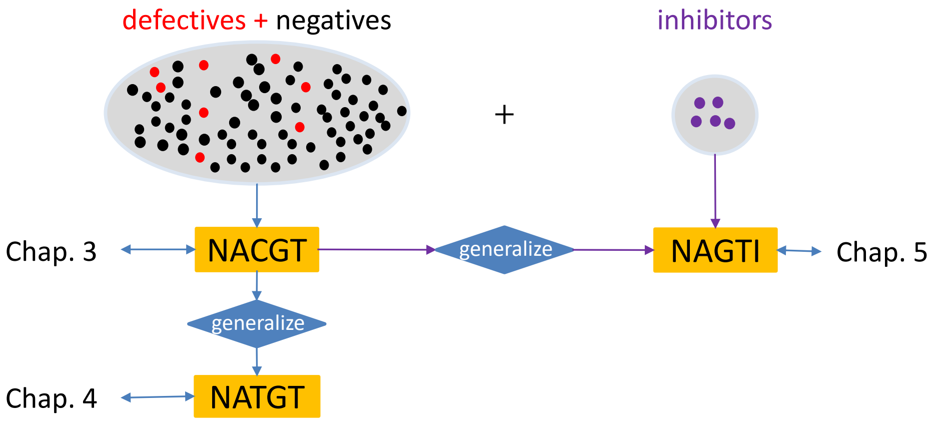


Figure 1.4: Thesis outline.

This work was done jointly with Isao Echizen, Minoru Kuribayashi, and Tetsuya Kojima, and was published at *2019 International Conference on Theory and Applications of Models of Computation (TAMC)* [53].

1.3 Organization of the thesis

The basic organization of this thesis is illustrated in Figure 1.4. We start with the preliminaries in Chapter 2. In Chapter 3, we present an improved nonrandom design and efficient decoding schemes for NACGT. The generalization of NACGT, i.e., NATGT, is addressed in Chapter 4 in which an efficient decoding algorithm is proposed. With inhibitor items added to NACGT, a new model of group testing is formed with the moniker NAGTI and addressed in Chapter 5. Finally, we conclude in Chapter 6 with the major open questions.

“What we know is a drop,
what we don’t know is an ocean.”
– Isaac Newton.

2
Preliminaries

2.1 Notation

A multiset, denoted with a capital letter with an "$\star$," is a set that allows multiple instances of its elements. For example, $A^\star = \{1, 2, 2\}$ is a multiset.

The union of l vectors is defined as follows. Given l binary vectors $\mathbf{y}_w = (y_{1w}, y_{2w}, \ldots, y_{uw})^T$ for $w = 1, \ldots, l$ and some integer $u \geq 1$, their union is defined as vector $\mathbf{y} = \vee_{i=1}^l \mathbf{y}_i = (\vee_{i=1}^l y_{1i}, \ldots, \vee_{i=1}^l y_{ui})^T$, where $\vee$ is the OR operator. The intersection of l columns of a $t \times n$ matrix $\mathcal{T}$ is defined as $\bigwedge_{i=1}^l \mathcal{T}_{j_i} = \left(\bigwedge_{i=1}^l m_{1j_i}, \ldots, \bigwedge_{i=1}^l m_{tj_i} \right)^T$. Notation $[m]$ represents set $\{1, 2, \ldots, m\}$.

For consistency, we use capital calligraphic letters for matrices, non-capital letters for scalars, capital letters for sets, and bold letters for vectors. All matrices here are binary. Following are several notations used throughout this thesis:

- $n; d; \mathbf{x} = (x_1, \ldots, x_n)^T; \mathbf{y}$: number of items; maximum number of defective items; representation vector of n items; representation vector of outcomes. For simplicity, suppose that n is the power of 2.

- $|\cdot|$: weight; i.e, number of non-zero entries of input vector or cardinality of input set.

- $\odot; \otimes; \circledcirc$: operation for NACGT; operation for NATGT; tensor product.

- $\mathcal{S}; \mathcal{B}$: $k \times n$ measurement matrices used to identify at most one defective item, where $k = 2\log_2 n$.

- $\mathcal{S}_j; \mathcal{B}_j; \mathcal{M}_j; \mathcal{M}_{i,*}; \mathcal{T}_{i,*}; \mathcal{T}_j; \mathcal{G}_{i,*}$: column j of matrix $\mathcal{S}$; column j of matrix $\mathcal{B}$; column j of matrix $\mathcal{M}$; row i of matrix $\mathcal{M}$; row i of matrix $\mathcal{T}$; column j of matrix $\mathcal{T}$; row i of matrix $\mathcal{G}$.

- $\mathrm{diag}(\cdot)$: diagonal matrix constructed by input vector.

- $\mathsf{supp}(\mathbf{v})$: support set for vector $\mathbf{v} = (v_1, \ldots, v_w)$, where $\mathsf{supp}(\mathbf{v}) = \{j \mid v_j \neq 0\}$. For example, the support vector for $\mathbf{v} = (1, 0, 0, -\infty)$ is $\mathsf{supp}(\mathbf{v}) = \{1, 4\}$.

- $\mathrm{e}, \log, \ln, \exp(\cdot)$: base of natural logarithm, logarithm of base 2, natural logarithm, exponential function.

- $\lceil x \rceil$, $\lfloor x \rfloor$: ceiling and floor functions of x.
- $\mathsf{W}(x)$: the Lambert W function in which $\mathsf{W}(x)\mathrm{e}^{\mathsf{W}(x)} = x$ and $\mathsf{W}(x) = O(\ln x - \ln\ln x)$.

Specific notations or other meanings of the notations above are specifically declared case by case.

2.2 Measurement matrix

For vector $\mathbf{x} = (x_1, \ldots, x_n)^T$, $x_j = 0$ means that item j is negative, and $x_j \neq 0$ means that item x_j is defective. For a $t \times n$ binary measurement matrix $\mathcal{T} = (t_{ij})$, item j is represented by column $\mathcal{T}_j$ and test i is represented by row $\mathcal{T}_{i,*}$; $t_{ij} = 1$ if item j belongs to test i, and $t_{ij} = 0$ otherwise.

Let $\mathsf{test}(S)$ be the test on subset $S \subseteq [n]$. The outcome of the test is either positive (1) or negative (0) and depends on the definition of D and S. The non-adaptive tests on n items using $\mathcal{T}$ are defined as

$$\mathbf{y} = \mathcal{T} \bullet \mathbf{x} = \begin{bmatrix} \mathsf{test}\left(\mathsf{supp}(\mathcal{T}_{1,*}) \cap \mathsf{supp}(\mathbf{x})\right) \\ \vdots \\ \mathsf{test}\left(\mathsf{supp}(\mathcal{T}_{t,*}) \cap \mathsf{supp}(\mathbf{x})\right) \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_t \end{bmatrix}, \tag{2.1}$$

where $y_i = \mathsf{test}\left(\mathsf{supp}(\mathcal{T}_{i,*}) \cap \mathsf{supp}(\mathbf{x})\right)$ is the outcome of test i corresponding to row $\mathcal{T}_{i,*}$, and $\bullet$ is the test operator. The procedure to obtain $\mathbf{y}$ is called *encoding*. The procedure to recover $\mathbf{x}$ from $\mathbf{y}$ and $\mathcal{T}$ is called *decoding*.

For CGT and TGT, notation $\bullet$ can be explicitly defined and vector $\mathbf{x}$ is viewed as a binary vector in which $x_j = 1$ (resp., $x_j = 0$) means item j is defective (resp., negative). With CGT, to avoid ambiguity, we change notation $\bullet$ to $\odot$ and use a $k \times n$ measurement matrix $\mathcal{M}$ instead of the $t \times n$ matrix $\mathcal{T}$. The outcome vector $\mathbf{y}$ in (2.1) is equal to

$$\mathbf{y} = \mathcal{M} \odot \mathbf{x} = \begin{bmatrix} \mathcal{M}_{1,*} \odot \mathbf{x} \\ \vdots \\ \mathcal{M}_{k,*} \odot \mathbf{x} \end{bmatrix} = \begin{bmatrix} \bigvee_{j=1}^{n} m_{1j} \wedge x_j \\ \vdots \\ \bigvee_{j=1}^{n} m_{kj} \wedge x_j \end{bmatrix} = \bigvee_{j=1, x_j=1}^{n} \mathcal{M}_j = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}, \tag{2.2}$$

where $y_i = \mathcal{M}_{i,*} \odot \mathbf{x} = \bigvee_{j=1}^{n} x_j \wedge m_{ij} = \bigvee_{j=1, x_j=1}^{n} m_{ij}$ for $i = 1, \ldots, k$.

With u-TGT, to avoid ambiguity, we change notation $\bullet$ to $\otimes$. Outcome vector $\mathbf{y}$ in (2.1) is equal to

$$\mathbf{y} = \mathcal{T} \otimes \mathbf{x} = \begin{bmatrix} \mathcal{T}_{1,*} \otimes \mathbf{x} \\ \vdots \\ \mathcal{T}_{t,*} \otimes \mathbf{x} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_t \end{bmatrix}, \tag{2.3}$$

where $y_i = \mathcal{T}_{i,*} \otimes \mathbf{x} = 1$ if $\sum_{j=1}^{n} t_{ij} x_j \geq u$, and $y_i = \mathcal{T}_{i,*} \otimes \mathbf{x} = 0$ otherwise for $i \in [t]$.

2.3 Disjunct matrices

2.3.1 General case

Disjunct matrices were first introduced by Kautz and Singleton [54] as *superimposed codes* and then generalized by Stinson and Wei [55] and D'yachkov et al. [56]. The formal definition of a disjunct matrix is as follows.

Definition 1 *An $m \times n$ binary matrix $\mathcal{T}$ is called a $(d, r; z]$-disjunct matrix if, for any two disjoint subsets $S_1, S_2 \subset [n]$ such that $|S_1| = d$ and $|S_2| = r$, there exists at least z rows in which there are all 1's among the columns in S_2 while all the columns in S_1 have 0's; i.e., $\left| \bigcap_{j \in S_2} \mathsf{supp}\,(\mathcal{T}_j) \setminus \bigcup_{j \in S_1} \mathsf{supp}\,(\mathcal{T}_j) \right| \geq z$.*

Matrix $\mathcal{M}$ is usually referred to as a $(d, r; z]$-disjunct matrix. Parameter $e = \lfloor (z-1)/2 \rfloor$ is referred to as the *error tolerance* of a disjunct matrix. It is clear that for any $d' \leq d$, $r' \leq r$, and $z' \leq z$, a $(d, r; z]$-disjunct matrix is also a $(d', r'; z']$-disjunct matrix. An illustration of $\mathcal{M}$ is as follows.

$$\mathcal{M} = \begin{array}{c|cccccccc|c}
 & & \multicolumn{2}{c}{\overbrace{\hphantom{1 \quad 1}}^{r}} & & \multicolumn{3}{c}{\overbrace{\hphantom{0 \quad 0 \quad 0}}^{d}} & & \\
\hline
 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\
 & \cdots & 1 & 1 & \cdots & 0 & 0 & 0 & \cdots & \text{row } 1 \\
 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\
 & \cdots & 1 & 1 & \cdots & 0 & 0 & 0 & \cdots & \text{row } z \\
 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\
 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\
\hline
\end{array}$$

Chen et al. [57] gave an upper bound on the number of rows for $(d, r; z]$-disjunct matrices as follows.

Theorem 2.1 *[57, Theorem 3.2] For any positive integers d, r, z, and n with $p = d+r \leq n$, there exists a $t \times n$ $(d, r; z]$-disjunct matrix with $t = O\left(z\left(\frac{p}{r}\right)^r \left(\frac{p}{d}\right)^d p \ln \frac{n}{p}\right)$.*

Cheraghchi [36] proposed a nonrandom construction for $(d, r; z]$-disjunct matrices in which the number of tests becomes larger than those of constructions as d or r increases.

Theorem 2.2 *[36, Lemma 29] For any positive integers d, r, z and n with $d + r \leq n$, there exists an $m \times n$ nonrandom $(d, r; z]$-disjunct matrix where $m = O\left((rd \ln n + z)^{r+1}\right)$. Moreover, each column of the matrix can be generated in time $\mathsf{poly}(m)$.*

2.3.2 Special case

We now consider the special case of a $(d, r; z]$-disjunct matrix in which $r = z = 1$. In this case, the $(d, r; z]$-disjunct matrix is called a d-disjunct matrix. To gain insight into d-disjunct matrices, we present the concept of an identity matrix inside a set of vectors.

Definition 2 *Any c column vectors with the same size contain a $c \times c$ identity matrix if a $c \times c$ identity matrix could be obtained by placing those columns in an appropriate order.*

Note that there may be more than one identity matrix inside those c vectors. For example, let $\mathbf{b}_1, \mathbf{b}_2$, and $\mathbf{b}_3$ be vectors of size 4×1:

$$\mathbf{b}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \mathbf{b}_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{b}_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}. \tag{2.4}$$

Then, $(\mathbf{b}_1, \mathbf{b}_2)$ and $(\mathbf{b}_2, \mathbf{b}_3)$ contain 2×2 identify matrices, whereas $(\mathbf{b}_1, \mathbf{b}_3)$ does not.

$$\begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Definition 1 is then restated for $r = z = 1$ as follows.

Definition 3 *A binary $t \times n$ matrix is called a d-disjunct matrix iff there exists a $(d+1) \times (d+1)$ identity matrix in a set of $(d+1)$ columns arbitrarily selected from the matrix.*

If $\mathcal{M}$ is d-disjunct, a vector $\mathbf{x}$ can be recovered from $\mathbf{y} = \mathcal{M} \odot \mathbf{x}$. With naive decoding, all items belonging to tests with negative outcomes are removed; the items remaining are considered to be defective. The decoding complexity of this approach is $O(tn)$, which is fairly high and not preferable in practice.

A matrix is said to be nonrandom if its columns are deterministically generated without using randomness. In contrast, a matrix is said to be random if its columns are randomly generated. We thus classify construction types on the basis of the time it takes to generate a matrix entry. A $t \times n$ matrix is said to be weakly explicit if each of its columns is generated in time (and space) $O(tn)$. It is said to be strongly explicit if each of its columns is generated in time (and space) $\mathsf{poly}(t)$. We first present a weakly explicit construction of a d-disjunct matrix.

Theorem 2.3 *[58, Theorem 1] Given $1 \leq d < n$, there exists a nonrandom $t \times n$ d-disjunct matrix that can be constructed in time $O(tn)$, where $t = O(d^2 \log n)$.*

Moreover, the decoding time is $O(tn)$, and each column is generated in time (and space) $O(tn)$.

The second construction is strongly explicit.

Theorem 2.4 *[59, Corollary 5.1] Given $1 \leq d < n$, there exists a random $t \times n$ d-disjunct matrix that can be decoded in time $\mathsf{poly}(t) = O(d^{11}\log^{17} n)$, where $t = 4800d^2\log n = O(d^2 \log n)$. Each column can be generated in time $O(t^2 \log n)$ and space $O(t \log n)$. There also exists a matrix that can be nonrandomly constructed in time $\mathsf{poly}(t, n)$ and space $\mathsf{poly}(t)$ while the construction time and space for each column of the matrix remain the same.*

Theorem 2.5 *[60, Theorem 16] Let $1 \leq d \leq n$. There exists a strongly explicit $k \times n$ d-disjunct matrix with $k = O(d^2 \log n)$ such that for any $k \times 1$ input vector, the decoding procedure returns the set of defective items if the input vector is the union of up to d columns of the matrix in $\mathsf{poly}(k)$ time. Moreover, each column of $\mathcal{M}$ can be generated in time $O(k^2 \log n)$.*

Finally, the last construction is nonrandom. We analyze this construction in detail for later comparison. Although the precise formulas were not explicitly given in [59], they can be derived.

Theorem 2.6 *[59, Corollary C.1] Given $1 \leq d < n$, a nonrandom $t \times n$ d-disjunct matrix can be decoded in time $O\left(\frac{d^9(\log n)^{16+1/3}}{(\log(d\log n))^{7+1/3}}\right) = \mathsf{poly}(t)$, where $t = O(d^2\log^2 n)$. Moreover, each entry (column) can be generated in time (and space) $O(t)$ ($O(t^{3/2})$). When $d = 2$, the number of tests is $2\log n \times (2\log n - 1)$, the decoding time is longer than $\frac{2^9(\log n)^{16+1/3}}{(\log(2\log n))^{7+1/3}}$, and each entry is generated in time $\log^2 n$ and space $\log n$.*

The procedure for obtaining $\mathbf{x}$ from $\mathbf{y}$ is denoted as $\mathsf{supp}(\mathbf{x}) = \mathsf{decode}(\mathcal{M}, \mathbf{y})$. Since $\mathbf{y}$ may not be $\mathcal{M} \odot \mathbf{x}$, the function $\mathsf{decode}(\mathcal{M}, \mathbf{y})$ may produce an empty set or a subset that is not equal to $\mathsf{supp}(\mathbf{x})$ for $\mathbf{y} \neq \mathcal{M} \odot \mathbf{x}$.

2.4 Reed-Solomon codes

In this subsection, we first review the concept of $(\eta, r, \varphi)_q$ code C:

Definition 4 *Let η, r, φ, q be positive integers. An $(\eta, r, \varphi)_q$ code is a subset of Σ^η such that*

1. *Σ is a finite field and is called the alphabet of the code: $|\Sigma| = q$. Here we set $\Sigma = \mathbb{F}_q$.*
2. *Each codeword is considered to be a vector of $\mathbb{F}_q^{\eta \times 1}$.*
3. *$\varphi = \min_{\mathbf{x},\mathbf{y} \in C} \Delta(\mathbf{x}, \mathbf{y})$, where $\Delta(\mathbf{x}, \mathbf{y})$ is the number of positions in which the corresponding entries of $\mathbf{x}$ and $\mathbf{y}$ differ.*
4. *The cardinality of C, i.e., $|C|$, is at least q^r.*

Parameters η, r, φ, and q are the block length, dimension, minimum distance, and alphabet size of C. If the minimum distance is not considered, we refer to C as $(\eta, r)_q$. Given a full-rank $\eta \times r$ matrix $\mathcal{G} \in \mathbb{F}_q^{\eta \times r}$, suppose that, for any $\mathbf{y} \in C$, there exists a message $\mathbf{x} \in \mathbb{F}_q^r$ such that $\mathbf{y} = \mathcal{G}\mathbf{x}$. In this case, C is called a linear code and denoted as $[v, r, \varphi]_q$. Let $\mathcal{M}_C$ denote an $\eta \times q^r$ matrix in which the columns are the codewords in C.

Reed-Solomon (RS) codes [61] are constructed by applying a polynomial method to a finite field $\mathbb{F}_q$. Here we review a common and widely used RS code, an $[\eta, r, \varphi]_q$-code C in which $|C| = q^r$ and $\varphi = \eta - r + 1$. Since φ is determined from η and r, we refer to $[\eta, r, \varphi]_q$-RS code as $[\eta, r]_q$-RS code.

2.5 Tensor product

Given an $f \times n$ matrix $\mathcal{A}$ and an $s \times n$ matrix $\mathcal{S}$, their tensor product $\circledcirc$ is defined as

$$\mathcal{R} = \mathcal{A} \circledcirc \mathcal{S} = \begin{bmatrix} \mathcal{S} \times \mathrm{diag}(\mathcal{A}_{1,*}) \\ \vdots \\ \mathcal{S} \times \mathrm{diag}(\mathcal{A}_{f,*}) \end{bmatrix} = \begin{bmatrix} a_{11}\mathcal{S}_1 & \dots & a_{1n}\mathcal{S}_n \\ \vdots & \ddots & \vdots \\ a_{f1}\mathcal{S}_1 & \dots & a_{fn}\mathcal{S}_n \end{bmatrix}, \tag{2.5}$$

where diag(.) is the diagonal matrix constructed by the input vector, $\mathcal{A}_{h,*} = (a_{h1}, \dots, a_{hn})$ is the hth row of $\mathcal{A}$ for $h = 1, \dots, f$, and $\mathcal{S}_j$ is the jth column of $\mathcal{S}$ for $j = 1, \dots, n$. The size of $\mathcal{R}$ is $r \times n$, where $r = fs$. One can imagine that an entry a_{hj} of matrix $\mathcal{A}$ would be replaced by the vector $a_{hj}\mathcal{S}_j$ after the tensor product is used. For instance, suppose that $f = 2, s = 3$, and $n = 4$. Matrices $\mathcal{A}$ and $\mathcal{S}$ are defined as

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad \mathcal{S} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.6}$$

Then $\mathcal{R} = \mathcal{A} \circledcirc \mathcal{S}$ is

$$\mathcal{R} = \mathcal{A} \circledcirc \mathcal{S} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \circledcirc \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.7}$$

$$= \begin{bmatrix} 1 \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & 0 \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & 1 \times \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} & 0 \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \\ \\ 0 \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & 1 \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & 1 \times \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} & 1 \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

“We want peace, total peace and
nothing but peace, and even if
we have to fight the bloodiest battle,
we are going to get it.”
– Anonymous.

3 Non-adaptive classical group testing

3.1 Introduction

If t tests are needed to identify up to d defective items among n items, they can be seen as a $t \times n$ measurement matrix. The procedure to get the matrix is called *construction*, the procedure to get the outcome of t tests using the measurement matrix is called *encoding*, and the procedure to get the defective items from t outcomes is called *decoding*. Note that the encoding procedure includes the construction procedure. The objective of NAGT is to design a scheme such that all defective items are "efficiently" identified from the encoding and decoding procedures. Six criteria determine the efficiency of a scheme: measurement matrix construction type, number of tests needed, decoding time, time needed to generate an entry for the measurement matrix, space needed to generate a measurement matrix entry, and probability of successful decoding. The last criterion reduces the number of tests and/or the decoding complexity. With high probability, Cai et al. [62] and Lee et al. [63] achieved a low number of tests and decoding complexity, namely $O(t)$, where $t = O(d \log d \cdot \log n)$. However, the construction type is random, and the whole measurement matrix must be stored for implementation, so it is limited to real-time applications. For example, in a data stream [18], routers have limited resources and need to be able to access the column in the measurement matrix assigned to an IP address as quickly as possible to perform their functions. The schemes proposed by Cai et al. [62] and Lee et al. [63], therefore, are inadequate for this application.

For exact identification of defective items, there are four main criteria to be considered: measurement matrix construction type, number of tests needed, decoding time, and time needed to generate measurement matrix entry. The measurement matrix is nonrandom if it always satisfies the preconditions after the construction procedure with probability 1. It is random if it satisfies the preconditions after the construction procedure with some probability. A $t \times n$ measurement matrix is more practical if it is nonrandom, t is small, the decoding time is a polynomial of t ($\mathsf{poly}(t)$), and the time to generate its entry is also $\mathsf{poly}(t)$. However, there is always a trade-off between these criteria.

Kautz and Singleton [54] proposed a scheme in which each entry in a $t \times n$ measurement matrix can be generated in $\mathsf{poly}(t)$, where $t = O(d^2 \log^2 n)$. However, the decoding time is $O(tn)$. Indyk et al. [59] reduced the decoding time to $\mathsf{poly}(t)$ while maintaining the order of the number of tests and the time to generate the entries. However, the number of tests in a nonrandom measurement matrix is not optimal.

In term of the pessimum number of tests, Guruswami and Indyk [64] proposed a linear-time decoding scheme in accordance with the number of tests of $O(d^4 \log n)$. To achieve an optimal bound on the number of tests, i.e., $O(d^2 \log n)$, while maintaining a decoding time of $\mathsf{poly}(t)$ and keeping the entry computation time within $\mathsf{poly}(t)$, Indyk et al. [59] proposed a random construction. Although they tried to derandomize their schemes, it takes $\mathsf{poly}(t, n)$ time to construct such matrices, which is impractical when d and n are sufficiently large.

Cheraghchi [65] achieved similar results. However, his proposed scheme can deal with the presence of noise in the test outcomes. Porat and Rothschild [58] showed that it is possible to construct a nonrandom $t \times n$ measurement matrix in time $O(tn)$ while maintaining the order of the number of tests, i.e., $O(d^2 \log n)$. However, each entry in the resulting matrix is identified after the construction is completed. This is equivalent to each entry being generated in time $O(tn)$. If we reduce the number of tests, the nonrandom construction proposed by Indyk et al. [59] is the most practical.

3.2 Contributions

3.2.1 Overview

There are two main contributions in this work. First, we have answered the question of whether there exists a scheme such that a larger measurement matrix, built from a given $t \times n$ measurement matrix, can be used to identify up to d defective items in time $O(t \log n)$. Second, a $t \times n$ nonrandom measurement matrix with $t = O\left(\frac{d^2 \log^2 n}{(\log(d \log n) - \log\log(d \log n))^2}\right)$ can be obtained to identify up to d defective items in time $\mathsf{poly}(t)$. This is much better than the best well-known bound $t = O\left(d^2 \log^2 n\right)$. There is a special case for $d = 2$ in which there exists a $4 \log^2 n \times n$ nonrandom measurement matrix such that it can be used to identify up to two defective items in time $4 \log^2 n$. Numerical results show that our proposed scheme is the most practical and experimental results confirm our theoretical analysis. For instance, at most $2^7 = 128$ defective items can be identified in less than 16s even for $n = 2^{100}$.

3.2.2 Comparison

We compare variants of our proposed scheme with existing schemes in Table 3.1. As mentioned above, six criteria determine the efficiency of a scheme: measurement matrix construction type, number of tests needed, decoding time, time needed to generate measurement matrix entry, space needed to generate a measurement matrix entry, and probability of successful decoding. Since the last criterion is only used to reduce the number of tests, it is not shown in the table. If the number of tests and the decoding time are the top priorities, the construction in $\langle \mathbf{11} \rangle$ is the best choice. However, since the probability of successful decoding is at least $1 - \epsilon$ for any $\epsilon > 0$, some defective items may not be identified.

From here on, we assume that the probability of successful decoding is 1; i.e., all defective items are identified. There are trade-offs among the first five criteria. When $d = 2$, the number of tests with our proposed scheme ($\langle \mathbf{8} \rangle$) is slightly larger than that with $\langle 7 \rangle$, although our proposed scheme has the best performance for the remaining criteria. When $d > 2$, the comparisons are as follows. First, if the generation of a measurement matrix must be certain, the best choices are $\langle 1 \rangle, \langle 2 \rangle, \langle \mathbf{3} \rangle, \langle \mathbf{4} \rangle, \langle 5 \rangle$, and $\langle \mathbf{6} \rangle$. Second, if the number of tests must be as low as possible, the best choices are $\langle 2 \rangle, \langle 5 \rangle$, and $\langle 9 \rangle$. Third, if the decoding time is most important, the best choices are three variations of our proposed scheme: $\langle \mathbf{4} \rangle, \langle \mathbf{6} \rangle$, and $\langle \mathbf{10} \rangle$. Fourth, if the time needed to generate a measurement matrix entry is most important, the best choices are $\langle 1 \rangle, \langle 3 \rangle, \langle \mathbf{4} \rangle, \langle 7 \rangle, \langle 9 \rangle$ and $\langle \mathbf{10} \rangle$. Finally, if the space needed to generate a measurement matrix entry is most important, the best choices are $\langle 1 \rangle, \langle 2 \rangle, \langle \mathbf{3} \rangle, \langle \mathbf{4} \rangle, \langle 7 \rangle, \langle 9 \rangle$ and $\langle \mathbf{10} \rangle$.

For real-time applications, because "defective items" are usually considered to be *abnormal system activities* [18], they should be identified as quickly as possible. It is thus acceptable to use extra tests to speed up their identification. Moreover, the measurement matrix deployed in the system should not be stored in the system because of saving space. Therefore, the construction type should be nonrandom,

Table 3.1: Comparison with existing schemes.

No.	Scheme	Construction type	Number of tests t	Decoding time	Time to generate an entry	Space to generate an entry
$\langle 1 \rangle$	Indyk et al. [59] (Theorem 2.6)	Nonrandom	$O(d^2 \log^2 n)$	$O\left(\frac{d^9 (\log n)^{16+1/3}}{(\log(d \log n))^{7+1/3}}\right)$	$O(t)$	$O(t)$
$\langle 2 \rangle$	Indyk et al. [59] (Theorem 2.4)	Nonrandom	$O(d^2 \log n)$	$\mathsf{poly}(t) = O\left(d^{11} \log^{17} n\right)$	$\mathsf{poly}(t, n)$	$\mathsf{poly}(t)$
$\langle \mathbf{3} \rangle$	**Proposed (Theorem 3.5)**	Nonrandom	$O\left(\frac{d^2 \log^2 n}{(\log(d \log n) - \log\log(d \log n))^2}\right)$	$O\left(\frac{d^{3.57} \log^{6.26} n}{(\log(d \log n) - \log\log(d \log n))^{6.26}}\right)$ $+O\left(\frac{d^6 \log^4 n}{(\log(d \log n) - \log\log(d \log n))^4}\right)$	$O(t)$	$O(t)$
$\langle \mathbf{4} \rangle$	**Proposed (Corollary 3)**	Nonrandom	$O\left(\frac{d^2 \log^3 n}{(\log(d \log n) - \log\log(d \log n))^2}\right)$	$O(t)$	$O(t)$	$O(t)$
$\langle 5 \rangle$	Porat-Rothschild [58] (Theorem 2.3)	Nonrandom	$O(d^2 \log n)$	$O(tn) = O(d^2 \log n \times n)$	$O(tn)$	$O(tn)$
$\langle \mathbf{6} \rangle$	**Proposed (Corollary 2)**	Nonrandom	$O(d^2 \log^2 n)$	$O(t) = O(d^2 \log^2 n)$	$O(tn)$	$O(tn)$
$\langle 7 \rangle$	Indyk et al. [59] (Theorem 2.6)	Nonrandom $d = 2$	$2 \log n (2 \log n - 1)$	$\frac{2^9 (\log n)^{16+1/3}}{(\log(2 \log n))^{7+1/3}}$	$\log^2 n$	$\log n$
$\langle \mathbf{8} \rangle$	**Proposed (Theorem 3.4)**	Nonrandom $d = 2$	$4 \log^2 n$	$4 \log^2 n$	4	$2 \log n$ $+ \log(2 \log n)$
$\langle 9 \rangle$	Indyk et al. [59] (Theorem 2.4)	Random	$O(d^2 \log n)$	$\mathsf{poly}(t) = O\left(d^{11} \log^{17} n\right)$	$O(t^2 \log n)$	$O(t \log n)$
$\langle \mathbf{10} \rangle$	**Proposed (Corollary 1)**	Random	$O(d^2 \log^2 n)$	$O(t) = O(d^2 \log^2 n)$	$O(t^2)$	$O(t \log n)$
$\langle \mathbf{11} \rangle$	**Proposed (Corollary 4)**	Random	$O(d \log n \cdot \log \frac{d}{\epsilon})$	$O(d \log n \cdot \log \frac{d}{\epsilon})$	$O(tn)$	$O(tn)$

and the time and space needed to generate an entry should be within $\mathsf{poly}(t)$. Thus, the best choice is $\langle \mathbf{4} \rangle$ and the second best choice is $\langle \mathbf{3} \rangle$.

3.3 Preliminaries

The frequently used notations are as follows:

- $\circ$: concatenation operator (to be defined later).
- $\mathcal{S}, \mathcal{B}$: $k \times n$ measurement matrices used to identify at most one defective item, where $k = 2 \log n$.
- $\mathcal{M} = (m_{ij})$: $t \times n$ d-disjunct matrix, where integer $t \geq 1$ is number of tests.
- $\mathcal{T} = (t_{ij})$: $\mathsf{T} \times n$ measurement matrix used to identify at most d defective items, where integer $\mathsf{T} \geq 1$ is number of tests.

3.3.1 List recoverable codes

There may be occasions in the physical world where a person might want to recover a similar codeword from a given codeword. For example, a person searching on a website such as Google might be searching using the word "intercept". However, mistyping results in the input word being "inrrercep". The website should suggest a *list* of similar words that are "close" to the input word such as "in**t**ercep**t**" and "in**t**erce**de**". This observation leads to the concept of *list-recoverable codes.* The basic idea of list-recoverable codes is that, given a list of subsets in which each subset contains at most ℓ symbols in a given alphabet Σ (a finite field), the decoder of the list-recoverable codes produces at most Γ codewords from the list. Formally, this can be defined as follows.

Definition 5 *[66, Definition 2.2] Given integers $1 \le \ell \le \Gamma$, a code $C \subseteq \Sigma^n$ is said to be (ℓ, Γ)-list-recoverable if for all sequences of subsets $S_1, S_2, \ldots, S_n$ with each $S_a \subset \Sigma$ satisfying $|S_a| \le \ell$, there are at most Γ codewords $\mathbf{c} = (c_1, \ldots, c_v) \in C$ with the property that $c_a \in S_a$ for $a \in \{1, 2, \ldots, v\}$. The value ℓ is referred to as the input list size.*

Note that for any $\ell' \le \ell$, an (ℓ, Γ)-list-recoverable code is also an (ℓ', Γ)-list-recoverable code. For example, if we set $\Sigma = \{a, b, \ldots, z\}, \ell = 2, v = 9$, and $\Gamma = 2$, we have the following input and output:

$$\begin{bmatrix} S_1 = \{e,g\} \\ S_2 = \{r,x\} \\ S_3 = \{o,q\} \\ S_4 = \{t,u\} \\ S_5 = \{e,i\} \\ S_6 = \{s\} \\ S_7 = \{i,q\} \\ S_8 = \{t,u\} \\ S_9 = \{e\} \end{bmatrix} \xRightarrow{\text{decode}} \mathbf{c} = \left\{ \begin{bmatrix} e \\ x \\ q \\ u \\ i \\ s \\ i \\ t \\ e \end{bmatrix}, \begin{bmatrix} g \\ r \\ o \\ t \\ e \\ s \\ q \\ u \\ e \end{bmatrix} \right\}.$$

Guruswami [66] (Section 4.4.1) showed that any $[v, r]_q$-RS code (defined in Section 2.4) is also an $\left(\left\lceil \frac{v}{r} \right\rceil - 1, O\left(\frac{v^4}{r^2}\right)\right)$-list-recoverable code. To efficiently decode RS code, Chowdhury et al. [67] proposed an efficient scheme, which they summarized in Table 1 of their paper with $\omega < 2.38$, as follows:

Theorem 3.1 *[67, Corollary 18] Let $1 \leq r \leq v \leq q$ be integers. Then, any $[v, r]_q$-RS code, which is also $\left(\lceil \frac{v}{r} \rceil - 1, O\left(\frac{v^4}{r^2}\right)\right)$-list-recoverable code, can be decoded in time $O(v^{3.57} r^{2.69})$.*

A codeword of the $[v, r]_q$-RS code can be computed in time $O(r^2 \log\log r) \approx O(r^2)$ and space $O(r \log q / \log^2 r)$ [68].

3.3.2 Concatenated codes

Concatenated codes C are constructed by using an $(v_1, k_1)_q$ *outer code* C_{out}, where $q = 2^{k_2}$ (in general, $q = p^{k_2}$ where p is a prime number), and an $(v_2, k_2)_2$ binary *inner code* C_{in}, denoted as $C = C_{\text{out}} \circ C_{\text{in}}$.

Given a message $\mathbf{m} \in \mathbb{F}_q^{k_1}$, let $C_{\text{out}}(\mathbf{m}) = (x_1, \ldots, x_{v_1}) \in \mathbb{F}_q^{v_1}$. Then $C_{\text{out}} \circ C_{\text{in}}(\mathbf{m}) = (C_{\text{in}}(x_1), C_{\text{in}}(x_2), \ldots, C_{\text{in}}(x_{v_1})) \in (\{0,1\}^{v_2})^{v_1}$. Note that C is an $(v_1 v_2, k_1 k_2)_2$ code.

Using a suitable outer code and a suitable inner code, d-disjunct matrices can be generated. For example, let C_{out} and C_{in} be $(3,1)_8$ and $(3,3)_2$ codes, where $|C_{\text{out}}| = 12$ and $|C_{\text{in}}| = 8$. The corresponding matrices are $\mathcal{H} = \mathcal{M}_{C_{\text{out}}}$ and $\mathcal{K} = \mathcal{M}_{C_{\text{in}}}$ as follows:

$$\mathcal{H} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 4 & 4 & 4 & 7 & 0 & 0 \\ 1 & 2 & 4 & 1 & 2 & 4 & 1 & 2 & 4 & 0 & 7 & 0 \\ 1 & 4 & 2 & 4 & 2 & 1 & 2 & 1 & 4 & 0 & 0 & 7 \end{bmatrix},$$

$$\mathcal{K} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix},$$

If we concatenate each element of $\mathcal{H}$ with its 3-bit binary representation such as matrix $\mathcal{K}$, we get a 2-disjunct matrix:

$$\mathcal{M} = \mathcal{H} \circ \mathcal{K} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

From this discussion, we can draw an important conclusion about decoding schemes using concatenation codes and list-recoverable codes.

Theorem 3.2 (Simplified version of Theorem 4.1 [59]) *Let $d, \Gamma \geq 1$ be integers. Let C_{out} be an $(v_1, k_1)_{2^{k_2}}$ code that can be (d, Γ)-list recovered in time $\alpha_1(v_1, d, \Gamma, k_1, k_2)$. Let C_{in} be $(v_2, k_2)_2$ codes such that $\mathcal{M}_{C_{\text{in}}}$ is a d-disjunct matrix that can be decoded in time $\alpha_2(v_2, d, k_2)$. Suppose that matrix $\mathcal{M} = \mathcal{M}_{C_{\text{out}} \circ C_{\text{in}}}$ is d-disjunct. Note that $\mathcal{M}$ is a $t \times n$ matrix where $t = v_1 v_2$ and $n = 2^{k_1 k_2}$. Further, suppose that any arbitrary position in any codeword in C_{out} and C_{in} can be computed in space $p_1(v_1, d, \Gamma, k_1, k_2)$ and $p_2(v_2, d, k_2)$, respectively. Then:*

1. *given any outcome produced by at most d positives, the positive positions can be recovered in time $v_1 \alpha_2(v_2, d, k_2) + \alpha_1(v_1, d, \Gamma, k_1, k_2) + 2\Gamma t = v_1 \alpha_2(v_2, d, k_2) + \alpha_1(v_1, d, \Gamma, k_1, k_2) + O(\Gamma t)$; and*

2. *any entry in $\mathcal{M}$ can be computed in $\log t + \log n + p_1(v_1, d, \Gamma, k_1, k_2) + p_2(v_2, d, k_2) = O(\log t + \log n) + O\left(\max\{p_1(v_1, d, \Gamma, k_1, k_2), p_2(v_2, d, k_2)\}\right)$ space.*

Since the decoding scheme requires knowledge from several fields that are beyond the scope of this work, we do not discuss it here. Readers are encouraged to refer to [59] for further reading.

3.3.3 Review of Bui et al.'s scheme

A scheme proposed by Bui et al. [69] plays an important role for constructions in later sections. It is used to identify at most one defective item while never producing a false positive. The technical details are as follows.

Encoding procedure: Lee et al. [63] proposed a $k \times n$ measurement matrix $\mathcal{S}$ that uses $\log n$-bit representation of an integer, to detect at most one defective item:

$$\mathcal{S} := \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \ldots \mathbf{b}_n \\ \overline{\mathbf{b}}_1 & \overline{\mathbf{b}}_2 \ldots \overline{\mathbf{b}}_n \end{bmatrix} = \begin{bmatrix} \mathcal{S}_1 \ldots \mathcal{S}_n \end{bmatrix}, \tag{3.1}$$

where $k = 2\log n$, $\mathbf{b}_j$ is the $\log n$-bit binary representation of integer $j-1$, $\overline{\mathbf{b}}_j$ is $\mathbf{b}_j$'s complement, and $\mathcal{S}_j := \begin{bmatrix} \mathbf{b}_j \\ \overline{\mathbf{b}}_j \end{bmatrix}$ for $j = 1, 2, \ldots, n$. The weight of every column in $\mathcal{S}$ is $k/2 = \log n$.

Given an input vector $\mathbf{g} = (g_1, \ldots, g_n) \in \{0,1\}^n$, measurement matrix $\mathcal{S}$ is generalized:

$$\mathcal{B} := \mathcal{S} \times \mathrm{diag}(\mathbf{g}) = \begin{bmatrix} g_1\mathcal{S}_1 & \ldots & g_n\mathcal{S}_n \end{bmatrix}, \tag{3.2}$$

where $\mathrm{diag}(\mathbf{g}) = \mathrm{diag}(g_1, \ldots, g_n)$ is the diagonal matrix constructed by input vector $\mathbf{g}$, and $\mathcal{B}_j = g_j\mathcal{S}_j$ for $j = 1, \ldots, n$. It is obvious that $\mathcal{B} = \mathcal{S}$ when $\mathbf{g}$ is a vector of all ones; i.e., $\mathbf{g} = \mathbf{1} = (1, 1, \ldots, 1) \in \{1\}^n$. Moreover, the column weight of $\mathcal{B}$ is either $k/2 = \log n$ or 0.

For example, consider the case $n = 8$, $k = 2\log n = 6$, and $\mathbf{g} = (1, 0, 1, 0, 1, 1, 1, 1)$. Measurement matrices S and $\mathcal{B}$ are

$$\mathcal{S} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}, \tag{3.3}$$

$$\begin{aligned}\mathcal{B} &= \mathcal{S} \times \text{diag}(\mathbf{g}) = \mathcal{S} \times \text{diag}(1, 0, 1, 0, 1, 1, 1, 1)\\ &= [1 \times \mathcal{S}_1 \; 0 \times \mathcal{S}_2 \; 1 \times \mathcal{S}_3 \; 0 \times \mathcal{S}_4 \; 1 \times \mathcal{S}_5 \; 1 \times \mathcal{S}_6 \; 1 \times \mathcal{S}_7 \; 1 \times \mathcal{S}_8]\\ &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}. \end{aligned} \tag{3.4}$$

Then, given a representation vector of n items $\mathbf{x} = (x_1, \ldots, x_n)^T \in \{0, 1\}^n$, the outcome vector is

$$\mathbf{y}' = \mathcal{B} \odot \mathbf{x} = \bigvee_{j=1}^{n} x_j \mathcal{B}_j \tag{3.5}$$

$$= \bigvee_{j=1}^{n} x_j g_j \mathcal{S}_j = \bigvee_{\substack{j=1 \\ x_j g_j = 1}}^{n} \mathcal{S}_j. \tag{3.6}$$

Note that, even if there is only one entry $x_{j_0} = 1$ in $\mathbf{x}$, index j_0 cannot be recovered if $g_{j_0} = 0$.

Decoding procedure: From (3.6), the outcome $\mathbf{y}'$ is the union of at most $|\mathbf{x}|$ columns in $\mathcal{S}$. Because the weight of each column in $\mathcal{S}$ is $\log n$, if the weight of $\mathbf{y}'$ is $\log n$, the index of one non-zero entry in $\mathbf{x}$ is recovered by checking the first half of $\mathbf{y}'$. On the other hand, if $\mathbf{y}'$ is the union of at least two columns in $\mathcal{S}$ or zero vector, the weight of $\mathbf{y}'$ is not equal to $\log n$. This case is considered here as a defective item is not identified. Therefore, given a $k \times 1$ input vector, we can either identify one defective item or no defective item in time $k = 2\log n = O(\log n)$. Moreover, the decoding procedure does not produce a false positive.

For example, given $\mathbf{x}_1 = (0, 1, 0, 0, 0, 0, 0, 0)^T, \mathbf{x}_2 = (0, 1, 1, 0, 0, 0, 0, 0)^T$, and $\mathbf{x}_3 = (0, 1, 1, 1, 0, 0, 0, 0)^T$, their corresponding outcomes using the measurement matrix $\mathcal{B}$ in (3.4) are $\mathbf{y}'_1 = (0, 0, 0, 0, 0, 0)^T, \mathbf{y}'_2 = (0, 1, 0, 1, 0, 1)^T$, and $\mathbf{y}'_3 = (0, 1, 0, 1, 0, 1)^T$. Since $|\mathbf{y}'_1| = 0$, there is no defective item identified. Since

$|\mathbf{y}_2'| = |\mathbf{y}_3'| = 3 = \log n$, the only defective item identified from the first half of $\mathbf{y}_2'$ or $\mathbf{y}_3'$, i.e., $(0,1,0)$ is 3. Note that, even if $|\mathbf{x}_1| \neq |\mathbf{x}_2|$, the same defective item is identified.

3.4 Efficient decoding scheme using a given measurement matrix

In this section, we present a simple but powerful tool for identifying defective items using a given measurement matrix. We thereby answer the question of whether there exists a scheme such that a larger $\mathsf{T} \times n$ measurement matrix built from a given $t \times n$ measurement matrix, can be used to identify up to d defective items in time $\mathsf{poly}(t) = t \times \log n = \mathsf{T}$. It can be summarized as follows:

Theorem 3.3 *For any $\epsilon \geq 0$, suppose each set of d columns in a given $t \times n$ matrix $\mathcal{M}$ contains a $d \times d$ identity matrix with probability at least $1 - \epsilon$. Then there exists a $\mathsf{T} \times n$ matrix $\mathcal{T}$ constructed from $\mathcal{M}$ that can be used to identify at most d defective items in time $\mathsf{T} = t \times 2\log n$ with probability at least $1 - \epsilon$. Further, suppose that any entry of $\mathcal{M}$ can be computed in time β and space γ, so every entry of $\mathcal{T}$ can be computed in time $O(\beta \log n)$ and space $O(\log \mathsf{T} + \log n) + O(\gamma \log n)$.*

Proof. Suppose $\mathcal{M} = (m_{ij}) \in \{0,1\}^{t \times n}$. Then the $\mathsf{T} \times n$ measurement matrix $\mathcal{T}$ is generated by using the tensor product of $\mathcal{M}$ and $\mathcal{S}$ in (3.1):

$$\mathcal{T} = \mathcal{M} \circledcirc \mathcal{S} \ = \begin{bmatrix} \mathcal{S} \times \mathrm{diag}(\mathcal{M}_{1,*}) \\ \vdots \\ \mathcal{S} \times \mathrm{diag}(\mathcal{M}_{t,*}) \end{bmatrix} = \begin{bmatrix} \mathcal{B}^1 \\ \vdots \\ \mathcal{B}^t \end{bmatrix} = \begin{bmatrix} m_{11}\mathcal{S}_1 & \ldots & m_{1n}\mathcal{S}_n \\ \vdots & \ddots & \vdots \\ m_{t1}\mathcal{S}_1 & \ldots & m_{tn}\mathcal{S}_n \end{bmatrix}, \quad (3.7)$$

where $\mathsf{T} = t \times k = t \times 2\log n$ and $\mathcal{B}^i = \mathcal{S} \times \mathrm{diag}(\mathcal{M}_{i,*})$ for $i = 1, \ldots, t$. Note that $\mathcal{B}^i$ is an instantiation of $\mathcal{B}$ when $\mathbf{g}$ is set to $\mathcal{M}_{i,*}$ in (3.2). Then, for any $n \times 1$ representation vector $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, the outcome vector is

$$\mathbf{y}^\star = \mathcal{T} \odot \mathbf{x} = \begin{bmatrix} \mathcal{B}^1 \odot \mathbf{x} \\ \vdots \\ \mathcal{B}^t \odot \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1' \\ \vdots \\ \mathbf{y}_t' \end{bmatrix}, \quad (3.8)$$

where $\mathbf{y}'_i = \mathcal{B}^i \odot \mathbf{x}$ for $i = 1, \ldots, t$; $\mathbf{y}'_i$ is obtained by replacing $\mathcal{B}$ by $\mathcal{B}_i$ in (3.5).

By using the decoding procedure in section 3.3.3, the decoding procedure is simply to scan all $\mathbf{y}'_i$ for $i = 1, \ldots, t$. If $|\mathbf{y}'_i| = \log n$, we take the first half of $\mathbf{y}'_i$ to calculate the defective item. Thus, the decoding complexity is $\mathsf{T} = t \times 2 \log n = O(\mathsf{T})$.

Our task now is to prove that the decoding procedure above can identify all defective items with probability at least $1 - \epsilon$. Let $D = \{j_1, \ldots, j_{|D|}\}$ be the defective set, where $|D| = g \leq d$. We will prove that there exists $\mathbf{y}'_{i_1}, \ldots, \mathbf{y}'_{i_g}$ such that j_a can be recovered from $\mathbf{y}'_{i_a}$ for $a = 1, \ldots, g$. Because any set of d columns in $\mathcal{M}$ contains a $d \times d$ identity matrix with probability at least $1 - \epsilon$, any set of $g \leq d$ columns $j_1, \ldots, j_g$ in $\mathcal{M}$ also contains a $g \times g$ identity matrix with probability at least $1 - \epsilon$. Let $i_1, \ldots, i_g$ be the row indexes of $\mathcal{M}$ such that $m_{i_a j_a} = 1$ and $m_{i_a j_b} = 0$, where $a, b \in \{1, 2, \ldots, g\}$ and $a \neq b$. Then the probability that rows $i_1, \ldots, i_g$ coexist is at least $1 - \epsilon$.

For any outcome $\mathbf{y}'_{i_a}$, where $a = 1, \ldots, g$, by using (3.6), we have

$$\mathbf{y}'_{i_a} = \mathcal{B}^{i_a} \odot \mathbf{x} = \bigvee_{\substack{j=1 \\ x_j m_{i_a j} = 1}}^{n} \mathcal{S}_j = \bigvee_{\substack{j \in D \\ x_j m_{i_a j} = 1}} \mathcal{S}_j = \mathcal{S}_{j_a}, \tag{3.9}$$

because there are only g non-zero entries $x_{j_1}, \ldots, x_{j_g}$ in $\mathbf{x}$. Thus, all defective items $j_1, \ldots, j_g$ can be identified by checking the first half of each corresponding $\mathbf{y}'_{i_1}, \ldots, \mathbf{y}'_{i_g}$. Since the probability that rows $i_1, \ldots, i_g$ coexist is at least $1 - \epsilon$, the probability that defective items $j_1, \ldots, j_g$ are identified is also at least $1 - \epsilon$.

We next estimate the computational complexity of computing an entry in $\mathcal{T}$. An entry in row $1 \leq i \leq \mathsf{T}$ and column $1 \leq j \leq n$ needs $\log \mathsf{T} + \log n$ bits (space) to be indexed. It belongs to vector $m_{i_0 j}\mathcal{S}_j$, where $i_0 = i/(2\log n)$ if $i \mod (2 \log n) \equiv 0$ and $i_0 = \lfloor i/(2 \log n) \rfloor$ if $i \mod (2\log n) \not\equiv 0$. Since each entry in $\mathcal{M}$ needs γ space to compute, every entry in $\mathcal{T}$ can be computed in space $O(\log \mathsf{T} + \log n) + O(\gamma \log n)$ after mapping it to the corresponding column of $\mathcal{S}$. The time to generate an entry for $\mathcal{T}$ is straightforwardly obtained as $\beta \log n = O(\beta \log n)$. $\square$

To have a better understanding, the decoding procedure in Theorem 3.3 is summarized as in Algorithm 3.1.

Algorithm 3.1 GetDefectives($\mathbf{y}, n$): detection of up to d defective items.

Input: number of items n; outcome vector $\mathbf{y}$
Output: defective items

```
1:  $s = 2 \log n$.
2:  $S = \emptyset$.
3:  Divide $\mathbf{y}$ into $m = t/s$ smaller vectors $\mathbf{y}_1, \ldots, \mathbf{y}_m$ such that $\mathbf{y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_m^T)^T$
    and their size are equal to $s$, where $t$ is the number of entries in $\mathbf{y}$.
4:  for $i = 1$ to $m$ do
5:      if $\mathrm{wt}(\mathbf{y}_i) = \log n$ then
6:          Get defective item $d_0$ by checking first half of $\mathbf{y}$.
7:          $S = S \cup \{d_0\}$.
8:      end if
9:  end for
10: return $S$.
```

Part of Theorem 3.3 is implicit in other papers (e.g., [51], [52], [62], [63]). However, the authors of those papers only considered cases specific to their problems. They mainly focused on how to generate matrix $\mathcal{M}$ by using complicated techniques and a non-constructive method, i.e., random construction (e.g., [62], [63]). As a result, their decoding schemes are randomized. Moreover, they did not consider the cost of computing an entry in $\mathcal{M}$. In two of the papers [51, 52], the decoding time was not scaled to $t \times \log n$ for deterministic decoding, i.e., $\epsilon = 0$. Our contribution is to *generalize* their ideas into the framework of non-adaptive group testing. We next instantiate Theorem 3.3 in the broad range of measurement matrix construction.

3.4.1 Case of $\epsilon = 0$

We consider the case in which $\epsilon = 0$; i.e., a given matrix $\mathcal{M}$ is always $(d-1)$-disjunct. There are three metrics for evaluating an instantiation: number of tests, construction type, and time to generate an entry for $\mathcal{T}$. We first present an instantiation of a strongly explicit construction. Let $\mathcal{M}$ be a measurement matrix generated from Theorem 2.4. Then $t = O(d^2 \log n)$, $\beta = O(t^2 \log n)$, and $\gamma = O(t \log n)$. Thus, we obtain efficient NAGT where the number of tests and the decoding time are $O(d^2 \log^2 n)$.

Corollary 1 *Let $1 \leq d \leq n$ be integers. There exists a random $\mathsf{T} \times n$ measurement matrix $\mathcal{T}$ with $\mathsf{T} = O(d^2 \log^2 n)$ such that at most d defective items can be identified in time $O(\mathsf{T})$. Moreover, each entry in $\mathcal{T}$ can be computed in time $O(\mathsf{T}^2)$ and space $O(\mathsf{T} \log n)$.*

It is also possible to construct $\mathcal{T}$ deterministically. However, it would take $\mathsf{poly}(t, n)$ time and $\mathsf{poly}(t)$ space, which are too long and too much for practical applications. Therefore, we should increase the time needed to generate an entry for $\mathcal{T}$ in order to achieve nonrandom construction with the same number of tests $\mathsf{T} = O(d^2 \log^2 n)$ and a short construction time. The following theorem is based on the weakly explicit construction of a given measurement matrix as in Theorem 2.3; i.e., $t = O(d^2 \log n)$, $\beta = O(tn)$, and $\gamma = O(tn)$.

Corollary 2 *Let $1 \leq d \leq n$ be integers. There exists a nonrandom $\mathsf{T} \times n$ measurement matrix $\mathcal{T}$ with $\mathsf{T} = O(d^2 \log^2 n)$ that can be used to identify at most d defective items in time $O(\mathsf{T})$. Moreover, each entry in $\mathcal{T}$ can be computed in time (and space) $O(\mathsf{T}n)$.*

Although the number of tests is low and the construction type is nonrandom, the time to generate an entry for $\mathcal{T}$ is long. If we increase the number of tests, one can achieve both nonrandom construction and low generating time for an entry as follows:

Corollary 3 *Let $1 \leq d \leq n$ be integers. There exists a nonrandom $\mathsf{T} \times n$ measurement matrix $\mathcal{T}$ with $\mathsf{T} = O\left(\frac{d^2 \log^3 n}{(\log(d \log n) - \log\log(d \log n))^2}\right)$ that can be used to identify at most d defective items in time $O(\mathsf{T})$. Moreover, each entry in $\mathcal{T}$ can be computed in time (and space) $O(\mathsf{T})$.*

The above corollary is obtained by choosing a measurement matrix as a d-disjunct matrix in Theorem 3.5 (Section 3.5): $t = O\left(\frac{d^2 \log^2 n}{(\log(d \log n) - \log\log(d \log n))^2}\right)$, $\beta = O(t)$, and $\gamma = O(t)$.

3.4.2 Case of $\epsilon > 0$

To reduce the number of tests and the decoding complexity, the construction process of the given measurement matrix must be randomized. We construct the matrix as follows. A given $t \times n$ matrix $\mathcal{M} = (m_{ij})$ is generated randomly, where $\Pr(m_{ij} = 1) = \frac{1}{d}$ and $\Pr(m_{ij} = 0) = 1 - \frac{1}{d}$ for $i = 1, \ldots, t$ and $j = 1, \ldots, n$. The value of t is set to $\mathrm{e}d \ln \frac{d}{\epsilon}$. Then, for each set of d columns in $\mathcal{M}$, the probability that a set does not contain a $d \times d$ identity matrix is at most

$$\binom{d}{1} \left(1 - \frac{1}{d}\left(1 - \frac{1}{d}\right)^{d-1}\right)^t \leq d \cdot \exp\left(-\frac{1}{d-1}\left(1 - \frac{1}{d}\right)^d t\right) \tag{3.10}$$

$$\leq d \cdot \exp\left(-\frac{t}{d-1} \cdot \mathrm{e}^{-1}\left(1 - \frac{1}{d}\right)\right) \tag{3.11}$$

$$\leq d \cdot \exp\left(-\frac{t}{\mathrm{e}d}\right) = d \cdot \exp\left(-\ln\frac{d}{\epsilon}\right) \tag{3.12}$$

$$\leq \epsilon. \tag{3.13}$$

Expression (3.10) is obtained because $(1 + x)^y \leq \exp(xy)$ for all $|x| \leq 1$ and $y \geq 1$. Expression (3.11) is obtained because $\left(1 + \frac{x}{n}\right)^n \geq \mathrm{e}^x \left(1 - \frac{x^2}{n}\right)$ for $n > 1$ and $|x| < v$. Therefore, there exists a $t \times n$ matrix $\mathcal{M}$ with $t = O\left(d \log \frac{d}{\epsilon}\right)$ such that each set of d columns contains a $d \times d$ identity matrix with probability at least $1 - \epsilon$, for any $\epsilon > 0$. Since $\beta = \gamma = O(tn)$, W can derive the following corollary.

Corollary 4 *Given integers $1 \leq d \leq n$ and a scalar $\epsilon > 0$, there exists a random $\mathsf{T} \times n$ measurement matrix $\mathcal{T}$ with $\mathsf{T} = O\left(d \log n \cdot \log \frac{d}{\epsilon}\right)$ that can be used to identify at most d defective items in time $O(\mathsf{T})$ with probability at least $1 - \epsilon$. Furthermore, each entry in $\mathcal{T}$ can be computed in time (and space) $O(\mathsf{T}n)$.*

While the result in Corollary 4 is similar to previously reported ones [62], [63], construction of matrix $\mathcal{M}$ is much simpler. It is possible to achieve the number of tests $t = O\left(d \log \frac{d}{\epsilon} \cdot \log n\right)$ when *each* set of d columns in $\mathcal{M}$ contains a $d \times d$ identity matrix with probability at least $1 - \epsilon$ for any $\epsilon > 0$. However, it is impossible to achieve this number for *every* set of d columns that contains a $d \times d$ identity matrix with probability at least $1 - \epsilon$. In this case, by using the same procedure used for generating random matrix $\mathcal{M}$ and by resolving $\binom{n}{d}\binom{d}{1}\left(1 - \frac{1}{d}\left(1 - \frac{1}{d}\right)^{d-1}\right)^t \leq \epsilon$, the number of tests needed is determined to be $t = O\left(d^2 \log n + d \log \frac{1}{\epsilon}\right)$. Since this number is greater than that when $\epsilon = 0$ ($O(d^2 \log n)$), it is not beneficial to consider the case that every set of d columns that contains a $d \times d$ identity matrix with probability at least $1 - \epsilon$.

3.5 Nonrandom disjunct matrices

It is extremely important to have nonrandom constructions for measurement matrices in real-time applications. Therefore, we now focus on nonrandom constructions. We have shown that the well-known barrier on the number of tests $O(d^2 \log^2 n)$ for constructing a d-disjunct matrix can be overcome.

3.5.1 Case of $d = 2$

When $d = 2$, the measurement matrix is $\mathcal{T} = \mathcal{S} \circledcirc \mathcal{S}$, where $\mathcal{S}$ is given by (3.1). Note that the size of $\mathcal{S}$ is $k \times n$, where $k = 2\log n$, and $\mathcal{T}$ is not a 2-disjunct matrix. We start by proving that any two columns in $\mathcal{S}$ contain a 2×2 identity matrix. Indeed, suppose $\mathbf{b}_w = (b_{1w}, \ldots, b_{(k/2)w})^T$, which is a $\log n$-bit binary representation of $0 \leq w - 1 \leq n - 1$. For any two vectors $\mathbf{b}_{w_1}$ and $\mathbf{b}_{w_2}$, there exists a position i_0 such that $b_{i_0 w_1} = 0$ and $b_{i_0 w_2} = 1$, or $b_{i_0 w_1} = 1$ and $b_{i_0 w_2} = 0$ for any $1 \leq w_1 \neq w_2 \leq n$. Then their corresponding complementary vectors $\overline{\mathbf{b}}_{w_1} = (\overline{b}_{1w_1}, \ldots, \overline{b}_{(k/2)w_1})^T$ and $\overline{\mathbf{b}}_{w_2} = (\overline{b}_{1w_2}, \ldots, \overline{b}_{(k/2)w_2})^T$ satisfy: $\overline{b}_{i_0 w_1} = 0$ and $\overline{b}_{i_0 w_2} = 1$ when $b_{i_0 w_1} = 0$ and $b_{i_0 w_2} = 1$, or $\overline{b}_{i_0 w_1} = 1$ and $\overline{b}_{i_0 w_2} = 0$ when $b_{i_0 w_1} = 1$ and $b_{i_0 w_2} = 0$. Thus, any two columns w_1 and w_2 in $\mathcal{S}$ always contain a 2×2 identity matrix. From Theorem 3.3 (set $\mathcal{M} = \mathcal{S}$), we obtain the following theorem.

Theorem 3.4 *Let $2 \leq n$ be an integer. A $4\log^2 n \times n$ nonrandom measurement matrix $\mathcal{T}$ can be used to identify at most two defective items in time $4\log^2 n$. Moreover, each entry in $\mathcal{T}$ can be computed in space $2\log n + \log(2\log n)$ with four operations.*

Proof. It takes $\gamma = 2\log n + \log(2\log n)$ bits to index an entry in row i and column j. Only two shift operations and a mod operation are needed to exactly locate the position of the entry in column $\mathcal{S}_j$. Therefore, at most four operations ($\beta = 4$) and $2\log n + \log(2\log n)$ bits are needed to locate an entry in matrix $\mathcal{T}$. The decoding time is straightforwardly obtained from Theorem 3.3 ($t = k = 2\log n$). $\square$

3.5.2 General case

Indyk et al. [59] used Theorem 3.2 and Parvaresh-Vardy (PV) codes [70] to come up with Theorem 2.6. Since they wanted to convert RS code into list-recoverable code, they instantiated PV code into RS code. However, because PV code is powerful in terms of solving general problems, its decoding complexity is high. Therefore, the decoding complexity in Theorem 2.6 is relatively high. Here, by converting RS code into list-recoverable code using Theorem 3.1, we carefully use Theorem 3.2 to construct and decode disjunct matrices. As a result, the number of tests and the decoding time for a nonrandom disjunct matrix are significantly reduced.

Let $W(x)$ be a Lambert W function in which $W(x)\mathrm{e}^{W(x)} = x$ for any $x \geq -\frac{1}{\mathrm{e}}$. When $x > 0$, $W(x)$ is an increasing function. One useful bound [71] for a Lambert W function is $\ln x - \ln\ln x \leq W(x) \leq \ln x - \frac{1}{2}\ln\ln x$ for any $x \geq \mathrm{e}$. Theorem 3.2 is used to achieve the following theorem with careful setting of C_{out} and C_{in}

Theorem 3.5 *Let $1 \leq d \leq n$ be integers. Then there exists a nonrandom d-disjunct matrix $\mathcal{M}$ with $t = O\left(\frac{d^2\ln^2 n}{(W(d\ln n))^2}\right) = O\left(\frac{d^2\log^2 n}{(\log(d\log n) - \log\log(d\log n))^2}\right)$. Each entry (column) in $\mathcal{M}$ can be computed in time (and space) $O(t)$ $(O(t^{3/2}))$. Moreover, $\mathcal{M}$ can be used to identify up to d' defective items, where $d' \geq \left\lfloor \frac{d}{2} \right\rfloor + 1$, in time*

$$O\left(\frac{d^{3.57}\log^{6.26} n}{(\log(d\log n) - \log\log(d\log n))^{6.26}}\right) + O\left(\frac{d^6\log^4 n}{(\log(d\log n) - \log\log(d\log n))^4}\right).$$

When d is the power of 2, $d' = d - 1$.

Proof. Construction: We use the classical method proposed by Kautz and Singleton [54] to construct a d-disjunct matrix. Let η be an integer satisfying $2^\eta < 2\mathrm{e}^{W(\frac{1}{2}d\ln n)} < 2^{\eta+1}$. Choose C_{out} as an $[v = q-1, r]_q$-RS code, where

$$q = \begin{cases} 2\mathrm{e}^{W(\frac{1}{2}d\ln n)} = \frac{d\ln n}{W\left(\frac{1}{2}d\ln n\right)} & \text{if } 2\mathrm{e}^{W(\frac{1}{2}d\ln n)} \text{ is the power of 2.} \\ 2^{\eta+1}, & \text{otherwise.} \end{cases} \tag{3.14}$$

Set $r = \left\lceil \frac{q-2}{d} \right\rceil$, and let C_{in} be a $q \times q$ identity matrix. The complexity of q is

$\Theta\left(e^{W(d\ln n)}\right) = \Theta\left(\frac{d\ln n}{W(d\ln n)}\right)$ in both cases because

$$2e^{W(\frac{1}{2}d\ln n)} = \frac{d\ln n}{W\left(\frac{1}{2}d\ln n\right)} \le q < 2\cdot 2e^{W(\frac{1}{2}d\ln n)} = \frac{2d\ln n}{W\left(\frac{1}{2}d\ln n\right)}.$$

Let $C = C_{\text{out}} \circ C_{\text{in}}$. We are going to prove that $\mathcal{M} = \mathcal{M}_C$ is d-disjunct for such q and r. It is well known [54] that if $d \le \frac{q-1-1}{r-1}$, $\mathcal{M}$ is d-disjunct with $t = q(q-1)$ tests. Indeed, we have

$$\frac{q-1-1}{r-1} = \frac{q-2}{\lceil\frac{q-2}{d}\rceil - 1} \ge \frac{q-2}{\frac{q-2}{d}+1-1} = d. \tag{3.15}$$

Since $q = O\left(\frac{d\ln n}{W(d\ln n)}\right)$, the number of tests in $\mathcal{M}$ is

$$\begin{aligned} t = q(q-1) = O\left(\frac{d^2\ln^2 n}{(W(d\ln n))^2}\right) &= O\left(\frac{d^2\ln^2 n}{(\ln(d\ln n) - \ln\ln(d\ln n))^2}\right) \\ &= O\left(\frac{d^2\log^2 n}{(\log(d\log n) - \log\log(d\log n))^2}\right), \end{aligned}$$

because $\ln x - \ln\ln x \le W(x) \le \ln x - \frac{1}{2}\ln\ln x$ for any $x \ge \mathrm{e}$. Since C_{out} is an $[v,r]_q$-RS code, each of its codewords can be computed [68] in time

$$O(r^2) = O\left(\left(\frac{\ln n}{\ln(d\ln n) - \ln\ln(d\ln n)}\right)^2\right) = O\left(\frac{t}{d^2}\right) = O(t),$$

and space

$$p_1 = O(r\log q/\log^2 r) = O(q\log q) = O(d\ln n) = O(t). \tag{3.16}$$

Our task is now to prove that the number of columns in $\mathcal{M}_C$, i.e., q^r, is at least n. The range of $\frac{d\ln n}{W\left(\frac{1}{2}d\ln n\right)} \le q < \frac{2d\ln n}{W\left(\frac{1}{2}d\ln n\right)}$ is:

$$d+2 < \frac{d\ln n}{\ln\left(\frac{1}{2}d\ln n\right) - \frac{1}{2}\ln\ln\left(\frac{1}{2}d\ln n\right)} \le q \le \frac{2d\ln n}{\ln\left(\frac{1}{2}d\ln n\right) - \ln\ln\left(\frac{1}{2}d\ln n\right)} < 2d\ln n \tag{3.17}$$

These inequalities were obtained because $\ln x - \ln\ln x \le W(x) \le \ln x - \frac{1}{2}\ln\ln x$

for any $x \geq \mathrm{e}$. Then we have:

$$\begin{aligned}
q^{(q-2)/d} &= \left(\frac{q^q}{q^2}\right)^{1/d} \geq \left(\frac{1}{q^2} \times \left(2\mathrm{e}^{W(\frac{1}{2}d\ln n)}\right)^q\right)^{1/d} \\
&\geq \left(\frac{2^q}{q^2} \times \left(\mathrm{e}^{W(\frac{1}{2}d\ln n)}\right)^q\right)^{1/d} \geq \left(\frac{2^q}{q^2} \times \left(\mathrm{e}^{W(\frac{1}{2}d\ln n)\times 2\mathrm{e}^{W(\frac{1}{2}d\ln n)}}\right)\right)^{1/d} \\
&\geq \left(\frac{2^q}{q^2} \times \mathrm{e}^{2\times\frac{1}{2}d\ln n}\right)^{1/d} \qquad (3.18) \\
&\geq n \times \left(\frac{2^q}{q^2}\right)^{1/d} > n. \qquad (3.19)
\end{aligned}$$

Equation (3.18) is achieved because $W(x)\mathrm{e}^{W(x)} = x$. Equation (3.19) is obtained because $\left(\frac{2^q}{q^2}\right)^{1/d} \geq 1$ for any $q \geq 5$. Since $\frac{q-2}{d} \leq r = \lceil \frac{q-2}{d} \rceil < \frac{q-2}{d} + 1$, the number of codewords in C_{out} is:

$$\begin{aligned}
n < q^{(q-2)/d} \leq q^r &< q^{(q-2)/d+1} = q \times q^{(q-2)/d} \qquad (3.20) \\
&< \frac{d\ln n}{W\left(\frac{1}{2}d\ln n\right)} \left(\frac{2^q}{q^2}\right)^{1/d} \times n. \qquad (3.21)
\end{aligned}$$

Equation (3.20) indicates that the number of columns in $\mathcal{M}_C$ is more than n. To obtain a $t \times n$ matrix, one simply removes $q^r - n$ columns from $\mathcal{M}_C$. The maximum number of columns that can be removed is $O(d\ln n \times n^2)$ because of (3.21).

Decoding: Consider the ratio $\frac{q-1}{r}$ implied by list size $d' = \lceil \frac{q-1}{r} \rceil - 1 = \left\lceil \frac{q-1}{\lceil (q-2)/d \rceil} \right\rceil - 1$ of $[q-1, r]_q$-RS code. Parameter d' is also the maximum number of defective items that $\mathcal{M}$ can be used to identify because of Theorem 3.2. We thus have

$$d' = \left\lceil \frac{q-1}{\lceil (q-2)/d \rceil} \right\rceil - 1 \geq d\left(1 - \frac{d-1}{q+d-2}\right) > \frac{d}{2},$$

because $q + d - 2 \geq 2d > 2(d-1)$. Since d' is an integer, $d' \geq \left\lfloor \frac{d}{2} \right\rfloor + 1$.

Next we prove that $d' = d - 1$ when d is the power of 2, e.g., $d = 2^x$ for some positive integer x. Since q is also the power of 2 as shown by (3.14), suppose that $q = 2^y$ for some positive integer y. Because $q > d$ in (3.17), $2^y > 2^x$. Then

$r = \lceil \frac{q-1}{d} \rceil = 2^{y-x}$. Therefore, $d' = \lceil \frac{q-1}{r} \rceil - 1 = 2^x - 1 = d - 1$.

The decoding complexity of our proposed scheme is analyzed here. We have:

- Code C_{out} is an $(d' = \left\lceil \frac{q-1}{\lceil (q-2)/d \rceil} \right\rceil - 1, \Gamma = O\left(\frac{n^4}{r^2}\right) = O(q^2 d^2))$-list recoverable code as in Theorem 3.1. It can be decoded in time

$$\alpha_1 = O(n^{3.57} r^{2.69}) = O\left(\frac{d^{3.57} \log^{6.26} n}{(\log(d \log n) - \log\log(d \log n))^{6.26}} \right).$$

  Moreover, any codeword in C_{out} can be computed in time $O(r^2) = O\left(\frac{t}{d^2}\right)$ and space $p_1 = O(t)$ as in (3.16).

- C_{in} is a $q \times q$ identity matrix. Then $\mathcal{M}_{C_{\text{in}}}$ is a q-disjunct matrix. Since $d' \leq d < q$, $\mathcal{M}_{C_{\text{in}}}$ is also a d'-disjunct matrix. It can be decoded in time $\alpha_2 = d'q$ and each codeword can be computed in space $p_2 = \log q$.

From Theorem 3.2, given any outcome produced by at most d' defective items, those items can be identified in time

$$\begin{aligned}
&v\alpha_2 + \alpha_1 + O(\Gamma t) \\
&= vd'q + O\left(\frac{d^{3.57} \log^{6.26} n}{(\log(d \log n) - \log\log(d \log n))^{6.26}} \right) + O\left(\frac{d^6 \log^4 n}{(\log(d \log n) - \log\log(d \log n))^4} \right) \\
&= O\left(\frac{d^{3.57} \log^{6.26} n}{(\log(d \log n) - \log\log(d \log n))^{6.26}} \right) + O\left(\frac{d^6 \log^4 n}{(\log(d \log n) - \log\log(d \log n))^4} \right).
\end{aligned}$$

Moreover, each entry (column) in $\mathcal{M}$ can be computed in time $O(t)$ $(O(tq) = O(t^{3/2}))$ and space $O(\log t + \log n) + O(\max\{p_1, p_2\}) = O(d \log n) = O(t)$ $(O(tq) = O(t^{3/2}))$. □

If we substitute d by $2^{\lfloor \log d \rfloor + 1}$ in the theorem above, the measurement matrix is $2^{\lfloor \log d \rfloor + 1}$-disjunct. Therefore, it can be used to identify at most $d' = 2^{\lfloor \log d \rfloor + 1} - 1 \geq d$ defective items. The number of tests and the decoding complexity in the theorem remain unchanged because $d < 2^{\lfloor \log d \rfloor + 1} \leq 2d$.

3.6 Evaluation

We evaluated variations of our proposed scheme by simulation using $d = 2, 2^3, 2^7, 2^{10}, 2^{12}$ and $n = 2^{20}, 2^{40}, 2^{60}, 2^{80}, 2^{100}$ in Matlab R2015a on an HP Compaq Pro 8300SF desktop PC with a 3.4-GHz Intel Core i7-3770 processor and 16-GB memory.

3.6.1 Numerical settings for n, d, and q

We focused on nonrandom construction of a $t \times n$ d-disjunct matrix $\mathcal{M}$ for which the time to generate an entry is $\mathsf{poly}(t)$. Given integers d and n, an $[v = q-1, r]_q$ code C_{out} and a $q \times q$ identity matrix C_{in} were set up to create $\mathcal{M} = \mathcal{M}_{C_{\text{out}} \circ C_{\text{in}}}$. The precise formulas for q, r, t are $q = 2\mathrm{e}^{W(\frac{1}{2}d \ln n)}$ or $q = 2^{\eta+1}$ as in (3.14), $r = \lceil \frac{q-2}{d} \rceil$, and $t = q(q-1)$. Note that the integer q is the power of 2. Moreover, $n' = q^r$ is the maximum number of items such that the resulting $t \times n'$ matrix generated from this RS code was still d-disjunct. Parameter $d' = \lceil \frac{q-1}{r} \rceil - 1 = \left\lceil \frac{q-1}{\lceil (q-2)/d \rceil} \right\rceil - 1$ is the maximum number of defective items that matrix $\mathcal{M}$ could be used to identify. The parameters $t_2 = 4800 d^2 \log n$ and $t_1 = d \log n (d \log n - 1)$ are the number of tests from Theorems 2.4 and 2.6. The numerical results are shown in Table 3.2.

Since the number of tests from Theorem 2.4 is $O(d^2 \log n)$, it *should* be smaller than the number of tests in Theorem 2.6, which is $t = O(d^2 \log^2 n)$, and Theorem 3.5, which is $t = O\left(\frac{d^2 \log^2 n}{(\log(d \log n) - \log\log(d \log n))^2} \right)$. However, the numerical results in Table 3.2 show the opposite. Even when $d = 2^{12} \approx 0.4\%$ of n, the number of tests from Theorem 2.4 was the largest. Moreover, there was no efficient construction scheme associated with it. The main reason is that the multiplicity of $O(d^2 \log n)$ is 4,800, which is quite large. Figure 3.1 shows the ratio between the number of tests from Theorem 2.4 and the number from Theorem 3.5 (our proposed scheme) and between the number from Theorem 2.6 and the number from Theorem 3.5 (our proposed scheme). The number of tests with our proposed scheme was clearly smaller than with the existing schemes, even when $n = 2^{100}$. This indicates that the matrices generated from Theorem 2.4 and Theorem 2.6 are *good in theoretical analysis* but *bad in practice.*

Table 3.2: Parameter settings for $[q-1, r]_q$-RS code and resulting $q(q-1) \times n$ d-disjunct matrix: number of items n, maximum number of defective items d, alphabet size q as in (3.14), number of tests $t = q(q-1)$, dimension $r = \lceil \frac{q-2}{d} \rceil$. Parameter $d' = \left\lceil \frac{q-1}{\lceil (q-2)/d \rceil} \right\rceil - 1$ is the maximum number of defective items that the $t \times n$ resulting matrix can be used to identify. Parameter $n' = q^r$ is maximum number of items such that resulting $q(q-1) \times n'$ matrix generated from this RS code is still d-disjunct. Parameters $t_2 = 4800d^2 \log n$ and $t_1 = d \log n (d \log n - 1)$ are number of tests from Theorems 2.4 and 2.6.

d	n	q	$t = q(q-1)$	r	d'	n'	$t_1 = d \log n (d \log n - 1)$	$t_2 = 4800d^2 \log n$
	2^{20}	$2^6 = 64$	$4,032$	8	$d-1$	2^{48}	$25,440$	$6,144,000$
	2^{40}	$2^7 = 128$	$16,256$	16	$d-1$	2^{102}	$102,080$	$12,288,000$
$2^3 = 8$	2^{60}	$2^7 = 128$	$16,256$	16	$d-1$	2^{102}	$229,920$	$18,432,000$
	2^{80}	$2^7 = 128$	$16,256$	16	$d-1$	2^{102}	$408,960$	$24,576,000$
	2^{100}	$2^8 = 256$	$65,280$	32	$d-1$	2^{256}	$639,200$	$30,720,000$
	2^{20}	$2^9 = 512$	$261,632$	4	$d-1$	2^{36}	$6,551,040$	$1,572,864,000$
	2^{40}	$2^{10} = 1,024$	$1,047,552$	8	$d-1$	2^{80}	$26,209,280$	$3,145,728,000$
$2^7 = 128$	2^{60}	$2^{10} = 1,024$	$1,047,552$	8	$d-1$	2^{80}	$58,974,720$	$4,718,592,000$
	2^{80}	$2^{11} = 2,048$	$4,192,256$	16	$d-1$	2^{176}	$104,847,360$	$6,291,456,000$
	2^{100}	$2^{11} = 2,048$	$4,192,256$	16	$d-1$	2^{176}	$163,827,200$	$7,864,320,000$
	2^{20}	$2^{11} = 2,048$	$4,192,256$	2	$d-1$	2^{22}	$419,409,920$	$100,663,296,000$
	2^{40}	$2^{12} = 4,096$	$16,773,120$	4	$d-1$	2^{48}	$1,677,680,640$	$201,326,592,000$
$2^{10} = 1,024$	2^{60}	$2^{13} = 8,192$	$67,100,672$	8	$d-1$	2^{104}	$3,774,812,160$	$301,989,888,000$
	2^{80}	$2^{13} = 8,192$	$67,100,672$	8	$d-1$	2^{104}	$6,710,804,480$	$402,653,184,000$
	2^{100}	$2^{14} = 16,384$	$268,419,072$	16	$d-1$	2^{224}	$10,485,657,600$	$503,316,480,000$
	2^{20}	$2^{13} = 8,192$	$67,100,672$	2	$d-1$	2^{26}	$6,710,804,480$	$1,610,612,736,000$
	2^{40}	$2^{14} = 16,384$	$268,419,072$	4	$d-1$	2^{56}	$26,843,381,760$	$3,221,225,472,000$
$2^{12} = 4,096$	2^{60}	$2^{15} = 32,768$	$1,072,398,336$	8	$d-1$	2^{120}	$60,397,731,840$	$4,831,838,208,000$
	2^{80}	$2^{15} = 32,768$	$1,072,398,336$	8	$d-1$	2^{120}	$107,373,854,720$	$6,442,450,944,000$
	2^{100}	$2^{15} = 32,768$	$1,072,398,336$	8	$d-1$	2^{120}	$167,771,750,400$	$8,053,063,680,000$

In contrast, a nonrandom d-disjunct matrix is easily generated from Theorem 3.5. It also can be used to identify at most $d-1$ defective items. If we want to identify up to d defective items, we must generate a nonrandom $(d+1)$-disjunct matrix in which the number of tests is still smaller than t_1 and t_2. Since the number of tests from Theorem 3.5 is the lowest, its decoding time is the shortest. In short, for implementation, we recommend using the nonrandom construction in Theorem 3.5.

3.6.2 Experimental results

Since the time to generate a measurement matrix entry would be too long if it were $O(tn)$, we focus on implementing the methods for which the time to generate a measurement matrix entry is $\mathsf{poly}(t)$, i.e., $\langle \mathbf{3} \rangle, \langle \mathbf{4} \rangle, \langle \mathbf{8} \rangle, \langle 9 \rangle, \langle \mathbf{10} \rangle$ in

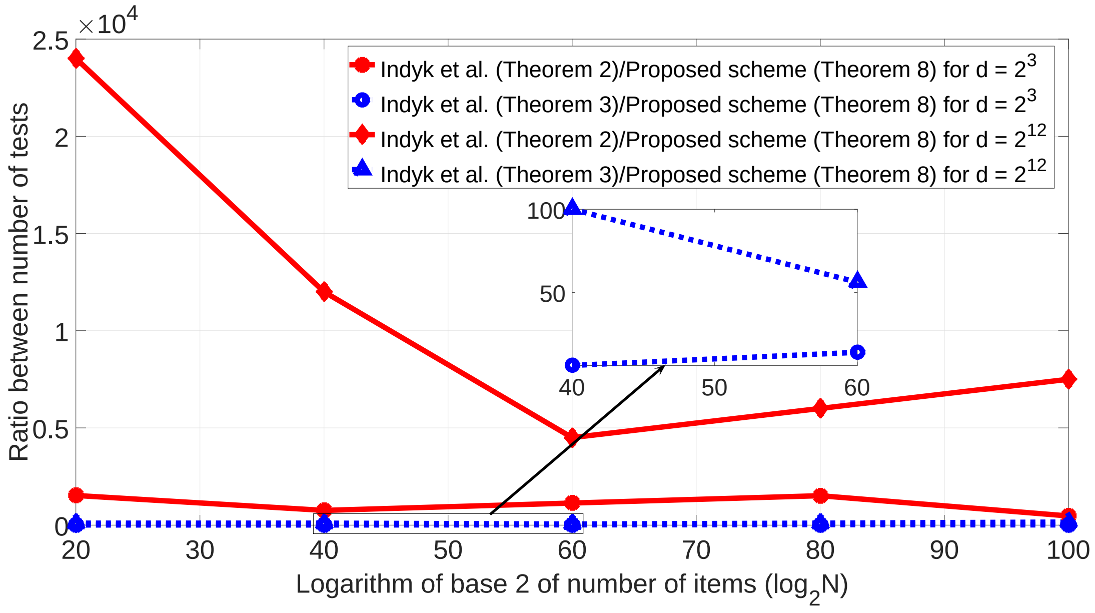


Figure 3.1: Ratio of number of tests from Theorem 2.4 and number from Theorem 2.6 to number with proposed scheme (Theorem 3.5) for $d = 2^3, 2^{12}$ and $n = 2^{20}, 2^{40}, 2^{60}, 2^{80}, 2^{100}$. Ratio was always larger than 1; i.e., the number of tests in the proposed scheme is smaller than the compared one.

Table 3.1. However, to incorporate a measurement matrix into applications, random constructions are not preferable. Therefore, we focus on nonrandom constructions. Since we are unable to program decoding of list-recoverable codes because it requires knowledge of algebra, finite field, linear algebra, and probability. We therefore tested our proposed scheme by implementing $\langle \mathbf{4} \rangle$ (Theorem 3.4) and $\langle \mathbf{8} \rangle$ (Corollary 3). This is reasonable because, as analyzed in section 3.6.1, the number of tests in Theorem 3.5 is the best for implementing nonrandom constructions. Since Corollary 3 is derived from Theorem 3.5, its decoding time should be the best for implementation.

We ran experiments for $d = 2$ from Theorem 3.4 and $d = 2^3, 2^7$ from Corollary 3. We did not run any for $d = 2^{10}, 2^{12}$ because there was not enough memory in our set up (more than 100 GB of RAM is needed). The decoding time was calculated in seconds and averaged over 100 runs. When $d = 2$, the decoding time was less than 1ms. As shown in Figure 3.2, the decoding time was linearly related to the number of tests, which confirms our theoretical analysis. Moreover, defective items

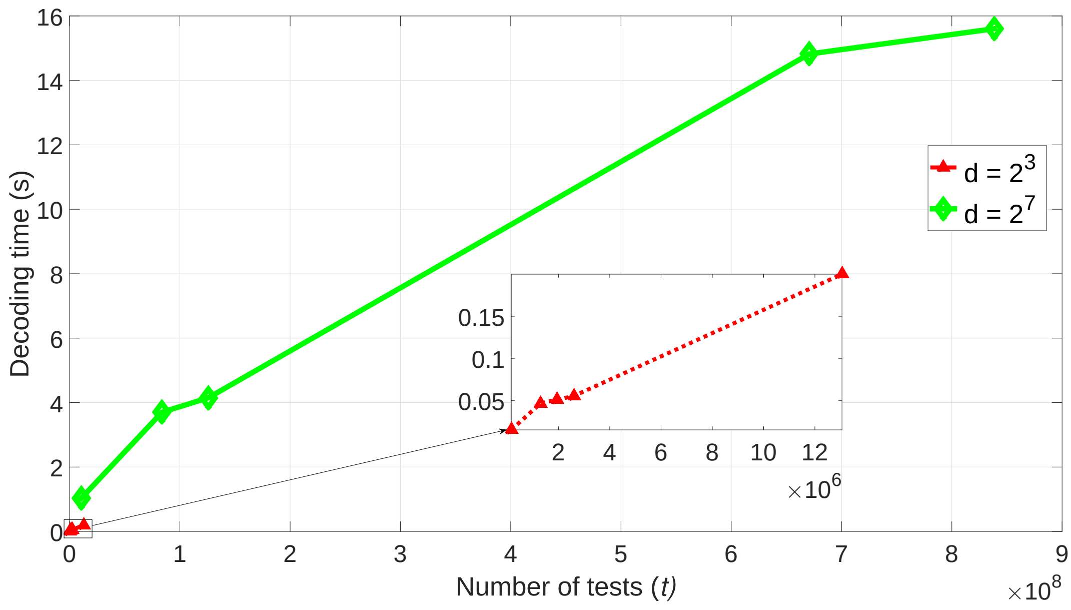


Figure 3.2: Decoding time for $d = 2^3$ and $d = 2^7$ from Theorem 3.4. Number of items n was $\{2^{20}, 2^{40}, 2^{60}, 2^{80}$, or $2^{100}\}$.

were identified extremely quickly (less than 16s) even when $n = 2^{100}$. The accuracy was always 1; i.e., all defective items were identified.

3.7 Conclusion

We have presented a scheme that enables a larger measurement matrix built from a given $t \times n$ measurement matrix to be decoded in time $O(t \log n)$ and a construction of a nonrandom d-disjunct matrix with $t = O\left(\frac{d^2 \log^2 n}{(\log(d \log n) - \log\log(d \log n))^2}\right)$ tests. This number of tests indicates that the upper bound for nonrandom construction is no longer $O(d^2 \log^2 n)$. Although the number of tests with our proposed schemes is not optimal in term of theoretical analysis, it is good enough for implementation. In particular, the decoding time is less than 16 seconds even when $d = 2^7 = 128$ and $n = 2^{100}$. Moreover, in nonrandom constructions, there is no need to store a measurement matrix because each column in the matrix can be generated efficiently.

Open problem: Our finding that n becomes much smaller than n' as q increases (Table 3.2) is quite interesting. Our hypothesis is that the number of tests needed may be smaller than $2\mathrm{e}^{W(\frac{1}{2}d\ln n)}\left(2\mathrm{e}^{W(\frac{1}{2}d\ln n)}-1\right)$. If this is indeed true, it paves the way toward achieving a very efficient construction and a shorter decoding time without using randomness. An interesting question is to answer the question that whether there exists a $t \times n$ d-disjunct matrix with $t \leq 2\mathrm{e}^{W(\frac{1}{2}d\ln n)}\left(2\mathrm{e}^{W(\frac{1}{2}d\ln n)}-1\right)$ that can be constructed in time $O(tn)$ with each entry generated in time (and space) $\mathsf{poly}(t)$ and with a decoding time of $O(t^2)$.

“Unity makes strength.”
– Anonymous.

4
Non-adaptive threshold group testing

4.1 Introduction

Consider a set of n items with up to d defective items. In Chapter 3, we have studied the classical non-adaptive group testing (NACGT) in which the outcome of a test on a subset of items is positive if the subset contains at least one defective item, and is negative otherwise. Damaschke [9] introduced a generalization of classical group testing known as *threshold group testing* (TGT). In this variation, the outcome of a test on a subset of items is positive if the subset contains at least u defective items, where u is a parameter, is negative if the subset contains no more than ℓ defective items, where $0 \leq \ell < u$, and is arbitrary otherwise. When $u = 1$ and $\ell = 0$, threshold group testing reduces to classical group testing. We note that ℓ is always smaller than the number of defective items. Otherwise, every test would yield a negative outcome and no information can be extracted from the test outcomes. Most of the previous work in this area, such as [9, 36–39], dealt with $g = u - \ell - 1 \geq 0$. When $g = 0$, i.e., $\ell = u - 1$, TGT is called TGT with threshold u and denoted as u-TGT. The focus here is on non-adaptive threshold group testing (NATGT) with threshold u, i.e., u-TGT with non-adaptive design and all our comparisons consider this regime.

In threshold group testing, Damaschke [9] showed that the set of positive items can be identified with $\binom{n}{u}$ tests with up to g false positives and g false negatives, where $g = u - \ell - 1$ is the *gap* parameter. Cheraghchi [36] showed that it is possible to find the defective items with $O(d^{g+2} \log d \cdot \log \frac{n}{d} \cdot 8^u u^u)$ tests, and that this trade-off is essentially optimal when u is constant. Recently, De Marco et al. [37] improved this bound to $O\left(d^{3/2} \log \frac{n}{d}\right)$ tests under the extra assumption that the number of defective items is exactly d, which is rather restrictive in application. Although the number of tests has been extensively studied, there have been few reports that focus on the decoding algorithm as well. Chen and Fu [39] proposed schemes based on NACGT for when $g = 0$ that can find the defective items using $O\left(\left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} d \log \frac{n}{d}\right)$ tests in time $O(n^u \log n)$. Chan et al. [38] presented a randomized algorithm with $O\left(\log \frac{1}{\epsilon} \cdot d\sqrt{u} \log n\right)$ tests to find the defective items in time $O(n \log n + n \log \frac{1}{\epsilon})$ given that the number of defective items is exactly d, $g = 0$, and $u = o(d)$. The cost of these decoding schemes increases with n. Our objective is to find an efficient decoding scheme to identify up to d defective items in NATGT when $g = 0$.

4.2 Contributions

We consider the case where $g = 0$, i.e., $\ell = u - 1$ ($u \geq 2$), and call this model u-NATGT. We first propose an efficient scheme for identifying up to d defective items in NATGT in time $t \times \mathsf{poly}(d^2 \log n)$, where t is the number of tests. Our main idea is to create at least a specified number of rows in the test matrix such that the corresponding test in each row contains exactly u defective items and such that the defective items in the rows are the defective items to be identified. We "map" these rows using a special matrix constructed from a disjunct matrix (defined later) and its complementary matrix, thereby converting the outcome in NATGT to the outcome in NACGT. The defective items in each row can then be efficiently identified.

Although Cheraghchi [36], De Marco et al. [37], and D'yachkov et al. [72] proposed nearly optimal bounds on the number of tests, there are no decoding algorithms associated with their schemes. Note that the number of tests is optimal in [72] and [36], i.e., $O(d^2 \log n)$, when the threshold u is a fixed constant. On the other hand, the scheme of Chen et al. [39] requires a smaller number of tests compared with our scheme. However, the decoding complexity of their scheme is exponential in the number of items n, which is impractical. Chan et al. [38] proposed a probabilistic approach to achieve a small number of tests, which combinatorially can be better than our scheme. However, their scheme is only applicable when the number of defective items is exactly d, the threshold u is much smaller than d ($u = o(d)$), and the decoding complexity remains high, namely $O(n \log n + n \log \frac{1}{\epsilon})$, where $\epsilon > 0$ is the precision parameter.

We present a test scheme that can be instantiated via either deterministic or randomized decoding. The deterministic decoding scheme identifies all defectives (in the worst case). On the other hand, randomized decoding reduces the number of tests; all defective items can be found with probability at least $1 - \epsilon$ for any $\epsilon > 0$. The decoding time is $t \times \mathsf{poly}(d^2 \log n)$. A comparison with existing work is given in Table 4.1.

Table 4.1: Comparison with existing work.

Scheme	Number of defective items	Number of tests t	Decoding complexity	Decoding type
Cheraghchi [36]	$\leq d$	$O(d^2 \log d \cdot \log \frac{n}{d} \cdot 8^u u^u)$	$\times$	$\times$
De Marco et al. [37]	d	$O\left(d^2 \cdot \sqrt{\frac{d-u}{du}} \cdot \log \frac{n}{d}\right)$	$\times$	$\times$
D'yachkov et al. [72]	$\leq d$	$O\left(d^2 \log n \cdot \frac{(u-1)!4^u}{(u-2)^u (\ln 2)^u}\right)$	$\times$	$\times$
Chen et al. [39]	$\leq d$	$O\left(\left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} d \log \frac{n}{d}\right)$	$O(n^u \log n)$	Deterministic
Deterministic decoding	$\leq d$	$O\left(\left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} d^3 \log n \cdot \log \frac{n}{d}\right)$	$t \times \mathsf{poly}(d^2 \log n)$	Deterministic
Chan et al. [38]	d	$O\left(\log \frac{1}{\epsilon} \cdot d\sqrt{u} \log n\right)$	$O(n \log n + n \log \frac{1}{\epsilon})$	Random
Randomized decoding	$\leq d$	$O\left(\left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} \left(u \log \frac{d}{u} + \log \frac{1}{\epsilon}\right) \cdot d^2 \log n\right)$	$t \times \mathsf{poly}(d^2 \log n)$	Random

4.3 Preliminaries

Here are some of the notations used:

1. $\mathcal{T}$: $t \times n$ measurement matrix used to identify up to d defective items in u-NATGT, where integer $t \geq 1$ is the number of tests.
2. $\mathcal{G} = (g_{ij})$: $h \times n$ matrix, where $h \geq 1$.
3. $\mathcal{M} = (m_{ij})$: $k \times n$ d-disjunct matrix used to identify up to u defective items in u-NATGT and d defective items in NACGT, where integer $k \geq 1$ is the number of tests.
4. $\overline{\mathcal{M}} = (\overline{m}_{ij})$: the $k \times n$ complementary matrix of $\mathcal{M}$; $\overline{m}_{ij} = 1 - m_{ij}$.
5. $\mathbf{x}_i = (x_{i1}, \ldots, x_{in})^T, D_i$: binary representation of items and set of indices of defective items in row $\mathcal{G}_{i,*}$. For example, if $n = 6$, the defective items are 1, 2, 3, and $\mathcal{G}_{1,*} = (1, 0, 1, 0, 1, 1)$, then $\mathbf{x}_1 = (1, 0, 1, 0, 0, 0)$ and $D_1 = \{1, 3\}$.

4.3.1 Problem definition

We index the population of n items from 1 to n. Let $[n] = \{1, 2, \ldots, n\}$ and D be the defective set, where $|D| \leq d$. A test is defined by a subset of items $P \subseteq [n]$. (d, u, n)-NATGT is a problem in which there are up to d defective items among n items. A test consisting of a subset of n items is positive if there are at least u

defective items in the test, and each test is designed in advance. Formally, the test outcome is positive if $|P \cap D| \geq u$ and negative if $|P \cap D| < u$.

We can model (d, u, n)-NATGT as follows: A $t \times n$ binary matrix $\mathcal{T} = (t_{ij})$ is defined as a measurement matrix, where n is the number of items and t is the number of tests. A vector $\mathbf{x} = (x_1, \ldots, x_n)^T$ is the binary representation vector of n items, where $|\mathbf{x}| \leq d$. An entry $x_j = 1$ indicates that item j is defective, and $x_j = 0$ indicates otherwise. The jth item corresponds to the jth column of the matrix. An entry $t_{ij} = 1$ naturally means that item j belongs to test i, and $t_{ij} = 0$ means otherwise. The outcome of all tests is $\mathbf{y} = (y_1, \ldots, y_t)^T$, where $y_i = 1$ if test i is positive and $y_i = 0$ otherwise. The procedure to get the outcome vector $\mathbf{y}$ is called the *encoding procedure.* The procedure used to identify defective items from $\mathbf{y}$ is called the *decoding procedure.* Outcome vector $\mathbf{y}$ is

$$\mathbf{y} = \mathcal{T} \otimes \mathbf{x} = \begin{bmatrix} \mathcal{T}_{1,*} \otimes \mathbf{x} \\ \vdots \\ \mathcal{T}_{t,*} \otimes \mathbf{x} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_t \end{bmatrix} \tag{4.1}$$

where $\otimes$ is a notation for the test operation in u-NATGT; namely, $y_i = \mathcal{T}_{i,*} \otimes \mathbf{x} = 1$ if $\sum_{j=1}^{n} t_{ij} x_j \geq u$, and $y_i = \mathcal{T}_{i,*} \otimes \mathbf{x} = 0$ if $\sum_{j=1}^{n} t_{ij} x_j < u$ for $i = 1, \ldots, t$. Our objective is to find an efficient decoding scheme to identify up to d defective items in (d, u, n)-NATGT.

4.3.2 Completely separating matrix

We now introduce the notion of completely separating matrices which are used to get efficient decoding algorithms for (d, u, n)-NATGT. A (u, w)-completely separating matrix is defined as follows:

Definition 6 *Let u, w, h, and n be positive integers. An $h \times n$ matrix $\mathcal{G} = (g_{ij})_{1 \leq i \leq h, 1 \leq j \leq n}$ is called a (u, w)-completely separating matrix if for any pair of subsets $I, J \subset [n]$ such that $|I| = u$, $|J| = w$, and $I \cap J = \emptyset$, there exists row l such that $g_{lr} = 1$ for any $r \in I$ and $g_{ls} = 0$ for any $s \in J$. Row l is called a singular row to subsets I and J. When $u = 1$, the matrix $\mathcal{G}$ is called a w-disjunct matrix.*

This definition is slightly different from the one described by D'yachkov et al. [56]. It is easy to verify that, if a matrix is a (u, w)-completely separating matrix, it is also a (u, v)-completely separating matrix for any $v \leq w$. Below we present the existence of such matrices.

Theorem 4.1 *Given integers* $1 \leq u + w < n$ *and* $uw > 0$, *there exists a* (u, w)-*completely separating matrix of size* $h \times n$, *where*

$$h = \left\lceil \frac{(u+w)^{u+w}}{u^u w^w} \left((u+w) \log \frac{\mathrm{e}n}{u+w} + u \log \frac{\mathrm{e}(u+w)}{u} \right) \right\rceil + 1$$

and e *is base of the natural logarithm.*

Proof. Consider a randomly generated $h \times n$ matrix $\mathcal{G} = (g_{ij})_{1 \leq i \leq h, 1 \leq j \leq n}$ in which each entry g_{ij} is assigned to 1 with probability p and to 0 with probability $1 - p$. For any pair of subsets $I, J \subset [n]$ such that $|I| = u$, $|J| = w$, the probability that a row is not singular is

$$1 - p^u (1-p)^w. \tag{4.2}$$

Subsequently, the probability that there is no singular row to subsets I and J is

$$f(p) = \left(1 - p^u (1-p)^w\right)^h. \tag{4.3}$$

Using a union bound, the probability that any pair of subsets $I, J \subset [n]$, where $|I| = u$, $|J| = w$, does not have a singular row; i.e., the probability that $\mathcal{G}$ is not a (u, w)-separating matrix, is

$$g(p, h, u, w, n) = \binom{n}{u+w} \binom{u+w}{u} f(p) = \binom{n}{u+w} \binom{u+w}{u} \left(1 - p^u (1-p)^w\right)^h.$$

To ensure that there exists a (u, w)-separating matrix $\mathcal{G}$, one needs to find p and h such that $g(p, h, u, w, n) < 1$. Choosing $p = \frac{u}{u+w}$, we have:

$$\begin{aligned} f(p) = \left(1 - p^u (1-p)^w\right)^h &= \left(1 - \left(\frac{u}{u+w}\right)^u \left(1 - \frac{u}{u+w}\right)^w\right)^h \\ &\leq \exp\left(-h \cdot \frac{u^u w^w}{(u+w)^{u+w}}\right), \end{aligned} \tag{4.4}$$

where (4.4) holds because $1 - x \leq \mathrm{e}^{-x}$ for any $x > 0$. Thus we have

$$g(p,h,u,w,n) = \binom{n}{u+w}\binom{u+w}{u} f(p) \leq \left(\frac{\mathrm{e}n}{u+w}\right)^{u+w} \left(\frac{\mathrm{e}(u+w)}{u}\right)^{u} f(p) \tag{4.5}$$

$$\leq \left(\frac{\mathrm{e}n}{u+w}\right)^{u+w} \left(\frac{\mathrm{e}(u+w)}{u}\right)^{u} \exp\left(-h \cdot \frac{u^u w^w}{(u+w)^{u+w}}\right) \tag{4.6}$$

$$< 1$$

$$\Longleftrightarrow \left(\frac{\mathrm{e}n}{u+w}\right)^{u+w} \left(\frac{\mathrm{e}(u+w)}{u}\right)^{u} < \exp\left(h \cdot \frac{u^u w^w}{(u+w)^{u+w}}\right)$$

$$\Longleftrightarrow h > \frac{(u+w)^{u+w}}{u^u w^w}\left((u+w)\log\frac{\mathrm{e}n}{u+w} + u\log\frac{\mathrm{e}(u+w)}{u}\right) \tag{4.7}$$

In the above, we have (4.5) because $\binom{a}{b} \leq \left(\frac{\mathrm{e}a}{b}\right)^b$ and (4.6) by using (4.4). From (4.7), if we choose

$$\begin{aligned} h &= \left\lceil \frac{(u+w)^{u+w}}{u^u w^w}\left((u+w)\log\frac{\mathrm{e}n}{u+w} + u\log\frac{\mathrm{e}(u+w)}{u}\right)\right\rceil + 1 \\ &= O\left(\mathrm{e}^{u+w}(u+w)\log\frac{n}{u+w}\right), \text{ because } u+w < n, \end{aligned}$$

then $g(p,h,u,w,n) < 1$; i.e., there exists a (u,w)-completely separating matrix of size $h \times n$. □

Suppose that $\mathcal{G}$ is an $h \times n$ (u,w)-completely separating matrix. If w is set to $d-u$, then **every** $h \times d$ submatrix, which is constructed by any d columns of $\mathcal{G}$, is a $(u,d-u)$-completely separating matrix. This property is too strong and increases the number of rows in $\mathcal{G}$. To reduce the number of rows, we relax this property to a "for-each" guarantee as follows: **each** $h \times d$ submatrix, which is constructed by d columns of $\mathcal{G}$, is a $(u,d-u)$-completely separating matrix with high probability. The following corollary describes this idea in more detail.

Corollary 5 *Let u,d,n be any given positive integers such that $1 \leq u < d < n$. For any $\epsilon > 0$, there exists a random $h \times n$ matrix such that for each $h \times d$ submatrix, which is constructed by picking a set of d columns, is a $(u,d-u)$-completely* *separating matrix with probability at least* $1 - \epsilon$*, where*

$$h = \left\lceil \left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} \left(u \log \frac{\mathrm{e}d}{u} + \log \frac{1}{\epsilon}\right) \right\rceil$$

and e *is base of the natural logarithm.*

Proof. Consider a random $h \times n$ matrix $\mathcal{G} = (g_{ij})_{1 \leq i \leq h, 1 \leq j \leq n}$ in which each entry g_{ij} is assigned to 1 with probability of $\frac{u}{d}$ and to 0 with probability of $1 - \frac{u}{d}$. Our task is to prove that *each* $h \times d$ matrix $\mathcal{G}'$, constructed by d columns of $\mathcal{G}$, is a $(u, d-u)$-completely separating matrix with probability at least $1 - \epsilon$ for any $\epsilon > 0$. Specifically, we prove that $h = \left\lceil \left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} \left(u \log \frac{\mathrm{e}d}{u} + \log \frac{1}{\epsilon}\right) \right\rceil$ is sufficient to achieve such $\mathcal{G}'$. Similar to the proof in Theorem 4.1, the probability that $\mathcal{G}'$ is not a $(u, d-u)$-completely separating matrix up to ϵ is

$$\binom{d}{u} \left(1 - \left(\frac{u}{d}\right)^u \left(1 - \frac{u}{d}\right)^{d-u}\right)^h \leq \left(\frac{\mathrm{e}d}{u}\right)^u \exp\left(-h \left(\frac{u}{d}\right)^u \left(\frac{d-u}{d}\right)^{d-u}\right) \leq \epsilon \tag{4.8}$$

$$\iff \frac{1}{\epsilon}\left(\frac{\mathrm{e}d}{u}\right)^u \leq \exp\left(h \left(\frac{u}{d}\right)^u \left(\frac{d-u}{d}\right)^{d-u}\right)$$

$$\iff h \geq \left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} \left(u \log \frac{\mathrm{e}d}{u} + \log \frac{1}{\epsilon}\right)$$

We get (4.8) because $1 - x \leq e^{-x}$ for any $x > 0$ and $\binom{a}{b} \leq \left(\frac{\mathrm{e}a}{b}\right)^b$. This completes the proof. $\square$

4.4 Proposed scheme

The basic idea of our scheme, which uses a divide and conquer strategy, is to create at least κ rows of matrix $\mathcal{G}$, e.g., $i_1, i_2, \ldots, i_\kappa$ such that $|D_{i_1}| = \cdots = |D_{i_\kappa}| = u$ and $D_{i_1} \cup \ldots \cup D_{i_\kappa} = D$. Then we "map" these rows by using a special matrix that enables us to convert the outcome in NATGT to the outcome in NACGT. The defective items in each row can then be efficiently identified. We present a particular matrix that achieves efficient decoding for each row in the following

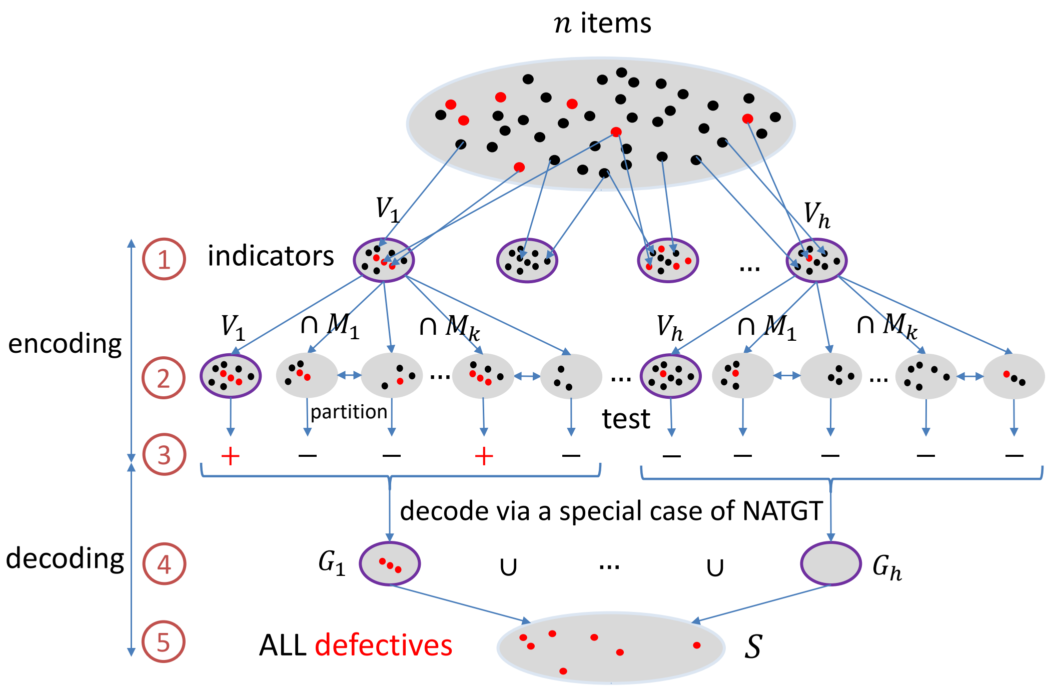


Figure 4.1: An illustration of the proposed scheme.

section. This idea is illustrated in Figure 4.1.

In Figure 4.1, the defective and negative items are represented as red and black dots, respectively. There are five steps in the proposed scheme. The encoding procedure includes Steps 1, 2, and 3. Steps 4 and 5 correspond to the decoding procedure. A row $\mathcal{G}_{i,*}$ can be represented by a support set $V_i = \{j \mid g_{ij} = 1\}$. Let $\mathcal{M} = (m_{i'j})$ be a $k \times n$ d-disjunct matrix and $M_{i'} = \{j \mid m_{i'j} = 1\}$ be the support set of $\mathcal{M}_j$ for $i' = 1, \ldots, k$ and $j = 1, \ldots, n$.

In the encoding procedure, we create h "indicating subsets" $V_1, \ldots, V_h$ in Step 1 in which $V_{i_l} \cap D_{i_l} = D_{i_l}$ for $l = 1, \ldots, \kappa$. Our objective is to extract D_{i_l} from V_{i_l} efficiently. In Step 2, each subset V_i is mapped to $2k+1$ subsets. They are the V_i and k dual subsets in which each dual subset ($V_i \cap M_{i'}$ and $V_i \setminus V_i \cap M_{i'}$ for $i' = 1, \ldots, k$) is a partition of a V_i created from $\mathcal{M}$. Step 3 simply gets the outcomes of all tests generated in Step 2.

In the decoding procedure, Step 4 gets the defective set G_i from the $2k+1$ subsets created from V_i as an instance of NATGT, for $i = 1, \ldots, h$. As a result, the cardinality of G_i is either u or 0. Finally, the defective set D is the union of

$G_1, \ldots, G_h$ in Step 5.

4.4.1 When the number of defective items equals the threshold

In this section, we consider a special case in which the number of defective items equals the threshold, i.e., $|\mathbf{x}| = u$. Given a measurement matrix $\mathcal{M}$ and a representation vector of u defective items $\mathbf{x}$ ($|\mathbf{x}| = u$), what we observe is $\mathbf{y} = \mathcal{M} \otimes \mathbf{x} = (y_1, \ldots, y_k)^T$. Our objective is to recover $\mathbf{y}' = \mathcal{M} \odot \mathbf{x} = (y_1', \ldots, y_k')^T$ from $\mathbf{y}$. Then $\mathbf{x}$ can be recovered if we choose $\mathcal{M}$ as a d-disjunct matrix described in Theorem 2.5. To achieve this goal, we create a measurement matrix:

$$\mathcal{A} = \begin{bmatrix} \mathcal{M} \\ \overline{\mathcal{M}} \end{bmatrix} \tag{4.9}$$

where $\mathcal{M} = (m_{ij})$ is a $k \times n$ d-disjunct matrix as described in Theorem 2.5 and $\overline{\mathcal{M}} = (\overline{m}_{ij})$ is the complement matrix of $\mathcal{M}$, $\overline{m}_{ij} = 1 - m_{ij}$ for $i = 1, \ldots, k$ and $j = 1, \ldots, n$. We note that $\mathcal{M}$ can be decoded in time $\mathsf{poly}(k) = \mathsf{poly}(d^2 \log n)$ because $k = O(d^2 \log n)$. Let us assume that the outcome vector is $\mathbf{z}$. Then we have:

$$\mathbf{z} = \mathcal{A} \otimes \mathbf{x} = \begin{bmatrix} \mathcal{M} \otimes \mathbf{x} \\ \overline{\mathcal{M}} \otimes \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \overline{\mathbf{y}} \end{bmatrix} \tag{4.10}$$

where $\mathbf{y} = \mathcal{M} \otimes \mathbf{x} = (y_1, \ldots, y_k)^T$ and $\overline{\mathbf{y}} = \overline{\mathcal{M}} \otimes \mathbf{x} = (\overline{y}_1, \ldots, \overline{y}_k)^T$. The following lemma shows that $\mathbf{y}' = \mathcal{M} \odot \mathbf{x}$ is always obtained from $\mathbf{z}$; i.e., vector $\mathbf{x}$ can always be recovered.

Lemma 4.2 *Given integers $2 \leq u \leq d < n$, there exists a strongly explicit $2k \times n$ matrix such that if there are exactly u defective items among n items in u-NATGT, the u defective items can be identified in time* $\mathsf{poly}(k)$*, where $k = O(d^2 \log n)$.*

Proof. We construct the measurement matrix $\mathcal{A}$ in (4.9) and assume that $\mathbf{z}$ is the observed vector as in (4.10). Our task is to create vector $\mathbf{y}' = \mathcal{M} \odot \mathbf{x}$ from $\mathbf{z}$. One can get it using the following rules, where $l = 1, 2, \ldots, k$:

1. If $y_l = 1$, then $y_l' = 1$.

2. If $y_l = 0$ and $\overline{y}_l = 1$, then $y'_l = 0$.

3. If $y_l = 0$ and $\overline{y}_l = 0$, then $y'_l = 1$.

We now prove the correctness of the above rules. Because $y_l = 1$, there are at least u defective items in row $\mathcal{M}_{l,*}$. Then, the first rule is implied.

If $y_l = 0$, there are less than u defective items in row $\mathcal{M}_{l,*}$. Because $|\mathbf{x}| = u$, we have $\overline{y}_l = 1$, and the threshold is u, there must be u defective items in row $\overline{\mathcal{M}}_{l,*}$. Moreover, since $\overline{\mathcal{M}}_{l,*}$ is the complement of $\mathcal{M}_{l,*}$, there must be no defective item in test l of $\mathcal{M}$. Therefore, we have $y'_l = 0$, and the second rule is implied.

If $y_l = 0$, there are less than u defective items in row $\mathcal{M}_{l,*}$. Similarly, if $\overline{y}_l = 0$, there are less than u defective items in row $\overline{\mathcal{M}}_{l,*}$. We now show that the number of defective items in row $\mathcal{M}_{l,*}$ or $\overline{\mathcal{M}}_{l,*}$ cannot be equal to zero. Indeed, if the number of defective items in row $\mathcal{M}_{l,*}$ (resp., $\overline{\mathcal{M}}_{l,*}$) equals zero, then $\overline{y}_l = 1$ (resp., $y_l = 1$) because $\overline{\mathcal{M}}_{l,*}$ is the complement of $\mathcal{M}_{l,*}$. This contradicts the assumption that $y_l = 0$ and $\overline{y}_l = 0$. Therefore, the number of defective items in row $\mathcal{M}_{l,*}$ is not equal to zero. Consequently, the test outcome corresponding to row $\mathcal{M}_{l,*}$ in NACGT is positive, i.e., $\mathbf{y}'_l = 1$. The third rule is thus implied.

Since we get $\mathbf{y}' = \mathcal{M} \odot \mathbf{x}$, the matrix $\mathcal{M}$ is a d-disjunct matrix and $u \leq d$, u defective items can be identified in time $\mathsf{poly}(k)$ by Theorem 2.5, 3.5, or Corollary 3. □

Example: We demonstrate Lemma 4.2 by setting $u = d = 2$, $k = 9$, and $n = 12$ and defining a 9×12 2-disjunct matrix $\mathcal{M}$ as follows:

$$\mathcal{M} = \begin{bmatrix} 0&0&0&0&0&0&1&1&1&1&0&0 \\ 0&0&0&1&1&1&0&0&0&1&0&0 \\ 1&1&1&0&0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&1&0&0&1&0&1&0 \\ 0&1&0&0&1&0&0&1&0&0&1&0 \\ 1&0&0&1&0&0&1&0&0&0&1&0 \\ 0&1&0&1&0&0&0&0&1&0&0&1 \\ 0&0&1&0&1&0&1&0&0&0&0&1 \\ 1&0&0&0&0&1&0&1&0&0&0&1 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 0\\0\\1\\0\\0\\0\\0\\0\\0 \end{bmatrix}, \overline{\mathbf{y}} = \begin{bmatrix} 1\\1\\0\\1\\0\\0\\0\\1\\0 \end{bmatrix}, \mathbf{y}' = \begin{bmatrix} 0\\0\\1\\0\\1\\1\\1\\0\\1 \end{bmatrix} \tag{4.11}$$

Assume that the defective items are 1 and 2, i.e., $\mathbf{x} = [1, 1, 0, 0, 0, 0, 0, 0, 0]^T$; then the observed vector is $\mathbf{z} = [\mathbf{y}^T\ \overline{\mathbf{y}}^T]^T$. Using the three rules in the proof of Lemma 4.2, we obtain vector $\mathbf{y}'$. We note that $\mathbf{y}' = \mathcal{M}_1 \bigvee \mathcal{M}_2 = \mathcal{M} \odot \mathbf{x}$. Using a decoding algorithm (which is omitted in this example), we can identify items 1 and 2 as defective items from $\mathbf{y}'$.

4.4.2 Encoding procedure

To implement the divide and conquer strategy, we need to divide the set of defective items into small subsets such that defective items in those subsets can be effectively identified. We suppose that there exists an $h \times n$ matrix $\mathcal{G}$ containing κ rows, denoted as $i_1, i_2, \ldots, i_\kappa$, with probability at least $1 - \epsilon$ such that (i) $|D_{i_1}| = \cdots = |D_{i_\kappa}| = u$ and (ii) $D_{i_1} \cup \ldots \cup D_{i_\kappa} = D$ for any $\epsilon \geq 0$, where D_i is the set of indices of defective items in row $\mathcal{G}_{i,*}$ (defined in Section 4.3). These conditions guarantee that all defective items are included in the decoded set.

After creating the matrix $\mathcal{G}$, we generate matrix $\mathcal{A}$ as in (4.9). Then the final measurement matrix $\mathcal{T}$ of size $(2k+1)h \times n$ is created as follows:

$$\mathcal{T} = \begin{bmatrix} \mathcal{G}_{1,*} \\ \mathcal{A} \times \mathsf{diag}(\mathcal{G}_{1,*}) \\ \vdots \\ \mathcal{G}_{h,*} \\ \mathcal{A} \times \mathsf{diag}(\mathcal{G}_{h,*}) \end{bmatrix} = \begin{bmatrix} \mathcal{G}_{1,*} \\ \mathcal{M} \times \mathsf{diag}(\mathcal{G}_{1,*}) \\ \overline{\mathcal{M}} \times \mathsf{diag}(\mathcal{G}_{1,*}) \\ \vdots \\ \mathcal{G}_{h,*} \\ \mathcal{M} \times \mathsf{diag}(\mathcal{G}_{h,*}) \\ \overline{\mathcal{M}} \times \mathsf{diag}(\mathcal{G}_{h,*}) \end{bmatrix} \tag{4.12}$$

The vector observed using u-NATGT after performing the tests given by the measurement matrix $\mathcal{T}$ is

$$\mathbf{y} = \mathcal{T} \otimes \mathbf{x} = \begin{bmatrix} \mathcal{G}_{1,*} \\ \mathcal{A} \times \mathsf{diag}(\mathcal{G}_{1,*}) \\ \vdots \\ \mathcal{G}_{h,*} \\ \mathcal{A} \times \mathsf{diag}(\mathcal{G}_{h,*}) \end{bmatrix} \otimes \mathbf{x} = \begin{bmatrix} \mathcal{G}_{1,*} \otimes \mathbf{x} \\ \mathcal{A} \otimes \mathbf{x}_1 \\ \vdots \\ \mathcal{G}_{h,*} \otimes \mathbf{x} \\ \mathcal{A} \otimes \mathbf{x}_h \end{bmatrix} = \begin{bmatrix} \mathcal{G}_{1,*} \otimes \mathbf{x} \\ \mathcal{M} \otimes \mathbf{x}_1 \\ \overline{\mathcal{M}} \otimes \mathbf{x}_1 \\ \vdots \\ \mathcal{G}_{h,*} \otimes \mathbf{x} \\ \mathcal{M} \otimes \mathbf{x}_h \\ \overline{\mathcal{M}} \otimes \mathbf{x}_h \end{bmatrix} = \begin{bmatrix} y_1 \\ \mathbf{y}_1 \\ \overline{\mathbf{y}}_1 \\ \vdots \\ y_h \\ \mathbf{y}_h \\ \overline{\mathbf{y}}_h \end{bmatrix} = \begin{bmatrix} y_1 \\ \mathbf{z}_1 \\ \vdots \\ y_h \\ \mathbf{z}_h \end{bmatrix} \tag{4.13}$$

where $\mathbf{x}_i = \mathsf{diag}(\mathcal{G}_{i,*}) \times \mathbf{x}$, $y_i = \mathcal{G}_{i,*} \otimes \mathbf{x}$, $\mathbf{y}_i = \mathcal{M} \otimes \mathbf{x}_i = (y_{i1}, \ldots, y_{ik})^T$, $\overline{\mathbf{y}}_i = \overline{\mathcal{M}} \otimes \mathbf{x}_i = (\overline{y}_{i1}, \ldots, \overline{y}_{ik})^T$, and $\mathbf{z}_i = [\mathbf{y}_i^T \ \overline{\mathbf{y}}_i^T]^T$ for $i = 1, 2, \ldots, h$.

We note that $\mathbf{x}_i$ is the vector representing the defective items corresponding to row $\mathcal{G}_{i,*}$. If $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$, then $D_i = \{l \mid x_{il} = 1, l \in [n]\}$. We thus have $|D_i| = |\mathbf{x}_i| \leq d$. Moreover, the condition $y_i = 1$ holds if and only if $|\mathbf{x}_i| \geq u$.

4.4.3 The decoding procedure

The decoding procedure is summarized as Algorithm 4.1, where $\mathbf{y}'_i = (y'_{i1}, \ldots, y'_{ik})^T$ is presumed to be $\mathcal{M} \odot \mathbf{x}_i$. The procedure is briefly explained as follows: Step 2 enumerates the h rows of $\mathcal{G}$. Step 3 checks if there are at least u defective items in row $\mathcal{G}_{i,*}$. Steps 4 to 14 calculate $\mathbf{y}'_i$, and Step 16 checks if all items in G_i are truly defective and adds them into D.

4.4.4 Correctness of the decoding procedure

Recall that our objective is to recover $\mathbf{x}_i$ from y_i and $\mathbf{z}_i = [\mathbf{y}_i^T \ \overline{\mathbf{y}}_i^T]$ for $i = 1, 2, \ldots, h$. Step 2 enumerates the h rows of $\mathcal{G}$. We have that y_i is the indicator for whether there are at least u defective items in row $\mathcal{G}_{i,*}$. If $y_i = 0$, it implies that there are less than u defective items in row $\mathcal{G}_{i,*}$. Since we only focus on rows $\mathcal{G}_{i,*}$, which have exactly u defective items, vector $\mathbf{z}_i$ is not considered if $y_i = 0$. This is achieved by Step 3.

When $y_i = 1$, there are at least u defective items in row $\mathcal{G}_{i,*}$. If there are

Algorithm 4.1 Decoding procedure for u-NATGT

Input: Outcome vector $\mathbf{y}$, $\mathcal{M}$.

Output: The set of defective items D.

```
1: $D = \emptyset$.
2: for $i = 1$ to $h$ do
3:     if $y_i = 1$ then
4:         for $l = 1$ to $k$ do
5:             if $y_{il} = 1$ then
6:                 $y'_{il} = 1$
7:             end if
8:             if $y_{il} = 0$ and $\overline{y}_{il} = 1$ then
9:                 $y'_{il} = 0$
10:            end if
11:            if $y_{il} = 0$ and $\overline{y}_{il} = 0$ then
12:                $y'_{il} = 1$
13:            end if
14:        end for
15:        Decode $\mathbf{y}'_i$ using $\mathcal{M}$ to get the defective set $G_i$.
16:        if $|G_i| = u$ and $\bigvee_{j \in G_i} \mathcal{M}_j \equiv \mathbf{y}'_i$ then
17:            $D = D \cup G_i$.
18:        end if
19:    end if
20: end for
21: Return $D$.
```

exactly u defective items in this row, they are always identified as described by Lemma 4.2. Our task is now to prevent false defectives by decoding $\mathbf{y}'_i$.

Steps 4 to 14 calculate $\mathbf{y}'_i$ from $\mathbf{z}_i$. If there are exactly u defective items in row $\mathcal{G}_{i,*}$, we have $\mathbf{y}'_i = \mathcal{M} \otimes \mathbf{x}_i$ and $|\mathbf{x}_i| = u$. If there are more than u defective items in row $\mathcal{G}_{i,*}$, vector $\mathbf{y}'_i = \mathcal{M} \otimes \mathbf{x}'_i$ for some vector $\mathbf{x}'_i \in \{0,1\}^n$ after implementing Steps 4 to 14. In the latter case, we may not recover $\mathbf{x}'_i$ correctly. Moreover, it is not clear whether the non-zero entries in $\mathbf{x}'_i$ are necessarily the indices of defective items. Therefore, our task is to decode $\mathbf{y}'_i$ using matrix $\mathcal{M}$ to get the defective set G_i, and then validate whether all items in G_i are defective.

There exists at least κ rows of $\mathcal{G}$ in which there are exactly u defective items, and we need to identify all defective items in these rows. Therefore, we only consider the case when the number of defective items obtained from decoding $\mathbf{y}'_i$ is equal to u; i.e., $|G_i| = u$. Our task is now to avoid false positives, which is accomplished by Step 16. There are two sets of defective items corresponding to $\mathbf{z}_i$: the first one is the true set, which is D_i and is *unknown*, and the second one is G_i, which is expected to be D_i (albeit not surely) and $|G_i| = u$. Note that $|D_i| \geq u$ because $y_i = 1$. If $G_i \equiv D_i$, we can always identify u defective items and the condition in Step 16 always holds because of Lemma 4.2. We need to consider the case $G_i \not\equiv D_i$; i.e., when there are more than u defective items in row $\mathcal{G}_{i,*}$. We break down this case into two categories:

1. When $|G_i \setminus D_i| = 0$: in this case, all elements in G_i are defective items. We need to consider whether $\bigvee_{j \in G_i} \mathcal{M}_j \equiv \mathbf{y}'_i$. If this condition holds, we obtain the true defective items. If it does not hold, we do not take G_i into the set of defective items.

2. When $|G_i \setminus D_i| \neq 0$: in this case, we prove that $\bigvee_{j \in G_i} \mathcal{M}_j \equiv \mathbf{y}'_i$ does not hold; i.e., none of the elements in G_i are added to the defective set. Consider any $j_1 \in G_i \setminus D_i$. Since $|D_i| \leq d$ and $\mathcal{M}$ is a d-disjunct matrix, there exists a row, denoted τ, such that $m_{\tau j_1} = 1$ and $m_{\tau x} = 0$ for $x \in D_i$. Therefore, there are fewer than u defective items in row τ; i.e., $y_{i\tau} = 0$. Because $u \leq |D_i|$, we have $\overline{y}_{i\tau} = 1$, which implies that $y'_{i\tau} = 0$. However, we have $\bigvee_{x \in G_i} m_{\tau x} = \left(\bigvee_{x \in G_i \setminus \{j_1\}} m_{\tau x} \right) \bigvee m_{\tau j_1} = \left(\bigvee_{x \in G_i \setminus \{j_1\}} m_{\tau x} \right) \bigvee 1 = 1 \neq 0 = y'_{i\tau}$. Therefore, the condition $\bigvee_{j \in G_i} \mathcal{M}_j \equiv \mathbf{y}'_i$ does not hold.

Thus, Step 16 eliminates false positives. Finally, Step 21 returns the defective item set D.

4.4.5 Example for Algorithm 4.1

In this section, we demonstrate Algorithm 4.1 by setting $n = 5, u = 2, d = 4, h = 5$, and $k = 9$. Assume that the defective items are 1, 2, 3, and 4. The two matrices, $\mathcal{G}$ and $\mathcal{M}$, are as follows:

$$\mathcal{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \mathcal{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that matrix $\mathcal{M}$ is a 4-disjunct matrix and that matrix $\mathcal{G}$ satisfies conditions (i) and (ii) for the defective set $D = \{1, 2, 3, 4\}$. There are 0, 1, 2, 2, and 4 defective items in rows 1, 2, 3, 4, and 5 in $\mathcal{G}$, respectively. After measurement matrix $\mathcal{T}$ is created as in (4.12), the test outcome observed using u-NATGT is $\mathbf{y} = [y_1\ \mathbf{y}_1^T\ \overline{\mathbf{y}}_1^T\ y_2\ \mathbf{y}_2^T\ \overline{\mathbf{y}}_2^T\ y_3\ \mathbf{y}_3^T\ \overline{\mathbf{y}}_3^T\ y_4\ \mathbf{y}_4^T\ \overline{\mathbf{y}}_4^T\ y_5\ \mathbf{y}_5^T\ \overline{\mathbf{y}}_5^T]^T$, where:

$$\begin{aligned}
y_1 = 0,\ \mathbf{y}_1^T &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \overline{\mathbf{y}}_1^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
y_2 = 0,\ \mathbf{y}_2^T &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \overline{\mathbf{y}}_2^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
y_3 = 1,\ \mathbf{y}_3^T &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \overline{\mathbf{y}}_3^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \\
(\mathbf{y}_3')^T &= \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \\
y_4 = 1,\ \mathbf{y}_4^T &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \overline{\mathbf{y}}_4^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \\
(\mathbf{y}_4')^T &= \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix},
\end{aligned}$$

$$y_5 = 1,\ \mathbf{y}_5^T = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \overline{\mathbf{y}}_5^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$
$$(\mathbf{y}_5')^T = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Algorithm 4.1 proceeds as follows. Because $y_1 = y_2 = 0$ (Step 3), there are less than $u = 2$ defective items in rows $\mathcal{G}_{1,*}$ and $\mathcal{G}_{2,*}$. Therefore, vectors $\mathbf{y}_1'$ and $\mathbf{y}_2'$ are not computed. Since $y_3 = 1$ (Step 3), there are at least $u = 2$ defective items in row $\mathcal{G}_{3,*}$; $\mathbf{y}_3'$ is thus computed from $\mathbf{y}_3$ and $\overline{\mathbf{y}}_3$ (Steps 4 to 14). The decoding procedure implemented at Step 15 produces $G_3 = \{1, 2\}$. Because $|G_3| = 2 = u$ and $\mathcal{M}_1 \vee \mathcal{M}_2 = \mathbf{y}_3'$, the set $D = D \cup G_3 = \{1, 2\}$ is obtained in Steps 16 to 18. Similarly, $\mathbf{y}_4', G_4$, and $D = D \cup G_4 = \{1, 2, 3, 4\}$ are obtained for row $\mathcal{G}_4$.

For the last row $\mathcal{G}_{5,*}$, because $y_5 = 1$ (Step 3), $\mathbf{y}_5'$ is obtained from $\mathbf{y}_5$ and $\overline{\mathbf{y}}_5$ (Steps 4 to 14). However, the decoding procedure implemented at Step 15 produces $G_5 = \emptyset$. Since $|G_5| = 0$, the condition at Step 16 does not hold. Therefore, G_5 is not added to the defective set D.

Finally, Step 21 returns the defective item set $D = \{1, 2, 3, 4\}$, which contains all defective items and no false defective items.

4.4.6 The decoding complexity

Because $\mathcal{T}$ is constructed using $\mathcal{G}$ and $\mathcal{M}$, the probability of successful decoding of $\mathbf{y}$ depends on these choices. Given an input vector $\mathbf{y}_i'$, we get the set of defective items from decoding of $\mathcal{M}$. The probability of successful decoding of $\mathbf{y}$ thus depends only on $\mathcal{G}$. Since $\mathcal{G}$ has κ rows satisfying (i) and (ii) with probability at least $1 - \epsilon$, all $|D|$ defective items can be identified by using $t = h(2k + 1)$ tests with probability of at least $1 - \epsilon$ for any $\epsilon \geq 0$.

The time to run Steps 4 to 14 is $O(k)$. Suppose that matrix $\mathcal{M}$ can be decoded in time $O(\mathsf{A})$ and that each column of $\mathcal{M}$ can be generated in time $O(\mathsf{B})$. It is natural that $k \leq O(\mathsf{B})$ because each column of $\mathcal{M}$ has k entries. It thus takes $O(\mathsf{A})$ time to run Step 15 and $u \times O(\mathsf{B})$ time to run Steps 16 to 18. Because the loop in Step 2 is run h times, the total decoding time is:

$$h \times (O(k) + \mathsf{A} + u \times O(\mathsf{B})) = O(h \times (\mathsf{A} + u\mathsf{B})).$$

We summarize the divide and conquer strategy in the following theorem:

Theorem 4.3 *Let $2 \leq u \leq d < n$ be integers and D be the defective set. Suppose that an $h \times n$ matrix $\mathcal{G}$ contains κ rows, denoted as $i_1, \ldots, i_\kappa$, such that (i) $|D_{i_1}| = \cdots = |D_{i_\kappa}| = u$ and (ii) $D_{i_1} \cup \ldots \cup D_{i_\kappa} = D$, where D_{i_l} is the index set of defective items in row $\mathcal{G}_{i_l,*}$. Suppose that a $k \times n$ matrix $\mathcal{M}$ is a d-disjunct matrix that can be decoded in time $O(\mathsf{A})$ and that each column of $\mathcal{M}$ can be generated in time $O(\mathsf{B})$. A $(2k+1)h \times n$ measurement matrix $\mathcal{T}$, as defined in (4.12), can thus be used to identify up to d defective items in u-NATGT in time $O(h \times (\mathsf{A} + u\mathsf{B}))$.*

The probability of successful decoding depends only on the event that $\mathcal{G}$ has κ rows satisfying (i) and (ii). Specifically, if that event happens with probability at least $1 - \epsilon$, the probability of successful decoding is also at least $1 - \epsilon$ for any $\epsilon \geq 0$.

4.5 Complexity of proposed scheme

We specify the matrix $\mathcal{G}$ in Theorem 4.3 to get the desired number of tests and decoding complexity for identifying up to d defective items. Note that when $u = d$, the number of defective items should be u (otherwise, every test would yield a negative outcome). In this case, Lemma 4.2 is sufficient to find the defective items. We consider the following notions of deterministic and randomized decoding. Deterministic decoding is a scheme in which all defective items are found with probability 1. It is achievable when **every** $h \times d$ submatrix of $\mathcal{G}$ is $(u, d-u)$-completely separating. Randomized decoding reduces the number of tests, in which all defective items can be found with probability at least $1 - \epsilon$ for any $\epsilon > 0$. It is achieved when **each** $h \times d$ submatrix is a $(u, d-u)$-completely separating matrix with probability at least $1 - \epsilon$.

4.5.1 Deterministic decoding

The following theorem states that there exists a deterministic algorithm for identifying all defective items by choosing $\mathcal{G}$ of size $h \times n$ to be a $(u, d-u)$-completely separating matrix in Theorem 4.1.

Theorem 4.4 *Let $2 \leq u \leq d \leq n$. There exists a $t \times n$ matrix such that up to d defective items in u-NATGT can be identified in time $t \times \mathsf{poly}(d^2 \log n)$, where*

$$t = O\left(\left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} \cdot d^3 \log n \cdot \log \frac{n}{d}\right)$$

Proof. On the basis of Theorem 4.3, a $t \times n$ measurement matrix $\mathcal{T}$ is generated as follows:

1. Choose an $h \times n$ $(u, d-u)$-completely separating matrix $\mathcal{G}$ as in Theorem 4.1, where $h = \left\lceil \left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} \left(d \log \frac{en}{d} + u \log \frac{ed}{u}\right) \right\rceil + 1$.
2. Choose a $k \times n$ d-disjunct matrix $\mathcal{M}$ as in Theorem 2.5, where $k = O(d^2 \log n)$ and the decoding time of $\mathcal{M}$ is $\mathsf{poly}(k)$.
3. Define $\mathcal{T}$ as (4.12).

Since $\mathcal{G}$ is an $h \times d$ $(u, d-u)$-completely separating matrix, for any $|D| \leq d$, an $h \times d$ pruning matrix $\mathcal{G}'$, which is created by removing $n-d$ columns $\mathcal{G}_x$ for $x \in [n] \setminus D$, is also a $(u, d-u)$-completely separating matrix with probability 1. From Definition 6, matrix $\mathcal{G}'$ is also a $(u, |D|-u)$-completely separating matrix. Then, there exists κ rows satisfying (i) and (ii). From Theorem 4.3, up to d defective items can be recovered using $t = h \cdot O(d^2 \log n)$ tests with probability 1, in time $h \cdot \mathsf{poly}(k)$. $\square$

4.5.2 Randomized decoding

For randomized decoding, matrix $\mathcal{G}$ is chosen such that the pruning matrix $\mathcal{G}'$ of size $h \times d$ created by removing $n-d$ columns $\mathcal{G}_x$ of $\mathcal{G}$ for $x \in [n] \setminus D$ is a $(u, d-u)$-completely separating matrix with probability at least $1-\epsilon$ for any $\epsilon > 0$. This results is an improved number of tests and decoding time compared to Theorem 4.4:

Theorem 4.5 *Let $2 \leq u \leq d \leq n$. For any $\epsilon > 0$, up to d defective items in u-NATGT can be identified using*

$$t = O\left(\left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} \left(u \log \frac{d}{u} + \log \frac{1}{\epsilon}\right) \cdot d^2 \log n \right)$$

tests with probability at least $1 - \epsilon$. The decoding time is $t \times \mathsf{poly}(d^2 \log n)$.

Proof. Using Theorem 4.3, a $t \times n$ measurement matrix $\mathcal{T}$ is generated as follows:

1. Choose an $h \times n$ matrix $\mathcal{G}$ as in Corollary 5, where $h = \left\lceil \left(\frac{d}{u}\right)^u \left(\frac{d}{d-u}\right)^{d-u} \left(u \log \frac{\mathrm{e}d}{u} + \log \frac{1}{\epsilon}\right) \right\rceil$.
2. Generate a $k \times n$ d-disjunct matrix $\mathcal{M}$ using Theorem 2.5, where $k = O(d^2 \log n)$ and the decoding time of $\mathcal{M}$ is $\mathsf{poly}(k)$.
3. Define $\mathcal{T}$ as (4.12).

Let $\mathcal{G}$ be an $h \times n$ matrix as described in Corollary 5. Then for any $|D| \leq d$, an $h \times d$ pruning matrix $\mathcal{G}'$, which is created by removing $n - d$ columns $\mathcal{G}_x$ for $x \in [n] \setminus D$, is a $(u, d-u)$-completely separating matrix with probability at least $1 - \epsilon$. From Definition 6, matrix $\mathcal{G}'$ is also a $(u, |D| - u)$-completely separating matrix. Then, there exist κ rows satisfying (i) and (ii) with probability at least $1 - \epsilon$. From Theorem 4.3, all $|D|$ defective items can be recovered using $t = h \cdot O(d^2 \log n)$ tests with probability at least $1 - \epsilon$ and in time $h \cdot \mathsf{poly}(k)$. $\square$

4.6 Simulation

In this section, we visualize the decoding times in Table 4.1. For deterministic decoding, the number of items n and the maximum number of defective items d are set to be $\{2^{20}, 2^{30}, 2^{40}, 2^{50}, 2^{60}\}$ and $\{100; 1,000\}$, respectively. For the randomized algorithm, the number of items n and the maximum number of defective items d are set to be[1] $\{2^{30}, 2^{50}, 2^{100}, 2^{300}, 2^{500}\}$ and $\{10; 100; 1,000\}$, respectively. The

[1]We note that the parameters are chosen for theoretical benchmarks and do not necessarily reflect the range encountered for practical applications.

threshold u is set to be $0.2d$. Finally, the precision ϵ for randomized algorithms is set to be 10^{-6}.

We see that for deterministic decoding, our proposed scheme is always better than the one proposed by Chen et al. [39] as shown in Figure 4.2. However, for randomized decoding, our proposed scheme is better than the one proposed by Chan et al. [38] for $d \leq \log n$ and large enough n, as shown in Figure 4.3. Since the decoding time in [38] is not affected much by the parameters d and u, we only plot one graph for it. Note that when $n \leq 2^{60}$, the decoding time in our proposed scheme is worse than the one in [38]. The main reason is that the decoding time of a d-disjunct matrix in Theorem 2.5 is high, i.e., $O(d^{11} \log^{17} n)$, and the number of rows in $\mathcal{G}$ is large. Therefore, if there exists any d-disjunct matrix with low decoding complexity, e.g., $O(d^2 \log^2 n)$, and the number of rows in $\mathcal{G}$ is sufficiently bounded, e.g., $O(d^2 \log^2 n)$, our proposed scheme would perform much better than the one in [38] when the number of items n is small, and would be practically feasible.

4.7 Conclusion

We introduced an efficient scheme for identifying defective items in NATGT. Its main idea is to convert the test outcomes in NATGT to NACGT by distributing defective items into tests properly. Then all defective items are identified by using some known decoding procedure in NACGT. However, the algorithm works only for $g = 0$. Extending the results to $g > 0$ is left for future work. Since the number of tests in the randomized decoding is quite large, reducing it is also an important task. Moreover, it would be interesting to consider noisy NATGT as well, in which erroneous tests are present in the test outcomes.

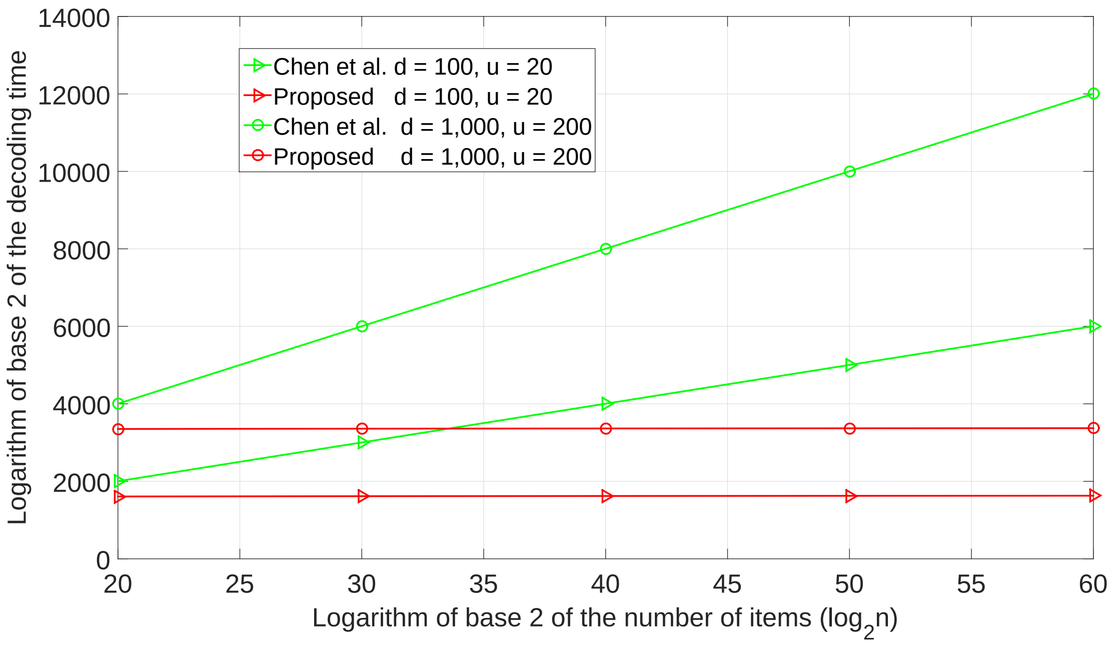


Figure 4.2: Decoding time in deterministic decoding

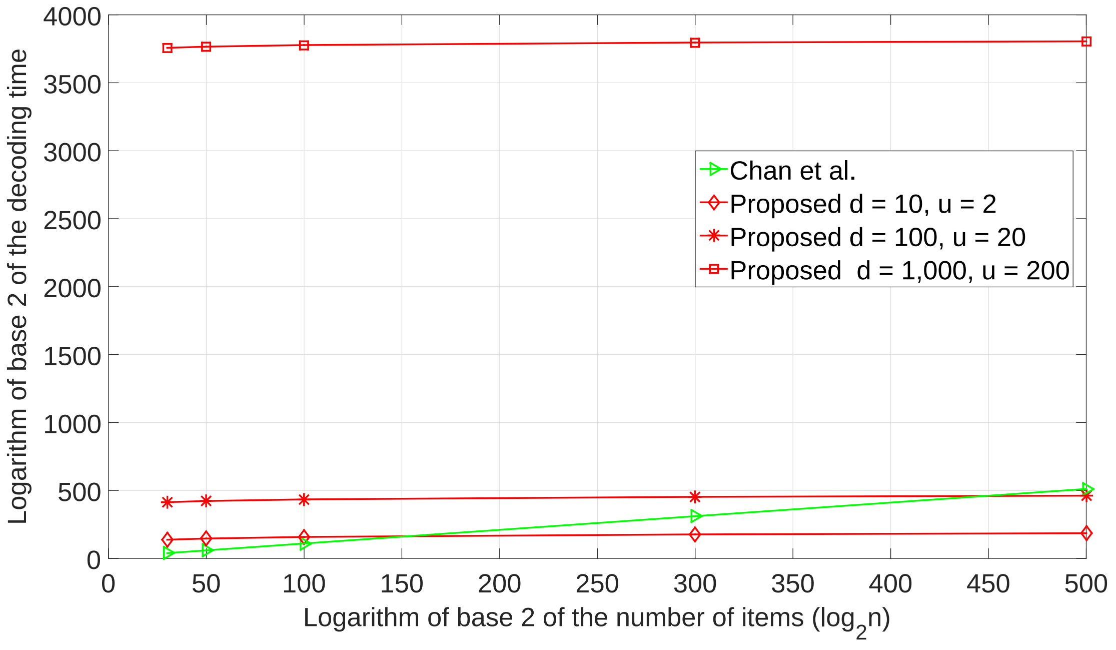


Figure 4.3: Decoding time in randomized decoding

"Nature - how, we don't know - has technology
that works in every living cell and
that depends on every atom being precisely in the right spot.
Enzymes are precise down to the last atom.
They're molecules. You put the last atom in, and it's done.
Nature does things with molecular perfection."
– Richard Smalley.

5 Non-adaptive group testing with inhibitors

5.1 Introduction

We have visited different types of defective items in a set of defective and negative items in Chapters 3 and 4. The development of NAGT applications in the field of molecular biology led to the introduction of another type of item: *inhibitor*. An item is considered to be an inhibitor if it interferes with the identification of defective items in a test, i.e., a test containing at least one inhibitor item returns negative outcome. In this "Group Testing with Inhibitors (GTI)" model, the outcome of a test on a subset of items is positive iff the subset has at least one defective item and no inhibitors. Due to great potential for use in applications, the GTI model has been intensively studied for the last two decades [6], [73], [74], [75], [10].

In NAGT using the GTI model (NAGTI), if t tests are needed to identify up to d defective items and up to h inhibitors among n items, it can be seen that they comprise a $t \times n$ measurement matrix. The procedure for obtaining the matrix is called the *construction procedure*. The procedure for obtaining the outcome of t tests using the matrix is called *encoding procedure*, and the procedure for obtaining the defective items and the inhibitor items from t outcomes is called the *decoding procedure*. Since noise typically occurs in biology experiments, we assume that there are up to e erroneous outcomes in the test outcomes. The objective of NAGTI is to *efficiently* classify all items from the encoding procedure and from the decoding procedure in the presence of noise.

There are two approaches when using NAGTI. One is to identify defective items only. Chang et al. [76] proposed a scheme using $O((d+h+e)^2 \log n)$ tests to identify all defective items in time $O((d+h+e)^2 n \log n)$. Using a probabilistic scheme, Ganesan et al. [77] reduced the number of tests to $O((d+h)\log n)$ and the decoding time to $O((d+h)n \log n)$. However, this scheme proposed is applicable only in a noise-free setting, which is restricted in practice. The second approach is to identify both defective items and inhibitors. Chang et al. [76] proposed a scheme using $O(e(d+h)^3 \log n)$ tests to classify n items in time $O(e(d+h)^3 n \log n)$. Without considering the presence of noise in the test outcome, Ganesan et al. [77] used $O((d+h^2)\log n)$ tests to identify at most d defective items and at most h inhibitor items in time $O((d+h^2)n \log n)$.

5.1.1 Problem definition

We address two problems. The first is how to efficiently identify defective items in the test outcomes in the presence of noise. The second is how to efficiently identify both defective items and inhibitor items in the test outcome in the presence of noise. Let z be an odd integer and $e = \frac{z-1}{2}$ be the maximum number of errors in the test outcomes.

Problem 5.1 *There are n items including up to d defective items and up to h inhibitor items. Is there a measurement matrix such that*

- *All defective items can be identified in time $\mathsf{poly}(d, h, e, \log n)$ in the presence of up to e erroneous outcomes, where the number of rows in the measurement matrix is much smaller than n?*
- *Each column of the matrix can be nonrandomly generated in polynomial time of the number of rows?*

Problem 5.2 *There are n items including up to d defective items and up to h inhibitor items. Is there a measurement matrix such that*

- *All defective items and inhibitors items can be identified in time $\mathsf{poly}(d, h, e, \log n)$ in the presence of up to e erroneous outcomes, where the number of rows in the measurement matrix is much smaller than n?*
- *Each column of the matrix can be nonrandomly generated in polynomial time of the number of rows?*

We note that some previous works such as [51, 52] do not consider inhibitor items. In these works, Problems 5.1 and 5.2 can be reduced to the same problem by eliminating all terms related to "inhibitor items."

5.1.2 Problem model

We model NAGTI as follows. Suppose that there are up to $1 \leq d$ defectives and up to $0 \leq h$ inhibitors in n items. Let $\mathbf{x} = (x_1, \ldots, x_n)^T \in \{0, 1, -\infty\}^n$ be the vector representation of n items. Note that the number of defective items must be at least one. Otherwise, the outcomes of the tests designed would yield negative. Item j is defective iff $x_j = 1$, is an inhibitor iff $x_j = -\infty$, and is negative iff $x_j = 0$. Suppose that there are at most d 1's in $\mathbf{x}$, i.e., $|D = \{j \mid x_j = 1, \text{ for } j = 1, \ldots, n\}| \leq d$, and at most h $-\infty$'s in $\mathbf{x}$, i.e., $|H = \{j \mid x_j = -\infty, \text{ for } j = 1, \ldots, n\}| \leq h$.

Let $\mathcal{Q} = (q_{ij})$ be a $q \times n$ binary measurement matrix which is used to identify defectives and inhibitors in n items. Item j is represented by column j of $\mathcal{Q}$ ($\mathcal{Q}_j$) for $j = 1, \ldots, n$. Test i is represented by row i in which $q_{ij} = 1$ iff the item j belongs to test i, and $q_{ij} = 0$ otherwise, where $i = 1, \ldots, q$. Then the outcome vector using the measurement matrix $\mathcal{Q}$ is

$$\mathbf{r} = \mathcal{Q} \circledast \mathbf{x} = \begin{bmatrix} r_1 \\ \vdots \\ r_q \end{bmatrix}, \tag{5.1}$$

where $\circledast$ is called the NAGTI operator, test outcome $r_i = 1$ iff $\sum_{j=1}^{n} q_{ij} x_j \geq 1$, and $r_i = 0$ otherwise for $i = 1, \ldots, q$. Note that we assume $0 \times (-\infty) = 0$ and there may be at most e erroneous outcomes in $\mathbf{r}$.

Given l binary vectors $\mathbf{y}_w = (y_{1w}, y_{2w}, \ldots, y_{Bw})^T$ for $w = 1, \ldots, l$ and some integer $B \geq 1$. The union of $\mathbf{y}_1, \ldots, \mathbf{y}_l$ is defined as vector $\mathbf{y} = \vee_{i=1}^{l} \mathbf{y}_i = (\vee_{i=1}^{l} y_{1i}, \ldots, \vee_{i=1}^{l} y_{Bi})^T$, where $\vee$ is the OR operator. Then when vector $\mathbf{x}$ is binary, i.e., there are no inhibitors in n items, (5.1) can be represented as

$$\mathbf{r} = \mathcal{Q} \circledast \mathbf{x} = \bigvee_{j=1}^{n} \mathcal{Q}_j x_j = \bigvee_{j \in D}^{n} \mathcal{Q}_j. \tag{5.2}$$

Our objective is to design the matrix $\mathcal{Q}$ such that vector $\mathbf{x}$ can be recovered when having $\mathbf{r}$ in time $\mathsf{poly}(q) = \mathsf{poly}(d, h, e, \log n)$.

5.1.3 Our contributions

Overview: Our objective is to reduce the decoding complexity for identifying up to d defectives and/or up to h inhibitors in the presence of up to e erroneous test outcomes. We present two deterministic schemes that can efficiently solve both Problems 5.1 and 5.2 with the probability 1. These schemes use two basic ideas: each column of a $t_1 \times n$ $(d+h, r; z]$-disjunct matrix (defined later) must be generated in time $\mathsf{poly}(t_1)$ and the tensor product (defined later) between it and a special signature matrix. These ideas reduce decoding complexity to $\mathsf{poly}(t_1)$. Moreover, the measurement matrices used in our proposed schemes are nonrandom, i.e., their columns can be nonrandomly generated in time polynomial of the number of rows. As a result, one can save space for storing the measurement matrices. Simulation results confirm our theoretical analysis. When the number of items is sufficiently large, the decoding time in our proposed scheme is smallest in comparison with existing work.

Comparison: We compare our proposed schemes with existing schemes in Table 5.1. There are six criteria to be considered here. The first one is construction type, which defines how to achieve a measurement matrix. It also affects how defectives and inhibitors are identified. The most common construction type is random; i.e., a measurement matrix is generated randomly. The six schemes evaluated here use random construction except for our proposed schemes.

The second criterion is decoding type: “Deterministic” means the decoding objectives are always achieved with probability 1, while “Randomized” means the decoding objectives are achieved with some high probability. Ganesan et al. [77] used randomized decoding schemes to identify defectives and inhibitors. The schemes in [76] and our proposed schemes use deterministic decoding.

The remaining criteria are: identification of defective items only, identification of both defective items and inhibitor items, error tolerance, the number of tests, and the decoding complexity. The only advantage of the schemes proposed by Ganesan et al. [77] is that the number of tests is less than ours. Our schemes outperformed the existing schemes in other criteria such as error-tolerance, the decoding type, and the decoding complexity. The number of tests with our proposed schemes for identifying defective items only (both defective items and inhibitor items, resp.) is smaller (larger, resp.) than that with the scheme proposed by Chang et al. [76]. The decoding complexity in our proposed scheme is much less than theirs when the number of items is sufficiently large.

Table 5.1: Comparison with existing schemes. “Deterministic” and “Randomized” are abbreviated as “Det.” and “Rnd.”. The $\surd$ sign means that the criterion holds for that scheme, while the $\times$ sign means that it does not. We set $e = \frac{z-1}{2}$, $\lambda = \frac{(d+h)\ln n}{\mathsf{W}((d+h)\ln n)} + z$, and $\alpha = \max\left\{\frac{\lambda}{(d+h)^2}, 1\right\}$, where $\mathsf{W}(x) = \Theta\left(\ln x - \ln\ln x\right)$.

Scheme	Construction type	Decoding type	Max. no. of # errors	Defectives only	Defectives and inhibitors	Number of tests (t)	Decoding complexity
Chang et al. [76]	Random	Det.	e	$\surd$	$\times$	$O((d+h+e)^2 \ln n)$	$O(tn)$
Ganesan et al. [77]	Random	Rnd.	0	$\surd$	$\times$	$O((d+h)\ln n)$	$O(tn)$
Proposed (Theorem 5.4)	**Nonrandom**	**Det.**	e	$\surd$	$\times$	$\Theta\left(\lambda^2 \ln n\right)$	$O\left(\frac{\lambda^5 \ln n}{(d+h)^2}\right)$
Chang et al. [76]	Random	Det.	e	$\surd$	$\surd$	$O(e(d+h)^3 \ln n)$	$O(tn)$
Ganesan et al. [77]	Random	Rnd.	0	$\surd$	$\surd$	$O((d+h^2)\ln n)$	$O(tn)$
Proposed (Theorem 5.5)	**Nonrandom**	**Det.**	e	$\surd$	$\surd$	$\Theta\left(\lambda^3 \ln n\right)$	$O\left(d\lambda^6 \times \alpha\right)$

5.2 Preliminaries

Notation is defined here for consistency. Capital letters with asterisk is denoted for multisets in which elements may appear multiple times. For example, $S = \{1, 2, 3\}$ is a set and $S^* = \{1, 1, 2, 3\}$ is a multiset. Here we assume $0 \times (-\infty) = 0$.

Some frequent notations are listed as follows:

- $\circledast$: operator for NAGTI.
- $\mathcal{S}$: $s \times n$ measurement matrix used to identify at most one defective item or one inhibitor item, where $s = 2\log n$.
- $\mathcal{M} = (m_{ij})$: $m \times n$ disjunct matrix, where integer $m \geq 1$ is number of tests.
- $\mathcal{T} = (t_{ij})$: $t \times n$ measurement matrix used to identify at most d defective items, where integer $t \geq 1$ is number of tests.
- $D; H$: index set of defective items; index set of inhibitor items.

5.3 Improved instantiation of nonrandom $(d, r; z]$-disjunct matrices

We first state the useful nonrandom construction of $(d, r; z]$-disjunct matrices, which is an instance of Theorem 2.2:

Theorem 5.3 *[36, Lemma 29] Let $1 \leq d, r, z < n$ be integers and C be a $[n_1 = q - 1, k_1]_q$-RS code. For any $d < \frac{n_1 - z}{r(k_1 - 1)} = \frac{q-1-z}{r(k_1-1)}$ and $n \leq q^{k_1}$, there exists a $t \times n$ nonrandom $(d, r; z]$-disjunct matrix where $t = O\left(q^{r+1}\right)$. Moreover, each column of the matrix can be constructed in time $O\left(q^{r+2}/(r^2 d^2)\right)$.*

An approximation of a Lambert W function $\mathsf{W}(x)$ [71] is $\ln x - \ln\ln x \leq \mathsf{W}(x) \leq \ln x - \frac{1}{2}\ln\ln x$ for any $x \geq \mathrm{e}$. Then an improved instatiation of nonrandom $(d, r; z]$-disjunct matrix is stated as follows:

Corollary 6 *For any positive integers d, r, z, and n with $d + r \leq n$, there exists a $t \times n$ nonrandom $(d, r; z]$-disjunct matrix with $t = \Theta\left(\lambda^{r+1}\right)$, where $\lambda = (rd\ln n)/(\mathsf{W}(d\ln n)) + z$. Moreover, each column of the matrix can be constructed in time $O\left(\lambda^{r+2}/(r^2 d^2)\right)$.*

Proof. From Theorem 5.3, we only need to find an $[n_1 = q - 1, k_1]_q$-RS code such that $d < \frac{n_1 - z}{r(k_1-1)} = \frac{q-1-z}{r(k_1-1)}$ and $q^{k_1} \geq n$. One chooses

$$q = \begin{cases} \frac{rd\ln n}{\mathsf{W}(d\ln n)} + z + 1 & \text{if } \frac{rd\ln n}{\mathsf{W}(d\ln n)} + z + 1 \text{ is the power of 2.} \\ 2^{\eta+1}, & \text{otherwise.} \end{cases} \tag{5.3}$$

where η is an integer satisfying $2^{\eta} < \frac{rd\ln n}{\mathsf{W}(d\ln n)} + z + 1 < 2^{\eta+1}$. We have $q = \Theta\left(\frac{rd\ln n}{\mathsf{W}(d\ln n)} + z\right)$ in both cases because $\frac{rd\ln n}{\mathsf{W}(d\ln n)} + z + 1 \leq q < 2\left(\frac{rd\ln n}{\mathsf{W}(d\ln n)} + z + 1\right)$.

Set $k_1 = \left\lceil \frac{q-z-1}{rd} \right\rceil \geq \frac{\ln n}{\mathsf{W}(d\ln n)}$. Note that the condition on d in Theorem 5.3 always holds because:

$$k_1 = \left\lceil \frac{q-z-1}{rd} \right\rceil \Longrightarrow k_1 < \frac{q-z-1}{rd} + 1 \Longrightarrow d < \frac{q-1-z}{r(k_1-1)} = \frac{n_1 - z}{r(k_1-1)}.$$

Finally, our task is to prove that $n \leq q^{k_1}$. Indeed, we have:

$$\begin{aligned} q^{k_1} \geq \left(\frac{rd \ln n}{\mathsf{W}(d \ln n)} + z + 1 \right)^{\frac{\ln n}{\mathsf{W}(d \ln n)}} &\geq \left(\frac{d \ln n}{\mathsf{W}(d \ln n)} \right)^{\frac{\ln n}{\mathsf{W}(d \ln n)}} \\ &\geq \left(\mathrm{e}^{\mathsf{W}(d \ln n) \mathrm{e}^{\mathsf{W}(d \ln n}} \right)^{1/d} = (\mathrm{e}^{d \ln n})^{1/d} = n. \end{aligned}$$

This completes our proof. □

The number of tests in our construction is better than the one in Theorem 2.2. Furthermore, there is no decoding scheme associated with matrices in this corollary. However, when $r = z = 1$, the scheme in [51] achieves the same number of tests and has an efficient decoding algorithm.

5.4 Identification of defective items

In this section, we answer Problem 5.1 that there exists a $t \times n$ measurement matrix such that: it can handle at most e errors in the test outcome; each column can be nonrandomly generated in time $\mathsf{poly}(t)$; and all defective items can be identified in time $\mathsf{poly}(d, h, e, \log n)$, where there are up to d defective items and up to h inhibitor items in n items. The main idea is to use the modified version of Algorithm 3.1 to identify all potential defective items. Then a sanitary procedure is proceeded to remove all false defective items.

Theorem 5.4 *Let $1 \leq d, h, d+h \leq n$ be integers, z be odd, and $\lambda = \frac{(d+h) \ln n}{\mathsf{W}((d+h) \ln n)} + z$. A set of n items includes up to d defective items and up to h inhibitors. Then there exists a $t \times n$ nonrandom matrix such that up to d defective items can be identified in time $O\left(\frac{\lambda^5 \log n}{(d+h)^2} \right)$ with up to $e = \frac{z-1}{2}$ errors in the test outcomes, where $t = \Theta(\lambda^2 \log n)$. Moreover, each column of the matrix can be generated in time $\mathsf{poly}(t)$.*

The proof is given in the following sections.

5.4.1 Encoding procedure

We set $e = \frac{z-1}{2}$ and $\lambda = \frac{(d+h)\ln n}{\mathsf{W}((d+h)\ln n)} + z$. Let an $m \times n$ matrix $\mathcal{M}$ be an $(d+h; z]$-disjunct matrix in Corollary 6 ($r = 1$), where

$$m = \Theta\left(\left(\frac{(d+h)\ln n}{\mathsf{W}((d+h)\ln n)} + z\right)^2\right) = O(\lambda^2).$$

Each column in $\mathcal{M}$ can be generated in time $t_1 = O\left(\frac{\lambda^3}{(d+h)^2}\right)$. Then the final $t \times n$ measurement matrix $\mathcal{T}$ is

$$\mathcal{T} = \mathcal{M} \circledcirc \mathcal{S}, \tag{5.4}$$

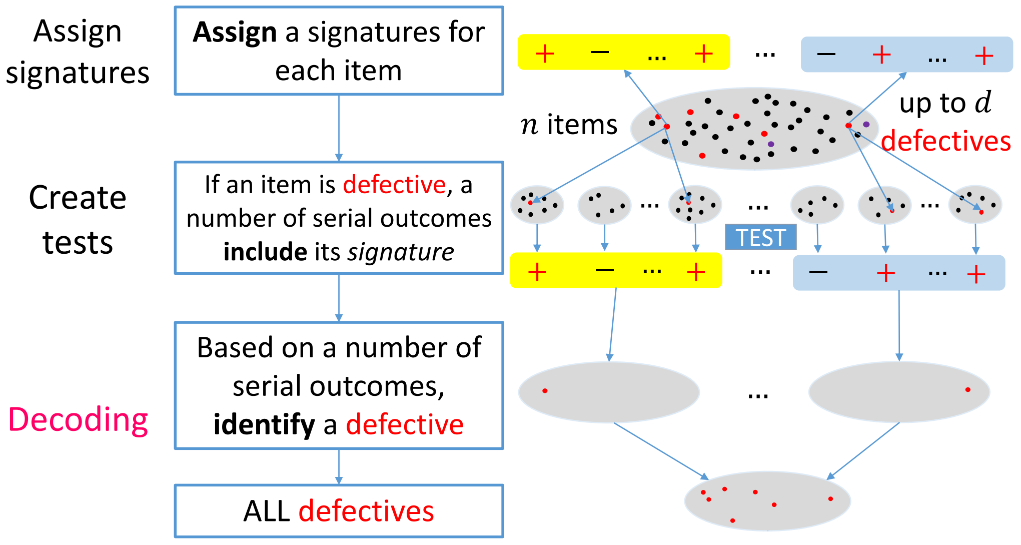
where the $s \times n$ matrix $\mathcal{S}$ is defined in (3.1) and $t = ms = \Theta\left(\lambda^2 \log n\right)$. Then it is easy to see that each column of $\mathcal{T}$ can be generated in time $t_1 \times s = \mathsf{poly}(t)$.

Any input vector $\mathbf{x} = (x_1, \ldots, x_n)^T \in \{0, 1, -\infty\}^n$ contains at most d 1's and at most h $-\infty$'s as described in section 5.1.2. Note that D and H are the index sets of the defective items and the inhibitor items, respectively. Then the binary outcome vector using the measurement matrix $\mathcal{T}$ is $\mathbf{y} = \mathcal{T} \circledast \mathbf{x} = [\mathbf{y}_1^T, \ldots, \mathbf{y}_m^T]^T$, where $\mathbf{y}_i = (\mathcal{S} \times \mathrm{diag}(\mathcal{M}_{i,*})) \circledast \mathbf{x} = [y_{(i-1)s+1}, \ldots, y_{is}]^T$, and $y_{(i-1)s+l} = 1$ iff $\sum_{j=1}^n s_{lj} m_{ij} x_j \geq 1$, and $y_{(i-1)s+l} = 0$ otherwise, for $i = 1, \ldots, m$, and $l = 1, \ldots, s$. We assume that there are at most e incorrect outcomes in the outcome vector $\mathbf{y}$.

5.4.2 Decoding procedure

Algorithm 3.1 is modified and denoted as GetDefectives$^*(\mathbf{y}, n)$ if we substitute S by multiset S^*; i.e., the output of GetDefectives$^*(\cdot)$ may have duplicated items which are used to handle the presence of erroneous outcomes in Sections 5.4 and 5.5. Line 7 is interpreted as "Add d_0 to set S^*".

Given outcome vector $\mathbf{y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_m^T)^T$, we can identify all defective items by using Algorithm 5.1. Step 1 is to identify all potential defectives and put them in the set S^*. Then Steps 3 to 8 are to remove duplicate items in the new potential defective set S_0. After that, Steps 9 to 16 are to remove all false defectives. Finally, Step 17 returns the defective set. The main idea of this algorithm can be illustrated



Assign signatures
Assign a signatures for each item
Create tests
If an item is defective, a number of serial outcomes include its signature
Decoding
Based on a number of serial outcomes, identify a defective
ALL defectives
n items
up to d defectives
TEST

Figure 5.1: Main idea of Algorithm 5.1.

as in Figure 5.2, where the signature of item j is $\mathcal{S}_j$.

5.4.3 Correctness of decoding procedure

Since matrix $\mathcal{M}$ is an $(d+h; z]$-disjunct matrix, there are at least z rows i_0 such that $m_{i_0 j} = 1$ and $m_{i_0 j'} = 0$ for any $j \in D$ and $j' \notin D \cup H \setminus \{j\}$. Since up to $e = (z-1)/2$ errors may appear in test outcome $\mathbf{y}$, there are at least $e+1$ vectors $\mathbf{y}_{i_0}$ such that the condition in Step 5 of Algorithm 3.1 holds. Consequently, each value $j \in D$ appears at least $e+1$ times. Therefore, Steps 1 to 8 return a set S_0 containing all defective items and some false defectives.

Steps 9 to 16 are to remove false defectives. For any index $j \notin D$, since there are at most $e = (z-1)/2$ erroneous outcomes, there is at least 1 row i_0 such that $t_{i_0 j} = 1$ and $t_{i_0 j'} = 0$ for all $j' \in D \cup H$. Because item $j \notin D$, the outcome of that row (test) is negative (0). Therefore, Step 12 is to check whether an item in S_0 is non-defective. Finally, Step 17 returns the set of defective items.

5.4.4 Decoding complexity

The time to run Step 1 is $O(t)$. Since $|S^*| \leq m$, it takes m time to run Steps 3 to 8. Because $|S^*| \leq m$, the cardinality of S_0 is up to m. The loop at Step 9 runs

Algorithm 5.1 GetDefectivesWOInhibitors($\mathbf{y}, n, e$): detection of up to d defective items without identifying inhibitors.

Input: a function to generate $t \times n$ measurement matrix $\mathcal{T}$; outcome vector $\mathbf{y}$; maximum number of errors e

Output: defective items

```
1: $S^* = \text{GetDefectives}^*(\mathbf{y}, n)$.                ▷ Identify all potential defectives.
2: $S_0 = \emptyset$.                                             ▷ Defective set.
3: foreach $x \in S^*$ do
4:     if $x$ appears in $S^*$ at least $e + 1$ times then
5:         $S_0 = S_0 \cup \{x\}$.
6:         Remove all elements that equal $x$ in $S^*$.
7:     end if
8: end foreach
9: for all $x \in S_0$ do                                        ▷ Remove false defectives.
10:     ▷ Get column corresponding to defective item $x$.
11:     Generate column $\mathcal{T}_x = \mathcal{M}_x \circledcirc \mathcal{S}_x$.
12:     if $\exists i_0 \in [t] : t_{i_0 x} = 1$ and $y_{i_0} = 0$ then    ▷ Condition for a false defective.
13:         $S_0 = S_0 \setminus \{x\}$.                          ▷ Remove false defectives.
14:         break;
15:     end if
16: end for
17: return $S_0$.                                                 ▷ Return set of defective item.
```

at most m times. Steps 11 and 12 take time $s \times \frac{m^{1.5}}{(d+h)^2}$ and t, respectively. The total decoding time is:

$$O(t) + m + m \times \left(s \times \frac{m^{1.5}}{(d+h)^2} + t \right) = O\left(\frac{sm^{2.5}}{(d+h)^2} \right) = O\left(\frac{\lambda^5 \log n}{(d+h)^2} \right).$$

5.5 Identification of defectives and inhibitors

In this section, we answer Problem 5.2 that there exists a $v \times n$ measurement matrix such that: it can handle at most e errors in the test outcome; each column can be nonrandomly generated in time $\mathsf{poly}(v)$; and all defective items and inhibitor items can be identified in time $\mathsf{poly}(d, h, e, \log n)$, where there are up to d defective items and up to h inhibitor items in n items.

Theorem 5.5 *Let $1 \leq d, h, d+h \leq n$ be integers, z be odd, and $\lambda = \frac{(d+h)\ln n}{\mathsf{W}((d+h)\ln n)} + z$. A set of n items includes up to d defective items and up to h inhibitors. Then there exists a $v \times n$ nonrandom matrix such that up to d defective items and up to h inhibitor items can be identified in time $O\left(d\lambda^6 \times \max\left\{\frac{\lambda}{(d+h)^2}, 1\right\}\right)$, with up to $e = \frac{z-1}{2}$ errors in the test outcomes, where $v = \Theta\left(\lambda^3 \log n\right)$. Moreover, each column of the matrix can be generated in time $\mathsf{poly}(v)$.*

To detect both up to h inhibitors and d defectives, we have to use two types of matrices: an $(d+h; z]$-disjunct matrix and an $(d+h-2, 2; z]$-disjunct matrix. The main idea is as follows. We first identify all defective items. Then all potential inhibitors are located by using an $(d+h-2, 2; z]$-disjunct matrix. The final procedure is to remove all false inhibitor items.

5.5.1 Identification of an inhibitor

Let $\veebar$ be the notation for the union of the column corresponding to the defective item and the column corresponding to the inhibitor item. We suppose that there is an outcome $\mathbf{o} := (o_1, \ldots, o_s)^T = \mathcal{S}_a \veebar \mathcal{S}_b$, where the defective item is a and the inhibitor item is b, and that $\mathcal{S}_a$ and $\mathcal{S}_b$ are two columns in the $s \times n$ matrix $\mathcal{S}$ in (3.1). Note that $o_i = 1$ iff $s_{ia} = 1$ and $s_{ib} = 0$, and $o_i = 0$ otherwise, for $i = 1, \ldots, s$. Assume that the defective item a is already known. The inhibitor item b is identified as in Algorithm 5.2.

The correctness of the algorithm is described here. Step 2 initializes the corresponding column of inhibitor b in $\mathcal{S}$. Since column $\mathcal{S}_a$ has exactly $s/2$ 1's, Steps 3 to 6 are to obtain $s/2$ positions of $\mathcal{S}_b$. Since the first half of $\mathcal{S}_a$ is the complement of its second half, it does not exist two indexes i_0 and i_1 such that $s_{i_0 a} = s_{i_1 a} = 1$, where $|i_0 - i_1| = \log n$. As a result, it does not exist two indexes i_0 and i_1 such that $s_{i_0 b} = s_{i_1 b} = -1$, where $|i_0 - i_1| = \log n$. Moreover, the first half of $\mathcal{S}_b$ is the complement of its second half. Therefore, the remaining $s/2$ entries of $\mathcal{S}_b$ can be obtained by using Steps 7 to 11. The index of inhibitor b can be identified by checking the first half of $\mathcal{S}_b$, which is done in Step 12. Finally, Step 13 returns the index of the inhibitor.

It is easy to verify that the decoding complexity of Algorithm 5.2 is $O(s)$.

Algorithm 5.2 GetInhibitorFromADefective($\mathbf{o}, \mathcal{S}_a, n$): identification of an inhibitor when defective item and union of corresponding columns are known.

Input: outcome vector $\mathbf{o} := (o_1, \ldots, o_s) = \mathcal{S}_a \vee \mathcal{S}_b$; number of items n; vector $\mathcal{S}_a$ corresponding to defective item a
Output: inhibitor item b

```
1: $s = 2\log n$.
2: Set $\mathcal{S}_b = (s_{1b}, \ldots, s_{sb})^T = (-1, -1, \ldots, -1)^T$.
3: **for** $i = 1$ to $s$ **do**                ▷ Obtain $s/2$ entries of $\mathcal{S}_b$.
4:     **If** $s_{ia} = 1$ and $o_i = 1$ **then** $s_{ib} = 0$. **end if**
5:     **If** $s_{ia} = 1$ and $o_i = 0$ **then** $s_{ib} = 1$. **end if**
6: **end for**
7: **for** $i = 1$ to $s/2$ **do**              ▷ Obtain $s/2$ remaining entries of $\mathcal{S}_b$.
8:     **If** $s_{ib} = -1$ **then** $s_{ib} = 1 - s_{i+s/2,b}$. **end if**
9:     **If** $s_{ib} = 0$ **then** $s_{i+s/2,b} = 1$. **end if**
10:    **If** $s_{ib} = 1$ **then** $s_{i+s/2,b} = 0$. **end if**
11: **end for**
12: Get index $b$ by checking first half of $\mathcal{S}_b$.
13: **return** $b$.                              ▷ Return the inhibitor item.
```

Example: Let $\mathcal{S}$ be the matrix in (3.1), where $n = 8$ and $s = 2\log n = 6$. Given item 1 is the unknown inhibitor and that item 3 is the known defective item, assume that the observed vector is $\mathbf{o} = (0, 1, 0, 0, 0, 0)^T$. The corresponding column of the defective item is $\mathcal{S}_3$. We set $\mathcal{S}_b = (-1, -1, -1, -1, -1, -1)^T$. We get $\mathcal{S}_b = (-1, 0, -1, 1, -1, 1)^T$ from Steps 3 to 6 and the complete column $\mathcal{S}_b = (0, 0, 0, 1, 1, 1)^T$ from Steps 7 to 11. Because the first half of $\mathcal{S}_b$ is $(0, 0, 0)^T$, the index of the inhibitor is 1.

5.5.2 Encoding procedure

We set $e = \frac{z-1}{2}$ and $\lambda = \frac{(d+h)\ln n}{\mathsf{W}((d+h)\ln n)} + z$. Let an $m \times n$ matrix $\mathcal{M}$ and a $g \times n$ matrix $\mathcal{G}$ be an $(d + h; z]$-disjunct matrix and an $(d + h - 2, 2; z]$-disjunct matrix in Corollary 6, respectively, where

$$m = \Theta\left(\left(\frac{(d+h)\ln n}{\mathsf{W}((d+h)\ln n)} + z\right)^2\right) = \Theta\left(\lambda^2\right),$$
$$g = \Theta\left(\left(\frac{(d+h)\ln n}{\mathsf{W}((d+h)\ln n)} + z\right)^3\right) = \Theta\left(\lambda^3\right).$$

Each column in $\mathcal{M}$ and $\mathcal{G}$ can be generated in time t_1 and t_2, respectively, where

$$t_1 = O\left(\frac{\lambda^3}{(d+h)^2}\right), t_2 = O\left(\frac{\lambda^4}{(d+h)^2}\right). \tag{5.5}$$

The final $v \times n$ measurement matrix $\mathcal{V}$ is

$$\mathcal{V} = \begin{bmatrix} \mathcal{M} \circledcirc \mathcal{S} \\ \mathcal{G} \circledcirc \mathcal{S} \\ \mathcal{G} \end{bmatrix} = \begin{bmatrix} \mathcal{T} \\ \mathcal{H} \\ \mathcal{G} \end{bmatrix}, \tag{5.6}$$

where $\mathcal{T} = \mathcal{M} \circledcirc \mathcal{S}$ and $\mathcal{H} = \mathcal{G} \circledcirc \mathcal{S}$. The sizes of matrices $\mathcal{T}$ and $\mathcal{H}$ are $t \times n$ and $h \times n$, respectively. Then we have $t = ms = 2m\log n$ and $h = gs = 2g\log n$. Note that the matrix $\mathcal{T}$ is the same as the one in (5.4). The number of tests of the measurement matrix $\mathcal{V}$ is

$$v = t + h + g = ms + gs + g = O((m+g)s) = \Theta\left(\lambda^3 \log n\right).$$

Then it is easy to see that each column of matrix $\mathcal{V}$ can be generated in time $(t_1 + t_2) \times s + t_2 = \mathsf{poly}(v)$.

Any input vector $\mathbf{x} = (x_1, \ldots, x_n)^T \in \{0, 1, -\infty\}^n$ contains at most d 1's and at most h $-\infty$'s as described in Section 5.1.2. The outcome vector using measurement matrix $\mathcal{T}$, i.e., $\mathbf{y} = \mathcal{T} \circledast \mathbf{x}$, is the same as the one in Section 5.4.1. The binary outcome vector using the measurement matrix $\mathcal{H}$ is

$$\mathbf{h} = \mathcal{H} \circledast \mathbf{x} = \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_g \end{bmatrix}, \tag{5.7}$$

where $\mathbf{h}_i = (\mathcal{S} \times \mathrm{diag}(\mathcal{G}_{i,*}))\circledast\mathbf{x} = [h_{(i-1)s+1}, \ldots, h_{is}]$, $h_{(i-1)s+l} = 1$ iff $\sum_{j=1}^{n} s_{lj} g_{ij} x_j \geq 1$, and $h_{(i-1)s+l} = 0$ otherwise, for $i = 1, \ldots, g$, and $l = 1, \ldots, s$. Therefore, the outcome vector using the measurement matrix $\mathcal{V}$ in (5.6) is:

$$\mathbf{v} = \mathcal{V} \circledast \mathbf{x} = \begin{bmatrix} \mathcal{T} \\ \mathcal{H} \\ \mathcal{G} \end{bmatrix} \circledast \mathbf{x} = \begin{bmatrix} \mathcal{T} \circledast \mathbf{x} \\ \mathcal{H} \circledast \mathbf{x} \\ \mathcal{G} \circledast \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \mathbf{h} \\ \mathbf{g} \end{bmatrix}, \tag{5.8}$$

where $\mathbf{y}$ is as same as the one in Section 5.4.1, $\mathbf{h}$ is defined in (5.7), and $\mathbf{g} = \mathcal{G} \circledast \mathbf{x} = (r_1, \ldots, r_g)^T$. We assume that $0 \times (-\infty) = 0$ and there are at most $e = (z-1)/2$ incorrect outcomes in the outcome vector $\mathbf{v}$.

5.5.3 Decoding procedure

Given outcome vector $\mathbf{v}$, number of items n, number of tests in matrix $\mathcal{M}$, number of tests in matrix $\mathcal{G}$, maximum number of errors e, and functions to generate matrix $\mathcal{V}$, $\mathcal{G}$, $\mathcal{M}$, and $\mathcal{S}$. The details of the proposed scheme is described in Algorithm 5.3. Steps 1 to 2 are to divide the outcome vector $\mathbf{v}$ into three smaller vectors $\mathbf{y}, \mathbf{h}$, and $\mathbf{g}$ as (5.8). Then Step 3 is to get the defective set. All potential inhibitors would be identified in Steps 5 to 12. Then Steps 14 to 23 are to remove most of false inhibitors. Since there may be some duplicate inhibitors and some remaining false inhibitors in the inhibitor set, Step 25 to 31 are to remove the remaining false inhibitors and make each element in the inhibitor set unique. Finally, Step 32 is to return the defective set and the inhibitor set. The main idea of this algorithm can be illustrated as in Figure 5.2, where the signature of item j is $\mathcal{S}_j$ and the signature of the defective item j_1 and the inhibitor item j_2 is $\mathcal{S}_{j_1} \underline{\vee} \mathcal{S}_{j_2}$.

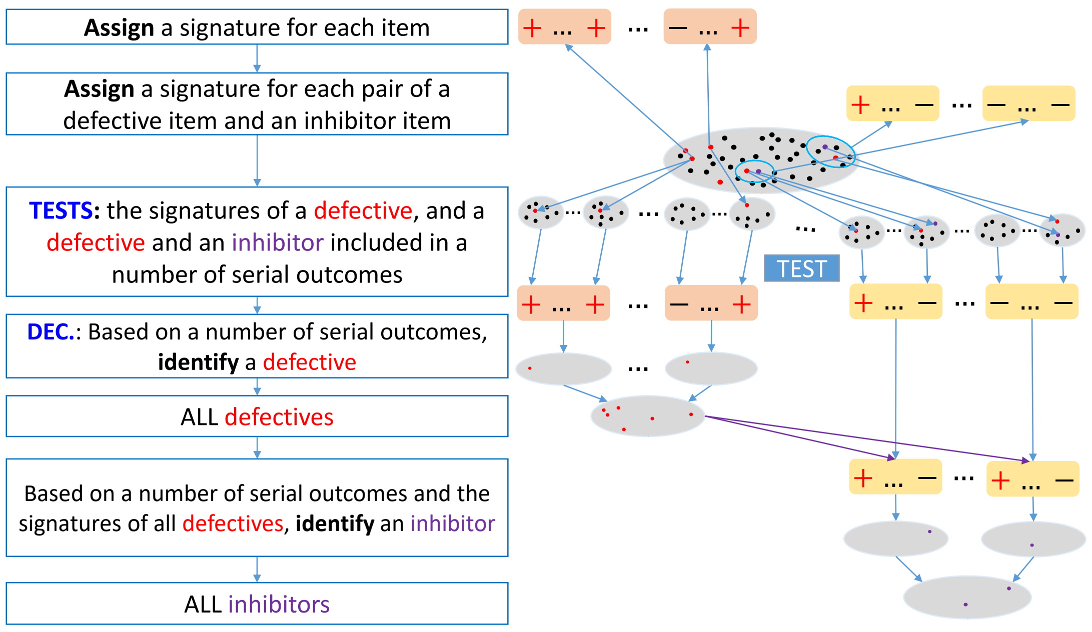

Assign a signature for each item
Assign a signature for each pair of a defective item and an inhibitor item
TESTS: the signatures of a defective, and a defective and an inhibitor included in a number of serial outcomes
DEC.: Based on a number of serial outcomes, **identify** a defective
ALL defectives
Based on a number of serial outcomes and the signatures of all defectives, **identify** an inhibitor
ALL inhibitors
TEST

Figure 5.2: Main idea of Algorithm 5.3.

5.5.4 Correctness of the decoding procedure

Because of the construction of $\mathcal{V}$, the three vectors split from the outcome vector $\mathbf{v}$ in Step 2 are $\mathbf{y} = \mathcal{T} \circledast \mathbf{x}, \mathbf{h} = \mathcal{H} \circledast \mathbf{x}$, and $\mathbf{g} = \mathcal{G} \circledast \mathbf{x}$. Therefore, the set D achieved in Step 3 is the defective set as analyzed in Section 5.4.

Let H be the true inhibitor set which we will identify. Since $\mathcal{G}$ is an $(d+h-2, 2; z]$-disjunct matrix $\mathcal{G}$, for any $j_1 \in H$ (we have not known H yet) and $j_2 \in D$, there exists at least z rows i_0's such that $g_{i_0 j_1} = g_{i_0 j_2} = 1$ and $g_{i_0 j'} = 0$, for all $j' \in D \cup H \setminus \{j_1, j_2\}$. Then, since there are at most $e = (z-1)/2$ errors in $\mathbf{v}$, there exists at least $e + 1 = (z-1)/2 + 1$ index i_0's such that $\mathbf{h}_{i_0} = \mathcal{S}_{j_1} \veebar \mathcal{S}_{j_2}$. As analyzed in Section 5.5.1, for any vector which is the union of the column corresponding to the defective item and the column corresponding to the inhibitor item, the inhibitor item is always identified if the defective item is known. Therefore, the set H_0^* obtained from Steps 7 to 12 contains all inhibitors and may contain some false inhibitors. Our next goal is to remove false inhibitors.

To remove the false inhibitors, we first remove all defective items in the set H_0^* as Step 16. Therefore, there are only inhibitors and negative items in the set H_0^* after implementing Step 16. One needs to exploit the property of the inhibitor that it will make the test outcome negative if there are at least one inhibitor and at least one defective in the same test. We pick an arbitrary defective item $y \in D$ and generate its corresponding column $\mathcal{G}_y$ in the matrix $\mathcal{G}$. Since $\mathcal{G}$ is an $(d+h-2, 2; z]$-disjunct matrix $\mathcal{G}$ and there are at most $e = (z-1)/2$ errors in $\mathbf{v}$, for any $j_1 \in H$ (we have not known H yet) and $y \in D$, there exists at least $z - e = e + 1$ rows i_0's such that $g_{i_0 j_1} = g_{i_0 y} = 1$ and $g_{i_0 j'} = 0$, for all $j' \in D \cup H \setminus \{j_1, y\}$. The outcome of these tests would be negative. Therefore, Steps 14 to 23 removes most of false inhibitors. Note that since there are at most e errors, the are at most e false inhibitors and each of them appears at most e times in the set H_0^*. Then Step 25 to 31 are to completely remove false inhibitors and make each element in the inhibitor set unique. Finally, Step 32 returns the sets of defective items and inhibitor items.

5.5.5 Decoding complexity

First, we find all potential inhibitors. It takes time $O(v)$ for Step 2. The time to get the defective set D is $O\left(\frac{sm^{2.5}}{(d+h)^2}\right) = O\left(\frac{\lambda^5 \log n}{(d+h)^2}\right)$ as analyzed in Theorem 5.4. Steps 7 and 8 have up to g and $|D| \leq d$ loops, respectively. Since Step 9 takes time $O(s)$, the running time from Steps 7 to 12 is $O(gds)$ and the cardinality of H_0^* is up to gd.

Second, we analyze the complexity of removing false inhibitors. Step 15 takes time t_1 as in (5.5). Since $|H_0^*| \leq gd$, the number of loops at Step 17 is at most gd. For the next step, it takes time t_2 for Step 18 as in (5.5). And it takes time $O(g)$ from Steps 19 to 22. As a result, it takes time $O(t_1 + gd(t_2 + g))$ for Steps 14 to 23.

Finally, Steps 25 to 31 are to remove duplicate inhibitors in the new defective set H. It takes time $O(gd)$ to do that because we know $|H_0^*| \leq gd$.

In summary, the decoding complexity is:

$$
\begin{aligned}
&O\left(\frac{sm^{2.5}}{(d+h)^2}\right) + O(gds) + O(t_1 + gd \times (t_2 + g)) + O(gd) \\
&= O\left(\frac{sm^{2.5}}{(d+h)^2}\right) + O(gd(t_2+g)) = O\left(\frac{\lambda^5 \log n}{(d+h)^2}\right) + O\left(d\lambda^3 \times \left(\frac{\lambda^4}{(d+h)^2} + \lambda^3\right)\right) \\
&= O\left(d\lambda^6 \times \max\left\{\frac{\lambda}{(d+h)^2}, 1\right\}\right).
\end{aligned}
$$

5.6 Simulation

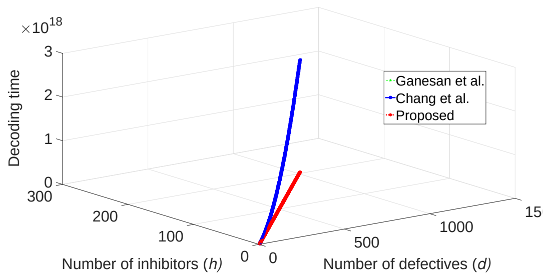
In this section, we visualize the number of tests and decoding times in Table 5.1. We evaluated variations of our proposed scheme by simulation using $d = 2, 4, \ldots, 2^{10}$, $h = 0.2d$, and $n = 2^{32}$ in Matlab R2015a on an HP Compaq Pro 8300SF desktop PC with a 3.4-GHz Intel Core i7-3770 processor and 16-GB memory. Two scenarios are considered here: identification of defective items (corresponding to section 5.4) and identification of defectives and inhibitors (corresponding to section 5.5). For each scenario, two models of noise are considered in test outcomes: noiseless setting and noisy setting. In the noisy setting, the number of errors is set to be as 100 times the summation of the number of defective items and the number of inhibitor items. Moreover, in some special cases, the number of items and the number of errors may be reconsidered.

All figures are plotted in 3 dimensions in which the x-axis (on the right of figures), y-axis (in the middle of figures), z-axis (the vertical line) represent number of defectives, number of inhibitors, and number of tests. Our proposed scheme, Ganesan et al.'s scheme, and Chang et al.'s scheme are visualized with red color with marker of circle, green color with marker of pentagram, and blue color with marker of asterisk. In the noisy setting, Ganesan et al.'s scheme is not plotted because the authors of that scheme did not consider the noisy setting.

For decoding time, when the number of items is sufficiently large, the decoding time in our proposed scheme is smaller than that of Chang et al.'s scheme and Ganesan et al.'s scheme.

5.6.1 Identification of defective items

We illustrate decoding time when defective items are the only items that we want to recover here. When there are no errors in test outcomes, as shown in Fig. 5.3, the decoding time in our proposed scheme is lowest. Since the decoding times in our proposed scheme and Ganesan et al.'s scheme are relatively equal, only one line is visible in the left subfigure of Fig. 5.3. Therefore, we zoomed in on those lines to see how close these two decoding times are. As plotted in the right subfigure of Fig. 5.3, when the number of defective items and the number of



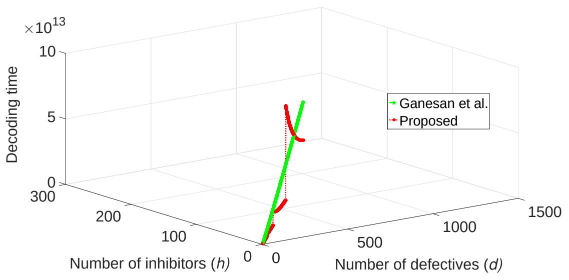


Figure 5.3: Decoding time vs. number of defectives and number of inhibitors for identifying only defective items when there are no errors in test outcomes.

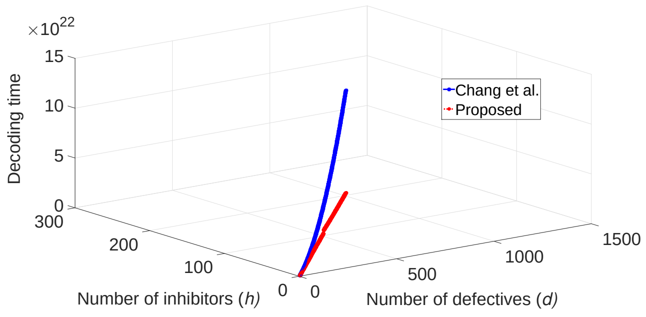
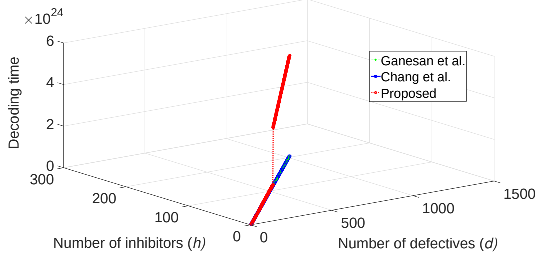


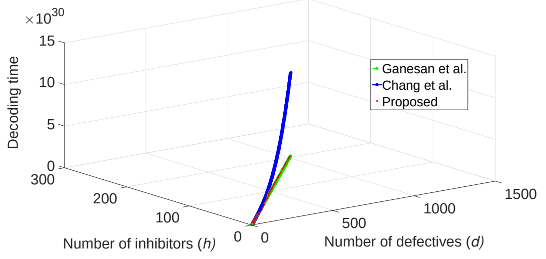
Figure 5.4: Decoding time vs. number of defectives and number of inhibitors for identifying only defective items with presence of erroneous outcomes.

inhibitor items are small, the decoding time in our proposed scheme is always smaller the one in Ganesan et al.'s scheme. As the number of defective items and the number of inhibitor items increase, the decoding time in our proposed scheme first becomes larger the one in Ganesan et al.'s scheme, though it becomes smaller after the number of items reaches some threshold. We note that if the number of defective items and inhibitor items are fixed while the number of total items is sufficiently large, the decoding time in our proposed scheme is always smaller than the ones in Chang et al.'s scheme and Ganesan et al.'s scheme.

When some erroneous outcomes are allowed, the decoding time in our proposed scheme is always smaller than the one in Chang et al.'s scheme as shown in Fig. 5.4.

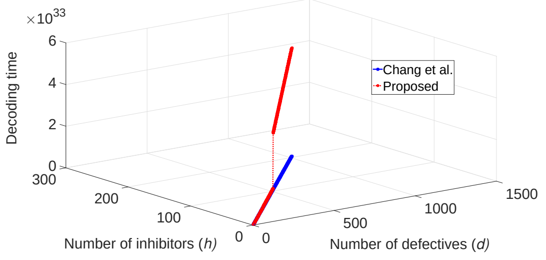


(a) $n = 2^{32}$

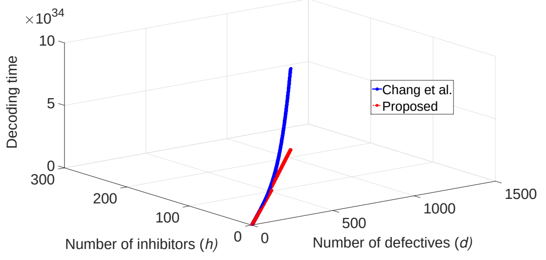


(b) $n = 2^{66}$

Figure 5.5: Decoding time vs. number of defectives and number of inhibitors for classifying items when there are no errors in test outcomes.



(a) $n = 2^{32}$



(b) $n = 2^{61}$

Figure 5.6: Decoding time vs. number of defectives and number of inhibitors for classifying items when there are some erroneous outcomes.

5.6.2 Identification of defectives and inhibitors

We illustrate decoding time for classifying all items. In principle, the complexity of the decoding time in our proposed scheme is smallest in comparison with the ones in Chang et al.'s scheme and Ganesan et al.'s scheme when the number of items is sufficiently large. When there are no errors in test outcomes, the decoding time of the proposed scheme is smallest when the number of items is at least 2^{66}, as shown in subfigure (b) of Fig. 5.5. When some erroneous outcomes are allowed, the decoding time in our proposed scheme is always smaller than the one in Chang et al.'s scheme when the number of items is at least 2^{61}, as shown in subfigure (b) of Fig. 5.6.

5.7 Conclusion

We have presented two schemes efficiently identifying up to d defective items and up to h inhibitors in the presence of e erroneous outcomes in time $\mathsf{poly}(d, h, e, \log n)$. This decoding complexity is substantially less than that of state-of-the-art systems in which the decoding complexity is $\mathsf{poly}(d, h, e, n)$. However, the number of tests with our proposed schemes is slightly higher. Moreover, we have not considered an inhibitor complex model [76] in which each inhibitor in this work would be transferred to a bundle of inhibitors. Such a model would be much more complicated and is left for future work.

Algorithm 5.3 GetInhibitors($\mathbf{v}, n, e, m, g$): identification of up to d defectives and up to h inhibitors.

Input: outcome vector $\mathbf{v}$; number of items n; number of tests in matrix $\mathcal{M}$; number of tests in matrix $\mathcal{G}$; maximum number of errors e; and functions to generate matrix $\mathcal{V}$, $\mathcal{G}$, $\mathcal{M}$, and $\mathcal{S}$

Output: defective items and inhibitor items

```
1: $s = 2\log n$.    ▷ number of rows in the matrix $\mathcal{S}$.
2: Divide vector $\mathbf{v}$ into three smaller vectors $\mathbf{y}, \mathbf{h}$, and $\mathbf{g}$ such that $\mathbf{v} = (\mathbf{y}^T, \mathbf{h}^T, \mathbf{g}^T)^T$ and number of entries in $\mathbf{y}, \mathbf{h}$, and $\mathbf{g}$ are $ms, gs$, and $g$, respectively.
3: $D$ = GetDefectivesWOInhibitors($\mathbf{y}, n, e$).    ▷ defective set.
4: ▷ Find all potential inhibitors.
5: Divide vector $\mathbf{h}$ into $g$ smaller vectors $\mathbf{h}_1, \ldots, \mathbf{h}_g$ such that $\mathbf{h} = (\mathbf{h}_1^T, \ldots, \mathbf{h}_g^T)^T$ and their size are equal to $s$.
6: $H_0^* = \emptyset$.    ▷ Initialize inhibitor multiset.
7: for $i = 1$ to $g$ do    ▷ Scan all outcomes in $\mathbf{h}$.
8:     foreach $x \in D$ do
9:         $i_0$ = GetInhibitorFromADefective($\mathbf{h}_i, \mathcal{S}_x, n$).
10:        Add item $i_0$ to multiset $H_0^*$.
11:    end foreach
12: end for
13: ▷ Remove most of false inhibitors.
14: Assign $(r_1, \ldots, r_g)^T = \mathbf{g}$.
15: Generate a column $\mathcal{G}_y$ for any $y \in D$.    ▷ Get the column of a defective.
16: $H_0^* = H_0^* \setminus D$.
17: foreach $x \in H_0^*$ do    ▷ Scan all potential inhibitors.
18:     Generate column $\mathcal{G}_x$
19:     if $\exists i_0 \in [g] : g_{i_0x} = g_{i_0y} = 1$ and $r_{i_0} = 1$ then
20:         Remove all elements that equal $x$ in $H_0^*$.    ▷ Remove the false inhibitor.
21:         break;
22:     end if
23: end foreach
24: ▷ Completely remove false inhibitors and duplicate inhibitors.
25: $H = \emptyset$.
26: foreach $x \in H_0^*$ do
27:     if $x$ appears in $H_0^*$ at least $e + 1$ times then
28:         $H = H \cup \{x\}$.
29:         Remove all elements that equal $x$ in $H_0^*$.
30:     end if
31: end foreach
32: return $D$ and $H$.    ▷ Return set of defective items.
```

“We can only see a short distance ahead, but
we can see plenty there that needs to be done.”
– Alan Turing.

6 Summary and future work

6.1 Summary of contributions

This thesis explored algorithmic aspects of group testing. It addressed how to design tests and how to utilize them for decoding for various group testing models. All of the decoding complexities proposed are sub-linear wrt the number of items. Further, most of the proposed schemes attain good performance in practice. The key underlying ideas binding the improvements achieved in algorithmic aspects together is to use divide-and-conquer strategies.

Since non-adaptive classical group testing has been widely used in practice for decades, an efficient nonrandom design of the measurement matrices is required. In Chapter 3, we presented a nonrandom design that is quite simple, is easy to implement, and requires low memory for practice]al application. Moreover, we overcame a 54–year bound on nonrandom construction for non-adaptive classical group testing. Empirical results demonstrated the superior performance compared of existing schemes.

By revising the definition of defective items, a new model of group testing is devised: threshold group testing. We presented in Chapters 4 efficient decoding schemes for this in which the decoding complexity is sub-linear wrt the number of items. These schemes outperform existing ones in which the decoding complexities are polynomial or exponential wrt the number of items. Nonetheless, the number of tests associated with decoding is too large for practical application.

The diversity in group testing basically reflects in biology. As inhibitor molecules inhibit the functioning of other molecules, inhibitor items added to a set of items in classical group testing. In Chapter 5, we presented an efficient decoding scheme that is again sub-linear wrt the number of items. The empirical results obtained by implementing the proposed scheme match those of theoretical analysis.

6.2 Directions for future work

While the problems in classical group testing have mostly been solved, several problems in non-classical group testing remains unsolved. This fact leads to several natural extensions of the work described in this thesis for non-classical group testing.

The number of tests in threshold group testing is impractically high. As the threshold increases, the number of tests exponentially increases. The number could be reduced by placing strict conditions on u and d, but such conditions would make any solutions associated with them impractical. Removing the strict conditions while keeping the number of tests low is thus an important task. Another important task is determining whether there exists a solution such that the decoding time is a polynomial of d, u, and $\log n$.

Those open problems for non-classical group testing described above are considered under the algorithmic viewpoint. Interesting tasks from the viewpoint of information theory include investigating the upper and lower bounds on the number of tests and the decoding time for non-classical group testing.

Moving from theory to practice, it is essential to apply group testing to real-world problems. It would be best to design tests in practice corresponding to tests in theory. However, this is a costly and time-consuming approach. A more promising approach is to use available biological datasets for creating tests instead of carrying out biological experiments.

Bibliography

[1] R. Dorfman, "The detection of defective members of large populations," *The Annals of Mathematical Statistics*, vol. 14, no. 4, pp. 436–440, 1943.

[2] A. J. Macula, D. C. Torney, and P. Villenkin, "Two-stage group testing for complexes in the presence of errors," *DIMACS Sries in Disc. Math. and Theor. Comput. Sci*, vol. 55, pp. 145–157, 2000.

[3] A. G. D'yachkov, I. V. Vorobyev, N. Polyanskii, and V. Y. Shchukin, "On a hypergraph approach to multistage group testing problems," in *2016 IEEE International Symposium on Information Theory (ISIT).* IEEE, 2016, pp. 1183–1191.

[4] L. Baldassini, O. Johnson, and M. Aldridge, "The capacity of adaptive group testing," in *2013 IEEE International Symposium on Information Theory.* IEEE, 2013, pp. 2676–2680.

[5] D. Du, F. K. Hwang, and F. Hwang, *Combinatorial group testing and its applications.* World Scientific, 2000, vol. 12.

[6] M. Farach, S. Kannan, E. Knill, and S. Muthukrishnan, "Group testing problems with sequences in experimental molecular biology," in *Compression and Complexity of Sequences 1997. Proceedings.* IEEE, 1997, pp. 357–367.

[7] D. C. Torney, "Sets pooling designs," *Annals of Combinatorics*, vol. 3, no. 1, pp. 95–101, 1999.

[8] A. De Bonis and U. Vaccaro, "Constructions of generalized superimposed codes with applications to group testing and conflict resolution in multiple

access channels," *Theoretical Computer Science*, vol. 306, no. 1-3, pp. 223–243, 2003.

[9] P. Damaschke, "Threshold group testing," in *General theory of information transfer and combinatorics.* Springer, 2006, pp. 707–718.

[10] F.-h. Chang, H. Chang, and F. K. Hwang, "Pooling designs for clone library screening in the inhibitor complex model," *Journal of combinatorial optimization*, vol. 22, no. 2, pp. 145–152, 2011.

[11] T. V. Bui, M. Kuribayashi, M. Cheraghchi, and I. Echizen, "A framework for generalized group testing with inhibitors and its potential application in neuroscience," *arXiv preprint arXiv:1810.01086*, 2018.

[12] H. Q. Ngo and D.-Z. Du, "A survey on combinatorial group testing algorithms with applications to dna library screening," *Discrete mathematical problems with medical applications*, vol. 55, pp. 171–182, 2000.

[13] W. J. Bruno, E. Knill, D. J. Balding, D. Bruce, N. Doggett, W. Sawhill, R. Stallings, C. C. Whittaker, and D. C. Torney, "Efficient pooling designs for library screening," *Genomics*, vol. 26, no. 1, pp. 21–30, 1995.

[14] A. J. Macula, "Probabilistic nonadaptive group testing in the presence of errors and dna library screening," *Annals of Combinatorics*, vol. 3, no. 1, pp. 61–69, 1999.

[15] H. Chen, D.-Z. Du, and F. K. Hwang, "An unexpected meeting of four seemingly unrelated problems: graph testing, dna complex screening, superimposed codes and secure key distribution," *Journal of Combinatorial Optimization*, vol. 14, no. 2-3, pp. 121–129, 2007.

[16] H. F. Kwang-ming and D. Ding-zhu, *Pooling Designs And Nonadaptive Group Testing: Important Tools For Dna Sequencing.* World Scientific, 2006, vol. 18.

[17] M. T. Goodrich, M. J. Atallah, and R. Tamassia, "Indexing information for data forensics," in *International Conference on Applied Cryptography and Network Security.* Springer, 2005, pp. 206–221.

[18] G. Cormode and S. Muthukrishnan, "What's hot and what's not: tracking most frequent items dynamically," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 249–278, 2005.

[19] A. Bar-Noy, F. K. Hwang, I. Kessler, and S. Kutten, "A new competitive algorithm for group testing," in *[Proceedings] IEEE INFOCOM'92: The Conference on Computer Communications.* IEEE, 1992, pp. 786–793.

[20] J. Hayes, "An adaptive technique for local distribution," *IEEE Transactions on Communications*, vol. 26, no. 8, pp. 1178–1186, 1978.

[21] S. Wu, S. Wei, Y. Wang, R. Vaidyanathan, and J. Yuan, "Partition information and its transmission over boolean multi-access channels," *IEEE Transactions on Information Theory*, vol. 61, no. 2, pp. 1010–1027, 2014.

[22] A. De Bonis and U. Vaccaro, "ϵ-almost selectors and their applications to multiple-access communication," *IEEE Transactions on Information Theory*, vol. 63, no. 11, pp. 7304–7319, 2017.

[23] A. Dyachkov, N. Polyanskii, V. Shchukin, and I. Vorobyev, "Separable codes for the symmetric multiple-access channel," *IEEE Transactions on Information Theory*, 2019.

[24] S. Ubaru and A. Mazumdar, "Multilabel classification with group testing and codes," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR.org, 2017, pp. 3492–3501.

[25] G. K. Atia and V. Saligrama, "Boolean compressed sensing and noisy group testing," *IEEE Trans. on Information Theory*, vol. 58, no. 3, pp. 1880–1901, 2012.

[26] D. Malioutov and M. Malyutov, "Boolean compressed sensing: Lp relaxation for group testing," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2012, pp. 3305–3308.

[27] M. Shi, T. Furon, and H. Jégou, "A group testing framework for similarity search in high-dimensional spaces," in *Proceedings of the 22nd ACM international conference on Multimedia.* ACM, 2014, pp. 407–416.

[28] P. Meerwald and T. Furon, "Group testing meets traitor tracing," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on.* IEEE, 2011, pp. 4204–4207.

[29] T. V. Bui, O. K. Nguyen, V. H. Dang, N. T. Nguyen, and T. D. Nguyen, "A variant of non-adaptive group testing and its application in pay-television via internet," in *Information and Communication Technology-EurAsia Conference.* Springer, 2013, pp. 324–330.

[30] D. R. Stinson, T. Van Trung, and R. Wei, "Secure frameproof codes, key distribution patterns, group testing algorithms and related structures," *Journal of Statistical Planning and Inference*, vol. 86, no. 2, pp. 595–617, 2000.

[31] A. C. Gilbert, M. A. Iwen, and M. J. Strauss, "Group testing and sparse signal recovery," in *2008 42nd Asilomar Conference on Signals, Systems and Computers.* IEEE, 2008, pp. 1059–1063.

[32] C. Wang, Q. Zhao, and C.-N. Chuah, "Optimal nested test plan for combinatorial quantitative group testing," *IEEE Transactions on Signal Processing*, vol. 66, no. 4, pp. 992–1006, 2017.

[33] S. Khattab, S. Gobriel, R. Melhem, and D. Mossé, "Live baiting for service-level dos attackers," in *IEEE INFOCOM 2008-The 27th Conference on Computer Communications.* IEEE, 2008, pp. 171–175.

[34] T. Madej, "An application of group testing to the file comparison problem," in *[1989] Proceedings. The 9th International Conference on Distributed Computing Systems.* IEEE, 1989, pp. 237–243.

[35] Y. Xuan, I. Shin, M. T. Thai, and T. Znati, "Detecting application denial-of-service attacks: A group-testing-based approach," *IEEE Transactions on parallel and distributed systems*, vol. 21, no. 8, pp. 1203–1216, 2009.

[36] M. Cheraghchi, "Improved constructions for non-adaptive threshold group testing," *Algorithmica*, vol. 67, no. 3, pp. 384–417, 2013.

[37] G. De Marco, T. Jurdziński, M. Różański, and G. Stachowiak, "Subquadratic non-adaptive threshold group testing," in *International Symposium on Fundamentals of Computation Theory.* Springer, 2017, pp. 177–189.

[38] C. L. Chan, S. Cai, M. Bakshi, S. Jaggi, and V. Saligrama, "Stochastic threshold group testing," in *Information Theory Workshop (ITW), 2013 IEEE.* IEEE, 2013, pp. 1–5.

[39] H.-B. Chen and H.-L. Fu, "Nonadaptive algorithms for threshold group testing," *Discrete Applied Mathematics*, vol. 157, no. 7, pp. 1581–1585, 2009.

[40] H. Chang, H.-B. Chen, H.-L. Fu, and C.-H. Shi, "Reconstruction of hidden graphs and threshold group testing," *Journal of combinatorial optimization*, vol. 22, no. 2, pp. 270–281, 2011.

[41] H. Abasi, N. H. Bshouty, and H. Mazzawi, "On exact learning monotone dnf from membership queries," in *International Conference on Algorithmic Learning Theory.* Springer, 2014, pp. 111–124.

[42] H. Chang, H.-L. Fu, and C.-H. Shih, "Learning a hidden graph," *Optimization Letters*, vol. 8, no. 8, pp. 2341–2348, 2014.

[43] H. Abasi, N. H. Bshouty, and H. Mazzawi, "Non-adaptive learning of a hidden hypergraph," *Theoretical Computer Science*, vol. 716, pp. 15–27, 2018.

[44] N. H. Bshouty, "Exact learning from an honest teacher that answers membership queries," *Theoretical Computer Science*, vol. 733, pp. 4–43, 2018.

[45] A. De Bonis, "New combinatorial structures with applications to efficient group testing with inhibitors," *Journal of Combinatorial Optimization*, vol. 15, no. 1, pp. 77–94, 2008.

[46] W. Wu, Y. Li, C.-h. Huang, and D.-Z. Du, "Molecular biology and pooling design," in *Data Mining in Biomedicine.* Springer, 2007, pp. 133–139.

[47] H.-B. Chen and A. De Bonis, "An almost optimal algorithm for generalized threshold group testing with inhibitors," *Journal of Computational Biology*, vol. 18, no. 6, pp. 851–864, 2011.

[48] M. T. Thai, D. MacCallum, P. Deng, and W. Wu, "Decoding algorithms in pooling designs with inhibitors and error-tolerance," *International journal of bioinformatics research and applications*, vol. 3, no. 2, pp. 145–152, 2007.

[49] F. K. Hwang and F. Chang, "The identification of positive clones in a general inhibitor model," *Journal of Computer and System Sciences*, vol. 73, no. 7, pp. 1090–1094, 2007.

[50] D.-Z. Du and F. K. Hwang, "Identifying d positive clones in the presence of inhibitors," *International journal of bioinformatics research and applications*, vol. 1, no. 2, pp. 162–168, 2005.

[51] T. V. Bui, M. Kuribayashi, T. Kojima, R. Haghvirdinezhad, and I. Echizen, "Efficient (nonrandom) construction and decoding for non-adaptive group testing," *Journal of Information Processing*, vol. 27, pp. 245–256, 2019.

[52] T. V. Bui, M. Kuribayashi, M. Cheraghchi, and I. Echizen, "Efficiently decodable non-adaptive threshold group testing," *IEEE Transactions on Information Theory*, 2019.

[53] T. V. Bui, M. Kuribayashi, T. Kojima, and I. Echizen, "Sublinear decoding schemes for non-adaptive group testing with inhibitors," in *International Conference on Theory and Applications of Models of Computation.* Springer, 2019, pp. 93–113.

[54] W. Kautz and R. Singleton, "Nonrandom binary superimposed codes," *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 363–377, 1964.

[55] D. R. Stinson and R. Wei, "Generalized cover-free families," *Discrete Mathematics*, vol. 279, no. 1-3, pp. 463–477, 2004.

[56] A. D'yachkov, P. Vilenkin, D. Torney, and A. Macula, "Families of finite sets in which no intersection of ℓ sets is covered by the union of s others," *Journal of Combinatorial Theory, Series A*, vol. 99, no. 2, pp. 195–218, 2002.

[57] H.-B. Chen, H.-L. Fu, and F. K. Hwang, "An upper bound of the number of tests in pooling designs for the error-tolerant complex model," *Optimization Letters*, vol. 2, no. 3, pp. 425–431, 2008.

[58] E. Porat and A. Rothschild, “Explicit nonadaptive combinatorial group testing schemes,” *IEEE Transactions on Information Theory*, vol. 57, no. 12, pp. 7982–7989, 2011.

[59] P. Indyk, H. Q. Ngo, and A. Rudra, “Efficiently decodable non-adaptive group testing,” in *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms.* SIAM, 2010, pp. 1126–1142.

[60] H. Q. Ngo, E. Porat, and A. Rudra, “Efficiently decodable error-correcting list disjunct matrices and applications,” in *International Colloquium on Automata, Languages, and Programming.* Springer, 2011, pp. 557–568.

[61] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[62] S. Cai, M. Jahangoshahi, M. Bakshi, and S. Jaggi, “Grotesque: noisy group testing (quick and efficient),” in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on.* IEEE, 2013, pp. 1234–1241.

[63] K. Lee, R. Pedarsani, and K. Ramchandran, “Saffron: A fast, efficient, and robust framework for group testing based on sparse-graph codes,” in *Information Theory (ISIT), 2016 IEEE International Symposium on.* IEEE, 2016, pp. 2873–2877.

[64] V. Guruswami and P. Indyk, “Linear-time list decoding in error-free settings,” in *International Colloquium on Automata, Languages, and Programming.* Springer, 2004, pp. 695–707.

[65] M. Cheraghchi, “Noise-resilient group testing: Limitations and constructions,” *Discrete Applied Mathematics*, vol. 161, no. 1-2, pp. 81–95, 2013.

[66] V. Guruswami *et al.*, “Algorithmic results in list decoding,” *Foundations and Trends® in Theoretical Computer Science*, vol. 2, no. 2, pp. 107–195, 2007.

[67] M. F. Chowdhury, C.-P. Jeannerod, V. Neiger, E. Schost, and G. Villard, "Faster algorithms for multivariate interpolation with multiplicities and simultaneous polynomial approximations," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2370–2387, 2015.

[68] J. Von Zur Gathen and M. Nöcker, "Exponentiation in finite fields: theory and practice," in *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes.* Springer, 1997, pp. 88–113.

[69] T. V. Bui, T. Kojima, M. Kuribayashi, and I. Echizen, "Efficient decoding schemes for noisy non-adaptive group testing when noise depends on number of items in test," *arXiv preprint arXiv:1803.06105*, 2018.

[70] F. Parvaresh and A. Vardy, "Correcting errors beyond the guruswami-sudan radius in polynomial time," in *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on.* IEEE, 2005, pp. 285–294.

[71] A. Hoorfar and M. Hassani, "Inequalities on the lambert w function and hyperpower function," *J. Inequal. Pure and Appl. Math*, vol. 9, no. 2, pp. 5–9, 2008.

[72] A. Dyachkov, V. Rykov, C. Deppe, and V. Lebedev, "Superimposed codes and threshold group testing," in *Information Theory, Combinatorics, and Search Theory.* Springer, 2013, pp. 509–533.

[73] A. De Bonis and U. Vaccaro, "Improved algorithms for group testing with inhibitors," *Information Processing Letters*, vol. 67, no. 2, pp. 57–64, 1998.

[74] A. De Bonis, L. Gasieniec, and U. Vaccaro, "Optimal two-stage algorithms for group testing problems," *SIAM J. on Comp.*, vol. 34, no. 5, pp. 1253–1270, 2005.

[75] F. K. Hwang and Y. Liu, "Error-tolerant pooling designs with inhibitors," *Journal of Computational Biology*, vol. 10, no. 2, pp. 231–236, 2003.

[76] H. Chang, H.-B. Chen, and H.-L. Fu, "Identification and classification problems on pooling designs for inhibitor models," *Journal of Computational Biology*, vol. 17, no. 7, pp. 927–941, 2010.

[77] A. Ganesan, S. Jaggi, and V. Saligrama, "Non-adaptive group testing with inhibitors," in *2015 IEEE Information Theory Workshop (ITW)*. IEEE, 2015, pp. 1–5.