Thesis for the degree of Doctor of Philosophy

New Constructions for Competitive and Minimal-Adaptive Group Testing

Muhammad Azam Sheikh

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Göteborg, Sweden 2013

New Constructions for Competitive and Minimal-Adaptive Group Testing

Muhammad Azam Sheikh

Printed by Chalmers Reproservice
Göteborg, Sweden, October 2013

ISBN 978-91-7385-913-4

This thesis has been typeset using LaTeX.

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie Nr 3594
ISSN 0346-718X
Technical Report No. 98D

Algorithms/LAB Research Group
Computing Science Division
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Göteborg, Sweden

Phone: +46 (0)31 772 1000
Fax: +46 (0)31 772 3663
E-mail: azams@chalmers.se

Mistakes concealed, every request granted
Āqa is even gracious, to a servant so derailed

–To Bapa Jan & All Those Whom I Hold Dear!

New Constructions for Competitive and Minimal-Adaptive Group Testing

Muhammad Azam Sheikh
Department of Computer Science and Engineering
Chalmers University Of Technology

Abstract

Group testing (GT) was originally proposed during the World War II in an attempt to minimize the *cost* and *waiting time* in performing identical blood tests of the soldiers for a low-prevalence disease. Formally, the GT problem asks to find $d \ll n$ *defective* elements out of n elements by querying subsets (pools) for the presence of defectives. By the information-theoretic lower bound, essentially $d \log_2 n$ queries are needed in the worst-case. An *adaptive* strategy proceeds sequentially by performing one query at a time, and it can achieve the lower bound. In various applications, nothing is known about d beforehand and a strategy for this scenario is called *competitive*. Such strategies are usually adaptive and achieve query optimality within a constant factor called the *competitive ratio*.

In many applications, queries are time-consuming. Therefore, *minimal-adaptive* strategies which run in a small number s of stages of parallel queries are favorable. This work is mainly devoted to the design of minimal-adaptive strategies combined with other demands of both theoretical and practical interest. First we target unknown d and show that actually competitive GT is possible in as few as 2 stages only. The main ingredient is our randomized estimate of a previously unknown d using nonadaptive queries. In addition, we have developed a systematic approach to obtain optimal competitive ratios for our strategies. When d is a known upper bound, we propose randomized GT strategies which asymptotically achieve query optimality in just 2, 3 or 4 stages depending upon the growth of d versus n.

Inspired by application settings, such as at American Red Cross, where in most cases GT is applied to small instances, *e.g.*, $n = 16$. We extended our study of query-optimal GT strategies to solve a given problem instance with fixed values n, d and s. We also considered the situation when elements to test cannot be divided physically (electronic devices), thus the pools must be disjoint. For GT with *disjoint* simultaneous pools, we show that $\Theta(sd(n/d)^{1/s})$ tests are sufficient, and also necessary for certain ranges of the parameters.

Keywords: minimal-adaptive group testing, competitive group testing, competitive ratio, randomization, disjoint pools, strict group testing, exact bounds

Acknowledgments

Sailing across the warm waters of PhD, I feel privileged to be surrounded by cool and caring people of the Computer Science and Engineering department and particularly the members of the Computing Science division.

I must start with my sailing master, Peter Damaschke, whose enlightening guidance and a habit of doing things in a disciplined and timely manner made me move across in a smooth, rapid and confident manner. There were waves of all sizes and shapes, but I cannot remember even a single instance when Peter left me despondent. His encouragement and professional vision was always there in getting through the hard times skillfully. I am thankful to Peter for providing me an invaluable opportunity to learn from him and successfully participate in some good piece of research work. I also express my appreciation to all new and old members of our research group.

Next, I would like to extend my gratitude to my follow-up committee members: Graham Kemp, Aarne Ranta, Philippas Tsigas, Gerardo Schneider, and Koen Lindström Claessen. They together played a constructive role to keep me on the right track. I also take this opportunity to give thanks to Ana Bove for providing a balanced division of the teaching workload. As a result, I had the opportunity to work with some nice people during my teaching duties. Among them, I would like to mention: Devdatt, Erland, Prasad, Birgit, Vinay, Leonid, Filippo, Willard, Bassel and Fredrik. Thanks to all of you, specially to Vinay for nice discussions and the sweets he shares with me. Leonid is also unique because he was my first guide at Chalmers and helped me at different occasions.

Acknowledgment is incomplete without thanking Jonna, Eva, Tiina, Marianne and Peter Helander who have helped me in several day-to-day matters at the department. Shafqat, Abdul Rahim and Usman for their company in extracurricular activities, and all others whom I cannot name here.

Fortunately, I am blessed with the wife a person like I can dream of! She, along with our kids, provided a continuous support with their smiles and love. Thank you for being with me. Finally, with all my heart, I would like to extend esteem gratitude and honor for my beloved parents and family.

List of Publications

The thesis is based on the work contained in the following papers, referred to by respective Roman numeral based names in the text:

Paper-I
"Competitive Group Testing and Learning Hidden Vertex Covers with Minimum Adaptivity." Discrete Mathematics, Algorithms and Applications 2(3), 291–311 (2010)

Paper-II
"Bounds for Nonadaptive Group Tests to Estimate the Amount of Defectives." Discrete Mathematics, Algorithms and Applications 3(4), 517–536 (2011)

Paper-III
"Randomized Group Testing both Query-Optimal and Minimal Adaptive." In: Bielikova, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turan, G. (eds.) SOFSEM 2012: Theory and Practice of Computer Science, LNCS, vol. 7147, pp. 214–225. Springer Berlin Heidelberg (2012)

Paper-IV
"A Toolbox for Provably Optimal Multistage Strict Group Testing Strategies." In: Du, D.Z., Zhang, G. (eds.) COCOON 2013: Computing and Combinatorics, LNCS, vol. 7936, pp. 446–457. Springer Berlin Heidelberg (2013)

Paper-V
"Two New Perspectives on Multi-Stage Group Testing." Algorithmica 67(3), 324–354 (2013)

Contents

Part I
Preamble

CHAPTER 1

Fundamental Concepts and Recent Applications

The chapter begins with a simple example to introduce the basic concept and uses it as a vehicle to present a real application of group testing. Then, we briefly describe the very first paper on group testing to further elaborate on the main idea. We also touch on the early history of the field. Next, we present some definitions and briefly talk about models and methods of group testing.

Moving towards the end of this chapter, we explain some of the recent applications of group testing. The chapter closes by setting up the scope of the thesis.

1.1 Number Guessing Game

Let us travel back in time to the early days of our hight school education. Almost everyone of us can remember the *number guessing* game usually used to trick a friend with our mental abilities, whereas actually just using some simple arithmetic operations. The basic setting of the game is to ask a friend to think of a number and keep it secret. Then, by asking him a few apparently irrelevant questions about the secret number, a guess is made with absolute certainty (assuming that questions are answered correctly).

1.1.1 Easygoing

One childish example is the following: Start with a bunch of positive integers and ask your friend to choose one of them as his secret number. For simplicity, let us assume that your friend has 32 integers to choose from. You arrange the numbers into 2 disjoint groups each containing of half of them, *i.e.*, 16 in each group in this case. Then, show one of the groups to your friend and ask if the unknown number is present in this group. Answer will be either "Yes" or "No". Regardless of which group is shown, the answer localizes the group containing the unknown number. Now, start with this group and repeat the above strategy. Playing the same way 4 times, you reach a group of 2 numbers only. Pick one of these 2 and ask whether this is the unknown number. If this is not, the other one must be.

Thus, for a collection of 32 numbers, the unknown number is revealed after asking a total of 5 questions only. However, asking questions sequentially one by one and splitting into half after every question enables your friend to easily follow that the "magic" behind is simply a binary splitting approach.

1.1.2 Imitating Magic

To make the above "guessing one out of n" trick more interesting and look like a real magic, consider the following. Let $S = \{0, ..., 31\}$ be our set of 32 positive integers and x be the unknown. Define the following groups (subsets of S):

$A = \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31\}$,
$B = \{2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31\}$,
$C = \{4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23, 28, 29, 30, 31\}$,
$D = \{8, 9, 10, 11, 12, 13, 14, 15, 24, 25, 26, 27, 28, 29, 30, 31\}$,
$E = \{16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31\}$.

Every group contains exactly 16 elements. Now, play the game again with your friend. This time, you can present all 5 groups at once or in any

order, and record the "Yes"/"No" answers about the presence of x in every group. From all the groups, discard all those numbers which appear in "No" answered groups. Then, there remains only one number which is common to all "Yes" groups, and that must be the x provided the answers were correct. In particular, a "Yes" response to all groups means $x = 31$ while a "No" to all implies that $x = 0$.

The game becomes even more interesting by letting your friend fix 2 unknown numbers x and y. Now you would like to guess both x and y using a minimum number of questions. We leave the interested reader to ponder over the minimum number of groups required for two unknowns. A possible direction is to look at the Contributions part of this thesis for a systematic approach to solve it. For now, we stop here and turn to put the above two versions of number guessing in the context of group testing.

1.1.3 From Number Guessing to Group Testing

Suppose we have a file whose copies are placed at 2 physically distinct sites, *e.g.*, network backup servers. For automatic backup and synchronization purposes, the files must be compared. A file is partitioned into a number of pages, and a *signature* uniquely represents every distinct page. Similarly, a *composite signature* provides a unique identity for a group of pages (subset of a file). In either case, the signature is simply a number which is a function of the contents of page(s). The sites exchange these signatures and compare them to determine pages with differing contents. When the composite signature value is different for the same group of pages of a file from 2 sites, then it implies that at least one of the pages have disagreeing contents. In a distributed computing environment, this is known as *file comparison* problem.

Here, we solve a toy example of the file comparison problem. Let copies of a file f consisting of 32 pages be placed on distributed sites s_1 and s_2. Due to a communication link failure, the site s_1 was down for some time. Meanwhile, an *unknown page* has been changed at s_2. Upon reconnecting, s_1 realizes that one unknown page has been updated at s_2, while the remaining 31 pages of f are the same at both sites. Assume that composite signature comparison incurs the same cost as signature comparison of a single page. Then, the task is to decide on a minimum number of signature comparisons which s_1 should request from pages of f at s_2 to recognize the outdated page at s_1. It is easy to see that the situation of our particular example of file comparison is completely analogous to the number guessing instance discussed in the preceding section. Every number corresponds to a page. The number(s) you show to your friend in a question correspond to the page(s)

whose (composite) signature should be compared. The answer will be "Yes" if signature values from both s_1 and s_2 are different and "No" otherwise. Thus, we can apply the same trick to solve this task. In the absence of any other constraints, both versions of the number guessing are equally applicable in this situation.

The above file comparison example is actually a very simplified interpretation of the problem solved by Madej [42], which is among one of the early applications of group testing. At that time, he successfully applied group testing to solve a more general version of the distributed file comparison problem. The main idea remains the same, although we have cooked the example data to adapt well in the context of the instance solved in number guessing. In this way, we let our reader to seamlessly travel from number guessing to group testing.

Using the above example, we now give a general description of the group testing problem. Suppose that we have a collection of apparently similar items, like the pages of a file. A few of them are different than others in "some" sense, *e.g.*, pages with changed contents. We call them defectives. A test is performed to determine the value of a specific binary feature of interest in the items. In our example, comparison of two signature values corresponds to a test. The items can be tested individually like the signature comparison of two copies of a page. Comparable to the composite signature comparison, a test can also be performed on a group of 2 or more items.

Just like the changed pages having dissimilar signature values whereas the unchanged pages have matching signatures, the defective items behave differently than the rest when tested against that particular binary feature. The test outcome is "Yes" if the group contains at least one defective, that is, at least one page is changed. Otherwise, the test outcome is "No" which resembles in our example to the case when no page is changed. As comparing two signature values is independent of the number of pages represented in the signatures, the cost of a test is independent of the group size. Thus, it is possible to detect the existence of any defective item in a group by performing a single group test. Upon a negative outcome of a group test we simply discard the items in that group because we know there is no defective in that group.

Let there be n items d of which are defective. If all items are tested individually, the cost will be linear in the number of items. However, relying on the fact that items are "group testable", using a collection of carefully chosen subsets of the n elements and testing them for the presence of defectives can potentially lead to much fewer tests required to discover all the defectives.

As we have seen in the file comparison example where we try to minimize the number of signature comparisons, the main objective of group testing is

to identify the defective items using a minimum number of group tests. This is roughly the main idea of group testing and the above two versions of the number guessing game actually solve the group testing problem for $n = 32$ and $d = 1$. The questions that you ask correspond to group tests and the answers of your friend represent the outcome of the tests. In fact, we have silently introduced you with two major types of group testing algorithms. The first setting, Section 1.1.1, which is based on a halving strategy represents adaptive group testing, while the second one exemplifies a nonadaptive algorithm for the group testing problem.

We don't go in further details as we will talk about formal definitions and related concepts in the later sections. Trickier settings of the number guessing game along the theme of group testing can be arranged to perform the "magic" in front of an audience as explained in [30, 5].

1.2 Origin and Early History of Group Testing

Digging into the literature we learn that the group testing (GT) problem has a long history dating back to World War II, around 1943, when it was originally proposed. It is mostly credited to a Swiss physician Robert Dorfman due to his seminal paper [17]. However, there are evidences that his colleagues, particularly David Rosenblatt, also contributed somehow, at least in the initial brainstorming discussions where the idea was first presented.

The motivation behind the development of the first GT strategy was to reduce the number of blood tests required for screening of draftees for the U.S. Army [17] against various diseases including syphilis. Chemical analysis of a blood sample was used to reveal presence (positive outcome) or absence (negative outcome) of syphilis germs. There were millions of draftees with a possibility of only a few thousands of them actually suffering from the disease because syphilis had a low prevalence.

Knowing that identical testing is required for a large group of people where only a fraction of them are actually infected, an economical alternative testing method was suggested. The idea is based on the simple observation that blood samples drawn from a number of individuals can be analyzed simultaneously with just one test. In order to achieve that, we can pool the blood samples together to form a group blood sample and perform the chemical analysis test. Now, a negative test outcome would mean that entire group is healthy, *i.e.*, nobody in the group is suffering from syphilis. Thus, a single test would be sufficient to decide about the whole group. On the contrary,

a positive test outcome confirms the presence of at least one infected person in the group. Depending on the group size, either we can go for individual testing or further dividing the group into subgroups. Each alternative has its own pros and cons. Negative outcome suggests a larger group size so that many individuals can be screened together. However, increasing the group size also increases the chances of getting a positive outcome, thus favors for a smaller group size.

The main question addressed by Dorfman [17] is to estimate a reasonable group size in order to take advantage of GT. He considered that each soldier has an independent probability r of suffering from syphilis. Now, assuming that a group of m soldiers is selected at random, $(1-r)^m$ gives the probability that there is no infected soldier in the group, that is, a negative outcome. In other words, with probability $(1-(1-r)^m)$ the test outcome is positive indicating at least one infected soldier in the group. It is easy to see that the success of the proposed idea largely depends on the group size m and the probability r. To highlight it, Dorfman did some calculations of optimal group sizes for different values of r and presented expected savings attainable using GT. Under the assumptions he made, GT can save 50% tests on average for $r = 0.07$ and the saving increases for decreasing r and reaches 80% for $r = 0.01$. However, for large values such as $r = 0.3$, the saving reduces to 1% only.

In his paper, Dorfman concluded the discussion by presenting two points which are now the basis for designing any GT algorithm.

- Performing a group test should be economical compared to individual testing of the members of the group.

- Number of infected individuals should be far less than the total number of individuals.

Thus, as long as the prevalence rate is small, the blood tests on groups of carefully chosen sizes will more often result in a negative outcome. Therefore, we expect to reveal status of the entire individuals with less number of tests compared to their total count.

For a long time, there were no follow-up ideas on GT. Dorfman's proposal of saving tests went into darkness and might have not been thought upon by other researchers. Apparently, it seems to appear again in a problem described by Feller in his book [28], which later sparked new interests on GT. Sterrett [51], Sobel and Groll [49] and Watson [58] seem to be the early successors of Dorfman to consider and further develop theory around GT. Sobel and Groll also explained several industrial applications of GT.

1.3 Preliminaries

Dorfman assumed prior probability of each individual being infected. Another situation could be that we know the actual value or an upper bound on the number of infected individuals but not their identities, similar to the scenarios discussed under Sections 1.1.1 and 1.1.2. Being more realistic, both the probability distribution and the upper bound might not be known and the task is to determine both the number (or proportion) and the identities of the individuals suffering from the disease. Similarly, there are variations depending on the construction of groups and the testing plan, *i.e.*, performing tests sequentially one group at a time versus multiple groups in parallel. We refer to a GT algorithm as a *group testing strategy*, or just a *strategy*. Before we talk about any strategies, we first give some formal definitions.

1.3.1 Basic Definitions

Let X be a set of size n. Usually n is very large and elements of X represent the items on which we want to perform testing. Each element has a binary status, *i.e.*, it is either *defective* or not. A defective element is also called *positive* while a nondefective element is known as *negative*. The set consisting of all defective elements is called the *defective set* or simply the set of *defectives* for short.

Group

A group is any subset of X, also called a *pool*. A pool is said to be a *positive pool* if it contains at least one defective, and otherwise a *negative pool*.

Group Test

A *query* on a pool is called a group test and reveals whether the pool is positive or negative. A negative outcome declares that all elements are nondefective. On the other hand, a positive outcome just confirms presence of one or more defectives without revealing their identity information. Throughout we use the words query or test synonymously to always refer to a group test.

Group Size vs. Testing Cost

In the basic setting, the cost of a query is fixed and assumed to be independent of the group size. Moreover, theoretically there is no limit on the size of a group.

Disjoint vs. Overlapping Pools

It is generally assumed that "enough" samples are available for each element so that it may be tested simultaneously in several pools. A GT strategy using same number of pools but requiring comparatively lesser number of instances per element is preferred over other strategies. However, in the following, we do not care about the extent of overlapping. In the next chapter, we will talk about the GT strategies when pools must be disjoint.

Broadly, GT can be categorized into statistical and combinatorial models. In the following we give a bit of their details and derive expressions on the least number of queries required.

1.3.2 Combinatorial Model

In the *combinatorial group testing* (CGT) model first studied by Li [40], besides the set X of elements, the number of defectives $d \leq n$ or an upper bound on d is also known *a priori*. Usually d is small compared to n, and the task is to find the defectives by asking a minimum number of queries to arbitrarily chosen pools. The GT problem discussed under number guessing falls in the combinatorial model of GT.

In CGT, the query number is minimized for the worst-case scenario. The only information CGT requires about the defectives is that it is a subset of X consisting of at most d elements. It should also be noted that in case of CGT, d does not necessarily grow at a constant rate for increasing value of n.

Promised vs. Probable d

Let P be the actual set of defectives and $t(n, d)$ be the minimum number of queries required in the worst-case for a GT strategy to solve a problem in the combinatorial model. For a given d, $|P| \leq d$ is expected to hold. However, since d is just an assumed upper bound for the defectives, the bad case $|P| > d$ may also happen. Depending upon the knowledge about d, two different versions of the GT problem are studied. If $|P| \leq d$ is known for sure, the GT problem is called *hypergeometric group testing*. Thus, a strategy for hypergeometric GT relies on the given bound d and its objective is to determine P. On the other hand, if a bound d on the number of defectives is given but $|P| \leq d$ is not promised, the GT problem is called *strict group testing*. Thus, a GT strategy under the strict GT model is given an upper bound d, and the strategy should identify P if $|P| \leq d$ holds. When the

unlikely case $|P| > d$ happens, the strategy just reports that there are more than d defectives and need not to identify them.

Since d is an upper bound, there exist $\sum_{i=0}^{d} \binom{n}{i}$ possible ways to choose P. Therefore, for hypergeometric GT $t(n,d) \geq \log\left(\sum_{i=0}^{d} \binom{n}{i}\right)$ tests are required to discover the right P. In the case of strict GT, the expression becomes $t(n,d) \geq \log\left(1 + \sum_{i=0}^{d} \binom{n}{i}\right)$. The additional 1 added to the sum is to take care of the outcome "$|P| > d$".

Query Complexity

By the information-theoretic lower bound, at least $\log \binom{n}{d}$ pools are needed if d out of n elements are defective. Here and in the following, logarithms are always base 2, if not said otherwise. We also omit ceiling brackets in expressions for simplicity.
Performing standard calculations using Stirling's approximation we can write:

$$\log \binom{n}{d} \approx n \log n - d \log d - (n-d) \log(n-d) + \frac{1}{2} \left(\log n - \log d - \log(n-d)\right) \tag{1.1}$$

Let $x = \frac{d}{n}$, simplifying the above expression we get:

$$\log \binom{n}{d} \approx d \left(\log\left(\frac{n}{d}\right) + f(x)\right) + c \tag{1.2}$$

where

$$f(x) = \left(1 - \frac{1}{x}\right) \log(1-x) \tag{1.3}$$

and the constant term

$$c = \frac{1}{2}\left(\log\left(\frac{1}{d}\right) - \log(1-x)\right) \tag{1.4}$$

The additive term $f(x)$ increases as the ratio x decreases and attains its maximum value of 1.44 for very small x.

The optimal *cut-off* point for the combinatorial GT model, *i.e.*, the ratio x below which GT becomes more efficient than the trivial individual testing, was studied in [29]. They proved that $n-1$ queries are necessary for $x \geq \frac{1}{3}$ if groups of at most two elements are allowed. Du and Hwang [18] have presented a general proof for slightly larger value, *i.e.*, $x \geq \frac{8}{21}$. At this cut-off value, we have $f(x) = 1.13$, which is still greater than 1. Thus, the worst-case lower bound for the CGT problem is $t(n,d) \geq d \log(\frac{n}{d}) + 1.44d$. For known d, the upper bound $t(n,d) \leq \log \binom{n}{d} + d - 1$ is proven by Hwang [34]

for his *generalized binary splitting algorithm*. When d is unknown, Schlaghoff and Triesch [47] proved an upper bound with a gap of d compared to the case of known d. Precisely, they prove that without knowing d in advance, in the worst case, $\log \binom{n}{d} + 2d$ tests are required.

1.3.3 Statistical Model

The statistical model requires a probability distribution over the defectives and is named as *probabilistic group testing* (PGT) problem. PGT is most often studied under the binomial probability distribution, called *binomial group testing*, *i.e.*, each element is independently defective with fixed probability r. The binomial distribution is called the standard assumption for PGT [16]. Research under the statistical model can be further divided in two problems: *classification* and *estimation*.

Estimation Problem

In the estimation problem of PGT, GT is carried out to estimate the proportion of the defectives in the given collection. In other words, the probability r is unknown and the task is to estimate it using GT. Apparently, Thompson [54], and Sobel and Elashoff [48] are among the pioneers who studied the estimation problem when little or nothing is known beforehand about the probability r.

Generally, the PGT estimation problem has been focused on determining the *sample size* in statistical *sampling*. That is, the process of determining the optimal number of individuals which must be included in a study to make inferences on the whole of a large population. For this purpose, the classical approach has been to apply maximum likelihood estimation assuming that the population adheres to binomial distribution.

The main application areas are in determining infection rates and disease prevalence. For example, in [54], Thompson starts by dividing a population of $n = pq$ insects into p pools of size q each. An insect is believed to be able to spread the "infection" by carrying a virus with some unknown probability r and a group of insects can be tested together for the presence of infectious insects. The objective is to estimate proportion of infectious insects without identifying them individually. Precisely, optimal number q which reflects the number of insects in a group capable of spreading the infection is of interest. Using binomial model and applying the maximum likelihood estimation method to the test outcomes, he has proposed a method to optimally choose q for different ranges of the unknown proportion r.

Estimation under a threshold of detection: As discussed in Section 1.3.1, theoretically, there is no limit on the pool size. Therefore, usually it is not considered as an important parameter when studying the query bounds for GT in general. However, due to the widespread use of GT and its variants in the fields like biology, chemical testing, disease detection in humans, animals as well as plants, testing a group beyond a certain size sometimes faces a so called pool "dilution effect". This simply means that the positive items are diluted in the pool resulting in a *false negative* outcome. Common reasons, among others, are the limitations of the testing instruments or the positive items are very sensitive to the presence of many negative items in the same pool.

In such a situation, a positive outcome from a group test can only be observed if the number of defectives in the group are above some threshold, known as *threshold of detection*. A generalized version of threshold GT considering upper and lower bound of detection threshold is first studied by Damaschke [15]. Later on, threshold GT has been adopted and developed in various theoretical and applied contexts [25, 7, 14, 8, 9, 1]. Recently, new constructions of probabilistic threshold GT with promising results are studied in [6].

In the presence of a threshold of detection, the conventional approach for the estimation problem uses several pools of small sizes. Yamamura and Hino [60] studied this problem and showed that they can estimate the proportion of defectives for any group size even in the presence of threshold of detection. The problem considered is the detection of genetically modified organisms (GMOs) in a given population. Analytical instrument used to infer the presence of GMOs in a given sample has a threshold of detection. Due to dilution, larger group sizes are prone to false negative outcome, that is, an outcome is negative even if the group contains one unit of GMOs. They provide maximum likelihood estimate for the proportion of defectives and also study optimal pool size satisfying a certain criterion on the defective rate.

We refer to section 2 of [60] for exact expressions of maximum likelihood estimates with and without threshold of detection. They also provide a good list of literature on the estimation problem.

Classification Problem

The classification problem is comparable to CGT, in the sense that the objective is to detect all defectives individually. Remember that in CGT the defective number is bounded by d and the query complexity is studied for the worst-case scenario. In the classification problem of PGT, defectives usually

follow a fixed probability model and a strategy tries to reduce the expected number of pools required to reveal the status of the entire collection from the binary test outcomes under the assumed probability model. Dorfman's model where *a priori* information about the probability of each individual having the disease is assumed, actually falls under PGT classification problem.

Query Complexity: We study the query complexity for the classification problem. Assuming that the items are selected independently and equally probably, the defectives rate r is essentially equal to $x = \frac{d}{n}$ as in the query complexity of the combinatorial model in the previous Section 1.3.2.

Thus, the Eq. (1.2) can be adapted in terms of r to get the query complexity for the PGT. Ignoring the constant term from the Eq.(1.2) and putting $d = rn$, the expected number of queries for n items under PGT is given by:

$$n\left(r\log\left(\frac{1}{r}\right) + (1-r)\log\left(\frac{1}{1-r}\right)\right) \tag{1.5}$$

and dividing by n simplifies to

$$r\log\left(\frac{1}{r}\right) + (1-r)\log\left(\frac{1}{1-r}\right) \tag{1.6}$$

expected queries per item. Using r as the success probability, the same expression for the lower bound also follows from the binary entropy function.

The study of the optimal *cut-off point* for the statistical model is due to Ungar [57]. It has been proved that individual testing minimizes the expected queries compared to any PGT strategy if $r \geq \frac{1}{2}\left(3-\sqrt{5}\right)$.

1.4 Group Testing Methods

Regardless of the model, a strategy solving a GT problem can be adaptive, nonadaptive, or works in a specified number of stages. In the following, we are mainly concerned about complexity bounds for the combinatorial model.

1.4.1 Adaptive Strategies

A strategy is called *adaptive* if queries are performed one at a time and choice of the elements for a later test can be based on all the previous test outcomes. An upper bound for adaptive strategies is $O(d\log n)$. It follows simply from the *halving strategy*, an example of which is Section 1.1.1. That

is, start by querying the whole set, and if the outcome is positive, divide the elements in two pools of equal size and query one of these. If it is positive, continue halving with this pool, otherwise with the second half. In this way, with $O(\log n)$ adaptive queries we can identify one defective and remove it from the set. Repeating it d times we get the upper bound. Since each iteration starts with a group test on the whole set, the process stops when no defectives are left. Therefore, in this adaptive strategy we do not need any prior knowledge of d and essentially $d \log n$ queries are sufficient.

Unknown d

While the lower bound for adaptive strategies is $d \log(\frac{n}{d})$, a strategy with $O(d \log(\frac{n}{d}))$ tests can be easily devised even for unknown d. Here, the real challenge is to bound the hidden factor in the query number by some small constant. Du and Hwang [19] were the first who studied the GT problem when nothing is known about d beforehand. Inspired by the study of online algorithms [43], they proposed *competitive group testing*. Let $t_A(d|n)$ denote the maximum number of queries required by an algorithm A if there are d unknown defectives. Then A is called *c-competitive* if there exist constants c and a such that $t_A(d|n) \leq ct(n,d) + a$ holds for $0 \leq d < n$. Beginning with [4, 22, 23], substantial work has been done to minimize the constant factor c, called the *competitive ratio*. To our best knowledge, the currently best competitive ratio for deterministic, adaptive strategies is 1.5 [47].

1.4.2 Nonadaptive Strategies

Nonadaptive strategies are on the other extreme, that is, all the pools should be prepared in advance without knowing the outcome of any test, and then queried simultaneously. Disjunct matrices form the basis of pooling designs for nonadaptive GT strategies. In logic, *disjunction* means two logical statements connected with an *or*. In the context of nonadaptive strategies, a *disjunct matrix* is a binary (0/1) matrix that satisfies a constraint based on the disjunction of subset of columns as specified below. Disjunction of 2 or more columns is simply a logical *or* applied row wise to the corresponding 0/1 entries of the columns involved.

Let M be a $t \times n$ binary matrix whose t rows correspond to pools and n columns represent elements. $M[t_i, c_j] = 1$ means that the pool t_i contains the element c_j while $M[t_i, c_j] = 0$ implies absence of the element c_j in the pool t_i. A column c_j is said to *contain* another column c_k if for all rows t_i, whenever $M[t_i, c_k] = 1$, we have $M[t_i, c_j] = 1$. The matrix M is called d-disjunct if and only if the disjunction of any d columns does not contain any of the other

$n - d$ columns. To interpret the outcome of a pooling design based on a d-disjunct matrix, it is easy to devise a *decoding* algorithm requiring $O(tn)$ comparisons.

Sometimes it happens that the same combinatorial structure is considered and developed in different contexts using varying names to solve "apparently" distinct problems. Nonadaptive GT strategies are one such example. Disjunct matrices are equivalent to binary superimposed codes [36], also called cover-free families [27].

Actually, the d-disjunctness constraint makes sure that for every possible set D of up to d defectives, there exists a collection C of pools which collectively contains all the elements except those of D, *i.e.*, $X \setminus D$. Now, if the d elements of D are the only defectives, all pools in C must be negative. Discarding negative pools and all the elements of these pools from the positive pools as well, the remaining up to d elements are the defectives and $|P| \leq d$ is verified. However, if d was not a correct bound, more than d candidates will remain, indicating $|P| > d$ has happened. Thus, a construction of pools using a d-disjunct matrix also fulfills the requirements of strict GT.

It is well known [46, 24] that a lower bound on $t(n, d)$ for nonadaptive strategies is $\Omega\left(\frac{d^2}{\log d} \log n\right)$. However, $O(d^2 \log n)$ pools are sufficient and currently the best factor is 4.28; see [10] and the references therein.

1.4.3 Multistage Strategies

In between the above two, as a third option, there are *multistage* strategies, also called *s-stage* strategies, where the testing plan is divided into a fixed number s of stages. A *stage* refers to one round of simultaneously querying pools. Stages are treated adaptively while all pools within a stage are prepared prior to the stage. Here the main advantage is that the queries prepared for the next stage can depend on the outcome of all previous stages. In the context of stages, nonadaptive strategies are 1-stage strategies whereas in adaptive strategies there is no limit on s.

Due to the time consuming nature of adaptive strategies, multistage strategies are more favorable in situations when the problem size n is huge and a group test may take several hours [38]. The very first strategy proposed by Dorfman [17] is actually a 2-stage PGT which was later extended to an *s-stage* strategy by Li [40] for CGT problem when exact d is given.

Trivial 2-stage Strategy

A 2-stage strategy is called *trivial* if parallel tests are run on a certain number of pools in stage 1. Then, based on the test outcomes, all candidate positives

are tested individually in stage 2. Thus, a trivial 2-stage strategy does not require any further decoding.

These 2-stage strategies require an upper bound d on the number of defectives and guarantee that all the defectives can be identified using $cd\log(\frac{n}{d})$ pools, c being some constant. This was first shown in [16] with a high constant c, more precisely $7.54\,d\log(\frac{n}{d})$. It was later improved to $4\,d\log(\frac{n}{d})$ [26] and currently $c = 1.9$ for all d, and asymptotically to $c = 1.44$ as d grows [10].

1.5 Applications

The GT strategy proposed to detect syphilis among World War II soldiers was not applied due to the limitations of the blood testing practices at that time [20]. However, since the idea is so appealing, later on it has been successfully adopted. Nowadays, GT is being used not only in screening people in low-prevalence diseases, but it has also been proven beneficial and has widely gained popularity in diverse disciplines outside of infectious disease detection.

The book [20] by Du and Hwang, 2*nd* edition published in 2000, is by far the only book covering a variety of GT algorithms and discusses the applications of GT in several areas. Later in 2006, the same authors have compiled another book [21], this time dedicated to nonadaptive GT and their applications to DNA sequencing. They mention several applications of GT in molecular biology.

These days, biology is one of the major application area of GT, where nonadaptive strategies are often referred to as *pooling designs*. Within biology, another notable application of GT is high-throughput biological experiments. Kainkaryam did his PhD thesis [35] on this topic. In particular, he has considered high-throughput drug screening and gene expression microarrays, and developed pooling designs applicable in actual experimentation environment.

It is very difficult to provide a comprehensive list of different problems which benefit from the theory of GT. As part of the thesis, we have randomly picked some of the recent papers where GT testing has been applied to solve some real-world problems. To involve the reader, we try to elaborate the versatile ways of applying GT concepts in different contexts.

In each of the following sections, the introductory discussion about the field is not meant to be a deeper technical description of the subject and why the GT is applicable there. Rather, the objective is to provide surface level information about the particular application scenario of GT for a problem in that field. It will help a novice reader to easily establish a connection

between GT and the problem where it has been applied.

1.5.1 Automatic Well-Testing

Oil wells are monitored on a regular basis to collect data which is in turn used to maximize oil recovery, to forecast the production, and to achieve other similar objectives. A well-testing is a repetitive process carried out twice or more times per month for a well. Typically, an oil sample is taken from a well and tested using certain procedures. Well-testing also helps to identify nonproductive wells for repairing if no fluid is detected. This is a costly process but well-testing is essential in oil field operations.

To avoid human intervention as well as to reduce cost, automatic well-testing systems are deployed. Wells are connected to a central system where testing is performed sequentially, one well at a time by automatic opening and closing of the valves which connect a well to the facility. A preset threshold of minimum quantity of oil is required to flow from the well to the facility to produce reliable results. The time it takes a well to reach this threshold is called the *testing period*. Thus, the wells with low production rates take relatively more time to reach this threshold than those of having high rates.

To the best of the authors' knowledge [45], they are the first to apply the concept of GT to automatic well-testing. Assuming that the setup is capable to divert *multiple* wells simultaneously to the testing facility, the proposed approach does not require any additional manpower or other infrastructure such as any extra hardware resources deployment at the testing facility.

The idea is that input flow will be the *sum* of the production from all the wells connected at one time. Processing the sum of the flow from multiple wells is analogous to a group test. Different combinations of wells are group tested at different times and based on their testing periods, the individual testing period for each well is determined separately which in turn is used to estimate individual production rates. Thus, the overall objective is to reduce the average testing period compared to sequential testing. Other parameters are also taken care of and simulation results show up to 70% reductions in the testing period.

1.5.2 High Betweenness Centrality in Complex Networks

Complex biological systems such as protein-protein interaction databases, networks of human social relationships like Facebook, or computer networks in general, are easily modeled as graphs where the entities of the network

are represented as nodes. Edges in the graph connect nodes corresponding to the interaction partners. Then, rich literature of GT on graphs provides a handful of tools to study various features of these complex networks.

Modeling via a graph makes it easy to analyze various properties of the underlying entities. One such important characteristic called *betweenness centrality* (BC) is central to many social, biological and computer networks. Betweenness centrality is a measure of the role of an entity in the flow of information across the network. In graph terminology, this translates to the total number of shortest paths from all nodes to all other nodes that pass through a certain node v.

Nodes with high BC values are vital in complex networks. In [56], the authors are pioneers in looking into the application of GT to find high BC value nodes in complex networks. Most existing algorithms compute BC values for all the nodes and then choose the highest ones. However, Ufimtsev and Bhowmick [56], using the fact that there are usually a few nodes which exceed certain threshold of BC value and that only the identities of such nodes (entities) are of main interest, have devised a GT strategy for unions of nodes. More precisely, they have applied nonadaptive GT, however the calculation of the group BC value for a collection of nodes is inherently sequential.

Their particular approach guarantees to find top 2 high BC nodes. They have also shown the applicability of their work by conducting experiments on real world networks, see Table 1 in their paper for details. As pointed out by the authors, further research in this direction can reveal more insights to enhance the GT strategy and tackle those instances of graphs which are not covered by the currently proposed solution.

1.5.3 Privacy Leaks

Due to privacy agreements, a social networking website provides restricted access to the information such as friends of a social network's user. Similarly, privacy concerns also exist for other large databases such as Netflix ratings of movies by individual users. However, if a certain type of queries are allowed on these databases, *e.g.*, friendship overlaps for a Facebook user, the response reveals a "little" privacy information contained in a single bit that indicates, for example, a common friend of two users.

A well known characteristic of these databases is sparsity. In [2], Asuncion and Goodrich have argued that using certain sparsity-exploiting queries similar to nonadaptive GT, one can clone all or a large part of such databases by simultaneous flooding with a relatively small number of queries each retrieving a little chunk of information. They call their GT based *data-cloning*

strategy a *nonadaptive mastermind* algorithm.

They define a general query model as follows. Let $\mathcal{X} = (X_1, X_2, ..., X_g)$, be a database where each X_i is a character string of length n taken over an alphabet of size c, which is also fixed for all X_i. Thus, $\mathcal{X}$ consists of g strings (vectors) of size n each. Now, a query Q can be seen as asking for a comparison of Q with each of the X_i. Thus, a query Q (string or vector) to $\mathcal{X}$ receives a vector of responses $(r_1, r_2, ..., r_g)$. Each r_i a score of similarity measure between Q and X_i according to some public reference string R, $|R| = n$.

Using this query model, they have shown that if at least $g' \leq g$ of the strings X_i have at most $d < n$ differences from the reference string R, then at least g' of the strings in $\mathcal{X}$ can be cloned using a maximum of $2(c-1)(2d \log n + \min\{d \log g, d^2 \log(\frac{en}{d})\})$ queries. The unknown number d represent the sparsity parameter. The queries are constructed randomly and asked nonadaptively so that the mastermind attack works in a sneaky manner.

To establish analogies with the GT problem, we first explain the application of this query model to learning friendship ties in a social network. In a practical scenario, n could be the number of people of a city, or students of a college, or employees of a large corporation and g be the number of those individuals whose friendship relations are to be learned. Let $c = 2$ and $\{0, 1\}$ be the allowed alphabets. Thus, the database $\mathcal{X}$ represents a $g \times n$ binary adjacency matrix; each column represents an individual, and each $X_i \in \mathcal{X}$ is a row which encodes friendship for an individual (1:friends, 0: not friends). Since not everyone in a town or an organization is friend of every other, it is realistic to assume that the matrix $\mathcal{X}$ is at most d-sparse (sum of 1s in a row $\leq d$) for some $d < n$. In other words, here it is assumed that the number of friends of each of the g individuals is upper bounded by the number d.

Query vectors correspond to a relatively small number of pseudonyms that the attacker creates in the social network and defines a certain number of random friendship ties. There are studies showing that users are likely to accept random friendship requests from the people in their community *e.g.*, belonging to the same city or university, or affiliated to the same business, job, hobbies etc. Using each of his pseudonyms, the attacker asks the social networking site whether this pseudonym has any friends in common to those of g. For everyone that has allowed for testing friends in common, the attacker will receive some useful information. Nonadaptively asking such apparently irrelevant queries, which are defined before for each of the pseudonyms, the attacker can target the privacy of any user in the network. Formally, a query Q is a binary string and in response the attacker receives a binary response vector $(r_1, r_2, ..., r_g)$. Response vector is a result of bitwise AND, that is, each

r_i is 1 if and only if there is a bit position that is 1 (a friend in common) in both X_i and Q. When they don't share at least one friend, $r_i = 0$, which is implied by $X_i \wedge Q = 0$, where $\wedge$ denotes bitwise AND.

Now, the situation can be easily seen as a variant of classical GT: Given a collection $\mathcal{X}$ of g sets consisting of n elements each. The sets X_i are not necessarily distinct and each has at most d_i defectives. The objective is to detect all the defectives in entire collections of the sets using a minimum number of nonadaptive queries. In [2] the authors have reduced it to the construction of disjunct matrix which satisfies their constraint. Actually, they have relaxed the condition for disjunctness and used randomized constructions for nonadaptive queries.

They performed empirical analysis of the proposed GT based attacking strategy on various real-world databases. As case studies they have applied it to friendship discovery in subsets of the Facebook and LiveJournal networks. They have also implemented their idea on a power grid network, and the database of movie-ratings vectors provided for the Netflix Prize contest. They have concluded that large portions of these data sets can be replicated using a number of queries that is much smaller than the length of the vectors in these databases. The authors also employ some problem-specific heuristics to further improve their results. For instance, in case of the LiveJournal database (where each vector has 5 million entries), using only 300 queries, they have reported to clone half of the database.

Network security generally deals with threats to a network from the outside, whereas, a network monitoring system monitors different components of a network such as servers, network connections, etc., and attempts to identify problems caused by these internal resources of the network. Both network security and monitoring are the key activities of a *network management system*.

Recently a complete book has been devoted to the applications of GT in the field of network security [52]. Detailed GT based models which are applicable in a variety of network security related issues has been mentioned and new challenges are highlighted. The focused scenarios are Denial-of-Service attacks, reactive jamming attacks in wireless sensor networks and randomized fault-tolerant GT applications in advanced security. Discussions below do not necessarily follow entirely from the book. They are written independently of the book, based on the literature related to both network security and monitoring. However, when necessary, the corresponding chapter of this book is cited.

1.5.4 Denial-of-Service Attack

Denial-of-Service (DoS) attack is a term related to Internet security which usually refers to a situation when an intended user is prohibited to access a particular website or service. Denial of service could be temporary or indefinite. Web servers of banks or credit card processing gateways are among the typical targets of DoS attacks.

In his PhD thesis [37], Khattab has employed GT combined with another algorithm which he calls *live baiting* to propose a complete defense system against DoS attacks. Live baiting is claimed to be the first work applying GT theory to the DoS attack problem. In the language of GT, all the service clients represent the collection of n objects, and an attacker who hides himself among the clients is considered as the defective whose identity is to be revealed. The service capacity of the server is divided into several virtual servers called buckets. Each client can access only a subset (limited capacity) of these buckets. Nonadaptive GT is used where buckets being the pools, and a client is considered in a pool (bucket) if it sends access requests to this bucket. Thus, to design nonadaptive pools, they construct a d-disjunct matrix as follows: Rows are the buckets and the columns are the clients. Each client is issued securely generated tokens which can be used to access those buckets only which are legal for this client. This models at which places of the disjunct matrix there are 1 entries. Now, constructing a d-disjunct matrix helps to decide the set of buckets assigned to each client in order to identify at most d attackers. Requests sent from clients to the buckets are monitored from time to time. Whenever the number of requests to a bucket is larger than the predetermined threshold, the outcome of this pool (bucket) is considered positive, otherwise negative. Whenever there are some positive buckets found, the decoding algorithm for d-disjunct matrices localizes the attacker. His study is backed by theoretical analysis of the proposed strategy. He also simulated his defense mechanism and ran prototype experiments.

While a traditional DoS attacker usually targets the network bandwidth, in application level DoS attacks, the flaws in the design and implementation of the targeted application are used as a means of carrying out unauthorized activities. Due to the nature of these attacks, traditional solutions deployed for DoS attacks do not work here. Therefore, application DoS attacks have emerged as an increased threat to Internet security.

In [53] a novel GT based approach has been proposed to counter application DoS attacks. The proposed framework runs on back end servers. The basic model is somewhat similar to the one explained above, however, they extend the basic nonadaptive GT model to take care of some of the constrains originated in this application. Theoretical analysis shows that

they can obtain short detection delay and low false positive/negative rate. Furthermore, apparently they are the first to use GT for application level DoS attacks. Their solution also provides a basic underlying framework to deal with general network attacks. Preliminary simulations results highlight efficiency and usefulness of the proposed scheme in real situations. Mainly, they have used nonadaptive GT to design an attacker and victim detection model. We refer to Sections 2 and 3 of [53] for details. They have mentioned in the conclusion that efficient constructions of d-disjunct matrices can lead to improved performance of their detection system.

Chapter 2 of the book [52] presents a comprehensive GT based model for DoS attacks.

1.5.5 Safety in Vehicular Ad-Hoc Networks

An Ad-Hoc network is a wireless network which unlike the wired or managed wireless networks, does not rely on dedicated devices such as routers or access points. Instead, nodes can join or leave the network freely and while in the network, each node participates in the dynamic message passing regarding, *e.g.*, routing, safety information exchange etc. Being unmanaged, Ad-Hoc networks can use flooding for exchanging data among the nodes. Flooding mechanism produces far more network traffic compared to routing in the managed networks.

VANET is a specialized version of Ad-Hoc networks used in modern transportation systems for efficient vehicle communication. On the heart of all tasks carried out by a vehicle in the network is the safety. For this purpose, each vehicle sends a safety related message every 300 ms [61]. The key element of every message is the vehicle's signature which makes it possible for other vehicles to maintain trust. Only one signature is received at any time through the network transmission system, while others are buffered at receivers for sequential verification. In this one at a time verification protocol, unauthorized signature from an attacher is easily recognized. However, depending upon the number of vehicles in a VANET at any time, it may become the processing bottleneck at each vehicle. Therefore, many useful safely related messages are discarded because of processing time limitations.

An immediate solution is to process the signature checks in batches. In [61], they proposed an identity-based batch signature verification scheme for VANETs. If there is no invalid identity, a batch verification succeeds, otherwise it fails. Upon a failure, all identities are tested individually. In the absence of attacker, or very rare attacks, a tremendous gain in the processing time of the signature verification can be achieved. But, a clever attacker can periodically send invalid signatures, thus causing the vehicles to do individual

verification most of the time.

To overcome this, they [61] considered GT with an objective of finding invalid signature (defectives) with a minimum number of batch verifications (group tests). Precisely, upon a failure in their basic setting, they consider it as a GT problem of identifying at least one invalid signature out of n signatures in the batch. Actually, there could be more than one attackers trying unauthorized access, or same attackers is producing multiple invalid signatures, an upper bound d on the number of invalid signatures is considered and an adaptive GT strategy is used. Using some information from the behavior and structure of the underlying network, they estimate an upper bound for d. Within a batch, invalid signatures are usually far less than the valid ones, thus the idea of GT has helped to significantly reduce the processing time of several identity verifications. In particular, they have used *generalized binary splitting algorithm* from [34].

Recently, in a 2013 paper [39], the batch verification scheme has been further tuned up for practical usage with some enhanced features concerning security and efficiency.

1.5.6 Network Tomography

In simplest words, network tomography refers to monitoring the health of various links in a network. Health of a link is analyzed based on characteristics like delay or loss rate of the link. For this purpose, some special packets called probes are sent out between end-hosts to take measurements. End-hosts are used because accessing internal nodes of a network is not always possible. Now, if a link is experiencing heavy traffic, end-to-end probe passing through this link will take time longer than expected, thus hinting towards a possible congestion in the network along that route. However, from this delay information, it is not always feasible to uniquely identify which link is actually congested. This issue is known as *identifiability* among the research community in this area.

The problem becomes a good candidate to apply GT because number d of the congested links are usually very small compared to the total n links in the networks. Information from end-to-end probes can be interpreted as group tests with outcomes, "good route" and "bad route". Since sending a probe increases network overhead, and knowing that there is already congestion ($d \geq 1$), using a minimum number of probes to identify the congested links is highly desirable.

In [3], a deterministic two-stage approach for network monitoring is proposed and the problem of identifiability is solved using GT. Their two stage approach should not be confused with the 2-stage GT. Here, they use the

first stage to discover whether a congestion has occurred or not. When congestion is confirmed, stage 2 takes over in which the congested nodes are individually identified. Let there be n links in the network and there exist m pairs of end-nodes. Each pair of the end-nodes uses a predetermined route passing through some links. In stage 1, a probe passing through largest number of links is selected first and sent over. This process continues until a "bad route" is found. Now, stage 2 chooses probes based on a simple adaptive GT strategy using binary splitting.

This is a long paper addressing both theoretical and practical issues. Furthermore, simulation in OPNET and experiments on the PlanetLab testbed highlight the advantage of the proposed approach against the conventional algorithm.

1.5.7 Fault Diagnosis in Optical Networks

Failure detection, isolation and correction are the key features of network management. The objective is to efficiently detect failure using different probing mechanism. A typical example of a Synchronous Optical Network (SONET) is that a probe signal is sent out to inquire about the state of health of the network. Based on the inquiry outcome of several sequentially sent out probes, a failure pattern is identified. Usually, sequential probing incurs more time and hence increased cost. With an objective of keeping the fault diagnosis a low cost solution in optical networks, nonadaptive GT has been used by [32] to develop new parallel probing schemes. The main design objective is to minimize the number of parallel probes thereby reducing the size of the GT instance to be solved nonadaptively.

A general approach using the concept of GT along with some other novel ideas has been proposed in [59] for multi-link failure localization in all kinds of optical networks.

1.5.8 Fault and Failure Detection in Wireless Sensor Networks

A sensor is a device which monitors values of certain parameters related to its environment and delivers this data in the form of electrical or optical signals to a central system. Wireless sensor networks (WSNs) are composed of several such sensors deployed physically at a certain location. Originally motivated by military applications, now besides others, WSNs are used in monitoring air and water quality, fire detection and land sliding control in forests, health monitoring applications and vehicle tracking.

The integrity and accuracy of the information provided by a WSN depends directly on the data sent by the sensors in the network. Quality of the hardware components used in a sensor and/or harshness of the physical environment sometimes results in improper functioning (faulty signals but sensor is alive) or in extreme cases complete failure (no signal at all) in some sensors. Due to the nature of the applications of WSNs, fault and failure detection of sensors in WSNs is very critical and a topic of active research in recent years.

Here, we refer to two papers [41, 55] addressing fault detection in a WSN. The beauty of both the approaches lies in a novel combination of GT with signal processing methods to come up with low cost and efficient solutions.

As evident from the complexity comparison of different fault detection algorithms in Table 1 of [41], a WSN of n nodes (sensors) requires at least $O(n)$ measurements (tests), i.e, one per sensor, to identify one faulty sensor. In the absence of any disaster in the WSN deployment environment, faults rarely occur under normal conditions. Using Kalman filtering technique from signal processing to *group test* a subset of sensors, nonadaptive GT is used in the proposed solution [41] to reveal the fault state of all sensors in the network with high probability. Using real data from a WSN consisting of 18 sensors 2 of which are faulty, they have compared their proposed strategy with other existing methods and showed that they achieve similar or better detection accuracy and improved computational complexity (fewer tests).

In [55], they have solved the same problem, but instead of a centralized fault detection approach, their main contribution is the combination of GT with some gossip mechanism to come up with a distributed detection of one or more defective sensors. The idea is based on the observation that measurements of a defective sensor varies significantly from its close neighbors. A specially designed probabilistic message passing protocol under which data (readings or measurements) exchanged among a group of neighboring sensors is gathered by a *master sensor* and this information is handled in a distributed manner to perform failure detection using GT. Hence, the objective is to minimize the messages exchanged by the sensors to successfully localize the defective ones. Developing a bound on the minimum number of messages required for the detection of one defective sensor with high probability, they extend the study to multiple defective sensors detection. Simulation results show that the proposed GT based approach is better than existing methods.

In [31], the concept of GT is applied to dead (failure) sensor detection in mobile ad hoc wireless networks. The failure detection is more critical than faulty sensor detection because a dead sensor simply becomes invisible to the whole network. The situation becomes worse for mobile ad hoc networks as the nodes are allowed to move, *i.e.*, there are no fixed neighbors for any fixed

node. Their GT approach is discussed in detail in the next section.

Interestingly, in a recent study [62], GT has also been applied to efficiently improve the neighbor discovery problem in wireless networks.

1.5.9 Conflict Resolution in Multiple Access Channels

Goodrich and Hirschberg [31] have considered a variant of the classic GT by allowing a group test to produce a *ternary* outcome: (a) *pure* implying no defective, (b) *tainted* if the group contains exactly one defective and (c) *impure* depicting the presence of two or more defectives. In other words, if there is no or only a single defective, the group test reveals the number, otherwise reports the presence of at least two defectives. In case of tainted outcome, they even allow the result to actually identify the defective which they call *identifying* tests, compared to the *anonymous* tests when just presence of a defective is confirmed.

The study is motivated by the nature of the situation that arises during conflict resolution in a multiple access channel (MAC) of a wireless network. A MAC represents any device in the wireless network which is shared among others in the network. Here is a typical example from [31], "when a collection of wireless devices compete to communicate with a particular access point, the access point becomes a multiple access channel (MAC), which requires a conflict-resolution method to allow all devices to send their packets in a timely manner." They establish equivalence between their specific *ternary-result* GT model and conflict resolution in multiple access channels (MACs) and present both randomized and deterministic algorithms based on adaptive testing.

Let n be the total number of devices in the network and d of them are allowed to share a MAC on one-at-a-time basis, independent of each other and without knowing beforehand who else is going to access the channel in the next time interval. Trying to gain access to transmit on this shared MAC corresponds to a group test in the context of the *ternary-result* GT model.

- At a particular time, if only one of these d devices is accessing this MAC, it gains access and the remaining $d - 1$ are informed about its identity (tainted outcome).

- If no one is attempting access, this information is also made available to the entire group (pure outcome).

- A conflict occurs when 2 or more devices from this group try to gain access (impure outcome).

Upon a conflict, nobody is allowed to access and all d devices are informed that a conflict has occurred and it is time to run a conflict-resolution algorithm. Similar rounds of retrying to gain access of the MAC are repeated until the conflict is resolved and one device is allowed to use the MAC.

The best results from the literature as mentioned by [31] require $2.14d$ expected tests (packets sent on the network) for known d and $2.25d+O(\log d)$ for the case of unknown d. The constant factor in the leading term is important in this context because it corresponds to the throughput of a MAC.

The proposed *ternary-result* GT based strategy achieves expected $2.054d$ tests for a known d and $2.08d + O(\log d)$ tests if d is not known.

1.6 Scope of the Thesis

Usually a GT strategy assumes that the query outcomes are *error-free*, *i.e.*, there are no testing errors and whatever (positive/negative) result we observe is always correct. Testing error is mostly problem-specific, *e.g.*, in some biology applications test outcomes are not always correct.

Another source of error called the *design error* can affect the outcome of a GT strategy. Design error is basically due to the nature of the GT strategy itself when it fails to identify all defective elements.

A strategy can be based on *deterministic* or *randomized* constructions of pools. While deterministic strategies do not have a design error, randomized strategies sometimes allow for a small design error. Obviously both deterministic and randomized strategies aim at minimizing the number of tests, however, in some cases randomization can achieve better complexity bounds, *e.g.*, if the output is allowed to be incorrect with a small prescribed error probability. For example, as mentioned in Section 1.4.2, at least $\Omega(\frac{d^2}{\log d}\log n)$ queries are required by a deterministic 1-stage strategy. However, a randomized 1-stage strategy which needs asymptotically only $1.45d\log n$ queries to identify up to d defectives for small fixed error probability ϵ is reported in [10].

Besides some results on deterministic strategies, the first three papers in the contributions list are mainly based on randomized constructions. The focus is to study multistage competitive GT strategies with the objective of minimizing the number of stages and at the same time aiming at the query number close to the information-theoretic lower bound. Next two papers consider optimal strict GT strategies for given n, d and number s of stages. In the context of multistage strategies, as part of Paper-V, we also study the query complexity for GT with disjoint simultaneous pools; a situation which arises when the elements are indivisible objects and at one time, an element can belong to at most one pool.

CHAPTER 2

Contributions Overview and Future Extensions

The thesis is based on the publications appended in Part-II. Each paper addresses certain aspects of the group testing problem and adds new results along that direction. The purpose of this chapter is to collectively put the key ideas from our papers in a broader perspective.

In the beginning of this chapter, we motivate the need in our direction of work by presenting some key research questions. Then, we explain the extent to which our contributions can answer these questions. Going beyond just providing the list of important outcomes of our research work, we have also included simple and easily comprehensible elaborations of the underlying techniques. The objective is that an interested reader can get overview of the contributions without digging deeper into the technicalities. It also hints towards the results where the author of this thesis has substantial contributions. Later in the end, we present some pointers to pursue further research.

2.1 Motivation and Key Research Questions

In Section 1.4.1 of the previous chapter we have discussed competitive GT strategies. Just to refresh, such strategies work for unknown d and at the same time require a query number within a constant factor of the optimum. However, the best results in this direction [23, 47] are based on adaptive GT, whereas in many applications of GT the time consuming nature of adaptive strategies is hardly acceptable.

Then in the quest for minimum adaptivity, we have 2-stage strategies which are also more favorable in these situations. As explained in Section 1.4.3, these strategies require that the searcher must know d, or some close upper bound on d, in advance. Usually in GT applications, d is unknown and some large enough d is assumed as upper bound. Such a 2-stage strategy guarantees an almost optimal query complexity (within constant factor) relative to this assumed d only. Although, guessing the upper bound apparently solves the problem, but in many situations nothing might be known *a priori* to make an educated guess. This leaves in a state of making a "free choice" about d which may be too small (large) thus leading the strategy to end up in chaos (unnecessarily many tests).

Taking one step further towards optimality, a more natural and practical scenario is when one has a fixed problem size n, and given the values for d and s, the objective is to determine the worst-case test number required by an optimal GT strategy. However, usually the strategies claiming logarithmic query complexity exhibit optimality for very large n. Furthermore, since there is no restriction on the group size, in theory a strategy may ask for a group test on any collection of elements. However, for applications, *e.g.*, in chemical and blood testing, due to dilution effects and/or limitations of the apparatus used for testing, a group test cannot be run beyond certain limits. Thus, for these and other similar situations, optimal GT strategies for a given problem instance should be studied.

Usually, in a GT strategy the pools queried in parallel are highly overlapping. The assumption behind, as mentioned in Section 1.3.1, is that each element can participate and be tested in several pools simultaneously. Sometimes, overlapping may not be possible, *e.g.*, when the elements to test cannot be divided physically (electronic devices), or the samples of the elements are costly, or we have only a limited amount of samples per element. This implies that in a situation when the elements cannot occur in more than one pool at the same time, a GT strategy with few stages and logarithmic test complexity but relying on overlapping pools within a stage cannot be used at all.

Concerning the above points, it is worthwhile to ponder over the following questions:

1. *Can competitive GT strategies still reach $O(d \log n)$ queries if only a constant number of stages are allowed?*

   That is, exploring the possibility of performing GT under the constraints: (a) without *a priori* knowledge of d, (b) in a few stages, and (c) using a number of pools close to the information-theoretic lower bound.

2. *Under what conditions can a GT strategy be both query-optimal and minimal-adaptive?*

   Here the focus is on designing GT strategies which, at least asymptotically, require only $d \log n$ queries and at the same time run in a small number s of stages. Regarding the defective elements, there are different possibilities: (a) a known value of d, (b) d is completely unknown, (c) constant defective rate r, and (d) d grows slowly with n. Given the initial information about the defectives, the objective is to study the possibility of query optimal strategies using a number of stages as few as possible.

3. *What is the optimal worst-case test number for given n, d and s?*

   The study in this direction requires development of combinatorial tools which will enable the searcher to design an optimal GT strategy for a particular problem instance.

4. *If the pools must be disjoint, what are the complexity bounds and how can a GT strategy with disjoint pools achieve it?*

   Here, the key is to figure out the characteristics of a GT strategy when, within a stage, overlapping pools are not allowed. The points to be considered include: What is the effect of number of stages in this context? How does the growth of d versus n affect the strategy to achieve the lower bound?

2.2 Results at a Glance

Now, we highlight our main results addressing the research questions set up in the previous section.

2.2.1 Competitive Group Testing in Only 2 or 3 Stages

We start with the first and foremost focus of our research. Apparently, we were the first to study this combination of demands, *i.e.*, competitive and minimal-adaptive GT strategies. We distinguish between deterministic and randomized strategies and separately state the two versions of the problem as follows:

- *Can we achieve competitive GT that insists on $O(d \log n)$ pools in a constant number of stages using deterministic strategies?*

  Unfortunately the answer is No. There cannot be a deterministic competitive GT strategy that succeeds in a constant number of stages to achieve $O(d \log n)$ queries. In Paper-I it is proved that a strategy with the above demands would require $\Omega\left(\frac{\log d}{\log \log d}\right)$ stages.

- *Do there exist randomized constructions for previously unknown d that work in a few stages of parallel queries and closely achieve the optimal query bounds?*

  Still the answer is No! This is because the proof in Paper-I is more general and also extends to randomized constructions with strict demands on stages and query number.

Then, a natural idea is that a strategy with such set of demands should start by estimating the magnitude of defectives in the given problem instance. Hence, to answer the research question 1, we can follow a two steps procedure:

- **Step-1:** Use nonadaptive queries and determine d; exactly or an upper bound.

- **Step-2:** Using d from step-1, apply the best known 1- or 2-stage GT strategy.

While step-2 just involves selecting an appropriate strategy from the already established results, finding the number but not the identities of defective elements using nonadaptive queries is the actual challenge.

Although there has been several studies under the statistical model of GT focused on approximating the proportion of defective individuals in a

population, to the best of our knowledge we are the first to answer the following question:

Can we estimate a previously unknown d using nonadaptive queries?

Once again our initial result is negative, *i.e.*, determining d exactly would be as hard as the GT problem itself. Hence, the known lower bounds for the GT carry over to this seemingly "simpler" problem. Thus, it would require $\Omega(\frac{d^2}{\log d}\log n)$ nonadaptive queries.

Our next choice is to try randomization and hope for a nonadaptive strategy to estimate a close and reliable upper bound for d. This time, allowing a small fixed error probability, we succeeded with randomization.

1-stage Randomized Estimator

For curious readers we briefly outline our 1-stage estimator. To prepare a pool we fix some probability q and put every element in the pool with probability $1-q$. The pool is negative with probability q^d, since this is the probability that d defectives are outside the pool. We increment $1-q$ in small steps such that we prepare $O(\log n)$ random pools of exponentially growing size and then query them simultaneously. The query results are independent because we put elements independently in each pool. Now, pools of sizes smaller than $\frac{n}{d}$ will most probably be negative. Similarly those having sizes larger than $\frac{n}{d}$ will most probably be positive. Thus, such pools convey very little information about the magnitude of d. However, the cut-off point between negative and positive pools can be used to estimate d. Actually, we have shown that using $O(\log n)$ randomized nonadaptive queries and allowing for a small prescribed failure probability ϵ, a "conservative bound" $\hat{d}$ can be learned such that with probability $1-\epsilon$, the expected ratio $\frac{\hat{d}}{d}$ is bounded by some constant expected factor c, independently of d. Results presented so far are based on Paper-I.

Outline of our nonadaptive randomized estimator makes it very clear that considered demands on ϵ and the ratio c can not be achieved using the maximum likelihood estimation technique which is conventionally applied to estimate the probability of each element being defective.

Related to the proposed randomized strategy, the next obvious question can be formulated as follows:

Do we really need $O(\log n)$ pools to find an upper bound for d?

Main result in this direction is that a lower bound of $\Omega(\log n)$ pools is proved in Paper-II, for our particular but very natural way of choosing randomized pools for the 1-stage estimator.

Optimal Competitive Ratio

Let g be some constant and suppose that we use $L = g \log n$ nonadaptive queries to get the estimate $\hat{d}$ with expected factor $c = \frac{\hat{d}}{d}$ for a given error probability ϵ. As discussed in Section 1.4.3, there are 2-stage GT strategies where $c'd \log n$ queries are sufficient for known d. Just to remind, for the best result in this direction, asymptotically the constant factor is $c' = 1.44$ [10].

Now, for unknown d we can use our randomized estimation to get $\hat{d} = cd$ in stage 1, and appending the best known 2-stage strategy afterwards we get a 3-stage competitive group testing strategy with $g \log n + c'cd \log n$ or $(\frac{g}{d} + c'c)d \log n$ expected queries in total. In GT strategies for unknown d, the constant factor in the query number is called the competitive ratio. In our case, the competitive ratio is determined by the factor $\frac{g}{d} + c'c$. For $d = 1$, the competitive ratio becomes $g + c'c$ while asymptotically for growing d we have $c'c$. Since in any case the factor c has direct impact on the total expected query number, an optimal value for c can save many pools.

Apparently, at the cost of a large g, we can make c arbitrarily close to 1. However, the worst-case competitive ratio $g + c'c$ depends on both. Therefore, to minimize the competitive ratio, an optimal strategy should balance the number $g \log n$ of pools and the expected ratio c. Driven by this, our next goal was to achieve optimal values of the constants c and g for given error probability ϵ.

Getting an optimal trade-off turned out to be a highly nontrivial problem in itself. Therefore, we first formalize it as an independent problem of estimating d from the test outcomes with the objective to achieve optimal factors. Then we will present our systematic approach to solve it.

Let us represent a positive test outcome with 1 and a negative test outcome with 0. As discussed above, we fix some q and select each element independently with the same probability $1 - q$. In this way, we characterize each randomized pool by only one number: the probability q_k *not* to put an element in the kth pool. For the given problem size n, we fix some value for g, *e.g.*, $1, 1.5, 2$, *etc.*, and prepare $L = g \log n$ pools using corresponding q_k probabilities where $k = 0, 1, ..., L-1$. Let $s = s_0 \dots s_{L-1}$ be the binary string representing nonadaptive query outcomes for our randomized estimator, the problem can now be defined as:

For already fixed $L = g \log n$ and the probabilities q_k, predict the unknown number $d \in [1, n]$ from the string s of test outcomes such that the expected accuracy $\frac{\hat{d}}{d}$ is minimized, but at the same time $\hat{d} < d$ holds with probability at most ϵ.

The essence of this problem depends on two things. First, how do we define

the sequence of probabilities q_k? Second and most importantly, once the pool sizes are fixed, how do we use the information contained in the result string s to make an efficient guess?

***Ad Hoc* Rules:** As first attempts we tried out some *ad hoc* rules. Suppose that q_{kd} denotes the conditional probability that the kth pool is negative when there are d defectives. Since a pool is negative if and only if none of the defectives is selected, and elements are selected independently, this simply yields $q_{kd} = q_k^d$ and we have $d = \frac{\log q_{kd}}{\log q_k}$. Since q_k is known, if we can estimate $\hat{q}_{kd}$, we can predict the corresponding $\hat{d}$. To do so, we prepare a number of pools for fixed $g = \frac{L}{\log n}$ using the already defined probabilities q_k. Let us index the pools according to increasing size and perform parallel queries to get the result string s. Let i be the largest index of a negative pool. We call it the *main index* and let q_i be the probability with which this pool was prepared. Now, assuming $\hat{q}_{jd} = \frac{1}{2}$, where $j := i - m$ for some previously fixed m that depends on the desired failure probability bound, we can compute our estimate $\hat{d}_j = \frac{1}{\log\left(\frac{1}{q_j}\right)}$. Extensive simulations done in Matlab with independent random choices of d and very large n, suggested that for $0.01 \leq \epsilon \leq 0.05$, asymptotically we have $2.5 \leq c \leq 5$ for $1.7 \leq g \leq 2.7$. Structure of this simple approach does not allow us to observe a single value c, against fixed ϵ. We actually devised different rules to predict a reasonable m, as described in Section 7 of Paper-I.

A Linear Programming Formulation: The index $j := i - m$ represents the pool on which we based our estimation. Choosing the most relevant j actually plays an important role in minimizing the factor c and also observing the given failure probability constraint. Thus, instead of empirically predicting its value, we formulated the problem of minimizing c as a linear programming (LP) optimization problem. For given ϵ and predetermined pool sizes, our LP minimizes the expected upper bound on c by assigning probabilities of choosing every $d \in [1, n]$ against each possible string s of query outcomes (2^L in total). To explore the c vs. g trade-off, for the same ϵ we can increment g at fixed steps to get corresponding c values. For the LP implementation, we used GLPK and run it for different parameter combinations. The results suggest that always some g around 2 gives optimal results. GLPK could not handle a higher number of variables $n2^L$ arising when $n > 32$ and stopped us to go to the limits. For an exact LP formulation and further elaborations, we refer to Section 5 in Paper-II.

In order to calculate the actual limit of c for very large n, we defined a nonlinear constraint optimization problem. The new problem is actually

an "infinite extension" of our 1-stage randomized estimator, but requires an alternative tool to solve it. As mentioned in Paper-II, we can now find the asymptotic value of c for given ϵ and g. For instance, fixing $\epsilon = 0.01$, for very large n, asymptotically we achieved $c = 2.99$ when $g = 2$. Combining our estimate with the best result of $1.44d \log n$ for the 2-stage strategy [10] working for known d, the resulting 3-stage competitive group testing has competitive ratio 6.3 for $d = 1$, while it tends to 4.3 asymptotically for growing d. Although the constant factor for the 2-stage strategy is 1.44, we emphasize that, instead of the actual defective number, here in the query number $1.4d \log n$ the symbol d actually refers to the assumed upper bound. On the contrary, when we combine it with our estimate for d, now in our result, *e.g.*, $4.3d \log n$, symbol d refers to the actual number of defectives. If we instead append a probabilistic 1-stage strategy from [10], which requires $O(d \log n)$ queries and succeeds with high probability, we even get a competitive GT strategy that needs only 2 stages.

It is worth mentioning the interest of other researchers on our problem of estimating d. Soon after the publication of our results, Cheng [11] was the first one who carried over the randomized estimate to adaptive strategies. For given n elements and unknown $d \geq 1$, using $O(d \log d)$ adaptive queries, his randomized strategy which he calls "EXACTd" returns the exact number of defectives with high probability $1 - \frac{1}{d^{\Omega(1)}}$. Cheng, in a later paper along with Xu [13], has also considered to approximate the number of defectives. Again, their adaptive strategy is randomized, allowing for a small failure probability. They have shown that for $0 < \delta < 1$, an estimate $\hat{d}$ can be found with high probability such that it satisfies $(1 - \delta)d < \hat{d} < (1 + \delta)d$ and uses at most $O(\log^2 d + \frac{1}{\delta^2} \log d)$ randomized queries.

Yet another paper by Cheng together with Gou and Zheng [12] where, like ours, they first apply their randomized estimation procedure to find an upper bound on the number of defectives. Later, they use it in their new adaptive algorithm for known d which requires at most $d \log \frac{n}{d} + 2d + 1$ tests. Thus, with a total of $d \log \frac{n}{d} + 2d + O(d^{\frac{2}{3}} \log d)$ queries, their competitive GT strategy succeeds with hight probability $1 - \frac{1}{(2d)^{\Omega(1)}}$.

Although we both solve the competitive GT, we remark that their query complexity is not directly comparable to ours because they use adaptive GT for both estimating d and then finding the defectives. On the other hand, we allow only the first stage for estimation, and subsequently apply a 2-stage strategy to find the defectives. The query number and the constant factors, that we have discussed above in our results, are determined for 3-stage competitive GT.

2.2.2 Query-Optimal and Minimal-Adaptive Strategies

Now, we turn to the next combination of demands, that is, minimal-adaptive strategies where the constant factor in the leading term $d \log n$ is as close as possible to 1. Beginning with the case of known d, we will address all possibilities mentioned in question 2 one by one.

CASE-I: When d is a known upper bound

As discussed earlier, the best known deterministic 2-stage strategy requires $1.44d \log n$ queries asymptotically for known d [10]. Recently, in a lower bound proof [44] for 2-stage strategies which insist on $O(d \log n)$ queries, it is shown that when $d = n^{\delta}$, $\delta < 1$, then the hidden constant factor in the query number is strictly greater than 1. They have actually shown it for the statistical model of GT with independent random defectives, but asymptotically it extends to the combinatorial model as well. Therefore, we cannot have a query-optimal deterministic 2-stage GT strategy.

Likewise the previous section, we hope that using randomization together with some restriction on the growth rate of the defectives, we may succeed to come up with some exciting results. We start with the situation when defectives are rare, that is, an upper bound $d \ll n$ is known beforehand, and investigate the following:

When there are a few defectives while n is huge, can we design a 2-stage randomized GT strategy whose query complexity asymptotically reaches the entropy lower bound?

We have shown that there exists a 2-stage randomized strategy that succeeds with probability $1 - \epsilon$ for known d with a constant factor of 1 in the leading term $d \log n$ subject to some minor terms which depend on d and ϵ only. Theorem 3 in Paper-III provides exact details. The building block of this and the forthcoming results in this section is a novel way of combining standard results. We state those first.

Find ONE defective in One Stage(FODOS): If only one defective is present, it can by found by $\log n + 1$ parallel queries (*mathematical folklore*).

Find ALL defectives in One Stage(FADOS): Using $O(d \log n + d \log \frac{1}{\epsilon})$ parallel queries, at most d defectives can be found with success probability $1 - \epsilon$, due to Theorem 10 in Cheng and Du [10].

Fundamental 2-stage strategy: Now we can quickly sketch our 2-stage randomized strategy.

Given n and d, we divide the elements into a certain number q of disjoint subsets called *cells*. We choose q such that with high probability all cells contain at most one defective. Then, we treat the cells like elements and using the FADOS strategy over these cells (instead of the individual elements) identify up to d positive cells (those having at least one defective). Using $q = \Theta(d^2)$, stage 1 succeeds with probability $1 - \epsilon$ and requires only $O(d \log d) + O(d \log(\frac{1}{\epsilon}))$ queries.

Then, in stage 2, we can find the individual defective elements by applying FODOS to each of the positive cells in parallel. Clearly, stage 2 requires at most $d \log n + d$ queries. Thus asymptotically, *e.g.*, for every fixed d, the query complexity converges to the entropy lower bound. To avoid confusion with the other established 2-stage strategies, in the thesis we call this a *fundamental 2-stage* strategy because it plays a basic role in other cases which we have described below.

The above result is good, however, its applicability is restricted to the situation when d grows slower than any power function of n. Next, we explore whether allowing for another stage enables us to extend the result for d growing as any power function of n.

If d grows like $d = n^\delta$, for some constant exponent $\delta < 1$, can we devise a GT strategy that works in 3 stages and the query complexity asymptotically reaches the entropy lower bound?

The answer is positive! We have shown that there exists randomized 3-stage strategy where the complexity bound converges to the entropy lower bound. The construction allows for a small prescribed failure probability. We refer to Theorem 4 in Paper-III for technical details and the proof. Here is the high level sketch of the construction.

Stage 1: Randomly partition the n elements into d bags of size $\frac{n}{d}$ each. The setting is analogous to a special case of the *balls and bins* problem with $\frac{n}{d}$ bins and d balls. Thus, using standard calculations, with high probability fewer than $\log d$ defectives are present in each bag. We process these bags individually to find out the defectives. In order to optimally figure out the number of queries required for each bag, we divide the bags in two groups namely *sparse*: defectives in the bag are less than $\sqrt{\log d}$, and *dense*: defectives in the bag are more than $\sqrt{\log d}$. This is done based on the fraction of positive outcomes of randomly chosen pools from each bag. For this purpose, a total of $O(d \log \log d \sqrt{\log d})$ tests are used on all bags in stage 1.

Stage 2 and 3: Apply the fundamental 2-stage strategy to all bags in parallel. In stage 1 of the fundamental 2-stage strategy, we treat sparse and dense bags separately and carefully decide on the number q of cells in which the bag elements will be divided. The remainder of this stage works as mentioned in the description of the fundamental 2-stage strategy. Collectively after the first 2 stages, we have partitioned the n elements in up to d subsets consisting of at most $\frac{n}{d}$ candidate elements all together. The defectives are separated (belong to different subsets) with high probability. Thus, the last stage requires $d\log(\frac{n}{d})$ queries to localize the individual defectives.

CASE-II: Without prior knowledge of d

We have discussed in the previous Section 2.2.1 that randomization is necessary for nonadaptive estimate of a previously unknown d. Let g be a nonnegative parameter. We can have a 3-stage randomized GT strategy as follows: Using $g\log n$ pools, find an upper bound $O(d)$ for the unknown d. Here, as already discussed, the hidden constant depends only on the small fixed $g > 0$, and on a prescribed failure probability of underestimating d.

Using the upper bound for defectives from stage 1, once again, the fundamental 2-stage strategy takes the place of stage 2 and 3. The interesting part in this case is how we asymptotically approach the entropy lower bound. Ignoring the lower order terms, the main part of the complexity is: $(d+g)\log n + O(d\log d) = (1+\frac{g}{d})d\log n + O(d\log d)$. Here, the factor $(1+\frac{g}{d})$ can be bounded arbitrarily close to 1 by choosing g small enough uniformly for all d, whereas for every fixed g, it converges to 1 for growing d.

CASE-III: Defectives growing at a constant rate

Here, as opposed to the previous case, we assume that d grows as fast as n. This is actually a special case of the statistical model of GT where $r := \frac{d}{n}$. We again put the same question but in new adapted form:

For a fixed defective rate r, can we have a GT strategy that works in a small fixed number of stages such that the query complexity asymptotically reaches the entropy lower bound?

In Paper-III, we have shown that we can solve this GT problem in 4 stages using $(1+o(1))\log(\frac{1}{r})$ queries per defective. The term $o(1)$ vanishes for $r \to 0$. Actually, we have reviewed the literature for this case and observed that the $o(1)$ term cannot be avoided even for adaptive GT strategies.

For details and other technicalities, we refer to Section 3 of Paper-III. Here, we just want to emphasize that this result do not follow from the

fundamental 2-stage strategy. This is simply because there are lower order terms in the query complexity which are monotone in d. Thus, they do not affect the asymptotic behavior of the strategy because there we assume $d \ll n$ and the defectives grow at a much smaller rate than n. On the contrary, now considering a fixed defective rate r, defectives will grow with the problem size which means we cannot simply ignore terms which grow unbounded with the defective number. Therefore, we need more stages to avoid them.

For unknown r, we can use our defective estimation strategy at the cost of one extra stage.

The strategies mentioned above are easy to follow and simple to implement in a real situation. Acknowledging our particular way of combining standard results to construct efficient strategies, Paper-III got the *best paper award* at SOFSEM 2012.

2.2.3 A Toolbox to Design Exact Strategies

While working on the design of query optimal and minimal adaptive GT strategies, we realized that with the current state of developments in the field, the information-theoretic lower bound is only achievable asymptotically. Inspired by this, we started looking for query-optimal strategies when the problem size is fixed, initially for small values of n. Precisely, we assume that n, d and s are given, and the objective is to figure out an optimal strategy to solve this GT instance as stated in the research question 3.

We consider both hypergeometric and strict GT strategies in this context, however, we are mainly interested in the latter as these strategies ask for stronger demands.

Let $t(n, d, s)$ be the optimal worst-case number of tests needed by an s-stage strict GT strategy where the searcher must also verify that no more than d defectives are present. We write $t(n, d, s+)$ to indicate that further stages do not improve the worst-case test number.

Determining exact $t(n, d, s)$ values have found less attention by the GT research community. In this direction, there are only a few results reported in the literature. As part of Paper-V, we have collected these related results to the best of our knowledge. Together with our contributions, here we present a summary of these results:

$d = 1$ **is completely solved**

- $t(n, 1, 1) = \log n + o(\log_2 n)$ is the smallest k with $\binom{k}{\frac{k}{2}} \geq n$ due to [50].
- $t(n, 1, 2+) = \lceil \log_2 n \rceil + 1$, is a result from Paper-V.

Arbitrary d and $s = 1$

- Exact $t(n, d, 1)$ for all $n \leq 14$ and arbitrary d, and some results for large n are presented in [33].

Arbitrary d and s

Paper-IV, in principle, extends the work of Huang and Hwang [33] from $s = 1$ to multistage strategies. We develop several combinatorial tools to systematically study exact $t(n, d, s)$ for arbitrary d and $s \geq 1$. In this context, we begin with the following question:

Why is there a need to develop a combinatorial toolbox to figure out exact $t(n, d, s)$?

An upper bound is established once a strategy for a specific $t(n, d, s)$ is found. However, the real challenge is to find a matching lower bound.

For example, suppose we find a strategy for $t(10, 2, 2)$ which requires 8 tests. Proving $t(10, 2, 2) \geq 8$ is not an easy task. We emphasis that it cannot be done naively by considering all possible pooling designs in every stage because their number is doubly exponential in n. However, by a branch-and-bound approach, our combinatorial tools enable us to determine many exact $t(n, d, s)$.

Since the objective is to determine exact query numbers, it is extremely important to define some representation protocol which completely characterizes what is "known" to the searcher at any moment during the course of applying a GT strategy. In the following we begin with a summary of our notation and then outline some of the lower bound tools and also present our upper bound strategies.

- *Capturing the searcher's knowledge at any stage of a strict GT strategy.*

  We have developed lower bound tools which rely on the graph-theoretic representation of the searcher's knowledge at any stage of solving a strict GT problem. We say an element is a *candidate* to be defective unless it appears in a negative pool. Similarly we define a *candidate set* to be a set of at most d elements such that it is consistent with all previous test outcomes and represents one possibility for the set of defectives. Candidate elements which are not part of any candidate set are called *dummy elements*. Collectively, all this information can be transformed into a *candidate hypergraph* where the candidate elements are the vertices while the candidate sets represent hyperedges. The dummy elements are the isolated vertices.

Thus, we can depict the searcher's knowledge using a candidate hypergraph. By the definition it follows that a strict GT problem instance is solved if and only if a pooling design ends up with one candidate set only and no dummy elements.

- *Complete characterization of an optimal strategy to nonadaptively solve a strict GT after any stage $s > 1$.*

  Prior to the last stage, we model the current state of the searcher's knowledge using a candidate hypergraph and the first challenge is to answer the following:

  What are the essential features of a strict GT strategy for the last stage?

  As reviewed in the Section 1.4.2, a pooling design using a d-disjunct matrix solves the strict GT in one stage. In Theorem 4 of Paper-IV, we have answered the above question by presenting a generalization of the concept of d-disjunct matrix to arbitrary candidate hypergraphs. We have shown that given a candidate hypergraph, a 1-stage strategy solves strict GT if and only if for each pair of a candidate set C and a candidate element $v \notin C$, there exists a pool which includes v and is disjoint to C.

  Next, the searcher wants to know the exact number of pools required by an optimal strategy satisfying the above constraint:

  How to construct a provably optimal number of nonadaptive pools for a given candidate hypergraph?

  This has been answered in Theorem 5 of Paper-IV. We have proved that solving a 1-stage strict GT problem for a given candidate hypergraph is equivalent to coloring an appropriately defined conflict graph, where the conflicts are specified to account for the constraint detailed in the previous paragraph.

- *For all n, we study $t(n, 2, 2)$ in detail and develop theory around it.*

  Graph coloring being NP-hard does not stop us using this result for specific graphs. We have used it to solve many $t(n, 2, 2)$ exactly. Another interesting result in this direction is that for m to be the smallest integer with $\binom{m}{2} + 3 \geq n$, we have shown $t(n, 2, 2) \leq m + 3$. Using this structure, we have found several exact results, *e.g.*, $m = 5$ gives $t(10, 2, 2) = t(13, 2, 2) = 8$, $m = 6$ gives $t(15, 2, 2) = t(18, 2, 2) = 9$, and $m = 7$ gives $t(22, 2, 2) = t(24, 2, 2) = 10$.

Table 2.1: Exact $t(n, d, s)$ for $n \leq 10$ and all d, s.

n	d	s	$t(n, d, s)$	Lower bound
3	2	1+	3	3
4	2	1+	4	4
5	2	1+	5	5
6	2	1+	6	5
7	2	2	7	5
7	2	3+	6	5
7	3	1+	7	7
8	2	2+	7	6
8	3	1+	8	7
9	2	2+	7	6
9	3	1+	9	8
10	2	2	8	6
10	2	3+	7	6
10	3	3+	9	8
10	4	1+	10	9

Overall, for $t(n, 2, 2)$, it is shown that $t(n, 2, 2) \leq 2.5 \log n + o(\log n)$, whereas the information-theoretic lower bound is $t(n, 2, 2) \geq 2 \log n - 1$.

▶ *Proof of concept.*

Using our toolbox, we have successfully devised strategies which use an optimal number of pools. Multistage strategies allow us to save few tests compared to the case of $s = 1$.

For example, according to [33], $t(10, 2, 1) = 9$, whereas we have shown $t(10, 2, 2) = 8$ and $t(10, 2, 3) = 7$. We also prove that $t(10, 2, 3+) = 7$, *i.e.*, the worst-case test number remains 7 even for $s > 3$. Some results are collected in Table 2.1.

2.2.4 Disjoint Simultaneous Pools

Addressing the last research question, we study the problem of GT with disjoint simultaneous pools (DSPs) expressed in terms of problem parameters n,d and s. Here, the role of s is important because overlapping pools are not allowed. In this way, s is also an implicit upper bound on the number of samples required for every element.

In this direction, we start with an old result by [40] which provides an upper bound of $sd(\frac{n}{d})^{\frac{1}{s}}$ tests for a multistage strategy with DSPs. One immediate reflection is to consider the following:

What is the lower bound and does there exist any pooling design to achieve it?

We consider a constant number s of stages and prove an almost matching lower bound that holds at least if $d = o(n^{\frac{1}{s}})$.

Once again the main building block is the case of $d = 1$ defective. We first outline the lower bound for $d = 1$ and use it to extend our reasoning regarding the lower bound.

A lower bound for $d = 1$ using DSPs

Stage 1 to $s-1$: In stage 1, the searcher partitions the candidate elements into a certain number p_1 of *parts*. One arbitrary part which we call *rest* is kept aside while each of the remaining $p_1 - 1$ parts is a pool. These pools are queried simultaneously. If a positive pool is found, carry over its elements to the next stage. Otherwise, proceed with the elements in the rest; the defective element is surely present there because $d = 1$.

Later stages until stage $s - 1$ also behave the same way as stage 1: In stage s_i divide the elements into p_i parts and query $p_i - 1$ of them.

Last stage and the lower bound: At any stage, the adversary wants to maximize the number of candidate elements for the later stages. Therefore, the defective element must hide in the part with the largest size. To minimize the number of candidates after any stage s_i, the searcher divides the elements into p_i parts of equal sizes. Thus, after stage 1, there will be $\frac{n}{p_1}$ candidate elements for stage 2. Similarly, after stage $s - 1$, we will have $\frac{n}{p_1 \dots p_{s-1}}$ candidate elements for individual testing in the final stage. In this way, the last stage requires $\frac{n}{p_1 \dots p_{s-1}} - 1$ tests to separate the defective. Collectively, the query number for all the stages is given by the following expression:

$$(p_1 - 1) + \dots + (p_{s-1} - 1) + \frac{n}{p_1 \dots p_{s-1}} - 1 = p_1 + \dots + p_{s-1} + \frac{n}{p_1 \dots p_{s-1}} - s.$$

Now, again by the same argument as above, to minimize the maximum number of queries at any stage i, all p_i should be equal to some number, say p. The above expression now becomes: $(s-1)q + \frac{n}{q^{s-1}} - s$. Differentiating with respect to p and minimizing, we get $p = n^{\frac{1}{s}}$ giving the lower bound of $sn^{\frac{1}{s}} - s$ tests.

An s-stage GT strategy is called *nested* if any pool in any stage $s > 1$ is a proper subset of a positive pool (or the rest) from the previous stage.

Deterministically partitioning the candidates into $n^{\frac{1}{s}}$ parts of equal size in every stage, a nested strategy can find a defective using the same number of queries as suggested by the above lower bound.

Now, for general d, one may think to start with a partition of the candidate elements into d sets of size $\frac{n}{d}$ each. Then, apply the nested strategy for $d = 1$ independently to all these sets in parallel to prove a matching lower bound. However, coming up with a nested strategy using $sd(\frac{n}{d})^{\frac{1}{s}}$ pools is not enough to prove the asymptotic optimality. Because, the nested strategy explicitly relies on the fact that the searcher processes the d sets independently. Thus, it is just an upper bound and similar reasoning is used by Li [40] when he proved the upper bound long ago.

Our contribution is that we systematically study the problem to prove that the upper bound by the nested strategy is already nearly optimal. We have shown that a deviation from the nested strategy cannot save many tests. We present our results for both hypergeometric and strict GT. For technical details we refer to Paper-V, Section 3, because even the high-level ideas of the proofs are too sophisticated for a summary.

Competitive GT using DSPs

Once again we consider the case of unknown d, this time under the restriction of DSPs. Likewise Section 2.2.1, we use stage 1 for a randomized estimate of d and then the estimate is used in a nested strategy for the remaining $s - 1$ stages.

Randomized Estimate of Defective Ratio $(\frac{d}{n})$ using DSPs

High level idea of the strategy proposed in Section 3.5 of Paper-V is as follows: We apply a random permutation to the n elements and then partition them into $p = o(n^{\frac{1}{s}})$ pools of slowly growing sizes. Since the pools must be disjoint, the elements in all p pools should sum up to n. We achieve this by choosing the pools sizes $|p_i| = i^{s-1+\epsilon}$, where $\epsilon > 0$ is arbitrarily small. Since $n \approx \sum_{i=1}^{p} i^{s-1+\epsilon} \approx \frac{p^{s+\epsilon}}{s}$, we get $p = o(n^{\frac{1}{s}})$ as said above.

If X is the set of random permutation of n elements. Starting from p_1, each p_i is assigned $|p_i|$ consecutive numbers from the elements of X. Small (large) pools are most probably negative (positive). We can fix some prefix where only a few of them are positive while most of the remaining pools are positive. Among the positive pools in the prefix, with high probability, each contains one defective. Counting the positive pools and dividing by the number of elements in the prefix, we estimate $(\frac{d}{n})$.

Numerical Results for Competitive Ratio: In order to experimentally evaluate the performance of this strategy, we did some simulations using Matlab. Below we outline the setup and present the results.

For fixed n and s, we prepare p pools as reviewed above. Now, query all the pools in parallel. Let *posPools* be a structure containing the size of every positive pool and their total count. The structure *posPools* is now passed to the Algorithm 1.

For the graphs in Figure 2.1, we fix $s = 4$ and choose 30 different values $d = cn^{\frac{1}{s}}$ by iterating c from $1, ..., 30$, for each $n \in \{5000, 10000, 25000, 50000\}$. For every pair (n, d), we randomly select a set P of d numbers in the range $1, ..., n$. Now, we proceed as detailed above and by passing *posPools* to Algorithm 1, we get an estimate for $\frac{d}{n}$. We actually repeat the estimate 200 times for every pair of (n, d) and take the average which is drawn against the real $\frac{d}{n}$ in Figure 2.1. Thus, for 4 different choices of n, we get 30 such values for each. In the simulations, p is roughly $3n^{\frac{1}{4}}$ and $\epsilon \approxeq 0.9$.

Algorithm 1 Estimate $(\frac{d}{n})$ by applying an averaging rule to a selection of positive pools

```
1: procedure EstimateDefectivesRatio(posPools)
2:     posMed ← ComputeMedian(posPools)  ▷ The median value among the sizes of all positive pools.
3:     posAvg ← ComputeMean(posPools.size ≤ posMed)  ▷ The mean value of all positive pools whose sizes ≤ posMed
4:     dCounter ← 0
5:     sumPosPools ← 0
6:     for i = 1 to posPools.length do
7:         if posPools(i).size ≤ posAvg then
8:             dCounter ← dCounter + 1
9:             sumPosPools ← sumPosPools + posPools(i).size
10:        else if (posPools(i).size > posAvg) &&
11:                (posPools.size(i) ≤ posMed) then
12:            dCounter ← dCounter + posPools(i).size / posMed
13:            sumPosPools ← sumPosPools + posPools(i).size
14:        end if
15:    end for
16:    dbynPosPoolsAvg ← dCounter / sumPosPools
17:    return dbynPosPoolsAvg
18: end procedure
```

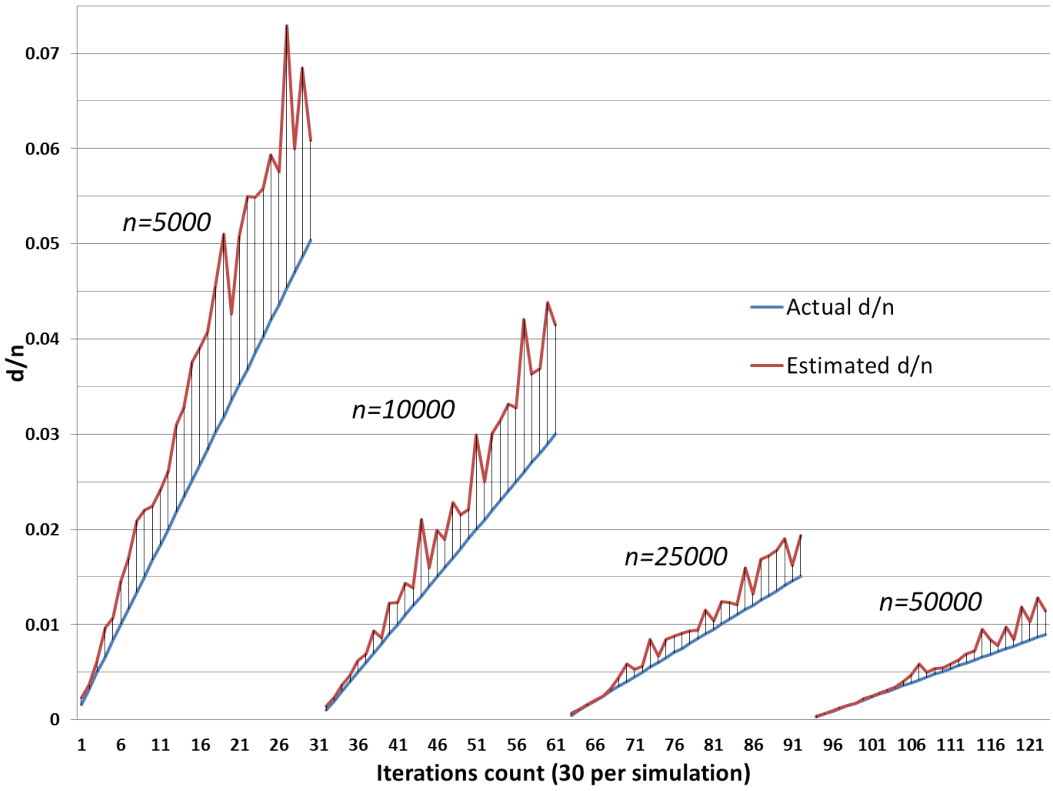

Figure 2.1: Actual vs. estimated defective ratio ($\frac{d}{n}$) using DSPs for different growth rates of the ratio.

A quick look at Figure 2.1 reveals that overall we have achieved good competitive ratios. In the worst case, *i.e.*, when $\frac{d}{n}$ grows very quickly, the average competitive ratio is 1.4. Further, when n is large ($\frac{d}{n}$ incrementing in small steps) the average competitive ratio decreases. Actually, a linear decrease of at least 0.1 is noticed in the average competitive ratio for the second and the third graph (left to right) in Figure 2.1. The average competitive ratio for the rightmost graph in Figure 2.1, where $\frac{d}{n}$ increases slowly, is 1.17. It is evident from the graphs that fluctuations and/or relatively large competitive ratios are observed for $\frac{d}{n} > 0.02$, while below this threshold, we see a consistent trend of small competitive ratio in all considered cases.

The initial results are promising and motivate us towards a more sophisticated experimental analysis in order to get further insight into the problem. For example, we need to study the estimate against different failure probabilities as we did in our 1-stage randomized estimate for classical GT.

2.3 Future Extensions

Here we present some open questions and potential extensions of the current work. We also talk about the need for a classification of GT strategies and lay out a model for it.

2.3.1 Randomized Nonadaptive Estimate of d

Our results for competitive GT are based on the nonadaptive randomized estimate of the number d of defectives as mentioned in Section 4 of Paper-I. We studied the problem independently for any $d \leq n$. However, knowing that in many situations defectives are rare while n is huge, one can save many tests if a 2-stage estimator is allowed where stage 1 roughly determines the magnitude of d. Then, stage 2 focuses on the range of the most likely d.

For a given failure probability ϵ and using $g \log n$ queries for $g > 0$ fixed, we studied the factor $c = \frac{\hat{d}}{d}$, where $\hat{d}$ is the estimate of the actual d. We conjecture that constant factors c obtained as a result of the translation-invariant strategy (see Section 6, Paper-II) are already optimal. However, since we have obtained them numerically, a challenging question is: Can the dependency of optimal c, ϵ and the query number be characterized in a closed analytical form?

2.3.2 Query-Optimal Results

We remark that our minimal adaptive strategies in Section 2, Paper-III achieve query optimality only asymptotically for the considered cases. However, it is worthwhile to analyze the hidden constant factors, at least for moderate values of n and d.

For the case of a fixed rate of defectives addressed in Section 3 of Paper-III, an open question is whether 4 stages are already the minimum to achieve query optimal results.

2.3.3 Exact GT strategies

In Paper-IV we study exact GT strategies which run in a fixed number s stages for a given collection of n elements d of which are defective. Ultimate goal in this direction is a smooth transition from optimal strategies for small n to asymptotically optimal ones. However, solving $d = 2$ has already proven to be very complicated.

One interesting question to consider is: Does there exist, for every d, some s such that further stages after s do not improve the test number, *i.e.*, $t(n, d, s) = t(n, d, n)$? So far only $t(n, 1, 2) = t(n, 1, n)$ is proved in Section 2, Paper-V.

2.3.4 Classification of GT Strategies

Optimal query complexity for a GT problem differs based on the testing method and construction of the pools: disjoint or overlapping, deterministic or randomized. A GT strategy that assumes an upper bound on number d cannot be applied when nothing is known about d.

Earlier in this chapter we have reviewed scenarios when, *e.g.*, allowing for a small prescribed failure probability ϵ, randomized strategies can obtain some results which otherwise cannot be achieved using deterministic constructions. Similarly, there are also differences due to the growth rate of defectives. For example, in the previous Section 2.2.2, we observed that a 2-stage randomized strategy with prescribed success probability can asymptotically achieve the entropy lower bound for $d \ll n$ and growing much slower than n, whereas we can approach the entropy lower bound in 4 stages when $d = o(n)$.

Thus, in practice it becomes difficult to decide which GT results from the literature are known and suitable for the particular needs of a given GT problem instance. To address this problem, we first need to characterize the available results with respect to:

- **Design Differences:** We differentiate between deterministic and randomized strategies. A strategy can be adaptive, nonadaptive or multistage. Within multistage strategies we further distinguish between strategies which work in a fixed number s of stages versus strategies with expected number of stages. Design difference also incorporate the case of known versus unknown number of defectives (or rate of defectives), disjoint or overlapping pools.

- **Reliability Demands:** We consider whether the query complexity of a strategy refers to an exact, guaranteed or expected upper bound. We need not only the asymptotic results, but here we also require the exact expressions for the query number including the minor terms which are normally neglected during asymptotic analysis. In this way, we will be able to compare them based on the exact query numbers arising from these expressions for specific input parameters. With respect to output reliability, we study whether the output is always correct, or probably correct according to a fixed probability of success. At the cost of more queries, a probably correct strategy may verify its result and report its correctness. In case of failure, one may choose to run it until a correct and verified output is achieved. Similarly, reliability issues also arise due to testing errors. Therefore, complexity bounds are also different when tests err within certain limits.

From the study of established results for the GT problem, our objective is to sort them out in the light of the above points. Ideally, the results should be organized in the form of an easily accessible online knowledge base.

Later, one can access this knowledge base and figure out which GT strategy can efficiently solve a given problem instance. Our special concern is to focus at the problem size, *i.e.*, to find out optimal strategies for fixed values of n and d. Most of the known results primarily present asymptotic query complexities. They do not say much about the exact query number for fixed problem size, especially for small n and d.

As first steps, to capture the variations in terms of design differences and reliability constraints, we have proposed a classification of probabilistic search strategies in Paper-I. There we have specified problem constraints in terms of *queries*, *stages* and *output reliability*. We also develop theory for GT with disjoint simultaneous pools and exact GT.

Bibliography

[1] Ahlswede, R., Deppe, C., Lebedev, V.: Threshold and Majority Group Testing. In: Aydinian, H., Cicalese, F., Deppe, C. (eds.) Information Theory, Combinatorics, and Search Theory, LNCS, vol. 7777, pp. 488–508. Springer Berlin Heidelberg (2013)

[2] Asuncion, A.U., Goodrich, M.T.: Nonadaptive Mastermind Algorithms for String and Vector Databases, with Case Studies. IEEE Transactions on Knowledge and Data Engineering 25(1), 131–144 (2013)

[3] Bai, L., Roy, S.: A Two-Stage Approach for Network Monitoring. Journal of Network and Systems Management 21(2), 238–263 (2013)

[4] Bar-Noy, A., Hwang, F.K., Kessler, I., Kutten, S.: A new competitive algorithm for group testing. Discrete Applied Mathematics 52(1), 29–38 (1994)

[5] Carron, I.: A Magic Trick (2010), `(Accessed September 2013)` `http://nuit-blanche.blogspot.fr/2010/07/magic-trick.html`

[6] Chan, C.L., Cai, S., Jaggi, S., Saligrama, V.: Near-optimal stochastic threshold group testing. arXiv preprint arXiv:1304.6027 (2013)

[7] Chang, H., Chen, H.B., Fu, H.L., Shi, C.H.: Reconstruction of hidden graphs and threshold group testing. Journal of Combinatorial Optimization 22(2), 270–281 (2010)

[8] Chang, H., Fu, H.L., Shih, C.H.: Threshold group testing on inhibitor model. Journal of Computational Biology 20(6), 464–470 (2013)

[9] Chen, H.B., De Bonis, A.: An almost optimal algorithm for generalized threshold group testing with inhibitors. Journal of Computational Biology 18(6), 851–864 (2011)

[10] Cheng, Y., Du, D.Z.: New constructions of one- and two-stage pooling designs. Journal of Computational Biology 15(2), 195–205 (2008)

[11] Cheng, Y.: An efficient randomized group testing procedure to determine the number of defectives. Operations Research Letters 39(5), 352–354 (2011)

[12] Cheng, Y., Guo, J., Zheng, F.: A new randomized algorithm for group testing with unknown number of defective items. Journal of Combinatorial Optimization pp. 1–10 (2013)

[13] Cheng, Y., Xu, Y.: An efficient FPRAS type group testing procedure to approximate the number of defectives. Journal of Combinatorial Optimization pp. 1–13 (2012)

[14] Cheraghchi, M.: Improved Constructions for Non-adaptive Threshold Group Testing. Algorithmica 67(3), 384–417 (2013)

[15] Damaschke, P.: Threshold group testing. In: Ahlswede, R., Bäumer, L., Cai, N., Aydinian, H., Blinovsky, V., Deppe, C., Mashurian, H. (eds.) General Theory of Information Transfer and Combinatorics, LNCS, vol. 4123, pp. 707–718. Springer Berlin Heidelberg (2006)

[16] De Bonis, A., Gasieniec, L., Vaccaro, U.: Optimal Two-Stage Algorithms for Group Testing Problems. SIAM Journal on Computing 34(5), 1253–1270 (2005)

[17] Dorfman, R.: The Detection of Defective Members of Large Populations. The Annals of Mathematical Statistics 14(4), 436–440 (1943)

[18] Du, D.Z., Hwang, F.K.: Minimizing a combinatorial function. SIAM Journal on Algebraic and Discrete Methods 3(4), 523–528 (1982)

[19] Du, D.Z., Hwang, F.K.: Competitive group testing. Discrete Applied Mathematics 45(3), 221–232 (1993)

[20] Du, D.Z., Hwang, F.K.: Combinatorial group testing and its applications, Series on Applied Mathematics, vol. 3. World Scientific (2000)

[21] Du, D.Z., Hwang, F.K.: Pooling Designs and Nonadaptive Group Testing: Important Tools for DNA Sequencing, Series on Applied Mathematics, vol. 18. World Scientific (2006)

[22] Du, D.Z., Park, H.: On competitive group testing. SIAM Journal on Computing 23(5), 1019–1025 (1994)

[23] Du, D.Z., Xue, G.L., Sun, S.Z., Cheng, S.W.: Modifications of competitive group testing. SIAM Journal on Computing 23(1), 82–96 (1994)

[24] Dýachkov, A.G., Rykov, V.V.: Bounds on the length of disjunctive codes. Problems of Information Transmission (Russian) 18(3), 7–13 (1982)

[25] D'yachkov, A., Rykov, V., Deppe, C., Lebedev, V.: Superimposed Codes and Threshold Group Testing. In: Aydinian, H., Cicalese, F., Deppe, C. (eds.) Information Theory, Combinatorics, and Search Theory, LNCS, vol. 7777, pp. 509–533. Springer Berlin Heidelberg (2013)

[26] Eppstein, D., Goodrich, M.T., Hirschberg, D.S.: Improved Combinatorial Group Testing Algorithms for Real-World Problem Sizes. SIAM Journal on Computing 36(5), 1360–1375 (2007)

[27] Erdös, P., Frankl, P., Füredi, Z.: Families of finite sets in which no set is covered by the union of r others. Israel Journal of Mathematics 51(1-2), 79–89 (1985)

[28] Feller, W.: An Introduction to Probability Theory and Its Applications. John Wiley & Sons, New York, USA, volume 1 edn. (1950)

[29] Fischer, P., Klasner, N., Wegenera, I.: On the cut-off point for combinatorial group testing. Discrete Applied Mathematics 91(1-3), 83–92 (1999)

[30] Gilbert, A.C., Strauss, M.J.: Combinatorial Group Testing Magic Trick (2013), http://theproofisinthepudding.wordpress.com /2013/07/30/combinatorial-group-testing-magic-trick/

[31] Goodrich, M.T., Hirschberg, D.S.: Improved adaptive group testing algorithms with applications to multiple access channels and dead sensor diagnosis. Journal of Combinatorial Optimization 15(1), 95–121 (2008)

[32] Harvey, N.J.A., Patrascu, M., Wen, Y., Yekhanin, S., Chan, V.W.S.: Non-Adaptive Fault Diagnosis for All-Optical Networks via Combinatorial Group Testing on Graphs. In: INFOCOM 2007 - 26th IEEE International Conference on Computer Communications. pp. 697–705. IEEE (2007)

[33] Huang, S.H., Hwang, F.K.: When is individual testing optimal for nonadaptive group testing? SIAM Journal on Discrete Mathematics 14(4), 540–548 (2001)

[34] Hwang, F.K.: A Method for Detecting all Defective Members in a Population by Group Testing. Journal of the American Statistical Association 67(339), 605–608 (1972)

[35] Kainkaryam, R.M.: Pooling designs for high-throughput biological experiments. Dissertation, The University of Michigan (2010)

[36] Kautz, W., Singleton, R.: Nonrandom binary superimposed codes. IEEE Transactions on Information Theory 10(4), 363–377 (1964)

[37] Khattab, S.: A Defense Framework Against Denial-of-Service in Computer Networks. Dissertation, University of Pittsburgh (2008)

[38] Knill, E.: Lower bounds for identifying subset members with subset queries. In: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms, SODA '95. pp. 369–377. Society for Industrial and Applied Mathematics (1995)

[39] Lee, C.C., Lai, Y.M.: Toward a secure batch verification with group testing for VANET. Wireless Networks 19(6), 1441–1449 (2013)

[40] Li, C.H.: A sequential method for screening experimental variables. Journal of the American Statistical Association 57(298), 455–477 (1962)

[41] Lo, C., Liu, M., Lynch, J.P., Gilbert, A.C.: Efficient Sensor Fault Detection Using Combinatorial Group Testing. In: 2013 IEEE International Conference on Distributed Computing in Sensor Systems. pp. 199–206. IEEE, Cambridge, MA, USA (2013)

[42] Madej, T.: An application of group testing to the file comparison problem. In: The 9th International Conference on Distributed Computing Systems. pp. 237–243. IEEE (1989)

[43] Manasse, M., McGeoch, L., Sleator, D.: Competitive algorithms for online problems. In: Proceedings of the twentieth annual ACM symposium on Theory of computing STOC '88. pp. 322–333. ACM (1988)

[44] Mézard, M., Toninelli, C.: Group Testing With Random Pools: Optimal Two-Stage Algorithms. IEEE Transactions on Information Theory 57(3), 1736–1745 (2011)

[45] Ortega, A., Lin, Y.T.: Group Testing: A Novel Approach To Increase The Effective Frequency Or Accuracy Of Well-tests Without Increasing Infrastructure Costs. In: Proceedings of SPE Western Regional & AAPG Pacific Section Meeting, 2013 Joint Technical Conference. pp. 1–11. Society of Petroleum Engineers, Monterey, CA, USA (2013)

[46] Ruszinkó, M.: On the upper bound of the size of the r-cover-free families. Journal of Combinatorial Theory, Series A 66(2), 302–310 (1994)

[47] Schlaghoff, J., Triesch, E.: Improved Results for Competitive Group Testing. Combinatorics, Probability and Computing 14(1), 191–202 (2005)

[48] Sobel, M., Elashoff, R.M.: Group testing with a new goal, estimation. Biometrika 62(1), 181–193 (1975)

[49] Sobel, M., Groll, P.A.: Group testing to eliminate efficiently all defectives in a binomial sample. The Bell System Technical Journal 38(5), 1179–1252 (1959)

[50] Spencer, J.: Minimal completely separating systems. Journal of Combinatorial Theory 8(4), 446–447 (1970)

[51] Sterrett, A.: On the detection of defective members of large populations. The Annals of Mathematical Statistics 28(4), 1033–1036 (1957)

[52] Thai, M.T.: Group Testing Theory in Network Security. SpringerBriefs in Optimization, Springer, New York, USA (2012)

[53] Thai, M.T., Znati, T.: Detecting Application Denial-of-Service Attacks: A Group-Testing-Based Approach. IEEE Transactions on Parallel and Distributed Systems 21(8), 1203–1216 (2010)

[54] Thompson, K.H.: Estimation of the Proportion of Vectors in a Natural Population of Insects. Biometrics 18(4), 568–578 (1962)

[55] Tošić, T., Thomos, N., Frossard, P.: Distributed sensor failure detection in sensor networks. Signal Processing 93(2), 399–410 (2013)

[56] Ufimtsev, V., Bhowmick, S.: Application of Group Testing in Identifying High Betweenness Centrality Vertices in Complex Networks. In: Eleventh Workshop on Mining and Learning with Graphs. pp. 1–8. Chicago, Illinois, USA (2013)

[57] Ungar, P.: The cutoff point for group testing. Communications on Pure and Applied Mathematics 13(1), 49–54 (1960)

[58] Watson, G.: A Study of the Group Screening Method. Technometrics 3(3), 371–388 (1961)

[59] Xuan, Y., Shen, Y., Nguyen, N.P., Thai, M.T.: Efficient Multi-Link Failure Localization Schemes in All-Optical Networks. IEEE Transactions on Communications 61(3), 1144–1151 (2013)

[60] Yamamura, K., Hino, A.: Estimation of the Proportion of Defective Units by Using Group Testing Under the Existence of a Threshold of Detection. Communications in Statistics - Simulation and Computation 36(5), 949–957 (2007)

[61] Zhang, C., Ho, P.H., Tapolcai, J.: On batch verification with group testing for vehicular communications. Wireless Networks 17, 1851–1865 (2011)

[62] Zhang, L., Luo, J., Guo, D.: Neighbor discovery for wireless networks via compressed sensing. Performance Evaluation 70(7-8), 457–471 (2013)