Group Testing

Hung-Lin Fu

Introduction

Group testing is a kind of search which refers to locating certain elements of a set having a prescribed property. The procedure is running on the tests of sets instead of individual elements. For example, let S be a set of 100 coins of the same value and two of them are fake coins (with larger weight but look the same). In order to find these two coins, we may weight each coin individually to determine which two coins in the worst case are fake. Clearly, it may 99 times to find them in the worst case. But, if we weight 10 coins at a time, we know where are the fake coins after several tests. After 10 weightings, we also discover the weight of a normal coin. Therefore, the fake coins can be found in much less times.

The idea of group search was proposed sometime in World War II. Robert Dorfman published a paper to describe such a novel idea[1]. The motivation arouse from a large project to weed out all syphilitic men called up for induction to the war. Testing an individual for syphilis involves drawing a blood sample and analysing the sample to determine the presence or absence of syphilis. However, testing every soldier individually would have been very costly. "Group Testing" idea was then proposed by a coworker of R. Dorfman, D. Rosenblatt.

Here is the proposal of group testing. First, we put together around 10 sera (of 10 soldiers) together and test. If one or more of the sera in the pool contain syphilitic antigen, then the test is recorded as a "positive" one. On the other hand, if no one in the pool contains a syphilitic antigen, then the test is recorded as a "negative" one. If not many soldiers carry syphilitic antigen, then we leave a small number of positive pools in which we can test

individually to determine the soldiers with syphilitic antigens.

Though this idea sound great, this idea of grouping blood samples for syphilis screening was not actually put to case upon historic report. But, thanks to the Human Genomic Project of sequencing human DNA sequence, the idea of group testing is alive and play an important role. Not only that many other applications were discovered in recent years[2]. We shall introduce some of them in the following sections after a formal introduction for group testing.

1 Basic notions

Consider a set I of n items known to contain exactly d defectives. We shall call this set up the **combinatorial group testing** (**CGT** in short).

Note that in the study of Dorfman[1], as well as Sobel and Groll[3], the group testing is under probabilistic model, i.e., a probabilistic distribution is attached to the set of defectives and the goal is to minimize the expected number of tests required to identify the defective set.

The combinatorial group testing(CGT) was firststudied by Li.[4]. He was concerned with the situation where industrial and scientific experiments are conducted only to determine which of the variables are important. Usually, only some critical variables exist among man candidates. Hence, Li assumed that there are exactly d critical variables in a set S of n variables to start with. This opens the so-called (d, n)-problem of group testing.

In CGT, we also make the following assumptions:

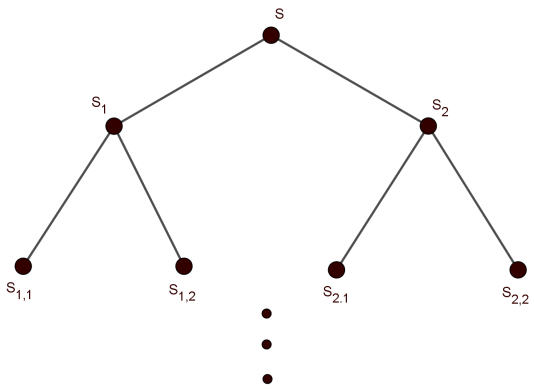
- The defectives look exactly like the good items.
- A test can be applied to arbitrary subset of S.
- There are "essentially" two outcomes: negative and positive.
- A "negative" outcome indicates that all items in the tested subset (group) are good. (The group is also said to be pure.)

- A "positive" outcome indicates at least one of the items in the tested subset is defective but not knowing which ones or how many are defectives. (Contaminated group)

Remark that if these assumptions are revised, then we have different models of group testings.

2 Algorithms

It is well known that a binary search can be represented by a binary tree. The algorithm of group testing which applies binary search is known as a binary splitting algorithm. The idea is to split the whole set S of items into two parts S_1 and S_2 , then test them. If one of them is negative, say S_2, then we need to test S_1 in the search of positives. If both of them are positive, then we split both of them and continue the process.
Now, assume that $\|S\| = n$, then it takes $\lceil \log_2 n \rceil$ tests to find at least one positive if there are any.



If the knowledge about the tests (positive or negative) can be adapted to decide which set of items is to be tested in next step, then the algorithm is called an adaptive algorithm. The above explanation can be referred to obtaining the following general upper bound for finding d positives in n items.

Proposition 2.1. *It takes at most* $d\lceil \log_2 n \rceil$ *tests to determine the* d *positives in a set of* n *items.*

In order to explain the second algorithm, let us denote a group testing by using a matrix-notation (array). Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of n items to be tested and $t_1, t_2, \ldots, t_m$ be m tests which we shall "apply" in group testing. Then, we can represent this experiment by using an $m \times n$ array $A = [a_{i,j}]_{m \times n}$ such that $a_{i,j} = 1$ if the j^{th} item s_j is included in the i^{th} test t_i, and $a_{i,j} = 0$ otherwise.

$$A := \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ \\ t_m \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & \ldots & s_n \end{array} \\ \left(\begin{array}{cccc} & & & \\ & & & \\ & a_{i,j} = 1 \quad or \quad 0 & & \\ & \blacksquare & & \\ & & & \\ & & & \end{array} \right) \end{array}$$

Figure 2.1 *A pooling design*

Now, let T_i be the i^{th} test and $y_i = 1$ if T_i is a positve test and $y_i = 0$ otherwise. $\vec{y} = (y_1, y_2, \ldots, y_m)^t$ is called the outcome vector of the group testing A (also known as a pooling design). In this set up, A can be referred as a linear transformation from $\{0,1\}^n$ to $\{0,1\}^m$. If it's possible to determine or estimate the positives in S from the outcome vector $\vec{y}$, then we have a non-adaptive algorithm represented by the pooling design A. This is because that we execute the m tests simultaneously (without knowing the outcome of the other tests). Here is an example of 7 items and 3 tests.

$$A: \begin{array}{c} \begin{array}{ccccccc} \bar{s_1} & \bar{s_2} & \bar{s_3} & \bar{s_4} & {s_5}^+ & \bar{s_6} & \bar{s_7} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Figure 2.2 *An example*

The tests are designed previously on sets $\{s_4, s_5, s_6, s_7\}$, $\{s_2, s_3, s_6, s_7\}$ and $\{s_1, s_3, s_5, s_7\}$. Now, if s_5 is the only positive item, then the outcome vector will be $\vec{y} = (1, 0, 1)^t$. On the other hand, if we have this outcome

vector, then we also conclude that s_5 is the only positive, provided that the assumption for the number of positives $d = 1$.
So, in order to determine the positives in a non-adaptive algorithm A, as a linear transformation, must be injective. This implies the size of domain has to be at most the size of range.

Proposition 2.2. *In a non-adaptive algorithm, if we can determine at most d positives out of n items, then we need at least t tests where $2^t \geq \sum_{i=1}^{d} \binom{n}{i}$, equivalently, $t \geq \left\lceil \log_2 \sum_{i=1}^{d} \binom{n}{i} \right\rceil$.*

Proof. This is a direct consequence of counting $|X|$ and $|Y|$ where $X \subseteq \{0,1\}^n$, $Y \subseteq \{0,1\}^m$ and $A\vec{x} = \vec{y}$ for each $\vec{x} \in \mathbb{X}$ which is a $(0,1)-$ vector with at most d $1^{'}$s. Notice that $\vec{x}$ is a column vector. □

It takes no time to realize the difference between the two algorithms mentioned above. In general, an adaptive algorithm will take less tests in locating the positive items comparing to a non-adaptive algorithm for the same (d, n)- problem. This is not difficult to see since previous knowledge of tests can be referred for taking next tests in an adaptive algorithm.

On the other hand, a non-adaptive algorithm well take less time to finish the testing without a doubt. But, more tests we may have to run in order to distinguish all the outcome vectors. Don't mention that when d is getting larger (so is n), the design of a suitable pooling design is a very tough job itself.

In what follows, we introduce some ideas in constructing a pooling design for a non-adaptive algorithm. First, we need a Boolean sum idea.
Let A be an $m \times n$ (0,1)-array , $[a_{i,j}]$. We denote each column by a set $B_j = \{i | a_{i,j} = 1\}$. For example, in Figure 2.2, $B_5 = \{1, 3\}$. Now, the Boolean sum of B_j and B'_j is simply $B_j \cup B'_j$. Again in Figure 2.2, $B_3 \cup B_5 = B_7$ and that thus the Boolean sum of B_3 and B_5 is B_7.

Definition 2.3. *A pooling design $A = [a_{i,j}]$ is said to be d-separable if any two distinct Boolean sum of d columns are distinct.*

Clearly, the pooling design in Figure 2.2 is 1-separable, but not 2-separable since $B_2 \cup B_6 = B_3 \cup B_5$. As mentioned above, if we can distinguish all possible outcomes, then we can determine the positives. Hence, if A is a d-separable pooling design then A can be applied to solve a (d, n)-problem.

Proposition 2.4. *If a set of n items contains exactly d positives then a d-separable pooling design can be apply to determine all the d positives.*

In case that there are "at most" d positives in the set of n items, then the separability has to be stronger.

Definition 2.5. *A pooling design is called $\bar{d}$-separable if all Boolean sum of at most d columns are distinct.*

It is therefore a direct consequence that a $\bar{d}$-separable pooling design can be applied to find at most d positives.

Even the design is obtained to fulfill the purpose, the decoding process is not easy at all since we have to go through all possible outcomes to determine the set of positives. Nowadays, the best way to locate d (or less) positives is to use the so called d-disjunct matrices.

Definition 2.6. *A pooling design (or matrix) is said to be d-disjunct if given any $d+1$ columns $B_1, B_2, \ldots, B_{d+1}$, then $B_{d+1} \nsubseteq \bigcup_{i=1}^{d} B_i$.*

By direct checking, we can see a d-disjunct matrix is also a $\bar{d}$-separable matrix. Furthermore, the decoding procedure in using d-disjunct matrices is comparatively simpler. We shall use the following example to explain the idea of decoding. First, we have a 2-disjunct matrix in Figure 2.3 (exercise). Now, assume that B_7 and B_{11} are columns corresponding to positive items s_7 and s_{11}. Then, we have the outcome vector at the end of the matrix.

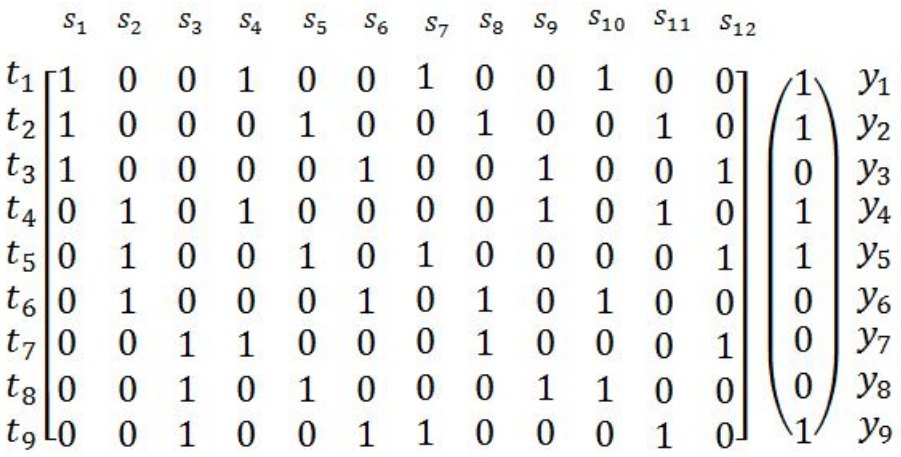


Figure 2.3 *2-disjunct matrix*

Now, consider any column B_i with B_7 and B_{11}. Since $B_i \nsubseteq B_7 \cup B_{11}$, there exist at least one row, say k^{th} row, $a_{k,i} = 1$ whenever $a_{k,7} = a_{k,11} = 0$, Therefore, we can use the 0-rows of the outcome vector to find negative items. For example, $y_3 = 0$ shows that s_1, s_6, s_9 and s_{12} are negatives. Similarly, since $y_6 = 0$, s_2, s_6, s_8 and s_{10} are negatives. After checking all the 0's in the outcome vector, the set of negatives is determined. Note that any negative item can be located and the decoding procedure is therefore a simple one.

Proposition 2.7. *In a (d, n)-problem, a d-disjunct matrix can be applied to determine all d positives. Furthermore, its decoding process is linear w.r.t. the number of tests.*

It is worth of mention that a pooling design ($(0, 1)$-array) can be thought as a binary code of length m with n codewords. On the other hand, we can also use binary codes to play the role of a non-adaptive group testing. Since a combinatorial design and a hypergraph can be represented by an incidence matrix respectively, they can also be applied to construct pooling designs.

We shall end this section with a fundamental problem in group testing.

Problem: Let $T(d, n)$ and $t(d, n)$ denote the minimum number of tests to determine the set of d positives by using adaptive and non-adaptive algorithms respectively. Find $T(d, n)$ and $t(d, n)$. (Note that $T(1, n) = t(1, n) =$

$\lceil \log_2 n \rceil$ is known.)

3 Complex models

The group testing we have discussed so far has a set of positive items each can induce a positive effect. Sometimes it takes a set of elements combined to induce a positive effect. For example, in drug test, the positive effect comes from the combination of several distinct drugs. Therefore, starting from now, the set of positive items is replaced by a collection of subsets of the set of items, $\mathbb{D} = D_1, D_2, \ldots, D_t$ where each D_i is called a positive complex. Usually, we assume $D_i \nsubseteq D_j$ for all $i \neq j$. The first introduction of this notion is by Torney[5] who gave the complexes of eukaryotic DNA transcription and RNA translation as examples.

Besides its applications to molecular biology, the group testing in searching for positive complexes (Complex model) is also interesting as a genuine generalization of the classic group testing from searching subsets to searching subgraphs induced by positive edges in a hypergraph. Note that the set of items remains the same, but the set of positive complexes can be thought as the set of hyper-edges.

Group testing on complex model

- An experiment is applied to an arbitrary set $S \subseteq N$ (the set of items) with two possible outcomes; a negative outcome implies S does not contain any D_i in $\mathbb{D}$ (the set of positive complexes) and a positive outcome implies otherwise, i.e., $D_j \subseteq S$ for some j.

- If graphs are concerned, then an experiment is applied to an arbitrary vertex subset S of the vertex set N with two possible outcomes; a negative outcome implies that the subgraph induced by S, $<S>_G$, does not contain any positive edge $D_i \in \mathbb{D}$, and a positive outcome implies otherwise, i.e., D_j is an edge in $<S>_G$ for some j.

There are two basic types of algorithms to find the positive complexes:

1. Adaptive algorithm

2. non-adaptive algorithm.

Here we only introduce the idea of the first one. For simplicity, we also assume that all complexes (positive) are of size two, Therefore, the model can be thought as finding a subgraph (simple) defined on the set of items N.

Learning a hidden subgraph

The hidden subgraph might have several edges or no edges. Since the number of edges is unknown, extra tests are necessary. The most common idea is to find the first edge and then delete the edge and keep moving forward to find the others. We use the following example to explain the idea.

Example 3.1. *$N = \{1, 2, 3, 4, 5, 6, 7\}$ and $E = \{\{2, 5\}, \{3, 5\}\}$.*

For convenience, each test will be called as a query. Furthermore, $Q(i)$ denotes the i^{th} query.

- *$Q(1)$: Does N contain an edge?* *Yes.*
- *$Q(2)$: Does $<S = \{1, 2, 3, 4\}>$ contain an edge?* *No.*
- *$Q(3)$: Does $<\{1, 2, 3, 4, 5, 6\}>$ contain an edge?* *Yes.*
  So edges are hidden between $\{1, 2, 3, 4\}$ and $\{5, 6\}$.
- *$Q(4)$: Does $<\{1, 2, 3, 4, 5\}>$ contain an edge?* *Yes.*
- *$Q(5)$: Does $<\{1, 2, 5\}>$ contain an edge?* *Yes.*
- *$Q(6)$: Does $<\{1, 5\}>$ contain an edge?* *No.*

We conclude $\{2, 5\}$ is an edge. Delete this edge and go back to ask

- $Q(7) := Q(4)$.

- $Q(8) := Q(5)$.
- $Q(9)$ *gives the answer. But, we can not stop here.* $Q(10)$ *is necessary.*
- $Q(10)$: *Does* $<\{5,6,7\}>$ *contain an edge.* *No.* ***(STOP)***

Note here that if all edges are 2-subsets, then we may take the set of items as $\binom{7}{2}$ subsets and find two positive items in this case. For more details about finding positive complexes by way of hidden subgraphs, the reader may refer to [6][7]. On the other hand, if non-adaptive algorithm is applied, then the book by Ding-Zhu Du and Frank K. Hwang provide a very clear idea[2].

Group testing with error outcomes

In an experiment, it is possible to make an error in answering whether the test is positive or negative. If indeed a test on a set of items which contains a positive item and the conclusion says "negative", then the correctness of the group testing will be in doubt. Consequently, we are not able to find the positive items. Therefore, a better idea has to be there to avoid such a mistake.

Based on the fact that the error of one test is not going to affect the other tests in non-adaptive algorithm, it is easier to handle group testing with error outcomes. We remark here that in an adaptive algorithm, an earlier conclusion of a test can affect the following tests, but we don't know which test is an errorous one. The readers may refer to [8][9] for handling such circumstances.

Now, we explain the idea of constructing a pooling design such that even at most e errors occur in experiments, we can still find all the positive items.

Let A be a $(0,1)$-matrix which has t rows and n columns. Clearly, A can be viewed as a non-adaptive algorithm with t tests. Assume that $\vec{x}^t$ is an n-dim row vector $(x_1, x_2, \ldots, x_n)$ such that $x_i = 1$ if the i^{th} item is positive and $x_i = 0$ otherwise. For convenience, we use $\|\vec{x}^t\|$ to denote the number of 1's in $\vec{x}^t$. Therefore, we are aiming at finding the d positive items by using $A\vec{x} = \vec{y}$. That is, if $\vec{y}$ is given, then $\vec{x}$ can be determined.

So, if $\|\vec{x}^T\| \leq d$, then $X = \{\|\vec{x}\| \mid \|\vec{x}^t\| \leq d\}$ is of size $\sum_{i=0}^{d} \binom{n}{i}$.

This implies that the set of outcome vectors Y contains at least this amount of vectors as well. But, if errors occurred somewhere, then Y must have extra property as we have learned in coding theory.

First, Y can be viewed as a binary code of length t. In order to find the

correct outcome, the code distance $d(Y)$ must be larger than "1". In a word, if $d(Y) \geq 2r+1$, then we can correct at most r errors. This idea provided the backgroup of constructing a suitable pooling design A which can achieve this goal. Without a doubt, the number of rows is going to be larger than the one where no errors occur.

From Proposition 2.7, we notice that in a (d, n)-problem, a d-disjunct matrix can be applied to determine "$\leq d$" positives. Hence, a stronger matrix is need to do the same job if we allow at most e errors to occur.

Definition 3.2. *A matrix is said to be $(d; z)$-disjunct if for any $d+1$ columns, $C_0, C_1, \ldots, C_d$, $|C_0 \setminus \bigcup_{i=1}^{d} C_i| \geq z$.*
Clearly, if $z = 1$, then we have a d-disjunct matrix.

Theorem 3.3. *A $(d; 2e+1)$-disjunct matrix can be applied to determine up-to-d positives with at most e errors.*

Proof. Let C^- be an arbitrary negative item and D be the set of positive items, namely, $C_1, C_2, \ldots, C_d$. Since the matrix is $(d; 2e+1)$-disjunct, $|C^- \setminus \bigcup_{i=1}^{d} C_i| \geq 2e+1$. This implies that there are at least $2e+1$ rows, the entries in column C^- is 1 and 0 in C_i's, $i = 1, 2, \ldots, d$.

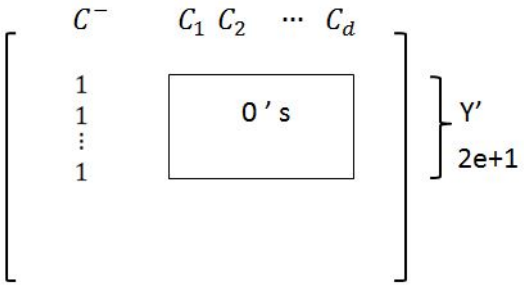


Figure 2.4

In this occasion, the $2e+1$ coordinates of Y' are supposed to be all $0's$ since all positive items are not involved in these tests. Even there are e errors, Y' still contains at least $e+1$ $0's$. This observation provides a

decoding algorithm for finding all the negatives and therefore all positives can be identified.
(•) For each column C and each outcome vector Y, let ${t_0}^Y(C) = |C \backslash Y|$ and ${t_1}^Y(C) = |C \cap Y|$. Then, ${t_0}^Y(C) > e$ implies that C is a negative item. (Here, C and Y can be recognized as sets, the support of Y.) Since all negatives satisfy this property, we conclude the proof. □

Group testing with thresholds

The idea was first introduced by Damaschke at 2005[10]. In many experiments, if positive items are not large enough then the outcome may not be able to recognized as positive. On the other hand, it is possible to have a gap say between l and u, such that the outcome is not precisely determined. So, extra effort is needed.

Definition 3.4. *Let l and u be two positive integers where $l < u$, called the lower and upper threshold. A group test "S" in threshold model is positive if S contains at least u positives, and negative if at most l positives are present. Furthermore, if the number of positives in S is between l and u, the outcome can give an arbitrary answer. $g =_{def} u - l - 1$ is called the gap.*

We remark here that when $g > 0$, the group testing turns out to be very difficult to identify all positives no matter adaptive or non-adaptive algorithms are applied. Here, we introduce the idea of handling the case of $g = 0$ and non-adaptive algorithms are utilized. First, we need a definition.

Definition 3.5. *A matrix is $(d, u; z]$ - disjunct if for any $d + u$ columns $C_1, C_2, \ldots, C_{d+u}$, there exist z rows intersecting $C_1, C_2, \ldots, C_u$, i.e.,*

$$|\bigcap_{i=1}^{u} C_i \backslash \bigcup_{i=u+1}^{u+d} C_i| \geq z.$$

As mentioned above, z plays the role of detecting errors. Now, $\bigcup_{i=u+1}^{u+d} C_i$ plays the role of threshold. Therefore, we have the following result to deal with threshold group testing with possible e errors.

Theorem 3.6. *Apply $(d-u+1, u, 2e+1]$-disjunct matrix as a pooling design to the threshold group testing with no gap and error-tolerance e. Then for a u-subset X of N, ${t_0}^V(X) \leq e$ if and only if X is a set of positive items.*

Proof. Instead of checking each column, we consider a set of columns X. By the same idea as in complex model, $X \nsubseteq P$ (the set of positives) iff ${t_0}^V(X) \geq e+1$. This concludes the proof. □

Pooling designs are common tools to efficiently distinguish positive clones from negative clones in clone library screening. In some applications, there is a third type of clones called "inhibitors" whose effect is in a sense to obscure the positive clones in pools. Various inhibitor models have been proposed in the literature. The readers can refer to [11][12] for more references about this model.

Applications of Group Testing

First, we notice that constraints on group testing are the main tools in applications. With these requirements, an experiment or test could be carried out which accomondates the situation in real world. Here are some of them and we will explain more details when they are applied.

1. Each group of test has a size limitation, say $k \in \mathbb{N}$.
2. The items are arranged in an early stage as the vertices of a graph, and then only certain subgraphs (groups are the vertex set) can be tested, for example, a group induces a path (subgraph).
3. The outcome of a test can be generalized to h-ary outcome $(0, 1, 2, \ldots, h-1^+)$, i^+ denotes at least i positives. The classical group testing can be recognized as a binary outcome: $(0, 1^+)$.

4. The outcomes may have errors.

5. There are threshold for being positive or negative.

6. Besides positive and negative items, there are inhibitors.

Now, we are ready to introduce some practical applications.

Multiaccess channels

As the name shows, a multiaccess channel is a communication channel that connects many users. Each user can listen and transmit on the channel, but the channel allows only one user to transmit at one period of time. In case of more users who transmit at the same time, then the signals collide and "noise effect" occurs. Clearly, multiaccess channels are important for various real world applications such as wireless computer networks and also cellular phone networks. Therefore, how to assign transmission times to the users so that their messages do not collide is an important issue.

Suppose that we have n users. A simple method is to give each user their own tie slot to transmit, requiring n slots. However, this is very inefficient and cost a lot of tie slots. It is natural to assume that only a few users will want to transmit at any given time slot - otherwise a multiaccess channel is not practical at all.

Now, with the idea of group testing, this sharing problem is usually tackled by dividing into "epochs". We say an user is active if they have a message at the start of an epoch. If a "new" message is generated during an epoch, the user will (only) become active at the start of the next epoch. Note that an epoch ends when every active user has successfully transmitted their messages. So, the problem is to find all the uses in a given epoch, and schedule a time slot for them to transmit (if they have not already done so.) This is then a group testing problem where the query ask "a set of users to attempt a transmission if they are active". The outcome of the query is a ternary outcome $(0, 1, 2^+)$, corresponding to no active users, exactly one user

and more than one users. Only the third case needs a second "query" to continue the search for suitable epoch.

Compressed sensing (Machine learning)

Machine learning is a field of computer science for a long while. But, due to the emerge of Data Science, it becomes a more general topic in different expertise especially AI. As we can expect, there are tons of applications in human life. One of the main part of machine learning is "learning by examples", where the task is to approximate some unknown function when given its value at a number of specific points. Therefore, this function learning problem can be tackled with a group-testing approach, mainly using "Non-adaptive Algorithms".

In a simple version, we have an unknown function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ where $f(\vec{x}) = \vec{a}\vec{x}$, and $\vec{a} \in \mathbb{Z}_2^n$. Here, we use OR for addition and AND for multiplication in binary set-up. We also assume that $\vec{a}$ contains at most d non-zero entries where d is very small comparing to n. The aim is to construct (reconstruct) f using t points ($\vec{x}'s$) evaluations. Of course, we are looking for small t.

Now, we notice that recovering f is equivalent to finding $\vec{a}$ and $f(\vec{x}) = \dot{\vec{a}}\vec{x} = 1$ reflects to some test (carried out by $\vec{x}$) is positive. So, if t points are applied, say $\vec{x_1}, \vec{x_2}, \ldots, \vec{x_t}$, then $f(\vec{x_i}) = \dot{\vec{a}}\vec{x}_i = y_i$ gives the i^{th} component of an outcome vector $\vec{y}^t = (y_1, y_2, \ldots, y_t)$. Our aim is to recover f, finding $\vec{a}$, by using the outcome vector. This is clearly a group testing problem.

In reality, the function f we plan to recover is not as simple. The function can be defined as $f : \mathbb{C}^n \to \mathbb{C}$ where $f(\vec{x}) = \dot{\vec{a}}\vec{x}$ and the inner product can be defined accordingly. To recover $\vec{a}$ in this format is exactly the compressed sensing problem. The input $\vec{x}$ is called a measurement and $\vec{a}$ can be viewed as a signal. In a sense, we may refer " compressed sensing " as a kind of " continuous " group testing. Many related works have been published by using the idea of group testing, see [13]

Data forensics

Data forensics is a field dedicated to finding methods for compiling digital evidence of a crime. Such crimes typically involve an adversary modifying the data, documents or databases of a victim, with examples including the altering of tax records, a virus hiding its presence, or an identity thief modifying personal data. There are also crimes from collusion attacks on multimedia fingerprints. In order to reduce the occurrence of crimes mentioned above, some tools such as one-way cryptographic hash or the construction of better pooling designs have been studied and provided. Group testing plays a key role as well.

Applications in Molecular Biology

Each human cell contains 23 pairs of chromosomes and each chromosome is a packed DNA which is 5000 times shorter than the extended form. It is known that a DNA consists of two chains, entwined forming a double spiral to form a DNA double helix. Each chain is a sequence of four types of nucleotides, A (Adenine), G (Guanine), T (Thymine) , and C (Cytosine). Therefore, each chain can be recognized as a string of alphabets {A, G, C, T}. The two strings stay together in double helix with a duality relation where one of them can obtained from the other by the mapping A $\rightarrow$ T, G $\rightarrow$ C, T $\rightarrow$ A, C $\rightarrow$ G. As a matter of fact the two strings can be separated under proper " heating ", but they have a tendency to bind to each other when put together. This binding tendency is known as "hybridization".

The idea of the application is that we can use hybridization to find out whether a DNA string T, called a target sequence, contains a specific substring S. Let S^{-1} denote the complementary strand of S. (If S: A T G T C, then S^{-1}: T A C A G .) Now, mix T and S^{-1}. Clearly, if T contains S, then S^{-1} will hybridize with S. This effect is magnified by the so-called polymer chain reaction (PCR) technique. This technique enables the hybridization observable and also measurable. A PCR is essentially a technology to make

copies of existing DNA pieces by using hybridization.

A primer is a short DNA sequence either preceding or succeeding the DNA sequence $S(S^{-1})$ of interest. Again, if S exists in T, the hybridization effect would make a primer of S^{-1} to grow into an S^{-1} along S under action of "enzyme". Heating is able to break up the hybridization pair to allow eah half to grow with other copies of S and S^{-1}. Hence, the primers can be used as probes to observe the existence of S in T. (PCR)

In a multiplex PCR, many primers are used simultaneously in one test. Suppose the set P of primers are ordered according to their positions in T. In fact, we do not know the order yet. Then two primers are adjacent if there is no other primer in P between them. Clearly, they are not far apart in T. In a multiplex test, a PCR product is obtained for each pair of adjacent primers. In theory, by counting the number of PCR products of different lengths, we know the number of pairs of adjacent primers. But, in practice, this information is not entirely accurate due to experimental errors and the possibilities that two PCR products have the same lengths.

Based on the explanation able,we can use a mathematical model to obtain a more reliable answer for the relationship between primers. By letting the primers as vertices of a graph, we are looking for the incident structure of this graph. Such a model is known as "learning hidden graphs" which was introduced earlier in this article. More references about this topic can be referred to [6][7].

There are other applications of group testing in molecular biology. For examples, Physical mapping, Contig sequencing, Finding exon boundaries in cDNA (proteins) and Protein-Protein interactions. We are not ale to include all of them here. The interested readers may refer to the literatures such as [14] or the books [2].

References

[1] R. Dorfman, The detection of defective members of large populations, *Ann. Math. Statist.* 14 (1943),436-440

[2] D. Z. Du and F. K. Hwang, Pooling designs and nonadaptive group testing, 2006, *World Scientific Publishing*.

[3] M. Sobel and P. A. Groll, Group testing to eliminate efficiently all defectives in a binomial sample, *Bell System Tech. J.* 28(1959) 1179-1252.

[4] C. H. Li, A sequential method for screening experimental variables, *J Amer. Statist. Assoc.* 57 (1962), 455-477

[5] D. C. Torney, Sets pooling designs, *Ann. Combin.* 3 (1999), 95-101.

[6] Reconstruction of hidden graphs and threshold group testing (with Hui-Lan Chang, Hong-Bin Chen and Chi-Huai Shih), *Combin. Optim.* 22 (2011), no. 2, 270-281.

[7] Learning a hidden graph (with Huilan Chan and Chie-Huai Shih), *Optim. Letters*, 8 (2014), no. 8,2341-2348.

[8] A Pelc, Solution of Ulam's problem for searching with a lie, *JCT(A)* 44 (1987), 129-140.

[9] W. Guzicki, Ulam's searching game with two lies, *JCT(A)*, 54 (1990), 1-19.

[10] P. Damaschke, Threshold group testing, *Lecture Notes Comput. Sci.* 4123, 707-718.

[11] Identification and classification problem on pooling designs for inhibitor models (with Hui-Lan Chang and Hong-Bin Chen), *J. Compu. Biology*, Vol.17 (2010), No. 7, 927-941.

[12] Threshold group testing on inhibitor model (Huilan Chang and Chie-Huai Shih), *J. Comput. Biology*, Vol.20, No. 6, 2013, 1-7

[13] A. G. Kamal and S. Venkatest, Boolean compressed sensing and noisy group testing, *IEEE Transaction Inform. Th.*, 58(3), 2012, 1880-1901.

[14] A. Schliep, D. C. Torney and S. Rahmann, Group testing with DNA chips: generating designs and decoding experiments, *proceedings of the* 2^{nd} *IEEE Computer Society Bioinformatics Conference*, 2003.