Multistage group testing algorithms

Vorobyev Ilya

Skoltech

COMBINATORICS AND GEOMETRY DAYS II, April 13-16, 2020

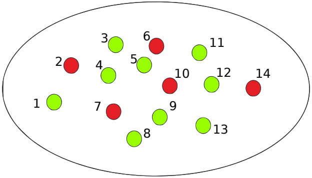
Outline

Group Testing Problem

Given: a set of t coins $[t] = \{1, 2, \ldots, t\}$. Some subset S_{un} of coins are fake, $|S_{un}| = s$.
Goal: to find fake coins using the minimal number of tests.
Group test: $Q \subset [t]$.
The test result:

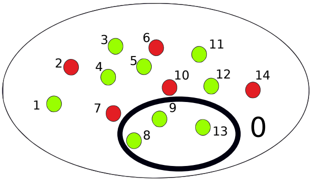
$$y = \begin{cases} 1, & \text{if } Q \cap S_{un} \neq \varnothing \\ 0, & \text{if } Q \cap S_{un} = \varnothing \end{cases}$$

Combinatorial Group Testing



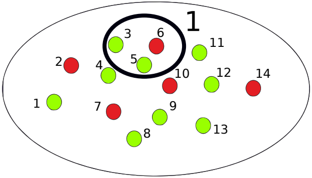
Example:
$t = 14$,
$S_{un} = \{2, 6, 7, 10, 14\}$.

Combinatorial Group Testing



Example:
$t = 14$,
$S_{un} = \{2, 6, 7, 10, 14\}$.
$S = \{8, 9, 13\}$.
$S \cap S_{un} = \varnothing$, thus the test result is negative(0).

Combinatorial Group Testing



Example:
$t = 14$,
$S_{un} = \{2, 6, 7, 10, 14\}$.
$S = \{3, 5, 6\}$.
$S \cap S_{un} = \{6\}$, thus the test result is positive(1).

Different types of search algorithms

- adaptive or sequential – later tests depend on the results of previous tests.
- non-adaptive or parallel – all tests are predefined and carried out in parallel.
- multistage - algorithm consists of the several stages, where tests of stage i depend on the results of tests from stages $1, 2, \dots i-1$.

Different types of search algorithms

- adaptive or sequential – later tests depend on the results of previous tests.
- non-adaptive or parallel – all tests are predefined and carried out in parallel.
- multistage - algorithm consists of the several stages, where tests of stage i depend on the results of tests from stages $1, 2, \dots i-1$.

By $N_p(t, s)$ we denote the minimal number of tests in an algorithm, which finds s defects among t elements using p stages.

Known results

s - fixed, $t \to \infty$.

Adaptive algorithms

$N_\infty(t, s) = s \log_2 t(1 + o(1))$.

Known results

s - fixed, $t \to \infty$.

Adaptive algorithms

$N_\infty(t, s) = s \log_2 t(1 + o(1))$.

Non-adaptive algorithms

D'yachkov, Rykov 1983; Erdos, Frankl, Furedi 1985;

$$c_1 \frac{s^2}{\log_2 s} \log_2 t(1 + o(1)) \leqslant N_1(t, s) \leqslant c_2 s^2 \log_2 t(1 + o(1))$$

Known results

s - fixed, $t \to \infty$.

Adaptive algorithms

$N_\infty(t,s) = s\log_2 t(1+o(1))$.

Non-adaptive algorithms

D'yachkov, Rykov 1983; Erdos, Frankl, Furedi 1985;

$$c_1\frac{s^2}{\log_2 s}\log_2 t(1+o(1)) \leqslant N_1(t,s) \leqslant c_2 s^2\log_2 t(1+o(1))$$

Two-stage algorithms

D'yachkov, Rashad 1990; De Bonis, Gasieniec, Vaccaro 2005;

$$N_2(t,s) \leqslant cs\log_2 t$$

Conjecture

For any s there exists $p(s)$ such that $N_{p(s)}(t, s) = s \log_2 t(1 + o(1))$.

Conjecture

For any s there exists $p(s)$ such that $N_{p(s)}(t, s) = s \log_2 t(1 + o(1))$.

It's true for $s = 2, 3, 4$.

$$N_2(t, 2) = 2 \log_2 t(1 + o(1))$$

$$N_4(t, 3) = 3 \log_2 t(1 + o(1))$$

$$N_4(t, 4) = 4 \log_2 t(1 + o(1))$$

Outline

Search problems on hypergraphs

Problem Statement

Given: a hypergraph $H = (V, E)$.
One edge $e \in E$ is defective.
Goal: to find the edge e using the minimal number of special questions.
Question: subset Q of vertexes V.
Answer:

$$y = \begin{cases} 1, & \text{if } Q \cap e \neq \varnothing \\ 0, & \text{if } Q \cap e = \varnothing \end{cases}$$

Search problems on hypergraphs

Problem Statement

Given: a hypergraph $H = (V, E)$.
One edge $e \in E$ is defective.
Goal: to find the edge e using the minimal number of special questions.
Question: subset Q of vertexes V.
Answer:

$$y = \begin{cases} 1, & \text{if } Q \cap e \neq \varnothing \\ 0, & \text{if } Q \cap e = \varnothing \end{cases}$$

Search problem for complete s-uniform hypergraph on t vertexes $\Leftrightarrow$ problem for t coins, s of which are fake.

Matrix representation

Any non-adaptive algorithm consisting of N tests can be represented by a binary $N \times t$ matrix X such that each test corresponds to the row, and each element stands for the column.

We put $x_i(j) = 1$ if the j-th element is included in i-th test; otherwise, $x_i(j) = 0$.

For any subset $S \subset T$ define the binary vector

$$r(X, S) = \bigvee_{j \in S} x(j),$$

Then outcomes of tests can be represented by a binary vector $y = r(X, S_{un})$.
Example $t = 4$, $N = 4$, $S_{un} = \{1, 2\}$.

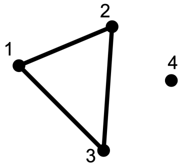
1	2	3	4	**r(X, {1, 2})**
0	1	1	0	1
1	0	1	1	1
1	1	1	0	1
0	0	0	1	0

Given a s-uniform hypergraph $H(V, E)$, we want to find a defective hyperedge e_{un} in it. Suppose that we have already performed some set of tests X.

Construct a hypergraph $H'(X, y) = (V, E')$ with the same vertex set V. The set of hyperedges E' equals to the set of all hyperedges from the original hypergraph, which are consistent with tests X, i.e, all hyperedges $e \in E$, $|e| = s$, such that $r(X, e_{un}) = r(X, e)$.

Example $t = 4$, $N = 4$, $s = 2$, $H = K_4$.

1	2	3	4	**r(X, {1, 2})**
0	1	1	0	1
1	0	1	1	1
1	1	1	0	1
0	0	0	1	0



Outline

2 defectives

Theorem

There exists a 2-stage algorithm, which finds 2 defective coins using $2\log_2 t(1 + o(1))$.

Remark

It is know that $N_1(t, 2) > 2.0008\log_2 t(1 + o(1))$ (Coppersmith D., Shearer J. B. 1998).

2 defectives

Theorem

There exists a 2-stage algorithm, which finds 2 defective coins using $2\log_2 t(1 + o(1))$.

Remark

It is know that $N_1(t, 2) > 2.0008\log_2 t(1 + o(1))$ (Coppersmith D., Shearer J. B. 1998).

Theorem

There exists a 3-stage algorithm, which finds a defective edge in an arbitrary graph $G = (V, E)$ *using* $\log_2 |E|(1 + o(1))$.

Our approach

High level description of our approach:

1. Construct a matrix X of size $N \times t$ such that for every possible outcome vector $y \in \{0, 1\}^N$ a graph $H(X, y)$ contains at most D edges, where $D = o(N)$.
2. Use this matrix X as a testing matrix for the first stage of our algorithm.
3. Test "suspicious" items individually.

The total number of test in this algorithm is at most $N + 2D \sim N$.

Sketch of the algorithm

Construct a matrix X of size $N \times t$ such that a graph $H(X, y)$ doesn't have a matching of size L or a vertex of degree L for every possible outcome vector $y \in \{0,\ 1\}^N$.

Sketch of the algorithm

Construct a matrix X of size $N \times t$ such that a graph $H(X, y)$ doesn't have a matching of size L or a vertex of degree L for every possible outcome vector $y \in \{0, 1\}^N$.

Proposition 1

If the maximum vertex degree and the maximum cardinality of a matching in a graph $G = (V, E)$ don't exceed L, then $|E| \leqslant 2L^2$.

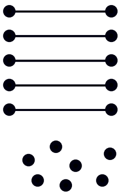
Sketch of the algorithm

Construct a matrix X of size $N \times t$ such that a graph $H(X, y)$ doesn't have a matching of size L or a vertex of degree L for every possible outcome vector $y \in \{0,\ 1\}^N$.

Proposition 1

If the maximum vertex degree and the maximum cardinality of a matching in a graph $G = (V, E)$ don't exceed L, then $|E| \leqslant 2L^2$.

Proof of the Proposition 1



Consider any maximum matching M in the graph $G = (V, E)$. Every edge e has at least one endpoint in M. Therefore, the total number of edges is upper bounded by $2L^2$.

Sketch of the algorithm

Construct a matrix X of size $N \times t$ such that a graph $H(X, y)$ doesn't have a matching of size L or a vertex of degree L for every possible outcome vector $y \in \{0,\ 1\}^N$.

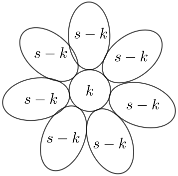
Proposition 1

If the maximum vertex degree and the maximum cardinality of a matching in a graph $G = (V, E)$ don't exceed L, then $|E| \leqslant 2L^2$.

The existence of such matrix can be proved by random coding for $N = \left(2 + \frac{1}{L}\right) \log_2 t$.

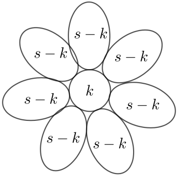
Outline

Let's try to generalize



Call the set of edges $e_1, e_2, \ldots, e_L$ a (s, k, L)-configuration if $e_i \cap e_j = U$, $|U| = k$, for any i and j.

Let's try to generalize

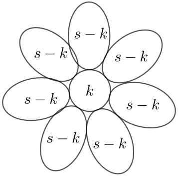


Call the set of edges $e_1, e_2, \dots, e_L$ a (s, k, L)-configuration if $e_i \cap e_j = U$, $|U| = k$, for any i and j.

Proposition 2

If a s-uniform hypergraph $H = (V, E)$ doesn't contain (s, k, L) configurations for $k = 0, 1, \dots s-1$, then the number of edges $|E|$ is at most $s!L^s$.

Let's try to generalize



Call the set of edges $e_1, e_2, \ldots, e_L$ a (s, k, L)-configuration if $e_i \cap e_j = U$, $|U| = k$, for any i and j.

Problem: Such algorithms need more than $s \log_2 t$ tests.
We need too many tests at the first stage to exclude (s, k, L) configurations for $k > s/2$.

For $s = 3 (s = 4)$ construct a matrix X of size $N \times t$ such that a graph $H(X, y)$ doesn't have $(3, 1, L = \log_2 \log_2 t)$ $((4, 1, L = \log_2 \log_2 t)$ and $(4, 2, L = \log_2 \log_2 t))$ configurations for every possible outcome vector $y \in \{0,\ 1\}^N$.

For $s = 3 (s = 4)$ construct a matrix X of size $N \times t$ such that a graph $H(X, y)$ doesn't have $(3, 1, L = \log_2 \log_2 t)$ $((4, 1, L = \log_2 \log_2 t)$ and $(4, 2, L = \log_2 \log_2 t))$ configurations for every possible outcome vector $y \in \{0,\ 1\}^N$.

Lemma

Hypergraph H can be colored in $poly(L)$ colors such that there are no two vertexes of the same color in one edge.

For $s = 3 (s = 4)$ construct a matrix X of size $N \times t$ such that a graph $H(X, y)$ doesn't have $(3, 1, L = \log_2 \log_2 t)$ $((4, 1, L = \log_2 \log_2 t)$ and $(4, 2, L = \log_2 \log_2 t))$ configurations for every possible outcome vector $y \in \{0,\ 1\}^N$.

Lemma

Hypergraph H can be colored in $poly(L)$ colors such that there are no two vertexes of the same color in one edge.

2nd stage: find colors of defective vertexes. Estimate degrees of defective vertexes.
3rd stage: find one defective vertex using a binary search.
4th stage: find two remaining vertexes among the neighbors of the first defective.

For $s = 3 (s = 4)$ construct a matrix X of size $N \times t$ such that a graph $H(X, y)$ doesn't have $(3, 1, L = \log_2 \log_2 t)$ $((4, 1, L = \log_2 \log_2 t)$ and $(4, 2, L = \log_2 \log_2 t))$ configurations for every possible outcome vector $y \in \{0,\ 1\}^N$.

Lemma

Hypergraph H can be colored in $poly(L)$ colors such that there are no two vertexes of the same color in one edge.

2nd stage: find colors of defective vertexes. Estimate degrees of defective vertexes.
3rd stage: find one defective vertex using a binary search.
4th stage: find two remaining vertexes among the neighbors of the first defective.

Careful choice of the matrix X gives an optimal answer.

For $s = 3 (s = 4)$ construct a matrix X of size $N \times t$ such that a graph $H(X, y)$ doesn't have $(3, 1, L = \log_2 \log_2 t)$ $((4, 1, L = \log_2 \log_2 t)$ and $(4, 2, L = \log_2 \log_2 t))$ configurations for every possible outcome vector $y \in \{0, \ 1\}^N$.

Lemma

Hypergraph H can be colored in $poly(L)$ colors such that there are no two vertexes of the same color in one edge.

2nd stage: find colors of defective vertexes. Estimate degrees of defective vertexes.
3rd stage: find one defective vertex using a binary search.
4th stage: find two remaining vertexes among the neighbors of the first defective.

Careful choice of the matrix X gives an optimal answer.

General case: Lemma is incorrect for $s > 4$.

Outline

Contribution:

1. The conjecture $N_{p(s)}(t, s) = s \log_2 t(1 + o(1))$ was proved for $s = 2, 3, 4$.
2. A 3-stage optimal algorithm to find the defective edge in an arbitrary graph was developed.

Future research directions:

1. Prove conjecture for $s > 4$.
2. Design an optimal multistage algorithm to find the defective hyperedge in an arbitrary hypergraph.

Thank you for your attention!