Problem Chosen	2021 MCM/ICM Summary Sheet	Team Control Number
A		**46**

Group Testing Method in Testing COVID-19

Summary

When testing a large number of disease samples with a low prevalence rate, in order to improve efficiency, the method of group testing is often used, which is to mix the samples to be tested together for a test.

In specific implementation, due to the large sample size, the test samples are usually divided into several groups according to a certain principle, and the k samples in each group are mixed for group testing. The test results of this group of samples are judged based on the test results of the mixed samples.

The purpose of our article is to establish a group testing model that considers false negative and false positive cases and use this model to guide the detection strategy for COVID-19.

In the simple case of single grouping without considering false negative and false positive cases, by calculating the mathematical expectation of the number of detections of a single sample, we get a simple probability model. In the process of multiple grouping, we use the recurrence relationship to find the number of inspections after the i-th grouping T_i. Use T_i and the mathematical expectation $a_k T_k p_k$ of a single measurement sample to get the total number of detections N, and use it to find E, which is the mathematical expectation of the number of tests per person.

In the detection after analysis, since the expression of E is complicated and not conducive to direct solution, we use search heuristic algorithm to get the approximate value of a_i in the process, and use this series of approximate values to find appropriate grouping method.

Considering that the false negative rate of various current detection methods is relatively high, we improved the model based on repeated sampling testing. In addition, by analyzing the relationship between the positive and negative sample detection value and the true value, we deduce the influence of false negative rate and false positive rate on E.

Through further verification, Our model has a high accuracy rate for false negative detection. In other words, this method can effectively detect asymptomatic infections.

Finally, we established a community infection model, by dividing the city into communities of different risk levels, and different communities adopt different detection schemes to optimize the detection process.

This model can effectively guide us to correctly adopt the strategy of mixed sample detection in infectious disease detection to improve detection efficiency, and has certain application and promotion value.

Keywords: Mathematical Expectation;Group Testing Method;Search Heuristic lgorithm;

Contents

1 Introduction

1.1 Background

Single-sample testing can guarantee timely and effective when the sample size is small. However, in the case of a large sample size, single-sample testing can no longer have sufficient timeliness. Literature tells us that group testing can not only save a lot of financial and material resources than single sample test, but also improve efficiency and save a lot of time. Therefore, group testing is a good choice for a country or region with serious epidemic situation. The testing procedure in group testing are as follows:

Here we consider a group testing scheme with several stages: at some stage, if a group tests negative,then all individuals in the group are declared to be negative and released; otherwise the group is subdivided into smaller ones, and we proceed to the next stage. At the last stage, the group size is 1, and all the remaining (i.e. not have been released) individuals are tested separately.

1.2 Problem Statement

A new wave of covid 19 attacks City A and the number of tests suddenly expands. In order to improve the detection efficiency and formulate detection measures in time, we adopt the group testing method. Under such circumstances,we established a mathematical model to solve the following problems:

- There are false positives and false negatives in COVID-19 testing. Our mathematical model needs to simulate the process of multiple assignments in the test and avoid the interference of false positives and false negatives in the test using known false positive rate and false negative rate.
- To stop early the Covid-19 spread, it is highly important to test people who are infected but without symptoms. This requires our mathematical model to have a sensitivity comparable to single-sample testing for asymptomatic infections.
- Propose another scheme for the group testing process. It does not necessarily have multiple stages and non overlapping groups.
- Based on your modeling work, write a letter to CDC of City A on your recommendation of group testing strategy for this city

2 Assumptions and Symbols

2.1 Model Hypothesis

- The false positive rate and false negative rate are constants that are only related to the detection method
- The parameter p is only related to the severity of the epidemic, which is a constant in a specific area.
- The number of infected persons during the test remains unchanged.
- There are no operational problems such as sample contamination during the detection process

2.2 Symbols and Definitions

Table 1: Notation

Symbols	Definitions
p	The positive rate of the city
q	The negative rate of the city
i	Number of Groups
a_i	Number of people in each grouping
b_i	Number of groups in each grouping
T_i	The total number of detections for the i-th grouping
N	The total number of detections
E	The Mathematical Expectation of tests per person.

P.S. Other symbols instruction will be given in the text.

3 Mixed Sample Testing Model

In this part, we will establish a mixed sample detection model and improve the model based on actual conditions.

3.1 Primary Model based on Mathematical Expection

First, we build a simple model, this model does not consider the test error, that is, the test result is absolutely accurate, there is no false negative rate and false positive rate.

3.1.1 Mathematical Expectation Model of Single Grouping

In order to clearly illustrate the problem, let us first consider the case of grouping only once.

Assuming that there is a batch of samples to be tested, the probability of each sample being tested as positive. (that is, the sample is abnormal) is p. Obviously, the probability of the test being negative (that is, the sample is normal) is $q = 1 - p$. Divide this batch of samples into several groups, each with k.

To simplify the problem, we assume that the detection method is highly sensitive to the items to be tested, that is, if one sample in the mixed sample is positive, the mixed sample test result is also positive; if the mixed sample test result is negative, then consider that the test results of this group are all negative

Use X to denote the number of tests required by each person in each group, then X is a discrete random variable.So:

$$X = \begin{cases} \frac{1}{k}(\text{when test result is positive}) \\ \frac{1}{k} + 1(\text{when test result is negative}) \end{cases} \tag{1}$$

The distribution law of X can be obtained as:

$$P_{(X=\frac{1}{k})} = q^k \tag{2}$$

$$P_{(X=1+\frac{1}{k})} = 1 - q^k \tag{3}$$

In order to reduce the detection workload, for a given q = 1-p, select the number of people k in each group so that

$$E_{(X)} = 1 + \frac{1}{k} - q^k \tag{4}$$

The smaller the value of $E_{(X)}$, the better.

3.1.2 Mathematical Expectation Model of Multiple Stages Grouping

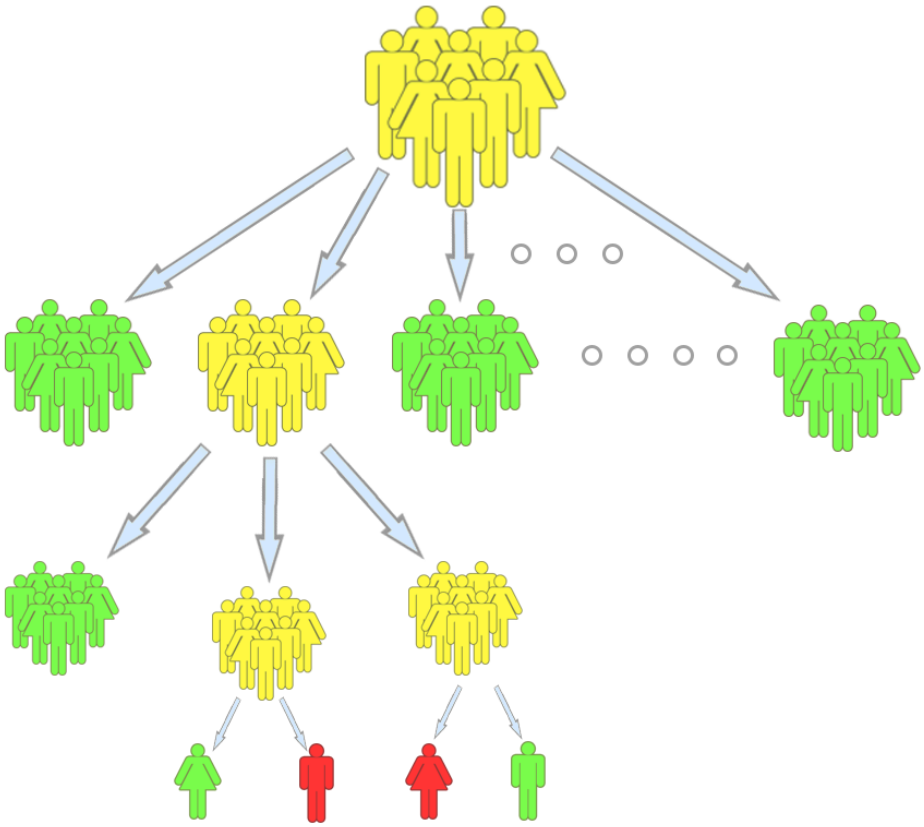

Figure 1: Group Detection Figure
Yellow represents people with positive individuals, green represents negative individuals, and red represents positive individuals

In this part, we consider the case of multi-stage grouping. In order to explain the problem clearly, we give the following definitions:

- Suppose there were a_i people in each group at the i-th grouping (specifically,a_i represents the total number of people tested).
- Suppose there were b_i groups at the i-th grouping.
- Suppose p is the positive rate of the city, $q = 1 - p$ is the negative rate of the city.
- Suppose T_i is the number of detections required after the i-th grouping.

From these definitions, we can derive the following conclusions:

1. For the i-th grouping, the probability of a positive test for each group is $p_I = 1 - q^{a_I}$.
2. $a_i = a_{i+1}b_{i+1}$
3. $a_i = \dfrac{a_0}{\prod\limits_{j=0}^{i} b_j} \geq 1$

From the above definition, we get the following relationship of T_i:

$$\begin{cases} T_1 = b_1 \\ T_2 = T_1 p_1 b_2 = b_1 p_1 b_2 \\ T_3 = T_2 p_2 b_3 = b_1 p_1 b_2 p_2 b_3 \\ T_4 = T_3 p_3 b_4 = b_1 p_1 b_2 p_2 b_3 p_3 b_4 \\ \cdots \end{cases} \tag{5}$$

Get its general form:

$$T_i = \begin{cases} b_i, i = 1 \\ \prod\limits_{j=1}^{i} b_j \prod\limits_{j=1}^{i-1} p_j, i \geq 2 \end{cases} \tag{6}$$

Formula Analysis Because the test subject is a human within a city, a large number of samples, so we chose the formula b. According to $b_i = \dfrac{a_{i-1}}{a_i}$,$p_i = 1 - q^{a_i}$,we simplify the formula to get

$$T_i = \begin{cases} \dfrac{a_0}{a_1}, i = 1 \\ \dfrac{a_0}{a_i} \prod\limits_{j=1}^{i-1} (1 - q^{a_j}), i \geq 2 \end{cases} \tag{7}$$

Assuming that N is the total number of detections, assuming that k grouping is performed (testing is performed after the kth grouping, if a group of samples is detected as positive, then the grouping is not continued, and all the members in this group are directly tested), get the following relationship:

$$N = \sum_{i=1}^{k} T_i + a_k T_k p_k \tag{8}$$

Substitute T_i,p_i to get

$$N = \begin{cases} \dfrac{a_0}{a_1} + a_1 \dfrac{a_0}{a_1}(1 - q^{a_1}), k = 1 \\ \dfrac{a_0}{a_1} + \sum\limits_{i=2}^{k} (\dfrac{a_0}{a_i} \prod\limits_{j=1}^{i-1} (1 - q^{a_j})) + a_k (\dfrac{a_0}{a_k} \prod\limits_{j=1}^{k-1} (1 - q^{a_j}))(1 - q^{a_k}), k \geq 2 \end{cases} \tag{9}$$

Among them, $a_k T_k p_k$ represents the number of times each person needs to test separately after k tests. Let $E = \frac{N}{a_0}$ to denote the average number of tests per person. From the equation 9:

$$E = \begin{cases} \frac{1}{a_1} + (1 - q^{a_1}), k = 1 \\ \frac{1}{a_1} + \sum_{i=2}^{k} (\frac{1}{a_i} \prod_{j=1}^{i-1} (1 - q^{a_j})) + \prod_{j=1}^{k} (1 - q^{a_j}), k \geq 2 \end{cases} \tag{10}$$

Our goal is to find a set of $a_1, a_2 \cdots a_k$ and k to minimize E.

3.2 Introduce false negatives and false positives

In order to simplify the problem, we did not assume the wrong situation in the previous model. In this part, we considered this situation.

For the detection problem, the samples can be divided into four cases: true positive cases (TP), true negative cases (TN), false positive cases (FP), and false negative cases (FN) according to the combination of their detection situation and the real situation. The relationship between them is as follows:

Detection \ True	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Table 2: Relationship among TP, FP, FN and TN.

Suppose that the probability of a person being actually positive (including true positives and false negatives) is p, and the probability of a person being actually negative (including true negatives and false positives) is $q = 1p$. And the probability that a negative sample is falsely detected as positive is p_{np}, and the probability that a positive sample is falsely detected as negative is p_{pn}.The p_{n-p} and p_{p_n} values for nucleic acid detection methods can be obtained from [1] and [4].

According to the defination above:

- True positive rate (TPR):The number of true positive samples detected divided by the number of all true positive samples.
- False positive rate (FPR):The number of false positive samples detected divided by the number of all true negative samples.
- True negative rate (TNR):The number of true negative samples detected divided by the number of all true negative samples.
- False negative rate (FNR):The number of false negative samples detected divided by the number of all true positive samples.

Their expressions are as follows:

$$\begin{cases} TPR = \dfrac{TP}{TP+FN} \\ FPR = \dfrac{FP}{FP+TN} \\ TNR = \dfrac{TN}{TN+FP} \\ FNR = \dfrac{FN}{TP+FN} \end{cases} \tag{11}$$

Assuming the total number of samples is K, the number of true positive samples: $TP = Kp(1-p_{p-n})$; the number of false positive samples: $FP = Kqp_{n-p}$; the number of true negative samples: $TN = Kq(1-p_{n-p})$; Number of false negative samples: $FN = Kpp_{p-n}$. Substituting into the formula 11, we get:

$$\begin{cases} TPR = 1-p_{p-n} \\ FPR = p_{n-p} \\ TNR = 1-p_{n-p} \\ FNR = p_{p-n} \end{cases} \tag{12}$$

The ratio of the number of samples that tested positive to the total number of samples (i.e. the probability that the sample tested positive):

$$\hat{p} = \frac{TP+FP}{K} = p(1-p_{p-n}) + qp_{n-p} \tag{13}$$

The ratio of the number of samples that tested negative to the total number of samples (i.e. the probability that the sample tested negative):

$$\hat{q} = 1-\hat{p} = \frac{TN+FN}{K} = q(1-p_{n-p}) + pp_{p-n} \tag{14}$$

Replace q in the formula 10 with $\hat{q}$ to get the mathematical expectation of the measured value of the number of detections $\hat{E}$ in the case of false detection:

$$\hat{E} = \begin{cases} \dfrac{1}{a_1} + (1-\hat{q}^{a_1}), k = 1 \\ \dfrac{1}{a_1} + \sum\limits_{i=2}^{k}(\dfrac{1}{a_i}\prod\limits_{j=1}^{i-1}(1-\hat{q}^{a_j})) + \prod\limits_{j=1}^{k}(1-\hat{q}^{a_j}), k \geq 2 \end{cases} \tag{15}$$

3.3 Multiple Testing Model for The Same Sample

Due to the limitations of detection methods, there are false detections in actual testing. The probability of testing an actual positive sample as negative (false negative) is relatively large, while the probability of testing an actual negative sample as positive(false positive) is relatively small. The former error is more harmful.

Asymptomatic infections are usually false negatives. In order to detect asymptomatic infections, we need to increase the number of inspections to improve accuracy. We choose to perform multiple tests on the same sample, but the increase in the number of tests is also

accompanied by increase in cost and time. We assume that the probability of an accurate test being positive is 80%, then the accuracy rate of repeated tests is $(1-0.2\times0.2)\times100\%=96\%$, which has reached a relatively high accuracy rate. So we choose to test the same sample at most twice in the next model.

Group according to the following detection principles:

- First, perform the first test on a batch of samples. After the test, the batch of samples is divided into two parts: the test result is positive and the test result is negative.
- The second test is performed on the samples that were negative in the first test, and the samples are divided into two parts after the test: the test result is positive and the test result is negative.
- If the results of both tests are negative, the sample is judged to be negative, and the rest are classified as positive samples.

Let $\hat{p}_i = 1-\hat{q}^{a_i}$, we get the following expression:

$$\begin{cases} T_1 = b_1 + b_1\hat{q}^{a_1} = b_1(1+\hat{q}^{a_1}) \\ T_2 = T_1\hat{p}_1 b_2 + T_1\hat{p}_1 b_2\hat{q}^{a_2} = b_1 b_2\hat{p}_1(1+\hat{q}^{a_1})(1+\hat{q}^{a_2}) \\ T_3 = T_2\hat{p}_2 b_3 + T_2\hat{p}_2 b_3\hat{q}^{a_3} = b_1 b_2 b_3\hat{p}_1\hat{p}_2(1+\hat{q}^{a_1})(1+\hat{q}^{a_2})(1+\hat{q}^{a_3}) \\ T_4 = T_3\hat{p}_3 b_4 + T_3\hat{p}_3 b_4\hat{q}^{a_4} = \cdots \\ \cdots \end{cases} \tag{16}$$

The general expression for T_i is as follows

$$T_i = \begin{cases} b_i(1+\hat{q}^{a_i}), i=1 \\ \prod_{j=1}^{i} b_j \prod_{j=1}^{i-1} p_j \prod_{j=1}^{i}(1+\hat{q}^{a_i}), i\geq 2 \end{cases} \tag{17}$$

Substitute $b_i = \frac{a_{i-1}}{a_i}$,$\hat{p}_i = 1-\hat{q}^{a_i}$ to simplify the formula, get:

$$T_i = \begin{cases} \frac{a_0}{a_1}(1+\hat{q}^{a_i}), i=1 \\ \frac{a_0}{a_i}\prod_{j=1}^{i-1}(1-\hat{q}^{a_j})\prod_{j=1}^{i}(1+\hat{q}^{a_j}), i\geq 2 \end{cases} \tag{18}$$

Assuming that a total of k grouping is performed, the required number of detections is:

$$N = \sum_{i=1}^{k} T_i + a_k T_k \hat{p}_k(1+\hat{q}^{a_k}) \tag{19}$$

In the same way as in 3.1.2, substitute T_i into equation9 to get the corresponding values of N in the case of multiple detections as follows:

$$N = \begin{cases} \frac{a_0}{a_1}(1+\hat{q}^{a_1}) + a_1\frac{a_0}{a_1}(1-\hat{q}^{2a_1})(1+\hat{q}^{a_1}), k=1 \\ \frac{a_0}{a_1}(1+\hat{q}^{a_1}) + \sum_{i=2}^{k}(\frac{a_0}{a_i}\prod_{j=1}^{i-1}(1-\hat{q}^{a_j})\prod_{j=1}^{i}(1+\hat{q}^{a_j})) + a_0\prod_{j=1}^{k}(1-\hat{q}^{2a_j})(1+\hat{q}^{a_k}), k\geq 2 \end{cases} \tag{20}$$

Let $E = \frac{N}{a_0}$ represent the average number of tests for each person. From the equation 10 and 20, we can get:

$$E = \begin{cases} \frac{1}{a_1}(1+\hat{q}^{a_1}) + (1-\hat{q}^{2a_1})(1+\hat{q}^{a_1}), k = 1 \\ \frac{1}{a_1}(1+\hat{q}^{a_1}) + \sum_{i=2}^{k}(\frac{1}{a_i}\prod_{j=1}^{i-1}(1-\hat{q}^{a_j})\prod_{j=1}^{i}(1+\hat{q}^{a_j})) + \prod_{j=1}^{k}(1-\hat{q}^{2a_j})(1+\hat{q}^{a_k}), k \geq 2 \end{cases} \tag{21}$$

4 Model Solving

In this section, we'll try to find a set of $a_1, a_2 \cdots a_k$ and k to minimize E. In the optimization problem for the above parameters, we will use nonlinear optimization method and two heuristic algorithms, simulated annealing [2] and genetic algorithm [3]. In addition, we also use the information theory model to solve the theoretical optimal value of the parameter.

4.1 Use heuristic algorithm to solve parameters

4.1.1 Simulated Annealing Algorithm

Let the initial temperature be $T_{initial} = 100$, the termination temperature be $T_{final} = 1 \times 10^{-8}$, the annealing rate be $\alpha = 0.99$, the annealing process can be expressed as

$$T(n+1) = \alpha T(n) \quad n = 1, 2, 3, \cdots \tag{22}$$

Suppose the optimal value corresponding to the old solution before each cooling down is E_{old}. After cooling down, each variable of the old solution will increase or decrease randomly under the premise of meeting the conditions to obtain a new solution, and find the corresponding Optimal value E_{new}, and then judge whether to accept the new solution according to Metropolis criteria:

$$P = \begin{cases} 1, f_{new} \leq f_{old} \\ e^{\frac{f_{old}-f_{new}}{T}}, f_{new} > f_{old} \end{cases} \tag{23}$$

Where P is the probability of choosing a new solution to replace the old one

In order to improve the possibility of obtaining the optimal solution during the annealing process, at first, we generated 1000 sets of decreasing random number sequences with a length of k as the initial sequence. Whenever the temperature decreases, we take the corresponding of the 1000 sets of sequences. The sequence with the smallest E is used as the new solution at the current temperature.

The following are the images of the temperature and its corresponding optimal value in the annealing process corresponding to the number of stages when $\hat{q} = 99.99\%$:

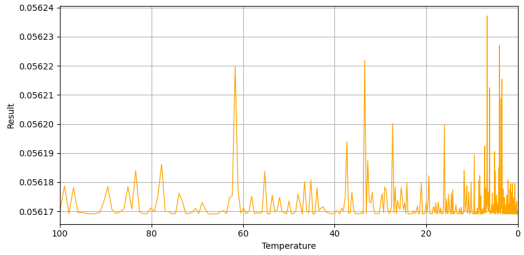

Figure 2: $k = 1$

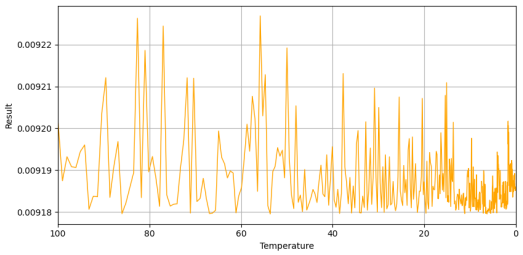

Figure 3: $k = 2$

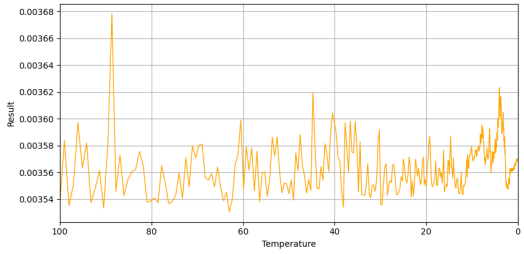

Figure 4: $k = 3$

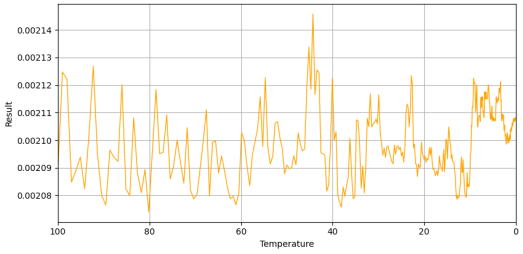

Figure 5: $k = 4$

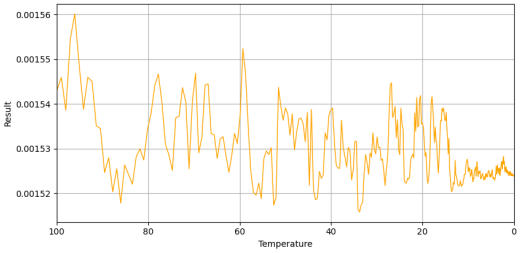

Figure 6: $k = 5$

The images of the temperature and its corresponding optimal value in the annealing process corresponding to different stages when $\hat{q} = 99.9\%$or $\hat{q} = 99\%$ are in the appendices E.

4.1.2 Genetic Algorithm

The process of genetic algorithm can be described as: Randomly generate initial population $\Longrightarrow$ Use fitness function to evaluate fitness $\Longrightarrow$ Select parents $\Longrightarrow$ according to fitness to generate offspring, and make offspring chromosomes cross and mutate.

Suppose the initial population size is $Popsize = 50$, the population chromosome crossover rate is $p_{crossover} = 90\%$, the population chromosome mutation rate is $p_{mutation} = 9\%$, the fitness function is E, The selection method is the Roulette Wheel Selection.

In order to improve the possibility of obtaining the optimal solution after executing the algorithm, the maximum generation number can be set to $Generation = 10000$, and the smallest fitness in each generation E_{now} can be compared with the smallest fitness in the

previous generations Compare E_{opt} and update E_{opt} to the minimum value, which is the optimal fitness.

The following are the images of the current generation and its corresponding optimal value in the evaluation process corresponding to the number of stages when $\hat{q} = 99.99\%$:

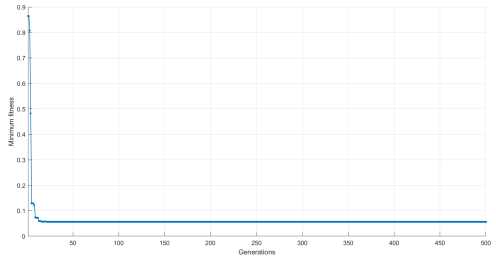

Figure 7: $k = 1$

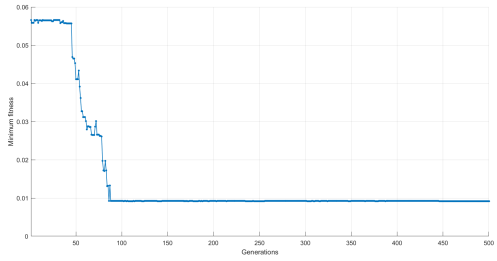

Figure 8: $k = 2$

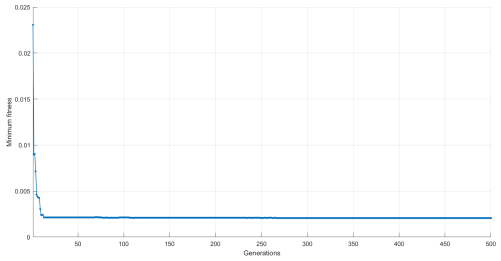

Figure 9: $k = 3$

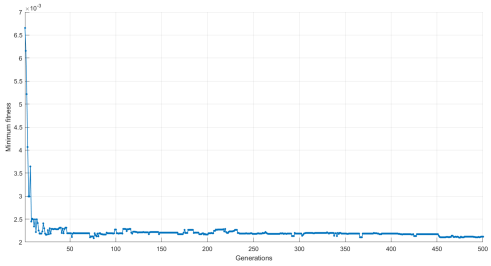

Figure 10: $k = 4$

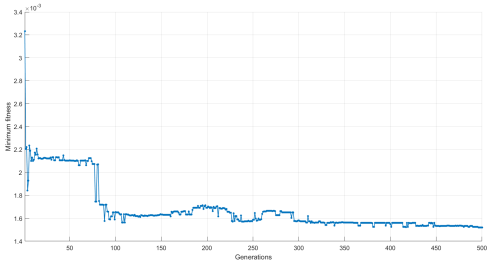

Figure 11: $k = 5$

The following are the images of the current generation and its corresponding optimal value in the evaluation process corresponding to the number of stages when $\hat{q} = 99.9\%$or $\hat{q} = 99\%$ are in the appendices E.

4.2 Non-Linear Model Optimization

In order to verify the rationality of the optimal solution of the function obtained by the above two heuristic algorithms, we introduced the nonlinear programming solver in the operations research software Lingo.

After inputting the function expression and constraint conditions into the Lingo window and performing Solve, compare the obtained solution with the optimal value obtained by the

heuristic algorithm, and the table is as follows($q = 99.99\%$):

Stages	Optimal Expectation		
	Simulated Annealing	Genetic Algorithm	Non-Linear Model
1	0.05616910	0.05621353	0.05616910
2	0.00917949	0.00918937	0.00917946
3	0.00353056	0.00356402	0.00353029
4	0.00207394	0.00207808	0.00207264
5	0.00151584	0.00152096	0.00151504

Table 3: Optimal Expectation of tests per person under three different algorithms($\hat{q} = 99.99\%$)

Stages	Simulated Annealing				
	a_1	a_2	a_3	a_4	a_5
1	71.47114	/	/	/	/
2	439.9232	71.61322	/	/	/
3	1186.684	443.0021	71.25979	/	/
4	2025.299	1178.800	443.5190	65.92744	/
5	2994.677	2018.771	1135.898	435.6160	69.20123

Table 4: Optimal Grouping Scheme under the Simulated Annealing Algorithm($\hat{q} = 99.99\%$)

Stages	Genetic Algorithm				
	a_1	a_2	a_3	a_4	a_5
1	68.66507	/	/	/	/
2	437.9305	69.42801	/	/	/
3	1155.862	386.8132	86.68050	/	/
4	2045.456	1202.401	419.9731	83.02808	/
5	2658.864	1978.317	1234.891	451.7704	76.24633

Table 5: Optimal Grouping Scheme under the Genetic Algorithm($\hat{q} = 99.99\%$)

Stages	Non-Linear Model				
	a_1	a_2	a_3	a_4	a_5
1	71.47083	/	/	/	/
2	440.8435	71.47108	/	/	/
3	1171.588	440.8415	71.47127	/	/
4	2058.032	1171.600	440.8400	71.47052	/
5	2913.617	2057.980	1171.592	440.8367	71.47084

Table 6: Optimal Grouping Scheme under the Non-Linear Model($\hat{q} = 99.99\%$)

According to the comparison results, the optimal solutions obtained by two heuristic algorithms are basically the same as the solution obtained by the nonlinear programming solver, indicating that the result obtained by the heuristic algorithm is reasonable, and under this model, the optimal group scheme solution for different stages is the scheme in the above table.

4.3 Group optimization model based on information theory

Assuming that the probability of being negative in this area is q, then the mathematical expectation of positive individuals among n in this area is $E_{(n)} = nq$. Let $m = [E_{(n)}]$, there are C_n^m infections with equal probability. According to information theory [5], this requires $\frac{\ln C_n^m}{\ln 2}$ bits information. And each detection obtains 1 bit of information, so the minimum value of the average detection times is $\frac{\ln C_n^m}{\ln 2}$.

To minimize the number of tests, it is required to obtain the most information from each test, which requires that the positive probability of each test is close to 0.5: Analyzing the first few tests, the general relationship of the number of tests a_i is as follows:

$$\begin{cases} a_1 = \log q 0.5 \\ a_2 = \log 2q - 10.5 \\ a_3 = \log 4q - 30.5 \\ \cdots \\ a_n = \log 2^{n-1} q - 2^{n-1} + 1 \end{cases} \tag{24}$$

We used python language programming to solve parameters. For example, when $p = 0.9999$, the approximate integer value of a_i is: $a_1 = 6931, a_2 = 3465, a_3 = 1733, a_4 = 866, a_5 = 433, a_6 = 216, a_7 = 108, a_8 = 54, a_9 = 27, a_{10} = 13, a_{11} = 6, a_{12} = 3, a_{13} = 1$, these series of parameters are theoretically optimal without limiting the number of detections for a single individual parameter. In actual detection, especially when q is large, considering the time requirement of detection and most people tends to be unwilling to be tested multiple times, this grouping may not be the optimal situation in actual situations.

4.4 Comparison of Several Models to Solving Parameters

Using simulated annealing algorithm, genetic algorithm, and non-linear optimization method to obtain the detection times of each individual E when $\hat{q}$=99.99%. The chart in which $\hat{q}$=99.9% are in the appendix 6.2.1.

Stages	Optimal Expectation		
	Simulated Annealing	Genetic Algorithm	Non-Linear Model
1	0.05616910	0.05621353	0.05616910
2	0.00917949	0.00918937	0.00917946
3	0.00353056	0.00356402	0.00353029
4	0.00207394	0.00207808	0.00207264
5	0.00151584	0.00152096	0.00151504

Table 7: Optimal Expectation of tests per person under three different algorithms($\hat{q} = 99.99\%$)

The optimal value of the parameter a_i obtained using the three optimization methods is as follows

Stages	Simulated Annealing				
	a_1	a_2	a_3	a_4	a_5
1	71.47114	/	/	/	/
2	439.9232	71.61322	/	/	/
3	1186.684	443.0021	71.25979	/	/
4	2025.299	1178.800	443.5190	65.92744	/
5	2994.677	2018.771	1135.898	435.6160	69.20123

Table 8: Optimal Grouping Scheme under the Simulated Annealing Algorithm($\hat{q} = 99.99\%$)

Stages	Genetic Algorithm				
	a_1	a_2	a_3	a_4	a_5
1	68.66507	/	/	/	/
2	437.9305	69.42801	/	/	/
3	1155.862	386.8132	86.68050	/	/
4	2045.456	1202.401	419.9731	83.02808	/
5	2658.864	1978.317	1234.891	451.7704	76.24633

Table 9: Optimal Grouping Scheme under the Genetic Algorithm($\hat{q} = 99.99\%$)

Stages	Non-Linear Model				
	a_1	a_2	a_3	a_4	a_5
1	71.47083	/	/	/	/
2	440.8435	71.47108	/	/	/
3	1171.588	440.8415	71.47127	/	/
4	2058.032	1171.600	440.8400	71.47052	/
5	2913.617	2057.980	1171.592	440.8367	71.47084

Table 10: Optimal Grouping Scheme under the Non-Linear Model($\hat{q} = 99.99\%$)

The parameter optimization algorithm based on information theory can find the optimal grouping method under the given probability $\hat{q}$. Because there are many parameters a_i when $\hat{q}$ is large, and the parameters a_i in the algorithm affect each other, it is not appropriate to omit some parameters. In order to save space, only the case of $0.900 \leqslant \hat{q} \leqslant 0.995$ is taken here.

$\hat{q}$	Information Theory Model						
	a_1	a_2	a_3	a_4	a_5	a_6	a_7
0.9	6.578813	3.106284	1.356915	/	/	/	/
0.95	13.513407	6.578813	3.106284	1.356915	/	/	/
0.99	68.967564	34.309618	16.979748	8.312950	3.975530	1.797290	0.678458
0.995	138.282573	68.967564	34.309618	16.979748	8.312950	3.975530	1.797290

Table 11: Optimal Grouping Scheme under the Information Theory Model

5 Community sampling model considering transmission route

5.1 Model Establishment

Considering that there are often one or several outbreak points during the spread of infectious diseases, which have certain transmission routes, the above model assumes that all individuals in the city are equally likely to be infected. This assumption does not conform to the actual situation. Moreover, the above model requires multiple sampling operations for each individual in city a, which will incur additional costs in actual applications. Considering that the activities of virus carriers are mainly in the community where the virus is located, and the probability of the virus spreading in a smaller community is higher, we propose a group detection method that is more suitable for actual conditions–a community-based group testing method . Here we first make these following reasonable assumptions about the situation in City A:

- Assuming that the population of city A is m, and there are a total of n communities. The population of each community is equal
- Suppose that the community a in city A first broke out with COVID-19.
- Assume that the infected people in community a mainly live in community a and nearby areas
- All communities have the same size and population density

Based on the assumptions above, the mixed sample detection method in City A is as follows:

- According to the distance from the community a, the n communities in City A are divided into the following 4 categories numbered 1, 2, 3, and 4:
  1. Community a and other communities that the infected person has been passed through.
  2. Other communities in the distance $d \leqslant d_1$ to the community a.
  3. The community with a distance of $d_1 < d \leqslant d_2$ to the community a.
  4. The community whose distance to the community a is $d > d_2$.
- For the 4 types of communities mentioned above, use different q to obtain different grouping people a_i for group testing (for example: for category1 $q = 95\%$, for category2 $q = 99\%$, for category3 $q = 99.9\%$, for category4 $q = 99.99\%$)

- After the test is completed, collect the corresponding E value of each group of tests, and combine the E value with the number of people in the community to find the required number of tests.

5.2 Model solving

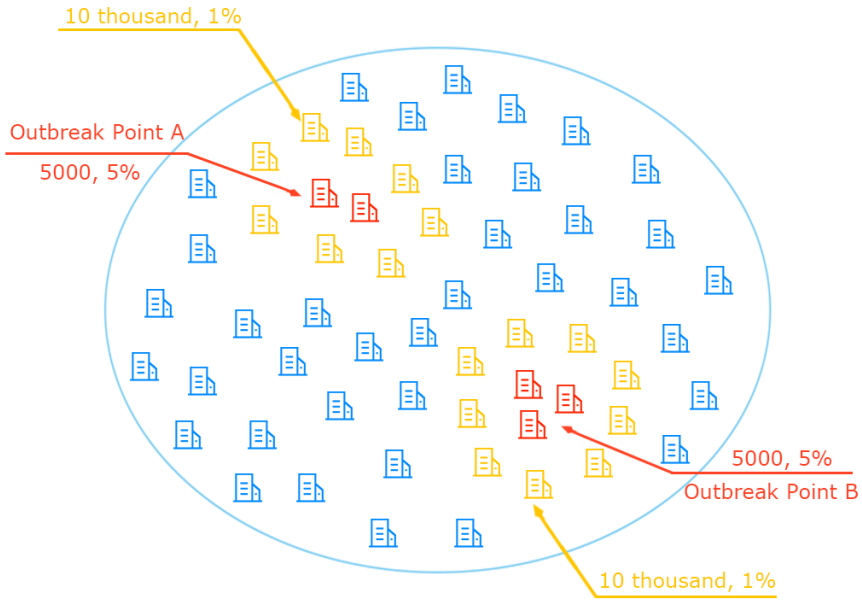

Figure 12: Community Infection Example

Assume that the infected only move in and around the community. The infection situation in the city is as shown in the figure above.From the picture, we can find that the epidemic in the city broke out from two points and spread to the surrounding communities.From the picture, we can find that the COVID-19 epidemic in the city broke out from two points and spread to the surrounding communities. And the epidemic has spread to the community where the infected person lives and other communities nearby. In this example, since the patient has activities in the nearest communities, we combine the first and second types of analysis. Since E here is the weighted average of the mathematical expectation of the number of detections per person in several regions, we can get:

$$\bar{E} = \frac{1}{100}E_{0.95} + \frac{1}{50}E_{0.999} + \frac{97}{100}E_{0.9999} \tag{25}$$

Substitute the parameters obtained through the simulated annealing algorithm to get: $E = 0.008063$, and the E tested according to the traditional method is 0.136656.This method greatly optimizes the detection efficiency.

5.3 Conclusion

Such grouping takes into account the spatial distribution of infectious disease patients and helps to further reduce the number of tests. In addition, when the track of positive patients are known, the division of the four categories (The value of d_i) and the number of categories can be flexibly adjusted according to the possible route of virus transmission, thereby further reducing the amount of testing.

6 Sensitivity analysis

6.1 The influence of parameters p_1 and p_2 on the model

We consider that there are other detection methods like antibody detection and antigen detection in actual testing.Here, we consider that different detection methods will bring different p_1 and p_2, so we analyze the impact of different detection methods on the model as part of the sensitivity analysis.

6.1.1 Antibody Testing

Antibody testing can mainly be used to determine people who are already immune to the virus. It is only suitable for roughly counting the number of infected people and the proportion of infections in areas with a large number of infected people, or as an auxiliary indicator for nucleic acid testing. It is not suitable for direct detection of current infection virus. Its hysteresis characteristics are inconsistent with our precise detection requirements. So we will not discuss this method here

6.1.2 Antigen Testing

Antigen detection has the advantages of short detection time (within 20 minutes) and low detection cost. Positive cases can be quickly detected when the patient's viral load is high during acute infection, but there is a certain gap in sensitivity compared with nucleic acid detection. Because the antigen detection speed is relatively fast but the accuracy is poor, it is more suitable for use in subsequent detections with a large number of groups and a small number of people in the group, or as a supplementary detection method for nucleic acid detection. Antigen detection, because there is no cascade amplification effect of nucleic acid detection, it is basically not affected by the pre-sample. Therefore, there will be basically no false positives in antigen detection.

According to an article [6], a 2×2 contingency table that comprehensively considers the experimental data of the four reagents is as follows

P True / Detecdion	Positive	Negative	Total
Positive	$130(TP)$	$1(FP)$	131
Negative	$116(FN)$	$101(TN)$	207
Total	246	102	348

Table 12: 2×2 contingency table of tested sample in the article [6]

According to 3.2.We can get the parameter $\hat{p}_{n-p}, \hat{p}_{p-n}$ as follows:

$$\hat{p}_{n-p} = \frac{1}{102}, \hat{p}_{p-n} = \frac{116}{246} \tag{26}$$

Replace p_{n-p} and p_{p-n} in 3.2 with $\hat{p}_{n-p}$ and $\hat{p}_{p-n}$, and other parameters remain unchanged, we can find the $\hat{E}$ value corresponding to $\hat{p}_{n-p}$ and $\hat{p}_{p-n}$.

For antigen testing, the parameter a_i calculated by using the simulated annealing algorithm($a_1 = 2062.54389462, a_2 = 1154.52468685, a_3 = 423.930888, a_4 = 76.07392628$) is calculated according to equation**??** when $K = 4$. The value of $\hat{q} = 0.99014$, the value of $\hat{p} = 0.009855$, and the mathematical expectation E at this time is approximately equal to 0.13497.

For nucleic acid detection, according to the literature, [1] [4]$p_{np} = 0.025$ and $p_{pn} = 0.2$,$\hat{q} \approx 0.975$,$\hat{p} \approx 0.025$. At this time, $E \approx 0.3094445169094562$. Since there is a big gap between the two E values, which shows that the model is sensitive to changes in the false positive rate p_{n-p} and the false negative rate p_{p-n}.

6.2 The influence of parameter q on the model

6.2.1 The influence of parameter q on the solution of heuristic algorithm

Stages	Optimal Expectation		
	Simulated Annealing	Genetic Algorithm	Non-Linear Model
1	0.17490093	0.17493990	0.17490090
2	0.04989758	0.05008053	0.04989703
3	0.02523711	0.02527931	0.02522867
4	0.01706420	0.01707519	0.01704290
5	0.01348516	0.01352807	0.01344487

Table 13: Optimal Expectation of tests per person under three different algorithms($\hat{q} = 99.9\%$)

Compared with the situation of (in 4.4, $q = 99.99\%$), the average number of tests per person E has increased significantly. This is consistent with the common sense that the positive rate increases and the number of tests increases.

Stages	Simulated Annealing				
	a_1	a_2	a_3	a_4	a_5
1	23.14276	/	/	/	/
2	81.82794	23.28634	/	/	/
3	170.8594	82.94629	23.61882	/	/
4	247.3794	164.4525	76.90970	21.77930	/
5	353.3163	247.9944	160.2385	71.83592	21.97613

Table 14: Optimal Grouping Scheme under the Simulated Annealing Algorithm($\hat{q} = 99.9\%$)

Stages	Genetic Algorithm				
	a_1	a_2	a_3	a_4	a_5
1	23.65130	/	/	/	/
2	84.68692	23.65130	/	/	/
3	165.5591	85.44987	22.88835	/	/
4	247.9572	169.3738	86.97576	21.97533	/
5	331.8811	222.7800	176.2403	81.79709	26.86202

Table 15: Optimal Grouping Scheme under the Genetic Algorithm($\hat{q} = 99.9\%$)

Stages	Non-Linear Model				
	a_1	a_2	a_3	a_4	a_5
1	23.14530	/	/	/	/
2	81.99162	23.14565	/	/	/
3	166.6976	81.99301	23.14522	/	/
4	255.1531	166.6978	81.99285	23.14531	/
5	334.6486	255.1531	166.6978	81.99284	23.14529

Table 16: Optimal Grouping Scheme under the Non-Linear Model($\hat{q} = 99.9\%$)

Compared with the same table in 4.4, it can be found that when q decreases, the corresponding value of a_i decreases. This is consistent with the fact that with the increase of i in the parameter optimization model, the probability of corresponding q gradually increases, and the fact that a_i decreases accordingly, and also in line with our experience

6.2.2 The influence of parameter q on the solution of information theory model

According to the table4.4, we can find that the groups of a_i derived from the information theory model also decrease with the decrease of q, which is similar to the experience mentioned above.

7 Strengths and weaknesses

7.1 Strengths

- **Uses Widely**
  This model only needs to adjust the parameters p_{p-n} and p_{n-p} and it can be used in multiple detection methods such as nucleic acid detection, antigen or antibody detection. The model considers a variety of factors and has a certain generalization ability. It can effectively give a certain minimum number of detections X at p and the corresponding number of groups a_i at this time.

- **Considered the false detection situation**
  The model considers false negative and false positive cases, reasonably offsets the influence of false detection in the calculation, and estimates the number of detections more accurately.

- **Multiple Detection**
  Due to the limitations of current testing methods, the actual diagnosis of positive patients often requires multiple tests. Our model has taken into account the influence of multiple detections on the number of detections and the false detection rate during the establishment, which has strong practical application value.

- **High Accuracy**
  In the case of small grouping times(≤ 3), the model can achieve a relatively high accuracy rate. Considering that there are not too many groupings under actual conditions, the accuracy of our model is sufficient.

7.2 Weaknesses

- **The parameter value given by the heuristic algorithm used is uncertain**
  Two heuristic algorithms, genetic algorithm and simulated annealing, are used in the parameter determination process for a given number of detections. Due to the uncertainty of the random number generated in the algorithm solving process, the parameter values given by the algorithm also have certain uncertainty.

- **Did not consider the spread of the virus during the test**
  In actual situations, individuals who have been tested during the test may also be infected with the virus. Our model ignores the test time and virus transmission factors, and the predicted results may deviate from the true value.

- **The established multiple detection model is relatively complicated**
  After considering the multiplication relationship of the probability in multiple detections, the formula for the number of detections N is slightly more complicated and not convenient enough to use.

- **Did not consider the case of more repeated detections**
  For the convenience of solving, our model only considers the case of at most two repeated tests. However, in actual situations, due to the low accuracy of the detection method and the physical condition of each person, there may be repeated detections three times, four times, or even more than five times. We ignore the increase in the number of detections caused by these factors.

8 A letter to CDC of City A

Dear CDC of City A,

We have heard the news that a newer wave of covid-19 attacks City A and the number of tests suddenly expands. This news makes us very sorry, and we are also very grateful to you for hiring us to find a solution to the problem. The following is our conclusion.

We first established three models based on general group testing methods. These three models are progressive, and the last model takes into account inspection errors and repeated detections, so it is more usable.

This model comprehensively considers the problems that exist in each test such as the large number of test samples and the high false negative rate, as well as the method of avoiding false detections in the actual test.

This model takes the mathematical expectation E of each person's test times as the objective function. The goal is to find a set of $a_1, a_2 \cdots a_k$ and k to minimize E. In a grouping method, if the value of E reaches the minimum, it means that this grouping method is what we are looking for.

Based on our previous analysis, we use the simulated annealing algorithm, genetic algorithm and related knowledge of information theory are used to optimize the parameters a_i required by the above model. Respectively, given or not given the maximum number of detections for a single sample, the value of a_i that minimizes the total number of detections X is obtained. After that, we also established a sampling detection model that considers the spread of the virus and analyzed the probability related to it.

The above parts together constitute a mixed sample detection model with strong generalization ability and wide application range. And use this model to solve the optimization problem of the parameter a_i and the estimation problem of the total detection times X. This model has certain significance in guiding the detection and prevention and control of COVID-19s epidemics, and improving the utilization of detection reagents.

According to our result, the grouping stages and the number of people in a group have a great relationship with the proportion of positive patients in the population. If the proportion of positive patients in the population is very low(0.01%), in order to reduce the total number of detections, the grouping stages should be increased, and the number of people in each group will be correspondingly large. Conversely, if the proportion of positive patients in the population is high(5%), in order to reduce the total number of tests, the grouping stage of the test will be reduced, and the number of people in each group will be correspondingly reduced. We have given the specific grouping methods and corresponding data in the paper for your reference.

Finally, considering the actual situation, we have further optimized the model. Since the positive patients are not evenly distributed in the population, they usually show the characteristics of cluster distribution. Therefore, the proportion of positive patients in different communities in a city may not be the same, and the same group testing method should not be used for all areas of the city. We divide the various areas of the city into different risk levels, and each risk level uses a different grouping detections method. For areas with high risk levels, increase the amount of detections, and for areas with the highest risk level, reduce the amount of detection. We believe that not all areas need all detections, and sampling detections are enough for low-risk areas. This can reduce sampling, storage and transportation, save a lot of costs, and facilitate the rational allocation of resources.

Hope our suggestions can be helpful to you.

Sicerely yours,
Your friends

References

[1] Andrew N Cohen and Bruce Kessel. False positives in reverse transcription pcr testing for sars-cov-2. *medRxiv*, 2020.

[2] Anton Dekkers and Emile Aarts. Global optimization and simulated annealing. *Mathematical Programming*, 50(1):367–393, 1991.

[3] Christopher R Houck, Jeff Joines, and Michael G Kay. A genetic algorithm for function optimization: a matlab implementation. *Ncsu-ie tr*, 95(09):1–10, 1995.

[4] Lauren M Kucirka, Stephen A Lauer, Oliver Laeyendecker, Denali Boon, and Justin Lessler. Variation in false-negative rate of reverse transcriptase polymerase chain reaction–based sars-cov-2 tests by time since exposure. *Annals of Internal Medicine*, 2020.

[5] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[6] Thomas Weitzel, Paulette Legarraga, Mirentxu Iruretagoyena, Gabriel Pizarro, Valeska Vollrath, Lorena Porte, Rafael Araos, and José M Munita. Comparative evaluation of four rapid sars-cov-2 antigen detection tests using universal transport medium. *Travel Medicine and Infectious Disease*, 2020.

Appendices

Appendix A Code For Simulated Annealing Algorithm

This is a python program that uses the simulated annealing algorithm to solve the optimal value of a_i

```
from matplotlib import pyplot as plt
import random as rd
import numpy as np
import math

lb = 1
hb = 5000

def decr_rand(height,width):
    matrix = np.empty([height, width])
    for i in range(height):
        matrix[i][0] = rd.randint(width, hb)
        for j in range(1, width):
            matrix[i][j] = rd.randint(width - j, matrix[i][j - 1] - 1)
    return matrix


class SimAnnealing:
    def __init__(self, func, iteration=1000, T_ini=100, T_fin=1e-8, alpha=0.99, stages=4
        self.func = func
        self.iteration = iteration
        self.stages = stages
        self.T_ini = T_ini
        self.T_fin = T_fin
        self.alpha = alpha
        self.T_cur = T_ini
        self.procession = {'Res': [], 'T': []}
        self.lx = decr_rand(iteration, stages)

    def Generate(self, lx):
        new_lx = np.empty([self.iteration, self.stages])
        for i in range(lx.shape[0]):
            while True:
                temp = lx[i][0] + self.T_cur * (rd.random() - rd.random())
                if temp < hb and temp > lb: break
            new_lx[i][0] = temp
            for j in range(1, lx.shape[1]):
                while True:
                    temp = lx[i][j] + self.T_cur * (rd.random() - rd.random())
                    if temp < lx[i][j - 1] and temp > lb: break
                new_lx[i][j] = temp
        return new_lx

    def Metropolis(self, res_cur, res_past):
        if res_cur <= res_past:
            return True
        else:
            p = math.exp((res_past-res_cur)/self.T_cur)
```

```
            if rd.random() < p:
                return True
            else:
                return False

    def getOpt(self):
        res_lis = []
        for i in range(self.iteration):
            res_lis.append(self.func(self.lx[i]))
        opt_res = min(res_lis)
        opt_index = res_lis.index(opt_res)
        return opt_res, opt_index

    def solvement(self):
        fin_lx = self.lx[0].copy()
        fin_res = 1
        while self.T_cur > self.T_fin:
            new_lx = self.Generate(self.lx)
            for i in range(self.iteration):
                res_past = self.func(self.lx[i])
                res_cur = self.func(new_lx[i])
                if self.Metropolis(res_cur, res_past):
                    self.lx[i] = new_lx[i]
            opt_res, ind = self.getOpt()
            self.procession['Res'].append(opt_res)
            self.procession['T'].append(self.T_cur)
            self.T_cur = self.T_cur * self.alpha
            if opt_res < fin_res:
                fin_lx = self.lx[ind].copy()
                fin_res = opt_res
        print(f'Final result:{fin_res}.x:{fin_lx}')


def func1(lx, q=0.9999):
    res = 1.0 / lx[0]
    temp = 1.0
    for i in range(1, len(lx)):
        temp = temp * (1 - math.pow(q, lx[i - 1]))
        res = res + temp / lx[i]
    res = res + temp * (1 - math.pow(q, lx[-1]))
    return res


def func2(lx, q=0.9999):
    res = 1.0 / lx[0] * (1 + math.pow(q, lx[0]))
    temp = 1 + math.pow(q, lx[0])
    for i in range(1, len(lx)):
        temp = temp * (1 - math.pow(q, lx[i - 1])) * (1 + math.pow(q, lx[i]))
        res = res + temp / lx[i]
    res = res + temp * (1 - math.pow(q, lx[-1])) * (1 + math.pow(q, lx[-1]))
    return res


proc = SimAnnealing(func2)
proc.solvement()
plt.xlabel('Temperature')
plt.ylabel('Result')
plt.grid()
```

```
plt.xlim(0, 100)
plt.gca().invert_xaxis()
plt.plot(proc.procession['T'], proc.procession['Res'], color='orange', linewidth=1)
plt.show()
```

Appendix B Code For Genetic Algorithm

This is the matlab program that uses the genetic algorithm to solve the optimal value of a_i
This is the population crossover function of genetic algorithm:

```
function scro=crossover(population,seln,pc);
BitLength=size(population,2);%the number of the second dimension of the population
pcc=IfCroIfMut(pc);
if pcc==1
   chb=round(rand*(BitLength-2))+1;
   scro(1,:)=[population(seln(1),1:chb) population(seln(2),chb+1:BitLength)];
   scro(2,:)=[population(seln(2),1:chb) population(seln(1),chb+1:BitLength)];
else
   scro(1,:)=population(seln(1),:);
   scro(2,:)=population(seln(2),:);
end
```

This is the fitness calculation function of genetic algorithm:

```
function [Fitvalue,cumsump]=fitnessfun(population);
global BitLength
global bitlength
global boundsbegin
global boundsend
global KKK
global q

popsize=size(population,1);
for i=1:popsize
    for j=1:KKK
        temp = 0;
        for t = 1:j-1
            temp = temp + bitlength(t);
        end
        x(j)=transform2to10(population(i, temp + 1:temp + bitlength(j)));
    end

    for j=1:KKK
        xx(j)=boundsbegin(j)+x(j)*(boundsend(j)-boundsbegin(j))/(power(2,bitlength(j))-1
    end
    Fitvalue(i)=targetfun(xx);
end
Fitvalue=Fitvalue';

%Calculate the probability of selection
DFitvalue = Fitvalue .^ -1;
fsum=sum(DFitvalue);
Pperpopulation=DFitvalue / fsum;

%Calculate cumulative probability
cumsump(1)=Pperpopulation(1);
```

```
for i=2:popsize
   cumsump(i)=cumsump(i-1)+Pperpopulation(i);
end
cumsump=cumsump';
```

This is the **main program** of genetic algorithm

```
clc;
clear all;
close all;
global BitLength
global bitlength
global boundsbegin
global boundsend
global KKK
global q

KKK=4
q=0.9999
bounds=[0, 5000;0, 2000;0, 600;0, 100];%Set the range of k independent variables

precision=1;
boundsbegin=bounds(:,1);
boundsend=bounds(:,2);


for i = 1:KKK
    bitlength(i)=ceil(log2((boundsend(i)-boundsbegin(i))' ./ precision));
end

BitLength=sum(bitlength);
popsize=50;

Generationnmax=500;  %Generationnmax=1000; %Maximum generation

pcrossover=0.90; %Probability of crossover
pmutation=0.09; %Probability of mutation

population=round(rand(popsize,BitLength));

[Fitvalue,cumsump]=fitnessfun(population);
Generation=1;
while Generation<Generationnmax+1
   for j=1:2:popsize

      seln=selection(population,cumsump);

      scro=crossover(population,seln,pcrossover);
      scnew(j,:)=scro(1,:);
      scnew(j+1,:)=scro(2,:);
      %mutation
      smnew(j,:)=mutation(scnew(j,:),pmutation);
      smnew(j+1,:)=mutation(scnew(j+1,:),pmutation);
   end
   population=smnew;
   %Calculate the fitness of the new population
   [Fitvalue,cumsump]=fitnessfun(population);
   %Record the best fitness and average fitness of the current generation
```

```
    [fmin,nmin]=min(Fitvalue);
    fmean=mean(Fitvalue);
    ymin(Generation)=fmin;
    ymean(Generation)=fmean;
    %Record the best individual of the current generation
    for j=1:KKK
         temp = 0;
         for t = 1:j-1
             temp = temp + bitlength(t);
         end
         x(j)=transform2to10(population(nmin, temp + 1:temp + bitlength(j)));
    end

    %x=transform2to10(population(nmin,:));
    for j=1:KKK
         xx(j)=boundsbegin(j)+x(j)*(boundsend(j)-boundsbegin(j))/(power(2,bitlength(j))-1
    end

    %xx=boundsbegin+x*(boundsend-boundsbegin)/(power(2,BitLength)-1);
    %xmin(Generation)=xx;
    Generation=Generation+1;
end
Generation=Generation-1;
Bestpopulation=xx
Besttargetfunvalue=targetfun(xx)

figure(1);
hand1=plot(1:Generation,ymin);
set(hand1,'linestyle','-','linewidth',1.8,'marker','*','markersize',6)
hold on;
hand2=plot(1:Generation,ymean);
set(hand2,'color','r','linestyle','-','linewidth',1.8,'marker','h','markersize',6)
xlabel('');ylabel('');xlim([1 Generationnmax]);
legend('','');
box off;hold off;
```

This is a random function to judge whether crossover and mutation are needed:

```
function pcc=IfCroIfMut(mutORcro);
test(1:100)=0;
l=round(100*mutORcro);
test(1:l)=1;
n=round(rand*99)+1;
pcc=test(n);
```

This is the function that performs mutation operation in genetic algorithm:

```
function snnew=mutation(snew,pmutation);
BitLength=size(snew,2);
snnew=snew;
pmm=IfCroIfMut(pmutation);
if pmm==1
   chb=round(rand*(BitLength-1))+1;
   snnew(chb)=abs(snew(chb)-1);
end
```

This is the function that performs mutation selection in genetic algorithm:

```
function seln=selection(population,cumsump);

for i=1:2
   r=rand;
   prand=cumsump-r;
   j=1;
   while prand(j)<0
       j=j+1;
   end
   seln(i)=j;
end
```

This is the objective function to be optimized:

```
sfunction y=targetfun(x); %target function
global BitLength
global bitlength
global boundsbegin
global boundsend
global KKK
global q


if KKK == 1
    y=1 ./ x .* (1 + q .^ x) + (1 - q .^ (2 .* x)) .* (1 + q .^ x);
else
    y1 = 1 ./ x(1) .* (1 + q .^ x(1));
    y2 = 0;
    for i = 2:KKK
        temp = 1;
        for j = 1:i-1
            temp = temp .* (1 - q .^ x(j));
        end
        for j = 1:i
            temp = temp .* (1 + q .^ x(j));
        end
        temp = temp * 1 ./ x(i);
        y2 = y2 +temp;
    end

    y3 = 1;
    for j = 1:KKK
        y3 = y3 .* (1-q .^ (2 .* x(j)));
    end
    y3 = y3 .* (1 + q .^ KKK);

     y = y1 + y2 + y3;
end
```

This is the conversion function between binary code and decimal code:

```
function x=transform2to10(Population);
BitLength=size(Population,2);
x=Population(BitLength);
for i=1:BitLength-1
   x=x+Population(BitLength-i)*power(2,i);
end
```

Appendix C Code for Information Theory Model

```
import  numpy as np
p=0.99
d=np.ones(15)
i=0
while(d[i]>=1 ):
        d[i] = np.log(0.5) / np.log(p * (2 ** i) - (2 ** i) + 1)
        i=i+1;
        print("%d:%f\n" % (i, d[i-1]))
```

Appendix D Lingo Code for Non-linear Optimization

```
model:
q=0.9999;

!k==1;
!min=1/x*(1+q^x)+(1-q^(2*x))*(1+q^x);

!k==2;
!min=1/x1*(1+q^x1)+1/x2*(1+q^x2)*(1-q^(2*x1))+(1-q^(2*x1))*(1-q^(2*x2))*(1+q^x2);
!x1>x2;

!k==3;
!min=1/x1*(1+q^x1)+1/x2*(1+q^x2)*(1-q^(2*x1))+1/x3*(1+q^x3)*(1-q^(2*x1))*(1-q^(2*x2))+(1
!x1>x2;
!x2>x3;

!k==4;
min=1/x1*(1+q^x1)+1/x2*(1+q^x2)*(1-q^(2*x1))+1/x3*(1+q^x3)*(1-q^(2*x1))*(1-q^(2*x2))+1/x
x1>x2;
x2>x3;
x3>x4;

!k==5;
!min=1/x1*(1+q^x1)+1/x2*(1+q^x2)*(1-q^(2*x1))+1/x3*(1+q^x3)*(1-q^(2*x1))*(1-q^(2*x2))+1/
!x1>x2;
!x2>x3;
!x3>x4;
!x4>x5;

!k==6;
!min=1/x1*(1+q^x1)+1/x2*(1+q^x2)*(1-q^(2*x1))+1/x3*(1+q^x3)*(1-q^(2*x1))*(1-q^(2*x2))+1/
!x1>x2;
!x2>x3;
!x3>x4;
!x4>x5;
!x5>x6;

end
```

Appendix E Figures for Simulated Annealing Algorithm

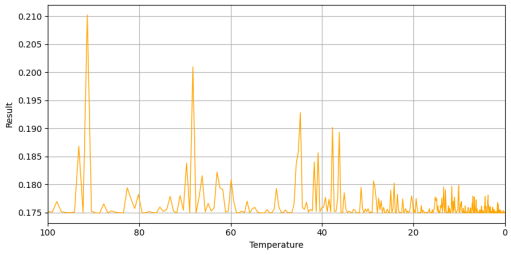

Figure 13: $k = 1$

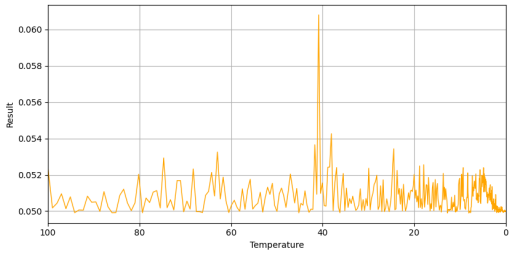

Figure 14: $k = 2$

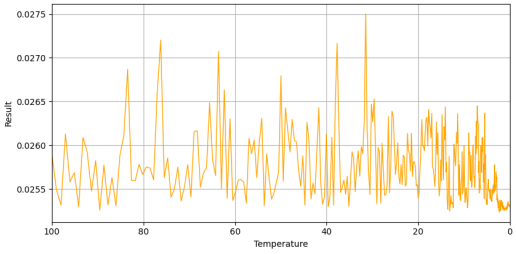

Figure 15: $k = 3$

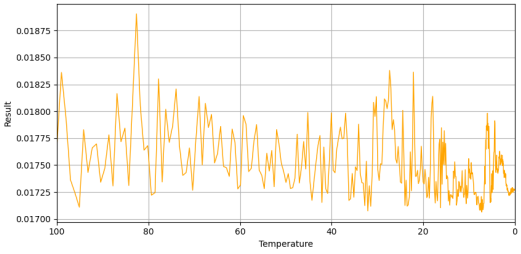

Figure 16: $k = 4$

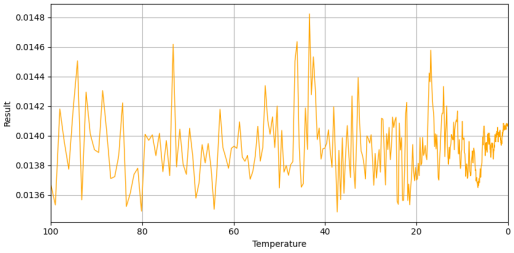

Figure 17: $k = 5$