Non-Adaptive Probabilistic Group Testing

Anant Maheshwari
CS466
Student ID: 20717738
Undergraduate Computer Science, 3B

August 2020

Abstract

This is a survey paper with a main focus of analysing some key Non-Adaptive Probabilistic algorithms from the field of Group Testing, alongside some numerical results that outline the benefits of these algorithms.

1 Introduction

1.1 Background

With the COVID-19 pandemic, we have recently seen a worldwide need to increase testing capacity. However, constraints on the number of available tests in communities has created a need for innovation not only in producing cheaper tests, but also in finding ways to test large populations with the minimal number of tests possible. This has stimulated a lot of interest over the past few months in the field of Group Testing. In fact, Group Testing is already being used to combat the pandemic in Germany and Israel. Recently, the United States Food and Drug Administration (FDA) has been issuing approvals for some versions of Group Testing.

Group Testing is broadly the study of finding defective items in a large set. In the medical infection setting, that implies classifying each member of a population as infected or uninfected, while minimizing the total number of tests. The field of Group Testing has its origins in a single report by Dorfman in 1943 [5], when there was a need to efficiently test a large number of WW2 recruits for Syphilis. Dorfman presented a simple algorithm known as Single Pooling, which could result in a dramatic increase in testing capacity.

Since then, there has been a variety of new results and improvements, creating two independent categories for research in this field. Namely, each Group Testing algorithm is either Probabilistic or Combinatorial, and it is either Adaptive or Non-Adaptive.

In the Probabilistic setting, the infected individuals are assumed to have some known probability distribution, and hence the expected number of tests to identify all the infected

individuals is analyzed. However, Combinatorial Group Testing is the study of algorithms to minimize the worst-case number of tests needed if we do not know the distribution of infected people in advance [6]. Adaptive Group Testing allows flexibility in the algorithm design to perform some tests, and then use the results from previous tests to inform the decision about the next tests to perform [1]. Non-Adaptive Group Testing is precisely enforcing the restriction that our tests must be predetermined (not influenced by previous tests) [9].

In this survey paper, we focus on some interesting and insightful Probabilistic, Non-Adaptive algorithms. We will begin with the original Single Pooling algorithm from Dorfman, and then show a new algorithm called Double Pooling invented in 2020 by Broder and Kumar [2] that improves upon Dorfman's result. Note that we will discuss and analyze Multiple Pooling, where we generalize the Double Pooling algorithm from Broder and Kumar. Then, we will look at the 2011 algorithm from Chan et al., Combinatorial Orthogonal Matching Pursuit [3], that provides us with the opportunity to see how randomized algorithms can be very powerful for Group Testing if we allow some amount of error in our test results. Lastly, we will show some numerical results to illustrate the benefits of these algorithms in terms of decreasing the number of tests needed, thus improving testing capacity.

1.2 Group Testing Problem Formulation

For the Group Testing problem, we have a set of n people, denoted $P_1, P_2, ...P_n$. We are given a testing procedure $\mathsf{Test}(S)$ which given a non-empty subset $S \subseteq \{1, ..., n\}$ returns

1) **negative** if all P_i such that $i \in S$ are infection-free.

2) **positive** if there exists some P_i such that $i \in S$ and P_i is infected.

Our goal is to consider different algorithms that minimize the number of calls to our testing procedure Test, while determining $\forall i \in \{1, ..., n\}$ whether P_i is infected or not.

Note that a naive approach is to simply call our testing procedure Test on each P_i, $i \in \{1, ..., n\}$ individually, which would indeed inform us $\forall i \in \{1, ..., n\}$ whether P_i is infected or not. However, such an algorithm would require n calls to our procedure Test (one call for every member of the population), which motivates our desire to study Group Testing algorithms that can allow us to achieve a lower number of calls to Test.

In practise we would need to consider false negatives and false positives, as well as limits on the amount of repeated tests that can be done using the same sample. We will ignore such limitations for the purpose of our analysis.

2 Group Testing Algorithms and Analysis

In this section, we will present three related approaches for non-adaptive probabilistic group testing, namely Single Pooling, Multiple Pooling (note, Double Pooling is contained within Multiple Pooling), and Combinatorial Orthogonal Matching Pursuit. For Single and Multiple

Pooling, note that in the original 1943 Dorfman paper [5], as well as the recent 2020 Broder and Kumar paper [2], the analysis of these algorithms is performed in the setting where each person arriving has probability p of being infected, independent of all other people. However, in the notes from Michiel Smid [8], we perform the analysis where we know there are k infected people in a population in advance. We will perform the analysis here using both setups, and then compare and discuss the results for expected number of calls to Test for a population of size n.

2.1 Single Pooling

The original result that pioneered the field of Group Testing came from the 1943 Dorfman paper which presented the following algorithm (which was used for syphilis testing in WW2 recruits):

Algorithm 1: Dorfman Single Pooling (1943)

1. Choose a random permutation Π of $\{P_1, ..., P_n\}$. Divide the population into n/s groups of size s. Denote these groups $G_1, ...G_{\lceil n/s \rceil}$.
2. Run the procedure Test on each of the groups, that is run Test(G_i) $\forall i \in \{1, ..., \lceil n/s \rceil\}$.
3. For any group G_i where Test returns positive, run Test on every $P_j \in G_i$ (rerun on everyone in "infected" group G_i)

We analyze the expected number of calls to our Test procedure when running the Dorfman Single Pooling Algorithm using the two different setups as previously mentioned.

Claim 2.1. *Let X denote the number of calls to* Test *in Dorfman Single Pooling. Assume we have a population of size n, and probability p that a given person P_i is infected (independent of all other P_j, $\forall j \in \{1, ..., n\} | j \neq i$). If we use groups of size s then:*

$$E[X] = n/s + n - n(1-p)^s$$

Proof. Following the analysis of Dorfman's Algorithm from Broder and Kumar (as this proof approach is easier to generalize for Double/Multiple Pooling), we consider the expected number of tests a specific person P_i contributes to the total number of tests.

Let X_i denote the number of tests contributed by person P_i ($\forall i \in \{1, ..., n\}$). Then,

$$X = \sum_{i=1}^{n} X_i$$

For a person P_i that is positive (has the infection), we have that they contribute $1/s$ tests to account for their share of the group of size s they belong to, as well as the individual test for P_i (which must occur, since the group of a positive P_i will test positive, and so we will

test each member individually). Hence, a positive P_i contributes an expected $1/s + 1$ tests.

For a person P_i that is negative (infection-free), we have $1/s$ tests to account for their share of the group of size s, but we also need to now consider the probability that a negative P_i requires another call to Test. Note that this would occur if another member in the group of P_i was positive, resulting in retesting the entire group (including P_i) individually. Since we have that a given person is infected with probability p, we have that with probability $(1-p)$ a person is not infected, and so $(1-p)^{s-1}$ denotes the probability that all other members in the group of P_i were not infected. But then, $1-(1-p)^{s-1}$ denotes the probability that at least one of the other members in the group of P_i was infected, and this is precisely the condition that would result in P_i being re-tested. Hence, a negative P_i contributes an expected $1/s + (1-(1-p)^{s-1})$ tests.

But then, since a person is positive with probability p, and negative with probability $1-p$, we have that

$$\begin{aligned} E[X_i] &= p(1/s+1) + (1-p)(1/s + (1-(1-p)^{s-1})) \\ &= p/s + p + (1-p)/s + (1-p) - (1-p)^s \\ &= 1/s + 1 - (1-p)^s \end{aligned}$$

But then, by linearity of expectation,

$$\begin{aligned} E[X] &= \sum_{i=1}^{n} E[X_i] \\ &= \sum_{i=1}^{n} 1/s + 1 - (1-p)^s \\ &= n[1/s + 1 - (1-p)^s] \\ &= n/s + n - n(1-p)^s \end{aligned}$$

□

Claim 2.2. *Let X denote the number of calls to* Test *in Dorfman Single Pooling. Assume we have a population of size n, and we have k infected people in our population. If we use groups of size s then:*

$$E[X] = n + n/s - (n-k)\binom{n-s}{k} / \binom{n-1}{k}$$

Proof. Since we have a fixed number of infected people k in our population, let $I \subseteq \{1, ..., n\}$ denote the indices of the infected people in $\{P_1, ...P_n\}$ (that is, $\forall i \in I, P_i$ is infected). Let N be defined as $\{1, ..., n\} \setminus I$ (that is, $\forall j \in N, P_j$ is uninfected). Note that $|I| = k$.

$$X_i = \begin{cases} 1 & \text{if } P_i \text{ is tested individually} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

We thus have

$$X = n/s + \sum_{i \in I} X_i + \sum_{j \in N} X_j$$

Consider some P_i such that $i \in I$. We have that $E[X_i] = 1$, since we will test any infected member individually as per the algorithm. It remains to consider P_j such that $j \in N$. We wish to determine $E[X_j]$. Note that an uninfected person is only tested individually if there is some other member of the same group that is positive. Hence,

$$E[X_j] = 1 - P(\text{no other member of the group is positive})$$

Now, consider the conditions needed for there to be no group member of P_j that is positive. Since we have k positive people, it must be the case that these k people fall into other groups outside of the $s-1$ potential remaining locations in the group of P_j. Since there are $\binom{n-1}{k}$ potential locations for the k infected people, and we are enforcing that these k infected people are in the remaining $n-s$ locations (not in the same group as P_j), we have that

$$P(\text{no other member of the group is positive}) = \binom{n-s}{k} / \binom{n-1}{k}$$

Hence,

$$E[X_j] = 1 - \binom{n-s}{k} / \binom{n-1}{k}$$

Now, we use linearity of expectation to complete the argument. An important note as that we are leveraging the idea that linearity of expectation does not require independence. It is as such true that if we consider the people one by one, since we are given that there are $|I| = k$ infected people, the probability an individual is tested individually could be influenced by test results of others in the population. However, we can safely ignore this dependency, and leverage the power of linearity of expectation to complete the analysis.

$$\begin{aligned} E[X] &= E[n/s + \sum_{i \in I} X_i + \sum_{j \in N} X_j] \\ &= n/s + \sum_{i \in I} E[X_i] + \sum_{j \in N} E[X_j] \\ &= n/s + \sum_{i \in I} 1 + \sum_{j \in N} (1 - \binom{n-s}{k} / \binom{n-1}{k}) \\ &= n/s + k + (n-k)(1 - \binom{n-s}{k} / \binom{n-1}{k}) \\ &= n + n/s + (n-k)\binom{n-s}{k} / \binom{n-1}{k} \end{aligned}$$

□

Now, we notice that Claim 2.1 can be thought of as the same as Claim 2.2, if we let $p := k/n$ (k infected people in a population of size n), and we consider this for large n (i.e. as $n \to \infty$). The intuition here is that for large enough n, even if we know there are k positive people in the population, knowing the results of one would not create any significant dependency with the expected results of another person (since n is large).

We have now thus shown two different analyses of Dorfman Single Pooling. We will present some numerical results in terms of the improvements Single Pooling provides (towards our goal of minimizing the number of calls to Test) in the Algorithmic Engineering section.

2.2 Multiple Pooling

Broder and Kumar recently invented an improvement on Single Pooling, which extends the ideas of Single Pooling by assigning each person P_i to 2 groups, chosen from 2 random and independent permutations of $\{P_1, ...P_n\}$. Now, if any of the 2 groups a person P_i belongs to tests negative, then that person is clearly negative. However, if both of the groups P_i belongs to tests positive, then we test P_i individually. It is interesting to notice here that this idea can be thought of as leveraging the Power of Two Choices paradigm [7], where we are now providing 2 independent random permutations to perform our grouping scheme on. This increases the probability that a negative person ends up in a group of negative people for at least one of the 2 tests (which would thus allow us to avoid retesting a negative individual, a key factor in decreasing the number of tests).

We formalize the algorithm below, but take this idea directly to c pools, where we take c random permutations of our population and thus assign each person P_i to a group of size s in each of the c pools. Again, it still remains true that if any pool a person P_i belongs to tests negative, then we have that P_i is negative, otherwise if all pools P_i belongs to test positive we would test P_i individually.

Note that when $c = 2$, this is precisely the Broder and Kumar Double Pooling Algorithm.

Algorithm 2: Multiple Pooling

1. Choose c random permutations $\Pi_1, ..., \Pi_c$ of $\{P_1, ..., P_n\}$. For each permutation Π_i ($\forall$i $\in \{1, ..., c\}$, divide the population into n/s groups of size s. Denote these groups $G_1^i, ...G_{\lceil n/s \rceil}^i$, for each pool(permutation) $i \in \{1, ..., c\}$.
2. Run the procedure Test on each of the groups, that is run Test(G_i^q) $\forall i \in \{1, ..., \lceil n/s \rceil\}$ and $\forall q \in \{1, ..., c\}$.
3. For any person P_i, if all of the c groups P_i belong to test positive, then we test P_i individually.

We will again analyze this using two setups, as we did with single pooling.

Claim 2.3. *Let X denote the number of calls to* Test *in Multiple Pooling. Assume we have a population of size n, and probability p that a given person P_i is infected (independent of*

all other P_j, $\forall j \in \{1, ..., n\} | j \neq i$). If we use groups of size s and we use c pools for Multiple Pooling, then:

$$E[X] = cn/s + pn - n(1-p)(1-(1-p)^{s-1})^c$$

Proof. Similar to single pooling, we consider the expected number of tests a specific person P_i contributes to the total number of tests.

Let X_i denote the number of tests contributed by person P_i ($\forall i \in \{1, ..., n\}$). Then,

$$X = \sum_{i=1}^{n} X_i$$

We have that a positive P_i contributes an expected $c/s+1$ tests, to account for their part of the c groups they belong to, and then 1 test since any positive P_i will be tested individually.

For a person P_i that is negative, we have c/s tests to account for their share of the c groups of size s they belong to. We need to now consider the probability that a negative P_i requires another call to Test (i.e is tested individually). Note that this would occur if another member in the group of P_i was positive for each of the c groups P_i belongs to. $(1-p)^{s-1}$ denotes the probability that all other members in the same group as P_i in one of the pools test negative. But then, $1-(1-p)^{s-1}$ denotes the probability that at least one of the other members in the group of P_i was positive in that same pool. But then since we have c pools, the following expression gives the probability that for all of the pools, at least one group member of P_i tested positive:

$$\prod_{i=1}^{c}(1-(1-p)^{s-1}) = (1-(1-p)^{s-1})^c$$

This is precisely the condition that would result in P_i being re-tested. Hence, a negative P_i contributes an expected $c/s + (1-(1-p)^{s-1})^c$ tests.

But then, since a person is positive with probability p, and negative with probability $1-p$, we have that

$$\begin{aligned} E[X_i] &= p(c/s+1) + (1-p)(c/s + (1-(1-p)^{s-1})^c) \\ &= (cp)/s + p + c/s - (cp)/s + (1-p)(1-(1-p)^{s-1})^c \\ &= c/s + p + (1-p)(1-(1-p)^{s-1})^c \end{aligned}$$

But then, by linearity of expectation,

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n} E[X_i] \\
&= \sum_{i=1}^{n} c/s + p + (1-p)(1-(1-p)^{s-1})^c \\
&= n[c/s + p + (1-p)(1-(1-p)^{s-1})^c] \\
&= cn/s + pn + n(1-p)(1-(1-p)^{s-1})^c
\end{aligned}
$$

□

Claim 2.4. *Let X denote the number of calls to* Test *in Multiple Pooling. Assume we have a population of size n, and we have k infected people in our population. If we use groups of size s and we use c pools for Multiple Pooling, then:*

$$E[X] = cn/s + k + (n-k)(1 - \binom{n-s}{k} / \binom{n-1}{k})^c$$

Proof. Similar to Single Pooling, let I $\subseteq$\{1,...,n\}be such that $\forall$i$\in I$,P$_i$ is infected. Let N be defined as $\{1, ..., n\} \setminus I$ (that is, $\forall$j$\in N$,P$_j$ is uninfected). Note that $|I| = k$.

$$X_i = \begin{cases} 1 & \text{if } P_i \text{ is tested individually} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We thus have

$$X = cn/s + \sum_{i \in I} X_i + \sum_{j \in N} X_j$$

Consider some P_i such that $i \in I$. We have that $E[X_i] = 1$, since we will test any infected member individually as per the algorithm. Now, it remains to consider $E[X_j]$ for some $j \in N$. That is, we want to consider the probability that for some uninfected person P_j, for each of the c groups P_j belongs to, at least one other person in that group is infected. Define the event

$$A_l = \text{none of the other } s-1 \text{ people in the same group as } P_j \text{ in } \Pi_l \text{ are infected}$$

But then, notice that $X_j = 1$ if and only if the event $\bigwedge_{l=1}^{c} \overline{A_l}$ occurs. As well, from our Single Pooling analysis, we have that $\forall l \in \{1, ..., c\}, P(\overline{A_l}) = 1 - \binom{n-s}{k} / \binom{n-1}{k}$. Hence,

$$
\begin{aligned}
E[X_j] &= P(X_j = 1) \\
&= P(\bigwedge_{l=1}^{c} \overline{A_l}) \\
&= \prod_{l=1}^{c} P(\overline{A_l}) \\
&= \prod_{l=1}^{c} (1 - \binom{n-s}{k} / \binom{n-1}{k}) \\
&= (1 - \binom{n-s}{k} / \binom{n-1}{k})^{c}
\end{aligned}
$$

Then, by linearity of expectation:

$$
\begin{aligned}
E[X] &= E[cn/s + \sum_{i \in I} X_i + \sum_{j \in N} X_j] \\
&= cn/s + \sum_{i \in I} E[X_i] + \sum_{j \in N} E[X_j] \\
&= cn/s + \sum_{i \in I} 1 + \sum_{j \in N} (1 - \binom{n-s}{k} / \binom{n-1}{k})^{c} \\
&= cn/s + k + (n-k)(1 - \binom{n-s}{k} / \binom{n-1}{k})^{c} \\
&= cn/s + k + (n-k)(1 - \binom{n-s}{k} / \binom{n-1}{k})^{c}
\end{aligned}
$$

□

We can similar intuition as we had for Single Pooling to relate these two analysis (i.e Claim 2.4 and 2.5) by setting $p := k/n$ and considering the expressions for large n (as $n \to \infty$). Note that we describe some numerical results of the benefits of Multiple Pooling in the Algorithmic Engineering section.

2.3 Combinatorial Orthogonal Matching Pursuit

We will now introduce a new idea where in our setup, we will eliminate the requirement that the Group Testing algorithm must be certain to succeed. That is, we will permit the algorithm to have some small probability of error, where it could classify an individual incorrectly (e.g. false positive or false negative). Note that we are not saying here that

our Test procedure has any probability of error (we are still assuming that this procedure is always correct), but rather the Group Testing algorithm we are describing which calls Test is now allowed to have a small probability of incorrect classification. Allowing this small probability of error allows us to design randomized algorithms that can greatly improve the results from Single Pooling and Multiple Pooling.

With this setup in mind, we will describe the Chan et al. 2011 probabilistic algorithm, Combinatorial Orthogonal Matching Pursuit ("COMP") [3], and analyse the number of calls to Test using this algorithm. We will also aim to simplify the preprocessing and setup needed for this algorithm, by avoiding constructing the testing matrix representation (as per Chan's paper) and instead simply using our input parameters from Multiple Pooling to describe and analyze the algorithm.

Using a similar setup as Multiple Pooling, we have k is the number of infected people in the population of size n, where we have people $P_1, ..., P_n$. The main idea of this algorithm is instead of dividing our population to groups of a fixed size, we now fix the total number of test T we will perform (which we will bound in the analysis). We assign each member of our population to each of the tests with probability p. Note that p is not the infection rate as it was in previous sections, it is now the probability of assigning a member of our population to a given test. We formalize the algorithm below.

Algorithm 3: Combinatorial Orthogonal Matching Pursuit

1 For each of the people in our population P_i ($i \in \{1, ..., n\}$), include them in the input set $S \subseteq \{1, ..., n\}$ for Test T_j with probability p ($\forall$j$\in\{1,...,$T$\}$)

2 For each person P_i ($i \in \{1, ..., n\}$), if every Test T_j such that $i \in S$ (where S is the input set for test T_j) is positive (has the infection), then the person P_i is declared positive. Otherwise, the person is declared negative (infection-free).

Claim 2.5. *The COMP algorithm requires no more than $T = ek(1+\delta)\ln(n)$ calls to* Test *to find up to k infected people in a population of size n with probability of error no more than $n^{-\delta}$*

Proof. We begin by noting that the COMP algorithm would never produce false negatives, as if a person is declared negative, it means at least one test including that person was negative (and so by the correctness of Test, the person is indeed negative). However, there is a possibility of false positives, since if for every test that a negative person is included in, there is at least one positive person in that test as well, then the negative person will be reported as a positive (hence a false positive). Our proof approach is thus to minimize the probability of having false positives.

Let j be the index of a negative person, and let $1, ..., k$ be the indices of all of the positive people. That is, P_j is negative, and $P_1, ..., P_k$ are all positive. For a given test T_i ($i \in \{1, ..., T\}$) if we include P_j in the test, and we do not include $P_1, ..., P_k$ in the test (since they are positive), then this test would ensure P_j is not a false positive. That is, a specific test would ensure P_j is reported as negative with probability $p(1-p)^k$, since we include P_j

in the test with probability p, and we do not include the k infected people with probability $(1-p)^k$. But then, the probability that a test T_i ($i \in \{1, ..., T\}$) "hides" P_j, that is the probability T_i returns positive because at least one of $\{P_1, ..., P_k\}$ were also included in T_i, is $1 - p(1-p)^k$. Now, the probability that this occurs for all T tests (which would ensure P_j ends up being a false positive) is $(1 - p(1-p)^k)^T$. Let P_e be the probability of having false positives. Taking a union bound over all the $(n-k)$ uninfected people gives

$$P_e \leq (n-k)(1-p(1-p)^k)^T$$

From Chan's paper, we have that optimizing this expression with respect to p suggests choosing $p = 1/k$. We also set T as $\beta k \ln(n)$. Hence, we have

$$P_e \leq (n-k)(1 - \frac{1}{k}(1-\frac{1}{k})^k)^{\beta k \ln(n)}$$

The next step in this analysis from Chan's paper seems to have an error in the calculation. $(1-\frac{1}{k})^k \leq e^{-1}$ is a known bound, however using this bound here should imply that $P_e \geq (n-k)(1-\frac{1}{k}(e^{-1}))^{\beta k \ln(n)}$, however in the analysis Chan claims that $P_e \leq (n-k)(1-\frac{1}{k}(e^{-1}))^{\beta k \ln(n)}$, which seems to be incorrect. However, we proceed assuming this result holds to complete this proof.

$$\begin{aligned}
P_e &\leq (n-k)(1 - \frac{1}{k}(1-\frac{1}{k})^k)^{\beta k \ln(n)} \\
&\leq (n-k)(1-\frac{1}{ke})^{\beta k \ln(n)} \\
&\leq (n-k)(e^{(-\frac{1}{ke}\beta k \ln(n))}) \\
&= (n-k)(n^{(-\frac{1}{e}\beta)}) \\
&= n^{(1-\frac{1}{e}\beta)} - k(n^{(-\frac{1}{e}\beta)}) \\
&\leq n^{(1-\frac{1}{e}\beta)}
\end{aligned}$$

But then, we have setting $\beta = (1+\delta)e$ gives

$$\begin{aligned}
P_e &\leq n^{(1-\frac{1}{e}(1+\delta)e)} \\
&\leq n^{(1-(1+\delta))} \\
&\leq n^{-\delta}
\end{aligned}$$

Hence, if the number of tests T is at least $(1+\delta)ek\ln(n)$, then we achieve the desired probability of error $n^{-\delta}$. □

We expect that this algorithm would allow us to significantly reduce the number of calls to Test compared with Single Pooling and Multiple Pooling assuming the number of infected people k is not too large. Namely, if $k \in o(\frac{n}{\ln n})$, the number of calls to Test for COMP $\in o(n)$. We will try to verify that this algorithm indeed does better than Single Pooling on a numerical example in the Algorithmic Engineering section.

3 Algorithmic Engineering

We present here some numerical results that illustrate how the previously discussed algorithms are able to decrease the number of required calls to Test. We will also numerically show the ideal group sizes for Single Pooling and Multiple Pooling by providing some sample calculations using different group sizes.

3.1 Single Pooling

We begin by calculating the expected number of calls to Test for a few different group sizes when using the Dorfman Single Pooling algorithm.

We begin by using Claim 2.1, with $n = 1000$ and $p = 0.01$. Recall that in this version, we are given a probability p that a given person is infected. Recall that X denotes the number of calls to Test. We get:

s(group size)	E[X]
5	249.01
10	195.618
15	206.608
20	232.093

Now, we use Claim 2.2, with $n = 1000$ and $k = 10$. Recall that in this version of the problem, we are given a fixed number of infected people k. We then have:

s(group size)	E[X]
5	249.106
10	196.032
15	207.529
20	233.682

Notice that we get very similar results between using Claim 2.1 and Claim 2.2, which is what we would expect since as we mentioned in the Algorithms and Analysis section, setting $p := k/n$ and using large n would make these two analysis approximately the same.

Also notice that as we can see in the tables, it appears that for the input parameters chosen ($n = 1000, p = 0.01, k = 10$), we see an optimal group size of $s \approx 10$.

We can already notice a substantial improvement where we can have ≈ 196 calls to Test just with Single Pooling, as opposed to 1000 calls to Test if we were to simply test everyone individually.

3.2 Multiple Pooling

We proceed in a similar way to highlight the improvements that can be achieved with Multiple Pooling. We will focus on Double Pooling, Triple Pooling, and Quadruple Pooling in these

numerical results. The goal here is to give an idea of the improvements and limitations of each of these algorithms numerically.

We will use Claim 2.4, with $n = 1000$ and $k = 10$. We do this for Double, Triple, and Quadruple Pooling, that is setting $c = 2, 3, 4$. We will leave out the equivalent analysis using Claim 2.3 (where we would set $p = 0.01$), as this gives approximately the same results as using Claim 2.4 (just as we saw in the previous section) and so would not be insightful. We also include Single Pooling ($c = 1$) from the previous section to help visually compare between the algorithms. We get:

s(Group Size)	Single Pooling	Double Pooling	Triple Pooling	Quadruple Pooling
5	249.106	411.545	610.061	810.002
10	196.032	217.476	310.65	410.056
15	207.529	160.631	212.287	276.969
20	233.682	140.47	165.346	210.938
25	264.571	136.506	140.08	172.185
30	296.941	141.633	126.643	147.597
35	329.439	152.601	120.822	131.663
40	361.421	167.627	120.487	121.701
45	392.561	185.6	124.405	116.264
50	422.689	205.762	131.784	114.507
55	451.717	227.557	142.062	115.89
60	479.607	250.56	154.809	120.043

We highlight in green the optimal expected number of calls to Test, based on our visual inspection using the input parameters $n = 1000, k = 10$, and for each of Single, Double, Triple, and Quadruple Pooling. Note that each entry of the table corresponds to the expected number of tests required for the group size s given by the row and the algorithm given by the column. We see that we can achieve ≈ 196 calls to Test for Single Pooling, ≈ 137 calls for Double Pooling, ≈ 121 calls for Triple Pooling, and ≈ 115 calls for Quadruple Pooling, which shows that these algorithms can result in a very significant reduction in the number of required tests.

Interestingly, we also notice that the approximate optimal group sizes appear to increase as we increase the number of pools. This makes sense from our analysis, since if we consider the expression from Claim 3.4:

$$E[X] = cn/s + k + (n-k)(1 - \binom{n-s}{k} / \binom{n-1}{k})^c$$

We see that as we increase c (the number of pools), the cn/s term begins to dominate, and the $(n-k)(1 - \binom{n-s}{k}/\binom{n-1}{k})^c$ begins to decrease (since we are raising a fraction to an exponent, and increasing the exponent). Hence, since we wish to minimize the dominating cn/s term, it makes sense that we would end up increasing s proportionally to the amount we increase c, as our goal is to minimize $E[X]$.

More intuitively, for Multiple Pooling with a larger number of pools (e.g. Quadruple Pooling), we have that a negative (infection-free) person has a much lower chance of being retested than in Single Pooling, since our retesting conditions are stronger (the person must have belonged to a positive group in all of the pools). Hence, we see the $(n-k)(1-\binom{n-s}{k}/\binom{n-1}{k})^c$ term decreasing (since it counts the expected number of negative people being retested). However, for a larger number of pools our main bottleneck now becomes the initial group sizes, as this decides the amount of groups that must be tested initially (the cn/s term). Hence, we want to increase the size of the groups (s) such that we can decrease the amount of initial groups tested.

3.3 Combinatorial Orthogonal Matching Pursuit

We briefly discuss some numerical results for this algorithm, to provide a reference point compared with Single Pooling and Multiple Pooling. Using the same parameter values, where $n = 1000$, $k = 10$, we see that we need to have at least $(1+\delta)e10\ln(1000)$ calls to Test, to achieve probability of error $1000^{-\delta}$. To illustrate some benefits and tradebacks, setting $\delta = 1$ would inform us that using COMP we need $(1+1)e10\ln(1000) \approx 376$ tests to achieve probability of error $1000^{-1} = 0.001$. But then, if we set $\delta = 0.001$, we would need at least ≈ 188 calls to Test and we would have at least one false positive with probability ≈ 0.99. We can see that there is a tradeoff between achieving a low error probability and achieving a low number of tests. Also interestingly, although COMP (≈ 188 calls to Test if we allow a high probability of a false positive) does better than Single Pooling (optimal ≈ 196 calls to Test), it actually does worse than Multiple Pooling (e.g. Quadruple Pooling has optimal ≈ 115 calls to Test).

4 Conclusion and Motivation for Future Research

We have seen in this survey some fascinating algorithms that apply ideas from Probability and Randomized Algorithms to the Group Testing problem. We also showed two different ways to analyze Single Pooling and Multiple Pooling, giving us a deeper insight into these algorithms. We saw that these ideas can result in a massive improvement to the amount of testing needed to find infected people in a population. Surprisingly, Quadruple Pooling allowed us to test 1000 people using only ≈ 115 tests. We also saw that allowing some false positives led to an elegant randomized algorithm (COMP) for Group Testing.

There are many future directions and questions this review has raised that would be interesting to explore further. For example, we discussed Multiple Pooling with c pools, and we saw huge improvements in the Algorithmic Engineering section when $c = 4$ (Quadruple Pooling). This raises the question, what is the largest value of c for which we could still see improvements (i.e. is there some asymptotic number of pools where the returns start to diminish)? We saw some numerical values that hinted at the optimal group sizes for each of Single, Double, Triple, Quadruple Pooling, but it would be interesting to review the literature that explores deriving an expression for the optimal group sizes (the Broder and

Kumar paper describes using the Lambert W Function for this purpose, there is actually a paper which involved researchers from the University of Waterloo on this function that it would be interesting to gain ideas from [4]). Furthermore, we saw a relationship between increasing the number of pools and an increase in the optimal group size, could we derive an expression that relates these quantities precisely? Furthermore, could we analyze these algorithms after introducing some error to our Test procedure (since in practise testing protocols could have some error probability, and it is possible that pooling samples could increase this probability)?

It has been a lot of fun to explore this topic, and I look forward to seeing the impact Group Testing has on the current pandemic. It is truly fascinating to see how these core ideas from Probability and Randomized Algorithms are both elegant in a theoretical sense, but also very applicable to real-world problems.

References

[1] G. K. Atia and V. Saligrama. Boolean compressed sensing and noisy group testing. *IEEE Transactions on Information Theory*, 58(3):1880–1901, 2012.

[2] Andrei Z. Broder and Ravi Kumar. A note on double pooling tests. *CoRR*, abs/2004.01684, 2020.

[3] C. L. Chan, P. H. Che, S. Jaggi, and V. Saligrama. Non-adaptive probabilistic group testing with noisy measurements: Near-optimal bounds with efficient algorithms. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1832–1839, 2011.

[4] Robert Corless, Gaston Gonnet, D. Hare, David Jeffrey, and D. Knuth. On the Lambert w function. *Advances in Computational Mathematics*, 5:329–359, 01 1996.

[5] Robert Dorfman. The detection of defective members of large populations. *Ann. Math. Statist.*, 14(4):436–440, 12 1943.

[6] D. Du and F. Hwang. *Combinatorial Group Testing and Its Applications*. Applied Mathematics. World Scientific, 2000.

[7] Michael David Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing*. PhD thesis, 1996. AAI9723118.

[8] Michiel Smid. Group testing. unpublished.

[9] Wikipedia contributors. Group testing — Wikipedia, the free encyclopedia, 2020. [Online; accessed 14-August-2020].