COVID-19 Group Testing Optimization Strategies

Mengfan (Daisy) Chen, Jeffrey Chu, Ethan Dinh-Luong, Vincent Lee

Section B09

Mentor: Arya Mazumdar

I. ABSTRACT

The COVID-19 pandemic that has persisted for more than two years has been combated by efficient testing strategies that reliably identifies positive individuals to slow the spread of the pandemic. Opposed to other pooling strategies within the domain, the methods described in this paper prioritize true negative samples over overall accuracy. In the Monte Carlo simulations, both nonadaptive and adaptive testing strategies with random pool sampling resulted in high accuracy approaching at least 95% with varying pooling sizes and population sizes to decrease the number of tests given. A uniform disjoint method increased the performance by 2% when working with smaller pool sizes and populations. A split tensor rank 2 method attempts to identify all infected samples within 961 samples, converging the number of tests to 99 as the prevalence of infection converges to 1%.

II. INTRODUCTION

COVID-19 rapid tests have been the primary method of identifying positive individuals to slow the spread of the pandemic, but this process is inefficiently rate-limited by not only the thousands of tests run each day, but also the resource cost to run these samples individually. Each rapid test can take up to 6 hours to run in the laboratory[1], which can quickly be delayed based on the sheer number of daily tests given each day. Instead of running individual tests, a group-testing schema, proposed by Robert Dorfman during the Second World War, combined multiple individual samples to form one cohesive pool that is tested once[2]. If a positive test is returned, one test is wasted and the group would be retested again, but if a negative test is returned, several tests are conserved as every member in the pool is labeled as negative.

Although this pooling method conserves tests, researchers are determining a way to efficiently optimize testing strategies in an effort to reduce the number of tests while retaining high accuracy. Such papers demonstrating algorithms such as Tapestry[3] and 2-Step Adaptive Pooling [4] demonstrate two prominent pooling strategies: nonadaptive and adaptive pooling schemes. In a nonadaptive scheme, the entire pooling strategy from start to finish is decided prior

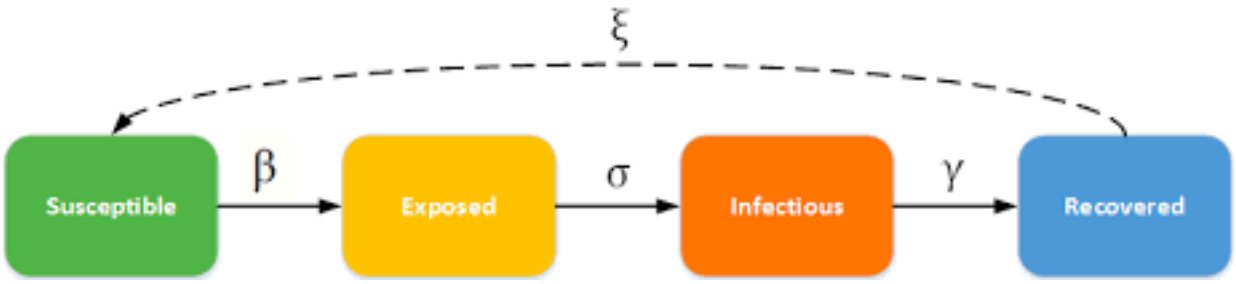
to analyzing the samples, while in an adaptive pooling method, the pooling strategy changes depending on the results of the prior results.

Another strategy explored in this report is a tensor pooling strategy. Tensors (mathematical objects relaying arrays to the coordinates in space) are used in this method to compute the pooling matrix of our samples. An example of this is using a tensor of rank 2 (a matrix) with a sample size of 64. The tensor will be of size $ceiling(\sqrt[r]{n})$ by $ceiling(\sqrt[r]{n})$ where n is the number of samples and r is the rank of the tensor, thus creating a tensor with dimensions 8 by 8. By placing samples in order inside our tensor, each sample has a unique row and column combination. Thus, we can consider each row and column as a pool within our pooling matrix.

In addition, a graphing strategy was tested in this report. Because real COVID data is limited, how credible is the result of this model on simulated data? If the model is tested on actual data in the future, how different will this result be from the existing data result?

The most significant difference between simulated and real data is that all simulated test data are independent: there is no connection between them. In other words, their chances of being pooled and infected are unrelated. Obviously, in reality, this is untrue. Each person has their own social circle and social network. When a person is infected, they are most likely to infect someone within their social network. Unlike the simulated data where everyone is independent of each other, the reality is that when a person is infected, the probability of infecting a family member should not be the same as a randomly selected person outside their social network. Therefore, a social network model of COVID-19 was created for more practical model testing and training in this report. To combine network information with testing methods, the network was analyzed to prioritize critical nodes and high-risk nodes for contracting COVID-19.

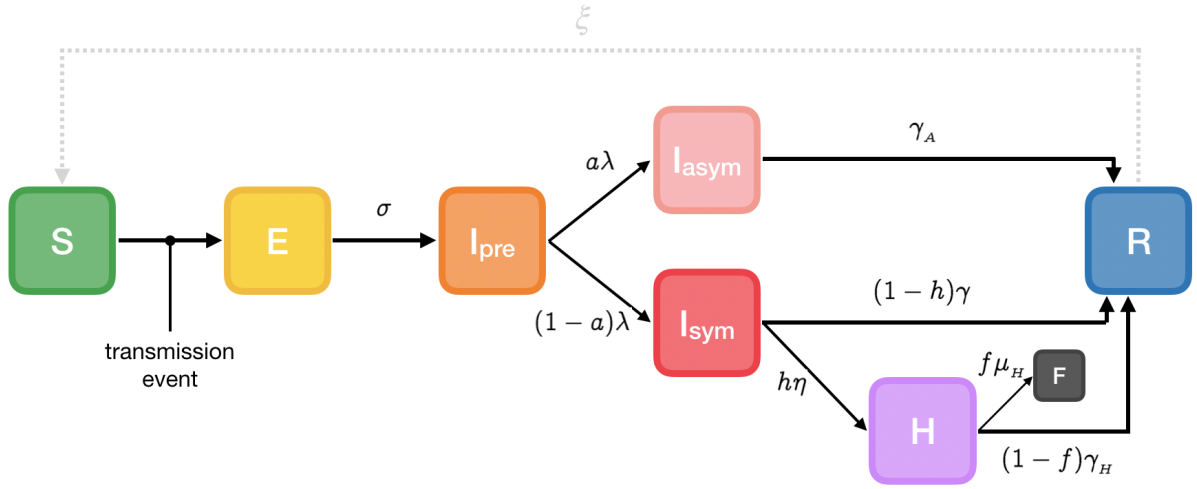
In our project, we tried the SEIR infectious disease model (Figure 1) at first. This model divides the population into four categories: Susceptible (S), Exposed (E), Infectious (I), and Recovered (R) groups [6]. To explore this graphing strategy, we use degree centrality to assess critical nodes, and a degree centrality testing method is used to find out the positives.



[Figure 1: Disease states of SEIR model]

To advance our project, we employed the extended-SEIR model which divided the subpopulation Infectious group (I) into pre-symptomatic (Ipre), asymptomatic (Iasym), symptomatic (Isym), and hospitalized (H) (Figure 1.1). We prefer extended model because this takes into account the latent period of the virus and different transmission rates in different symptomatic stages in Covid-19: the symptomatic individual has a different degree of contagion than asymptomatic individual, and asymptomatic people, who do not have the symptom, will not go into the hospitalized stage.

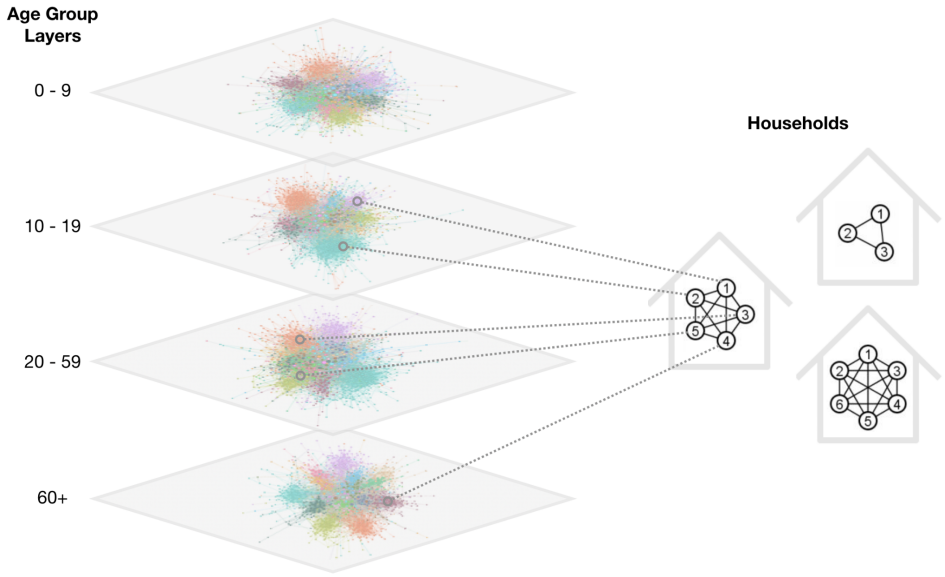
In this model, there are eight states of individuals in the spread of the disease: when a person is exposed to the virus, they change from the susceptible state (S) to the exposed state (E). When the virus develops in the body (Ipre) for a while (incubation period), they will be infected and become a patient. However, the COVID-19 patient may develop symptoms (Isym) or not (Iasym). Symptomatic patients with severity will be hospitalized (H) and will either recover (R) or die (D). On the other hand, Mild patients will recover after a while.



[Figure 1.1: Disease states of extended-SEIR model]

Besides the infectious disease model, the community contact and demographic connection is also considered in this project. The disease model is a good network for simulating the spreading processes of the disease, but it fails to address the complex social connections among different communities. For example, the level of contacts or connections within the households should be different from that of out-of-household contacts, as household connections are more intimate than others. Therefore the success rate of disease transmission should not be the same in these two places. The success rate of transmission at home should be much higher than that of other social places, like schools and workspaces.

To address this problem, a demographic model is employed in this project (Figure 1.2). This model not only includes the community-level connections, but also other realistic demographic properties like age-stratification and household sizes, calibrated with the US demographic statistics. Moreover, this demographic model also generates separate layers of network to mimic out-of-household regular contacts in different age groups. For example, there is a separate contacting network for people whose age is between 10 and 19 which represents secondary school community contacts. With these demographic information, combined with infectious disease network information, we create a machine learning pooling strategy. The more detailed explanation is discussed in the method section.



[Figure 1.2: Households and age group layers of the demographic model]

III. METHODS

Pooling Strategies

To simulate COVID testing, a matrix-based analysis was used as the primary testing strategy. Given $\boldsymbol{n}$ samples and $\boldsymbol{m}$ tests, an $\boldsymbol{n}$ x $\boldsymbol{m}$ matrix denoted which samples belonged to a pooled test, where each $\boldsymbol{m}$ row represented the pooled test, where if an individual $\boldsymbol{n}$ was in the pooled test ($\boldsymbol{x_{mn} = 1}$) or not in the pool test ($\boldsymbol{x_{mn} = 0}$). If at least 1 person in pool $\boldsymbol{m}$ is COVID-19 positive, the output-tested vector would return a 1 (specifically $\boldsymbol{y_m = 1}$), indicating the entire pool contains a positive sample. Conversely, a 0 (specifically $\boldsymbol{y_m = 0}$) indicated a negative pooled sample, where all samples in the pool are declared negative and released from further testing. In a nonadaptive strategy, this was all done in one step, where all tests were run at the same time to determine its positive candidates. In an adaptive strategy, tests were separated into steps, where the pooling strategies in proceeding steps changed based on the prior result. In our method, an adaptive strategy follows that if a test is negative, all samples within that row are removed from the tests, while all the remaining positive candidates are tested once again but under different parameters given by the remaining sample size.

To ensure that samples were equally tested amongst the simulation, disjoint pools were adapted into both the nonadaptive and adaptive strategies. Rather than randomly pick the samples at the start of each test, all samples were shuffled, then divided according to pool sizes. Given a pool size s, sample size $\boldsymbol{n}$, and starting test number $t = n / s$, this guarantees that in the first t tests, every sample is tested only once and that there is no overlap between the first t tests. After all samples are run equally in the first t tests, the remaining pool (whether nonadaptive or adaptive pooling) is reshuffled and divided once again for the following t tests.

In both strategies, infection rate and the number of tests were varied to determine an optimal condition for further analysis, while the number of samples and pooling size were kept constant. Given a large sample size n and rate of infection r, the optimal pooling size for both strategies was equal to $s = 1/r$. With this pooling size, there is a high probability that there is at least one negative pool, allowing for a large portion of the sample size to be declared negative in a minimal number of tests.

Although this may seem to optimize our pooling size, in reality complications arise from these high pooling sizes. Large pools above 32 samples may dilute the samples where positive COVID samples are not detectable through normal laboratory means[5]. In order for COVID to

be detectable in larger pool sizes, the laboratory would require longer lengths to examine, which is even less optimal. To reflect this reality, pool sizes were limited to a maximum of 32, as positive COVID samples were detectable at this size, but were less favorable with larger sizes[5].

To improve identifying negative samples, least squares regression algorithm was run, where the pooling problem was represented by the form $||Ax - b||_2^2$ where the prediction x-vector that determined positive and negative tests was minimized.

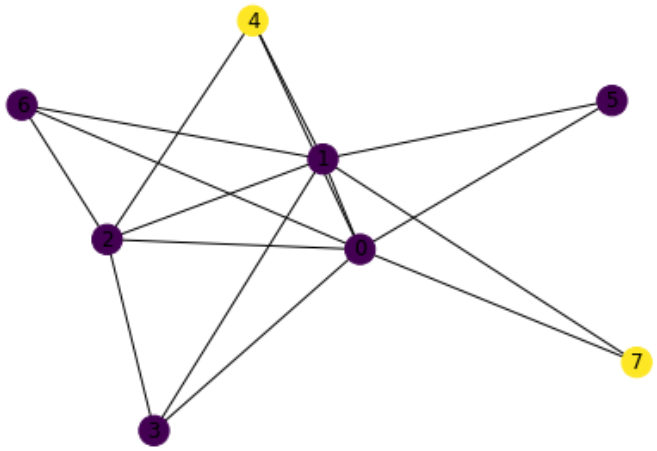
Tensor Pooling

When generalizing the tensor method, each tensor of rank r-1 was considered as a pool within the pooling matrix. This allowed to accurately identify which samples were infected given their unique id in R^r, especially during low prevalence rates in the population. After the pooling matrix was completed, the samples were pooled together, run through qPCR, and displayed the output of infected pools. Each sample within an unaffected pool was labeled as negative for the virus. All other samples are either individually tested, processed through another testing method, or labeled as positive.

The number of samples in each pool within our tensor method will be given by $(ceiling(\sqrt[r]{n}))^{r-1}$. The number of tests in each tensor run is calculated to be $r \bullet ceiling(\sqrt[r]{n})$. In an example in which we have 64 samples with a tensor rank 2, our total number of samples within each pool is given by $(ceiling(\sqrt[2]{64}))^{2-1} = 8$. The number of tests for this tensor run is $2 \bullet ceiling(\sqrt[2]{64}) = 16$. This ensures a consistent number of tests and samples per pool for each run of our tensor method.

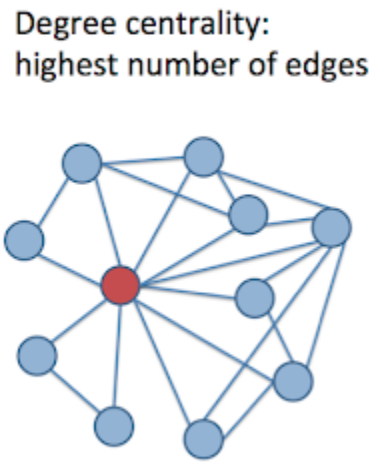
Centrality Pooling

With the infectious state of each person given by the SEIR disease model, we record the state of each individual at each time stage to understand the spread of the disease in the population. At the same time, we also randomly created connections in this social network to mimic relationships within communities: everyone has several friends, where each friend has their own friends, and so on. This is demonstrated in Figure 2. Thus, our testing results on this population will be more realistic and trustworthy, giving us more information about improving its efficiency in the real-world setting.



[Figure 2: 8 people social network with random connections and 2 infectious people (4 and 7)]

When we have a network model, we think about how to use the characteristics of the network to improve our model. One of the ideas is to use the degree centrality data of the model to help us create pools. The degree of centrality assigns each node's importance based on the number of connections to that node. In other words, if this node had a significant number of connections, this node had a high degree of centrality (Figure 3) [7].



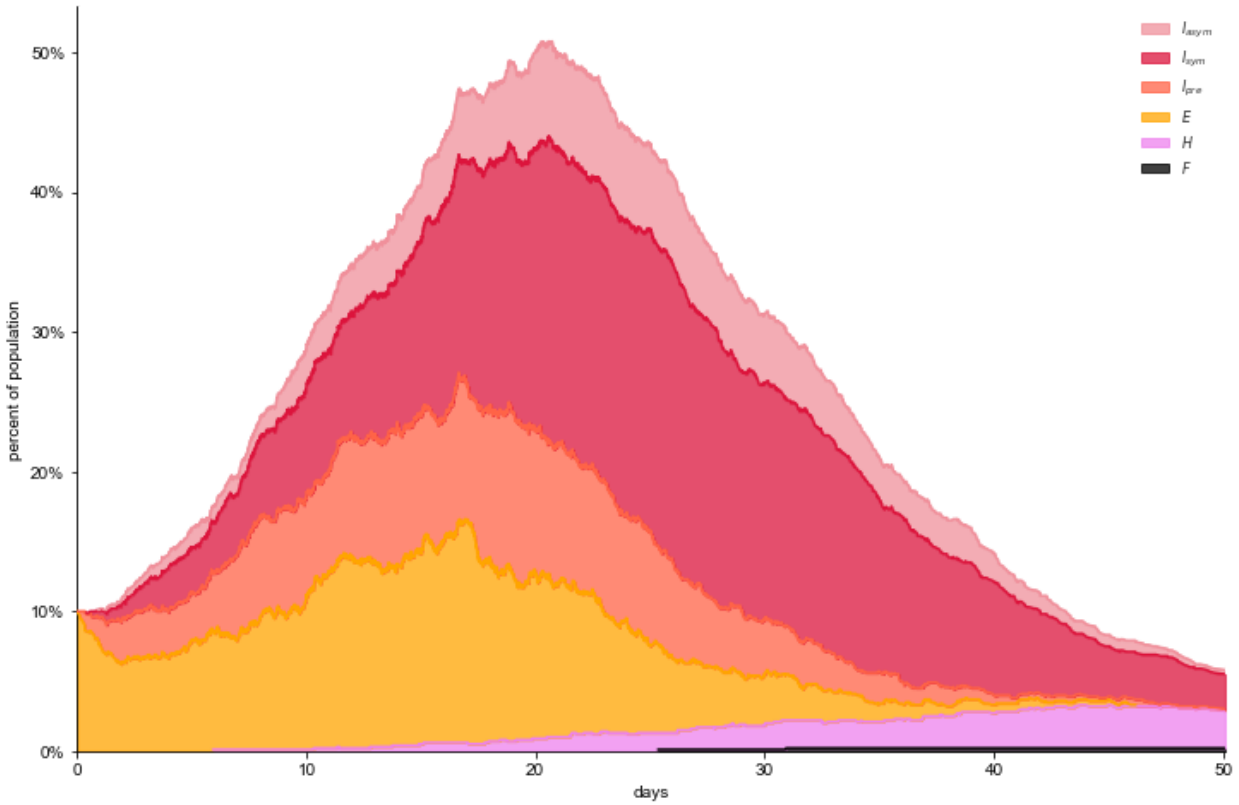
[Figure 3: high degree centrality node example in network]

The basic idea behind the degree centrality pooling is that we pool people based on their degree centrality instead of pooling people randomly. The idea is to prioritize those with high degree centrality because their chance of infection is greater, so the result of their group test is very likely to be positive, allowing us to find many positives in the first few tests. The remaining people are, therefore, much likely to be negative.

In conclusion, the higher degree centrality the more testing and smaller sampling sizes are. For example, we even perform many individual tests for people with top 5% of the degree centrality as their connections are very high in this network, suggesting their higher chance of infection.

Machine Learning Pooling

To advance our graphing strategy, we use the extended-SEIR infectious disease network model and also demographic model to address community level contacts. A simulation of COVID-19 spread over time within the communities was performed with the demographic network. We analyze the state of each individual at each time stage to understand the spread of the disease in the population. Figure 4 is a sample spread of the COVID-19 in the social network for fifty days with 15 initially infected people in a 1000 population. By simulating this spread in the network, our testing results on this population are more realistic, providing more information for testing.



[Figure 4: COVID-19 spread in the social network for fifty days with 15 initially infected people in 1000 population]

With so much information provided by these networks, we decide to use machine learning to assist us on testing as it would give our suggestions on whether these people are positive or negative based on the training. Therefore, we first extract two features from the disease network: degree centrality and eigenvector centrality. The eigenvector centrality, similar to degree centrality, assigns nodes' importance based on the eigenvector of the network adjacency matrix. In other words, the higher the eigenvector centrality is, the more connections to other high score nodes. We also use household size and age group from demographic networks. The last feature we use is infected count which is the number of times an individual appears in a positive group test. Following figure 5 is our sample training data.

	DegreeCentrality	EigenCentrality	Ages	HouseholdSize	count	positive
0	0.009005	0.011888	0-9	4	1.0	0
1	0.016508	0.006470	0-9	2	1.0	0
2	0.012506	0.003544	0-9	4	1.0	0
3	0.033517	0.006830	0-9	4	2.0	0
4	0.026513	0.009194	0-9	5	2.0	0
...	...	...	...	...	...	...
1995	0.002001	0.000169	60-69	4	1.0	0
1996	0.002001	0.000411	70-79	4	1.0	1
1997	0.001001	0.000110	60-69	2	4.0	0
1998	0.002001	0.000436	70-79	4	1.0	0
1999	0.001001	0.000332	70-79	2	0.0	0

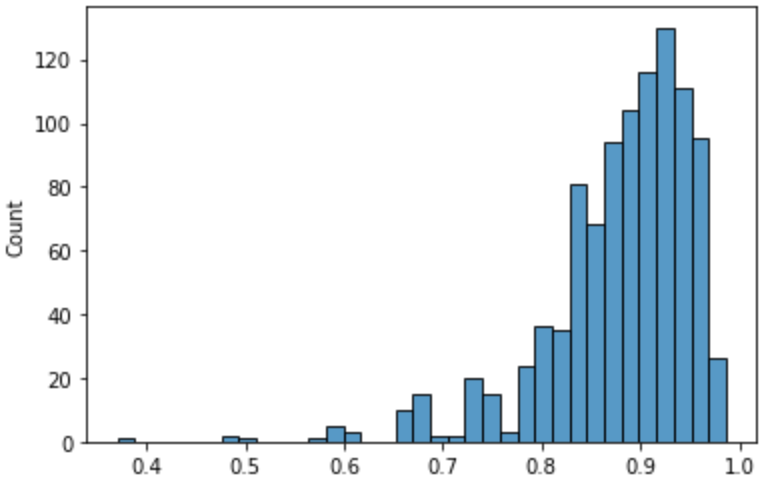
[Figure 5: sample training data for machine learning model]

Our trained machine learning model will return predictive probabilities of positive and negative for each node. Therefore, by looking at the predictive probabilities, we can estimate whether this person has a higher chance of being infected. We then sort predictive probabilities, and as we did in degree centrality pooling, we prioritize people with high predictive probabilities for positives. Thus, our trained model can help us target potential positive people, reducing the number of testsings.

IV. RESULTS

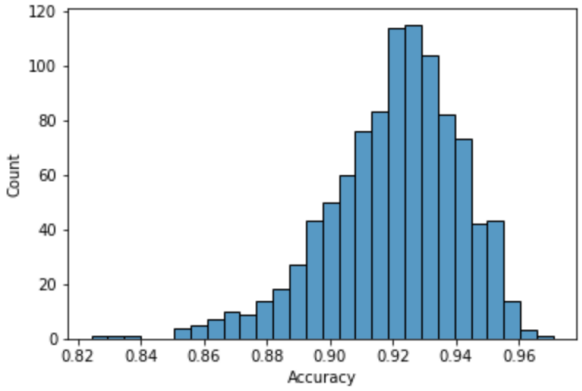
Non Adaptive Pooling

Using the same parameters as Two Step Adaptive Pooling (2-STAP)[4] with a sample size of 961 sample size, 52 tests, and 5 infected samples, our nonadaptive pooling algorithm had an average accuracy of 97.02% for a pool size of 192.



[Figure 6: Accuracy of the algorithm with pool size 192 over 1000 simulations]

Although the accuracies were high, a pool of 192 may be too dilute to detect and may not reflect conditions within the laboratory. Therefore, we limited the pool size to 64, and the results yielded an accuracy of 92%. While limiting the pool size, we accounted for the dilute samples by including a False Negative Rate of 0.1 with a pool size of 64[5].



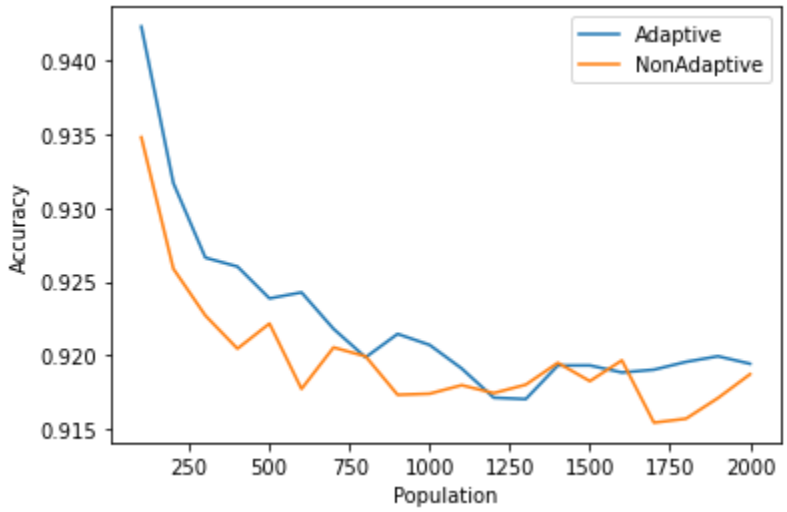
[Figure 7: Accuracy of the algorithm with pool size 64 and False Negative Rate]

Furthermore, running our non-adaptive pooling algorithm with 10 and 15 infected and a pool size of 64, yielding an average accuracy of 91% and 82%, respectively.

Adaptive Pooling

We ran our first adaptive algorithm (running initial tests, ruling out negative indexes, and pooling from the remaining indexes) with 70 max tests and 10 infected, while varying population and the simulation showed similar accuracies to the non-adaptive pooling scheme. The adaptive

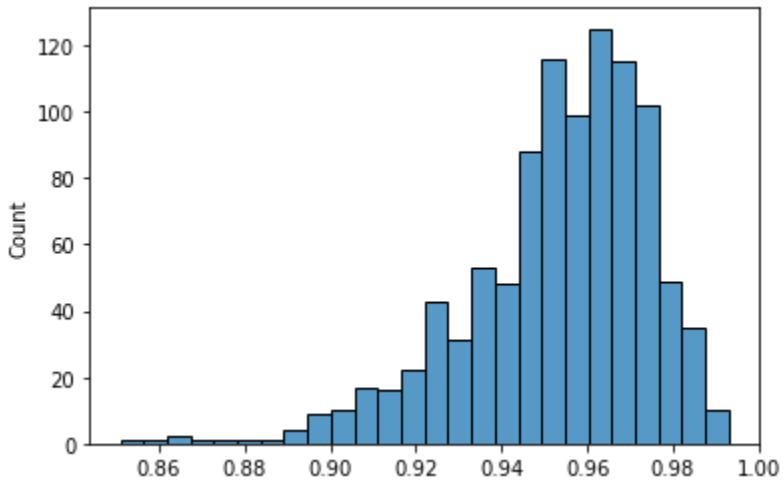
algorithm performs better at smaller population sizes, but has relatively similar accuracies with the nonadaptive strategy as the population size increases.



[Figure 8: Adaptive vs Non Adaptive accuracy with 70 tests, 10 infected across varying Populations]

Disjoint Pooling

Using similar parameters used in the nonadaptive simulations, the results for 5, 10, and 15 infected samples with a maximum pool size of 64 yielded an average accuracy score of 98%, 95% and 90% respectively.



[Figure 9: Accuracy of disjoint pooling for 10 infected samples]

Limited Adaptive Pooling

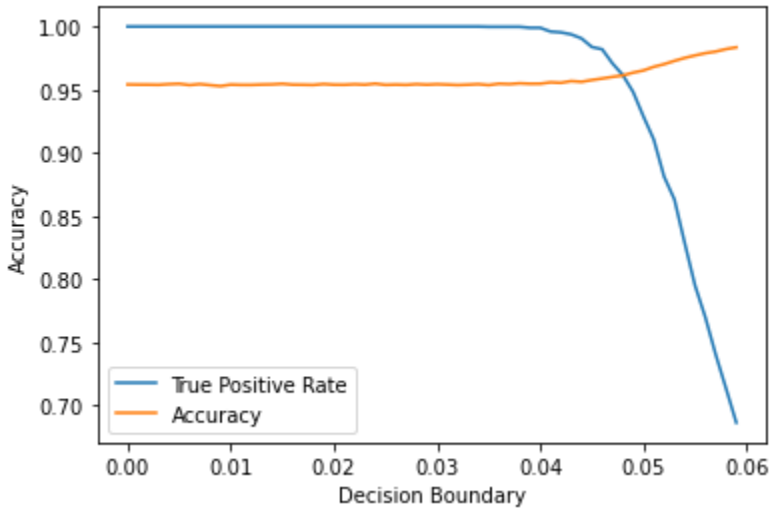
With a maximum pool size of 16, population size of 500, and 5 infected samples, we varied the number of adaptive steps to determine an optimal amount of adaptive steps before running the remaining tests as a nonadaptive scheme. With varied adaptive steps with a maximum of 40 tests, the accuracy between the steps did not fluctuate significantly, where each accuracy was similar within 1%.

Adaptive Cycles	Accuracy
2	91.7%
3	90.9%
4	90.7%

Table 1: Accuracies comparing Adaptive Cycle Steps in a Limited Adaptive Model

Inclusion of Least Squares Estimation

Least squares estimation was used to try and detect additional negative tests in an effort to improve the accuracy of the above pooling models. With the returned estimation x-vector, the accuracy was analyzed across different decision boundaries.



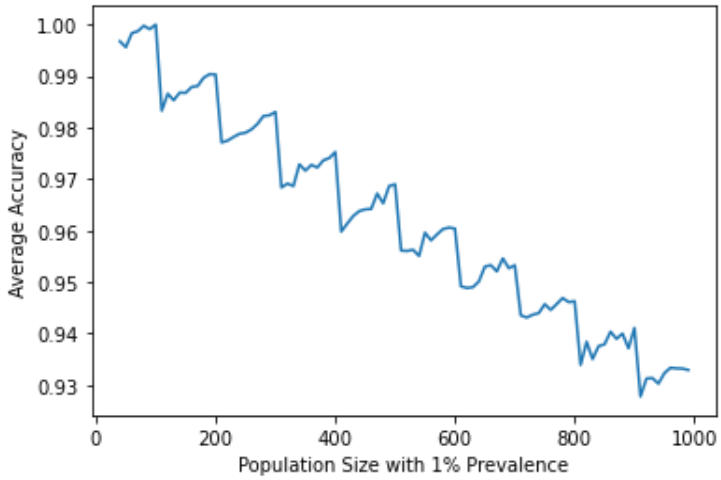
[Figure 10: Measured Accuracy across Least Estimation Estimation Decision Boundaries]

Summarized in Figure 10, there was little noticeable change when the regression was implemented.

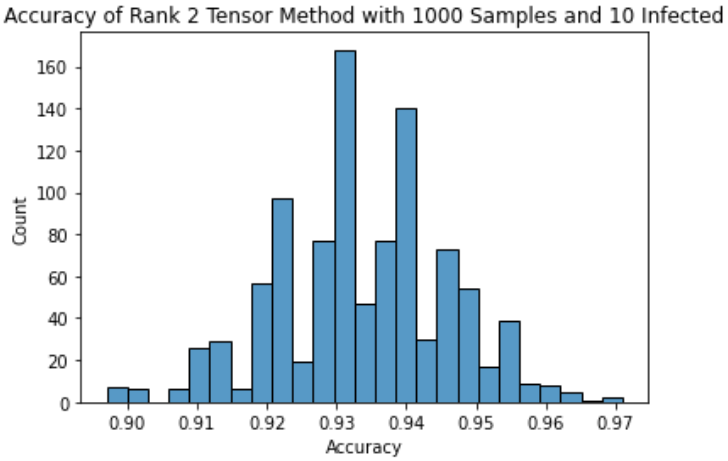
Tensor Pooling

Our initial tensor pooling method starts with placing each sample inside a rank 2 tensor (matrix) and then using the columns and rows of the matrix as the pools of our group testing procedure. Each pool will have a total of $(ceiling(\sqrt[2]{n}))$ samples and the tensor run will go

through $2 \bullet ceiling(\sqrt[2]{n})$ tests. We first test the accuracy of our base model as the population size increases. We first run the rank 2 tensor model through an increasing number of samples with a 1% prevalence rate. After the tensor run, all samples that were not labeled as negative will be labeled as positive. We then check for the accuracy of this method. Each run was done 100 times for each sample size and then averaged to find the average accuracy.



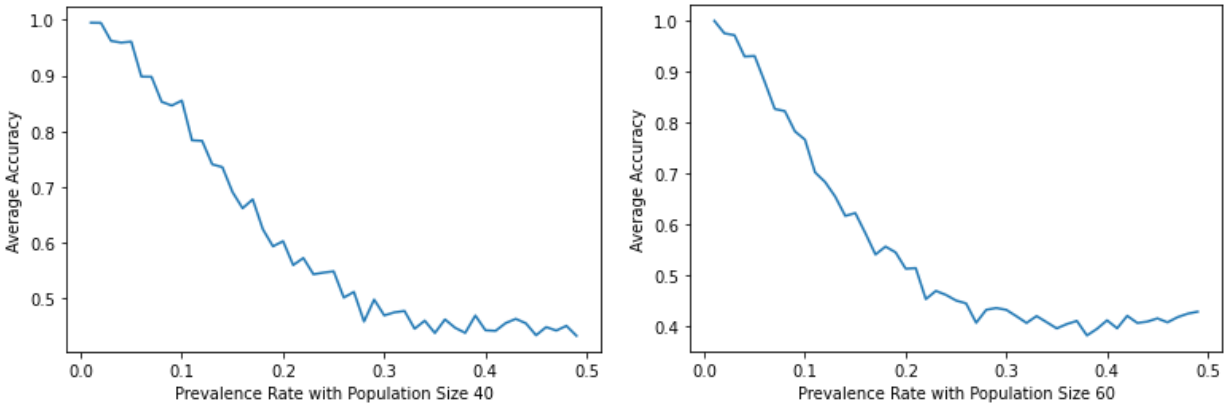
[Figure 11: Accuracy of Increasing Number of Samples Tested with 1% Prevalence]



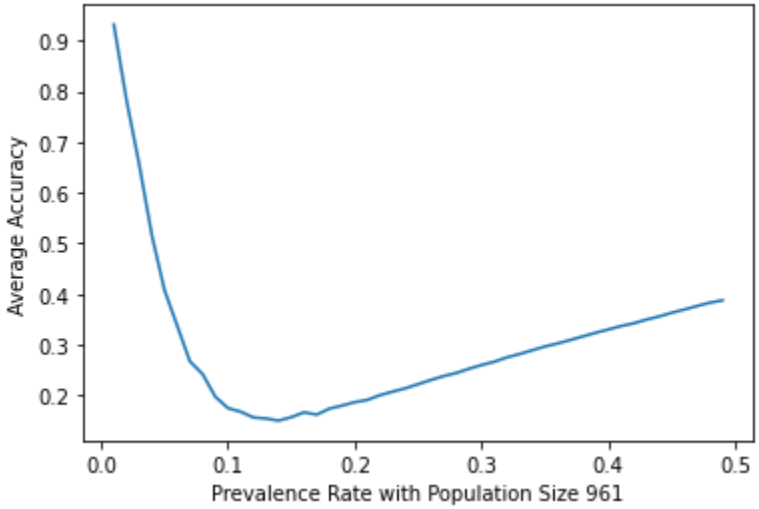
[Figure 12: Accuracy of rank 2 Tensor Method with 1000 samples and 10 infected]

Simulations show that a common trend in this rank 2 tensor method is that the accuracy will drop as the number of samples tested increases. The rank 2 tensor method received high accuracy while the number of samples tested was below 600, but began to drop below 95% as the number of samples increased above 600. The average accuracy for the rank 2 tensor method using 1000 samples with 10 infected (1% prevalence rate) is between 93% and 94%.

We then test the accuracy of our base model as the prevalence rate increases. We set similar methods as stated in the simulation above, with differing prevalence rates while setting the number of samples to 40, 60, and 961, respectively, mimicking Tapestry Numbers.



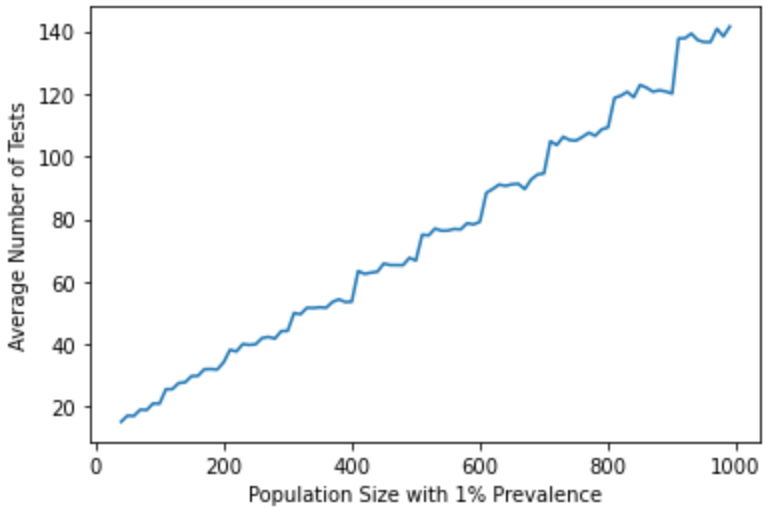
[Figure 13: Accuracy of Increasing Prevalence Rate for Sample size 40 and 60]



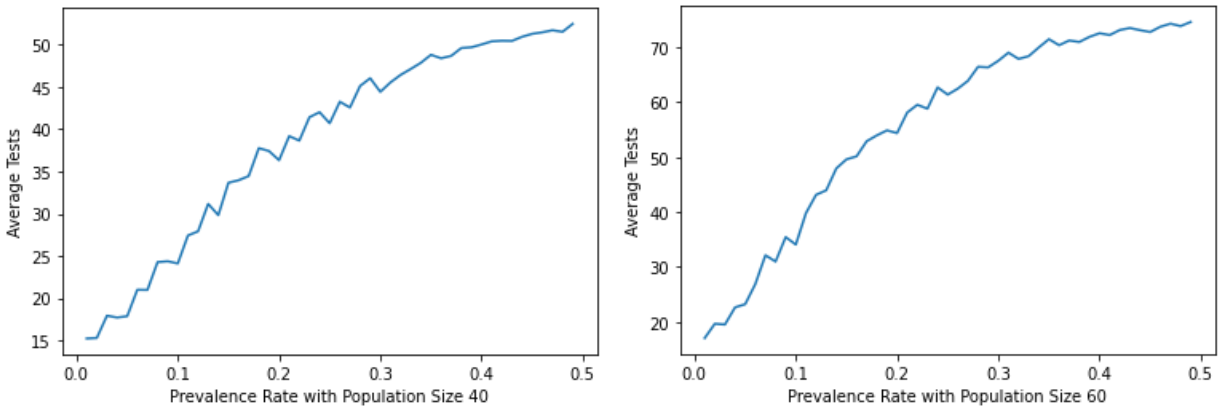
[Figure 14: Accuracy of Increasing Prevalence Rate for Sample size 961]

Based on our results, it shows that as the prevalence rate increases, the accuracy of our method decreases in general for a sample size of 40 and 60. With the sample size of 961, an increase in the prevalence rate significantly decreases the accuracy of our model, but then increases after a prevalence rate of around 15%.

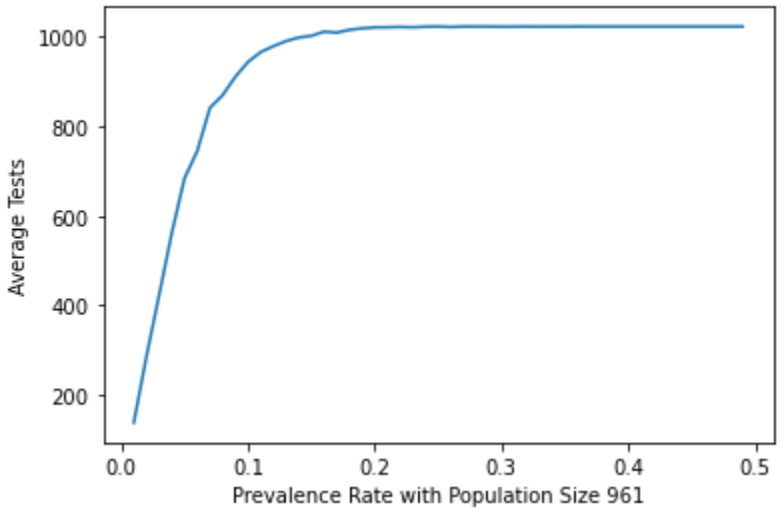
We then simulate results when our method is used to detect all infected samples by individually testing all samples that were not labeled as negative after our tensor run. Similar to the simulations above, we test both on increasing sample size and increasing prevalence rate with the same parameters.



[Figure 15: Number of Tests for Increasing Number of Samples Tested with 1% Prevalence]



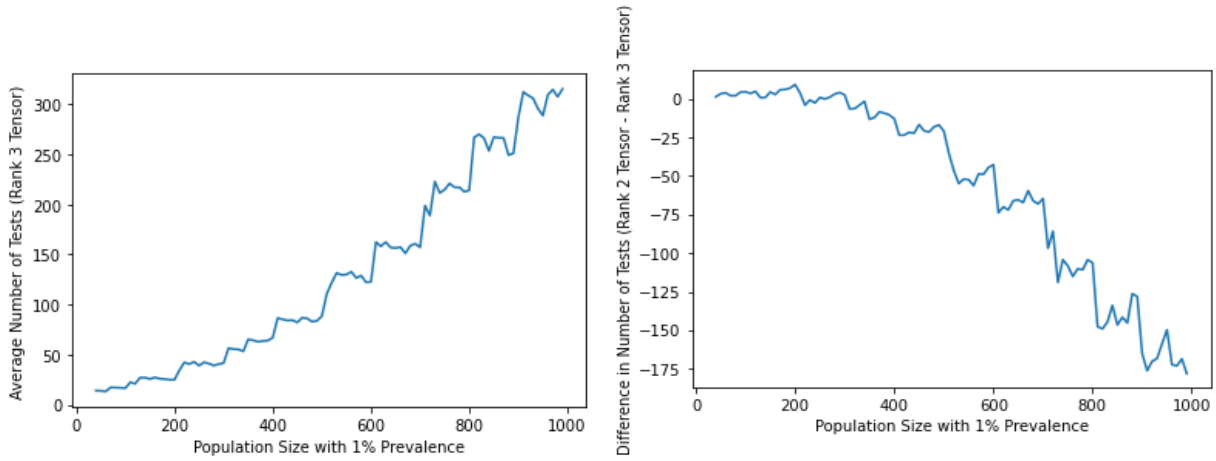
[Figure 16: Number of Tests for Increasing Prevalence Rate for Sample size 40 and 60]



[Figure 17: Number of Tests for Increasing Prevalence Rate for Sample size 961]

Based on our results, it shows that as the number of samples tested increases, the average number of tests needed increases at a semi-linear rate. Furthermore, as the number of infected samples increases, so does the number of tests. However, as the prevalence of the virus in our population increases to exceedingly high rates (around 25% prevalence rate for sample size 40 and 60 and around 15% prevalence rate for sample size 961), the tensor method fails to do better than individual testing. Additionally, a sample size of 40 and 10% prevalence rate only uses around 20 tests in order to detect all samples, competing with Tapestry numbers. A sample size of 60 and 10% prevalence rate only use around 30 tests and a sample size of 961 and 1% prevalence rate only use around 130 tests, indicating the consistency in competitiveness with Tapestry.

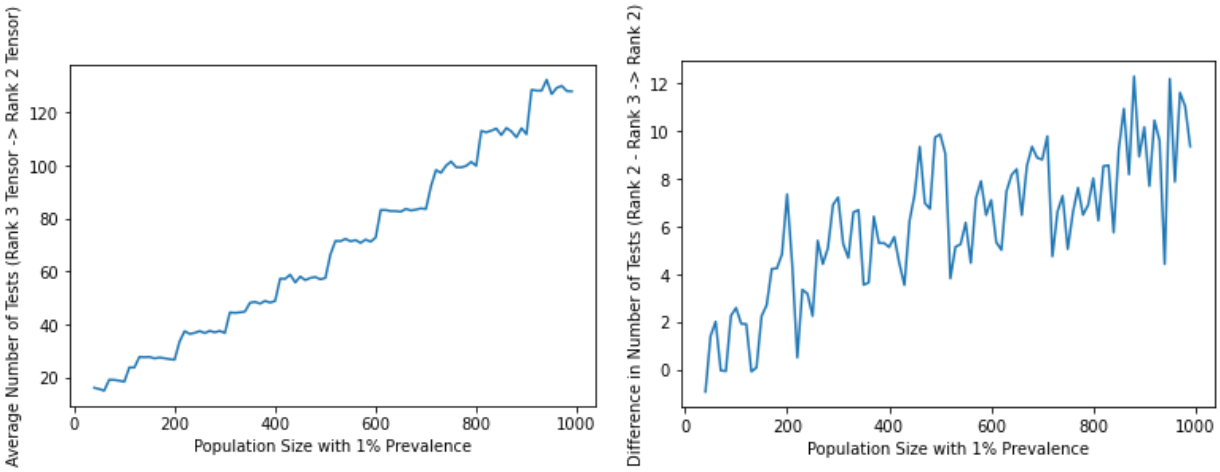
We test the difference between rank 2 tensor and rank 3 tensor when calculating the number of tests needed to detect all infected samples. Methods for rank 2 tensor are given by the above followed by individual testing to detect all infected samples. Methods for the rank 3 tensor are similar, but are instead placing the samples inside a rank 3 tensor (cube) and using the layers or faces of the cube as pools. We test each method by increasing the size of our sample and placing a 1% prevalence rate in our population. Each method was run 100 times for each sample size and then averaged to find our average number of tests.



[Figure 18: Number of tests and increasing sample size and 1% prevalence (rank 3 tensor method and difference between rank 2 tensor - rank 3 tensor)]

Results show that both tensor rank 2 and tensor rank 3 methods increase in the number of tests needed to detect all infected samples as the sample size increases. This phenomenon is theorized to occur because of the increased number of infected samples as the sample size increases. However, when looking at the number of tests, the rank 2 tensor method outperforms the rank 3

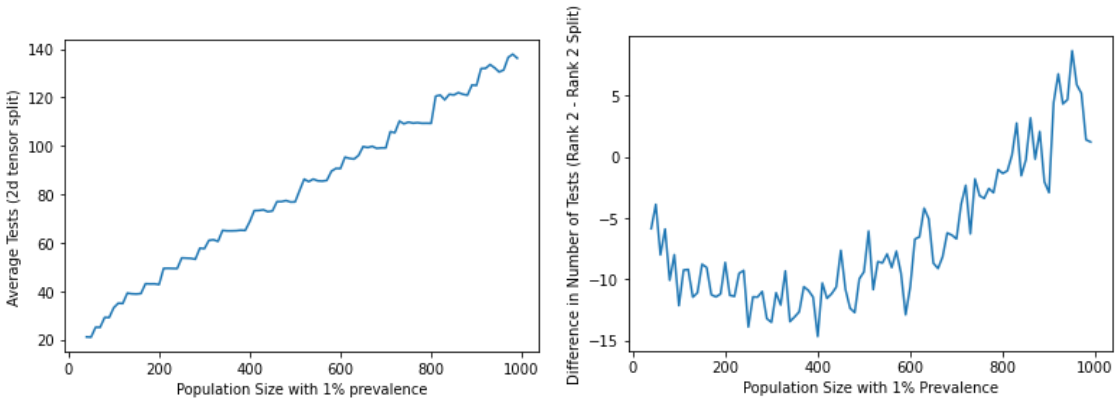
tensor method. The rank 3 tensor method will require, on average, higher amounts of tests to detect all infected samples within the population. This may be due to a large portion of samples that were not labeled as negative going through individual testing. Thus, we then change our method such that we run the rank 2 tensor method on the samples that were not labeled as negative after our rank 3 tensor method, with our final procedure being individual testing.



[Figure 19: Number of tests and increasing sample size and 1% prevalence (rank 3 -> rank 2 tensor method and difference between rank 2 tensor - rank 3 -> rank 2 tensor)]

Because the number of tests for each method was determined to be around the same, we concluded that using the tensor rank 2 method was a more feasible choice. This was because of complications arising from these high pool sizes. The number of samples within each pool increases by $\frac{(ceiling(\sqrt[3]{n}))^2}{(ceiling(\sqrt[2]{n}))}$ as n increases. Thus, for the following tests we consider using the tensor rank 2 methods as our preferable choice.

Our last test for the tensor method considers splitting the sample into two groups. The first stage will involve running the tensor rank 2 algorithm on both groups separately and the second stage will involve running independent tests on the remaining samples that have not been correctly identified.

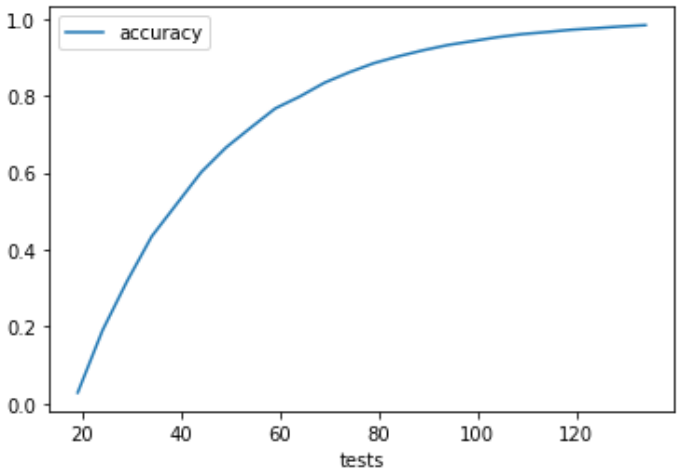


[Figure 20: Number of tests and increasing sample size and 1% prevalence (rank 2 tensor split method and difference between rank 2 tensor - rank 2 tensor split)]

Because the number of tests for each method was determined to be around the same, we concluded that using the rank 2 tensor method was a more feasible choice. This is due to the computational complexity it requires to run the rank 2 tensor split method. While the rank 2 tensor method only has to create one testing matrix, the rank 2 tensor split method has to create two testing matrices. Thus, the rank 2 tensor method is our current proposed method.

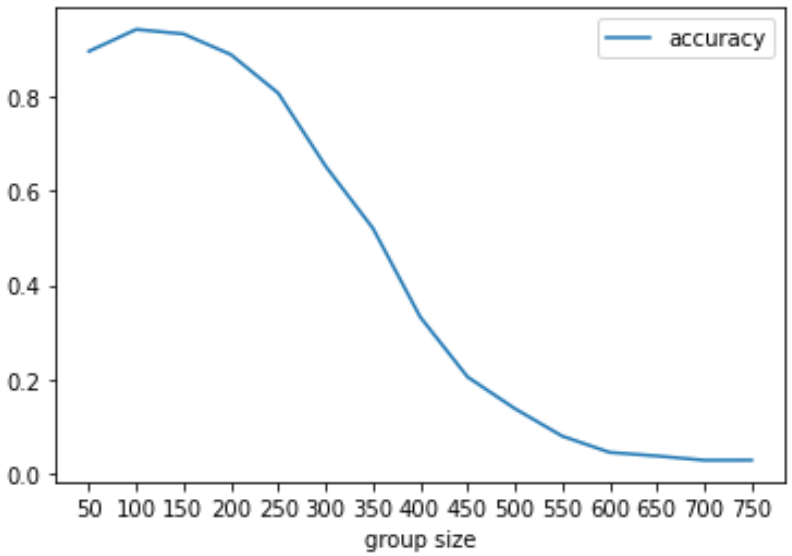
Degree Centrality Pooling

When the group size is 60, in a population with 1000 people, 20 infected people, the accuracy is around 80% for 60 tests and the accuracy still increases as group tests increase (Figure 21).



[Figure 21: Accuracy vs. number of testing when group size is 60]

When the testing number is 65, in a population with 1000 people, 20 infected people, the accuracy reaches its peak when there are about 100 people in a pool. The accuracy first increases and then decreases when pool size increases (Figure 21.1).

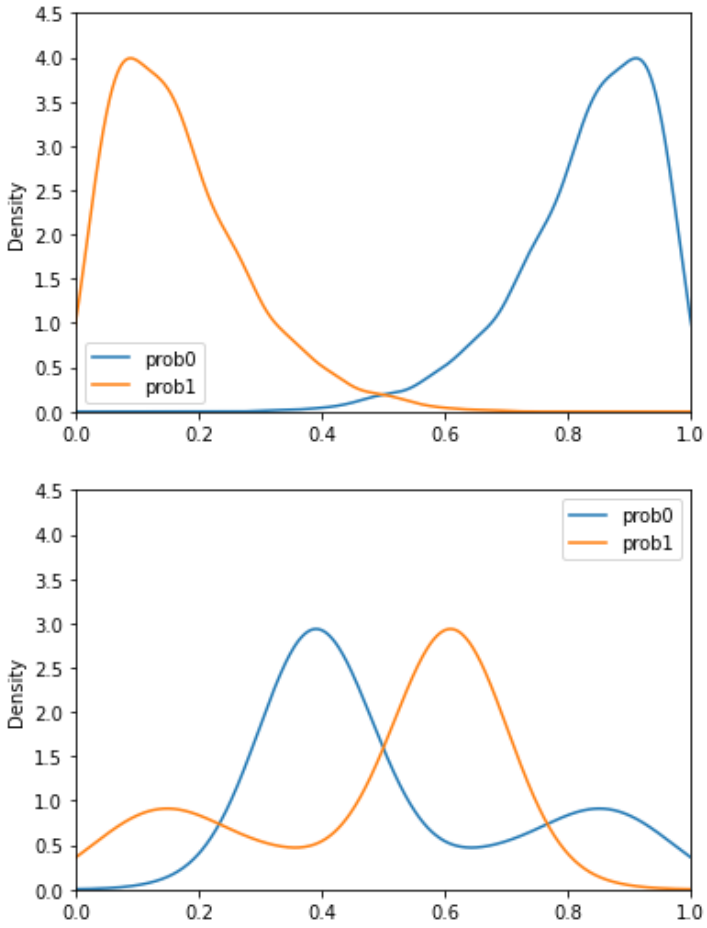


[Figure 21.1: Accuracy vs group size when number of tests is 65]

Machine Learning Pooling

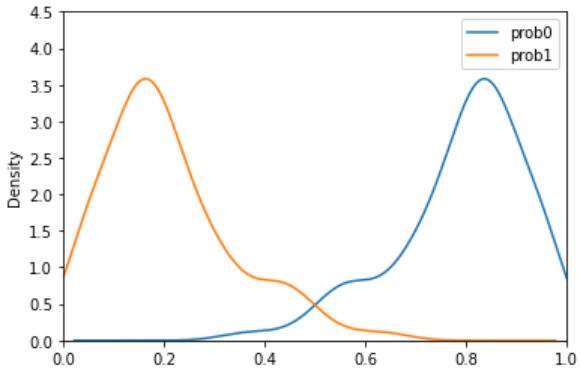
To solve the imbalance training dataset problem, we use a balanced random forest classifier to train our model. It performs bagging and bootstrapping to make the dataset more balanced and the sampling strategy is to sample only minority classes. Moreover, the class weights are applied, meaning it punishes models more harshly if it predicts positive people wrong.

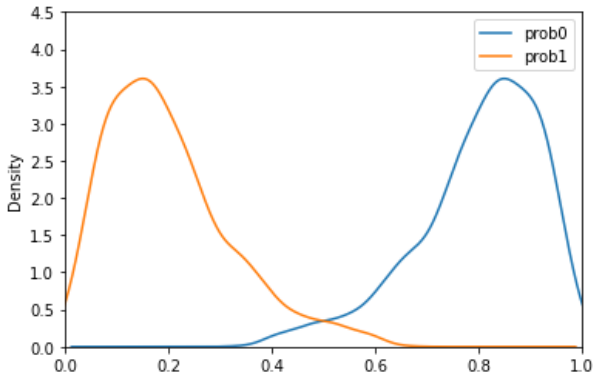
The classifier returns probabilities for this node being positive and negative. We think these probabilities can be used to improve our testing accuracy. In the figure 22, it is clear to see that the difference of predictive probabilities for positive and negative people are significantly different.



[Figure 22: ML's predictive probabilities results.
Top: for positive people; Bottom: for negative people]

However, the trained classifier does not apply to other generated networks as different randomness is added in every other generated network. The differences of its predictive probabilities are not significant differences as it did in the last network (Figure 23).





[Figure 23: ML's predictive probabilities results for new network
Top: for positive people; Bottom: for negative people]

To overcome this problem, we decided to use a portion of the data to train the network to ensure the classifier applies to the testing dataset. However, it forms a dilemma: in order to do that we need to perform a large number of individual tests to get positive status for training at the beginning. We also need to do many group testing beforehand to get enough infected count features. These together cost a lot of testing. But if we do not do that, our classifier won't be trained well and will perform poorly for the rest of the population.

Our conclusion is that we do not have enough real world data. If we do, it solves both problems listed above: we do not have to perform many testings beforehand as trained data is already provided; the trained classifier definitely would apply to new testing data as they all come from real world networks.

V. DISCUSSION

Non Adaptive Pooling

Both the non adaptive pooling schemes seem to underperform against the 2-STAP numbers, which makes sense because the 2STAP algorithm is an adaptive pooling scheme. Compared to Tapestry numbers, we slightly underperform, but that is a result of us limiting the max pool size to make our simulation more realistic. Still, we only see an average of 1-3% decrease in accuracy, even though we limited the pool size to be 64 in our algorithm..

Our non adaptive methods were able to reach accuracies greater than 93% when run with Tapestry numbers. This is very promising as we were able to decrease the amount of tests from 961(if you were running individual COVID-19 RT-qPCR), to no more than 90 tests. Also, our

methods being non adaptive makes it realistic, as we do not need to ask patients to stay until initial test results are back. The biggest shortcomings we see in our non adaptive pooling algorithms is that it does not accurately model real life situations. For example, people being infected should not be randomly picked from a uniform distribution, but rather, in real life people are infected via networks. Also, COVID-19 RT-qPCR tests spit out percentages rather than binary outputs of positive or negative.

Going forward, our objective is to not necessarily improve the accuracy by making pool sizes bigger, but it is to better model real life, by keeping pool sizes at a maximum of 64, while not sacrificing on accuracy. We plan to model real life issues better by implementing two things next quarter. Instead of outputting a binary output, we plan to output a percentage chance of infection, as this is what COVID-19 RT-qPCR tests actually output. Second, we plan to change our pooling algorithm into something that implements networks, which is described more in detail in the degree centrality pooling section. Ideally we want our final proposal to be something that is non-adaptive as this is most realistic for a real life trial.

Adaptive Pooling

Our first initial adaptive pooling algorithm did not do much better than the non-adaptive pooling algorithm, so we quickly moved on and implemented a recursive pooling algorithm. The recursive pooling algorithm, reached similar accuracies compared with the 2 STAP algorithm mentioned above. It reached accuracies of .99 consistently.

However, we do understand that our recursive adaptive pooling did not use realistic pool sizes, since at this point in our project, we did not yet introduce the idea of a max pool size of 64. Because of this, it is hard to conclude that we would see similar results if we actually tried our recursive algorithm in real life. We understand that saliva could be diluted so much in a large pool, that makes it nearly impossible to identify the COVID-19 virus even if it is present. Therefore, we are hopeful, but cautious. Our hope is to move from adaptive to non adaptive pooling as holding people in a testing site, and making them wait until initial tests are run is not ideal for anybody. Also adding the restriction of pool size and noise.

Limited Adaptive Pooling

Our limited adaptive pooling results compete with 2-STAP measures, but do use quite a large amount of adaptive cycles. This might cause problems with wait time when it is actually implemented in real life. If wait time is not an issue, the results of our limited adaptive pooling is actually quite promising, as we lower tests, while reaching pretty good accuracies. Looking at our results however, we are only convinced in our results if we run at least 6 adaptive cycles. Going forward, we want to shift more of our focus on pooling via networks, as this models the world more accurately.

Tensor Pooling

Our tensor rank 2 pooling method competes with Tapestry numbers and has a possibility of further improvement based on more research. The simplicity in the pooling method allows the flexibility to be used as a nonadaptive algorithm or an adaptive algorithm with few steps. Additionally, we find that the pool size within our tensor method works well for decreasing the dilution of positive samples within pools. It is not until we have 4097 samples within a tensor rank 2 method that the pool sizes will exceed 64. However, our method begins to suffer as the prevalence rates increase and relies more and more on individual testing as the number of infected samples within our sample increases. A possible theory suggests that as the number of infected samples increases, more rows and columns of our tensor become labeled as infected (the rows and columns indicate pools and the results of those pools are infected if only 1 sample in the pool is infected). Because the creation of the pooling matrix requires each sample to have a unique row and column id, adding one more infected sample to our sample will always flip at least one of the pools from not infected to infected. Thus, finding ways to increase the amount of overlap between infected samples may help to decrease the amount of unaffected pools being flipped.

Future work with this method can be achieved through more research in encoding and decoding algorithms. The similarity between this method and many encoding and decoding algorithms gives us a starting point on what additional tweaks are needed to improve this method. Additionally, this method can be combined with viral pools in order to accurately detect infected samples even before individual testing is performed. Another improvement on this method could be towards the parameters of the tensor pools, such as changing the size or arrangement of the pooling matrix to minimize the number of tests or isolate and detect infected

samples. Pushing this tensor method to optimal parameters can lead us towards finding a solution that detects infected samples with high accuracy, ensuring faster detection of the SARS-CoV-2 virus and quicker results for patients.

Degree Centrality Pooling

The result of degree centrality pooling is not very promising. In my opinion, the ideal accuracy for 60 tests in a population of 1000 with around 30 infected people should be at least 85% accuracy, compared to the result of the random pooling algorithm we tried first. I think one of the limitations is the SEIR model is an oversimplified model which cannot fully represent the realistic status of the Covid-19 in real world. Also, with this simple model, we can only access a limited amount of data. For example, we can only access network information like centrality, but cannot gain connective or demographic information. To advance this testing model, we employ a more sophisticated disease model which is used in machine learning testing strategy in our project.

Machine Learning Pooling

With advanced and more complex social network models, we have access to more information like household sizes, age-stratification and out-of-household contacts. These improve our testing method in various ways, for example, we are able to try to find patterns in the population to target high risk people, saving us plenty of tests. However, as we explore this strategy, we find out we cannot further advance due to the lack of real world data. With real world data, we can confidently train the mode which definitely applies to new testing populations as they both come from real world networks; moreover, with the real world data, we no longer need to do multiple individual and group tests beforehand. However, we do think machine learning strategy is a promising strategy as the trained model can give us predictive probabilities whether this person is positive or negative, providing a lot of information during the tests. Combining these tests, the testing method can specifically pool and test high risk people, saving resources and time. Our team reckon if we have real world data, the machine learning method or even neural network testing method should be considered to carry out.

VI. REFERENCES

[1] Appleby, J. “What Takes So Long? A Behind-The-Scenes Look At The Steps Involved In COVID-19 Testing.” *Kaiser Health News*, 30 March 2020, https://khn.org/news/what-takes-so-long-a-behind-the-scenes-look-at-the-steps-involved-in-covid-19-testing/

[a] Dorfman, R. “The detection of defective members of large populations,” *The Annals of Mathematical Statistics*, vol. 14, no. 4, pp. 436–440, 1943.

[2] Ghosh, S. “Tapestry: A Single-Round Smart Pooling Technique for COVID-19 Testing.” *medRxiv*, 2020, https://www.medrxiv.org/content/10.1101/2020.04.23.20077727v2.

[3] Heidarzadeh, A., and K. Narayanan. “Two-Stage Adaptive Pooling with RT-qPCR for COVID-19 Screening.” 2020, https://www.medrxiv.org/content/10.1101/2020.07.05.20146936v1.full.

[4] Yelin, I. et al. “Evaluation of COVID-19 RT-qPCR Test in Multi sample Pools.” *Clinical infectious diseases : an official publication of the Infectious Diseases Society of America* vol. 71,16 (2020): 2073-2078. doi:10.1093/cid/ciaa531

[5] Ryansmcgee. “SEIRS Model Description · Ryansmcgee/Seirsplus Wiki.” *GitHub*, https://github.com/ryansmcgee/seirsplus/wiki/SEIRS-Model-Description.

[6] Weingart, S. “Networks Demystified 2: Degree – the scottbot irregular.” *The Scottbot Irregular*, 17 December 2011, https://scottbot.net/networks-demystified-2-degree/.