POOLING MATRIX DESIGNS FOR GROUP TESTING

Yong Hong Ivan Tan[†]

Project advisors: Delin Chu [‡], Timo Sprekeler [§], Johannes J Brust [¶]

Abstract. The main objective of this article is to find the group testing strategy which minimizes the number of groups to test while identifying all positives. This manuscript explores the Hypercube Approach, the Kirkman Triple and Polynomial Pools Algorithms which are used to design group testing strategies. This work enhances the Polynomial Pools Algorithm with the Projective Geometry Design and proposes an algorithm which returns an effective group testing strategy when compared to other well-known algorithms.

1. Introduction.

Group testing is a method of identifying certain objects while minimizing the number of tests. This is achieved by testing groups of objects rather than testing every object individually. For instance, we have six light bulbs connected in a straight line and we know that one of them is defective. We are interested in identifying the defective bulb while minimizing the number of tests required. Suppose all the first three light bulbs light up, but all the first four light bulbs do not light up. Then, we can conclude that the fourth light bulb is defective by two tests.

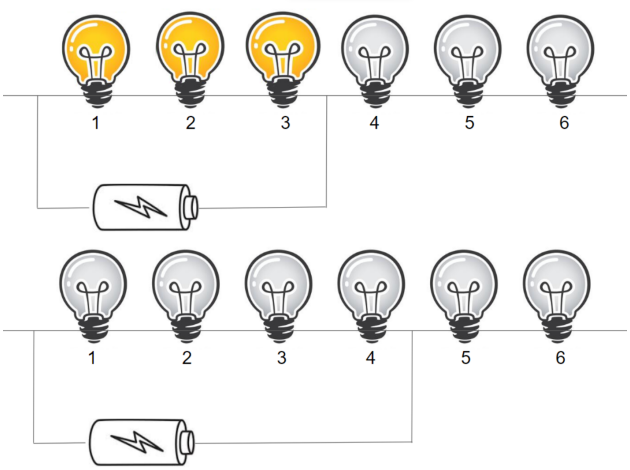

Figure 1: Arrangement of Light Bulbs in a Straight Line.

Therefore, with a large number of light bulbs, one can significantly narrow the search for the defective bulb by testing half of the number of light bulbs first. Here, a defective bulb is taken to be a positive sample while a non-defective bulb is taken to be a negative sample. By the light bulb example, a group of samples will be tested negative if all samples in the group are negative. Otherwise, the group will be tested positive. In group testing, due to the large number of N samples to be tested individually, it can be advantageous to assign them into M groups instead. A pooling matrix is a $M \times N$ matrix where the (i, j)-entry is 1 if the jth sample is contained in the ith group, and 0 otherwise. Given N samples and k positive samples, we will compare group testing strategies by the number of tests required to identify the positives.

[†]National University of Singapore (ivantanyh@gmail.com).
[‡]National University of Singapore (matchudl@nus.edu.sg).
[§]National University of Singapore (timo.sprekeler@nus.edu.sg).
[¶]Arizona State University (jjbrust@asu.edu).

2. Group Testing Strategies By Direct Decoding. Different from the light bulb example, in reality the tester does not know the number of positive samples. Therefore, a typical group testing strategy is as follows. A sample is decoded negative if it belongs to a negative group. A sample is decoded positive if it belongs to a positive group and all other samples in the group are decoded negative. Otherwise, the sample is undecodable and will be tested individually [12]. The following subsections will describe some group testing strategies. The Hypercube [11], see subsection 2.2, Kirkman Triple [8, 6], see subsection 2.3, and Polynomial Pools Algorithms [1, 2], see subsection 2.4, will be presented. Afterwards, given N samples and k positive samples, we will compare the algorithms based on their ability to decode all samples while minimizing the number of tests, see subsection 2.5.

2.1. Dorfman's Two Stage Approach. Dorfman's Two Stage Approach [10] encapsulates the essential idea of group testing. In its simplest form, the approach pools all samples into one group. Only one test is applied to the group at first. If the test is negative, one can conclude that all samples must be negative. If the test is positive, all samples have to be retested individually. This is a significant drawback. As an example, we will assume that we have 64 blood samples and two samples are infected. A group of blood samples is tested positive if at least one of the blood samples in the group is infected. Otherwise, the group will be tested negative. Suppose we are interested in identifying the two infected samples. If we group all blood samples together, we will end up having to do 65 tests. However, to minimize the additional tests for a positive group, we can pool them into eight groups of eight samples each. Then, we test the eight groups. If both the two infected samples are contained in the same group, the group testing results will be similar to the left. Otherwise, the group testing results will be similar to the right. Here, a positive sample is taken to be an infected sample while a negative sample is taken to be an uninfected sample.

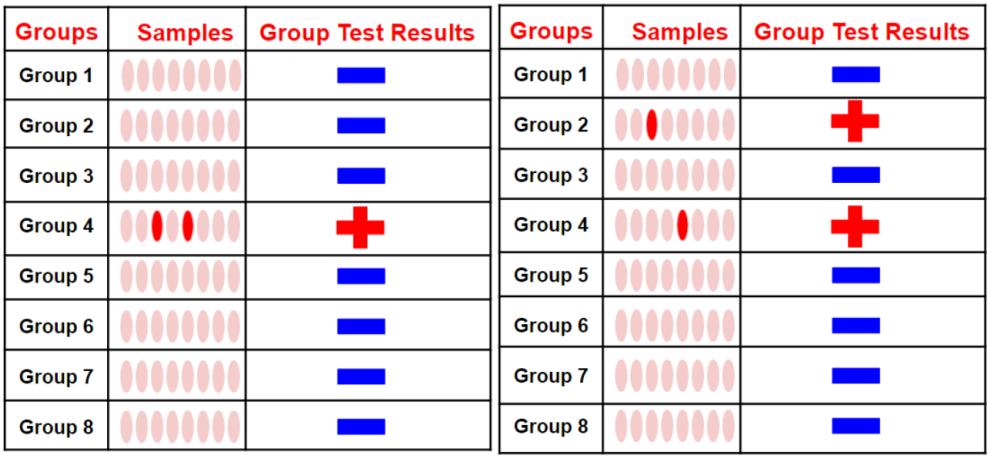

Figure 2: Dorfman's Two Stage Approach in Group Testing. Infected samples are in red.

If both infected samples are in the same group, we can identify the infected samples with 16 tests, since we need to test all eight samples individually in the positive group four. Otherwise, we can identify the infected sample with 24 tests, since we need to test a total of 16 samples individually from groups two and four. Therefore, the expected number of tests is $16 \cdot \frac{64 \cdot 7}{64 \cdot 63} + 24 \cdot (1 - \frac{64 \cdot 7}{64 \cdot 63}) = 23.11$.

2.2. Hypercube Designs. Hypercube designs minimize the number of groups by arranging samples in a D dimensional cube. Each $D-1$ dimensional cube in it constitutes one group.

2.2.1. Two Dimensional Hypercube Design. The two dimensional hypercube design [7] arranges the 64 samples in an eight by eight square. Each row and column constitutes one group. Hence, 16 groups will be tested first. A sample is decoded negative if either the row or column containing it is negative. Otherwise, a sample is decoded positive if all other samples in the same column or row as it are decoded negative. Else, the sample is undecodable. Suppose, the two positive samples are located in the same column four. Then, we can identify the two positive samples directly in rows one and five without any further testings below. Here, samples which are decoded negative are strike in blue.

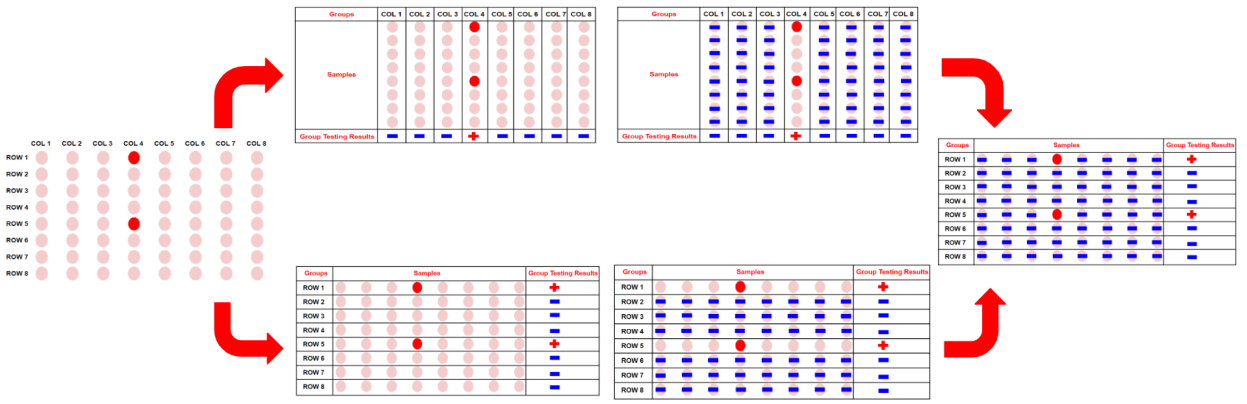

Figure 3: 2D Hypercube when the infected samples are in the same column.

Suppose the two positive samples are neither in the same row nor column. We will be left with four undecodable samples which have to be tested individually below.

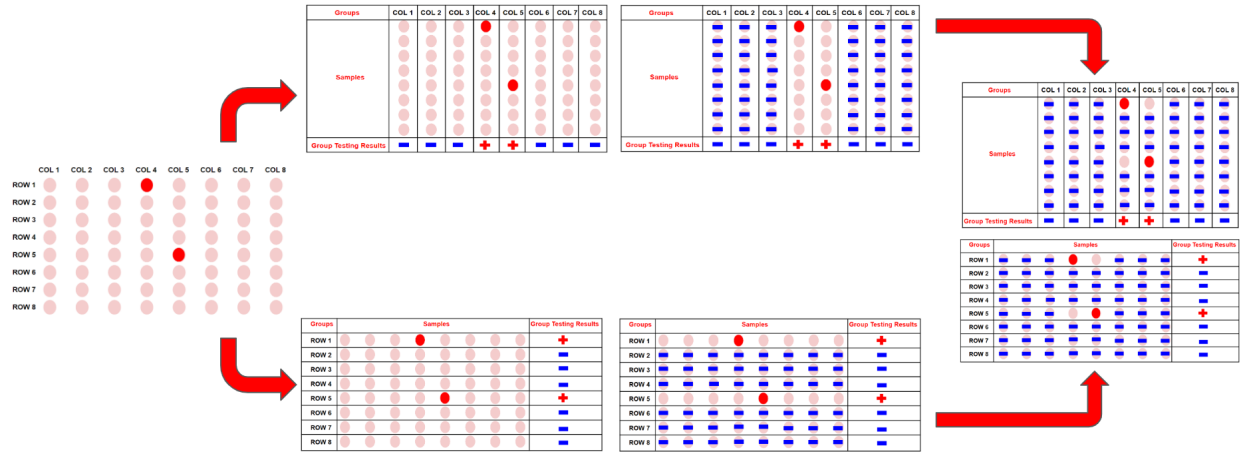

Figure 4: 2D Hypercube when the infected samples are neither in the same column nor row.

Hence, 20 tests will be required to identify the two positive samples if the two positive samples are neither in the same row nor column. Otherwise, 16 tests are sufficient. Therefore, the expected number of tests required is $16 \cdot \frac{64 \cdot 14}{64 \cdot 63} + 20 \cdot (1 - \frac{64 \cdot 14}{64 \cdot 63}) = 19.11$, which is lower than the expected number of tests required in the approach from subsection 2.1.

2.2.2. Three Dimensional Hypercube Design. The three dimensional hypercube design [11] arranges the 64 samples in a four by four by four cube and group them by slicing the cube horizontally, vertically and sideways. The 12 groups will be tested first, where each group contains 16 samples. Suppose the two positive samples are in a straight line. We can identify them directly in groups one and three without any further testings below.

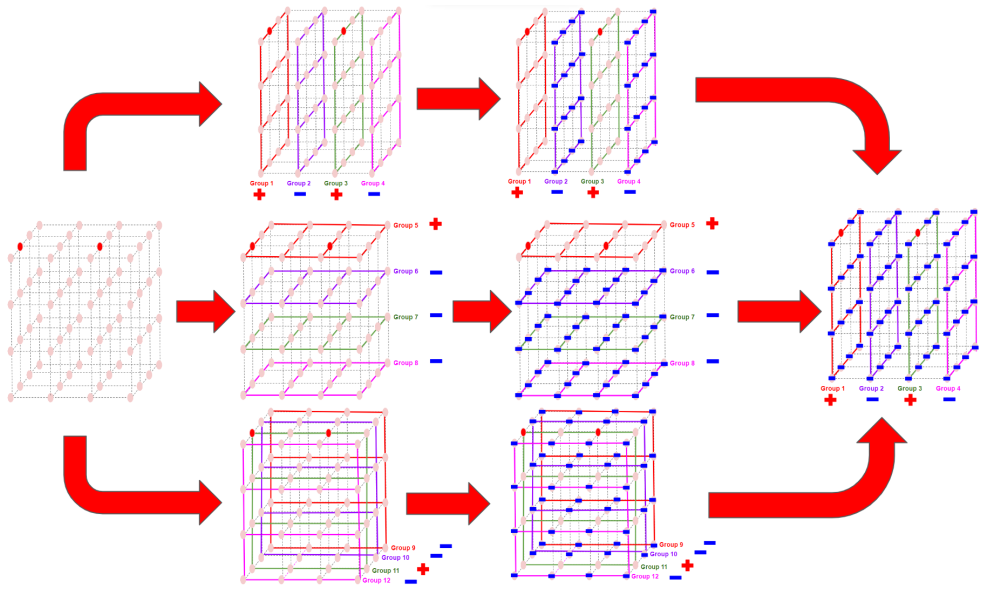

Figure 5: 3D Hypercube when the infected samples are in a straight line.

Suppose the two positive samples are contained in a similar group, group one. We will be left with four undecodable samples which have to be tested individually below.

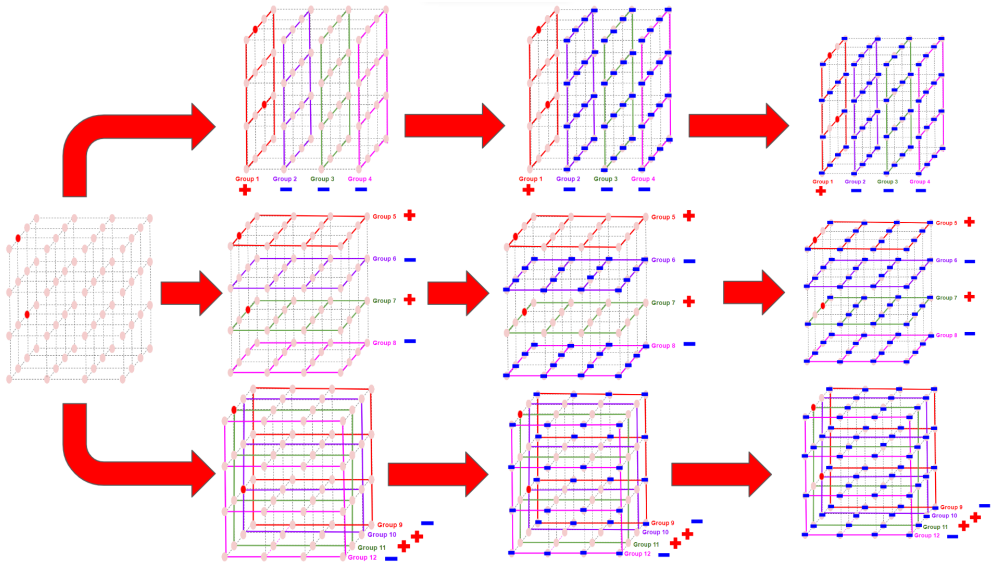

Figure 6: 3D Hypercube when the infected samples are in a similar group but not in a straight line.

Suppose the two positive samples are not contained together in any similar group. We will be left with eight undecodable samples which have to be tested individually below.

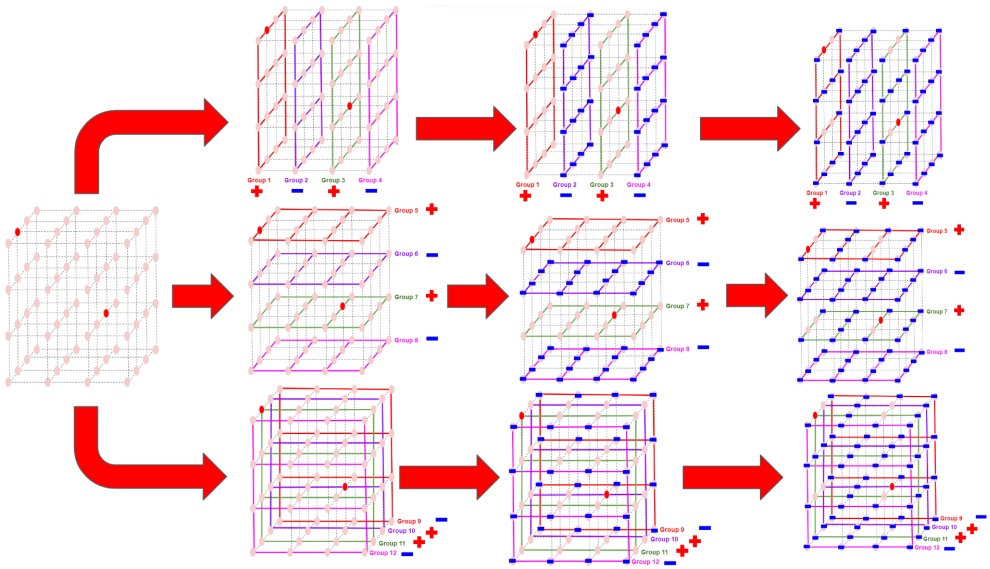

Figure 7: 3D Hypercube when the infected samples are contained in different groups.

Hence, 12 tests will be required to decode all samples if both positive samples are contained in a straight line. 16 tests will be required if both positive samples are contained in a similar group but not in a straight line. Otherwise, 20 tests is sufficient. Therefore, the expected number of tests required is $12 \cdot \frac{64 \cdot 9}{64 \cdot 63} + 16 \cdot \frac{64 \cdot (36-9)}{64 \cdot 63} + 20 \cdot (1 - \frac{64 \cdot 36}{64 \cdot 63}) = 17.14$, which is lower than the expected number of tests in both the approaches from subsections 2.1 and 2.2.1.

2.2.3. General Hypercube Design. For N samples and k positive samples, a D dimensional hypercube can be constructed in Algorithm 2.1 [11]. For simplicity, given N samples and $k > 1$ positive samples, we will choose the hypercube design which minimizes the number of tests required when all positive samples are in different groups. This will be $L(D-1)+h+\min\{1,m\}+k^D$ tests. When $k = 1$, the positive sample can be directly identified without further tests.

Algorithm 2.1 D Dimensional Hypercube Design

Let $L \leq \lfloor \sqrt{N} \rfloor$ and $D = \lceil \frac{log(N)}{log(L)} \rceil$. By Division Algorithm, there exist integers $0 \leq h \leq \frac{N}{L^{D-1}}$ and $0 \leq m \leq L^{D-1} - 1$ such that $N = hL^{D-1} + m$. We can arrange the N samples in a D dimensional cube. Each of the $D-1$ dimensional cube within it constitutes one group. Hence, $L(D-1) + h + \min\{1, m\}$ groups will be tested first.

When $D = 2$ and $L = \lfloor \sqrt{N} \rfloor$, by the Division Algorithm, there exist integers $0 \leq h \leq \frac{N}{L}$ and $0 \leq m \leq L - 1$ such that $N = Lh + m$. From Figure 8, we can arrange the samples in a h by $(L + \min\{1, m\})$ grid. Each of the h rows constitutes one group and each of the $L + \min\{1, m\}$ columns constitutes one group. Hence, $L + h + \min\{1, m\}$ groups will be tested first.

When $L \leq \lfloor \sqrt{N} \rfloor$ and $D = 3$, by the Division Algorithm, there exist integers $0 \leq h \leq \frac{N}{L^2}$ and $0 \leq m \leq L - 1$ such that $N = hL^2 + m$. From Figure 9, we can arrange the N samples in a L by L by $(h + \min\{1, m\})$ cube and group them. $2L + h + \min\{1, m\}$ groups will be tested first.

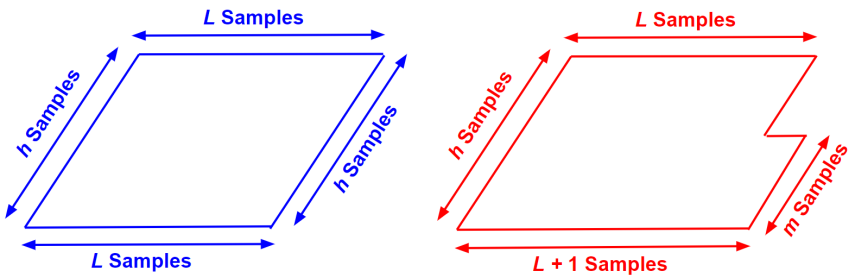

Figure 8: 2D Hypercube. The blue grid is for the case when $m = 0$. L columns contain h samples each and h rows contain L samples each. The red grid is for the case when $m > 0$. L columns contain h samples each and the last column contains m samples. m rows contain $L + 1$ samples each and the remaining $h - m$ rows contain L samples each.

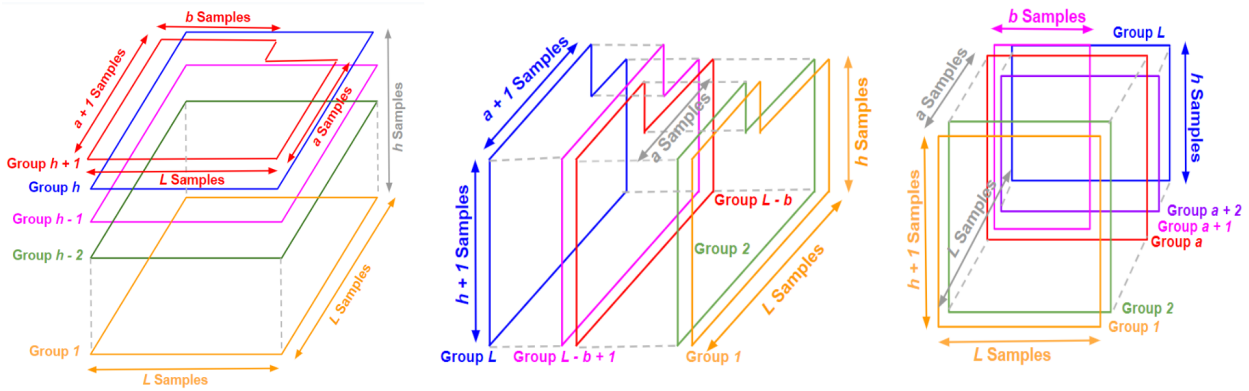

Figure 9: 3D Hypercube. $m = aL + b$ for integers $0 \leq a \leq L - 1$ and $0 \leq b \leq L - 1$

2.3. Kirkman Triple Algorithm. The Kirkman Triple Algorithm represents a group testing strategy where each sample is contained in three groups and every pair of samples is contained in one group. The strategy can exactly decode at most $k = 2$ positive samples. This is because all negative samples in a positive group can be decoded as they belong to a negative group. The design of the Kirkman Triple Algorithm is motivated by Theorems 5 and 6 of [6]. The algorithm works in constructing Kirkman Triple Pooling Matrices, which represent the group testing strategies of pooling $N = (4t+1)p^z$ samples into $M = 2p^z + 1$ groups and $N = (9t+1)p^z$ samples into $M = 3p^z$ groups, where $p^z = 6t+1$ is a prime power for some integer t, see Appendix B. Moreover, Tapestry Pooling Matrices [9] allow us to pool $N = 40$ samples into $M = 16$ groups or $N = 60$ samples into $M = 24$ groups, while ensuring similar properties as the Kirkman Triple Pooling Matrices.

2.4. Polynomial Pools Algorithm. The PPOL Algorithm [2] is the two dimensional case of the Polynomial Pools Algorithm of arranging and grouping samples in a two dimensional rectangular grid, see subsection 2.4.1. This can be extended to arranging and grouping samples in a multidimensional grid, see subsection 2.4.2[1]. Then, the Projective Geometry Design can be implemented, see subsection 2.4.3, to further reduce the number of groups to test while decoding all samples.

2.4.1. PPOL Algorithm. PPOL stands for Packing the Pencil of Lines [2]. Suppose $N = np^z$ samples are to be assigned into $M = mp^z$ groups, where $1 \leq n, m \leq p^z$ for a prime power p^z. We require each group to contain n samples and each sample to be in m groups. The PPOL Algorithm which achieves this property is described in Algorithm 2.2. For each $0 \leq c \leq m-1$, we can define a collection of p^z parallel groups to be $\{\{hp^z + [b \biguplus (h \bigodot c)] + 1 : 0 \leq h \leq n-1\} : b \in F_{p^z}\}$. Here, $\biguplus$ and $\bigodot$ denote addition and multiplication respectively in F_{p^z}, see Appendix A. Groups which are parallel to each other do not contain any common samples. Each pair of non-parallel groups contains exactly one sample in common, see Appendix C. To assign $N = np^z$ samples into $M = mp^z$ groups, we have to assign the samples into m collections of p^z parallel groups by varying $0 \leq c \leq m-1$. As an example, consider the prime power $p^z = 8$. We want to assign $N = 64$ samples into $M = 48$ groups using the PPOL Algorithm, where $n = 8$ and $m = 6$. For each integer $0 \leq c \leq 5$, we can define a collection of eight parallel groups to be $\{\{8h + [b \biguplus (h \bigodot c)] + 1 : 0 \leq h \leq 7\} : b \in F_8\}$. Here, $\biguplus$ and $\bigodot$ denote the addition and multiplication respectively in F_8 as described below.

Addition Table in F_8

⊎	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

Multiplication Table in F_8

⊙	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

Figure 10: Addition and Multiplication Tables in F_8

For instance, $5 \biguplus 3 = 3 \biguplus 5 = 6$ and $5 \bigodot 3 = 3 \bigodot 5 = 4$. The assignment of $N = 64$ samples into a collection of eight parallel groups is described in Algorithm 2.2 below, for $c = 4$.

Algorithm 2.2 PPOL Algorithm for pooling np^z samples into a collection of p^z parallel groups

Step A : Arrange the samples such that a row contains p^z samples while a column contains n Samples. The sample labelled $(i-1)p^z + j$ is contained in the ith row and jth column.

Step B : For each of the n rows, relabel the samples from 0 to $p^z - 1$. In each row, 0 denotes the leftmost sample and $p^z - 1$ denotes the rightmost sample.

Step C : For each integer $0 \leq c \leq m-1$, we are to determine a collection of p^z parallel groups where each group contains a sample in each row, see figure 11. For a fixed $0 \leq c \leq m-1$, we will achieve a collection of the p^z parallel groups. Here, $\biguplus$ and $\bigodot$ denote addition and multiplication respectively in F_{p^z}. The first three parallel groups are listed below.

- **Group 1 (In Green)** $= \{hp^z + h \bigodot c + 1 : 0 \leq h \leq n-1\}$
- **Group 2 (In Blue)** $= \{hp^z + [1 \biguplus (h \bigodot c)] + 1 : 0 \leq h \leq n-1\}$
- **Group 3 (In Pink)**$= \{hp^z + [2 \biguplus (h \bigodot c)] + 1 : 0 \leq h \leq n-1\}$

By fixing $c = 4$, the assignment of 64 samples into a collection of eight parallel groups is illustrated in figure 12, where the eight samples to be assigned in each group are in **black**. $\biguplus$ and $\bigodot$ denote the addition and multiplication operators respectively in F_8.

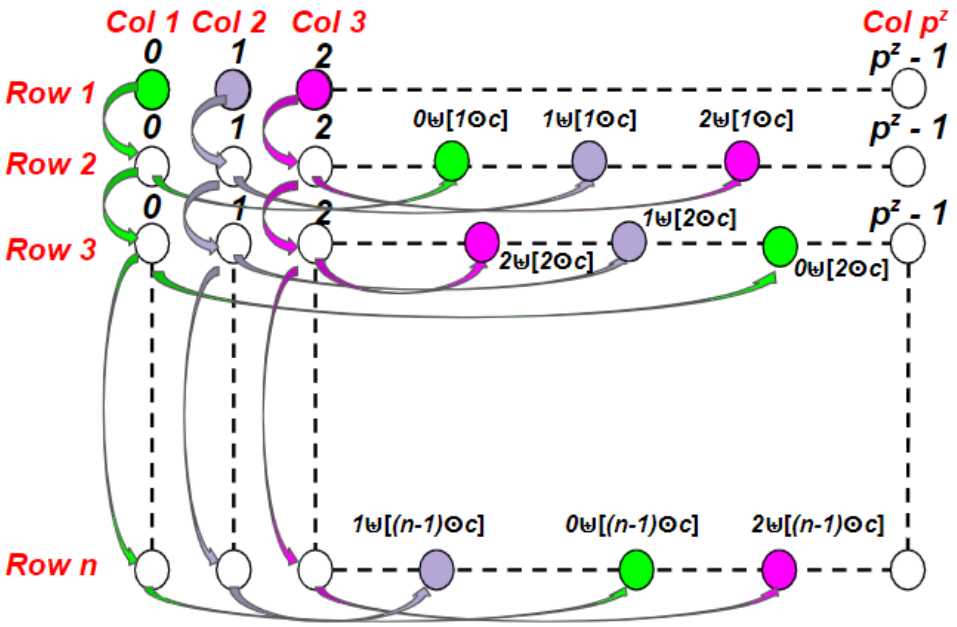

Figure 11: Pooling np^z samples into p^z parallel groups.

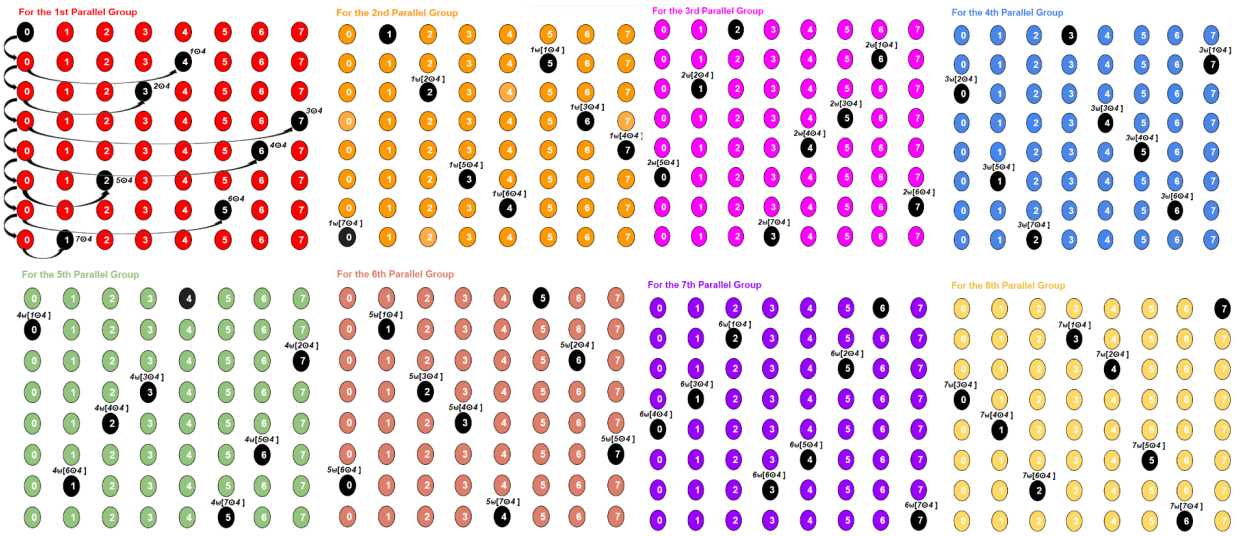

Figure 12: Pooling 64 samples into 8 parallel groups for $c = 4$.

2.4.2. Multidimensional Polynomial Pools Designs. Suppose the number of samples N is such that $(p^z)^{d-1} < N \leq (p^z)^d$ and $p^z|N$, for prime power p^z and $d \geq 2$. By the Division Algorithm, $N = a_1(p^z)^{d-1} + a_2p^z$, for some integers $1 \leq a_1 \leq p^z$ and $0 \leq a_2p^z \leq (p^z)^{d-1} - 1$. The Polynomial Pools (PP) Algorithm, to pool the N samples into p^z parallel groups, is described in Algorithm 2.3. When $d = 2$, the PP Algorithm reduces to the PPOL Design, see subsection 2.4.1. For any $d > 2$, we can fix $c_1, c_2, \cdots, c_{d-1} \in F_{p^z}$ to construct a collection of parallel groups, $\{\textbf{Group } w : 1 \leq w \leq p^z\}$.

Algorithm 2.3 General Polynomial Pools Design for pooling $a_1(p^z)^{d-1} + a_2p^z$ samples into a collection of p^z parallel groups

---

Step A : The N samples are arranged in a cuboid where the first a_1 vertical rows contain $(p^z)^{d-1}$ samples each. The $(a_1+1)th$ vertical row contains a_2p^z samples. Suppose among the N samples, we want to form p^z parallel groups of $a_1(p^z)^{d-2}+a_2$ samples each. Again, a collection of parallel groups do not contain any samples in common.

Step B : Fix a $c_1 \in F_{p^z}$ and in the frontmost face, group the $(1 + a_1)p^z$ samples there into p^z parallel groups by Algorithm 2.2. See figure 13.

Step C : Next, we can group the samples recursively, see figure 14. The first three parallel groups are listed below.

- **Group 1 (In Green)=** $A_1 \cup B_1$
  $A_1 = \cup_{h=0}^{a_1-1}\{h(p^z)^{d-1} + x : x \in$ **group** $(h \bigodot c_1 + 1)$ **of** $(p^z)^{d-1}$ **samples** $\}$
  $B_1 = \{a_1(p^z)^{d-1} + x : x \in$ **group** $(a_1 \bigodot c_1 + 1)$ **of** a_2p^z **samples** $\}$
- **Group 2 (In Blue)** $= A_2 \cup B_2$
  $A_2 = \cup_{h=0}^{a_1-1}\{h(p^z)^{d-1} + x : x \in$ **group** $(1 \biguplus [h \bigodot c_1] + 1)$ **of** $(p^z)^{d-1}$ **samples** $\}$
  $B_2 = \{a_1(p^z)^{d-1} + x : x \in$ **group** $(1 \biguplus [a_1 \bigodot c_1] + 1)$ **of** a_2p^z **samples** $\}$
- **Group 3 (In Pink)** $= A_3 \cup B_3$
  $A_3 = \cup_{h=0}^{a_1-1}\{h(p^z)^{d-1} + x : x \in$ **group** $(2 \biguplus [h \bigodot c_1] + 1)$ **of** $(p^z)^{d-1}$ **samples** $\}$
  $B_3 = \{a_1(p^z)^{d-1} + x : x \in$ **group** $(2 \biguplus [a_1 \bigodot c_1] + 1)$ **of** a_2p^z **samples** $\}$

Here, **Group** $w = A \cup B$, where:

- $A = \{[\sum_{i=1}^{d-1} h_i(p^z)^{d-i}] + (w-1) \biguplus [\biguplus_{i=1}^{d-1}(h_i \bigodot c_i)] + 1 : 0 \leq h_1 \leq a_1 - 1, h_i \in F_{p^z}\}$
- $B = \{a_1(p^z)^{d-1} + x : x \in$ group $((w-1) \biguplus (a_1 \bigodot c_1) + 1)$ of a_2p^z samples$\}$

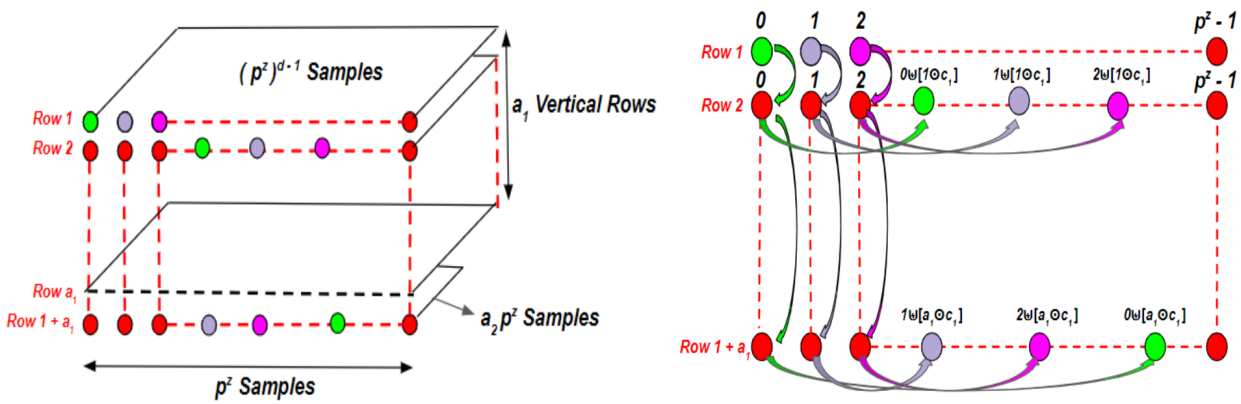

Figure 13: Grouping of Samples in Front Face.

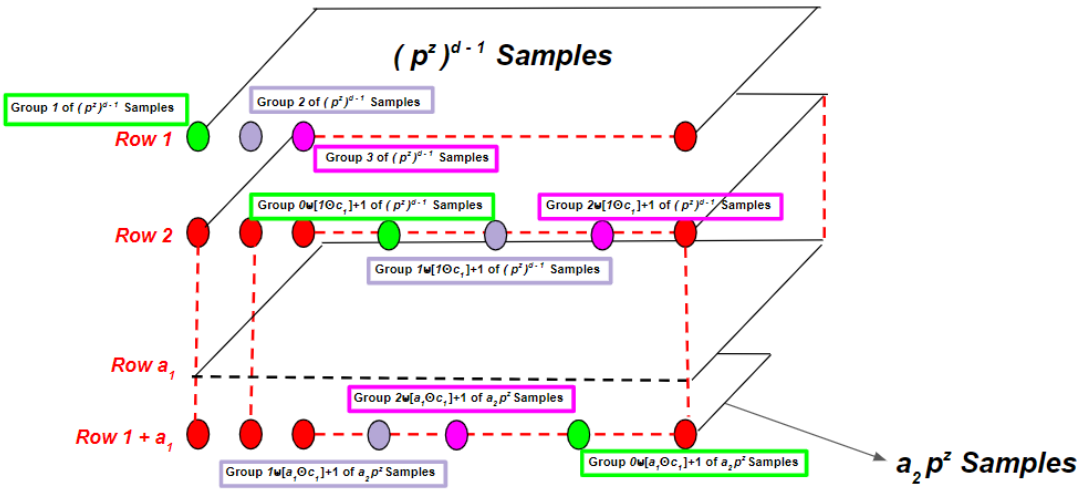

Figure 14: Multidimensional Grouping of Samples

Assign $c_i = a^i = a \odot a \odot \cdots \odot a$, for $1 \leq i \leq d-1$ and $a \in F_{p^z}$. Here, $\biguplus$ and $\odot$ denote addition and multiplication respectively in F_{p^z}. For each $a \in F_{p^z}$, we define a collection of p^z parallel groups to be $\{\text{Group } ap^z + b + 1 = A \cup B : b \in F_{p^z}\}$, where:

- $A = \{\sum_{i=1}^{d-1} h_i (p^z)^{d-i} + [\, [\, \biguplus_{i=1}^{d-1} (a^i \odot h_i)\,] \biguplus b\,] + 1 :\ h_i \in F_{p^z},\ 0 \leq h_1 \leq a_1 - 1\}$
- $B = \{\alpha_1 (p^z)^{d-1} + x : x \in group\ (b \biguplus (\alpha_1 \odot a^{d-1}) + 1)\ of\ a_2 p^z\ samples\}$

Theorem 2.1. *The following holds for the pooling of $(p^z)^{d-1} < N \leq (p^z)^d$ samples into $M = mp^z$ groups via the Polynomial Pools Algorithm.*

- *Every group is uniquely represented by $ap^z + b + 1$ for $a, b \in F_{p^z}$.*
- *Two distinct groups containing a similar sample each belongs to a distinct collection of p^z parallel groups respectively.*
- *Any pair of distinct samples is contained in at most $d-1$ groups, see Appendix D.*
- *The number of groups in the Polynomial Pools Pooling Matrix Design is at most p^{2z}.*

Suppose there are k positive samples. A negative sample has to be in $k(d-1)+1$ groups to be decoded if it is assigned to $k(d-1)$ positive groups. Hence, $p^z(k(d-1)+1)$ groups are sufficient to decode the N samples. As an example, let there be $N = 384$ samples. We are to assign them into $M = 48$ groups using the Polynomial Pools Algorithm. Here, $p^z = 8$, $d = 3$, $p^{2z} \leq N \leq p^{3z}$ and $m = 6$. The Polynomial Pools Algorithm can decode $k = 2$ positive samples since $m \geq k(d-1)+1$. The pooling of $N = 384$ samples into 6 distinct collections of 8 parallel groups, see Algorithm 2.4. For each $0 \leq a \leq 5$, $\{\textbf{Group}\ w :\ 1 \leq w \leq 8\}$ forms a collection of eight parallel groups, where $\textbf{Group}\ w = \{(w-1) \biguplus [h_1 \odot a] \biguplus [h_2 \odot a^2] + 8h_2 + 64h_1 + 1 : 0 \leq h_1 \leq 5, 0 \leq h_2 \leq 7\}$. By varying a, we can form another collection of eight parallel groups. This motivates the construction of the 48×384 P-Best Matrix, see section 3.

Algorithm 2.4 Polynomial Pools Design for pooling $N = 384$ samples into $M = 48$ groups

Step A : The ith vertical row, jth horizontal row and zth column entry, represents the sample labelled $64(i-1)+8(j-1)+z$, see figure 15. Suppose we want to pool the $N = 384$ samples into $M = 48$ groups, which consists of 6 distinct collections of 8 parallel groups

Step B : Fix $a = 2 \in F_8$ and in the front face, group the 48 samples here into 8 parallel groups by Algorithm 2.2, see figure 16. As highlighted in **Black**, $\{h \bigodot a + 64h + 1 : 0 \leq h \leq 5\}$ is contained in Group 1. In general, for $0 \leq a \leq 5$ and $1 \leq w \leq 8$, by grouping in the front face, $\{(w-1) \biguplus [h \bigodot a] + 64h + 1 : 0 \leq h \leq 5\} \subset$ *group* w

Step C : Next, we can group the samples in each vertical row. As highlighted in **Black**, see figure 17, for $a = 2$, by applying Algorithm 2.2 on each vertical row with $c = 4$,
Group 1 $= \{[h_1 \bigodot a] \biguplus [h_2 \bigodot a^2] + 8h_2 + 64h_1 + 1 : 0 \leq h_1 \leq 5, 0 \leq h_2 \leq 7\}$

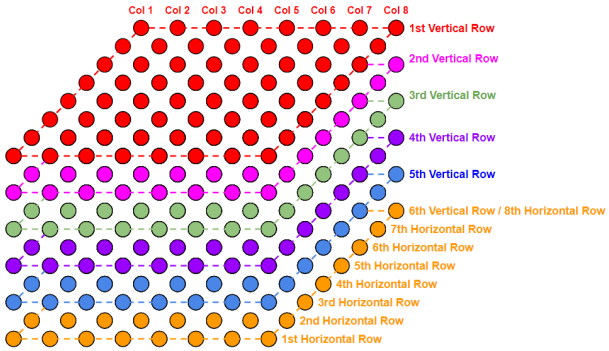

Figure 15: Arrangement of the 384 Samples.

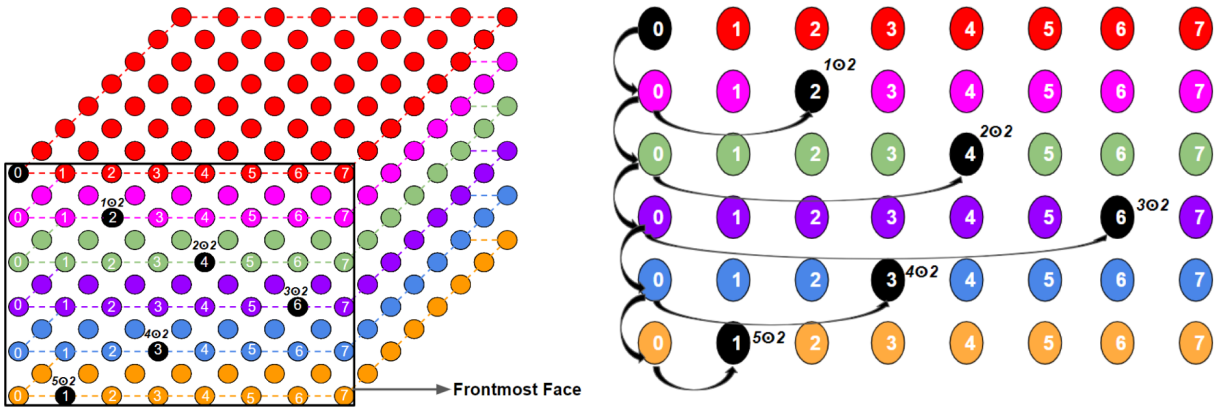

Figure 16: Grouping of Samples in Front Face

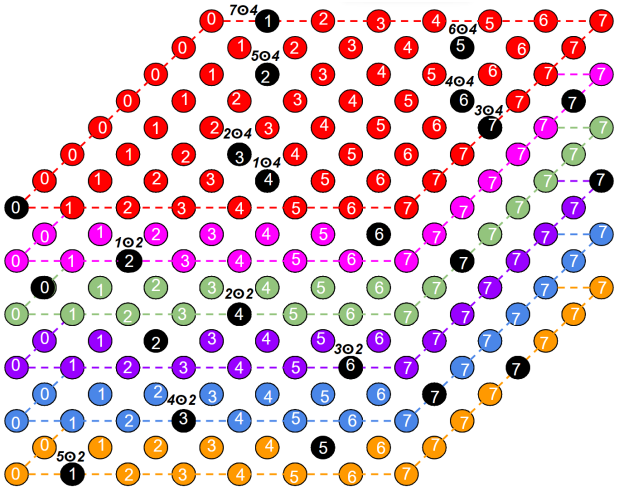

Figure 17: Grouping of Samples Recursively

2.4.3. Projective Geometry Design. The Projective Geometry Design serves to further minimize the number of groups M to identify the k positive samples via the Polynomial Pools Algorithm. Let $(p^z)^{d-1} < N \leq (p^z)^d$ and $p^z|N$. By the Polynomial Pools Algorithm, see subsection 2.4.2, $(k(d-1)+1)p^z$ groups are sufficient in decoding the N samples if k samples are positive. This forms $k(d-1)+1$ distinct collections of p^z parallel groups, where each parallel group contains $n = \frac{N}{p^z}$ samples. Now, suppose we have $N+h$ samples, where $1 \leq h \leq k(d-1)+1$. For each $1 \leq j \leq h$, among the jth collection of p^z parallel groups, we can assign the $(N+j)th$ sample into all the p^z parallel groups in addition to the n samples in each of them. Hence, among the first h distinct collections of p^z parallel groups, each parallel group contains $n+1$ samples. However, among the $(h+1)th$ to $(k(d-1)+1)th$ distinct collection of p^z parallel groups, each parallel group contains n samples. By applying this design to pool $N+h$ samples into $(k(d-1)+1)p^z$ groups, we can directly decode the statuses of the $N+h$ samples if we know that k samples are positive. Such a design is known as the Projective Geometry Design [1]. The Projective Geometry Design can be applied to the scenario when the number of samples N is large and we have to partition the samples into smaller sets. Separate pooling designs are applied in each set [13]. This is described in Algorithm 2.5 for a given number of samples N, number of positive samples k and prime power p^z. Here, the samples are tested in $m_1 p^z(k(d-1)+1)+\min\{m_2, (k(d-1)+1)p^z\}$ groups.

Algorithm 2.5 Effective Polynomial Pools Design by Prime Power

Let d be a nonnegative integer which satisfies $k(d-1)+1 \leq p^z$ and $d \leq \frac{log(N)}{log(p^z)} + 1$.
Let $N = m_1((p^z)^d + k(d-1)+1) + m_2$, where $m_1 \geq 0$ and $0 \leq m_2 \leq (p^z)^d + k(d-1)$

Step 1 : We will first pool each of the m_1 sets of $(p^z)^d + k(d-1)+1$ samples separately into $(k(d-1)+1)p^z$ groups via the Projective Geometry Design. Each group has $(p^z)^{d-1}+1$ samples

Step 2 : Among the remaining m_2 samples, we will pool them as follows.

- If $0 \leq m_2 < (k(d-1)+1)p^z$, we will test the remaining m_2 samples individually.
- If $(k(d-1)+1)p^z \leq m_2 \leq (p^z)^d$, we will group the m_2 samples accordingly via the Polynomial Pools Method in into $(k(d-1)+1)p^z$ groups, see subsection 2.4.2.
- Otherwise, we will assign the m_2 samples into $(k(d-1)+1)p^z$ groups via the Projective Geometry Design described above.

From Algorithm 2.5, we can vary $d \leq \frac{log(N)}{log(p^z)} + 1$ such that $k(d-1)+1 \leq p^z$, for each prime power p^z. This is to seek a Polynomial Pools Design which minimizes the number of groups to decode all samples. The code in MATLAB to implement this algorithm is accessible here Effective PP Matrix Design by Prime Power. An implementation of the code for $N = 23$ samples, $k = 1$ positive sample and prime power $p^z = 3$ is described in Appendix E.

2.5. Comparison Of Direct Decoding Methods. Given N samples and k positive samples, we want to deduce the minimum number of tests which are sufficient to identify all positives among the algorithms in subsections 2.2 to 2.4. Algorithm 2.6 compares these algorithms by iterating through each prime power $p^z \leq N$. Then, it returns the algorithm which minimizes the number of tests. The code in MATLAB to perform Algorithm 2.6 is accessible through the link Effective Matrix Design. An implementation of the code for $N = 23$ samples and $k = 1$ positive sample is described in Appendix E. However, one has to acknowledge that the pooling design returned may not be the most effective. There are other efficient methods to pool N samples into a smaller number of groups while identifying all k positives. One of such methods is the P-Best Matrix which uses a Compressed Sensing Method, see section 3.

Algorithm 2.6 Effective Matrix Design

Step 1 : For each prime power $p^z \leq N$, deduce the effective Polynomial Pools Design, see Algorithm 2.5, which minimizes the number of groups to identify all k positives.

Step 2 : Suppose $k \leq 2$. If p^z obeys the criteria to construct a Kirkman Triple Pooling Matrix, see subsection 2.3, then the Kirkman Triple Algorithm will be considered if $M < N$.

Step 3 : Lastly, consider pooling matrices constructed via the Hypercube Algorithms, see subsection 2.2, by comparing the number of tests required in the worst case.

3. P-Best Pooling Matrix Design. The P-Best Pooling Matrix is constructed for specifically 384 samples and 48 groups, see Algorithm 2.4. However, it does not use the same direct decoding method as a Polynomial Pools Pooling Matrix, see subsection 2.4.2, to identify positive samples. Instead, the P-Best Matrix can detect up to five true positive samples and one false positive sample via the Gradient Projection for Sparse Reconstruction (GPSR) Compressed Sensing Decoder [5]. Moreover, the P-Best Matrix incurs a lower cost for group testing as compared to the 114×348 Polynomial Pools Pooling Matrix for $k = 5$ positive samples [1]. The P-Best Pooling Matrix is applicable in typical group testings as physical PCR assays have 384 sample spots. In this section, we will briefly describe this pooling design which was implemented to screen healthcare workers [5].

3.1. P-Best Detection Algorithm. Let $A \in \{0,1\}^{48\times 384}$ denote the P-Best Matrix. Let $y \in (\mathbb{R}_{\geq 0})^{48\times 1}$ denote the quantitative viral loads of all the 48 groups. In here, we assume that the quantitative measurements of all viral loads cannot be negative. Samples and groups with $C(t)$ values less than 40 are considered positive [5]. Suppose $y = Ax$. The detection of the positive samples in x is described in Algorithm 3.1 [5], assuming that there are at most five positives.

Algorithm 3.1 P-Best Detection Algorithm

Step 1 : The estimated viral load of all the samples in x is determined via the COMP Algorithm followed by a Compressed Sensing Decoder, see subsections 3.2 and 3.3. Let N' denote the number of non-zero estimated viral loads in x and $n = \min\{20, N'\}$.

Step 2 : The n samples with the highest estimated viral loads from Step 1 will be identified, given that they are most likely to be positive. Then, we will consider all the possible 2^n subsets containing some of these n samples. Each subset is a vector $x' \in \{0,1\}^{384\times 1}$, where only the entries corresponding to the selected samples in x' are denoted as 1, which represents the positives.

Step 3 : Out of all the possible vectors x', the vector x' for which $||\psi(Ax') - \psi(y)||_1$ achieved its minimum will be selected, where $\psi : \mathbb{R}^{48\times 1} \longrightarrow \{0,1\}^{48\times 1}$, such that for $1 \leq i \leq 48$, the ith entry of $\psi(y)$ is 1 if the ith entry of y is not zero. Otherwise, the ith entry of $\psi(y)$ will be zero.

3.2. COMP Algorithm. The COMP Algorithm is used to decode the sure negative samples and high confidence positive samples. A sample is described to be sure negative if it belongs to a group with 0 viral load. Sure negative samples have 0 viral load. A sample is described to be high confidence positive if it belongs to a group with viral load greater than 0 such that all the other samples are sure negative [8]. The COMP Algorithm will return a smaller sub-matrix after eliminating all the rows in the original P-Best Matrix which represents the negative groups and the columns representing the sure negative samples. Then, it is left for us to apply a Compressed Sensing Decoder to decode the statuses of the remaining undecodable samples x' based on the smaller $M \times N$ submatrix A' and the remaining non-zero groups y', where $M \leq 48$ and $N \leq 384$.

3.3. Compressed Sensing Algorithms. In this section, we will briefly describe numerical methods which can be implemented to decode the viral loads of the remaining N samples in x'. For $\epsilon \geq 0$, consider the problem of finding a vector x' which satisfies the following condition, where $||x'||_0$ denotes the number of non-zero entries in x' and $x' \geq 0_{N\times 1}$ denotes a non-negative vector $x' \in \mathbb{R}^{N\times 1}$.

$$\min\{||x'||_0 : ||y' - A'x'||_2 \leq \epsilon \ , x' \geq 0_{N\times 1}\} \tag{3.1}$$

However, solving the above problem may be NP hard [8]. Therefore, several Compressed Sensing Algorithms have been developed to numerically solve the above problem. In this section, we will explore the Gradient Projection for Sparse Reconstruction (GPSR), which is a possible numerical method to solve the quadratic program, $argmin\{||x'||_1 : x' \geq 0_{N\times 1}$ and $||y' - A'x'||_2^2 \leq \epsilon\}$, which is a convex relaxation of (3.1). This is equivalent to solving the following unconstrained optimization problem, where $\tau \geq 0$ is a real constant.

$$\text{argmin}\{\frac{1}{2}||y' - A'x'||_2^2 + \tau||x'||_1 : x' \geq 0_{N\times 1}\} \tag{3.2}$$

By letting $z = x'$, it is equivalent to solving

$$\text{argmin}\{c^T z + \frac{1}{2} z^T B z : z \geq 0_{N\times 1}\} \tag{3.3}$$

Where $c = \tau 1_{N\times 1} - (A')^T y'$ and $B = (A')^T A'$.

Hence, we define $F(z)$, $\nabla F(z)$ and $H_F(z)$ as follows :

$$F(z) = c^T z + \frac{1}{2} z^T B z \tag{3.4}$$

$$\nabla F(z) = c + Bz \tag{3.5}$$

$$H_F(z) = B \tag{3.6}$$

where $B = (A')^T A'$ is a positive semi-definite matrix. The algorithm to perform the GPSR Compressed Sensing Method are described in Appendix F [4], with the goal of minimizing $F(z)$ while ensuring that $z \geq 0_{N\times 1}$. In [4], the Basic GPSR Algorithm and the Second Order GPSR Algorithm, otherwise known as the GPSR BB Algorithm, have been proposed for the sparse recovery of $z \geq 0_{N\times 1}$. The comparison of the performance of the GPSR Compressed Sensing Algorithm with other types of Compressed Sensing methods is illustrated in the diagram below [4]. Compared to the Orthogonal Matching Pursuit (OMP), one can conclude that the GPSR Algorithm is better in error performance as the estimated vector x' incurs close to no error if the original vector x' is a sparse vector with very few non zero entries. Here, MSE $= \frac{1}{N}||\text{Actual } x' - \text{Estimated } x'||_2^2$.

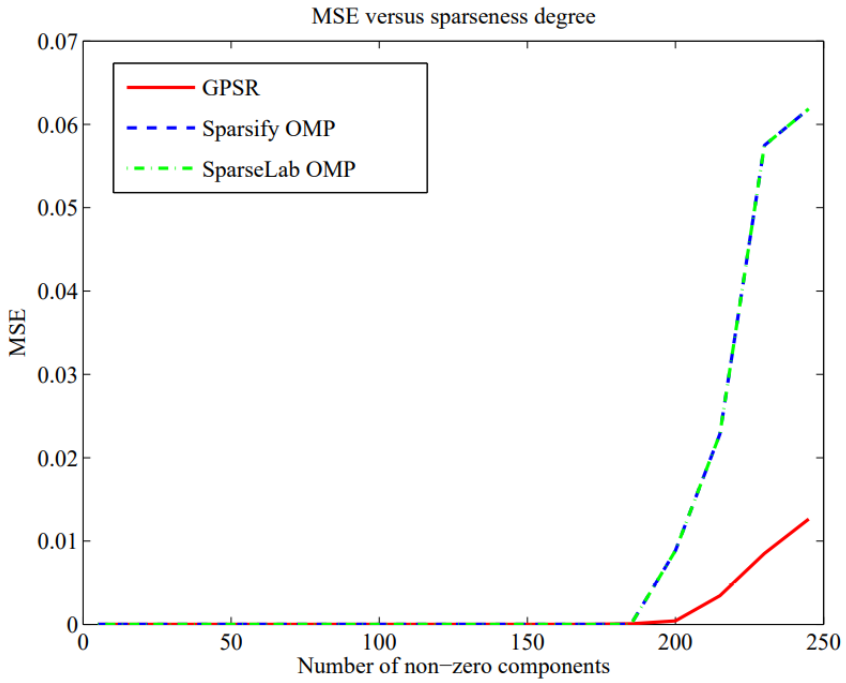

Figure 18: Comparison of Compressed Sensing Methods [4]

4. Conclusions. An Effective Pooling Matrix Design represents the group testing strategy which minimizes the number of groups when exactly identifying all positive samples. This manuscript focuses on comparing pooling matrices constructed by the Hypercube [11], Kirkman Triples [6, 8] and Polynomial Pools [1, 2] Algorithms in performing direct decoding methods if we know that up to k out of N samples are positive. However, Compressed Sensing Methods are able to numerically decode out the statuses of the samples while minimizing the number of groups to test by solving optimization problems [5, 4]. Hence, this remains a further aspect for us to explore.

Appendix A. Constructions and operations of prime power fields.

Theorem A.1. *For a prime number p, $(Z_p, \bigoplus, \bigotimes)$ is a finite field of p elements such that for all $x, y \in Z_p$, $x \bigoplus y \equiv x + y \ [mod\ p]$ and $x \bigodot y \equiv x * y \ [mod\ p]$.*

Theorem A.2. *Let $F_p[x]$ denote the set of all polynomials over Z_p. For all $z \geq 2$, F_{p^z} is isomorphic to $F_p[x]/(b(x))$ where $b(x) \in F_p[x]$ is a monic irreducible degree z polynomial over the field Z_p, where p is a prime number.[3]*

Proof. Given Z_p is a field, $F_p[x]$ is an euclidean domain. Hence, for all $f(x) \in F_p[x]$, $f(x)$ can be represented uniquely as such, $f(x) = [q(x) \bigotimes b(x)] \bigoplus k(x)$ where $q(x), k(x) \in F_p[x]$ and $0 \leq$ degree of $k(x) <$ degree of $b(x) = z$. Therefore, $k(x) \equiv f(x) \ [mod\ b(x)]$ uniquely for each $f(x) \in F_p[x]$. $F_p[x]/(b(x)) = \{\bigoplus_{h=0}^{z-1}[c_h \bigotimes x^h] : c_h \in Z_p\}$ is isomorphic to a set of polynomials over Z_p of degree at most $z-1$. All $\bigoplus_{h=0}^{z-1}[c_h \bigotimes x^h] \in F_p[x]/(b(x))$ can be uniquely represented by $\sum_{i=0}^{z-1} c_i p^i \in F_{p^z}$ and we hence, define $(F_{p^z}, \biguplus, \bigodot) \cong (F_p[x]/(b(x)), \biguplus, \bigodot)$ to be the finite field of p^z elements. We define the addition and multiplication operators in $F_p[x]/(b(x))$ to be as such

- For all $f(x), g(x) \in F_p[x]/(b(x))$, $f(x) \biguplus g(x) \equiv f(x) \bigoplus g(x) \ [mod\ b(x)]$.
- For all $f(x), g(x) \in F_p[x]/(b(x))$, $f(x) \bigodot g(x) \equiv f(x) \bigotimes g(x) \ [mod\ b(x)]$. ∎

Theorem A.3 (Operations in prime power fields). *Recall that the following properties hold in all fields.*

- $(\forall_{x,y} \in F_{p^z})(x \biguplus y = y \biguplus x \in F_{p^z})$
- $(\exists!_0 \in F_{p^z})(\forall_x \in F_{p^z})(x \biguplus 0 = 0 \biguplus x = x)$
- $(\forall_x \in F_{p^z})(\exists!_{-x} \in F_{p^z})(x \biguplus (-x) = (-x) \biguplus x = 0)$
- $(\forall_{x,y} \in F_{p^z} \setminus \{0\})(x \bigodot y = y \bigodot x \in F_{p^z} \setminus \{0\})$
- $(\exists!_1 \in F_{p^z} \setminus \{0\})(\forall_x \in F_{p^z})(x \bigodot 1 = 1 \bigodot x = x)$
- $(\forall_x \in F_{p^z} \setminus \{0\})(\exists!_{x^{-1}} \in F_{p^z} \setminus \{0\})(x \bigodot x^{-1} = x^{-1} \bigodot x = 1)$
- $(\forall_k \in F_{p^z})(k \bigodot 0 = 0 \bigodot k = 0)$

Appendix B. Explicit constructions of Kirkman Triple Pooling Matrix. The appendix describes on the explicit construction of the Kirkman Triple Pooling Matrix introduced in subsection 2.3. The operations in the $(F_{p^z}, \biguplus, \bigodot)$ field are described in Appendix A.

Theorem B.1 (Pooling matrices for $M = 2p^z + 1$ groups and $N = (4t+1)p^z$ samples). *Let $g = \lfloor \frac{p^z - 1}{2} \rfloor \in F_{p^z}$, be the generator of the F_{p^z} field. Let m be the solution to $2 \bigodot g^m = g^t \biguplus 1$. A possible $M \times N$ pooling matrix can be constructed here where the following holds for $0 \leq e \leq p^z - 1$:*

- *The $(e+1, e(4t+1)+1) - entry$, $(e+p^z+1, e(4t+1)+1) - entry$ and $(2p^z+1, e(4t+1)+1) - entry$ are both Ones.*
- *For $0 \leq j \leq 2$ and $1 \leq i \leq t$, the $(g^{i+2jt} \biguplus e+1, e(4t+1)+jt+i+1) - entry$, $(g^{i+2jt+t} \biguplus e+1, e(4t+1)+jt+i+1) - entry$ and $([g^{i+2jt+m} \biguplus e] + p^z + 1, e(4t+1)+jt+i+1) - entry$ are both Ones.*
- *For $1 \leq j \leq t$, the $([g^{j+m+t} \biguplus e] + p^z + 1, e(4t+1)+3t+j+1) - entry$, $([g^{j+m+3t} \biguplus e] + p^z + 1, e(4t+1)+3t+j+1) - entry$ and $([g^{j+m+5t} \biguplus e] + p^z + 1, e(4t+1)+3t+j+1) - entry$ are both Ones.*

Theorem B.2 (Pooling matrices for $M = 3p^z$ groups and $N = (9t+1)p^z$ samples). *A possible $M \times N$ Kirkman Triple Pooling Matrix can be constructed here where the following holds for $0 \le e \le p^z - 1$:*

- *The $(e+1, e(6t+1)+1)-entry$, $(e+p^z+1, e(6t+1)+1)-entry$ and $(e+2p^z+1, e(6t+1)+1)-entry$ are both Ones.*
- *For $0 \le j \le t-1$, the $(g^j \biguplus e+1, e(6t+1)+2+j)-entry$, $(g^{j+2t} \biguplus e+1, e(6t+1)+2+j)-entry$, $(g^{j+4t} \biguplus e+1, e(6t+1)+2+j)-entry$, $([g^j \biguplus e]+p^z+1, e(6t+1)+t+2+j)-entry$, $([g^{j+2t} \biguplus e]+p^z+1, e(6t+1)+t+2+j)-entry$, $([g^{j+4t} \biguplus e]+p^z+1, e(6t+1)+t+2+j)-entry$, $([g^j \biguplus e]+2p^z+1, e(6t+1)+2t+2+j)-entry$, $([g^{j+2t} \biguplus e]+2p^z+1, e(6t+1)+2t+2+j)-entry$, $([g^{j+4t} \biguplus e]+2p^z+1, e(6t+1)+2t+2+j)-entry$, $(g^j \biguplus e+1, (6t+1)^2+jp^z+e+1)-entry$, $([g^{j+2t} \biguplus e]+p^z+1, (6t+1)^2+jp^z+e+1)-entry$ and $([g^{j+4t} \biguplus e]+2p^z+1, (6t+1)^2+jp^z+e+1)-entry$ are both Ones.*
- *For $t \le j \le 2t-1$, the $([g^j \biguplus e]+1, e(6t+1)+2t+2+j)-entry$, $([g^{j+2t} \biguplus e]+p^z+1, e(6t+1)+2t+2+j)-entry$ and $([g^{j+4t} \biguplus e]+2p^z+1, e(6t+1)+2t+2+j)-entry$ are both Ones.*
- *For $3t \le j \le 4t-1$, the $([g^j \biguplus e]+1, e(6t+1)+t+2+j)-entry$, $([g^{j+2t} \biguplus e]+p^z+1, e(6t+1)+t+2+j)-entry$ and $([g^{j+4t} \biguplus e]+2p^z+1, e(6t+1)+t+2+j)-entry$ are both Ones.*
- *For $5t \le j \le 6t-1$, the $([g^j \biguplus e]+1, e(6t+1)+2+j)-entry$, $([g^{j+2t} \biguplus e]+p^z+1, e(6t+1)+2+j)-entry$ and $([g^{j+4t} \biguplus e]+2p^z+1, e(6t+1)+2+j)-entry$ are both Ones.*
- *For $2t \le j \le 3t-1$, the $(g^j \biguplus e+1, (7t+1)(6t+1)+(j-2t)p^z+e+1)-entry$, $([g^{j+2t} \biguplus e]+p^z+1, (7t+1)(6t+1)+(j-2t)p^z+e+1)-entry$ and $([g^{j+4t} \biguplus e]+2p^z+1, (7t+1)(6t+1)+(j-2t)p^z+e+1)-entry$ are both Ones.*
- *For $4t \le j \le 5t-1$, the $(g^j \biguplus e+1, (8t+1)(6t+1)+(j-4t)p^z+e+1)-entry$, $([g^{j+2t} \biguplus e]+p^z+1, (8t+1)(6t+1)+(j-4t)p^z+e+1)-entry$ and $([g^{j+4t} \biguplus e]+2p^z+1, (8t+1)(6t+1)+(j-4t)p^z+e+1)-entry$ are both Ones.*

Appendix C. Properties of PPOL Algorithm, refer to subsection 2.4.1.

Theorem C.1. *For each $c \in F_{p^z}$, we can define a collection of p^z parallel groups to be $\{\{hp^z + [b \biguplus (h \bigodot c)] + 1 : h \in F_{p^z}\} : b \in F_{p^z}\}$. Groups which are parallel to each other do not contain any samples in common.*

Proof. $(\forall_{b_1,b_2} \in F_{p^z})(\forall_{c,h} \in F_{p^z})(b_1 \neq b_2 \leftrightarrow b_1 \biguplus [h \bigodot c] \neq b_2 \biguplus [h \bigodot c])$.
Therefore, for all $b_1, b_2 \in F_{p^z}$ and $b_1 \neq b_2$,
$\{hp^z + [b_1 \biguplus (h \bigodot c)] + 1 : h \in F_{p^z}\} \bigcap \{hp^z + [b_2 \biguplus (h \bigodot c)] + 1 : h \in F_{p^z}\} = \emptyset$

Theorem C.2. *For each $c_1, b_1, b_2 \in F_{p^z}$ and $c_2 \in F_{p^z} \setminus \{c_1\}$, we define $\{hp^z + [b_1 \biguplus (h \bigodot c_1)] + 1 : h \in F_{p^z}\}$ and $\{hp^z + [b_2 \biguplus (h \bigodot c_2)] + 1 : h \in F_{p^z}\}$ to be a pair of non parallel groups. Each pair of non parallel groups contains exactly one sample in common.*

Proof. $hp^z + [b_1 \biguplus [h \bigodot c_1]] + 1 = hp^z + [b_2 \biguplus [h \bigodot c_2]] + 1 \rightarrow h \bigodot [c_1 \biguplus (-c_2)] = b_2 \biguplus (-b_1)$.
As $c_1 \in F_{p^z}$ and $c_2 \in F_{p^z} \setminus \{c_1\}$, $c_1 \biguplus (-c_2) \in F_{p^z} \setminus \{0\}$, we can uniquely determine $[c_1 \biguplus (-c_2)]^{-1}$ such that $[c_1 \biguplus (-c_2)]^{-1} \bigodot [c_1 \biguplus (-c_2)] = 1$. Therefore, $h = [b_2 \biguplus (-b_1)] \bigodot [c_1 \biguplus (-c_2)]^{-1}$ is the unique solution which satisfies $hp^z + [b_1 \biguplus [h \bigodot c_1]] + 1 = hp^z + [b_2 \biguplus [h \bigodot c_2]] + 1$

Appendix D. Properties of Polynomial Pools Algorithm, refer to subsection 2.4.2.

Theorem D.1. *Any Pair of Distinct Samples grouped via the Polynomial Pool Algorithm is contained in at most $d-1$ Groups.*

Proof. Let the number of samples be $N = (p^z)^d$. Suppose, we know that two samples s_1, s_2 are both contained in group $ap^z + b + 1$, where for $1 \leq i \leq d-1$, $h_i, h_i' \in F_{p^z}$, $s_1 = \sum_{i=1}^{d-1} h_i (p^z)^{d-i} + [\, b \biguplus [\, \biguplus_{i=1}^{d-1} [\, h_i \bigodot a^i \,]\,]\,]$ and $s_2 = \sum_{i=1}^{d-1} h_i' (p^z)^{d-i} + [\, b \biguplus [\, \biguplus_{i=1}^{d-1} [\, h_i' \bigodot a^i \,]\,]\,]$. Given each group is uniquely represented by $ap^z + b + 1$ for an $a, b \in F_{p^z}$, s_1 and s_2 will be contained in the same group iff $-u_1 \biguplus [\, \biguplus_{i=1}^{d-1} [\, h_i \bigodot a^i \,] = -b = -u_2 \biguplus [\, \biguplus_{i=1}^{d-1} [\, h_i' \bigodot a^i \,]$, where $u_1 = s_1 - \sum_{i=1}^{d-1} h_i (p^z)^{d-i}$ and $u_2 = s_2 - \sum_{i=1}^{d-1} h_i' (p^z)^{d-i}$. Therefore,

$$[u_2 \biguplus -u_1] \biguplus [\, \biguplus_{i=1}^{d-1} [\, [\, h_i \biguplus -h_i' \,] \bigodot a^i \,]\,] = 0 \tag{D.1}$$

Since the above equation represents a polynomial of variable a of degree at most $d-1$ over F_{p^z}, there are at most $d-1$ values of $a \in F_{p^z}$ such that equation (D.1) is satisfied. Hence, we can show that samples s_1 and s_2 can be contained together in at most $d-1$ distinct groups. ∎

Appendix E. Code runs in Matlab.

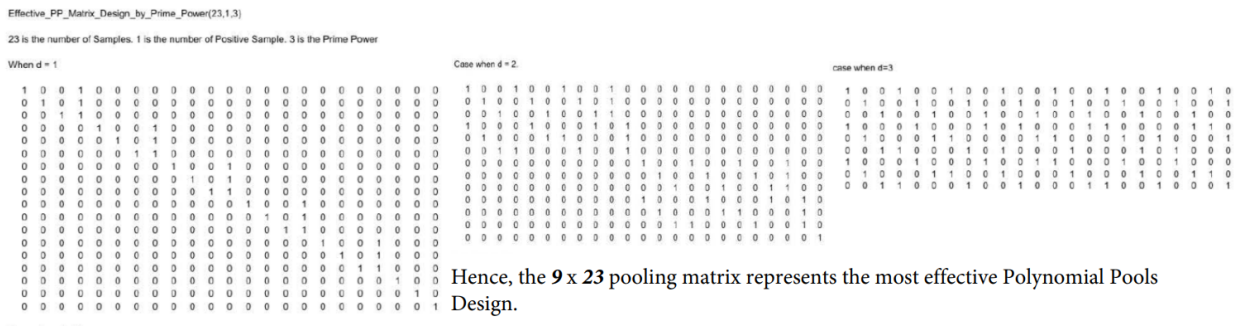

Figure 19: Effective PP Design for 23 samples, 1 positive and prime power 3, refer to subsection 2.4.3

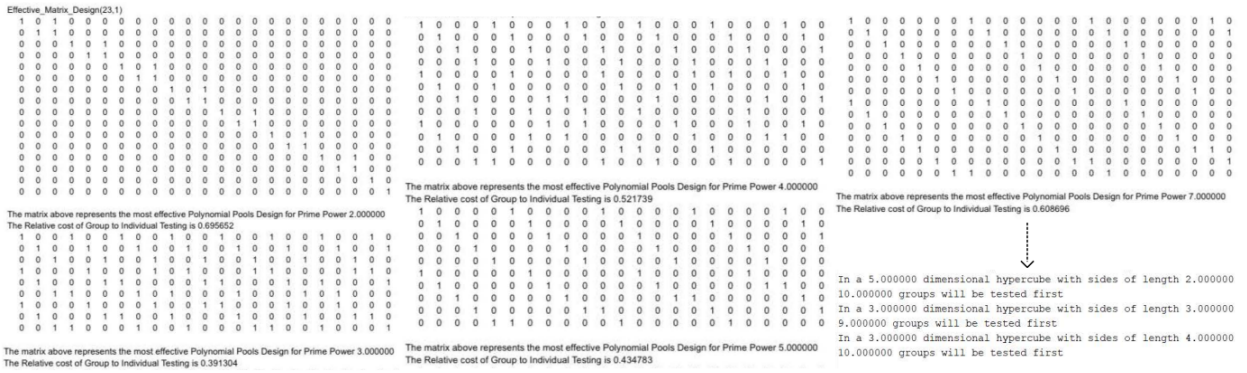

Figure 20: Effective Matrix Design for 23 samples and 1 positive sample, refer to subsection 2.5

The code to perform the above operations in Python and R are accessible through the links Effective Matrix Design in Python and Effective Matrix Design in R Studio respectively

Appendix F. GPSR Algorithms, refer to subsection 3.3.

Algorithm F.1 GPSR Basic Algorithm

Step 1 : Set $z_0 = 0_{N\times 1}$, $k = 0$, $\lambda \in (0,1)$, $\mu \in (0, \frac{1}{2})$ and $\alpha_{min},\ \alpha_{max} \in R$.

Step 2 : Define $g_k \in R^{N\times 1}$ such that $(g_k)_i = (\nabla F(z_k))_i$ if $(z_k)_i > 0$ or $(\nabla F(z_k))_i < 0$. Otherwise, $(g_k)_i = 0$.

Step 3 : Set $F(z_k - \alpha g_k) = c^T(z_k - \alpha g_k) + \frac{1}{2}(z_k - \alpha g_k)^T B(z_k - \alpha g_k)$.
$\nabla_\alpha F(z_k - \alpha g_k) = -(g_k)^T(c + Bz_k) + \alpha(g_k)^T B(g_k)$ and $\nabla^2_\alpha F(z_k - \alpha g_k) = (g_k)^T B(g_k) \geq 0$. Therefore, $F(z_k - \alpha g_k)$ is minimized when $\nabla_\alpha F(z_k - \alpha g_k) = 0$ and is achieved when
$\alpha = \frac{\nabla F(z_k)^T g_k}{(g_k)^T B g_k} = \alpha_k = \frac{(g_k)^T g_k}{(g_k)^T B g_k}$

Step 4 (Backtracking Line Search with Steepest Descent Method) : Let $\beta_k = mid\{\alpha_{min}, \alpha_k, \alpha_{max}\}$ and γ_k be the first value in $\{(\lambda)^h \beta_k : h \geq 0\}$ such that $F((z_k - \gamma_k \nabla F(z_k))_+) \leq F(z_k) - \mu(\nabla F(z_k))^T(z_k - (z_k - \gamma_k \nabla F(z_k))_+)$. Set $z_{k+1} = (z_k - \gamma_k \nabla F(z_k))_+$.

Step 5 (Termination) : Suppose we aim to determine a termination criterion which takes into account on how much the sparsity pattern in z_k has changed in recent iterations.
Define $\tau_k = \{1 \leq i \leq N : ith\ entry\ of\ z_k \neq 0\}$. If $\frac{|(\tau_{k+1} \cup \tau_k) \backslash (\tau_{k+1} \cap \tau_k)|}{|\tau_{k+1}|} \leq threshold$, set z_{k+1} to be the estimated value of x'. Otherwise, $k \leftarrow k + 1$ and go back to Step 2.

Algorithm F.2 GPSR Barzilai and Borwein Algorithm

Step 1 : Set $z_0 \geq 0_{N\times 1}$, $k = 0$, $a_{min},\ a_{max} \in R$ and $a_{min} \leq a_0 \leq a_{max}$

Step 2 : Compute $\delta_k = (z_k - a_k \nabla F(z_k))_+ - z_k$

Step 3 : Set $F(z_k - \lambda\delta_k) = c^T(z_k - \lambda\delta_k) + \frac{1}{2}(z_k - \lambda\delta_k)^T B(z_k - \lambda\delta_k)$.
$\nabla_\lambda F(z_k - \lambda\delta_k) = -(\delta_k)^T(c + Bz_k) + \lambda(\delta_k)^T B(\delta_k)$ and $\nabla^2_\lambda F(z_k - \lambda\delta_k) = (\delta_k)^T B(\delta_k) \geq 0$. Therefore, $F(z_k - \lambda\delta_k)$ is minimized when $\nabla_\lambda F(z_k - \lambda\delta_k) = 0$ and is achieved when
$\lambda = \frac{(\delta_k)^T \nabla F(z_k)}{(\delta_k)^T B \delta_k}$. Set $\lambda_k = mid\{0, \frac{(\delta_k)^T \nabla F(z_k)}{(\delta_k)^T B \delta_k}, 1\}$ and $z_{k+1} = z_k + \lambda_k \delta_k$

Step 4 : If $(\delta_k)^T B \delta_k = 0$, set $a_{k+1} = a_{max}$. Otherwise, set $a_{k+1} = mid\{a_{min}, \frac{(\delta_k)^T \delta_k}{(\delta_k)^T B \delta_k}, a_{max}\}$

Step 5 (Termination) : Similar to that of Algorithm F.1

Acknowledgments. This project was submitted as my honours thesis for my bachelor's degree in the National University of Singapore, Department of Mathematics. I would like to thank my project supervisor, Associate Professor Chu Delin, for recommending this as a topic for my Final Year Project. Besides, I would like to thank my project examiner, Dr Timo Sprekeler, who encouraged me to explore the materials in greater depths. Next, I would like to thank Dr Johannes J Brust for his group testing algorithms which has allowed me to view statistical pooling from a different angle, using the properties of finite prime power fields. Furthermore, I would like to thank Associate Professor Roger Tan Choon Ee, for advising me to write my ideas to cater to the general understanding of the readers. Lastly, I would like to thank the Reviewer and Associate Editor for providing constructive comments to improve the manuscript.

REFERENCES

[1] David Brust and Johannes J. Brust *Effective Matrix Designs for COVID-19 Group Testing, BMC Bioinformatics 24, 26 (2023)*, available at https://www.medrxiv.org/content/10.1101/2022.08.23.22279137v1.full.pdf+html

[2] Johannes J. Brust, University of California San Diego, *Matrix Designs for COVID-19 Group Testing* (Spring 2022), available at http://www.johannesbrust.com/image68.pdf, International Linear Algebra Society, Issue Number 68, Pages 9 to 16.

[3] Keith Conrad , University of Connecticut, *Finite Fields*, available at https://kconrad.math.uconn.edu/blurbs/galoistheory/finitefields.pdf

[4] Mario A. T. Figueiredo, Robert D. Nowak, Stephen J. Wright *Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems* (Year 2007) available at https://nowak.ece.wisc.edu/GPSR.pdf

[5] Noam Shental *Efficient high-throughput SARS-CoV-2 testing to detect asymptomatic carriers* available at https://www.science.org/doi/10.1126/sciadv.abc5961

[6] Ray-Chaudhuri, D. K. and Wilson, Richard M. *Solution of the Kirkman Schoolgirl Problem*, Proceedings of Symposia in Pure Mathematics, 1971, Vol 19 available at https://doi.org/10.1090/pspum/019 Internet Access: https://math.stackexchange.com/questions/4509843/efficient-way-to-rotate-through-partitions-with-subsets-of-size-three/4510645#4510645

[7] Rajamani Barathidasan, Ferdina Marie Sharmila, Ratchagadasse Vimal Raj, Gounassegarane Dhanalakshmi, Gunalan Anitha, Rahul Dhodapkar *Pooled sample testing for COVID-19 diagnosis: Evaluation of bi-directional matrix pooling strategies*(16th October 2021), available at https://www.sciencedirect.com/science/article/pii/S0166093422000714

[8] Sabyasachi Ghosh, IIT Bombay *A Compressed Sensing Approach to Pooled RT-PCR Testing for COVID-19 Detection*, available at https://www.scienceopen.com/document_file/be3451c2-2288-4ec1-85de-f3dc818abb67/PubMedCentral/be3451c2-2288-4ec1-85de-f3dc818abb67.pdf

[9] Sabyasachi Ghosh *Tapestry: A Single-Round Smart Pooling Technique for COVID-19 Testing* available at https://www.medrxiv.org/content/10.1101/2020.04.23.20077727v2

[10] Varlam Kutateladze and Ekaterina Seregin, *Fast and Efficient Data Science Techniques for COVID-19 Group Testing* (July 2021), available at https://jds-online.org/journal/JDS/article/561/file/pdf.

[11] Wolfgang Rauch, Universität Innsbruck, *High prevalence group testing in epidemiology with geometrically inspired algorithms (June 2023)*, available at https://www.researchsquare.com/article/rs-2966307/v1.pdf

[12] Yi-Jheng Lin, Che-Hao Yu, Tzu-Hsuan Liu, Cheng-Shang Chang, Fellow, IEEE, and Wen-Tsuen Chen, Life Fellow, IEEE, *Constructions and Comparisons of Pooling Matrices for Pooled Testing of COVID-19* (15 Jun 2021), available at https://arxiv.org/pdf/2010.00060.pdf.

[13] Yaniv Erlich, Anna Gilbert, Hung Ngo, Atri Rudra, Nicolas Thierry-Mieg, Mary Wootters, Dina Zielinski, and Or Zuk *Biological screens from linear codes: theory and tools* (25th December 2015) available at https://www.biorxiv.org/content/biorxiv/early/2015/12/25/035352.full.pdf