This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3121709, IEEE Transactions on Network Science and Engineering

Constructions and Comparisons of Pooling Matrices for Pooled Testing of COVID-19

Yi-Jheng Lin, Che-Hao Yu, Tzu-Hsuan Liu, Cheng-Shang Chang, *Fellow, IEEE,*
and Wen-Tsuen Chen, *Life Fellow, IEEE*

Abstract—In comparison with individual testing, group testing is more efficient in reducing the number of tests and potentially leading to tremendous cost reduction. There are two key elements in a group testing technique: (i) the pooling matrix that directs samples to be pooled into groups, and (ii) the decoding algorithm that uses the group test results to reconstruct the status of each sample. In this paper, we propose a new family of pooling matrices from packing the pencil of lines (PPoL) in a finite projective plane. We compare their performance with various pooling matrices proposed in the literature, including 2D-pooling, P-BEST, and Tapestry, using the two-stage definite defectives (DD) decoding algorithm. By conducting extensive simulations for a range of prevalence rates up to 5%, our numerical results show that there is no pooling matrix with the lowest relative cost in the whole range of the prevalence rates. To optimize the performance, one should choose the right pooling matrix, depending on the prevalence rate. The family of PPoL matrices can dynamically adjust their construction parameters according to the prevalence rates and could be a better alternative than using a fixed pooling matrix.

Index Terms—group testing, perfect difference sets, finite projective planes.

1 Introduction

COVID-19 pandemic has deeply affected the daily life of many people in the world. The current strategy for dealing with COVID-19 is to reduce the transmission rate of COVID-19 by preventive measures, such as contact tracing, wearing masks, and social distancing. One problematic characteristic of COVID-19 is that there are asymptomatic infections [1]. As those asymptomatic infections are unaware of their contagious ability, they can infect more people if they are not yet been detected [2]. As shown in the recent paper [3], massive COVID-19 testing in South Korea on Feb. 24, 2020, can greatly reduce the proportion of undetectable infected persons and effectively reduce the transmission rate of COVID-19.

Massive testing for a large population is very costly if it is done one at a time. For a population with a low prevalence rate, group testing (or pool testing, pooled testing, batch testing) that tests a group by mixing several samples together can achieve a great extent of saving testing resources. As indicated in the recent article posted on the US FDA website [4], the group testing approach has received a lot of interest lately. Also, in the US CDC's guidance for the use of pooling procedures in SARS-CoV-2 [5], it defines three types of tests: (i) *diagnostic testing* that is intended to identify occurrence at the individual level and is performed when there is a reason to suspect that an individual may be infected, (ii) *screening testing* that is intended to identify occurrence at the individual level even if there is no reason to suspect an infection, and (iii) *surveillance testing* includes ongoing systematic activities, including collection, analysis, and interpretation of health-related data. The general guidance for diagnostic or screening testing using a pooling strategy in [5] (quoted below) basically follows the two-stage group testing procedure invented by Dorfman in 1943 [6]:

> *"If a pooled test result is negative, then all specimens can be presumed negative with the single test. If the test result is positive or indeterminate, then all the specimens in the pool need to be retested individually."*

The Dorfman two-stage algorithm is a very simple group testing strategy. Recently, there are more sophisticated group testing algorithms proposed in the literature, see, e.g., [7], [8], [9], [10]. Instead of pooling a sample into a single group, these algorithms require diluting a sample and then splitting it into multiple groups (pooled samples). Such a procedure is specified by a *pooling matrix* that directs each diluted sample to be pooled into a specific group. The test results of pooled samples are then used for decoding (reconstructing) the status of each sample. In short, there are two key elements in a group testing strategy: (i) the pooling matrix, and (ii) the decoding algorithm.

As COVID-19 is a severe contagious disease, one should be very careful about the decoding algorithm used for reconstructing the testing results of persons. Though decoding algorithms that use soft information for group testing, including various compressed sensing algorithms in [8], [9], [10], [11], [12], might be more efficient in reducing the number of tests, they are more prone to have false positives and false negatives. A false positive might cause a person to be quarantined for 14 days and thus losing 14 days of work. On the other hand, a false negative might have an infected person wandering around the neighborhood and cause more people to be infected. In view of this, it is important to have group testing results that are as "definite" as individual testing results (in a noiseless setting).

- *The authors are with the Institute of Communications Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan R.O.C.*
Email: s107064901@m107.nthu.edu.tw; chehaoyu@gapp.nthu.edu.tw; carina000314@gmail.com; cschang@ee.nthu.edu.tw; wtchen@cs.nthu.edu.tw.

Manuscript received October 13, 2020; revised June 11, 2021 and September 6, 2021; accepted October 17, 2021. This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant 109-2221-E-007-091-MY2, and in part by Qualcomm Technologies under Grant SOW NAT-435533. (Corresponding author: Yi-Jheng Lin.)

Following the CDC guidance [5], we use the decoding algorithm, called the *definite defectives (DD)* algorithm in the literature (see Algorithm 2.3 of the monograph [13]), that can have definite testing results. The DD algorithm first identifies negative samples from a negative testing result of a group (as advised by the CDC guidance [5]). Such a step is known as the combinatorial orthogonal matching pursuit (COMP) step in the literature [13]. Then the DD algorithm identifies positive samples if they are in a group with only one positive sample. Not every sample can be decoded by the DD algorithm. As the Dorfman two-stage algorithm, samples that are not decoded by the DD algorithm go through the second stage, and they are tested individually. We call such an algorithm the two-stage DD algorithm.

One of the main objectives of this paper is to compare the performance of various pooling matrices proposed in the literature, including 2D-pooling [7], P-BEST [8], and Tapestry [9], [10], using the two-stage DD decoding algorithm. In addition to these pooling matrices, we also propose a new construction of a family of pooling matrices from packing the pencil of lines (PPoL) in a finite projective plane. The family of PPoL pooling matrices has very nice properties: (i) both the column correlation and the row correlation are bounded by 1, and (ii) there is a freedom to choose the construction parameters to optimize performance. To measure the amount of saving of a group testing method, we adopt the performance measure, called the *expected relative cost* in [6]. The expected relative cost is defined as the ratio of the expected number of tests required by the group testing technique to the number of tests required by the individual testing. We then measure the expected relative costs of these pooling matrices for a range of prevalence rates up to 5%. Some of the main findings of our numerical results are as follows:

(i) There is no pooling matrix that has the lowest relative cost in the whole range of the prevalence rates considered in our experiments. To optimize the performance, one should choose the right pooling matrix, depending on the prevalence rate.

(ii) The expected relative costs of the two pooling matrices used in Tapestry [9], [10] are high compared to the other pooling matrices considered in our experiments. Its performance, in terms of the expected relative cost, is even worse than the (optimized) Dorfman two-stage algorithm. However, Tapestry is capable of decoding most of the samples in the first stage. In other words, the percentages of samples that need to go through the second stage are the smallest among all the pooling matrices considered in our experiments.

(iii) P-BEST [8] has a very low expected relative cost when the prevalence rate is below 1%. However, its expected relative cost increases dramatically when the prevalence rate is above 1.3%.

(iv) 2D-pooling [7] has a low expected relative cost when the prevalence rate is near 5%. Unlike Tapestry, P-BEST, and PPoL that rely on robots for pipetting, the implementation of 2D-pooling is relatively easy by humans.

(v) There is a PPoL pooling matrix with column weight 3 that outperforms the P-BEST pooling matrix for the whole range of the prevalence rates considered in our experiments (up to 5%). We suggest using that PPoL pooling matrix up to the prevalence rate of 2% and then switch to other PPoL pooling matrices with respect to the increase of the prevalence rate. The detailed suggestions are shown in Table 4 of Section 6.

The paper is organized as follows: in Section 2, we briefly review the group testing problem, including the mathematical formulation and the DD decoding algorithm. In Section 3, we introduce the related works that are used in our comparison study. We then propose the new family of PPoL pooling matrices in Section 4. In Section 6, we conduct extensive simulations to compare the performance of various pooling matrices using the two-stage DD algorithm. The paper is concluded in Section 7, where we discuss possible extensions for future works.

2 Review of Group Testing

2.1 The problem statement

Consider the group testing problem with M samples (indexed from $1, 2, \dots, M$), and N groups (indexed from $1, 2, \dots, N$). The M samples are pooled into the N groups (pooled samples) through an $N \times M$ binary matrix $H = (h_{n,m})$ so that the m^{th} sample is pooled into the n^{th} group if $h_{n,m} = 1$ (see Figure 1). Such a matrix is called the *pooling matrix* in this paper. Note that a pooling matrix corresponds to the biadjacency matrix of an $N \times M$ bipartite graph. Let $x = (x_1, x_2, \dots, x_M)$ be the binary state vector of the M samples and $y = (y_1, y_2, \dots, y_N)$ be the binary state vector of the N groups. Then

$$y = Hx, \tag{1}$$

where the matrix operation is under the Boolean algebra (that replaces the usual addition by the OR operation and the usual multiplication by the AND operation). The main objective of group testing is to decode the vector x given the observation vector y under certain assumptions. In this paper, we adopt the following basic assumptions for binary samples:

(i) Every sample is binary, i.e., it is either positive (1) or negative (0).

(ii) Every group is binary, and a group is positive (1) if there is at least one sample in that group is positive. On the other hand, a group is negative (0) if all the samples pooled into that group are negative.

If we test each sample one at a time, then the number of tests for M samples is M, and the average number of tests per sample is 1. The key advantage of using group testing is that the number of tests per sample can be greatly reduced. One important performance measure of group testing, called the *expected relative cost* in [6], is the ratio of the expected number of tests required by the group testing technique to the number of tests required by the individual testing. The main objective of this paper is to compare the expected relative costs of various group testing methods.

2.2 The definite defectives (DD) decoding algorithm

In this section, we briefly review the definite defectives (DD) algorithm (see Algorithm 2.3 of [13]). The DD algorithm first identifies negative samples from a negative testing result of a group. Such a step is known as the combinatorial orthogonal matching pursuit (COMP) step. Then the DD algorithm identifies positive samples if they are in a group with only one positive

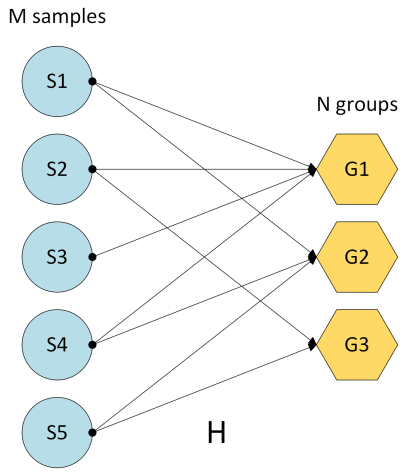


Figure 1: Pooled testing represented by a bipartite graph.

ALGORITHM 1: The definite defectives (DD) algorithm for binary samples

```
Input An $N \times M$ pooling matrix $H$ and a binary
  $N$-vector $y$ of the group test result.
Output an $M$-vector for the test results of the $M$
  samples.
0: Initially, every sample is marked “un-decoded.”
1: If there is a negative group, then all the samples pooled
  into that group are decoded to be negative.
2: The edges of samples decoded to be negative in the
  bipartite graph are removed from the graph.
3: Repeat from Step 1 until there is no negative group.
4: If there is a positive group with exactly one (remaining)
  sample in that group, then that sample is decoded to
  positive.
5: Repeat from Step 4 until no more samples can be
  decoded.
```

sample. The detailed steps of the DD algorithm are outlined in Algorithm 1.

In Figure 2, we provide an illustrating example for Algorithm 1. In Figure 2 (a), the test result of $G2$ is negative, and thus the three samples $S1$, $S4$ and $S5$, are decoded to be *negative*. In Figure 2 (b), the edges that are connected to the samples $S1$, $S4$ and $S5$, are removed from the bipartite graph. In Figure 2 (c), the test results of the two groups $G1$ and $G3$ are positive. As $S2$ is the only sample in $G3$, $S2$ is decoded to be *positive*.

Note that one might not be able to decode all the samples by the above decoding algorithm. For instance, if a particular sample is pooled into groups that all have at least one positive sample, then there is no way to know whether that sample is positive or negative. As shown in Figure 3, the sample $S3$ cannot be decoded by the DD algorithm as the test results of the three groups are the same no matter if $S3$ is positive or not.

As shown in Lemma 2.2 of [13], one important guarantee of the DD algorithm is that there is no false positive.

Proposition 1. *([13], Lemma 2.2) Assume that all the testing results are correct. Then (i) all the samples that are decoded to be negative in Step 1 of Algorithm 1 are definite negatives, and (ii) all the samples that are decoded to be positive in Step 4 of Algorithm 1 are definite positives. As such, there are no false positives in Algorithm 1.*

In order to resolve all the “un-decoded” samples, we add another stage by individually testing each “un-decoded” sample. This leads to the following two-stage DD algorithm in Algorithm 2.

ALGORITHM 2: The two-stage definite defectives (DD2) algorithm for binary samples

```
Input An $N \times M$ pooling matrix $H$ and a binary
  $N$-vector $y$ of the group test result.
Output an $M$-vector for the test results of the $M$
  samples.
1: Run the DD algorithm in Algorithm 1.
2: For those “un-decoded” samples, test them one at a
  time.
```

3 Related Works

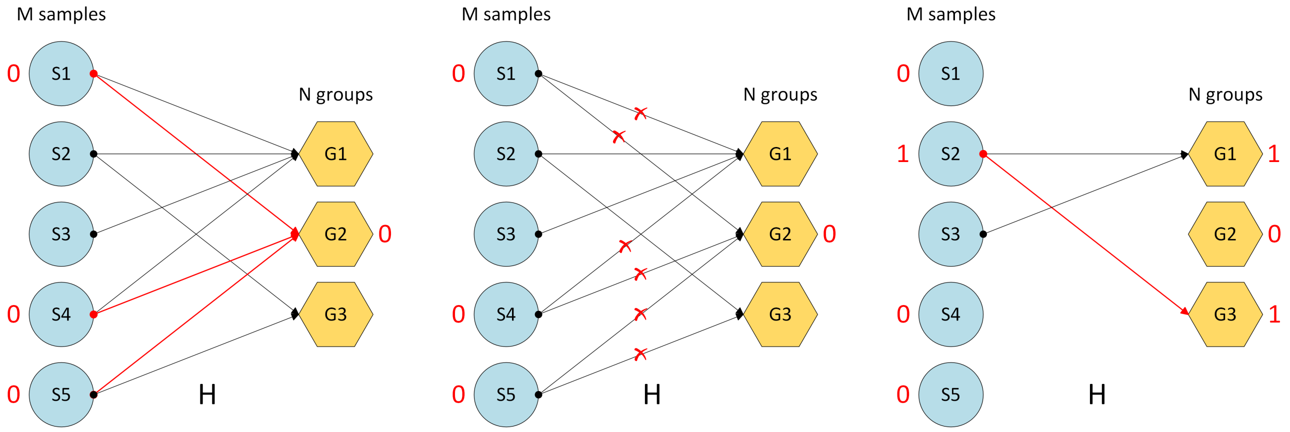
In [14], [15], [16], it was shown that a single positive sample can still be detected even in pools of 5-32 samples for the standard RT-qPCR test of COVID-19. Such an experimental result provides supporting evidence for group testing of COVID-19. In the following, we review four group testing strategies proposed in the literature for COVID-19.

The Dorfman two-stage algorithm [17]: For the case that $N = 1$, i.e., every sample is pooled into a single group, the DD2 algorithm is simply the original Dorfman two-stage algorithm [6], i.e., if the group of M samples is tested negative, then all the M samples are ruled out. Otherwise, all the M samples are tested individually. Suppose that the prevalence rate is r_1. Then the expected number of tests to decode the M samples by the Dorfman two-stage algorithm is $1 + (1 - (1 - r_1)^M)M$. As such, the expected relative cost (defined as the ratio of the expected number of tests required by the group testing technique to the number of tests required by the individual testing in [6]) is $\frac{M+1}{M} - (1 - r_1)^M$. As shown in Table I of [6], the optimal group size M is 11 with the expected relative cost of 20% when the prevalence rate r_1 is 1%.

2D-pooling [7]: On a 96-well plate, there are 8 rows and 12 columns. Pool the samples in the same row (column) into a group. This results in 20 groups for 96 samples. One advantage of this simple 2D-pooling strategy is to minimize pipetting errors.

P-BEST [8]: P-BEST [8] uses a 48 × 384 pooling matrix constructed from the Reed-Solomon code [18] for pooled testing of COVID-19. For the pooling matrix, each sample is pooled into 6 groups, and each group contains 48 samples. In [8], the authors proposed using a compressed sensing algorithm called the Gradient Projection for Sparse Reconstruction (GPSR) algorithm for decoding. Though it is claimed in [8] that the GPSR algorithm can detect up to 1% of positive carriers, there is no guarantee that every decoded sample (by the GPSR algorithm) is correct.

Tapestry [9], [10]: The Tapestry scheme [9], [10] uses the Kirkman triples to construct their pooling matrices. For the pooling matrix in [9], [10], each sample is pooled into 3 groups (in their experiments, some samples are only pooled into 2 groups). As such, it is sparser than that used by P-BEST. However, one of the restrictions for the pooling matrices constructed from the Kirkman triples is that the column weights must be 3. Such a restriction limits its applicability to optimize its performance according to



(a) Step 1: All the samples pooled into that negative groups are decoded to be negative.

(b) Step 2: The edges of negative samples are removed.

(c) Step 4: Exactly one sample in a positive group is decoded to be positive.

Figure 2: An illustration for the DD algorithm.

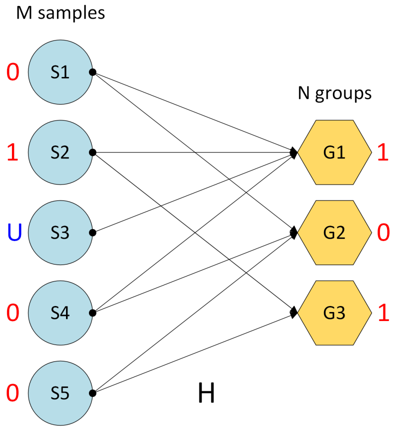


Figure 3: An un-decoded sample.

the prevalence rate. We note that a compressed-sensing-based decoding algorithm was proposed in [9], [10]. Such a decoding algorithm further exploits the viral load (Ct value) of each pool and reconstructs the Ct value of each positive sample. It is claimed to be viable not just with low ($< 4\%$) prevalence rates but even with moderate prevalence rates (5%-10%).

4 PPoL Constructions of Pooling Matrices

In this section, we propose a new family of pooling matrices from packing the pencil of lines (PPoL) in a finite projective plane. Our idea of constructing PPoL pooling matrices was inspired by the constructions of channel hopping sequences in the rendezvous search problem in cognitive radio networks and the constructions of grant-free uplink transmission schedules in 5G networks (see, e.g., [19], [20], [21], [22]), in particular, the channel hopping sequences constructed by the PPoL algorithm in [19].

A pooling matrix is said to be (d_1, d_2)-regular if there are exactly d_1 (resp. d_2) nonzero elements in each column (resp. row). In other words, the degree of every left-hand (resp. right-hand) node in the corresponding bipartite graph is d_1 (resp. d_2). The total number of edges in the bipartite graph is $d_1 M = d_2 N$ for a (d_1, d_2)-regular pooling matrix H. Define the (compressing) gain

$$G = \frac{M}{N} = \frac{d_2}{d_1}. \tag{2}$$

4.1 Perfect difference sets and finite projective planes

As our construction of the pooling matrix is from packing the pencil of lines in a finite projective plane, we first briefly review the notions of difference sets and finite projective planes.

Definition 2. (Difference sets) *Let $Z_p = \{0, 1, \ldots, p-1\}$. A set $D = \{a_0, a_1, \ldots, a_{k-1}\} \subset Z_p$ is called a (p, k, λ)-difference set if for every $(\ell \text{ mod } p) \neq 0$, there exist at least λ ordered pairs (a_i, a_j) such that $a_i - a_j = (\ell \bmod p)$, where $a_i, a_j \in D$. A $(p, k, 1)$-difference set is said to be* perfect *if there exists exactly one ordered pair (a_i, a_j) such that $a_i - a_j = (\ell \bmod p)$ for every $(\ell \text{ mod } p) \neq 0$.*

Definition 3. (Finite projective planes) *A finite projective plane of order m, denoted by $PG(2, m)$, is a collection of $m^2 + m + 1$ lines and $m^2 + m + 1$ points such that*

(P1) *every line contains $m + 1$ points,*
(P2) *every point is on $m + 1$ lines,*
(P3) *any two distinct lines intersect at exactly one point, and*
(P4) *any two distinct points lie on exactly one line.*

When m is a prime power, Singer [23] established the connection between an $(m^2 + m + 1, m + 1, 1)$-perfect difference set and a finite projective plane of order m through a collineation that maps points (resp. lines) to points (resp. lines) in a finite projective plane. Specifically, suppose that $D = \{a_0, a_1, \ldots, a_m\}$ is an $(m^2 + m + 1, m + 1, 1)$-perfect difference set with

$$a_0 = 0 < a_1 = 1 < a_2 < \ldots, < a_m < m^2 + m + 1. \tag{3}$$

(i) Let $\{0, 1, \ldots, m^2 + m\}$ be the $m^2 + m + 1$ points.
(ii) Let $p = m^2 + m + 1$ and $D_\ell = \{(a_0 + \ell) \bmod p, (a_1 + \ell) \bmod p, \ldots, (a_m + \ell) \bmod p\}$, $\ell = 0, 1, 2, \ldots, p-1$ be the $m^2 + m + 1$ lines.

Then these m^2+m+1 points and m^2+m+1 lines form a finite projective plane of order m.

4.2 The construction algorithm

In this section, we propose the PPoL algorithm for constructing pooling matrices. For this, one first constructs an $(m^2+m+1, m+1, 1)$-perfect difference set, $D = \{a_0, a_1, \dots, a_m\}$ with

$$a_0 = 0 < a_1 = 1 < a_2 < \dots, < a_m < m^2+m+1. \tag{4}$$

Let $p = m^2+m+1$ and

$$D_\ell = \{(a_0+\ell) \bmod p, (a_1+\ell) \bmod p, \dots, (a_m+\ell) \bmod p\}, \tag{5}$$

$\ell = 0, 1, 2, \dots, p-1$ be the p lines in the corresponding finite projective plane.

It is easy to see that the $m+1$ lines in the corresponding finite projective plane that contain point 0 are $D_0, D_{p-a_1}, D_{p-a_2}, \dots, D_{p-a_m}$. These $m+1$ lines are called the pencil of lines that contain point 0 (as the pencil point). As the only intersection of the $m+1$ lines is point 0, these $m+1$ lines, excluding point 0, are disjoint, and thus can be packed into Z_p. This is formally proved in the following lemma.

Lemma 4. *Let $D^0_{p-a_i} = D_{p-a_i} \backslash \{0\}$, $i = 1, 2, \dots, m$. Then $\{D_0, D^0_{p-a_1}, \dots, D^0_{p-a_m}\}$ is a partition of Z_p.*

Proof. First, note that $\{D_0, D_{p-a_1}, \dots, D_{p-a_m}\}$ are the $m+1$ lines that contain point 0. As any two distinct lines intersect at exactly one point, we know that for $i \neq 0$,

$$D_0 \cap D^0_{p-a_i} = \varnothing,$$

and that for $i \neq j$,

$$D^0_{p-a_i} \cap D^0_{p-a_j} = \varnothing.$$

Thus, they are disjoint.

As there are $m+1$ points in D_0 and m points in $D^0_{p-a_i}$, $D_0 \cup D^0_{p-a_1} \cup \dots \cup D^0_{p-a_m}$ contains $m+1+m^2$ points. These $m+1+m^2$ points are exactly the set of m^2+m+1 points in the finite projective plane of order m. ■

In Algorithm 3, we show how one can construct a pooling matrix from a finite projective plane. The idea is to first construct a bipartite graph with the line nodes on the left and the point nodes on the right. There is an edge between a point node and a line node if that point is in that line. Then we start trimming this line-point bipartite graph to achieve the needed compression ratio. Specifically, we select the subgraph with the m^2 line nodes that do not contain point 0 (on the left) and the $d_1 m$ point nodes in the union of d_1 pencil of lines (on the right).

Note that in Algorithm 3, the number of samples has to be m^2. However, this restriction may not be met in practice. A simple way to tackle this problem is by adding additional dummy samples to ensure that the total number of samples is m^2. In the literature, there are some sophisticated methods (see, e.g., the recent work [24]) that further consider the "balance" issue, i.e., samples should be pooled into groups as evenly as possible.

ALGORITHM 3: The PPoL algorithm

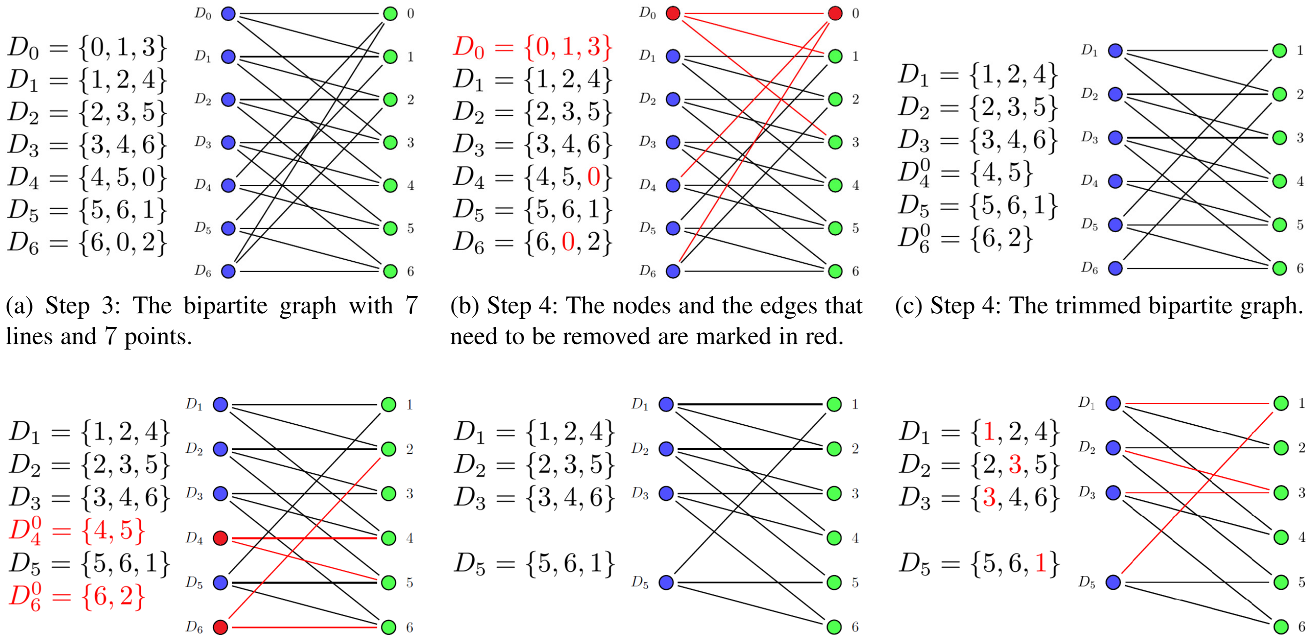
```
Input The number of samples $M = m^2$ with $m$ being a
  prime power, and the degree of each sample
  $1 \leq d_1 \leq m+1$.
Output An $N \times M$ binary pooling matrix $H$ with
  $M = m^2$ and $N = d_1 m$.
1: Let $p = m^2+m+1$ and construct a perfect difference
   set $D = \{a_0, a_1, \dots, a_m\}$ in $Z_p$ (with $a_0 = 0$ and
   $a_1 = 1$).
2: For $\ell = 0, 1, \dots, p-1$, let
   $D_\ell = \{(a_0+\ell) \bmod p, (a_1+\ell) \bmod p, \dots, (a_m+\ell) \bmod p\}$
   be the $p$ lines.
3: Construct a bipartite graph with the $p$ lines on the left
   and the $p$ points on the right. Add an edge between a
   point node and a line node if that point is in that line.
4: Remove point 0 and line 0 from the bipartite graph
   (and the edges attached to these two nodes). Let
   $G = (g_{n,\ell})$ be the $(m^2+m) \times (m^2+m)$ biadjacency
   matrix of the trimmed bipartite graph with $g_{n,\ell} = 1$ if
   point $n$ is in $D_\ell$.
5: Let $D^0_{p-a_i} = D_{p-a_i} \backslash \{0\}$, $i = 0, 1, 2, \dots, m$, be the
   $m+1$ pencil of lines that contain point 0.
6: Remove the $(p-a_i)^{th}$ column, $i = 1, 2, \dots, m$, in $G$
   to form an $(m^2+m) \times m^2$ biadjacency matrix $\tilde{G}$. Note
   that these $m$ columns correspond to the $m$ lines
   containing point 0.
7: Let $B = \cup_{i=0}^{d_1-1} D^0_{p-a_i}$ (select the first $d_1$ pencil of
   lines that contain point 0). Remove rows of $\tilde{G}$ that are
   not in $B$ to form a $d_1 m \times m^2$ biadjacency matrix $H$.
```

Example 5. (A worked example of the PPoL algorithm in Algorithm 3) *Let $m = 2$, $d_1 = 1$ be the inputs of Algorithm 3. In Step 1, let $p = m^2+m+1 = 7$ and construct the perfect difference set $D = \{a_0, a_1, a_2\} = \{0, 1, 3\}$ in Z_7. In Step 2, let $D_0, D_1, \dots, D_6$ be the 7 lines, where $D_0 = \{0, 1, 3\}$, $D_1 = \{1, 2, 4\}$, $D_2 = \{2, 3, 5\}$, $D_3 = \{3, 4, 6\}$, $D_4 = \{4, 5, 0\}$, $D_5 = \{5, 6, 1\}$, and $D_6 = \{6, 0, 2\}$. In Step 3, construct the bipartite graph with the 7 lines on the left and the 7 points on the right, and add an edge between a point node and a line node if that point is in that line. This bipartite graph is shown in Figure 4 (a). In Step 4, first remove point 0 and line 0 along with the edges attached to these two nodes from the bipartite graph. The nodes and the edges that need to be removed are marked in red in Figure 4 (b), and the trimmed bipartite graph is shown in Figure 4 (c). Then, let $G = (g_{n,\ell})$ be the 6×6 biadjacency matrix of the trimmed bipartite graph with $g_{n,\ell} = 1$ if point n is in D_ℓ, i.e.,*

$$G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \!\! \begin{array}{c} \begin{array}{cccccc} D_1 & D_2 & D_3 & D_4 & D_5 & D_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array} \tag{6}$$

In Step 5, let $D^0_{p-a_0} = D^0_0 = \{1, 3\}$, $D^0_{p-a_1} = D^0_6 = \{6, 2\}$ and $D^0_{p-a_2} = D^0_4 = \{4, 5\}$ be the 3 pencil of lines that contain point 0. In Step 6, remove the $(p-a_1)^{th}$ = 6th and the $(p-a_2)^{th}$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3121709, IEEE Transactions on Network Science and Engineering



(a) Step 3: The bipartite graph with 7 lines and 7 points.

(b) Step 4: The nodes and the edges that need to be removed are marked in red.

(c) Step 4: The trimmed bipartite graph.

(d) Step 6: The two lines that need to be removed are marked in red.

(e) Step 6: The bipartite graph after removing the two lines.

(f) Step 7: The points in set B along with the edges attached to these nodes are marked in red.

Figure 4: An example to demonstrate how the PPoL algorithm in Algorithm 3 works.

= 4th columns in G to form a 6×4 biadjacency matrix $\tilde{G}$, i.e.,

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} = \tilde{G} \tag{7}$$

The two lines that need to be removed are marked in red in Figure 4 (d), and the bipartite graph after removing the two lines are shown in Figure 4 (e). In Step 7, let $B = \cup_{i=0}^{d_1-1} D_{p-a_i}^0 = D_{p-a_0}^0 = D_0^0 = \{1,3\}$. Then, remove rows of $\tilde{G}$ that are not in B to form a 2×4 biadjacency matrix H, i.e.,

$$\tilde{G} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} = H \tag{8}$$

The points in set B along with the edges attached to these nodes are marked in red in Figure 4 (f). The output of Algorithm 3 in this example is the 2×4 binary pooling matrix H.

Proposition 6. *The degree of a line node is d_1 and the degree of a point node is m.*

Proof. As the remaining lines are the lines not containing point 0, each line then intersects with $D_{p-a_i}^0$ at exactly one point. Since there are d_1 pencil of lines that contain point 0, each line then intersects with $B = \cup_{i=1}^{d_1} D_{p-a_i}^0$ at exactly d_1 points. On the other hand, each of the points in B is in a line that contains point 0. As the lines that contain point 0 are removed, each point in B is in m lines of the remaining m^2 lines. ■

Proposition 7. *There is at most one* common *nonzero element in two rows (resp. columns) in the pooling matrix H from Algorithm 3, i.e., the inner product of two row vectors (resp. column vectors) is at most 1.*

Proof. This is because the bipartite graph with the biadjacency matrix H is a subgraph of the line-point bipartite graph corresponding to a finite projective plane. From (P3) and (P4) of Definition 3, any two distinct lines intersect at exactly one point, and any two distinct points lie on exactly one line. Thus, there is at most one common nonzero element in two rows (resp. columns) in H from Algorithm 3. ■

Corollary 8. *The girth (the minimum length of a cycle) of the bipartite graph with biadjacency matrix H is at least 6.*

Proof. As the length of a cycle in a bipartite graph must be an even number, it suffices to show that there does not exist a cycle of length 4. We prove this by contradiction. Suppose that there is a cycle of length 4. Suppose that this cycle contains two line nodes L_1 and L_2 and two point nodes P_1 and P_2. Then the intersection of the two lines L_1 and L_2 contains two points P_1 and P_2. This contradicts (P3) in Definition 3. ■

Theorem 9. *Consider using the $d_1 m \times m^2$ pooling matrix H from Algorithm 3 for a binary state vector x in a noiseless setting. If the number of positive samples in x is not larger than $d_1 - 1$,* *then every sample can be correctly decoded by the DD algorithm in Algorithm 1.*

Proof. Suppose that there are at most $d_1 - 1$ positive samples. We first show that every negative sample can be correctly decoded by the DD algorithm in Algorithm 1. Consider a negative sample. Since there are at most $d_1 - 1$ positive samples that can be pooled into the d_1 groups of this negative sample, and two different samples can be in a common group at most once (Proposition 7), there must be at least one group without positive samples (among the d_1 groups of this negative sample). Thus, this negative sample can be correctly decoded. Now consider a positive sample. Since there are at most $d_1 - 2$ positive samples that can be pooled into the d_1 groups of this positive sample, and two different samples can be in a common group at most once (Proposition 7), there must be at least one group in which this positive sample is the only positive sample. Thus, every positive sample can be correctly decoded. ■

4.3 Connection between the PPoL algorithm and the shifted transversal design

We note that there are other methods that can also generate bipartite graphs that satisfy the property in Proposition 7. For instance, in the recent paper [25], Täufer used the shifted transversal design to generate "mutlipools" (in Definition 1 of [25]) that satisfy the property in Proposition 7 when m is a prime (in Theorem 3 of [25]). In this section, we establish the connection between the PPoL design and the shift transversal design when m is restricted to a prime. We do this by identifying a mapping between these two designs in the following example.

Example 10. *Consider $m = 3$ in the PPoL algorithm. Then let $p = m^2+m+1 = 13$, and $D_0 = \{a_0, a_1, a_2, a_3\} = \{0, 1, 4, 6\}$ be a perfect difference set in Z_{13}. By using the PPoL algorithm in Algorithm 3, we obtain a bipartite graph with 9 samples (lines) and 12 groups (points) in Figure 5. In the following, we discuss the four cases with $d_1 = 1, 2, 3, 4$, respectively.*

(i) If $d_1 = 1$, $D^0_{p-a_0} = D^0_0 = \{1, 4, 6\}$. Then D_1, D_{10}, D_8 are in group 1, D_4, D_3, D_{11} are in group 4, and D_5, D_2, D_6 are in group 6. Thus, every sample is contained in $d_1 = 1$ group. (See the black points and lines in Figure 5.)

(ii) If $d_1 = 2$, $D^0_{p-a_0} = D^0_0 = \{1, 4, 6\}$ and $D^0_{p-a_1} = D^0_{12} = \{12, 3, 5\}$. Then, in addition to the pooling results in (i), D_1, D_4, D_5 are in group 5, D_{10}, D_3, D_2 are in group 3, and D_8, D_{11}, D_6 are in group 12. Thus, every sample is contained in $d_1 = 2$ groups. (See the black and green ones in Figure 5.)

(iii) If $d_1 = 3$, $D^0_{p-a_0} = D^0_0 = \{1, 4, 6\}$, $D^0_{p-a_1} = D^0_{12} = \{12, 3, 5\}$, and $D^0_{p-a_2} = D^0_9 = \{9, 10, 2\}$. Then, in addition to the pooling results in (i) and (ii), D_8, D_3, D_5 are in group 9, D_{10}, D_4, D_6 are in group 10, and D_1, D_{11}, D_2 are in group 2. Thus, every sample is contained in $d_1 = 3$ groups. (See the black, green, and red ones in Figure 5.)

(iv) If $d_1 = 4$, $D^0_{p-a_0} = D^0_0 = \{1, 4, 6\}$, $D^0_{p-a_1} = D^0_{12} = \{12, 3, 5\}$, $D^0_{p-a_2} = D^0_9 = \{9, 10, 2\}$, and $D^0_{p-a_3} = D^0_7 = \{7, 8, 11\}$. Then, in addition to the pooling results in (i), (ii) and (iii), D_1, D_3, D_6 are in group 7, D_8, D_4, D_2 are in group 8, and D_5, D_{10}, D_{11} are in group 11. Thus, every sample is contained in $d_1 = 4$ groups. (See the black, green, red, and orange ones in Figure 5.)

The above PPoL pooling strategy is the same as $(N, n, k) = (m^2, m, d_1)$-multipool in the shifted transversal design [25] if we arrange the 9 samples in the 3×3-square in Table 1. Specifically, pooling along rows yields the three groups $\{D_1, D_{10}, D_8\}$, $\{D_4, D_3, D_{11}\}$, and $\{D_5, D_2, D_6\}$. This corresponds to the case with $d_1 = 1$ in the PPoL design. On the other hand, pooling along columns yields the three groups $\{D_1, D_4, D_5\}$, $\{D_{10}, D_3, D_2\}$, and $\{D_8, D_{11}, D_6\}$. This corresponds to the case with $d_1 = 2$ in the PPoL design. Moreover, pooling with slope 1 (resp. 2) corresponds to the case with $d_1 = 3$ (resp. $d_1 = 4$).

Table 1: Arrangement of the 9 samples in a 3×3 rectangular grid.

D_1	D_{10}	D_8
D_4	D_3	D_{11}
D_5	D_2	D_6

In fact, these two constructions are closely related to orthogonal Latin squares [26]. For $n = 3$ (which is a prime power), there are exactly $n - 1 = 2$ mutually orthogonal Latin squares: $\{C^{(r)} = c^{(r)}_{i,j} : r = 1, 2\}$, where $c^{(r)}_{i,j} = (r * i + j)$ is in GF(3). With the "vertical" and "horizontal" cases, the maximum number of multiplicity k in the shifted transversal design is $n + 1 = 4$. Similarly, the maximum number of d_1 in the PPoL algorithm is $m + 1 = 4$. Moreover, pooling matrices that satisfy the decoding property in Theorem 9 are known as the superimposed codes in [27].

$$D_1 = \{1, 2, 5, 7\}$$
$$D_2 = \{2, 3, 6, 8\}$$
$$D_3 = \{3, 4, 7, 9\}$$
$$D_4 = \{4, 5, 8, 10\}$$
$$D_5 = \{5, 6, 9, 11\}$$
$$D_6 = \{6, 7, 10, 12\}$$
$$D_8 = \{8, 9, 12, 1\}$$
$$D_{10} = \{10, 11, 1, 3\}$$
$$D_{11} = \{11, 12, 2, 4\}$$

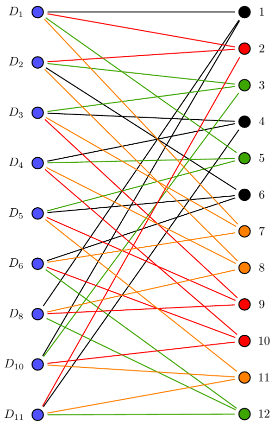


Figure 5: The bipartite graph obtained by using Algorithm 3 for Example10.

4.4 Probabilistic analysis of the PPoL pooling matrices

In this section, we conduct a probabilistic analysis of the PPoL pooling matrices. We make the following assumption:

(A1) All the samples are i.i.d. Bernoulli random variables. A sample is positive (resp. negative) with probability r_1 (resp. r_0). The probability r_1 is known as the prevalence rate in the literature.

Note that $r_1 + r_0 = 1$. Also, let q_1 (resp. q_0) be the probability that the group end of a randomly selected edge is positive (resp.

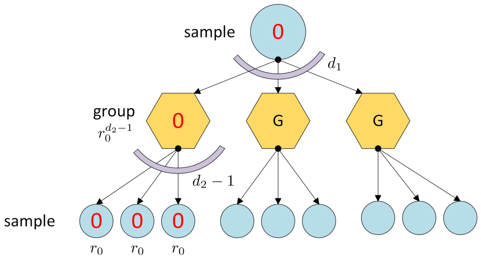


Figure 6: Computing the conditional probability p_0 by the tree evaluation method.

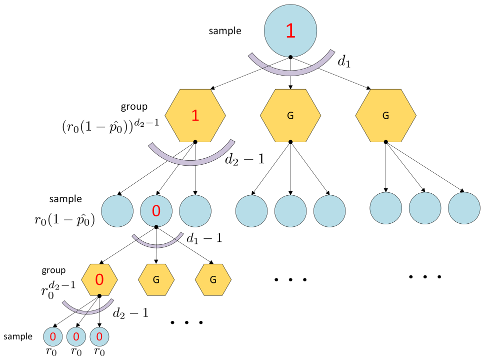


Figure 7: Computing the conditional probability p_1 by the tree evaluation method.

negative). Excluding the randomly selected edge, there are $d_2 - 1$ remaining edges in that group, and thus

$$q_0 = (r_0)^{d_2-1}, \quad (9)$$

$$q_1 = 1-(r_0)^{d_2-1}. \quad (10)$$

Let p_0 be the conditional probability that a sample *cannot* be decoded, given that the sample is a negative sample. Note that a negative sample can be decoded if at least one of its edges is in a negative group, excluding its edge (see Figure 6). Consider a negative sample, called the tagged sample. Since the girth of the bipartite graph of the pooling matrix is 6 (as shown in Corollary 8), the samples in the d_1 groups of the subtree of the tagged sample are distinct (see the tree expansion in Figure 6). Thus,

$$p_0 = (q_1)^{d_1} = (1-(r_0)^{d_2-1})^{d_1}. \quad (11)$$

Let $\hat{p}_0$ be the conditional probability that the sample end of a randomly selected edge *cannot* be decoded, given that the sample end is a negative sample. Note that the excess degree of a sample (excluding the randomly selected edge) is $d_1 - 1$. Analogous to the argument for (11) (see the bottom subtree of the tree expansion in Figure 7), we have

$$\hat{p}_0 = (q_1)^{d_1-1} = (1-(r_0)^{d_2-1})^{d_1-1}. \quad (12)$$

Let p_1 be the conditional probability that a sample *cannot* be decoded given that the sample is a positive sample. Note that a positive sample can be decoded if at least one of its edges is in a group in which all the edges are removed except the edge of the positive sample. Since an edge is removed if its sample end is a negative sample and that sample end is decoded to be negative, the probability that an edge is removed is $(1-\hat{p}_0)r_0$. If the tree expansion in Figure 7 is actually a tree, then

$$p_1 = (1-(r_0(1-\hat{p}_0))^{d_2-1})^{d_1}. \quad (13)$$

We note that the tree expansion in Figure 7 may *not* be a tree for a PPoL pooling matrix generated from Algorithm 3, the identity in (13) is only an approximation. A sufficient condition for the tree expansion in Figure 7 to be a tree of depth 4 is that the girth of the bipartite graph is larger than 8. (If the graph in Figure 7 is not a tree, i.e., there is a loop in that graph, then the girth of the bipartite graph is less than or equal to 8.) Unfortunately, the girth of a PPoL pooling matrix can only be proved to be at least 6. Since a sample cannot be decoded with probability $r_0p_0 + r_1p_1$, the average number of tests needed for the DD2 algorithm in Algorithm 2 to decode the M samples is $N + M(r_0p_0 + r_1p_1)$. The expected relative cost for the DD2 algorithm with an $N \times M$ pooling matrix is

$$\frac{N + M(r_0p_0 + r_1p_1)}{M} = \frac{1}{G} + r_0p_0 + r_1p_1, \quad (14)$$

where $G = M/N$ is the (compressing) gain of the pooling matrix in (2). Note that for a (d_1, d_2)-regular pooling matrix, we have from (2) that $G = d_2/d_1$. Thus, we can use (11), (13) and (14) to find the (d_1, d_2)-regular pooling matrix that has the lowest expected relative cost (though (13) is only an approximation for the pooling matrices constructed from the PPoL algorithm). In Table 2, we use grid search to find the (d_1, d_2)-regular pooling matrix with the lowest expected relative cost for various prevalence rates r_1 up to 10%. The search regions for the grid search are $2 \leq d_1 \leq 8$ and $d_1 \leq d_2 \leq 31$. In the last column of this table, we also show the expected relative cost of the Dorfman two-stage algorithm (Table I of [6]). As shown in this table, using the DD2 algorithm (with the optimal pooling matrices) has significant gains over the Dorfman two-stage algorithm. Unfortunately, not every optimal (d_1, d_2)-regular pooling matrix in Table 2 can be constructed by using the PPoL algorithm in Algorithm 3. In the next section, we will look for suboptimal pooling matrices that have small performance degradation.

Table 2: The (d_1, d_2)-regular pooling matrix with the lowest expected relative cost from (14).

r_1	d_1	d_2	cost (14)	Dorfman [6]
1%	3	31	0.1218	0.20
2%	4	29	0.1881	0.27
3%	4	22	0.2545	0.33
4%	4	17	0.3147	0.38
5%	3	12	0.3678	0.43
6%	3	11	0.4166	0.47
7%	3	10	0.4627	0.50
8%	2	7	0.5035	0.53
9%	2	6	0.5416	0.56
10%	2	6	0.5760	0.59

5 Noisy Decoding

In this section, we consider decoding for noisy binary samples. For this, we introduce the noisy model in [13].

Definition 11. *Define the probability transition function $p(1|k,\ell)$ (resp. $p(0|k,\ell)$) such that a group containing k samples, ℓ of*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3121709, IEEE Transactions on Network Science and Engineering

which are positive, the test result for the group is positive (resp. negative).

For the noiseless model discussed in the previous section, we have

$$p(1|k,\ell) = \left\{ \begin{array}{ll} 1 & \text{if } \ell \geq 1 \\ 0 & \text{otherwise} \end{array} \right. , \tag{15}$$

$$p(0|k,\ell) = \left\{ \begin{array}{ll} 1 & \text{if } \ell = 0 \\ 0 & \text{otherwise} \end{array} \right. . \tag{16}$$

There are several noisy models proposed in the literature (see, e.g., the monograph [13]). Among them, the *dilution noise* model might be a suitable one for the rt-PCR test. In the dilution noise model, the test result of a group containing ℓ positive samples follows a binomial distribution with parameters ℓ and $1-\epsilon$. Intuitively, a positive sample included in a group can be "diluted" with probability ϵ. The parameter ϵ is called the dilution probability. The transition probability functions for the dilution noise model are

$$p(1|k,\ell) = 1-\epsilon^{\ell}, \tag{17}$$

$$p(0|k,\ell) = \epsilon^{\ell}, \tag{18}$$

for all $k, \ell \geq 0$. Another way to view the dilution model is to view the bipartite graph (of the pooling matrix) as a random weighted graph, where the edge weights of the edges are independent Bernoulli random variables with parameter $1-\epsilon$. In the following analysis, we say an edge is diluted (resp. not diluted) if its edge weight is 0 (resp. 1). When an edge is diluted, the sample end of that edge does not affect the testing result of the group end of that edge. On the other hand, when an edge is not diluted and its sample end is positive, then the group end of that edge is positive.

For the dilution model, there might be false negatives and false positives if we use the DD algorithm for decoding. This is because a positive sample might be diluted during the pooling process and thus mistakenly decoded as a negative sample. On the other hand, a negative sample might be pooled into a group with a false negative and thus be mistakenly decoded as a positive sample by the DD algorithm (that assumes the only remaining sample in a positive group is positive). In order to ensure that there are no false positives, we could only run the COMP step in the DD algorithm and have the un-decoded samples tested one at a time at the second stage. However, there are still false negatives due to dilution. To reduce the false negatives, we propose using the K-combinatorial orthogonal matching pursuit (K-COMP) algorithm (see Algorithm 4) that only decodes negative samples if there are in at least K negative groups. When $K=1$, this reduces to the original COMP step in the DD algorithm.

Now we provide a probabilistic analysis of the K-COMP algorithm. As in Section 4.4, we let q_0 be the probability that the group end of a randomly selected edge is negative. Excluding the randomly selected edge, there are d_2-1 remaining edges in that group. Conditioning on the event that ℓ edges of these d_2-1 remaining edges are not diluted, the probability that the group end of a randomly selected edge is negative is r_0^{ℓ} (as in (9)). Thus, we have

$$\begin{aligned} q_0 &= \sum_{\ell=0}^{d_2-1} r_0^{\ell} \left(\begin{array}{c} d_2-1 \\ \ell \end{array} \right) (1-\epsilon)^{\ell} \epsilon^{d_2-1-\ell} \\ &= (r_0+r_1\epsilon)^{d_2-1}. \end{aligned} \tag{19}$$

ALGORITHM 4: The K-combinatorial orthogonal matching pursuit (K-COMP) algorithm for diluted binary samples

```
Input An $N \times M$ pooling matrix $H$ and a binary
  $N$-vector $y$ of the group test result.
Output an $M$-vector for the test results of the $M$
  samples.
0: Initially, every sample is marked "un-decoded."
1: If a sample is pooled in at least $K$ negative groups,
   then that sample is decoded to be negative.
2: For those "un-decoded" samples, test them one at a
   time.
```

Following the argument in Section 4.4, let p_0 be the conditional probability that a sample *cannot* be decoded, given that the sample is a negative sample. Note that a negative sample can be decoded if at least K of its edges are in negative groups, excluding its edges. Thus,

$$\begin{aligned} p_0 &= 1 - \sum_{k=K}^{d_1} \left(\begin{array}{c} d_1 \\ k \end{array} \right) (q_0)^k (1-q_0)^{d_1-k} \\ &= \sum_{k=0}^{K-1} \left(\begin{array}{c} d_1 \\ k \end{array} \right) (q_0)^k (1-q_0)^{d_1-k}. \end{aligned} \tag{20}$$

where q_0 is in (19). We note that (20) reduced to (11) when $K=1$.

Now, we compute the *false negative rate*, FNR, which is defined as the conditional probability that a sample is decoded to be negative, given that the sample is a positive sample. Consider a positive sample. Conditioning on the event that $\tilde{d}_1$ edges of these d_1 edges of this positive sample are diluted, the probability that this positive sample is decoded to be negative is

$$\sum_{k=K}^{\tilde{d}_1} \left(\begin{array}{c} \tilde{d}_1 \\ k \end{array} \right) (q_0)^k (1-q_0)^{\tilde{d}_1-k}, \tag{21}$$

as shown in (20). Thus,

$$\begin{aligned} FNR &= \sum_{\tilde{d}_1=K}^{d_1} \Big(\sum_{k=K}^{\tilde{d}_1} \left(\begin{array}{c} \tilde{d}_1 \\ k \end{array} \right) (q_0)^k (1-q_0)^{\tilde{d}_1-k} \Big) \\ & \qquad \left(\begin{array}{c} d_1 \\ \tilde{d}_1 \end{array} \right) (\epsilon)^{\tilde{d}_1} (1-\epsilon)^{d_1-\tilde{d}_1}. \end{aligned} \tag{22}$$

In particular, for $K=1$, we have

$$FNR = 1-(1-q_0\epsilon)^{d_1} \tag{23}$$

Now, we compute the *true positive rate*, TPR, or *sensitivity*, which is defined as the conditional probability that a sample is decoded to be positive, given that the sample is a positive sample.

$$TPR = 1 - FNR = (1-q_0\epsilon)^{d_1}. \tag{24}$$

The expected number of un-decoded samples after Step 1 of the K-COMP algorithm is $M(r_0p_0 + r_1 \cdot TPR)$. Thus, the expected relative cost for the K-COMP algorithm is

$$\frac{N+M(r_0p_0+r_1 \cdot TPR)}{M} = \frac{1}{G} + r_0p_0 + r_1 \cdot TPR. \tag{25}$$

6 Numerical Results

6.1 Noiseless decoding

In this section, we compare the performance of various pooling matrices by using the DD2 algorithm in Algorithm 2. The first four pooling matrices are constructed by using the PPoL algorithm in Algorithm 3 with the parameters $(m, d_1) = (31, 3)$, $(23, 4)$, $(13, 3)$, and $(7, 2)$, respectively. The fifth pooling matrix is the pooling matrix used in P-BEST [8]. The sixth matrix is the 15×35 pooling matrix constructed by the Kirkman triples. The next two pooling matrices are used in Tapestry [9], [10]. The last pooling matrix is the 2D-pooling matrix in [7]. In Table 3, we show the basic information of these pooling matrices. The size of an $N \times M$ pooling matrix indicates that the number of groups is N, and the number of samples is M. The parameter d_1 is the number of groups in which a sample is pooled. On the other hand, d_2 is the number of samples in a group. Note that there are some pooling matrices that are not (d_1, d_2)-regular. For instance, in the 2D-pooling matrix, there are 8 groups with 12 samples and 12 groups with 8 samples. Also, both the 16 × 40 matrix and the 24 × 60 matrix used in Tapestry are not (d_1, d_2)-regular. The column marked with *row cor.* (resp. *col. cor.*) is the maximum of the inner product of two rows (resp. columns) in a pooling matrix. For a pooling matrix, the column marked with *girth* is the minimum length of a cycle in the bipartite graph corresponding to that pooling matrix. The column marked with *(comp.) gain* is the compressing gain G of a pooling matrix, which is the ratio of the number of columns (samples) to the number of rows (groups), i.e., $G = M/N$. As shown in Table 3, both the row correlation and the column correlation of the pooling matrices constructed from the PPoL algorithm in Algorithm 3 are 1. So are the 15 × 35 pooling matrix constructed by the Kirkman triples. Such a correlation result is expected from Proposition 7. On the other hand, the row correlation and the column correlation of the pooling matrix in P-BEST [8] are 6 and 2, respectively. Also, the girth of the pooling matrix in P-BEST is only 4, which is smaller than the other four matrices. The girth of the 16 × 40 pooling matrix in Tapestry is also 4. This shows that the pooling matrices from the PPoL algorithm are more “spread-out” than the pooling matrix in P-BEST and the 16 × 40 pooling matrix in Tapestry.

Table 3: Basic information of some pooling matrices.

H	size	d_1	d_2	row cor.	col. cor.	girth	(comp.) gain
PPoL-(31,3)	93×961	3	31	1	1	6	10.33
PPoL-(23,4)	92×529	4	23	1	1	6	5.75
PPoL-(13,3)	39×169	3	13	1	1	6	4.33
PPoL-(7,2)	14×49	2	7	1	1	8	3.5
P-BEST Matrix [8]	48×384	6	48	6	2	4	8
Kirkman Matrix 15×35	15×35	3	7	1	1	6	2.33
Tapestry Matrix 16×40 [9]	16×40	2-3	6-9	3	2	4	2.5
Tapestry Matrix 24×60 [9]	24×60	2-3	6-7	1	1	6	2.5
2D-pooling Matrix [7]	20×96	2	12(8)	1	1	8	4.8

In practical situations, the prevalence rates of COVID-19 are basically in the range of 0% to 5%. As such, we conduct 10,000 independent experiments for each value of the prevalence rate r_1 in this range to compare the performance of pooling matrices in Table 3. Each numerical result is obtained by averaging over these 10,000 independent experiments. Thus, we believe the simulation results should be applicable to practical situations.

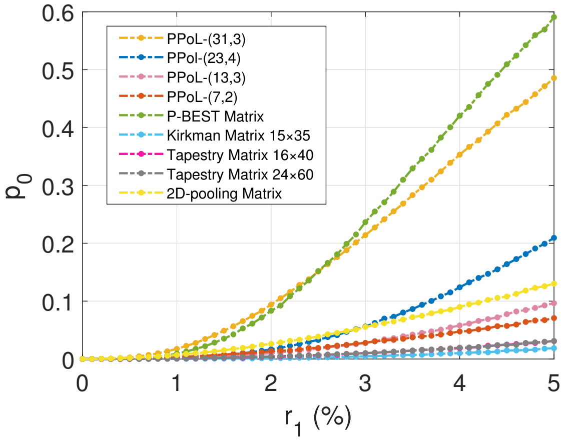


Figure 8: The conditional probability p_0 (that a sample cannot be decoded given it is a *negative* sample) as a function of the prevalence rate r_1 for various pooling matrices.

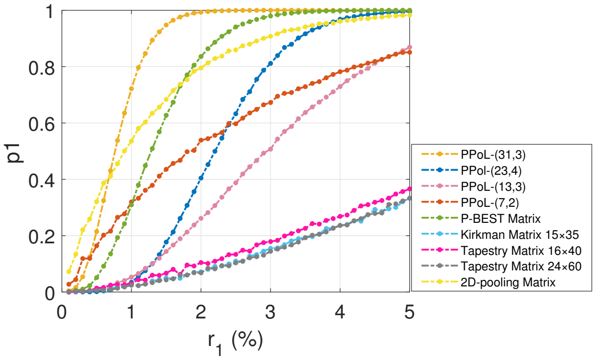


Figure 9: The conditional probability p_1 (that a sample cannot be decoded given it is a *positive* sample) as a function of the prevalence rate r_1 for various pooling matrices.

In Figure 8, we show the (measured) conditional probability p_0 (that a sample cannot be decoded given it is a *negative* sample) for these pooling matrices. For the PPoL pooling matrices, the measured p_0's match extremely well with the theoretical results from (11). As shown in this figure, the Kirkman matrix and the two matrices in Tapestry have the best performance. This is because their d_2's (the number of samples in a group) are small (below 9 for these three matrices). As such, the probability that a group is tested negative is higher than the other pooling matrices. Note that these three matrices also have low (compressing) gains, 2.33-2.5. On the other hand, P-BEST has the worst performance for p_0 as the number of samples in a group for that matrix is 48, which is the largest among all these pooling matrices.

In Figure 9, we show the (measured) conditional probability p_1 (that a sample cannot be decoded given it is a *positive* sample) for these pooling matrices. Once again, the Kirkman matrix and the two matrices in Tapestry have the best performance. This is mainly due to the low (compressing) gains of these three matrices. Though not shown in Figure 9, we note that the measured p_1's are very close to those from (13), and thus the tree expansion in Figure 7 is actually tree-like.

As discussed in Section 4.4, the probability that a sample cannot be decoded is $r_0 p_0 + r_1 p_1$. Such a probability is also the probability that a sample needs to go through the second stage for individual testing. In Figure 10, we show the probability $r_0 p_0 + r_1 p_1$ as a function of the prevalence rate r_1 for various pooling matrices. As shown in this figure, the Kirkman matrix and the two matrices in Tapestry have the best performance. Once

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3121709, IEEE Transactions on Network Science and Engineering

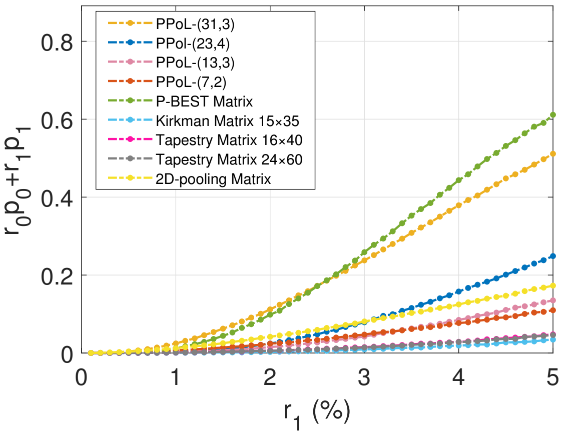


Figure 10: The probability $r_0p_0 + r_1p_1$ (that a sample cannot be decoded at the first stage and should be tested individually at the second stage) as a function of the prevalence rate r_1 for various pooling matrices.

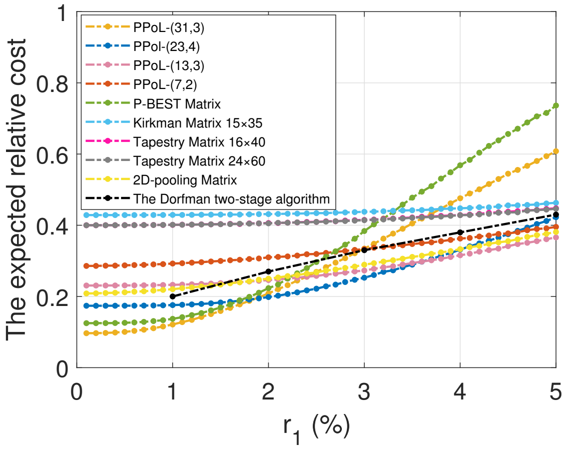


Figure 11: The expected relative cost as a function of the prevalence rate r_1 for various pooling matrices.

again, this is mainly due to the low (compressing) gains of these three matrices. We note that it takes time to do the second test. The numerical results in Figure 10 imply that using the Kirkman matrix (or the two matrices in Tapestry) has the shortest expected time to obtain a testing result.

A fair comparison of these pooling matrices is to measure their expected relative costs (defined in [6]). Recall that the expected relative cost is the ratio of the expected number of tests required by the group testing technique to the number of tests required by the individual testing. In Figure 11, we show the (measured) expected relative costs for these pooling matrices. In this figure, we also plot the curve for the Dorfman two-stage algorithm (the black curve) with the optimal group size M chosen from Table 1 of [6] for the prevalence rates, $1\%, 2\%, \ldots, 5\%$. To our surprise, the curves for the Kirkman matrix and the two matrices in Tapestry are above the black curve. This means that the expected relative costs of these three matrices are higher than the (optimized) Dorfman two-stage algorithm. Thus, if the additional amount of time to go through the second stage is not critical, using other pooling matrices could lead to more cost reduction than using these three matrices. There are several pooling matrices that have very low relative costs when the prevalence rates are below 1%. The P-BEST pooling matrix is one of them. However, the relative cost of the P-BEST pooling matrix increases dramatically when the prevalence rates are above 1.3%. Moreover, the P-BEST pooling matrix has a higher relative cost than the (optimized) Dorfman two-stage algorithm when the prevalence rate is above 2.5%. On the other hand, 2D-pooling has a very low relative cost when the prevalence rates are above 2.5%. To summarize, there does not exist a pooling matrix that has the lowest relative cost in the whole range of the prevalence rates considered in our experiments.

To optimize the performance, one should choose the right pooling matrix, depending on the prevalence rate. However, this might be difficult as the exact prevalence rate of a new outbreak of COVID-19 in a region might not be known in advance. Our suggestion is to use suboptimal PPoL matrices for a range of prevalence rates, as shown in Table 4. As shown in this table, the costs computed from the theoretical approximations in (14) and the costs measured from simulations are very close, and they are within 2% of the minimum costs for (d_1, d_2)-regular pooling matrices in Table 2. From our numerical results in Figure 11, we suggest using the PPoL matrix with $d_1 = 3$ and $d_2 = 31$ when the prevalence rate r_1 is below 2%. In this range of prevalence rates, its expected relative cost is even smaller than that of P-BEST. Moreover, it can achieve an 8-fold reduction in test costs when the prevalence rate is near 1% (as shown in Table 4), and most samples can be decoded in the first stage (as shown in Figure 10). When the prevalence rate r_1 is between 2%-4%, we suggest using the PPoL matrix with $d_1 = 4$ and $d_2 = 23$. In this range of prevalence rates, using such a pooling matrix can still achieve (at least) a 3-fold reduction in test costs. Roughly, 17% of samples need to go through the second stage when the prevalence rate is near 4% (as shown in Figure 10). When the prevalence rate r_1 is between 4%-7%, we suggest using the PPoL matrix with $d_1 = 3$ and $d_2 = 13$, and it can still achieve (at least) a 2-fold reduction in test costs. When the prevalence rate r_1 is between 7%-10%, we suggest using the PPoL matrix with $d_1 = 2$ and $d_2 = 7$. Though its expected relative cost is still lower than that of the Dorfman two-stage algorithm, the difference is small.

Table 4: Suboptimal PPoL pooling matrices. r_1: prevalence rates; d_1 and d_2: parameters of PPoL pooling matrices; cost (14): costs computed from the theoretical approximations in (14); cost (sim): costs measured from simulations; Dorfman [6]: costs by the Dorfman two-stage algorithm.

r_1	d_1	d_2	cost (14)	cost (sim)	Dorfman [6]
1%	3	31	0.1218	0.12	0.20
2%	4	23	0.1973	0.20	0.27
3%	4	23	0.2552	0.25	0.33
4%	3	13	0.3170	0.32	0.38
5%	3	13	0.3685	0.37	0.43
6%	3	13	0.4243	0.42	0.47
7%	2	7	0.4651	0.47	0.50
8%	2	7	0.5035	0.50	0.53
9%	2	7	0.5422	0.54	0.56
10%	2	7	0.5809	0.58	0.59

6.2 Noisy decoding

In this section, we compare the performance of various pooling matrices in the noisy case. The pooling matrices are the same

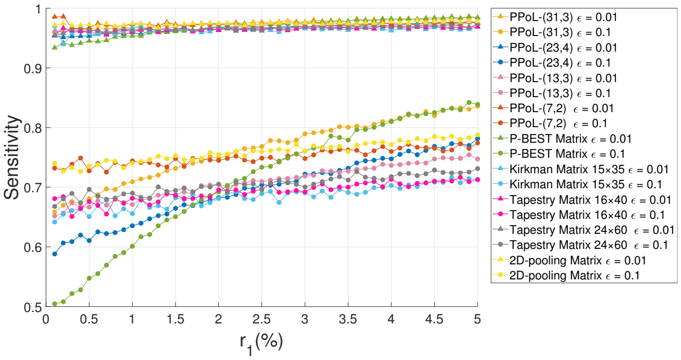


Figure 12: The sensitivity as a function of the prevalence rate r_1 for various pooling matrices under the dilution noise ϵ by using the 1-COMP decoding algorithm.

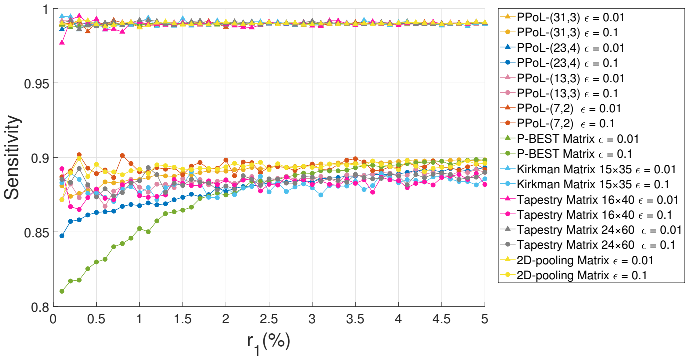


Figure 14: The sensitivity as a function of the prevalence rate r_1 for various pooling matrices under the dilution noise ϵ by using the 2-COMP decoding algorithm.

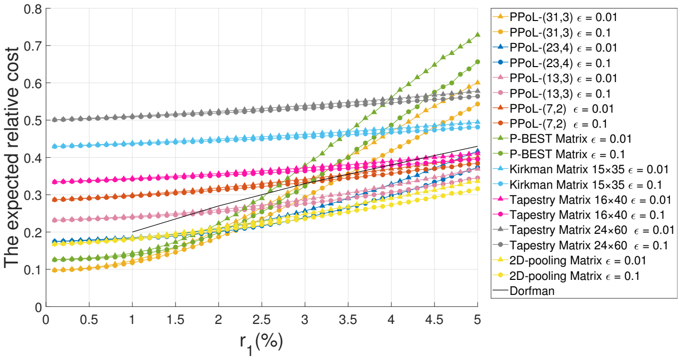


Figure 13: The expected relative cost as a function of the prevalence rate r_1 for various pooling matrices under the dilution noise ϵ by using the 1-COMP decoding algorithm.

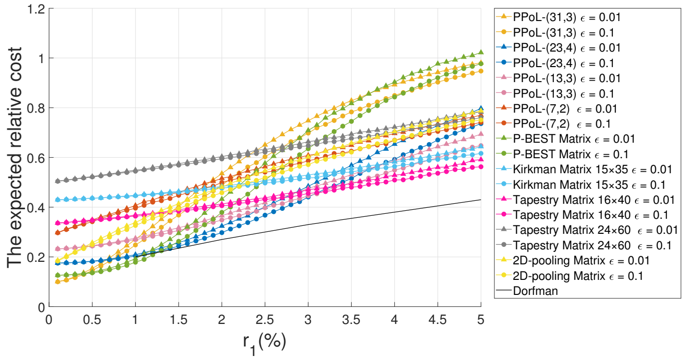


Figure 15: The expected relative cost as a function of the prevalence rate r_1 for various pooling matrices under the dilution noise ϵ by using the 2-COMP decoding algorithm.

as those in Section 6.1. We consider two dilution probabilities $\epsilon = 0.01$ and $\epsilon = 0.1$ in the dilution noise model. For the noisy decoding, we use the K-COMP algorithm with $K = 1$ and $K = 2$ in Algorithm 4.

To compare the performance of these pooling matrices in the noisy case, we conduct 10,000 independent experiments for each value of the prevalence rate r_1, ranging from 0% to 5%. Each numerical result is obtained by averaging over these 10,000 independent experiments. In Figure 12, we show the *sensitivity* for these pooling matrices using the 1-COMP decoding algorithm. For $\epsilon = 0.01$, we observe that the performances of all pooling matrices are comparable. For $\epsilon = 0.1$, the PPoL(7,2) matrix and the 2D-pooling matrix have the best performance, while the P-BEST pooling matrix has the worst result when the prevalence rate r_1 is less than 1.5%. The results can be explained from the value of d_1, the degree of each sample. Specifically, if one of the edges of a positive sample is diluted, then such a sample may be decoded as a negative one. Consequently, the larger d_1 results in the worse performance. In this figure, we also observe that the sensitivity increases in r_1. The reason is that by using the 1-COMP decoding algorithm, there are more un-decoded samples as r_1 increases. Such un-decoded samples are tested individually at the second stage. This contributes to more true positive samples.

In Figure 13, we show the expected relative costs for these pooling matrices using the 1-COMP decoding algorithm. When r_1 is below 1.5%, the value of ϵ has little effect on the expected relative costs for all pooling matrices. This is because the number of positive samples is small under low prevalence rates, and hence most of the samples can be decoded at the first stage. The same argument also explains that the higher (compressing) gain of the pooling matrix leads to a lower expected relative cost when r_1 is below 2%. Moreover, as r_1 increases, we observe that the expected relative costs of the P-BEST matrix and the PPoL(31,3) rise dramatically. The reason is that they have larger d_2's. Specifically, if a single group contains more samples, this group is more likely to be positive and thus cannot be decoded at the first stage.

In Figure 14 and Figure 15, we show the sensitivity and the expected relative costs, respectively, for the pooling matrices using the 2-COMP decoding algorithm. Compare with the results of $K = 1$, the sensitivity of $K = 2$ shows a considerable improvement when $\epsilon = 0.1$. The reason is that for $K = 2$, a sample can be decoded as negative only when this sample is pooled in at least 2 negative groups. This greatly enhances the sensitivity, but the expected relative costs increase because more samples need to be tested at the second stage.

In Figure 13 and Figure 15, we also plot the curve for the Dorfman two-stage algorithm (the black curve) with its optimal group size for the prevalence rates, $1\%, 2\%, \ldots, 5\%$. We can see that when $K = 1$, the PPoL(31,3), the P-BEST matrix, the 2D-pooling matrix, and the PPoL(23,4) have lower expected relative costs than that of the Dorfman two-stage algorithm because of their higher (compressing) gains. When $K = 2$, none of these matrices outperforms the Dorfman two-stage algorithm in terms of the expected relative costs.

To sum up, in the dilution noise model, the sensitivity of the 1-COMP decoding algorithm in Algorithm 4 decrease significantly with respect to the increase of the dilution noise. Though using the 2-COMP decoding algorithm in Algorithm 4 results in a considerable improvement, the expected relative costs may be higher than those by the Dorfman two-stage algorithm. Thus, the simple Dorfman method might be a better strategy for pooled testing in a noisy setting.

7 Conclusion

In this paper, we proposed a new family of PPoL polling matrices that have maximum column correlation and row correlation of 1 for a wide range of column weights. Using the two-stage definite defectives (DD2) decoding algorithm, we compare their performance with various pooling matrices proposed in the literature, including 2D-pooling [7], P-BEST [8], and Tapestry [9], [10]. Our numerical results showed no pooling matrix with the lowest expected relative cost in the whole range of the prevalence rates. To optimize the performance, one should choose the right pooling matrix, depending on the prevalence rate. As the family of PPoL matrices can dynamically adjust their construction parameters according to the prevalence rates, it seems that using such a family of pooling matrices might lead to better cost reduction than using a fixed pooling matrix. We also consider a noisy setting in this paper. Our numerical results show a trade-off between the high sensitivity and the low expected relative costs. As such, when the dilution noise is not negligible, the simple Dorfman method might be a better strategy for pooled testing.

In this paper, we only considered binary samples. For ternary samples, there are three test outcomes: negative (0), weakly positive (1), and strongly positive (2). It seems possible to extend the DD2 algorithm for binary samples to the setting with ternary samples by using successive cancellations.

References

[1] "Coronavirus disease (COVID-19) outbreak," Jan 2020. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019

[2] H. Nishiura, T. Kobayashi, T. Miyama, A. Suzuki, S.-m. Jung, K. Hayashi, R. Kinoshita, Y. Yang, B. Yuan, A. R. Akhmetzhanov *et al.*, "Estimation of the asymptomatic ratio of novel coronavirus infections (COVID-19)," *International Journal of Infectious Diseases*, vol. 94, p. 154, 2020.

[3] Y.-C. Chen, P.-E. Lu, C.-S. Chang, and T.-H. Liu, "A time-dependent SIR model for COVID-19 with undetectable infected persons," *IEEE Transactions on Network Science and Engineering, DOI: 10.1109/TNSE.2020.3024723*, 2020.

[4] "Pooled sample testing and screening testing for COVID-19," Aug 2020. [Online]. Available: https://www.fda.gov/medical-devices/coronavirus-covid-19-and-medical-devices/pooled-sample-testing-and-screening-testing-covid-19

[5] "Interim guidance for use of pooling procedures in SARS-CoV-2 diagnostic, screening, and surveillance testing," June 2020. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/lab/pooling-procedures.html

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

[7] N. Sinnott-Armstrong, D. Klein, and B. Hickey, "Evaluation of group testing for SARS-CoV-2 RNA," *medRxiv, DOI: 10.1101/2020.03.27.20043968*, 2020.

[8] N. Shental, S. Levy, V. Wuvshet, S. Skorniakov, B. Shalem, A. Ottolenghi, Y. Greenshpan, R. Steinberg, A. Edri, R. Gillis *et al.*, "Efficient high-throughput SARS-CoV-2 testing to detect asymptomatic carriers," *Science Advances*, p. eabc5961, 2020.

[9] S. Ghosh, A. Rajwade, S. Krishna, N. Gopalkrishnan, T. E. Schaus, A. Chakravarthy, S. Varahan, V. Appu, R. Ramakrishnan, S. Ch *et al.*, "Tapestry: A single-round smart pooling technique for COVID-19 testing," *medRxiv*, 2020.

[10] S. Ghosh, R. Agarwal, M. A. Rehan, S. Pathak, P. Agrawal, Y. Gupta, S. Consul, N. Gupta, R. Goyal, A. Rajwade *et al.*, "A compressed sensing approach to group-testing for COVID-19 detection," *arXiv preprint arXiv:2005.07895*, 2020.

[11] J. Yi, R. Mudumbai, and W. Xu, "Low-cost and high-throughput testing of COVID-19 viruses and antibodies via compressed sensing: System concepts and computational experiments," *arXiv preprint arXiv:2004.05759*, 2020.

[12] A. Heidarzadeh and K. R. Narayanan, "Two-stage adaptive pooling with RT-qPCR for COVID-19 screening," *arXiv preprint arXiv:2007.02695*, 2020.

[13] M. Aldridge, O. Johnson, and J. Scarlett, "Group testing: An information theory perspective," *Foundations and Trends in Communications and Information Theory*, vol. 15, no. 3-4, pp. 196–392, 2019.

[14] S. Lohse, T. Pfuhl, B. Berkó-Göttel, J. Rissland, T. Geißler, B. Gärtner, S. L. Becker, S. Schneitler, and S. Smola, "Pooling of samples for testing for SARS-CoV-2 in asymptomatic people," *The Lancet Infectious Diseases*, 2020.

[15] B. Abdalhamid, C. R. Bilder, E. L. McCutchen, S. H. Hinrichs, S. A. Koepsell, and P. C. Iwen, "Assessment of specimen pooling to conserve SARS CoV-2 testing resources," *American Journal of Clinical Pathology*, vol. 153, no. 6, pp. 715–718, 2020.

[16] I. Yelin, N. Aharony, E. Shaer-Tamar, A. Argoetti, E. Messer, D. Berenbaum, E. Shafran, A. Kuzli, N. Gandali, T. Hashimshony *et al.*, "Evaluation of COVID-19 RT-qPCR test in multi-sample pools," *Clinical Infectious Diseases*, vol. 71, no. 16, pp. 2073–2078, 2020.

[17] C. Gollier and O. Gossner, "Group testing against COVID-19," *Covid Economics*, vol. 2, 2020.

[18] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[19] Y.-J. Lin and C.-S. Chang, "PPoL: A periodic channel hopping sequence with nearly full rendezvous diversity," *arXiv preprint arXiv:2104.08461*, 2021.

[20] C.-S. Chang, W. Liao, and C.-M. Lien, "On the multichannel rendezvous problem: Fundamental limits, optimal hopping sequences, and bounded time-to-rendezvous," *Mathematics of Operations Research*, vol. 40, no. 1, pp. 1–23, 2015.

[21] C.-S. Chang, D.-S. Lee, and C. Wang, "Asynchronous grant-free uplink transmissions in multichannel wireless networks with heterogeneous qos guarantees," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1584–1597, 2019.

[22] C.-S. Chang, J.-P. Sheu, and Y.-J. Lin, "On the theoretical gap of channel hopping sequences with maximum rendezvous diversity in the multichannel rendezvous problem," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2021.

[23] J. Singer, "A theorem in finite projective geometry and some applications to number theory," *Transactions of the American Mathematical Society*, vol. 43, no. 3, pp. 377–385, 1938.

[24] D. Hong, R. Dey, X. Lin, B. Cleary, and E. Dobriban, "HYPER: Group testing via hypergraph factorization applied to COVID-19," *medRxiv*, 2021.

[25] M. Täufer, "Rapid, large-scale, and effective detection of COVID-19 via non-adaptive testing," *Journal of Theoretical Biology*, vol. 506, 2020.

[26] L. Euler, "Recherches sur une nouvelle espece de quarres magiques," *Verhandelingen uitgegeven door het zeeuwsch Genootschap der Wetenschappen te Vlissingen*, vol. 9, pp. 85–239, 1782.

[27] W. Kautz and R. Singleton, "Nonrandom binary superimposed codes," *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 363–377, 1964.

Yi-Jheng Lin received the B.S. degree in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2018. He is currently working toward the Ph.D. degree with the Institute of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan. His research interests include wireless communication and cognitive radio networks.

Che-Hao Yu received the B.S. degree in mathematics and the M.S. degree in communications engineering from National Tsing-Hua University, Hsinchu, Taiwan, in 2018 and 2020, respectively. Since 2020, he has been with Phison Electronics Corp., Taiwan. His research focuses on 5G wireless communication.

Tzu-Hsuan Liu received the B.S. degree in communication engineering from National Central University, Taoyuan, Taiwan, in 2018 and the M.S. degree from the Institute of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan, in 2020. Since January 2021, she has been with MediaTek Inc., Hsinchu, Taiwan. Her research focuses on 5G wireless communication.

Cheng-Shang Chang (S'85-M'86-M'89-SM'93-F'04) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1983, and the M.S. and Ph.D. degrees from Columbia University, New York, NY, USA, in 1986 and 1989, respectively, all in electrical engineering.

From 1989 to 1993, he was employed as a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. Since 1993, he has been with the Department of Electrical Engineering, National Tsing Hua University, Taiwan, where he is a Tsing Hua Distinguished Chair Professor. He is the author of the book Performance Guarantees in Communication Networks (Springer, 2000) and the coauthor of the book Principles, Architectures and Mathematical Theory of High Performance Packet Switches (Ministry of Education, R.O.C., 2006). His current research interests are concerned with network science, big data analytics, mathematical modeling of the Internet, and high-speed switching.

Dr. Chang served as an Editor for Operations Research from 1992 to 1999, an Editor for the *IEEE/ACM TRANSACTIONS ON NETWORKING* from 2007 to 2009, and an Editor for the *IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING* from 2014 to 2017. He is currently serving as an Editor-at-Large for the *IEEE/ACM TRANSACTIONS ON NETWORKING*. He is a member of IFIP Working Group 7.3. He received an IBM Outstanding Innovation Award in 1992, an IBM Faculty Partnership Award in 2001, and Outstanding Research Awards from the National Science Council, Taiwan, in 1998, 2000, and 2002, respectively. He also received Outstanding Teaching Awards from both the College of EECS and the university itself in 2003. He was appointed as the first Y. Z. Hsu Scientific Chair Professor in 2002. He received the Merit NSC Research Fellow Award from the National Science Council, R.O.C. in 2011. He also received the Academic Award in 2011 and the National Chair Professorship in 2017 from the Ministry of Education, R.O.C. He is the recipient of the 2017 IEEE INFOCOM Achievement Award.

Wen-Tsuen Chen (M'87-SM'90-F'94) received his B.S. degree in nuclear engineering from National Tsing Hua University, Taiwan, and M.S. and Ph.D. degrees in electrical engineering and computer sciences both from University of California, Berkeley, in 1970, 1973, and 1976, respectively. He has been with the Department of Computer Science of National Tsing Hua University since 1976 and served as Chairman of the Department, Dean of College of Electrical Engineering and Computer Science, and the President of National Tsing Hua University. In March 2012, he joined the Academia Sinica, Taiwan as a Distinguished Research Fellow of the Institute of Information Science until June 2018. Currently he is Sun Yun-suan Chair Professor of National Tsing Hua University. His research interests include computer networks, wireless sensor networks, mobile computing, and parallel computing. Dr. Chen received numerous awards for his academic accomplishments in computer networking and parallel processing, including Outstanding Research Award of the National Science Council, Academic Award in Engineering from the Ministry of Education, Technical Achievement Award and Taylor L. Booth Education Award of the IEEE Computer Society, and is currently a lifelong National Chair of the Ministry of Education, Taiwan. Dr. Chen is the Founding General Chair of the IEEE International Conference on Parallel and Distributed Systems and the General Chair of the IEEE International Conference on Distributed Computing Systems. He is an IEEE Fellow and a Fellow of the Chinese Technology Management Association.