BMC Bioinformatics

RESEARCH

Open Access

Effective matrix designs for COVID-19 group testing

David Brust[1] and Johannes J. Brust[2*]

*Correspondence:
jjbrust@ucsd.edu

[1] Institute of Future Fuels, German Aerospace Center (DLR), Jülich, Germany
[2] Department of Mathematics, University of California, San Diego, San Diego, USA

Abstract

Background: Grouping samples with low prevalence of positives into pools and testing these pools can achieve considerable savings in testing resources compared with individual testing in the context of COVID-19. We review published pooling matrices, which encode the assignment of samples into pools and describe decoding algorithms, which decode individual samples from pools. Based on the findings we propose new one-round pooling designs with high compression that can efficiently be decoded by combinatorial algorithms. This expands the admissible parameter space for the construction of pooling matrices compared to current methods.

Results: By arranging samples in a grid and using polynomials to construct pools, we develop direct formulas for an Algorithm (Polynomial Pools (PP)) to generate assignments of samples into pools. Designs from PP guarantee to correctly decode all samples with up to a specified number of positive samples. PP includes recent combinatorial methods for COVID-19, and enables new constructions that can result in more effective designs.

Conclusion: For low prevalences of COVID-19, group tests can save resources when compared to individual testing. Constructions from the recent literature on combinatorial methods have gaps with respect to the designs that are available. We develop a method (PP), which generalizes previous constructions and enables new designs that can be advantageous in various situations.

Keywords: Group testing, COVID-19, Combinatorial design, Affine plane

Background

The development of vaccines for COVID-19 has constituted a major breakthrough for the return to pre-pandemic life. However, while vaccines are becoming globally widespread, levels of caution prevail. Even with vaccines, monitoring for the evolution of mutations or detecting new outbreaks calls for continued vigilance. Therefore, *testing* is likely to prevail as a vital mechanism to inform decision making in the near future. In order to conserve scarce testing resources, the Centers for Disease Control and Prevention (CDC) in the United States have endorsed so-called group/pooling test methods (likewise, group testing is also being deployed in Central Europe, China, India, Israel and more nations). Such methods date to Dorfman's early work in 1943 [1], and can be

expressed using linear systems. The basic principle underlying pooling tests is the observation that to efficiently detect positive cases among a population with a relatively low occurrence prevalence, it can be advantageous to test groups of samples instead of testing all individual samples.

One-round methods

For a set of N samples $\mathcal{S} = \{v_0, v_1, v_2, \ldots, v_{N-1}\}$, a basic pooling test pools (mixes) all samples into one group, and subsequently uses one test on this group. If the pool is negative then it can be concluded that each individual sample must be negative. If the pool tests positive at least one sample is positive, and all individuals are retested. This strategy is an example of a 2-round pooling test (in general multi-stage). Based on information from previous testing rounds multi-stage tests update the samples to be tested in the next round. Such methods enable desirable reductions in the number of total needed tests, however they come at the expense of longer times to receive results (since each round adds approximately the same amount of time). An approach that overcomes potentially long wait times is a one-round method [2].

The simultaneous identification of the samples can be represented by a linear system. Since being positive (having the disease) or being negative (not having the disease) are binary states, binary digits i.e., $v \in \{0, 1\}$ will be used throughout. Additions and multiplications with v are defined by bit-wise OR $(+)$ and AND $(*)$ operations. Most of the arithmetic carries over expectedly, with the exception that $1 = 1 + 1$ in binary form. Suppose now that for certain integers $q \geq 2$ and $d \geq 2$ the samples $\mathcal{S} = \{v_0, v_1, v_2, \ldots, v_{N-1}\}$ with $N = q^d$ have been collected. We would like to infer the states of the elements of vector $\mathbf{v} \in \{0, 1\}^N$ based on $M < N$ observed test results, encoded in vector $\mathbf{w} \in \{0, 1\}^M$. Which elements of $\mathbf{v}$ are included in which test of $\mathbf{w}$ is determined by the binary matrix $\mathbf{M} \in \{0, 1\}^{M \times N}$. Specifically, if test i contains sample c then $\mathbf{M}_{i,c} = 1$. The Polymerase Chain Reaction (PCR) technology is used to generate the test outcomes in $\mathbf{w}$ (by amplifying the detectable viral load in each pool). All essential information about the pooling tests is therefore encoded in the linear system

$$\mathbf{M}\mathbf{v} = \mathbf{w} \tag{1}$$

In such schemes pools of size $m < N$ are formed and simultaneously tested. The distinct composition of pools with respect to their individual samples is essential for one-round methods to exactly identify the true positive samples.

Pooling matrix M

The distribution of nonzeros in the pooling matrix is equivalent to the design of pooling tests. Importantly, meaningful constructions of $\mathbf{M}$ enable one to exactly identify up to a certain number of positive samples using only the matrix and the observed values in $\mathbf{w}$. Two properties that ensure that up to $k \geq 1$ positive samples are guaranteed to be identified are:

P.1. Each pair of different samples (v_i, v_j) occurs at most $(d - 1)$ times (i.e., any column pair has at most $(d - 1)$ common nonzeros),
P.2. Each sample is contained in $k(d - 1) + 1$ tests.

To see the role of these properties we describe the main processes for decoding the observed outcomes to the desired sample values (cf. [3, Corollary 1], too).

Decoding

Suppose that there are up to k positives among all samples. Select and consider an arbitrary sample. If any of the first $k(d-1)$ tests with this sample are negative the sample must be negative. Otherwise there are two possible cases. In case one, the sample was misfortunately always paired with one of the k positives, even though it is negative. The second case is that the sample is indeed positive. Hence, based on property P.2., if the next test with this sample (i.e., test $k(d-1)+1$) is negative the sample must be negative. If the test is positive the sample must be positive. By P.2., each sample is contained in $k(d-1)+1$ tests and therefore all samples can be identified. Even though not absolutely necessary for the decoding of samples, an additional desirable property may be to have uniform pool sizes, since equipment and PCR setups typically have uniform layouts. An effective and fast algorithm to decode the observed $\mathbf{w}$ into $\mathbf{v}$ is Combinatorial Orthogonal Matching Pursuit (COMP) [4]. This algorithm declares $v_j = 1$ if all tests in which v_j is contained are positive and correctly detects up to k positives when $\mathbf{M}$ has properties P.1. and P.2. A straightforward representation of the COMP decoding algorithm is, with "$.*$" meaning element-wise multiplication:

$$v_j = 1 \quad \text{if} \quad \mathbf{M}_{:,j} \,.* \, \mathbf{w}_{:} = \mathbf{M}_{:,j} \quad 1 \le j \le N.$$

Illustration

In an example scheme with $d = 2, q = m = 4$, $N = q^d = 16$ and $M = 8$ the linear system for doing M simultaneous tests (where rows are labeled [ℓ_1-ℓ_8] and columns [0-15]) is given by:

$$\begin{array}{c} \\ \ell_1 \\ \ell_2 \\ \ell_3 \\ \ell_4 \\ \ell_5 \\ \ell_6 \\ \ell_7 \\ \ell_8 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \end{array} \\ \left[\begin{array}{cccccccccccccccc} 1 & 1 & 1 & 1 & & & & & & & & & & & & \\ & & & & 1 & 1 & 1 & 1 & & & & & & & & \\ & & & & & & & & 1 & 1 & 1 & 1 & & & & \\ & & & & & & & & & & & & 1 & 1 & 1 & 1 \\ 1 & & & & & 1 & & & & & 1 & & & & & 1 \\ & 1 & & & 1 & & & & & & & 1 & & & 1 & \\ & & 1 & & & & & 1 & 1 & & & & & 1 & & \\ & & & 1 & & & 1 & & & 1 & & & 1 & & & \end{array}\right] \end{array} = \mathbf{M}$$

Here, the first row of $\mathbf{M}$ corresponds to pooling samples (v_0, v_1, v_2, v_3) in test 1 (labeled ℓ_1). Observe that $d - 1 = 1$ and that the matrix satisfies property P.1., since each pair of columns has at most one common nonzero. Moreover, note that each sample occurs in $k(d-1)+1 = 2$ tests since each column has 2 nonzeros (also here $k = 1$). Based on P.1.

and P.2., $\mathbf{M}$ can therefore be used to exactly identify $\mathbf{v}$ in (1) when $\mathbf{v}$ contains up to one nonzero element. This design consequently uses 8 tests to identify up to one positive among 16 samples. The "Gain" (G) in using $\mathbf{M}$, when compared to testing all samples individually, is the ratio: $G = N/(\text{Number of tests}) = 16/8 = 2$.

For the special case that $d = 2$ and when each test has the same number of m samples, $\mathbf{M}$ is called a *multipool* matrix, and $k + 1$ represents the multiplicity of the matrix [5]. Moreover, multipool matrices are a specific *block design* from combinatorial designs [6, Part II]. Concretely, a balanced incomplete block design (BIBD) is a pair $(\mathcal{S}, \mathcal{B})$ where $\mathcal{S}$ is a set of N elements and $\mathcal{B}$ is a collection of b m-subsets of $\mathcal{S}$ (called blocks) such that each element of $\mathcal{S}$ is contained in exactly r blocks and any 2-subset of $\mathcal{S}$ is contained in exactly λ blocks. Therefore, a mulitpool matrix corresponds to a BIBD with $\lambda = 1$ and $r = k + 1$.

Recent pooling methods applied to COVID-19

In the following, we review published pooling methods used in infectious disease detection and high-throughput screening. Emphasis is placed on the state of the art literature on pooling methods applied in the context of COVID-19 testing.

The "square array" and related "grid" methods are commonly used for the generation of pooling designs and their evaluation in the presence of testing error [7–9]. These designs are characterized by arranging the samples in the form of a grid that can be square. Samples are then grouped together by pooling along rows, columns or sloped lines of the grid.

An early study in the context of COVID-19 disease detection proposes a non-adaptive, one round grid design termed "multipool" [5]. The notation $(\overline{N},\overline{n},\overline{k})$-multipool is introduced to help distinguish the notation. $\overline{N}$ is the number of samples in the pooling scheme, $\overline{n}$ is the order and $\overline{k}$ is the "multiplicity", the number of pools each sample appears in that bounds the maximum number of positives the method guarantees to detect with the COMP algorithm [5]. The multipool design yields pools of size $\overline{n}$, a prime number, capable to test up to $\overline{n}^2$ samples in one round of testing. Summarizing, the samples are arranged in an $\overline{n} \times \overline{n}$ square grid. Pools are formed by pooling along rows, columns or sloped lines of the grid. Thus all samples are covered once by a parallel class [6, I.1.6] of $\overline{n}$ pools corresponding to one pooling direction. The resulting "multipool" consists of the collection of $k \leq m + 1$ parallel classes, capable of detecting up to $k - 1$ positive samples.

For certain situations, an important parameter in the construction of pooling designs is the pool size m which determines the number of tests needed in order to detect the expected amount of positives and hence the pooling efficiency [10, 11]. The authors of [10] first describe the dependency of pooling efficiency on the pool size and prevalence of positive samples in the context of two-stage pooling tests. The argument is based on the calculation of the expected value of analyses (tests) per sample as a function of pool size required to identify the anticipated number of positive samples. Derivation of this expected value with respect to the pool size and target prevalence yields a formula for the optimal pool size in their model. Tables with optimum pool sizes for a wide range of expected prevalences are provided for referencing. A pooling design method that is flexible and whose pool size can be tailored to match the optimum at given prevalence is expected to yield higher pooling efficiency.

Pooling designs whose set of pools can be partitioned into subsets which each partition all samples are called resolvable [6, I.1.6]. The so called "Shifted Transversal

Design" (STD) [3] is a non-adaptive, combinatorial pool testing method designed for the application in binary high throughput screening and is an example of resolvable designs. A variation of STD, adapted and applied to high-throughput drug screening is presented in [12]. The STD pooling scheme is centered around the parameter q, which must be prime. STD yields a set of pools which are grouped into $q+1$ "layers", each of which in turn partitions all samples thus making it resolvable. The first q layers share a regular structure and are therefore called homogeneous. The last remaining layer has a specific construction that differs from the homogeneous layers and is hence called singular. A central characteristic of STD is that pairs of samples should occur together in the same pool only a limited number of times (co-occurrence of samples). In [3] the co-occurrence is denoted by Γ and the relationship for the maximum number of t positives that are guaranteed to be discovered by a pooling scheme for a fixed value of q is given by $\Gamma t + 1 \leq k$ where k is the number of layers. Since STD based on q generates up to $q+1$ layers it holds that $\Gamma t \leq q$.

A recent study [13] proposes a new algorithm "packing the pencil of lines" (PPoL) for the construction of pooling tests in the context of COVID-19 infection detection. Its conception relies on bipartite graphs and properties of finite projective planes. The construction starts with a bipartite graph whose nodes are connected according to the incidence structure of points and lines prescribed by a finite projective plane of prime power order m. The initial graph consists of m^2+m+1 sample and m^2+m+1 pool nodes, which correspond to the points and lines of the finite projective plane respectively. Specifically, there is an edge between a sample node and a pool node in the graph, if the corresponding points and lines in the finite projective plane are incident. Trimming the graph by removing nodes and associated edges according to the PPoL algorithm results in a pooling scheme for testing m^2 samples with characteristic parameters (d_1, d_2). It can be represented as a $d_1 m \times m^2$ binary pooling matrix where the parameters d_1 and d_2 are the number of non-zero elements in each column and row of the pooling matrix, respectively. d_1 corresponds to the number of pools that each sample occurs in and $d_2 = m$ is the pool size. The compression gain of this pooling matrix is $G = \frac{m^2}{md_1} = \frac{d_2}{d_1}$ and the number of positives that can be detected is $d_1 - 1$.

In the same study, the authors also compare the performance of pooling designs generated with PPoL with published approaches from the literature in a probabilistic analysis. Samples are assumed as independent and identically distributed Bernoulli random variables, with a probability of being positive r_1 (prevalence) and negative r_0, respectively. Simulated pooling test results are decoded according to the different pooling matrices using the "two-stage definite defective decoding" (DD2) algorithm [14]. Performance is measured by "expected relative cost" (ERC) which is the expected value of required tests for the detection of positive samples with the DD2 algorithm, divided by the total number of samples in the pooling scheme. In terms of DD2, the ERC is expressed as $\frac{1}{G} + r_0 p_0 + r_1 p_1$, where p_0 and p_1 are the conditional probabilites that a sample cannot be decoded given it is negative and positive, respectively.

PPoL pooling matrices share a common structure and properties with the newly proposed pooling methods from this article, and the performance assessment described above should carry over to the proposed methods in this work.

PPoL [13], P-BEST [15], Tapestry [16, 17] and 2D pooling matrices [18] are compared, where 2D pooling [18] is a particularly simple example of grid designs, which considers pooling along the 8 rows and 12 columns of a standard 96-well plate to form 20 pools in total. The pooling designs in [15–17] are decoded by compressed sensing algorithms.

It was revealed that pooling schemes that offered the highest performance at low prevalence rates (P-BEST and PPoL-(31,3)) yielded lower performance for increasing prevalence rates. The opposite was also observed where pooling schemes with comparatively lower performance (Tapestry, 2D pooling) did not suffer from higher expected relative cost at higher rates of prevalence. Overall it was revealed that no single design can achieve the highest compression gain for all considered values of prevalence when evaluated with the DD2 algorithm. Instead, the best pooling scheme should be selected depending on the expected prevalence rate of positives.

Summarizing, most of the reviewed pooling designs rely on selecting suitable parameters for the construction. From our observation, limitations in admissible construction parameters for published pooling designs are common. Methods have been proposed, that center around the order parameter q and that can test $N = q^d$ samples with either $d = 2$ or $d \geq 2$. Often the order q is limited to prime numbers. In cases were q was permitted to assume values of prime powers, the constructions were restricted to $d = 2$. This limits the total amount of samples $N = q^2$ that can be tested in one round and thus the maximum achievable compression gain. Published methods often rely on affine planes in the construction of pooling designs restricting the value of the pool size parameter which for affine planes is the same as the order $m = q$. As has been demonstrated, the optimal pool size for use in pooling designs depends on the prevalence of positive samples. Designs relying only on fixed values of m therefore cannot be adapted to best match the expected prevalence rate.

Contributions

Addressing the observed limitations in the published methods, we propose novel constructions for one-round combinatorial pooling tests that expand the state of the art methods in three ways. First, the proposed method permits prime power values for the order of the design $q = p^n$, which makes the method more flexible with regard to input parameters. Second, it includes pooling matrices based on projective geometries, where the pool size assumes values of $m = q + 1$. Because optimal pool size depends on the prevalence of positives [10, 11], projective geometries extend the range of available pool sizes and facilitate matching the design to the expected prevalence ratio. Third, it introduces pooling designs based on prime power order $q = p^n$ with dimension $d \geq 2$ which expands the available parameter space for methods delivering high compression gains for large N compared to current methods.

Results

We organize our results into two parts. In the first part of the results we describe our new algorithm for generating pooling tests. In the second part of the results, we compare our algorithm to existing state-of-the-art pooling methods (Tables 2, 3, 4, 5).

Results 1

We propose a novel algorithm called PP (Polynomial Pools), which is based on arranging samples in a grid and generating pools using polynomials. The degree of the polynomial is $d-1$ and a sample is represented with a pair of coordinates $v=(x,y)$. When $d=2$ our method is analogous to the incidence matrix of an affine plane [6, VII.2.2], [19] and the polynomials reduce to lines. For $p=q$ a prime number the computations can be done using modulus arithmetic. Specifically, a finite field of order p is defined by $\mathbb{F}_p=\{0,1,2,\ldots\}\mathrm{mod}\, p$ and has the important property that arithmetic calculations remain within the field. We label pools by ℓ and group samples through the linear relation

$$y \;=\; a\,x+b, \quad a,b,x\in\mathbb{F}_p. \tag{2}$$

The total number of samples in the affine plane (i.e., $d=2$) is the square $N=p^2$. For fixed a, b in $\mathbb{F}_q$ and the interval of integers $0\le i\le p-1$ we compute pools using the formulas

$$\begin{aligned} x_i &= i \\ y_i &= (ax_i+b)\mathrm{mod}\, p \\ v_{px_i+y_i} &= (x_i,y_i) \end{aligned} \tag{3}$$

A line/pool containing p samples is then the set $\{v_{px_i+y_i}\}_{i=0}^{p-1}$. We generalize this method further in three ways. First, for $d=2$, we include the option of constructing a projective plane [20, VII.2.1]. The projective plane is based on $N=p^2+p+1$ samples with each pool containing $p+1$ samples. This makes extended designs possible that use pool sizes, which are not primes. Second, we exploit the fact that affine and projective planes can be generated from prime powers $q=p^n$ with $n\ge 1$. The finite field of order q (also known by Galois field) then defines computations. Note that when $p=q$ (i.e., $n=1$) and the order of the finite field is prime, then modulus arithmetic similar like in (3) is applicable. However, for prime powers $q=p^n (n>1)$ the computations in the Galois field use sophisticated techniques. Specifically, the integer elements in the finite field are mapped to polynomials with which arithmetic operations are performed (e.g., addition and multiplication) before being mapped back to integers. Moreover, we include a dimension parameter $d\ge 2$ which enables designs with high compression gains by generating matrices that grow fast with regard to the total number of samples $N=(p^n)^d$.

Algorithm

To generalize (3), we denote the finite field by $\mathbb{F}_q$ with order $q=p^n$, $n\ge 1$. Defining indices $1\le l\le d-1$, $0\le x_{i_l}\le q-1$ we compute the coordinates of a sample $v=(x,y)$ using polynomials

$$y \;=\; a^{d-1}\,x_{i_{d-1}}+\cdots+a\,x_{i_1}+b, \quad a,b,x_{i_l}\in\mathbb{F}_q. \tag{4}$$

Note that when $d=2$ and $p=q$ this reduces to the linear equations from (2). We represent the combination of looping through the x_{i_l} values by a set with q^{d-1} elements

Table 1 Computing a pool ℓ using formula (5) for $d = 3, q = 3, a = 1, b = 0$

i_1	i_2	x_i	y_{x_i}	$qx_i + y_{x_i}$	ℓ
0	0	0	0	0	v_0
0	1	1	1	4	v_4
0	2	2	2	8	v_8
1	0	3	1	10	v_{10}
1	1	4	2	14	v_{14}
1	2	5	0	15	v_{15}
2	0	6	2	20	v_{20}
2	1	7	0	21	v_{21}
2	2	8	1	25	v_{25}

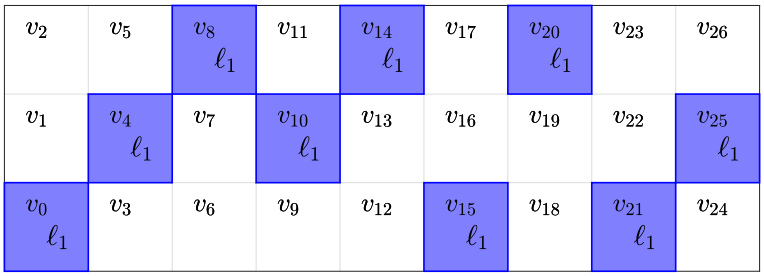

Fig. 1 For given parameters, samples are pooled using the polynomial relations from (5). Computation of the indices for samples from the pool is shown in Table 1

of a sequence of $d-1$ integers $x_{i_1} \times \cdots \times x_{i_{d-1}} = \{(i_1, \ldots, i_{d-1})\}$, where $0 \leq i_l \leq q-1$. Without loss of generality, the combination is such that i_{d-1} cycles every q times, i_{d-2} cycles every q^2 times until i_1 cycles only once. Formulas that compute the sample indices, and thus the corresponding pools, for fixed a and b, are given by

$$\begin{aligned} x_i &= \sum_{l=1}^{d-1} q^{d-1-l}\, i_l \\ y_{x_i} &= \sum_{l=1}^{d-1} a^l\, i_l + b \quad \text{(computed in } \mathbb{F}_q\text{)} \\ v_{qx_i+y_{x_i}} &= (x_i, y_{x_i}) \end{aligned} \tag{5}$$

Similar like in (3) computing y_{x_i} in (5), when $p = q$ (i.e., prime), can be done using the modulus $y_{x_i} = (\sum_{l=1}^{d-1} a^l\, i_l + b) \bmod q$. Otherwise the additions and multiplications in the expression for y_{x_i} are defined by the operations in the finite field. To illustrate the use of (5) we generate one pool ℓ when the slope and intercept are fixed at $a = 1$, $b = 0$ and the parameters are $d = 3$ and $q = p = 3$. Table 1 describes the calculations which correspond to the set of samples highlighted in Fig. 1.

An algorithm that computes sample pools according to polynomial pooling for a desired set of input parameters is given in Algorithm 1. The inputs include $q = p^n, d$ and k. The meaning of the parameters is: *order* (q) a prime power $q = p^n, n \geq 1, p$ (prime), which includes and extends prime number constructions; *dimension* (d) enables

scaling to large sample sizes $N = q^d$; *characteristic* (k) is the limit of positive samples up to which the design exactly decodes all samples. An additional "if else" statement is included, which can be used to generate q extra pools when $a = q$, i.e., the polynomials have slope of infinity in $\mathbb{F}_q$. The pools in this situation are formed by grouping q^{d-1} consecutive samples together. Note that this part of the algorithm is only invoked when the maximum number of positives for the design is requested $k = \lfloor \frac{q}{d-1} \rfloor$. In this case we compute the sample index and define the corresponding point by a different formula.

Algorithm 1: (`PP (PolynomialPools)`)

Inputs: $q = p^n, \quad 2 \leq d, \quad 1 \leq k \leq \left\lfloor \frac{q}{d-1} \right\rfloor$

for $0 \leq a \leq (k-1)(d-1), \quad 0 \leq b \leq q-1$

$0 \leq i_1 \leq q-1, \ \cdots, \quad 0 \leq i_{d-1} \leq q-1$

if $a < q$

$$x_i = \sum_{l=1}^{d-1} q^{d-1-l} \, i_l$$

$$y_{x_i} = \Big(\sum_{l=1}^{d-1} a^l \, i_l \Big) + b \qquad \texttt{\% Sum in } \mathbb{F}_q$$

$$v_{qx_i + y_{x_i}} = (x_i, y_{x_i})$$

else `% Slope at Infinity`

$$x_i = \sum_{l=1}^{d-1} q^{d-1-l} \, i_l$$

$$y_i^{\infty} = i_{d-1}, \quad x_i^{\infty} = \sum_{l=2}^{d-1} q^{d-1-l} \, i_l + q^{d-2} b$$

$$v_{q^{d-1} b + x_i} = (x_i^{\infty}, y_i^{\infty})$$

end if

Algorithm 1 generates designs that satisfy properties P.1. and P.2. of a pooling matrix, which guarantee to correctly decode all samples up to k positives. A design based on the algorithm is denoted PP(q, d, k). Lemmas 1 and 2 relate Algorithm 1 to the relevant properties P.1. and P.2. We prove the Lemmas in the Section "Methods".

Lemma 1 Any pair of samples v_i, v_j, for $i \neq j$ from a design of Algorithm 1 occurs at most $d-1$ times.

Lemma 2 Each sample v_i is contained in up to $k(d-1)+1$ tests.

By satisfying properties P.1. and P.2. the designs from Algorithm 1 guarantee to identify all samples with up to $1 \leq k$ positives. The size of a pool is the number of elements generated by one polynomial pool $\ell = \{v_{qx_i+y_{x_i}}\}$, which is $q^{d-1} =$ "# elements(ℓ)" (i.e., the number of combinations of $d-1$ integers $0 \leq i_l \leq q-1$ with $1 \leq l \leq d-1$). One can also see that designs from PP are resolvable. Figure 2 depicts the $q-1$ pools that are generated by (5) for a fixed slope a, when the intercept ranges through $0 \leq b \leq q-1$. Note that in Fig. 2 all N samples are pooled into exactly one of the pools, thus forming a resolution. This property comes from the observation that for a fixed slope, changing the intercept results in "parallel" polynomials and leads to "parallel" pools. Furthermore, by varying the parameters q and d many constructions are feasible, which include the possibility of designs with pool sizes that are for instance not prime.

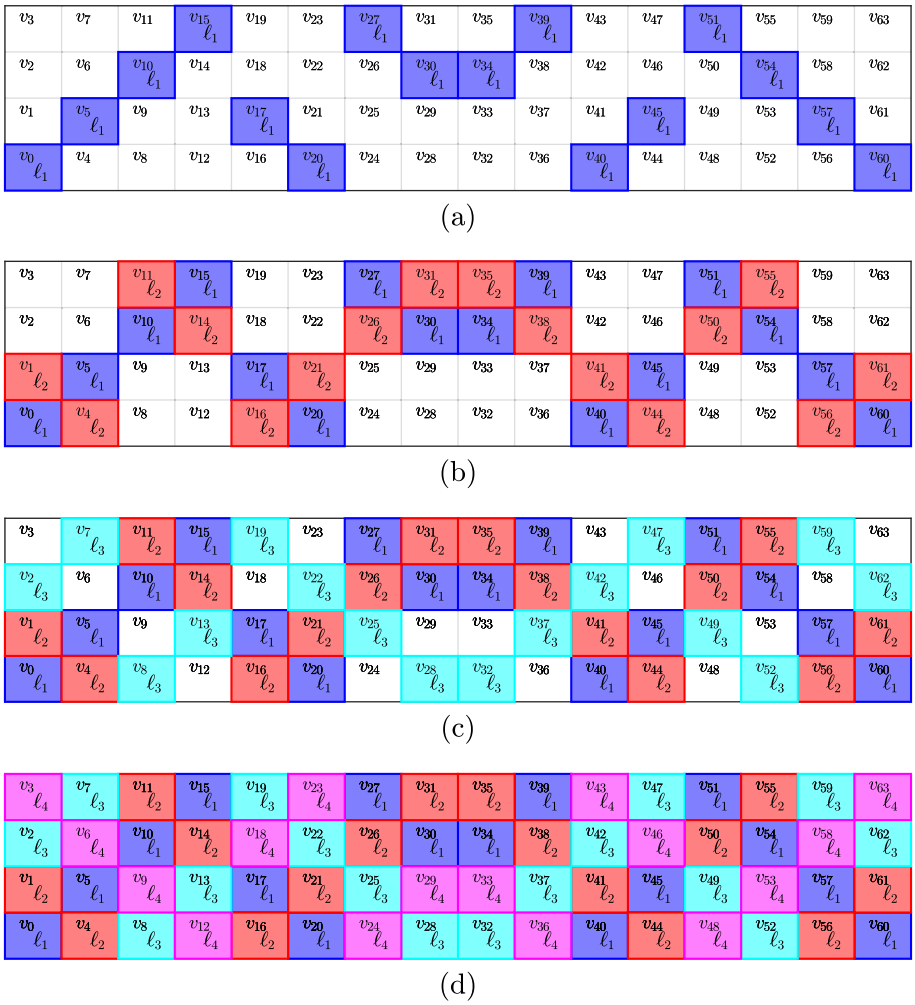

Fig. 2 Generating four pools using PP with order $q = 2^2 = 4$ and slope $a = 1$when the intercept b varies between $\{0, 1, 2, 3 = q-1\}$ in panels **a**–**d**

Projective geometry

Furthermore, for a construction from a projective geometry, we modify a design from PP$(q, 2, k)$. Specifically, we first construct pools from an affine plane of order q for $1 \leq k \leq q$ by invoking PP$(q, 2, k)$. Then, depending on whether $k = q$ or $k < q$ the design is obtained differently. For $k = q$, each pool of the affine plane is augmented with one variable. Denote by ℓ^{aff}_{qj+i} for $1 \leq i \leq q$ and $0 \leq j \leq q$ the pools from the affine plane. The projective plane is then obtained by generating the pools $\ell^{\text{proj}}_{qj+i} = \{\ell^{\text{aff}}_{qj+i}, v_{q^2+j+1}\}$ and adding one extra pool $\ell^{\text{proj}}_{q^2+q+1} = \{v_{q^2+j+1}\}_{j=0}^{q}$. The full projective plane ($k = q$) results in a square matrix, which is not advantageous for group testing. Therefore, we next develop an approach for $k < q$ which is practically more relevant. In particular, the first $q(k+1)$ pools are extended with one sample like before, $\ell^{\text{proj}}_{qj+i} = \{\ell^{\text{aff}}_{qj+i}, v_{q^2+j+1}\}$ for $1 \leq i \leq q$ and $0 \leq j \leq k$. Then we add $q - k$ "singleton" pools each containing only one sample so that $\ell^{\text{proj}}_{q(k+1)+s} = v_{q^2+k+s+1}$ with $1 \leq s \leq q - k$. A projective geometry enables designs with pool sizes $m = q + 1$ (and individual tests), and thus opens up new constructions. We denote designs from projective geometries by $\text{PP}_{\text{PG}}(q, 2, k)$.

Results 2

In the following we highlight characteristics of the PP method proposed in this work by comparing it to previously published methods presented in Section "Recent pooling methods applied to COVID-19".

Prime power order

We first compare PP to $(\overline{N},\overline{n},\overline{k})$-multipool designs [5], where overline notation was introduced in the leading paragraph of Section "Recent pooling methods applied to COVID-19". The construction of multipools is restricted to prime orders $\overline{n} = p$. The remaining design parameters follow as $\overline{N} = \overline{n}^2$ and $\overline{k} \leq \overline{n} + 1$. In the particular case of non-prime orders $\overline{n}$, the method is restricted to $\overline{k} \leq 3$. This limits the number of detectable positives to $\overline{k} - 1 = k = 2$ when non-prime orders are chosen. Table 2 compares the pooling designs that can be generated with the $(\overline{N},\overline{n},\overline{k})$-multipool method and with PP of non-prime order $n = 8$ and $N = 64$ for the detection of k positives (this set of parameters was proposed in [5, Section 4])

For the case considered in Table 2 the multipool method can generate designs for the detection of at most 2 positives whereas PP allows the construction of designs with $k \leq q$. Here we took advantage of the fact that PP can generate designs with prime power orders $q = p^n$, $n \geq 1$, in this case $q = 2^3 = 8$. We note that designs PP(8,2,k), $k \in \{7, 8\}$ are listed for completeness and do not have practical significance since the compression gain of these designs is not higher than individual testing.

Pool sizes and projective planes

Published methods often rely on the affine plane with pool size equivalent to the order of the affine plane $m = q$ [5, 13]. In the context of pool testing with two-stage decoding algorithms it has been suggested that pool size for optimal pooling efficiency depends on the prevalence [10, 11]. Table 3 shows pool size and corresponding prevalence values in the first two columns as obtained in [10]. Prevalences are defined

Table 2 Pooling desings of non-prime order

k	Multipools [5]		PP (this work)	
	Desc.	M	Desc.	M
1	(64,8,2)-multipool	16	PP(8,2,1)	16
2	(64,8,3)-multipool	24	PP(8,2,2)	24
3	n.a.	n.a.	PP(8,2,3)	32
4	n.a.	n.a.	PP(8,2,4)	40
5	n.a.	n.a.	PP(8,2,5)	48
6	n.a.	n.a.	PP(8,2,6)	56
7	n.a.	n.a.	PP(8,2,7)	64
8	n.a.	n.a.	PP(8,2,8)	72

Comparison of pooling designs with order $q = \overline{n} = 8$ from multipool [5] and PP methods to test $N = \overline{N} = 64$ samples. The designs from the multipool method can detect up to $k = 2$ positives, whereas PP can detect up to $k = 8$ positives

Table 3 Designs with optimal pool size for various prevalences

m	Prev.	N	M	G	Desc.
3	0.14	7	5	1.4	PP_{PG}(2,2,1) (1 singl.)†
4	0.071	14	8	1.8	PP(4,2,1)
5	0.043	23	10	2.3	PP(5,2,1)
6	0.031	32	15	2.1	PP_{PG}(5,2,1) (5 singl.)†
7	0.024	42	14	3	PP(7,2,1)
8	0.017	58	16	3.6	PP(8,2,1)
9	0.015	66	16	4.1	PP_{PG}(8,2,1)
10	0.01	96	31	3.1	PP_{PG}(9,2,1) (13 singl.)†
11	0.0083	121	22	5.5	PP(11,2,1)
12	0.0073	137	36	3.8	PP_{PG}(11,2,1) (14 singl.)†
13	0.0059	169	26	6.5	PP(13,2,1)
16	0.0039	256	32	8	PP(16,2,1)
18	0.0031	318	61	5.2	PP_{PG}(17,2,1) (27 singl.)†
32	0.00098	1024	64	16	PP(32,2,1)

† Number of singleton pools added to projective geometry. This ensures that each sample is tested in the case $k < q$ as described in section "Projective Geometry". Optimal pool sizes and prevalences are from [10]

by maximum detectable positives k divided by total samples N. Due to the discrete nature of pooling designs and their parameters, it is not always possible to match prescribed values of prevalence exactly, therefore the values of prevalence shown in Table 3 are within $\pm 5\%$ of the reported values.Designs based on projective geometries $PP_{PG}(q,2,k)$ exhibit pool size $m = q + 1$. As is apparent from Table 3 the projective geometries complement the PP designs with respect to the covered pool sizes. To the best of our knowledge, the combination of PP and PP_{PG} methods can yield a set of new designs not covered by previously published methods.

Prime power orders and higher dimensions

We refer to the paragraph of Algorithm 1 in Section "Algorithm", where we describe the dimension parameter d. There we highlight that designs with $d \geq 2$ yield high compression gains by generating matrices that grow fast with regard to the total number of samples $N = q^d$. Of the recently published methods, only STD [3] as well as PP, proposed in

this work, take advantage of the exponential growth in total sample number with dimension. Thus STD and PP are able to achieve significantly larger compression gains, especially at low rates of prevalence, than methods that have a fixed dimension of $d = 2$ e.g. PPoL [13] and "multipools" [5]. The significance of the dimension $d \geq 2$ is illustrated in Table 4 showing design parameters and achieved compression gains of different pooling designs for the detection of 2 positive among 961 samples. These values were proposed for the PPoL-(31,3) design in [13] for prevalences below 2%.

Table 4 lists the designs from PP (this work), STD [3], PPoL [13] and multipools [5] with the highest compression gain for these parameters. In this setting, designs generated by PP and STD are equivalent, the same applies to the designs generated by PPoL and multipools. The main difference to note is the fact that PPoL and multipools are restricted to dimension $d = 2$ and consequently achieve lower compression gains that are, for this set of paramters, half of the compressions achieved with STD and PP respectively.

Higher dimensions $d \geq 2$ allow the construction of designs with very large N. As has been discussed before, the only recently published method that takes advantage of variable dimension is STD [3]. However, STD is restricted to prime orders $q = p$. The PP method presented in this work permits construction based on prime order as well as prime power order. Therefore all designs that can be constructed by STD also can be constructed by PP. In the following, examples of designs for large values of N are given where application of variable dimension yields large compression gains, especially for low prevalences. Table 5 shows designs for numbers of samples $1000 \leq N \leq 25000$ for the detection of $1 \leq k \leq 50$ positives. Prime power orders $q = p^n$, $n \geq 1$ are highlighted with an asterisk, marking designs that are now available through the PP method proposed in this work.

For each set of parameters in Table 5 the design with the largest compression gain is shown. In general, large compression gains can be observed, especially when few positives are to be detected. Designs with prime power order $q = p^n$ are highlighted. We want to emphasize that these achieve the highest compression gain for this parameter set, surpassing designs based on prime order $q = p$.

Comparing with published designs

Further, Table 6 contrasts a broader range of published pooling designs with designs using PP. In [13] the PPoL method was compared to the methods from Table 6 excluding PP for varying rates of prevalence up to 5 % in terms of the two-stage decoding

Table 4 Pooling designs from published methods

q	d	M	G	Desc.
7	4	49	20	PP(7,4,2)
7	4	49	20	STD(961,4,7) [3]
31	2	93	10	PPoL-(31,3) [13]
31	2	93	10	(961,31,3)-multipools [5]

Parameters of pooling designs from published methods and from PP (this work) for guaranteed detection of $k = 2$ positives among $N = 961$ samples

Table 5 Parameters of large PP designs

q	N	k	d	M	G
4*	1000	1	5	20	50.0
11	1000	5	3	121	8.3
32*	1000	10	2	352	2.8
32*	1000	25	2	832	1.2
53	1000	50	2	2703	0.4
5	5000	1	6	30	166.7
19	5000	5	3	209	23.9
23	5000	10	3	483	10.4
71	5000	25	2	1846	2.7
71	5000	50	2	3621	1.4
5	15000	1	6	30	500.0
16*	15000	5	4	256	58.6
25*	15000	10	3	525	28.6
53	15000	25	3	2703	5.5
125*	15000	50	2	6375	2.4
8*	25000	1	5	40	625.0
16*	25000	5	4	256	97.7
31	25000	10	3	651	38.4
53	25000	25	3	2703	9.2
163	25000	50	2	8313	3.0

Designs and parameters for increasing sample numbers N for the detection of up to $k \in \{1, 5, 10, 25, 50\}$ positives

* Design based on prime power order $q = p^n$

Table 6 Overview of published methods and corresponing PP designs

		Published methods			PP (this work)		
N	k	M	G	Desc.	M	G	Desc.
961	2	93	10.3	PPoL-(31,3) [13]	49	20	PP(7,4,2)
529	3	92	5.75	PPoL-(23,4) [13]	63	8.4	PP(9*,3,3)
169	2	39	4.33	PPoL-(13,3) [13]	35	4.8	PP(7,3,2)
49	1	14	3.5	PPol-(7,2) [13]	12	4.1	PP(3,4,1)
348	5	48	8	P-BEST Matrix [15]	114	3.1	PP(19,2,5)
35	2	15	2.3	Kirkman Matrix	20	1.8	PP(4*,3,2)
40	2	16	2.5	Tapestry Matrix [16]	20	2	PP(4*,3,2)
60	2	24	2.5	Tapestry Matrix [16]	20	3	PP(4*,3,2)
96	1	20	4.8	2D-pooling Matrix [18]	15	6.4	PP(5,3,1)

Comparison of various published designs presented in [13] is extended to include PP designs

* Design based on prime power order $q = p^n$

algorithm DD2. Performance was measured by expected relative cost (ERC). It was concluded, that there was no single design with the lowest expected relative cost for all rates of prevalence and hence the design should be chosen to match expected prevalence rate. However, according to Figure 11 from [13] the PPoL-(31,3) design has advantages over other published methods as it showed lower ERC at low values of prevalence. At the same time its ERC stayed below competing designs for larger prevalences. Because all

PPoL designs are included in PP(q,2,k) designs with $q = p$ and further ones are possible with PP we extrapolate that the arguments made in [13] also apply to PP designs. Table 6 illustrates possible advantages with PPoL designs.

Discussion

We next describe practical aspects when applying our method.

Benefits

From the previous section ("Results") it is evident that PP enables designs that have not been possible with existing methods. For instance, multipool matrices for up to 2 positive samples with pool size $m = 8$ and $N = 64$ were suggested in [5]. However, when e.g., up to 3 positives are needed, multipool matrices are not applicable. Nevertheless, with the same pool and sample size PP enables designs for situations with up to 8 positive samples (cf. Table 2). This is reminiscent of the fact that PP permits prime number or prime power orders q.

Even though prime power orders are in principle also possible in [13], the methods therein are restricted to $d = 2$ and to affine geometries. PP enables constructions with $d \geq 2$, which can be useful for large scale testing because of very high compression gains.

For smaller, specifically targeted pool sizes, the option of using projective geometries, $\mathrm{PP}_{\mathrm{PG}}$, makes it possible to find one-round designs for optimal pool sizes from a two-round method [10] (cf. Table 3)

Finally, PP enables the combination of prime power orders with $2 \leq d$ and extends high throughput methods with $2 \leq d$, but which are constrained to prime number orders [3] (cf. Table 5).

We describe some additional practical considerations. When a certain fixed number of samples $\widehat{N}$ are to be tested, designs may not exist that exactly match $\widehat{N}$. It is then common to construct pooling tests that exceed this number, i.e., $\widehat{N} \leq N$, and to "use/tag" the excess samples $0 \leq N - \widehat{N}$ to be known negatives.

Another special situation arises with projective geometries. Since $\mathrm{PP}_{\mathrm{PG}}$ is obtained by augmenting a subset of $qk + 1$ pools from an affine plane with one sample and by adding a selection of "singleton" tests, the number of samples in tests is twofold: There are $q(k + 1)$ tests with $q + 1$ samples and $q - k$ tests with one sample.

Moreover, the PP designs are analyzed in the setting of one-round tests. Embedding pooling matrices with $d = 2$ in a two round setting results in different properties and trade-offs [13]. Future work can be based on the larger class of matrix designs from PP in two-stage pooling test settings.

For finite field computations with prime power order we use the libraries [21, 22].

Changing conditions and setup

When epidemiological conditions are changing, and the prevalence is difficult to estimate we suggest in a first approach to increase the parameter k (characteristic), when possible. This will increase the number of tests, however it enables the correct identification of more positive samples with the current design. When switching to

a different design is straightforward (e.g., because testing is automated by a robot arm), we suggest a discrete search for a new design based on the updated prevalence (i.e., enumerately searching for the highest gain among admissible PP(q, d, k) designs given an updated value of k and subject to $N \leq q^d$). Our method is directly applicable in most laboratory settings with a q-PCR assay. Standard well numbers on q-PCR machines are 24, 48, 96, 384 and 1,536. Therefore, so long the number of suggested tests of PP(q, d, k) is below the number of wells our method is directly applicable (e.g., all designs from Table 2 would fit on plates with $96 \leq$ wells).

Pool sizes

Generating designs with large pools dilutes the concentration of each sample. The article in [15] suggests that pooling 8 samples results in a reduction of ~ 3 PCR cycles (a factor of 2^3), pooling 16 samples results in a reduction of ~ 4 cycles (a factor of 2^4) and, thus in general, that pools of $8n$ samples result in a reduction of $\sim (2 + n)$ cycles. The work in [15] further investigates pools of sizes 48 and concludes that this number is feasible for identifying positive samples in a PCR test. The comparisons from Table 2(to the multipool matrices from [5]) are based on pool sizes of 8 samples, and are therefore well within pool sizes that have been experimentally confirmed. In addition, all designs in Table 3 have pools with upto 32 samples, while 6 out of 9 PP designs from Table 6 have pools with ≤ 48 samples.

Setting *k* too low

The value of the characteristic k determines how many positives can be correctly decoded. In the following we describe an approach for the situation that the value of k is below the actual observed number of positives. In particular, when there are more positive samples than can be correctly identified (a design with a value of k that is too low), then the decoding process may introduce false positives. Importantly, the process does not generate false negatives. Therefore, a practical strategy is to count the number of positive samples identified during decoding. If this number is greater than k, then more samples were positive than expected. Since in this case false positives have been identified, all samples that are labeled as positive are to be retested. Furthermore, if the tests are mainly performed to confirm suspected infections then the prevalence for the test may be significantly higher than the population one. Depending on the actual observed difference individual testing may be preferable in this situation. However, when testing is performed as a screening mechanism then it is probable that the samples are independent and that the prevalence in the tested samples approaches the population prevalence.

Additional applications

In principle, PP can be used for population-wide testing of other diseases with low prevalences, such as HIV. Other situations where it may become useful is in genome sequencing, when large numbers of samples are being processed [3] or in cryptography [23], among many more applications.

Moreover, we describe a detailed example in "Additional file 2", which lays out the steps in our group testing strategy. Note that our software implementations https://github.com/johannesbrust/PP automate the relevant matrix designs and decoding steps. The example further describes our method when the prevalence of positive samples is changing, as may be expected in realistic scenarios.

Conclusions

Since prevalences or testing settings may change with respect to COVID-19 it is important to have improved options for appropriate test designs. We propose a combinatorial method, Polynomial Pools (PP), that includes constructions from existing approaches and enables new designs. Because of this, PP is at least as effective as existing approaches, but additionally enables designs with more advantages.

Methods

To motivate the proof for Algorithm 1 (i.e., Lemmas 1,2) we use two basic principles:

- Each (x_i, y_{x_i}) represents one sample
- Each sample and therefore pairs of samples $\{(x_i, y_{x_i}); (x_j, y_{x_j})\}$ are computed using different polynomial slopes and intercepts (adapted for the "infinity" slope)

Suppose that $1 \leq k < p$ and let $j \neq i$. Select an arbitrary pair of samples $\{(x_i, y_{x_i}); (x_j, y_{x_j})\}$, computed using, say, a and b. Consider the different slope $\overline{a} \neq a$ and intercept $\underline{b} \neq b$. There are four possibilities to generate pools in PP, and consequently pairs of samples other than a base pair

Base: Slope and intercept given	a, b with $\{(x_i, y_{x_i}); (x_j, y_{x_j})\}$
Case 1: Slope differs	$\overline{a}, b$ with $\{(x_i, \overline{y}_{x_i}); (x_j, \overline{y}_{x_j})\}$
Case 2: Intercept differs	$a, \underline{b}$ with $\{(x_i, \underline{y}_{x_i}); (x_j, \underline{y}_{x_j})\}$
Case 3: Slope and intercept differ	$\overline{a}, \underline{b}$ with $\{(x_i, \overline{\underline{y}}_i); (x_j, \overline{\underline{y}}_j)\}$
Case 4: Slope at infinity	$\overline{a} = q$ with $\{(x_i^\infty, y_i^\infty); (x_j^\infty, y_j^\infty)\}$

Let $y^+ \in \{\overline{y}, \underline{y}, \overline{\underline{y}}\}$ represent a y value from one of the Cases 1–3. Note that the conditions for having up to $d-1$ occurrences of a pair are for $k \in \{i, j\}, t \in \{i, j\}, k \neq t$

$$\text{if } y_{x_k} = y_{x_k}^+ \text{ then } y_{x_t} = y_{x_t}^+ \text{ at most } d-2 \text{ times.} \tag{6}$$

Recall the statement for Lemma 1: *Any pair of samples v_i, v_j, for $i \neq j$ from a design of Algorithm 1 occurs at most $d-1$ times.*

Proof Using the four cases with $2 \leq d$ and $q = p^n, 1 \leq n$, points (and thus sample indices) are computed by polynomials over the finite field $\mathbb{F}_q$. In the first three cases the slopes are not "infinite" in $\mathbb{F}_q$ and points are computed using

$$x_k = \sum_{l=1}^{d-1} q^{d-1-l} k_l, \quad y_{x_k} = (\sum_{l=1}^{d-1} a^l k_l + b)_{\mathbb{F}_q}, \quad 0 \leq k_l \leq q-1, \quad k \in \{i, j\}$$

In the fourth case we describe the pools with slope of "infinity". Suppose that $i \neq j$.

Case 1: In this case the slopes differ only. Therefore, $a \neq \overline{a}$ and there is an integer g such that $a + g = \overline{a}$. Similarly, there are integers $k \neq t \in \{i, j\}$ and "subindices" k_l, t_l for $1 \leq l \leq d-1$ so that $k_l + g_l = t_l$. Condition (6) is in this case for $k \neq t \in \{i, j\}$

$$\text{if } y_{x_k} = \overline{y}_{x_k} \text{ then } y_{x_t} = \overline{y}_{x_t} \text{ at most } d-2 \text{ times.}$$

The difference $y_{x_k} - \overline{y}_{x_k} = 0$ for $k \in \{i, j\}$ and $k_l + g_l = t_l$ yields

$$y_{x_k} - \overline{y}_{x_k} = \left(\sum_{l=1}^{d-1} (a^l - \overline{a}^l) k_l \right)_{\mathbb{F}_q} = 0 \quad \text{and} \quad \left(\sum_{l=1}^{d-1} (a^l - \overline{a}^l) t_l \right)_{\mathbb{F}_q} = \left(\sum_{l=1}^{d-1} (a^l - \overline{a}^l) g_l \right)_{\mathbb{F}_q}$$

Note that from $\overline{a} = a + g$ it holds that $a^l - \overline{a}^l = a^l - (a+g)^l$. Thus, $a^l - \overline{a}^l$ is a polynomial of degree at most $l-1$ with respect to a since the first term in the binomial expansion of $(a-g)^l$ is a^l. Then

$$y_{x_t} - \overline{y}_{x_t} = \left(\sum_{l=1}^{d-1} (a^l - \overline{a}^l) t_l \right)_{\mathbb{F}_q} = \left(\sum_{l=1}^{d-1} (a^l - \overline{a}^l) g_l \right)_{\mathbb{F}_q} = \left(\sum_{l=1}^{d-1} (a^l - (a+g)^l) g_l \right)_{\mathbb{F}_q}$$

The last equality is a polynomial with degree up to $d-2$ in $\mathbb{F}_q$ in terms of a. Therefore, for $k \neq t$ when $y_{x_k} = \overline{y}_{x_k}$ then $y_{x_t} = \overline{y}_{x_t}$ up to $d-2$ times (at the roots of a polynomial). We subsequently conclude that pairs of samples, including the "base pair" $\{(x_i, y_{x_i}); (x_j, y_{x_j})\}$ occur up to $d-1$ times.

Case 2: The intercept differs and with $k \neq t \in \{i, j\}$

$$y_{x_k} - \underline{y}_{x_k} = (b - \underline{b})_{\mathbb{F}_q} \neq 0$$

Therefore $y_{x_i} \neq \underline{y}_{x_i}$ and $y_{x_j} \neq \underline{y}_{x_j}$ and all samples and corresponding pairs are different in this case.

Case 3: In this case both; slope and intercept differ. Therefore, $a \neq \overline{a}$, $b \neq \underline{b}$ and there is an integer g such that $a + g = \overline{a}$. As in Case 1 there are integers $k \neq t \in \{i, j\}$ and "subindices" k_l, t_l for $1 \leq l \leq d-1$ so that $k_l + g_l = t_l$. The relevant condition for $k \neq t \in \{i, j\}$ is

$$\text{if } y_{x_k} = \underline{\overline{y}}_{x_k} \text{ then } y_{x_t} = \underline{\overline{y}}_{x_t} \text{ at most } d-2 \text{ times.}$$

The difference $y_{x_k} - \underline{\overline{y}}_{x_k} = 0$ for $k \in \{i, j\}$ and $k_l + g_l = t_l$ yields

$$y_{x_k} - \underline{\overline{y}}_{x_k} = \left(\sum_{l=1}^{d-1} (a^l - \overline{a}^l) k_l + (b - \underline{b}) \right)_{\mathbb{F}_q} = 0 \quad \text{and}$$

$$\left(\sum_{l=1}^{d-1} (a^l - \overline{a}^l) g_l \right)_{\mathbb{F}_q} = \left(\sum_{l=1}^{d-1} (a^l - \overline{a}^l) t_l + (b - \underline{b}) \right)_{\mathbb{F}_q}$$

Similar like before note that from $\overline{a} = a + g$ it holds that $a^l - \overline{a}^l = a^l - (a+g)^l$. Thus, $a^l - \overline{a}^l$ is a polynomial of degree at most $l-1$ since the first term in the binomial expansion of $(a-g)^l$ is a^l. Then

$$y_{x_t} - \underline{y}_{x_t} = \left(\sum_{l=1}^{d-1}(a^l - \overline{a}^l)t_l + (b - \underline{b})\right)_{\mathbb{F}_q} = \left(\sum_{l=1}^{d-1}(a^l - (a+g)^l)g_l\right)_{\mathbb{F}_q}.$$

The right hand side in the last equality is a polynomial of degree up to $d-2$ with respect to a in $\mathbb{F}_q$. Therefore, for $k \neq t$ when $y_{x_k} = \underline{y}_{x_k}$ then $y_{x_t} = \underline{y}_{x_t}$ up to $d-2$ times (at the roots of a polynomial). We subsequently conclude that pairs of samples, including the "base pair" $\{(x_i, y_{x_i}); (x_j, y_{x_j})\}$ occur up to $d-1$ times.

Case 4: With the slope at "infinity" in $\mathbb{F}_q$ the formula for computing points is adapted. Recall that for $k \in \{i,j\}$, $x_k = \sum_{l=1}^{d-1} p^{d-l-1}k_l$ so that the formula for x_k^∞ becomes

$$x_k^\infty = \sum_{l=2}^{d-1} p^{d-l-1}i_l + bq^{d-2} = x_k - q^{d-2}(b - x_{k_{d-2}}), \quad k \in \{i,j\}$$

Therefore one sees that for $k \in \{i,j\}$

$$x_k^\infty = x_k \quad \text{implies} \quad x_{k_{d-1}} = b$$

To investigate the condition in (6) we note that

$$y_{x_k} = y_k^\infty \quad \text{yields} \quad (\sum_{l=1}^{d-1} a^l k_l + b)_{\mathbb{F}_q} - k_{d-1} = 0.$$

Therefore, for $k \neq t \in \{i,j\}$ (and with $k_{d-1} = t_{d-1}$),

$$\begin{aligned}\text{if } y_{x_k} = y_{x_k}^\infty \text{ then } y_{x_t} - y_{x_t}^\infty &= \left(\sum_{l=1}^{d-1} a^l t_l + b\right)_{\mathbb{F}_q} - t_{d-1} \\ &= \left(\sum_{l=1}^{d-1} a^l (t_l - k_l)\right)_{\mathbb{F}_q} - (t_{d-1} - k_{d-1}) = \left(\sum_{l=1}^{d-2} a^l (t_l - k_l)\right)_{\mathbb{F}_q}\end{aligned}$$

The final equality is a polynomial of degree up to $d-2$ with respect to a and it has up to $d-2$ roots. Therefore we conclude that a pair of samples occurs up to $d-1$ times in Case 4. In summary, in all cases with $2 \leq d$ and $q = p^n, 1 \leq n$ pairs of samples occur at most $d-1$ times according with the result of Lemma 1. □

Lemma 2 follows from the previous analysis. Recall the statement of Lemma 2: *Each sample v_i is contained in up to $k(d-1)+1$ tests.*

Proof First note that one pool corresponds to one combination of slope and intercept a and b (including the infinity slope). When only the intercept b differs (and a is fixed), each sample occurs exactly once for a combination of a and all possible values of b. Since

$0 \leq a \leq k(d-1)$ there are $k(d-1)+1$ different a values and thus each sample occurs in $k(d-1)+1$ tests. □

For $d = 2$ and q being a prime number the formulas in PP can be represented using modulus arithmetic and lines. In this situation a further proof of Lemma 1 is given in the "Additional file 1".

Abbreviations

PP	Polynomial pools
PG	Projective geometry
COMP	Combinatorial orthogonal matching pursuit [4]
STD	Shifted transversal design [3]
PPoL	Packing the pencil of lines [13]
P-BEST	Pooling-based efficient SARS-CoV-2 testing [15]
DD2	Two-stage definitive defectives [14]
ERC	Expected relative cost
PCR	Polymerase chain reaction

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-023-05145-y.

Additional file 1: Proof of Lemma 1.

Additional file 2: A detailed example.

Acknowledgements
We thank the associate editor and two referees for their careful reading of our manuscript and for their valuable comments.

Author contributions
DB produced Tables 2, 3, 4, 5 and 6 and implemented a Julia version of the proposed method. JJB derived the polynomial representation of pools and implemented a Matlab version of the proposed method. Both authors were major contributors in writing the manuscript.

Authors' information
DB is an engineer with more than 8 years of programming experience. JJB is an applied mathematician who developed affine and projective geometries for COVID-19 pool testing.

Funding
Not applicable.

Availability of data and materials
The datasets generated and/or analysed during the current study are available in the PP (PolynomialPools) repository, https://github.com/johannesbrust/PP

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 22 August 2022 Accepted: 10 January 2023
Published online: 24 January 2023

References

1. Dorfman R. On a problem in combinatorics. Camb Dublin Math J. 1847;14(4):436–40.
2. Du D-Z, Hwang FK. Combinatorial group testing and its applications. 2nd ed. Singapore: World Scientific; 1999. https://doi.org/10.1142/4252.
3. Thierry-Mieg N. A new pooling strategy for high-throughput screening: the shifted transversal design. BMC Bioinform. 2006;7:28.

4. Chan CL, Che PH, Jaggi S, Saligrama V. Non-adaptive probabilistic group testing with noisy measurements: near-optimal bounds with efficient algorithms. In: In 2011 49th annual Allerton conference on communication, control, and computing (Allerton); 2011. p. 1832–9.
5. Täufer M. Rapid, large-scale, and effective detection of covid-19 via non-adaptive testing. J Theor Biol. 2020;506:110450.
6. Colbourn CJ, Dinitz JH. Handbook of combinatorial designs. 2nd ed. New York: Chapman & Hall; 2007.
7. Berger T, Mandell JW, Subrahmanya P. Maximally efficient two-stage screening. Biometrics. 2000;56(3):833–40.
8. Kim H-Y, Hudgens MG, Dreyfuss JM, Westreich DJ, Pilcher CD. Comparison of group testing algorithms for case identification in the presence of test error. Biometrics. 2007;63(4):1152–63.
9. Kim H-Y, Hudgens MG. Three-dimensional array-based group testing algorithms. Biometrics. 2009;65(3):903–10.
10. Regen F, Eren N, Heuser I, Hellman-Regen J. A simple approach to optimum pool size for pooled sars-cov-2 testing. Int J Infect Dis. 2020;100:324–6.
11. Hanel R, Thurner S. Boosting test-efficiency by pooled testing for sars-cov-2-formula for optimal pool size. PLoS ONE. 2020;15(11):1–10.
12. Kainkaryam RM, Woolf PJ. poolhits: a shifted transversal design based pooling strategy for high-throughput drug screening. BMC Bioinform. 2007;9:256–256.
13. Lin YJ, Yu CH, Liu TH, Chang CS, Chen WT. Constructions and comparisons of pooling matrices for pooled testing of covid-19. IEEE Trans Netw Sci Eng. 2022;9(2):467–80.
14. Aldridge M, Johnson O, Scarlett J. Group testing: an information theory perspective. Found Trends® Commun Inf Theory. 2019;15(3–4):196–392.
15. Shental N, Levy S, Wuvshet V, Shosh S, Shalem B, Ottolenghi A, Greenshpan Y, Steinberg R, Edri A, Gillis R, Goldhirsh M, Moscovici K, Sachren S, Friedman LM, Nesher L, Shemer-Avni Y, Porgador A, Hertz T. Efficient high-throughput sars-cov-2 testing to detect asymptomatic carriers. Sci Adv. 2020;6.
16. Ghosh S, Rajwade A, Krishna S, Gopalkrishnan N, Schaus TE, Chakravarthy A, Varahan S, Appu V, Ramakrishnan R, Ch S, Jindal M, Bhupathi V, Gupta A, Jain A, Agarwal R, Pathak S, Rehan MA, Consul S, Gupta Y, Gupta N, Agarwal P, Goyal R, Sagar V, Ramakrishnan U, Krishna S, Yin P, Palakodeti D, Gopalkrishnan M. Tapestry: a single-round smart pooling technique for covid-19 testing. medRxiv 2020.
17. Ghosh S, Agarwal R, Rehan MA, Pathak S, Agrawal P, Gupta Y, Consul S, Gupta N, Goyal R, Rajwade AV, Gopalkrishnan M. A compressed sensing approach to group-testing for covid-19 detection. arXiv: Quantitative Methods. 2020.
18. Sinnott-Armstrong N, Klein D, Hickey B. Evaluation of group testing for sars-cov-2 rna. medRxiv 2020.
19. Brust JJ. Matrix designs for covid-19 group testing. ILAS IMAGE. 2022;68:9–16.
20. Stinson DR. Combinatorial designs: constructions and analysis. New York: Springer; 2004.
21. Cheng S. A toolbox for simple finite field operation (2022). https://www.mathworks.com/matlabcentral/fileexchange/32872-a-toolbox-for-simple-finite-field-operation. Accessed 7 Jul 2022.
22. Kluck T. GaloisFields.jl—finite fields for Julia (2022). https://github.com/tkluck/GaloisFields.jl. Accessed 7 Jul 2022.
23. Colbourn CJ, Dinitz JH, Stinson DR. Communications, cryptography, and networking. Surv Comb. 1999;267:37–41.

Publisher's Note