HYPER: Group testing via hypergraph factorization applied to COVID-19

David Hong[1] Rounak Dey[2] Xihong Lin[2,3,4,*] Brian Cleary[4,*] Edgar Dobriban[1,*]

[1]Department of Statistics, The Wharton School, University of Pennsylvania, Philadelphia, PA, USA

[2]Department of Biostatistics, Harvard T.H. Chan School of Public Health, Boston, MA, USA

[3]Department of Statistics, Harvard University, Cambridge, MA, USA

[4]Broad Institute of MIT and Harvard, Cambridge, MA 02142 USA

[*]Correspondence should be addressed to: *xlin@hsph.harvard.edu* (X.L.), *bcleary@broadinstitute.org* (B.C.) and *dobriban@wharton.upenn.edu* (E.D.).

Abstract

Large scale screening is a critical tool in the life sciences, but is often limited by reagents, samples, or cost. An important challenge in screening has recently manifested in the ongoing effort to achieve widespread testing for individuals with SARS-CoV-2 infection in the face of substantial resource constraints. Group testing methods utilize constrained testing resources more efficiently by pooling specimens together, potentially allowing larger populations to be screened with fewer tests. A key challenge in group testing is to design an effective pooling strategy. The global nature of the ongoing pandemic calls for something simple (to aid implementation) and flexible (to tailor for settings with differing needs) that remains efficient. Here we propose HYPER, a new group testing method based on hypergraph factorizations. We provide characterizations under a general theoretical model, and exhaustively evaluate HYPER and proposed alternatives for SARS-CoV-2 screening under realistic simulations of epidemic spread and within-host viral kinetics. We demonstrate that HYPER performs at least as well as other methods in scenarios that are well-suited to each method, while outperforming those methods across a broad range of resource-constrained environments, and being more flexible and simple in design, and taking no expertise to implement. An online tool to implement these designs in the lab is available at `http://hyper.covid19-analysis.org`.

Introduction

Biological screens that identify members of a large population with a disease have become invaluable tools for disease diagnosis and surveillance. When these screens are difficult to conduct or resources are limited, finding an efficient way to conduct the screen becomes critical. As such, widespread, scalable and frequent testing is a defining challenge in combatting COVID-19 in the face of local, national and global resource constraints. Pooled testing has recently arisen as a promising efficient scientific solution to the world-wide challenge of increasing SARS-CoV-2 testing capacity[1–14], encouraged in part by the finding that a single positive sample can be reliably detected by RT-qPCR in large pools[15].

One approach to achieve efficiency is to design the screen in such a way that leverages structure in the tested population[16]. This idea dates back at least to the seminal work of Dorfman[17], which proposed testing *pools* of samples when there is prior knowledge that the vast majority of samples will test negative. Dorfman testing is a two-stage approach with each individual assigned to exactly one pool. A negative test result for a pool at the first stage can be applied to all its members, eliminating the need to test them individually and potentially greatly increasing efficiency, depending on the pool size and prevalence of positive members of the population. A great strength of this approach is its simplicity and robustness in laboratory implementation. Pools are easy to form and putative positives are simply those individuals contained in a positive pool. Several early methods[3–5;18] for COVID-19 pooled testing focus on Dorfman testing. However, it is well-known that Dorfman testing can have sub-optimal efficiency[7;8;13;19]; alternative designs use tests more efficiently and can thus screen more individuals, especially in the face of major resource constraints.

There has been tremendous study and progress on pooled testing (also called group testing or specimen pooling) in general. Numerous works provide statistical[20–24], combinatorial[25–29], and information theoretic[2;30–38] perspectives, as well as software[39;40] to aid implementation, to name just a few. Additionally, there has been a lot of work on analyzing and optimizing group testing for various constraints and evaluation criteria[41–51], often in the low prevalence regime. Broadly speaking, the approaches fall into three categories: i) single-stage (or nonadaptive) approaches that identify positive individuals after only one round of pooled tests by using pools with carefully designed overlaps; ii) two-stage[50–53] approaches that declare putative positives after one round of pooled tests. The putative positives are then individually tested in the second round. Finally, iii) multi-stage[52;54] (or adaptive/hierarchical) approaches that carry out multiple rounds of pooled tests with pools at each round chosen based on previous rounds.

Many recent methods[6–13;55] for COVID-19 pooled testing (more than can be reviewed here) are seeking greater efficiency by splitting samples into multiple pools, and using a one-, two- or multi-stage approach. One such method is P-BEST[6], which is a single-stage approach designed for a prevalence around 1% that splits each of 384 individuals into 48 partially overlapping pools. The pool assignments are based on a Reed-Solomon error correcting code that enables identification from the single round of tests and provides robustness against, e.g.,

independent PCR failures. Positive individuals are identified by running a specialized decoding algorithm based on sparse regression. A popular two-stage method is array pooling[7], which arranges individuals into either an 8×12 or 16×24 grid (corresponding to plate sizes common in laboratory environments), then takes each column and each row to be a pool, resulting in 20 pools for 96 individuals or 40 pools for 384 individuals, respectively. Each individual is split into two pools and is a putative positive if both those pools test positive. This approach is potentially more efficient than Dorfman testing while retaining some simplicity. Moreover, it is well-suited for clinics that already store samples in these grids, especially if they have multi-channel pipettes that make row/column pooling convenient. Another method extends this idea to samples arranged in a q-dimensional hypercube[8] with sides of length three. The approach is multi-stage, with each of 3^q individuals placed into q "slice" pools (chosen from $3q$ total pools) in the first round, followed by adaptively formed smaller hypercubes. For increasing q, this design covers numerous individuals using only a few tests in the first round, and is highly efficient when prevalence is low. Together, the proposed pooling strategies offer several alternatives that address the urgent, global need for efficient screening.

However, given the global nature of the pandemic, there is a wide variety of settings with differing needs and constraints, in which the proposed combinatorial designs may have limited utility[19]. Designs that split samples into more than two or three pools, e.g., P-BEST and hypercubes, can be time-consuming and error-prone to execute by hand, making them best-suited for well-resourced labs with robotic-pipetting platforms. The decoding algorithm used by P-BEST and the multi-stage logistics of hypercubes can also make them nontrivial to implement without prior experience, expertise, and substantial lab infrastructure. Moreover, many of the proposed designs are somewhat rigid and are not trivially adapted to environments that can have widely varying numbers of individuals to test, available test kits, or prevalence of positive results. Array pooling with 150 individuals, for example, might be done using a partially-filled 16×24 grid or perhaps two 8×12 grids with one or both partially-filled, but it is unclear which to choose and whether either remains efficient under variable prevalence. For SARS-CoV-2 screening in resource-limited settings, adapting to these various conditions is critical to achieving the greatest effectiveness[19]. Therefore, for this application along with broader applications of pooled screening, there remains an outstanding need for a robust, simple, flexible strategy with performance that matches (or exceeds) the effectiveness of specialized strategies, and can be applied in diverse environments without special equipment or expertise.

We propose HYPER, a two-stage pooling strategy based on the combinatorics of hypergraph factorizations. In this approach, individuals are assigned to pools by cycling through a carefully ordered sequence, given by a combinatorial construction. While the underlying mathematics is sophisticated, the resulting pools are simple to implement by hand (individuals are split at most three ways), and putative positives can be easily identified with only pencil and paper. We also provide an online tool available at `http://covid19-analysis.org` to facilitate implementation. The design accommodates any number of individuals while maintaining robustness,

balance and efficiency. One simply cycles through the sequence until everyone is assigned. We characterize its performance through theoretical analysis, as well as through realistic simulations that model both viral kinetics and epidemic spread. Our findings demonstrate that HYPER is not only simple to implement and decode, but is also very efficient. Across a broad range of resource-constrained environments, it outperforms both array and P-BEST designs even in the scenarios where those methods excel in efficiency. Furthermore, in some scenarios we consider, it screens over three times as many effective individuals as both existing methods and 14 times more than individual testing. Its flexibility, in particular, makes it easy to adapt to each scenario. For SARS-CoV-2 and beyond, HYPER represents a valuable addition to the growing toolbox for performing large-scale, pooled screens.

Results

Limitations of existing designs and the need for balanced designs that are simple and flexible: A pooling design is an assignment of each of n individuals (or more generally, samples) to one or several of m pools. We seek a simple and flexible pooling design that is balanced in the following natural ways:

i) All individuals are assigned to the same number q of pools; we focus on $q = 1, 2, 3$ for practical implementation.

ii) The m pools are assigned as evenly as possible, meaning that the sizes of the pools are as close as possible to equal.

iii) The $\binom{m}{q}$ possible pool combinations are assigned as evenly as possible.

Similar conditions have been widely studied in group testing, see the references above. However, as explained below, methods that satisfy our constraints ($q \leq 3$, near-optimal efficiency and noise tolerance, at most two stages) are not available. The above balance conditions encourage the design to maximally utilize the overall pooling resources available, while allowing for robust implementation and quality control. For instance, the volume used for each sample, or produced for each pool and combination of pools, should be balanced. This also ensures equitable treatment of each individual sample. Ensuring uniform splits q and balanced pools is especially relevant for uniform treatment of the individuals since both affect the impact of dilution on each individual. Maximally balancing the pool combinations can be helpful for efficiency. As a simple example, suppose one individual is positive. After pooled testing, that individual will be identified as a putative positive and needs to be retested, *along with any others assigned to the same pool combination.* With imbalanced pool combinations, unless we are lucky, the positive individual may be assigned to one of the more frequently assigned combinations. This in turn would lead to an excess number of putative positives and additional tests, i.e., worse efficiency.

For $q = 1$, corresponding to classical Dorfman testing, each individual needs to be assigned to a single pool. Balance can be easily achieved here by cycling through the pools until all individuals are assigned. For example, to assign $n = 8$ individuals to $m = 6$ pools (A-F), assign individual 1 to pool A, individual 2 to B, and so on, yielding the following assignments:

Individual	1	2	3	4	5	6	7	8
Pool	A	B	C	D	E	F	A	B

All the $n = 8$ individuals are assigned to $q = 1$ pool, the $m = 6$ pools are assigned as evenly as possible (A and B are assigned twice and the rest are assigned once), and pool "combinations" are just the pools themselves when $q = 1$.

Creating maximally balanced designs for $q > 1$, and especially $q > 2$, can be much harder. A straightforward approach of listing the pools and assigning individuals to consecutive pairs (e.g., AB, CD, EF, AB, ...) under-utilizes the combinatorial space (for instance, it does not use AC). At the same time, cycling over all pairs sorted in lexicographic order is likely to be unbalanced in multiple ways. For instance, using all pairs in lexicographic order for $n = 8$ individuals assigned to $q = 2$ of $m = 6$ pools, yields

Individual	1	2	3	4	5	6	7	8
Pools	AB	AC	AD	AE	AF	BC	BD	BE

which assigns A five times and F only once. As a result, this design is *not* uniformly sensitive across individuals. For example, individual 1 undergoes a five-fold dilution in pool A and a four-fold dilution in pool B, while individual 5 is diluted in pool A but not at all in pool F. Uneven dilution of viral loads can lead to differential false negative test results, leaving individual 1 with a higher risk of false negatives than individual 5. Clearly, alternative orderings could be more balanced, and the challenge is how to systematically identify these.

One approach is to generate a random design[19;56;57], e.g., by assigning each individual to q of m pools independently and uniformly at random. Doing so treats both pools and pool combinations uniformly "on average", but any given random draw may assign them very unevenly in practice, especially when m, n, q are not very large. For the example above, this approach draws one of the $\binom{m}{q}^n = 15^8$ possible designs uniformly at random, so pool sizes (subjects per pool) can range all the way from zero to n. Likewise for pool combinations. In fact, 98.7% of the random draws in this case will not be maximally balanced, i.e., the pools or pool combinations are not assigned as evenly as possible, as can be verified by exhaustive search. In other words, it is rare for randomly generated designs to both treat individuals uniformly, and maximally use all available pool combinations. With high probability, selecting a design at random does not achieve our goal. With increasing m, n, q, the probability decreases exponentially, making random search practically very expensive for sufficiently large designs.

We can also consider searching among many random draws for a suitable design. In the small example above, it would take 180 draws to have a 90% chance of drawing at least one maximally balanced design. One

might try selecting then modifying the most balanced among many draws, but such an approach is ad-hoc, involves manual tweaking, and may still not be maximally balanced.

An exhaustive search, on the other hand, is a systematic approach but is only practical for very small cases. The $\binom{m}{q}^n$ possible designs to search through is an enormous and intractable number in general. Moreover, one must repeat the process for any change in the numbers of individuals n, number of splits q, or number of pools m; the design is not easily adapted from one environment to another. P-BEST[6] attains balanced pools and pool combinations with a Reed-Solomon error-correcting design that requires technical expertise to adapt, and may only remain balanced in limited settings. Array designs[7] can also be adapted to have balanced pools (by requiring the array to be a square), but they do not use all available pool combinations. They can have sub-optimal efficiency in certain of our simulation settings.

As a result, though it is quite natural to desire maximally balanced designs, the problem of easily generating them is highly nontrivial. In fact, it is not initially clear that designs maximally balanced in all three ways even exist in general, let alone, an efficient way to find them.

Here we show how certain deep results from combinatorics, such as Baranyai's theorem[58] imply that such designs do in fact exist, under the minimal required conditions. Moreover, we demonstrate how to generate them under mild conditions using the combinatorics of hypergraph factorizations. For $q = 3$, we leverage a non-trivial number-theoretic construction due to Beth[59;60]. These tools enable us to develop a simple, flexible and efficient pooled testing strategy with maximally balanced pool designs.

HYPER pooling method: We propose HYPER, a two-stage pooling strategy that uses maximally balanced pools built on hypergraph factorizations. Stage one consists of *pooled testing* to identify putative positives, who are tested individually in stage two. For convenience, we use $\mathrm{H}_{n,m,q}$ to denote the HYPER design with n individuals per batch, m pools, and q splits. We illustrate the details via a small example with $n = 12$ individuals each split into $q = 2$ out of $m = 6$ pools (Fig. 1a).

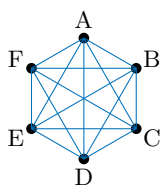
First, to create pools for stage one, we assign each of the n individuals to q of the m pools by cycling through an ordering of the $\binom{m}{q}$ possible pool combinations generated via hypergraph factorization (as described below). If $q = 1$, this sequence of possible pool assignments would consist of the m pools in an arbitrary order. In our more involved example (Fig. 1a), the sequence reads AB, CD, etc., so individual 1 is assigned to pools A and B, individual 2 to C and D, and so on. In this case, we did not use all $\binom{6}{2} = 15$ pairs in the sequence since there were only $n = 12$ individuals. However, if there were $n > 15$ individuals, we would cycle through the sequence again until all individuals were assigned. Hence, once we have the sequence, the pools are simple and flexible to form for any number of individuals n.

Next, we test each pool and *decode* the test results to identify putative positives. An individual found only in positive pools is a putative positive. Put another way, putative positives are the individuals left over after

a Example: $n = 12$ individuals, $m = 6$ pools, $q = 2$ splits; individuals 4 and 7 are positive ($\oplus$).

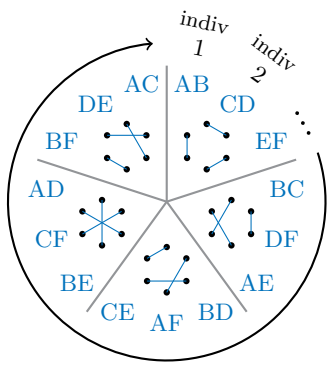
Online tool: `http://hyper.covid19-analysis.org`

Setup: factorize the hypergraph (for $m = 6$ pools with $q = 2$ splits)



vertices : pools (A-F)
edges : all possible indiv. pool assignments (AB,AC,...)

to obtain a sequence of individual pool assignments (AB, CD, ...):



Stage 1: pooled testing of the $n = 12$ individuals.

Individual pool assignments obtained from setup

	AB	CD	EF	BC	DF	AE	BD	AF	CE	BE	CF	AD	
Individual	1	2	3	4	5	6	7	8	9	10	11	12	Test
Pool A	●					●		●				●	$\ominus$
Pool B	●			●			●			●			$\oplus$
Pool C		●		●					●		●		$\oplus$
Pool D		●			●		●					●	$\oplus$
Pool E			●			●			●	●			$\ominus$
Pool F			●		●			●			●		$\ominus$
Decoding	$\ominus$	P	$\ominus$	P	$\ominus$	$\ominus$	P	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$	

Conclusion: individuals 2, 4 and 7 are putative positives (P).

Stage 2: individual testing of the putative positives (P).

Individual	2	4	7
Test Result	$\ominus$	$\oplus$	$\oplus$

Final conclusion: identify 4 and 7 as positive.

b

	HYPER	Random[19]	Array designs[7] 8×12	Array designs[7] 16×24	P-BEST[6]	Hypercube[8]
# individuals per batch (n)	any	any	96	384	384	3^q/stage
# pools (m)	variable	any	20	40	48	$3q$/stage
# splits (q)	≤ 3	any	2		6	q/stage
# stages	two	two	two		one	multi
balanced pools	✓	× w.h.p.*	×		✓	✓
balanced combinations	✓	× w.h.p.*	×		✓	×
simple to implement by hand	✓	✓	✓		×	✓?
flexible / easily adapted	✓	✓	×		×	×
simple to decode by hand	✓	✓	✓		×	×
corrects false positive	✓	✓	✓		✓	✓
corrects false negatives	optional	optional	×		✓	✓?

*with high probability, i.e., probability $\gg 0$.

Fig. 1 Overview of the HYPER method. a Illustrative example. Stage 1 tests pools that are formed by cycling through a sequence of pool assignments generated via hypergraph factorization. Putative positives are those individuals that are not in any negative pools (decoding). Stage 2 tests the putative positives individually. In this example, $n = 12$ individuals (2 of whom are actual positives) are each split into $q = 2$ of $m = 6$ pools; 3 are decoded as putative positives and both positives are successfully identified in stage 2. **b** Comparison of various features of HYPER with those of random designs[19], array designs[7], P-BEST[6], and Hypercube[8]. HYPER is a maximally balanced design that is simple to implement/decode by hand, flexible and efficient.

removing all individuals in negative pools. This is easily figured out with only pencil and paper, or a basic spreadsheet for larger designs (and one can also use our web implementation). In our example (Fig. 1a), pools A, E, F test negative so only individuals 2, 4 and 7 are putative positive. These samples are then individually tested in stage two.

Note that this way of decoding pooled testing results does not correct for any pooled tests with false negative results, e.g., as is done in P-BEST[6]. Error correction may be incorporated by introducing a tolerance so that individuals in just one or two negative pools are still considered putative positives[19]. Doing so can improve the overall sensitivity of HYPER. However, an important source of false negatives is the dilution of viral loads below the limit of detection (which is a major concern). An individual with small viral load is hence likely to yield false negatives in most of its pools, in which case error-correction may not significantly improve sensitivity. Meanwhile, it can dramatically reduce efficiency. In our example (Fig. 1a), allowing a tolerance of one negative pool would turn individuals 1, 5, and 9-12 into putative positives. Labs must consider how to properly weigh these tradeoffs according to their specific needs. Here we will focus primarily on an error tolerance of zero, i.e., no error-correction of false negatives.

The only missing piece now is how we use hypergraph factorization to create the sequence of pool combinations for $q = 2$ and $q = 3$. Here we provide a brief overview, with further details of the construction provided in Supplementary Methods. The idea is to order the $\binom{m}{q}$ combinations so that there is no overlap among the combinations in each consecutive block of m/q combinations (which is in fact the formal definition of a "hypergraph factorization", as we discuss below). In our example (Fig. 1a), the first $m/q = 3$ pairs (AB, CD, EF) do not overlap, as with the next three (BC, DF, AE), and so on. This property keeps pool sizes maximally balanced, and since each combination appears once, the combinations are balanced as well. Note that no pair is repeated and all pools contain 4 individuals. If we instead had only 11 individuals, the pools would no longer be perfectly even, since pools A and D would have only 3 individuals, but they would still be as even as possible. The same holds for any number of individuals.

Our task, therefore, is to divide the $\binom{m}{q}$ combinations into subsets of combinations, where each pool appears exactly once in each subset. As illustrated in Fig. 1a, we can think of each pool as one of m vertices in a *hypergraph*, where each *hyperedge* is a set of q of the m vertices (pools). Each hyperedge will correspond to a potential set of pools into which any individual sample is split. Our example has $\binom{6}{2} = 15$ hyperedges (blue edges in Fig. 1a). Putting together all $\binom{m}{q}$ hyperedges forms the so-called complete hypergraph K_m^q of order q on m vertices (K_6^2 in Fig. 1a). Any subset of the hyperedges that uses all vertices exactly once is called a 1-*factor*. Restated in these terms, our task is to divide the $\binom{m}{q}$ hyperedges of K_m^q into disjoint 1-factors. This is known in combinatorics as *hypergraph factorization*. In our example (Fig. 1a), we show a factorization of K_6^2 into five disjoint 1-factors, the first of which consists of hyperedges AB, CD, and EF. Hypergraph factorization has been the subject of intense study[27;28]. In particular, a deep result from combinatorics known as Baranyai's

theorem[58] shows that factorizations *exist* under minimal conditions. Moreover, there are also efficient algorithms to construct them up to $q = 3$ under mild conditions, see in particular[59–61]. More generally, and beyond what we need, the efficient construction of hypergraph factorizations is a challenging combinatorial problem.

The mathematical construction of hypergraph factorizations is somewhat sophisticated. However, the output is easy to use. We provide an online tool available at `http://hyper.covid19-analysis.org` that implements constructions for $q = 2$ (as long as m is even) and for $q = 3$ (as long as m is a multiple of 6 and $m - 1$ is a prime number; which covers a wide range of settings). We describe the construction details in Supplementary Methods. Using this, forming the HYPER pools and identifying putative positives becomes both simple and flexible.

Theoretical performance characterization: To analyze the performance of HYPER, we consider a standard statistical model for group testing[17;20–24]. We assume the disease prevalence is p, i.e., each individual is positive independently at random with probability p. In addition, each test has specificity $1 - \alpha$ and sensitivity β. Here we present only the key results; see Supplementary Methods for more details. This setting is different from the COVID-19 model we use in our simulations in the next section that accounts for the features of COVID-19. We emphasize that our method also extends beyond COVID-19, and thus it is valuable to have performance guarantees that capture more general settings.

For this analysis we suppose n is a multiple of m/q, and let $k = nq/m$ be the number of subjects per pool. This is not a significant restriction as m/q is typically a small integer in the examples we are interested in (e.g., in some simulations below we have $m = 24$ with $q = 3$). Denote $r = 1 - p$ and $u = \binom{m-2}{q-2} \cdot \lceil n/\binom{m}{q} \rceil$. Then, we can show that the expected number $\mathbb{E}(T)$ of tests T used by HYPER (including the tests from the second stage) is upper bounded as

$$\mathbb{E}(T) \leq m + n \cdot \left[1 - qp_1 + \binom{q}{2} p_2\right],$$

where

$$p_1 = 1 - \beta + (\beta - \alpha) \cdot r^k$$
$$p_2 = p_1^2 + (\beta - \alpha)^2 \cdot r^{2k-u}(1 - r^u).$$

This inequality becomes an equality when $q = 1, 2$, so the above formula characterizes the expected number of tests used by HYPER (Supplementary Methods). In fact, we derive a more general and sharper version of this upper bound valid for all q, using the Dawson-Sankoff inequality[62;63], which is a non-trivial generalization of the Bonferroni union-intersection inequalities from probability theory.

Using this, when $q = 2$ and we are in the noiseless large n case, i.e., sensitivity and specificity are close to 1 and sample size is large ($\alpha = 0$, $\beta = 1$, and $n \to \infty$), and the prevalence is small, i.e., $p \to 0$, we show that the

optimal number of pools m is approximately

$$m/n \approx 2p^{2/3} - p. \tag{1}$$

The corresponding approximate efficiency is $\mathbb{E}(T)/n \approx 3p^{2/3}$ (including stage two tests; Supplementary Figs. 1a and 1b). This shows that HYPER has a better efficiency than Dorfman designs, which have efficiency $2p^{1/2}$ when the prevalence is small as $p \to 0$[20]. In fact, its efficiency is asymptotically the same as three-stage Dorfman testing[20;56]. While there are more efficient designs available in the literature, they rely either on multiple rounds[8;20] or on taking q much larger[41–49]; each of which is outside of the constraints that we work with. See Supplementary Methods for more discussion.

For HYPER with $q \leq 2$ and $n \leq \binom{m}{q}$, the overall average specificity is $1 - \alpha\gamma$ and the overall average sensitivity is β^{q+1}, where $\gamma = [\beta + (\alpha - \beta) \cdot r^{k-1}]^q$ (Supplementary Fig. 1). The false negative probability (the probability that a sample is positive given that they were declared negative) of HYPER is

$$\frac{1}{1 + o\frac{1-\alpha\gamma}{1-\beta^{q+1}}}. \tag{2}$$

The true positive probability (the probability that a sample is positive given that they were declared positive) of HYPER is

$$\frac{1}{1 + o\frac{\alpha\gamma}{\beta^{q+1}}}. \tag{3}$$

Here $o = (1-p)/p$ is the odds ratio of prevalence. In this noisy group testing model, sensitivity and specificity calculations have been reported in Bilder's work for array designs, see e.g.,[21]; but it is beyond our scope to do a full comparison. Note also that the sensitivity of HYPER under this theoretical model can be improved via error-correction of false negatives in stage one, e.g., by including individuals in only one negative pool among the putative positives[19]. However, doing so can come at the cost of efficiency, and analyzing this tradeoff is beyond our present scope.

To summarize, these results characterize the expected efficiency, sensitivity and specificity under the standard theoretical model where tests have independent errors. This model captures important features of many applications beyond COVID-19 testing, and provides a useful setting to evaluate methods. Here we found that the efficiency of HYPER at low prevalence is competitive with other existing methods of comparable simplicity. Based on the specific application, the analysis can also be used to help guide the selection of HYPER designs that strike the appropriate balance of efficiency, sensitivity and specificity. For example, one might use a small number of splits q in settings where sensitivity loss due to independent test errors is of primary concern. On the other hand, if test errors are negligible, the best efficiency (for $q = 2$ splits) at low prevalence may be obtained by using $m/n \approx 2p^{2/3} - p$ pools per individual. Finally, the formulas for false negative and true positive probabilities help guide the interpretation of negative and positive results, respectively, declared by HYPER.

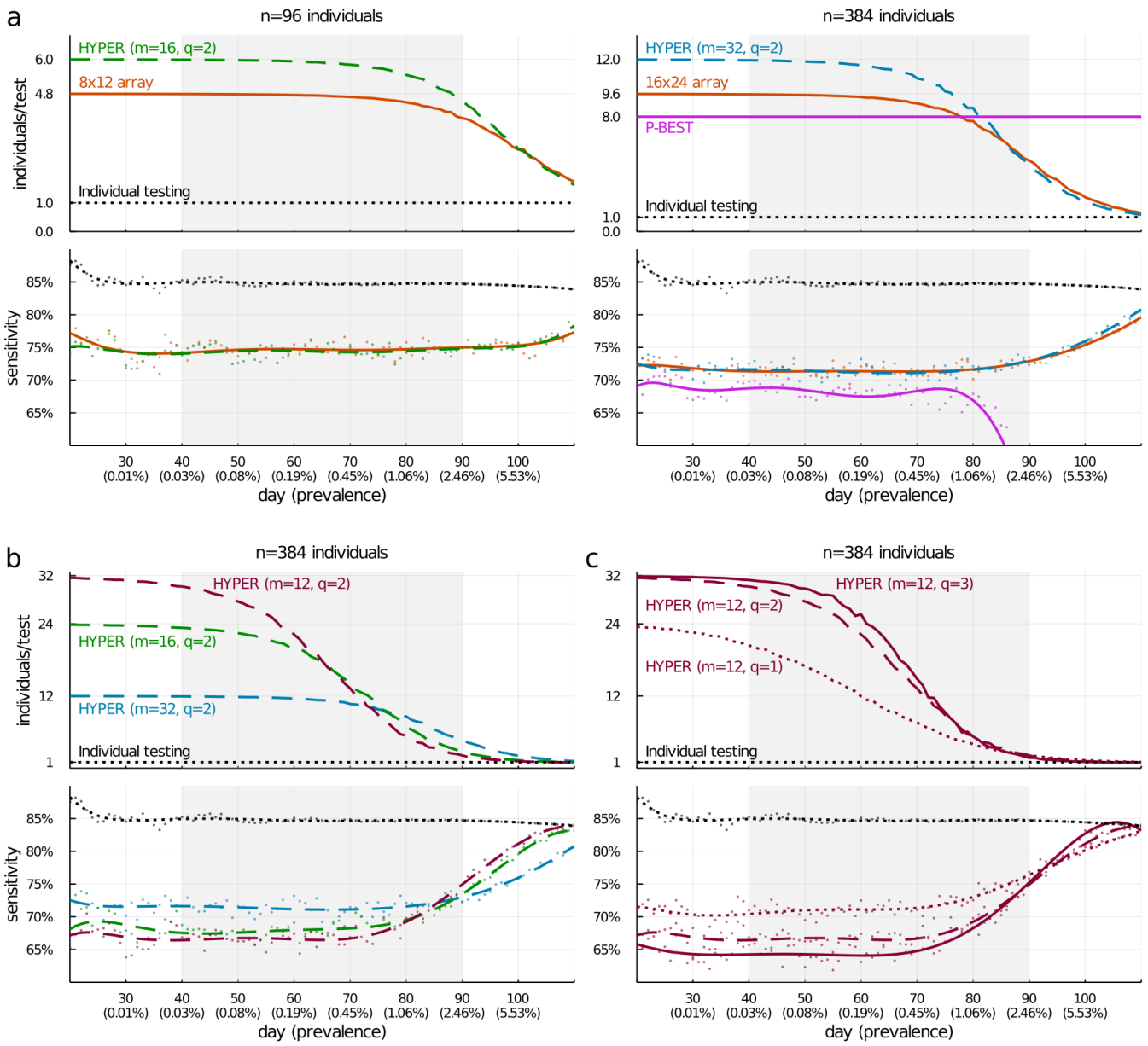


Fig. 2 Efficiency and sensitivity of pooled testing during a simulated epidemic. Average values of efficiency (relative to individual testing) and sensitivity of a variety of pooling designs are shown for each day, with results averaged across 200,000 random trials. For sensitivity, raw averages are shown as dots with degree-8 polynomial fits overlaid as curves; the curves for efficiency depict raw averages. During the days 40–90 (highlighted), the prevalence grows exponentially from 0.03% to 2.46%. **a** Comparison of HYPER with alternative methods that use $n = 96$ individuals per batch (left panels) or $n = 384$ (right panels). The particular HYPER designs were chosen to have the same maximum pool sizes ($nq/m = 12$ for $\mathrm{H}_{96,16,2}$; $nq/m = 24$ for $\mathrm{H}_{384,16,2}$) as the array designs. Sensitivity (**a**, bottom panels) depends heavily on pool sizes, due to dilution of viral loads. **b,c** HYPER evaluated with varying numbers of pools (**b**, $m = 32, 16, 12$) and numbers of splits (**c**, $q = 1, 2, 3$). The designs are affected by the increasing prevalence over time to varying degrees. As prevalence increases, efficiency decreases (as more stage-two tests become necessary), while sensitivity increases (as larger viral loads begin to "rescue" small viral loads that would have been missed).

Performance under a COVID-19 model: We study the performance of HYPER under the viral load based COVID-19 model of Cleary and Hay et al.[19]. We simulate: a) SARS-CoV-2 viral load kinetics in infected individuals; b) the dilution of viral loads during pooling that can lead to false negatives; and c) the evolution of infection prevalence in a large population over time during epidemic growth and decline. We focus here on a window during which the prevalence increases exponentially from 0.03% to 2.46% (days 40–90 in our simulation; we consider other windows below) and individual testing has a sensitivity (i.e., fraction of positive individuals that are identified) of roughly 85% (Fig. 2). We compare HYPER with two existing state-of-the-art methods (Fig. 2a): array designs[7] and P-BEST[6]. These methods use batches of $n = 96$ individuals (8×12 array; left panels) or $n = 384$ individuals (16×24 array and P-BEST; right panels). We consider both efficiency gain with respect to individual testing (number of individuals screened divided by average number of tests used, including stage-two tests) and average sensitivity (fraction of positive individuals identified after completion of all stages) of the methods.

For $n = 96$ (Fig. 2a, left panels), we consider the 8×12 array and a HYPER design, $\mathrm{H}_{96,16,2}$, chosen to dilute samples a similar amount and thus potentially achieve a similar sensitivity. Our simulation shows that both methods indeed have similar sensitivity, which is about 10 percentage points lower than the sensitivity of individual testing (Fig. 2a, bottom-left panel). This is due to dilution of viral loads below the limit of detection during stage-one tests. While the array design is roughly 4.8 times more efficient than individual testing for much of the 50-day window (Fig. 2a, top-left panel), consistent with earlier studies[7], the $\mathrm{H}_{96,16,2}$ design is roughly 6 times more efficient than individual testing. This corresponds to a 25% improvement in efficiency over the comparable array design, while achieving essentially the same sensitivity.

For $n = 384$ (Fig. 2a, right panels), we consider the $\mathrm{H}_{384,16,2}$ design, again chosen to potentially match the sensitivity of the array design (16×24) appropriate for this number of samples. The HYPER design is again roughly 25% more efficient than the array design (Fig. 2a, top-right panel) while having essentially equal sensitivity (Fig. 2a, bottom-right panel). In contrast to these two-stage methods, the one-stage approach of P-BEST has a constant efficiency gain of 8 times individual testing (Fig. 2a, top-right panel), but it significantly loses sensitivity around day 80 as prevalence grows (Fig. 2a, bottom-right panel). This is because the design and decoding algorithm are optimized for a prevalence around 1% and performance degrades beyond this operating point. In contrast, the sensitivity of HYPER increases around day 80 as prevalence increases. Notably, error-correcting does not appear to effectively handle the false negatives that arise here due to diluted viral loads falling below the limit of detection. P-BEST is generally the least sensitive among the three pooling strategies.

Finally, we compare various choices for the number of pools (Fig. 2b, $m = 32, 16, 12$) and the number of splits (Fig. 2c, $q = 1, 2, 3$) in HYPER. The behavior is generally consistent with the findings of Cleary and Hay et al.[19], which studied the performance of random designs *on average*. However, note that any given draw of the random designs may perform better or worse based only on chance. Here we demonstrate the same effects

and overall performance with fixed *deterministic* designs. Specifically, the efficiency gain of HYPER is roughly n/m early in the epidemic, during which low prevalence leads to pools frequently testing negative. Decreasing the number of pools m can significantly increase efficiency, but comes with a slight reduction in sensitivity. As the epidemic progresses, rising prevalence leads to increasingly many stage-two tests, though designs with larger q are more robust to this loss of efficiency. At the same time, larger q designs tend to be less sensitive. Overall, more efficient designs tend to be less sensitive, creating a trade-off that depends significantly on prevalence. One size does not fit all, underscoring the benefit of flexible designs like HYPER that can easily be adapted from one environment to another.

Choosing a pooling method given resource constraints: In practice, decision makers must often choose a pooling method given limited capacity (on average) for daily testing and sample collection. One approach is to maximize the *number of individuals screened per day*, i.e., the number of individuals n per batch times the number of batches b that can be run per day, given the limited testing and collection capacities. This metric accounts for the impact of resource constraints. However, it does not represent the *actual* number of infected individuals that the population screen can identify. A very efficient method can screen numerous individuals but may miss all the infected ones if it lacks sufficient sensitivity. Hence, a more natural goal is to choose a design that maximizes the number of individuals screened times the average sensitivity, which we call the *effective number of individuals screened per day* or simply the "effective screening capacity" for short. Note that, scaling this by prevalence measures the actual number of infected individuals that are identified by the screen. We evaluated the overall effective screening capacity for the epidemic window considered above (Fig. 2). We compared HYPER (across a sweep of n, m and q; Supplementary Table 1) with individual testing, array designs[7], and P-BEST[6], all across a range of resource constraints.

We will first consider a few specific scenarios to understand the tradeoffs with each design, then consider a larger grid to get a picture of the overall trends. We begin with a testing-scarce setting (Fig. 3a) with an average testing capacity of 12 tests per day being far outstripped by an average sample collection capacity of 3072 samples per day. In this case, individual testing only screens 12 individuals and achieves an effective screening capacity of 10.2 individuals (the average sensitivity of individual testing is 84.8%, due to the presence of positive individuals with viral load below the limit of detection). The best HYPER design (maximizing the effective screening capacity, given the constraints), obtains an effective screening capacity of 122.2 individuals, roughly 12 times more than individual testing. It does so by pooling $n = 192$ individuals per batch into $q = 2$ of $m = 6$ pools with an average of $b \approx 0.9$ batches run per day (recall that some of the testing capacity is needed for stage-two tests). Both array designs and P-BEST use more than 12 tests in a single run so do not appear here.

As testing capacity grows to 24 then 48 tests (Figs. 3b and 3c) with sample capacity unchanged, larger

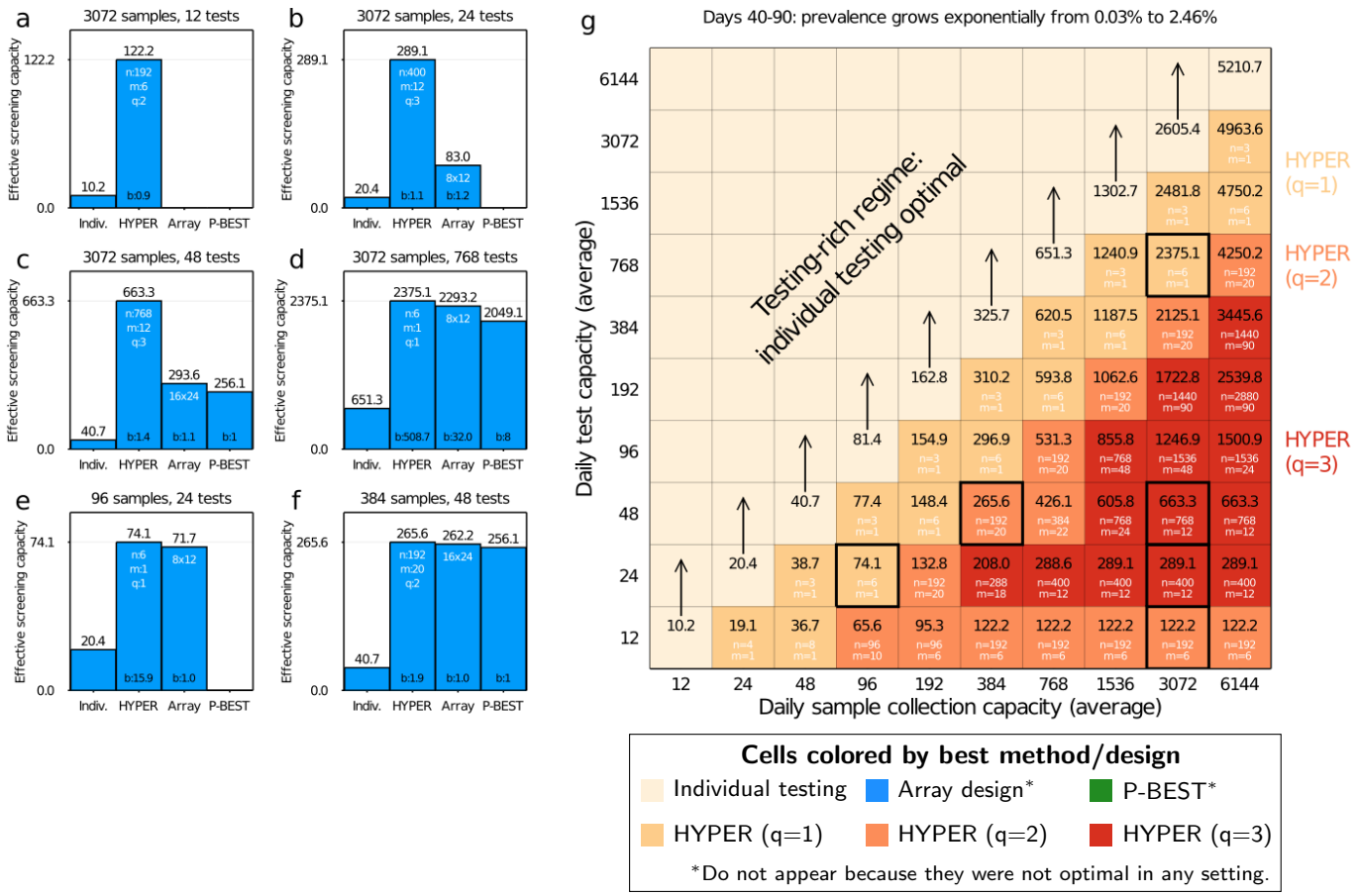


Fig. 3 Comparison of pooling methods under resource constraints. HYPER designs (Supplementary Table 1) were evaluated together with individual testing, array designs[7], and P-BEST[6], across a range of sample collection and testing capacities. The basis for comparison was the average effective number of individuals screened per day across days 40-90 of the simulation (Fig. 2), during which the prevalence increases exponentially from 0.03% to 2.46%. Bar plots on the left depict the effective screening capacities (bar height) in a testing-scarce setting (**a**), followed by increasingly testing-rich settings (**b-d**), and then settings well-suited for array designs and P-BEST (**e,f**) that have capacities matching their design parameters. When multiple designs for a given method were available within the constraints (i.e., various choices of HYPER designs, or a choice between the two 8×12 and 16×24 array designs), we use the most effective configuration and indicate it in white text within the appropriate bar. The average number of batches run per day is noted at the bottom of each bar. **g** Expanded comparison to a grid of sampling and testing budgets. Each cell is colored by the best method (where we separately identify HYPER designs with $q = 1, 2, 3$ splits in shades of orange / red), and shows the corresponding effective screening capacities (in black text). The best design configuration is written in white text. For HYPER, we write the number of individuals per batch n and the number of pools m for the best configuration; cell color already indicates the number of splits q. Note that n and m often do not match the daily sampling and testing budgets, respectively, since multiple batches can be run per day. The cases from (**a-f**) are outlined in black. See Supplementary Fig. 2 for additional details.

effective screening capacities become possible by using larger designs, including the 8×12 array followed by the 16×24 array and P-BEST. HYPER adapts to these settings as well, and the larger designs here are accompanied with a larger number of splits q. HYPER remains the most effective overall. For a testing capacity of 24 daily tests, HYPER achieves 289.1 effective individuals screened, which is over 14 times as many as individual testing (20.4 effective individuals screened) and over 3.4 times as many as the array designs (83.0 effective individuals screened). P-BEST cannot be run since it uses more tests than available, so it achieves zero effective individuals screened. For a testing capacity of 48 daily tests, HYPER achieves 663.3 effective individuals screened, which is over 16 times that of individual testing (40.7 effective individuals screened), over 2.2 times that of the array designs (293.6 effective individuals screened), and over 2.5 times that of P-BEST (256.1 effective individuals screened).

When the capacity grows to 768 tests per day (Fig. 3d), i.e., a quarter the sample collection capacity, the pooled designs remain more effective than individual testing, but now by less than 4 times. In this increasingly testing-rich regime, P-BEST and array designs are sample-constrained and under-utilize testing resources. P-BEST uses only $mb = 384$ of the 768 available tests, since all 3072 available samples are tested after $b = 8$ batches of $n = 384$ samples. The same is true for the 16×24 array design, although additional tests will also be used in stage two. The most effective HYPER design $\mathrm{H}_{6,1,1}$ corresponds to simple Dorfman testing, uses roughly 508 tests in the first stage, and achieves 2375.1 effective individuals screened.

Finally, we considered two settings well-tailored for the array designs and P-BEST: 96 samples with 24 tests (Fig. 3e) for which the 8×12 array is well-suited (recall that extra testing capacity is needed for stage 2), and 384 samples with 48 tests (Fig. 3f) for which the 16×24 array and P-BEST are well-suited. In particular, these are settings where the number of available individuals and pools closely match (a multiple of) the number of individuals and pools used by these designs in a single batch. This can help them maximally utilize both the testing and sample collection capacities, i.e., neither resource is under-utilized. The array designs and P-BEST performed similarly to HYPER in these favorable cases, but notably HYPER remained slightly more effective: 74.1 vs. 71.1 effective individuals screened for the first scenario, and 265.5 vs. 262.2 effective individuals screened for the second.

Expanding this analysis to a grid of sampling and testing capacities gives a broad view of overall trends. We consider a sweep with each ranging from 12 to 6144 (Fig. 3g). Note first that for all sample collection capacities, the effective screening capacity grows as testing capacity scales up, until testing capacity matches or outpaces sample collection capacity. Individually testing all samples collected is optimal from that point on. Likewise, for any given testing capacity, the effective screening capacity rises as sample collection capacity grows, eventually reaching an upper limit at which testing capacity becomes the limiting factor. Pooled testing increases this upper limit, enabling an effective number of individuals screened far beyond the actual number of available tests. For example, for a testing capacity of 96 tests per day, pooled testing achieves an effective

screening capacity of up to 1500.9 individuals per day, which is over 18 times the effective screening capacity of 81.4 individuals per day achievable by individual testing.

Across the testing-constrained regime, i.e., testing capacity below sample collection capacity, HYPER outperformed both P-BEST and the array designs (Fig. 3 and Supplementary Fig. 2). As observed before (Figs. 3a to 3f), the flexibility of HYPER generally makes it more readily tailored and optimized for each setting. Dorfman testing (i.e., HYPER with $q = 1$) is most effective when testing capacity is within a quarter of the sample capacity. However, as sample capacity begins to further outstrip testing capacity, combinatorial designs that involve more individuals n per batch and that use more splits q become most effective, consistent with earlier studies of analogous random designs[19].

The above results capture the average effectiveness of each method when deployed over a 50-day window during epidemic spread. We next investigated effectiveness on individual days, each corresponding to a different fixed prevalence. At low to moderate prevalence of 0.1% (Supplementary Fig. 3), 1.06% (Supplementary Fig. 4), or 1.36% (Supplementary Fig. 5), HYPER is consistently the most effective strategy across all settings. In an intermediate range with prevalence of 1.48% (Supplementary Fig. 6) and 2.46% (Supplementary Fig. 7) there is a subset of scenarios in which P-BEST outperforms HYPER, although the performance of each method is nearly equivalent in these settings. Outside these settings P-BEST is either not viable or substantially underperforms HYPER. At a higher prevalence of 3.15% (Supplementary Fig. 8) HYPER again performs best across all scenarios. We did not observe any scenarios in which array designs were most effective.

Discussion

Our results demonstrate the effectiveness of a new family of pooling designs that are adaptable to any number of samples, with only mild conditions on the number of pools, while remaining maximally balanced in three senses (number of assignments per individual, pool, and combination of pools). This flexibility is critical to selecting appropriate designs under the widely varying global demands and capabilities for SARS-CoV-2 testing. In addition, the balanced nature of the designs ensures equitable treatment of samples and facilitates robust and simple implementation. Despite the simplicity of implementing HYPER, the existence and construction of the designs relies on deep mathematical results, such as Baranyai's theorem from combinatorics[58] and Beth's construction from number theory[59;60].

Our evaluation of HYPER in both a general theoretical framework and a SARS-CoV-2 specific simulation can be used to guide the choice of design, depending on the setting and purpose of testing. For our theoretical setup, where each test has specificity $1 - \alpha$ and sensitivity β independent of all other tests, we showed that using roughly $m/n \approx 2p^{2/3} - p$ pools per individual maximizes the efficiency of HYPER designs, and we derived that HYPER has sensitivity β^q. When selecting a design for SARS-CoV-2 testing, one must also account for the impact of dilution in pooled tests (leading to false negatives), among other effects like the evolution of viral

loads over time. These considerations are important for selecting designs that remain effective during epidemic spread, which is critical in the ongoing pandemic and in preparations for the future. Here, insight can be gained from our results on a realistic COVID-19 simulation[19]. For example, the sensitivity of HYPER now depends on both the number of splits q and the number of pools m. Designs with fewer pools have larger pools, leading to more dilution and lower sensitivity. Since we also do not correct for false negatives, the sensitivity is lower than individual testing. In theory, adding error correction could increase sensitivity, but doing so is most effective when false negative PCR results are independent across tests. When most false negative results come from diluting positive samples below the LOD (as in our simulations), then error correction is not as effective (as we observed with P-BEST) because the failures are not independent. The sensitivity of HYPER is generally best when using designs with fewer splits q or more pools m. However, doing so generally reduces efficiency at low prevalence. The results illustrate a general trade-off between sensitivity and efficiency that must be balanced depending on the setting.

While pooled testing can substantially increase effectiveness depending on laboratory capacity and prevalence, it is important to consider the added logistical challenges. Notably, the gains in testing effectiveness that we demonstrate above do not account for the additional pipetting steps during pooling, or the logistical cost of temporarily storing and retrieving samples for stage two testing. However, simple (Dorfman) pooling designs are receiving increasing interest[3–5;14;18;64] for real-world testing, demonstrating that these logistical challenges can be overcome in practice in a variety of settings. In comparison to Dorfman designs, more complex designs (with $q > 1$) will require up to q times as many pipetting steps during stage one pooling. Depending on the relative timing and cost of each step in the protocol, this may shift the relative favorability of the strategies considered above. In particular, P-BEST, with $q = 6$ or more, may become relatively unfavorable if pooling steps are expensive, while array designs, which utilize multichannel pipettes, may become more favorable. We note that we have previously validated a HYPER design ($H_{96,6,2}$)[19] in the laboratory, and found that with an unoptimized workflow, stage one pooling of 192 samples could be completed manually in under 90 minutes. It is therefore especially likely that gains in effectiveness may outweigh additional logistical costs in settings that are constrained by testing throughput (e.g., due to limited reagents or equipment) but have an excess of laboratory technician capacity.

Although the alternative pooling designs we evaluated here were originally published as fixed designs, potentially optimized for a fixed prevalence (P-BEST), it is possible they could be adapted to different settings. Array designs could potentially be adapted to any grid that fits on standard (96-well or 384-well) laboratory plates. However, our results suggest that HYPER designs could in general match the sensitivity of array designs while having greater efficiency. Indeed, array designs had sub-optimal effectiveness in every setting we considered, including those tailored to best fit each array. P-BEST can in principle be adapted to different settings, although it is unclear how flexible it can be while retaining balance, and properly doing so

requires some expertise and experience. To use it, one might need to not only generate another Reed-Solomon code but also properly retune the decoder algorithm. Here we used the design and decoder provided online (`https://github.com/NoamShental/PBEST`) by the authors. We have so far been unable to evaluate Hypercube[8] in simulation since the code for it is currently unavailable.

So far we have limited HYPER to $q \leq 3$. This has the advantage of reducing the additional logistical burden (and potential for error) that comes with splitting samples into more pools. Moreover, the efficient construction of hypergraph factorizations is highly nontrivial for $q > 3$. However, higher q can have several advantages. For example, individuals in the same hyperedge (i.e., assigned to the same combination of pools) are identified as putative positives together as a block even if only one of them is actually positive. Using a higher q can significantly increase the number of hyperedges $\binom{m}{q}$, reducing the number of individuals sharing a single hyperedge. Results for HYPER here also indicated that high q designs can be highly effective when the sample capacity significantly outstrips testing capacity, consistent with earlier studies of analogous random designs[19]. Likewise, greater efficiency can be obtained by using a multi-stage approach with more than two stages, which is also more logistically challenging. In practice, one must weigh these opportunities for greater effectiveness against the increased complexity. Such designs may be especially promising for labs with access to robotic pipetting platforms.

To conclude, we present a simple, efficient and flexible pooled testing strategy that can be easily tailored and implemented without specialized expertise or equipment. To further facilitate implementation, we provide an online tool available at `http://hyper.covid19-analysis.org` that makes it easy to generate and carry out designs for a broad range of settings.

References

[1] Donoho, D., Lotfi, M. & Ozturkler, B. The mathematics of mass testing for COVID-19. `http://sinews.siam.org/Details-Page/the-mathematics-of-mass-testing-for-covid-19` (2020).

[2] Nalbantoglu, O. U. & Gundogdu, A. COVID-19 pandemic: Group testing. *Frontiers in Medicine* **7** (2020).

[3] Abdalhamid, B. *et al.* Assessment of specimen pooling to conserve SARS CoV-2 testing resources. *American Journal of Clinical Pathology* **153**, 715–718 (2020).

[4] Bilder, C. R., Iwen, P. C. & Abdalhamid, B. Pool size selection when testing for severe acute respiratory syndrome coronavirus 2. *Clinical Infectious Diseases* (2020).

[5] Hogan, C. A., Sahoo, M. K. & Pinsky, B. A. Sample pooling as a strategy to detect community transmission of SARS-CoV-2. *JAMA* **323**, 1967–1969 (2020).

[6] Shental, N. *et al.* Efficient high-throughput SARS-CoV-2 testing to detect asymptomatic carriers. *Science Advances* **6** (2020).

[7] Sinnott-Armstrong, N., Klein, D. & Hickey, B. Evaluation of group testing for SARS-CoV-2 RNA. Preprint at `http://www.medrxiv.org/content/10.1101/2020.03.27.20043968v1` (2020).

[8] Mutesa, L. *et al.* A pooled testing strategy for identifying SARS-CoV-2 at low prevalence. *Nature* **589**, 276–280 (2021).

[9] Ghosh, S. *et al.* Tapestry: A single-round smart pooling technique for COVID-19 testing. Preprint at `http://www.medrxiv.org/content/10.1101/2020.04.23.20077727v2` (2020).

[10] Ghosh, S. *et al.* A compressed sensing approach to group-testing for COVID-19 detection. Preprint at `http://arxiv.org/abs/2005.07895v1` (2020).

[11] Rajwade, A. & Gopalkrishnan, M. A compressed sensing approach to group-testing for COVID-19 detection. `http://signalprocessingsociety.org/newsletter/2020/07/compressed-sensing-approach-group-testing-covid-19-detection` (2020).

[12] Yi, J., Mudumbai, R. & Xu, W. Low-cost and high-throughput testing of COVID-19 viruses and antibodies via compressed sensing: System concepts and computational experiments. Preprint at `http://arxiv.org/abs/2004.05759v1` (2020).

[13] Verdun, C. M. *et al.* Group testing for SARS-CoV-2 allows for up to 10-fold efficiency increase across realistic scenarios and testing strategies. Preprint at `http://www.medrxiv.org/content/10.1101/2020.04.30.20085290v2` (2020).

[14] Lohse, S. *et al.* Pooling of samples for testing for SARS-CoV-2 in asymptomatic people. *The Lancet Infectious Diseases* **20**, 1231–1232 (2020).

[15] Yelin, I. *et al.* Evaluation of COVID-19 RT-qPCR test in multi sample pools. *Clinical Infectious Diseases* **71**, 2073–2078 (2020).

[16] Cleary, B. & Regev, A. The necessity and power of random, under-sampled experiments in biology. Preprint at `http://arxiv.org/abs/2012.12961v1` (2020).

[17] Dorfman, R. The detection of defective members of large populations. *Annals of Mathematical Statistics* **14**, 436–440 (1943).

[18] Barak, N. *et al.* Lessons from applied large-scale pooling of 133,816 SARS-CoV-2 RT-PCR tests. Preprint at `http://www.medrxiv.org/content/10.1101/2020.10.16.20213405v1` (2020).

[19] Cleary, B. *et al.* Using viral load and epidemic dynamics to optimize pooled testing in resource constrained settings. Preprint at `http://www.medrxiv.org/content/10.1101/2020.05.01.20086801v3` (2020).

[20] Finucan, H. M. The blood testing problem. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **13**, 43–50 (1964).

[21] Bilder, C. R. Group testing for identification. In *Wiley StatsRef: Statistics Reference Online* (American Cancer Society, 2019).

[22] Bilder, C. R. Group testing for estimation. In *Wiley StatsRef: Statistics Reference Online* (American Cancer Society, 2019).

[23] Bilder, C. R., Iwen, P. C., Abdalhamid, B., Tebbs, J. M. & McMahan, C. S. Tests in short supply? Try group testing. *Significance* **17**, 15–16 (2020).

[24] Johnson, N. L., Kotz, S. & Wu, X.-Z. *Inspection Errors for Attributes in Quality Control* (Chapman and Hall/CRC, New York, 1991).

[25] Du, D.-Z. & Hwang, F. K. *Combinatorial Group Testing and Its Applications* (World Scientific, Singapore, 2000), 2nd edn.

[26] Du, D.-Z. & Hwang, F. K. *Pooling Designs and Nonadaptive Group Testing* (World Scientific, Singapore, 2006).

[27] Beth, T., Jungnickel, D. & Lenz, H. *Design Theory: Volume 1* (Cambridge University Press, Cambridge, 1999).

[28] Beth, T., Jungnickel, D. & Lenz, H. *Design Theory: Volume 2* (Cambridge University Press, Cambridge, 1999).

[29] Colbourn, C. J. & Dinitz, J. H. *Handbook of Combinatorial Designs* (Chapman and Hall/CRC, Boca Raton, FL, 2006), 2nd edn.

[30] Aldridge, M., Johnson, O. & Scarlett, J. Group testing: an information theory perspective. Preprint at `http://arxiv.org/abs/1902.06002v2` (2019).

[31] Csóka, E. Application-oriented mathematical algorithms for group testing. Preprint at `http://arxiv.org/abs/2005.02388v1` (2020).

[32] Erlich, Y., Gordon, A., Brand, M., Hannon, G. J. & Mitra, P. P. Compressed genotyping. *IEEE Transactions on Information Theory* **56**, 706–723 (2010).

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

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

[35] Chan, C. L., Jaggi, S., Saligrama, V. & Agnihotri, S. Non-adaptive group testing: Explicit bounds and novel algorithms. *IEEE Transactions on Information Theory* **60**, 3019–3035 (2014).

[36] Gandikota, V., Grigorescu, E., Jaggi, S. & Zhou, S. Nearly optimal sparse group testing. *IEEE Transactions on Information Theory* **65**, 2760–2773 (2019).

[37] Mazumdar, A. Nonadaptive group testing with random set of defectives. *IEEE Transactions on Information Theory* **62**, 7522–7531 (2016).

[38] D'yachkov, A. G., Vorobyev, I. V., Polyanskii, N. A. & Shchukin, V. Y. On a hypergraph approach to multistage group testing problems. In *2016 IEEE International Symposium on Information Theory (ISIT)*, 1183–1191 (IEEE, 2016).

[39] Bilder, C. R., Zhang, B., Schaarschmidt, F. & Tebbs, J. M. binGroup: A package for group testing. *The R Journal* **2**, 56–60 (2010).

[40] Hitt, B., Bilder, C., Tebbs, J. & McMahan, C. A shiny app for pooled testing. `http://chrisbilder.com/shiny`.

[41] Berger, T. & Levenshtein, V. I. Asymptotic efficiency of two-stage disjunctive testing. *IEEE Transactions on Information Theory* **48**, 1741–1749 (2002).

[42] Levenshtein, V. I. A universal bound for a covering in regular posets and its application to pool testing. *Discrete Mathematics* **266**, 293–309 (2003).

[43] Bonis, A. D., Gasieniec, L. & Vaccaro, U. Optimal two-stage algorithms for group testing problems. *SIAM Journal on Computing* **34**, 1253–1270 (2005).

[44] Mezard, M. & Toninelli, C. Group testing with random pools: optimal two-stage algorithms. Preprint at `http://arxiv.org/abs/0706.3104v1` (2007).

[45] Mézard, M. & Toninelli, C. Group testing with random pools: Optimal two-stage algorithms. *IEEE Transactions on Information Theory* **57**, 1736–1745 (2011).

[46] Damaschke, P. & Muhammad, A. S. Randomized group testing both query-optimal and minimal adaptive. In *International Conference on Current Trends in Theory and Practice of Computer Science*, 214–225 (Springer, 2012).

[47] Coja-Oghlan, A., Gebhard, O., Hahn-Klimroth, M. & Loick, P. Optimal group testing. In *Proceedings of Thirty Third Conference on Learning Theory*, vol. 125, 1374–1388 (PMLR, 2020).

[48] Gebhard, O., Johnson, O., Loick, P. & Rolvien, M. Improved bounds for noisy group testing with constant tests per item. Preprint at `http://arxiv.org/abs/2007.01376` (2020).

[49] Scarlett, J. An efficient algorithm for capacity-approaching noisy adaptive group testing. In *2019 IEEE International Symposium on Information Theory (ISIT)*, 2679–2683 (IEEE, 2019).

[50] McMahan, C. S., Tebbs, J. M. & Bilder, C. R. Informative dorfman screening. *Biometrics* **68**, 287–296 (2012).

[51] Phatarfod, R. M. & Sudbury, A. The use of a square array scheme in blood testing. *Statistics in Medicine* **13**, 2337–2343 (1994).

[52] Kim, H.-Y., Hudgens, M. G., Dreyfuss, J. M., Westreich, D. J. & Pilcher, C. D. Comparison of group testing algorithms for case identification in the presence of test error. *Biometrics* **63**, 1152–1163 (2007).

[53] Kim, H.-Y. & Hudgens, M. G. Three-dimensional array-based group testing algorithms. *Biometrics* **65**, 903–910 (2009).

[54] Black, M. S., Bilder, C. R. & Tebbs, J. M. Optimal retesting configurations for hierarchical group testing. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **64**, 693–710 (2015).

[55] Lin, Y.-J., Yu, C.-H., Liu, T.-H., Chang, C.-S. & Chen, W.-T. Comparisons of pooling matrices for pooled testing of COVID-19. Preprint at `http://arxiv.org/abs/2010.00060v1` (2020).

[56] Broder, A. Z. & Kumar, R. A note on double pooling tests. Preprint at `http://arxiv.org/abs/2004.01684v1` (2020).

[57] Shental, N., Amir, A. & Zuk, O. Identification of rare alleles and their carriers using compressed se(que)nsing. *Nucleic Acids Research* **38** (2010).

[58] Baranyai, Z. On the factorization of the complete uniform hypergraph. In *Infinite and Finite Sets, Proc. Coll. Keszthely, 1973*, 91–108 (1975).

[59] Beth, T. Algebraische auflösungsalgorithmen für einige unendliche familien von 3–designs. *Le Matematiche* **29**, 105–135 (1974).

[60] Beth, T. *On resolutions of Steiner systems.* Arbeitsberichte des Instituts für Mathematische Maschinen und Datenverarbeitung, Informatik (Inst. für Math. Maschinen u. Datenverarbeitung (Informatik), Friedrich-Alexander-Univ. Erlangen-Nürnberg, 1978).

[61] Tamm, U. Applications of Baranyai's theorem in information theory. In *Proceedings of 6th Benelux – Japan Workshop on Coding and Information Theory, Essen, 1996* (1996).

[62] Dawson, D. A. & Sankoff, D. An inequality for probabilities. *Proceedings of the American Mathematical Society* **18**, 504–507 (1967).

[63] Galambos, J. Bonferroni inequalities. *Annals of Probability* **5**, 577–581 (1977).

[64] Seifried, E. & Ciesek, S. Pool testing of SARS-CoV-02 samples increases worldwide test capacities many times over. `http://aktuelles.uni-frankfurt.de/englisch/pool-testing-of-sars-cov-02-samples-increases-worldwide-test-capacities-many-times-over/` (2020).

Acknowledgements: D.H. was supported by the Dean's Fund for Postdoctoral Research of the Wharton School and NSF BIGDATA grant IIS 1837992. R.D. and X.L. were supported by a grant from the Partners in Health. E.D. was supported in part by NSF BIGDATA grant IIS 1837992.

Author contributions: All authors contributed to the design of the study and to discussions of all aspects. D.H., R.D., and E.D. performed the theory development and analysis. D.H., X.L., and B.C. performed and analyzed the simulations under the COVID-19 model. R.D. and X.L. developed the interactive online tool. D.H., B.C., and E.D. wrote the first draft of the manuscript. All authors reviewed and edited the manuscript.

Competing interests: None.

Supplementary Methods

Simulation under a COVID-19 model

We performed simulations studies using the COVID-19 model of Cleary and Hay et al.[19]. The model first simulates viral loads for a large population of 12,500,000 individuals across 357 days during which the epidemic grows then declines. It captures the evolution of both: a) viral loads within each individual, i.e., within-host viral kinetics, and b) infection prevalence in the overall population. See[19] for a detailed description.

Next, the model simulates pooled testing to determine the average efficiency gain (with respect to individual testing) and average sensitivity for each day. For the reader's benefit, we detail the process here. For HYPER designs, i.e., $\mathrm{H}_{n,m,q}$, the simulation proceeds for each trial r of day d as follows:

1. Draw n individuals uniformly at random from the population. Let $z_1, \ldots, z_n$ be their viral loads that day.

2. Generate the *sampled* viral load for each of the m pools $\mathcal{I}_1, \ldots, \mathcal{I}_m \subseteq \{1, \ldots, n\}$ as follows:

$$v_j = \sum_{i \in \mathcal{I}_j} \mathrm{Poisson}(z_i / |\mathcal{I}_j|), \qquad j = 1, \ldots, m,$$

   where $|\mathcal{I}_j|$ is the size of pool j, i.e., the number of individuals assigned to it.

3. Compute stage-one pooled testing results:

   - if $v_j >$ LOD then pool j tests positive, where the LOD (limit of detection) we use is 100.
   - otherwise, pool j tests negative with probability 0.99 (i.e., the false positive rate of PCR results is 1%).

4. Select putative positives as those individuals that are not in any negative pools.

5. Compute stage-two individual testing results for the putative positives: putative positive individual j tests positive if $z_j >$ LOD and tests negative otherwise.

6. Declare individuals identified by HYPER as those that tested positive in stage-two.

7. Record the following for the current trial r and day d:

   - the number of true positive individuals identified by HYPER: $n_{\mathrm{iden}}^{(r)}(d)$,
   - the number of tests expended: $T^{(r)}(d) = m +$ number of tests used in stage-two.
   - the number of true positive individuals seen: $n_{\mathrm{pos}}^{(r)}(d) =$ number of individuals with viral load > 0,

For each day, we repeat this for 500 initial trials, then continue until either at least 2,500 true positive individuals have been seen or a total of 200,000 trials have elapsed (including the initial 500). This is to reduce experimental

noise. Denoting R to be the total number of trials run, we then compute the following averages across trials

$$\bar{T}(d) = \frac{1}{R}\sum_{r=1}^{R} T^{(r)}(d), \qquad \bar{n}_{\text{iden}}(d) = \frac{1}{R}\sum_{r=1}^{R} n_{\text{iden}}^{(r)}(d), \qquad \bar{n}_{\text{pos}}(d) = \frac{1}{R}\sum_{r=1}^{R} n_{\text{pos}}^{(r)}(d),$$

then finally compute the average efficiency gain and average sensitivity for day d as follows:

$$\overline{\text{efficiency gain}}(d) = n/\bar{T}(d), \qquad \overline{\text{sensitivity}}(d) = \bar{n}_{\text{iden}}(d)/\bar{n}_{\text{pos}}(d).$$

Note that step 2 in the simulation above captures dilution due to pooling, since each individual's viral load gets divided by the pool size. The Poisson random variable models a small volume being sampled from each swab. Note also that step 5 models the individual testing of stage two as having no false positives. Doing so simplifies the simulation without meaningfully affecting our conclusions (e.g., the optimal pooling designs, which do not depend substantially on stage two specificity). We do include false positives in stage one, since the overall efficiency depends on the specificity there.

For the 8×12 and 16×24 array designs[7], the simulation proceeds in the same way except for step 2, where the corresponding array pools are used instead. Recall that the array method is a two-stage method like HYPER. For P-BEST[6], which is a one-stage method, steps 1-3 are the same (except that step 2 now uses the P-BEST pools). Steps 4-6 are replaced by running the P-BEST decoder to identify individuals. For this, we followed the example (including its tuning parameters) provided online by the authors at `https://github.com/NoamShental/PBEST/blob/f7ffebe6c7021ee40167239210806c5a1319f81e/mFiles/example_PBEST.m`. Finally, since P-BEST has no second stage of validation tests, the number of tests expended is always $T^{(r)}(d) = m = 48$.

Fig. 2 plots the average efficiency gains and average sensitivities of the various methods for each day in a 90-day window of epidemic growth. Here we included individual testing, which has a constant average efficiency gain of 1 (unity) since it is the baseline. Its average sensitivity on day d is equal to

$$\overline{\text{sensitivity}}(d) = \frac{\text{Number of individuals (on that day) with viral load} > \text{LOD}}{\text{Number of individuals (on that day) with viral load} > 0},$$

since individual testing identifies those individuals with viral load $>$ LOD, and true positive individuals are those with viral load > 0 (as before). The average sensitivities of the various methods appeared to generally have significant experimental noise. So, Fig. 2 plots the raw averages (i.e., $\overline{\text{sensitivity}}(d)$) as dots along with a degree-8 polynomial curve fitted to $\overline{\text{sensitivity}}(d)$ v.s. $\log_{10} p(d)$ across the plotting window of days $d = 20, \ldots, 110$, where $p(d)$ is the prevalence on day d.

In Fig. 2a, we compared HYPER designs $\text{H}_{96,16,2}$ and $\text{H}_{384,32,2}$ with their counterpart array designs and P-BEST. For the HYPER designs, the numbers n of individuals per batch were chosen to match the array designs and P-BEST. The numbers m of pools were chosen so that the corresponding pool sizes nq/m match the maximum pool sizes of the array designs (12 for the 8×12 array and 24 for the 16×24 array). Fig. 2b compares HYPER designs $\text{H}_{384,32,2}$, $\text{H}_{384,16,2}$, and $\text{H}_{384,12,2}$ that have varying numbers of pools. Fig. 2c compares HYPER designs $\text{H}_{384,12,1}$, $\text{H}_{384,12,2}$, and $\text{H}_{384,12,3}$ that have varying numbers of splits.

Comparison of pooling methods under resource constraints

We used the simulations above to evaluate the various methods (individual testing, HYPER, array designs, P-BEST) under resource constraints and over time. We considered two forms of resource constraints: i) a limited daily sample collection budget, and ii) a limited daily testing budget. We let both range from 12 to 6144, forming the grid of resource-constrained scenarios shown in Fig. 3g and Supplementary Fig. 2, with a few selected scenarios highlighted in Figs. 3a to 3f. These figures evaluate average performance of the various methods when deployed across days 40-90 of the simulation. Supplementary Figs. 3 to 8 repeat the analysis (using the same set of scenarios) for individual days, namely days 53, 80, 83, 84, 90, and 93. Hence, we will focus on describing Fig. 3 and Supplementary Fig. 2; Supplementary Figs. 3 to 8 are similar.

In each scenario, we evaluated each method by its *effective screening capacity*. As discussed in the main text, this performance metric measures how many individuals the method can screen under the resource constraints, with a correction applied to account for the associated sensitivity. To measure performance over time, we also consider averaging across a chosen set of days $\mathcal{D}$. Fig. 3 and Supplementary Fig. 2 consider days 40-90, so $\mathcal{D} = \{40, \ldots, 90\}$ there. Supplementary Figs. 3 to 8 examine individual days, which corresponds, e.g., to $\mathcal{D} = \{53\}$ in Supplementary Fig. 3. To compute average effective screening capacity, we first determine the number of batches $b(d)$ that can be run on each day d, and its corresponding average $\bar{b}$:

$$b(d) = \min\left\{\frac{\text{sample collection budget}}{n}, \frac{\text{testing budget}}{\bar{T}(d)}\right\}, \qquad \bar{b} = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} b(d).$$

If $\bar{b} < 0.9$ batches per day, i.e., fewer than 0.9 batches can be run per day on average, then the method is considered infeasible within the resource constraints. Then we set the method to have an average effective screening capacity of $\bar{C} = 0$. Setting the above threshold at 0.9 captures an assumed flexibility to use fewer or more tests across days. Otherwise, if $\bar{b} \geq 0.9$, we compute the effective screening capacity $C(d)$ for each day d and its corresponding average $\bar{C}$ as follows:

$$C(d) = \underbrace{n \times b(d)}_{\#\text{ individuals screened}} \times \overline{\text{sensitivity}}(d), \qquad \bar{C} = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} C(d).$$

Figs. 3a to 3f show the average effective capacities for the considered methods as bars, with the corresponding average number of batches noted at the bottom of each bar. Multiple configurations are available for both the array method (the 8×12 and 16×24 array designs) and HYPER (various choices of n, m and q). For these methods, we select the most effective among all configurations, i.e., the configuration with the highest average effective screening capacity $\bar{C}$. For HYPER, in particular, we optimized over the configurations listed in Supplementary Table 1. The chosen configuration is noted at the top of each bar in Figs. 3a to 3f.

Supplementary Fig. 2 shows the bar graphs for the full range of resource-constrained scenarios considered. Fig. 3g summarizes these findings by showing only which method was best (where we distinguish different choices of q in HYPER), the corresponding average effective screening capacity, and the corresponding configuration.

HYPER pool designs from hypergraph factorization

As we illustrated in Fig. 1a, HYPER assigns individuals to pools by cycling through a sequence of pool assignments given by hypergraph factorization. Namely, for $q = 2$ splits and $m = 6$ pools labelled A-F, we cycled through the 15 possible pairs of pools in the order: AB, CD, EF, BC, DF, AE, BD, AF, CE, BE, CF, AD, BF, DE, AC. Namely, individual 1 was assigned to pools A and B, individual 2 to C and D, and so on; after individual 15, we return to the beginning of the sequence and cycle through again. For $n = 18$ individuals, this would result in the following pool assignments:

Individual	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Pools	AB	CD	EF	BC	DF	AE	BD	AF	CE	BE	CF	AD	BF	DE	AC	AB	CD	EF

For use in a lab protocol, it can sometimes be helpful to re-order these assignments so that individuals assigned to the same pair of pools appear consecutively:

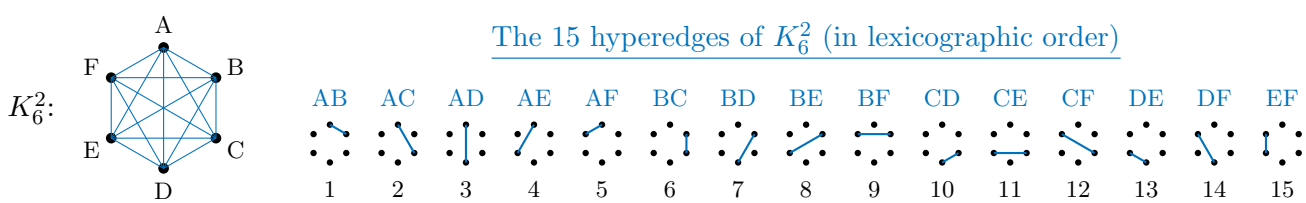
Individual	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Pools	AB	AB	CD	CD	EF	EF	BC	DF	AE	BD	AF	CE	BE	CF	AD	BF	DE	AC

This is useful if we plan to first combine the samples from individuals 1 and 2, then split that combined sample into pools A and B. Likewise for individuals 3 and 4, as well as 5 and 6, in this case. To avoid confusion, we emphasize that we have simply re-ordered the pool assignments to show the repeated ones one after the other. Thus, the table does not show AB, CD and EF as the first three pairs, but rather AB is repeated twice, then CD is repeated twice, and so on.

Hypergraph factorization

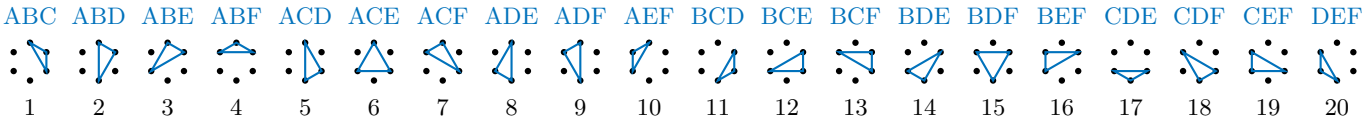
The sequence of pool assignments used in HYPER comes from hypergraph factorization. Here we describe the key ideas (and algorithms) for factorizations and the underlying theoretical results, in parallel with the application to group testing.

Suppose we are given a number m of vertices, which correspond to pools in group testing. We consider the *complete hypergraph* K_m^q of order $q \leq m$ on these m vertices, which is simply the collection of all *q-hyperedges*, i.e., subsets of size q of the m vertices. For $q = 2$, this is the complete graph on m vertices, which can be drawn as all edges connecting m vertices.



Drawing the corresponding complete hypergraph for $q = 3$ is harder, but one can quickly visualize the hyperedges as triangles connecting $q = 3$ vertices:

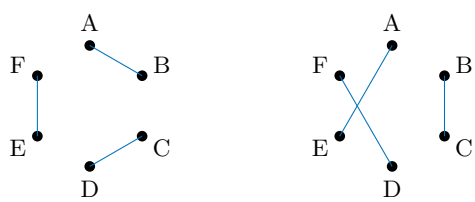
The 20 hyperedges of K_6^3 (in lexicographic order)



In general, there are $\binom{m}{q}$ hyperedges in the complete hypergraph K_m^q, given by all subsets of size q of the m vertices. In group testing, the hyperedges correspond to samples: each sample is placed into the pools contained in the subset of pools determined by a hyperedge.

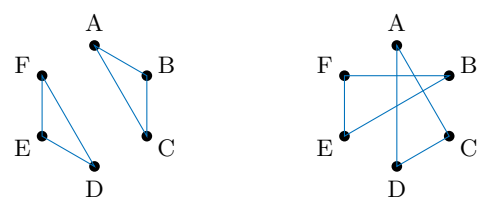
Connection to group testing. For HYPER group testing, we are interested in assigning samples to pools in a simple and balanced manner. The notion of balance corresponds to using each pool an equal number of times; or as close as possible to this. This can be achieved quite directly in simpler cases, but requires more work in more complex cases. Consider now the simplest case, where q, the number of pools into which each sample is placed divides the total number of pools. Under this number-theoretic condition, for any m pools, we can clearly split them into m/q non-overlapping subsets/hyperedges of size q, and thus for a set of m/q samples, we can use each of the m pools exactly once, achieving perfect balance. The above partitioning hyperedges are called a 1-*factor* of the hypergraph. For example, for $q = 2$ and $q = 3$ (both with $m = 6$) we could have:

Two example 1-factors of K_6^2



1. AB, CD, EF 2. BC, DF, AE

Two example 1-factors of K_6^3



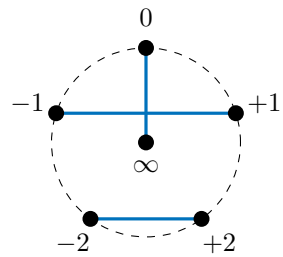
1. ABC, DEF 2. BEF, ACD

Baranyai's theorem. While this method to achieve balance is clear for $n = m/q$ samples, it is less clear if and how something similar can be achieved for more samples. In fact, a celebrated result in combinatorics, *Baranyai's theorem*[58], states that the complete hypergraph can be *factorized*, in the sense that the collection of $\binom{m}{q}$ hyperedges can be split/partitioned into $\binom{m}{q}/(m/q)$ non-overlapping and different 1-factors (each of which containing m/q hyperedges), such that each q-hyperedge appears in exactly one of the partitions. For group testing, this means that perfect balance can be achieved whenever the number of splits q divides the total number of pools m (e.g., for a two-pool split, we need an even number of pools). In theory, this solves exactly the problem we need.

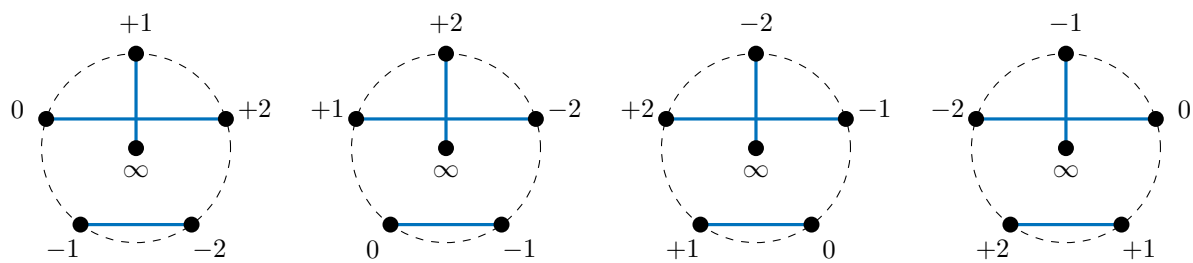
Baranyai's theorem guarantees the existence of the desired designs, but does not provide efficient algorithms for constructing them. In fact, at the current time, general constructions seem to be known only for $q = 2$ and

$q = 3$. For relatively small q, m one can certainly attempt to use brute-force enumeration to find appropriate designs. However, in this work we will focus on efficient and general constructions.

Hypergraph factorization for $q = 2$. For m even, we use the following efficient method for constructing a factorization of K_m^2. Here we follow the description in Section VII-5.5 of[29] and illustrate it using $m = 6$ as an example; see also page 595 of[27]. The construction begins with the following starter:



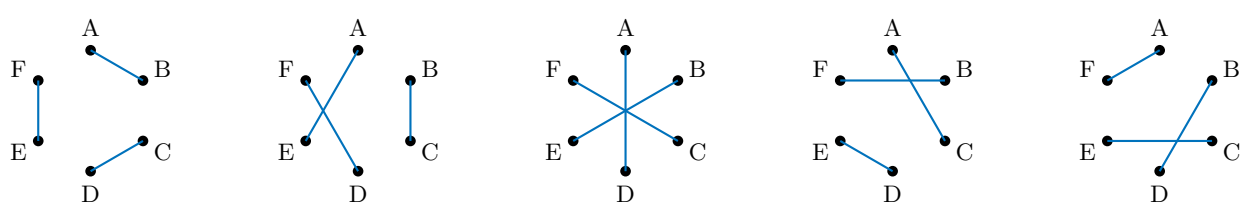
Namely, vertices $V = \{-u, \ldots, +u, \infty\}$ where $u = m/2-1$, i.e., $\mathbb{Z}_{m-1} \cup \{\infty\}$, with the starter 1-factor formed by edges $\{(0, \infty), (-1, +1), \ldots, (-u, +u)\}$. The remaining 1-factors are then generated by "rotating the diagram" as follows:



This yields $m-1$ many 1-factors in total, that taken together form the desired hypergraph factorization of K_m^2. To connect this back to the pools, simply relabel the vertices V using the pool names. For example, using

A : 0 B : ∞ C : +1 D : −1 E : +2 F : −2

for the above yields the following 1-factors:



AB, CD, EF BC, DF, AE BE, CF, AD BF, DE, AC BD, AF, CE

Hypergraph factorization for $q = 3$ via Beth's construction. We will leverage the non-trivial number-theoretic construction for $q = 3$ due to Thomas Beth[59;60], which is guaranteed to work when $m = 6k$ for some integer k (so the total number of pools is divisible by six), and $r = 6k - 1$ is a prime number (that is, a number that is not divisible by any other number between 1 and r). In what can be viewed as a lucky coincidence,

it so happens that most of the designs that we are interested in enjoy these properties: e.g., for pool sizes of $m = 6, 12, 24, 48$, each are divisible by six, and further we have that $r = 5, 11, 23, 47$ are prime numbers.

Algebraic background. We follow the description in[59] (which appears to be difficult to access online; the construction is also presented in the thesis[60], Section 3.1, and also referenced in[61]). The construction works as follows. Consider the finite field (or Galois field) of the prime order r, $GF(r) = \mathbb{Z}/r\mathbb{Z} = F_r$. This is defined as the set of numbers $\{0, 1, \ldots, r-1\}$, with addition, multiplication, and division by nonzero elements all defined modulo r (i.e., the result is always the residue after dividing by r). To this field, we append the symbol ∞, as the result of division by zero, so that $1/\infty = 0$. We also define $c + \infty = c \cdot \infty = \infty$ for all $c \neq 0 \in F_r$. This constitutes the so-called "projective line" $PG(1, r)$, with the "point" ∞ at infinity.

Beth's construction. Now, Beth considers the fractional linear map $\pi : PG(1, r) \to PG(1, r)$ given by $\pi(x) = -(1 + x)/x$. Here, 1 denotes the additive unit of the field, while addition and division are taken modulo r. A key observation is that π is a fixed-point-free map or order three; that is, it maps $x \to \pi(x) \to \pi \circ \pi(x) \to \pi \circ \pi \circ \pi(x) = x$, such that all intermediate values are distinct. Thus, these *orbits* of π are sets of size three that partition $PG(1, r)$. Let $O = \{A_i, i = 1, \ldots, (r + 1)/3\}$ be the partition of $PG(1, r)$ into orbits (and note that the size of $PG(1, r)$ is $r + 1$, hence there are $(r + 1)/3$ orbits).

Let also ω be a primitive element of F_r, that is an element such that $\omega^j \neq 1$, for any $j = 1, 2, \ldots, r - 2$. Then, Beth's result[59;60] states that the partitions induced by multiplying and translating O by specific values λ, g as $\lambda \cdot O + g$ form a 1-factorization of F_r. Here $\lambda \cdot O + g$ means that we take each of the hyperedges $A_i \in O$, and transform their elements affinely into $\lambda \cdot A_i + g$, thus obtaining another hyperedge. Specifically, λ needs to take the values of the powers of ω given by $\lambda = \omega^j$, $j = 1, \ldots, (r - 1)/2$, and g can take any value in F_r.

The key for us is that this construction can be evaluated very efficiently, by simply iterating over the orbits of π and the values λ, g.

Example: $r = 5$. For $m = 6$ pools, $r = m - 1 = 5$ is a prime number and Beth's construction applies. We go through the construction in this setting as an illustrative example.

1. For $r = 5$, we have $F_r = \{0, 1, 2, 3, 4\}$ and $PG(1, r) = \{0, 1, 2, 3, 4, \infty\}$.

2. We compute orbits of π by repeatedly applying π as follows:

$$\pi(0) = -\left(\frac{1+0}{0}\right) \equiv \infty, \qquad \pi(\infty) = -\left(\frac{1+\infty}{\infty}\right) \equiv 4, \qquad \pi(4) = -\left(\frac{1+4}{4}\right) \equiv 0,$$
$$\pi(1) = -\left(\frac{1+1}{1}\right) \equiv 3, \qquad \pi(3) = -\left(\frac{1+3}{3}\right) \equiv 2, \qquad \pi(2) = -\left(\frac{1+2}{2}\right) \equiv 1.$$

So we have the two orbits $O = \{\{0, \infty, 4\}, \{1, 3, 2\}\}$ that partition $PG(1, r)$.

3. We find a primitive element of F_r by looking at powers

$$
\begin{array}{llll}
1^1 = 1, & 2^1 = 2, & 3^1 = 3, & 4^1 = 4, \\
1^2 = 1, & 2^2 = 4, & 3^2 = 4, & 4^2 = 1, \\
1^3 = 1, & 2^3 = 3, & 3^3 = 2, & 4^3 = 4, \\
1^4 = 1, & 2^4 = 1, & 3^4 = 1, & 4^4 = 1.
\end{array}
$$

So we can choose $\omega \in \{2, 3\}$. We will (arbitrarily) choose $\omega = 2$.

4. Finally, we loop through λ and g. For $\omega = 2$, λ loops through $\{2, 4\}$.

	$\lambda = 2$	$\lambda = 4$
$g = 0$	$\{0, \infty, 3\}$ $\{2, 1, 4\}$	$\{0, \infty, 1\}$ $\{4, 2, 3\}$
$g = 1$	$\{1, \infty, 4\}$ $\{3, 2, 0\}$	$\{1, \infty, 2\}$ $\{0, 3, 4\}$
$g = 2$	$\{2, \infty, 0\}$ $\{4, 3, 1\}$	$\{2, \infty, 3\}$ $\{1, 4, 0\}$
$g = 3$	$\{3, \infty, 1\}$ $\{0, 4, 2\}$	$\{3, \infty, 4\}$ $\{2, 0, 1\}$
$g = 4$	$\{4, \infty, 2\}$ $\{1, 0, 3\}$	$\{4, \infty, 0\}$ $\{3, 1, 2\}$

Note that each cell in the resulting output table forms a partition of $\{0, 1, 2, 3, 4, \infty\}$ as desired. Looping over the cells and using the partitions to form pools yields the desired design.

General background on design theory To understand our designs, and how they fit into the broader context, it is valuable to introduce some basic concepts from design theory. See e.g.,[27;28] for excellent introductions, and we will follow notation and terminology from those references.

For us, a design I is a collection of points V and blocks B, and an assignment of some points to some blocks. In group testing, the points correspond to samples, and the blocks correspond to pools. The terminology of points and blocks is meant to be evocative of geometry, and indeed designs are closely connected to finite geometries such as affine and projective planes. Intuitively, points can sometimes be viewed as geometric points, while blocks can be viewed as lines. Points will be denoted with lowercase letters such as p, while blocks will be denoted with upper case letters such as B. The fact that point p is associated with (or incident on) block B is denoted as pIB.

Designs are called q-hypergraphs, if the size of the set of points incident on each block B is q, i.e., $(B) = \{p \in V : pIB\}$ has size q. For group testing, this means that each sample is assigned to q pools. We are thus interested in q-hypergraphs, for small values of q, such as $2, 3$.

A partition of a design is a disjoint union of its blocks into parts. A parallel class of a design is a collection of blocks such that each point is incident on exactly one block. This is analogous of the geometric idea that parallel lines do not intersect, and so if we partition the space into parallel lines, then each point belongs to exactly one such line. A design is called resolvable if it has a partition into parallel classes (a.k.a 1-factorization, parallelism or resolution).

For group testing, a resolution means that in each part of the partition of the blocks, we use each pool exactly once. Since our goal is to use pools in a balanced way, this is precisely what we want. Thus, from a design theory perspective, we are interested exactly in resolutions of q-hypergraphs. There is a great amount of work on existence and constructions of such resolutions, see e.g.,[27] Ch VIII and references therein.

One classical strategy is the permutation group action approach. Here, we start with a collection $\{B_j\}$, $j = 1, \ldots, J$ of base blocks (say of size q), viewed as subsets of the set of pools $[m] = \{1, \ldots, m\}$ where m is the number of pools, and a subgroup G of the permutation group S_m on m elements. If the action of G on $B_j \subset [m]$ $j = 1, \ldots, J$ leads to J non-overlapping orbits, then there are classical conditions under which the collection of these orbits forms a t-design (i.e., a hypergraph such that all t-subsets of points are incident on the same number of blocks), see Theorem III.8.2 on p. 207 of[27]. A more specific technique is the difference cycle method, which is an application of the permutation group action method when the cyclic group $G = \mathbb{Z}_l$ acts on the vertices $V = \mathbb{Z}_l$ by translation. This approach leads to resolutions of the complete hypergraph for both $q = 2$ and $q = 3$, see Section VIII.8 of[27]. These are precisely the algorithms that we use.

Performance characterization

To gain further insight into the performance of the proposed hypergraph design, here we study its operating characteristics.

Bound on expected number of tests

Our first result gives a sharp upper bound on the expected number of tests for hypergraph factorization designs. To get this result, we leverage the Dawson-Sankoff inequality[62], a nontrivial refinement of the Bonferroni union-intersection inequalities, which we use in the form given by[63]. We suppose here that tests have perfect sensitivity and specificity, and we will relax this later.

Theorem 1. *Consider hypergraph factorization designs with any number of samples n, number of pools m, number of pools per sample q, such that n is a multiple of m/q, and let $k = nq/m$ be the pool size. Let p be the prevalence level, and suppose that each sample is positive independently with probability p. Let T be the number of tests required, which is a random variable. For any positive integer $l \geq 2$, with $u = \binom{m-2}{q-2} \cdot \lceil n/\binom{m}{q} \rceil$, the*

expected number of tests is upper bounded by

$$\mathbb{E}T \leq m + n \cdot \left[1 - \frac{2q}{l}(1-p)^k + \frac{q(q-1)}{l(l-1)}(1-p)^{2k-u}\right].$$

The bound becomes an equality when (A) $q = 1$ or (B) $n < \binom{m}{q}$ and $q = 2$; and with taking $l = 2$ in both cases. For general q, the optimal choice for the parameter l is bounded by

$$l \leq \left\lfloor (q-1)(1-p)^{k-u} \right\rfloor + 2.$$

Proof of Theorem 1. Let R_i, $i = 1, \ldots, n$ be the indicator of the event that we need to re-test individual/sample i. Using the standard approach of calculating $\mathbb{E}T$, we find that the number of tests required is equal to m (one for each pool), plus any retests required. Hence,

$$\mathbb{E}T = m + \sum_{i=1}^{n} P(R_i).$$

Now, for our decoder, R_i happens precisely when all groups containing i are positive. Let T_i be the indicator that the ith sample is positive, which in this case is the same as the ith test outcome is positive. Let G_j be the groups containing sample j.

We use a refined version of the Bonferroni inequality known as the Dawson-Sankoff inequality[62] to bound $P(R_i)$. First let us recall the familiar Bonferroni union-intersection inequalities. Consider events $A_1, \ldots, A_n$, and for all $j \in \{1, \ldots, n\}$, define

$$S_j = \sum_{i_1 < i_2 < \ldots < i_j} P(A_{i_1} \cap \cdots \cap A_{i_j}).$$

Then, the well-known Bonferroni inequalities state that for even $h \in \{1, \ldots, n\}$,

$$P(\cup_{i=1}^{n} A_i) \geq \sum_{i=1}^{h} (-1)^{i-1} S_i.$$

We set $A_j = \{\forall_{k \in G_j} T_k = 0\}$. Then for any even h

$$P(\neg R_i) = P(\cup_{j=1}^{q} A_j) \geq \sum_{j=1}^{h} (-1)^{j-1} S_j.$$

Taking $h = 2$, we thus find

$$P(R_i) \leq 1 - S_1 + S_2.$$

We can get sharper results with the Dawson-Sankoff inequality[62]. In the form given by[63], this states that for any integer $l \geq 2$,

$$P(\cup_{j=1}^{q} A_j) \geq \frac{2S_1}{l} - \frac{2S_2}{l(l-1)}.$$

Now we can write $P(A_j) = (1-p)^{|G_j|}$ and for $j \neq j'$, we have $P(A_j \cup A_{j'}) = (1-p)^{|G_j \cup G_{j'}|}$.

It remains to bound $|G_j|$. Here hypergraph factorization designs are useful, because they try to balance $|G_j|$. In each consecutive block of m/q samples, they use each of the m pools exactly once. Thus, in each G_j, there is at most one new sample. Recall that k is the number consecutive of blocks of samples of size m/q, and we assumed that $k = nq/m$ is an integer. Based on the above, we have $|G_j| = k$.

Moreover, $|G_j \cup G_{j'}| = |G_j| + |G_{j'}| - |G_j \cap G_{j'}|$, and the intersection $G_j \cap G_{j'}$ has size at most $|G_j \cap G_{j'}| \leq u := \binom{m-2}{q-2} \cdot \lceil n/\binom{m}{q} \rceil$. The reason is that, since hypergraph designs are maximally balanced, they only intersect at most $\binom{m-2}{q-2}$ times in every consecutive block of $\binom{m}{q}$ samples. These intersections correspond to the number of ways of choosing the remaining $q-2$ pools out of the remaining $m-2$. Hence, $P(A_j) = (1-p)^k$ and for $j \neq j'$, $P(A_j \cup A_{j'}) \leq (1-p)^{2k-u}$. Thus, $S_1 \geq q(1-p)^k$ and $S_2 \leq \binom{q}{2}(1-p)^{2k-u}$. In addition, we have equality for S_2 when either (A) $q = 1$, in which case the intersection is empty and $u = 0$, or (B) $n < \binom{m}{q}$ and $q = 2$ (in which case we know that the groups intersect in exactly $u = 1$ sample, which is the sample defining them) or (C) n is a multiple of $\binom{m}{q}$ (in which case we know that the number of intersections is exactly given by u for each pair of groups). This leads to the desired result.

As is known from[62;63], the optimal choice for l is $l = \lfloor 2S_2/S_1 \rfloor + 2$. We can approximate the optimal choice using the calculations from the proof as

$$l \leq \left\lfloor \frac{2\binom{q}{2}(1-p)^{2k-u}}{q(1-p)^k} \right\rfloor + 2 \leq \left\lfloor (q-1)(1-p)^{k-u} \right\rfloor + 2.$$

This finishes the proof. □

Let us denote $r = 1-p$. When $n \leq \binom{m}{q}$, an optimal hypergraph design is obtained by minimizing the number of per-person tests $E(m,q) = \mathbb{E}T/n$, i.e.,

$$\min_{q,m \in \mathbb{N}^+} E(m,q) = \frac{m}{n} + p + r\left[1 - r^{nq/m-1}\right]^q.$$

This formula is exact for $q = 1$ and $q = 2$. One can check that we recover the bound from above with equality for these cases when $n \leq \binom{m}{q}$ (so $u = 0$ for $q = 1$ and $u = 1$ for $q = 2$), and by taking $l = 2$. In more detail, we can write (replacing above $k = nq/m$, and using $r = 1-p$)

$$E(m,1) = \frac{m}{n} + p + r\left[1 - r^{k-1}\right] = \frac{m}{n} + 1 - r^k$$
$$E(m,2) = \frac{m}{n} + p + r\left[1 - r^{k-1}\right]^2 = \frac{m}{n} + 1 - 2r^k + r^{2k-1}.$$

Moreover, by substituting into Theorem 1 $u = 0$ for $q = 1$ and $u = 1$ for $q = 2$, and by taking $l = 2$; the bounds

given there for $\mathbb{E}T/n$ (denote them $\tilde{E}(m,q)$) become

$$\tilde{E}(m,1) = \frac{m}{n} + [1-(1-p)^k] = \frac{m}{n} + 1 - r^k$$
$$\tilde{E}(m,2) = \frac{m}{n} + \left[1 - \frac{2q}{l}(1-p)^k + \frac{q(q-1)}{l(l-1)}(1-p)^{2k-u}\right]$$
$$= \frac{m}{n} + \left[1 - qr^k + \frac{q(q-1)}{2}r^{2k-1}\right] = \frac{m}{n} + 1 - 2r^k + r^{2k-1}.$$

This shows that the upper bounds are sharp for $q = 1, 2$.

However, for a larger q, this formula corresponds to the so-called *locally tree-like* approximation in the graphical model corresponding to the observation model, when computing the probability of R_i.

Optimal efficiency for $q = 2$

Our next result characterizes the optimal efficiency of hypergraph designs for $q = 2$ for small $p \to 0$.

Proposition 2 (Optimal efficiency for $q = 2$)**.** *In the noiseless large n case (i.e., $\alpha = 0$, $\beta = 1$, and $n \to \infty$), the optimal efficiency of HYPER with $q = 2$ is approximately $E^* \approx 3p^{2/3}$ in the limit $p \to 0$ with $m/n \approx 2p^{2/3} - p$.*

These results are valid in the regime of p where $n \leq \binom{m}{q}$, which restricts p to be larger than a certain threshold. When $p \to 0$, we expect the optimal m to decrease in this limit, eventually leading to $n > \binom{m}{q}$. In practice, this means that the formula is valid for a larger range of p when n is larger.

In comparison, the optimal efficiency for Dorfman testing ($q = 1$) is approximately $2p^{1/2}$ for small p e.g.,[20]. Thus hypergraph designs improve over Dorfman designs. The same asymptotic efficiency $E^* \approx 3p^{2/3}$ is also achieved by three-stage-testing[20] as well as double pooling tests[56]. However, our proposal is a two-stage approach. In comparison, both the hierarchical testing approaches proposed in[20] and[8] attain an asymptotic efficiency $E^* \approx ep\ln(1/p)$ as $p \to 0$, which is asymptotically more efficient; but it requires multi-stage tests that we avoid.

More generally, we note that there is a lot of work on optimality of group testing, for various cases, e.g., two-stage and multi-stage algorithms, adaptive and non-adaptive algorithms, in worst case or average case, etc[41–46]. As $p \to 0$, these works and others discuss a universal lower bound of order $\Theta(p\ln(1/p))$ on the efficiency. However, the specific rate at which $p \to 0$ with n can lead to better rates; and better algorithms. In particular, under the same probabilistic model as in our paper[44] construct certain tests where each sample is placed into $q = \lceil \ln(1/p)/\ln 2 \rceil$ pools, and show that these attain asymptotic efficiency $p\ln(1/p)$. The work[47] constructs a 2-stage algorithm with asymptotically optimal efficiency, requiring $q = m\ln(2)/(np)$. The work[48] discusses similar proposals for the noisy case. In our work, the constraints we work with do not allow q to grow with $p \to 0$. The work[49] proposes a 4-stage algorithm with asymptotically optimal efficiency.

Proof of Proposition 2. Taking the limit as $n \to \infty$ with $m/n = y$ fixed, where $\alpha = 0$ and $\beta = 1$, we obtain

$$E = \frac{T}{n} = \frac{m}{n} + 1 - 2(1-p)^{2n/m} + (1-p)^{4n/m-1} = y + 1 - 2(1-p)^{2/y} + (1-p)^{4/y-1}.$$

To optimize, we differentiate E with respect to y, obtaining

$$\frac{\partial E}{\partial y} = 1 + \frac{4(1-p)^{2/y}\log(1-p)}{y^2} - \frac{4(1-p)^{4/y-1}\log(1-p)}{y^2}.$$

The optimal y can be obtained by solving $0 = \partial E/\partial y$ for y in terms of p. We approximate this solution in the limit $p \to 0$ by taking the leading two terms of the Puiseux expansion (around $p = 0$) of the degree four Taylor approximation of $\partial E/\partial y$ (with respect to $p = 0$). This has one branch corresponding to a real solution yielding $y^\star \approx 2p^{2/3} - p$.

Substituting $y = 2p^{2/3} - p$ into E and computing a Taylor approximation yields

$$E^\star = 2p^{2/3} - p + 1 - 2(1-p)^{2/(2p^{2/3}-p)} + (1-p)^{4/(2p^{2/3}-p)-1} = 3p^{2/3} + O(p^{4/3}) \approx 3p^{2/3},$$

completing the derivation. □

Noisy case

We next study the noisy case, where each of the pooled tests can have false positives and false negatives. Our first result gives an exact formula for the expected number of tests for $q = 1, 2$, and moreover, also gives formulas for the false positive and false negative rates for each individual test.

Theorem 3 (Performance of hypergraph factorization: noisy case)**.** *Consider hypergraph factorization designs in a noisy observation model. Suppose n is a multiple of m/q. Suppose $q = 1$ or $q = 2$, and $n \leq \binom{m}{q}$. Let $k = nq/m$ and $r = 1 - p$. The expected number of tests has the following exact form:*

$$\mathbb{E}T = m + n\Big\{p \cdot \beta^q + r \cdot \gamma\Big\}.$$

Here $\gamma = [\beta + (\alpha - \beta) \cdot r^{k-1}]^q$. Denoting the odds ratio as $o = (1-p)/p$ the true negative and true positive probabilities for each individual sample's status T_i and the test results $\widehat{T}_i$ are, respectively,

$$P(T_i = 0 | \widehat{T}_i = 0) = \frac{1}{1 + o^{-1}\frac{1-\beta^{q+1}}{1-\alpha\gamma}}.$$
$$P(T_i = 1 | \widehat{T}_i = 1) = \frac{1}{1 + o\frac{\alpha\gamma}{\beta^{q+1}}}.$$

Our second result gives a more generally applicable upper bound on the expected number of tests, valid for all q.

Theorem 4 (Performance bound for hypergraph factorization: noisy case, upper bound)**.** *Consider hypergraph factorization designs in a noisy observation model with any number of samples n, number of pools m, number of pools per sample q, such that n is a multiple of m/q, and let $k = nq/m$. Let $u = \binom{m-2}{q-2} \cdot \lceil n/\binom{m}{q} \rceil$. Let p be the prevalence level, and suppose that each sample is positive independently with probability p. Suppose the*

pools and the re-tests have sensitivity $1-\alpha$ (where α is the level of each test), and specificity (or power) β. For any positive integer $l \geq 2$, the expected number of tests is upper bounded by

$$\mathbb{E}T \leq m + n \cdot \left[1 - \frac{2q}{l}p_1 + \frac{q(q-1)}{l(l-1)}p_2\right],$$

where

$$p_1 = 1 - \beta + (\beta - \alpha) \cdot r^k$$
$$p_2 = p_1^2 + (\alpha - \beta)^2 \cdot r^{2k-u}(1 - r^u).$$

Here $r = 1 - p$. The optimal choice of l, minimizing the upper bound, is achieved by $l = \lfloor (q-1)p_2/p_1 \rfloor + 2$.

Proof of Theorems 3 and 4. We will follow, to some extent, the notation and assumptions from Bilder's works, see e.g.,[21]. Let $\hat{H}_j$ be the binary result of testing group j, and H_j be the true status of the j-th group. By definition, $1-\alpha$ is the sensitivity of each grouped test, i.e., $1 - \alpha = P(\hat{H}_j = 0 | H_j = 0)$ (we use this notation in convention with the notion of α for the level of a test in hypothesis testing); and β is the specificity (or power) of each grouped test, i.e., $\beta = P(\hat{H}_j = 1 | H_j = 1)$. Moreover, we assumed that each test outcome is independent.

We have

$$\begin{aligned} P(R_i) &= P(R_i | T_i = 1)P(T_i = 1) + P(R_i | T_i = 0)P(T_i = 0) \\ &= P(R_i | T_i = 1) \cdot p + P(R_i | T_i = 0) \cdot (1 - p). \end{aligned}$$

The key is to determine the probabilities $P(R_i | T_i = t)$. Now R_i happens if and only if each of the groups containing i have a positive status, i.e., for all $i \in G_j$, we have $\hat{H}_j = 1$:

$$R_i = \cap_{i \in G_j} \{\hat{H}_j = 1\}.$$

Since $q = 1, 2$, and $n \leq \binom{m}{q}$, these groups are non-overlapping outside of i, and thus, their probabilities are independent conditional on i. We can thus write

$$\begin{aligned} P(R_i | T_i = t) &= P(\cap_{i \in G_j} \{\hat{H}_j = 1\} | T_i = t) \\ &= \prod_{i \in G_j} P(\hat{H}_j = 1 | T_i = t). \end{aligned}$$

Next, we can condition on H_j for each term, to write

$$\begin{aligned} P(\hat{H}_j = 1 | T_i = t) &= P(\hat{H}_j = 1 | H_j = 1)P(H_j = 1 | T_i = t) + P(\hat{H}_j = 1 | H_j = 0)P(H_j = 0 | T_i = t) \\ &= \beta \cdot P(H_j = 1 | T_i = t) + \alpha \cdot P(H_j = 0 | T_i = t) \\ &= \beta + (\alpha - \beta) \cdot P(H_j = 0 | T_i = t). \end{aligned}$$

Moreover, since $H_j = \max_{i \in G_j} T_i$, we have

$$\begin{aligned} P(H_j = 0 | T_i = 1) &= 0, \\ P(H_j = 0 | T_i = 0) &= \prod_{k \in G_j, k \neq i} P(T_k = 0) \\ &= (1-p)^{|G_j|-1}. \end{aligned}$$

Working our way back up, we can substitute these above to find

$$\begin{aligned} P(\hat{H}_j = 1 | T_i = 1) &= \beta \\ P(\hat{H}_j = 1 | T_i = 0) &= \beta + (\alpha - \beta) \cdot (1-p)^{|G_j|-1}. \end{aligned}$$

Letting $n_i = |\{i : i \in G_j\}|$ be the number of groups that i belongs to, we find

$$\begin{aligned} P(R_i | T_i = 1) &= \beta^{n_i} \\ P(R_i | T_i = 0) &= \prod_{i \in G_j} [\beta + (\alpha - \beta) \cdot (1-p)^{|G_j|-1}] \end{aligned}$$

Finally,

$$P(R_i) = \beta^{n_i} \cdot p + \prod_{i \in G_j} [\beta + (\alpha - \beta) \cdot (1-p)^{|G_j|-1}] \cdot (1-p).$$

Hence

$$\begin{aligned} \mathbb{E}T &= m + \sum_{i=1}^{n} P(R_i) \\ &= m + \sum_{i=1}^{n} \left\{ \beta^{n_i} \cdot p + \prod_{i \in G_j} [\beta + (\alpha - \beta) \cdot (1-p)^{|G_j|-1}] \cdot (1-p) \right\} \\ &= m + p \cdot \sum_{i=1}^{n} \beta^{n_i} + (1-p) \cdot \sum_{i=1}^{n} \left\{ \prod_{i \in G_j} [\beta + (\alpha - \beta) \cdot (1-p)^{|G_j|-1}] \right\}. \end{aligned}$$

Recall that for the hypergraph designs, $|G_j| \leq \lceil n/(m/q) \rceil$, and if n is an integer multiple of m/q as assumed here, then $|G_j| = nq/m$. Moreover, by construction, $n_i = q$. Hence we find

$$\mathbb{E}T = m + n \left\{ p \cdot \beta^q + (1-p) \cdot [\beta + (\alpha - \beta) \cdot (1-p)^{nq/m-1}]^q \right\}.$$

This gives the desired formula for the expected number of tests. Next we derive the per-test false negative and positive rates. Let $\widehat{T}_i$ be the indicator that the i-th sample is declared positive. This happens if all groups containing i are positive in the first round, and then the result of a second independent test $\widehat{T}_{i2}$ is also positive. Recall that T_i is the indicator that the ith sample is positive, and $P(T_i = 1) = p$.

We are interested in the probabilities $P(T_i = t | \widehat{T}_i = \hat{t})$, for $t, \hat{t} \in \{0, 1\}$. For $P(T_i = 1 | \widehat{T}_i = 1)$ this denotes the true positive probability; while $P(T_i = 0 | \widehat{T}_i = 0)$ this denotes the true negative probability. Using Bayes' rule, we can write

$$
\begin{aligned}
P(T_i = t|\widehat{T}_i = \hat{t}) &= \frac{P(\widehat{T}_i = \hat{t}|T_i = t)P(T_i = t)}{P(\widehat{T}_i = \hat{t})} \\
&= \frac{P(\widehat{T}_i = \hat{t}|T_i = t)P(T_i = t)}{(1-p)\cdot P(\widehat{T}_i = \hat{t}|T_i = 0) + p \cdot P(\widehat{T}_i = \hat{t}|T_i = 1)}.
\end{aligned}
$$

Thus the probabilities reduce to determining $P(\widehat{T}_i = t|T_i = t)$. We can write, assuming the result of the re-test is independent of the original tests, and assuming the re-test has the same operating characteristics as any grouped test, that

$$
\begin{aligned}
P(\widehat{T}_i = 1|T_i = t) &= P(R_i = 1, \widehat{T}_{i2} = 1|T_i = t) \\
&= P(R_i = 1|T_i = t) \cdot P(\widehat{T}_{i2} = 1|T_i = t)
\end{aligned}
$$

We have

$$
\begin{aligned}
P(\widehat{T}_{i2} = 1|T_i = 0) &= \alpha \\
P(\widehat{T}_{i2} = 1|T_i = 1) &= \beta.
\end{aligned}
$$

Hence, using our previous results, denoting $\gamma = \prod_{i \in G_j}[\beta + (\alpha - \beta)\cdot(1-p)^{|G_j|-1}]$,

$$
\begin{aligned}
P(\widehat{T}_i = 1|T_i = 0) &= \gamma \cdot \alpha \\
P(\widehat{T}_i = 1|T_i = 1) &= \beta^{n_i} \cdot \beta = \beta^{n_i+1}
\end{aligned}
$$

Working our way back,

$$
\begin{aligned}
P(T_i = 1|\widehat{T}_i = 0) &= \frac{P(\widehat{T}_i = 0|T_i = 1)P(T_i = 1)}{(1-p)\cdot P(\widehat{T}_i = 0|T_i = 0) + p \cdot P(\widehat{T}_i = 0|T_i = 1)} \\
&= \frac{p \cdot [1 - \beta^{n_i+1}]}{(1-p)\cdot(1-\alpha\gamma) + p \cdot [1 - \beta^{n_i+1}]}. \\
P(T_i = 1|\widehat{T}_i = 1) &= \frac{P(\widehat{T}_i = 1|T_i = 1)P(T_i = 1)}{(1-p)\cdot P(\widehat{T}_i = 1|T_i = 0) + p \cdot P(\widehat{T}_i = 1|T_i = 1)} \\
&= \frac{p \cdot \beta^{n_i+1}}{(1-p)\cdot\alpha\gamma + p \cdot \beta^{n_i+1}}.
\end{aligned}
$$

Recall that for the hypergraph designs, if n is an integer multiple of m/q, then $\gamma = [\beta + (\alpha - \beta)\cdot(1-p)^{nq/m-1}]^q$ and $n_i = q$. Hence, denoting the odds ratio as $o = (1-p)/p$ we find that the true negative and true positive probabilities are, respectively,

$$
\begin{aligned}
P(T_i = 0|\widehat{T}_i = 0) &= \frac{1}{1 + o^{-1}\frac{1-\beta^{q+1}}{1-\alpha\gamma}}. \\
P(T_i = 1|\widehat{T}_i = 1) &= \frac{1}{1 + o\frac{\alpha\gamma}{\beta^{q+1}}}.
\end{aligned}
$$

This finishes the proof of Theorem 4. Now, we proceed to Theorem 3. This follows in a similar way to the previous Theorem 1, but with more involved calculations. As before, the Dawson-Sankoff inequality[62], in the form given by[63] states that for any integer l,

$$P(\cup_{j=1}^{q} A_j) \geq \frac{2S_1}{l} - \frac{2S_2}{l(l-1)}.$$

Taking $A_j = \{\hat{H}_j = 0\}$, we find that the above equals $R_i^c = \cup_{j=1}^{q} A_j$. Now $S_1 = \sum P(A_j)$, $S_2 = \sum_{j<g} P(A_j \cap A_g)$. Thus it is enough to give a lower bound for $P(\hat{H}_j = 0)$ and an upper bound for $P(\hat{H}_j = \hat{H}_g = 0)$ for all $j \neq g$.

We can condition on H_j to write

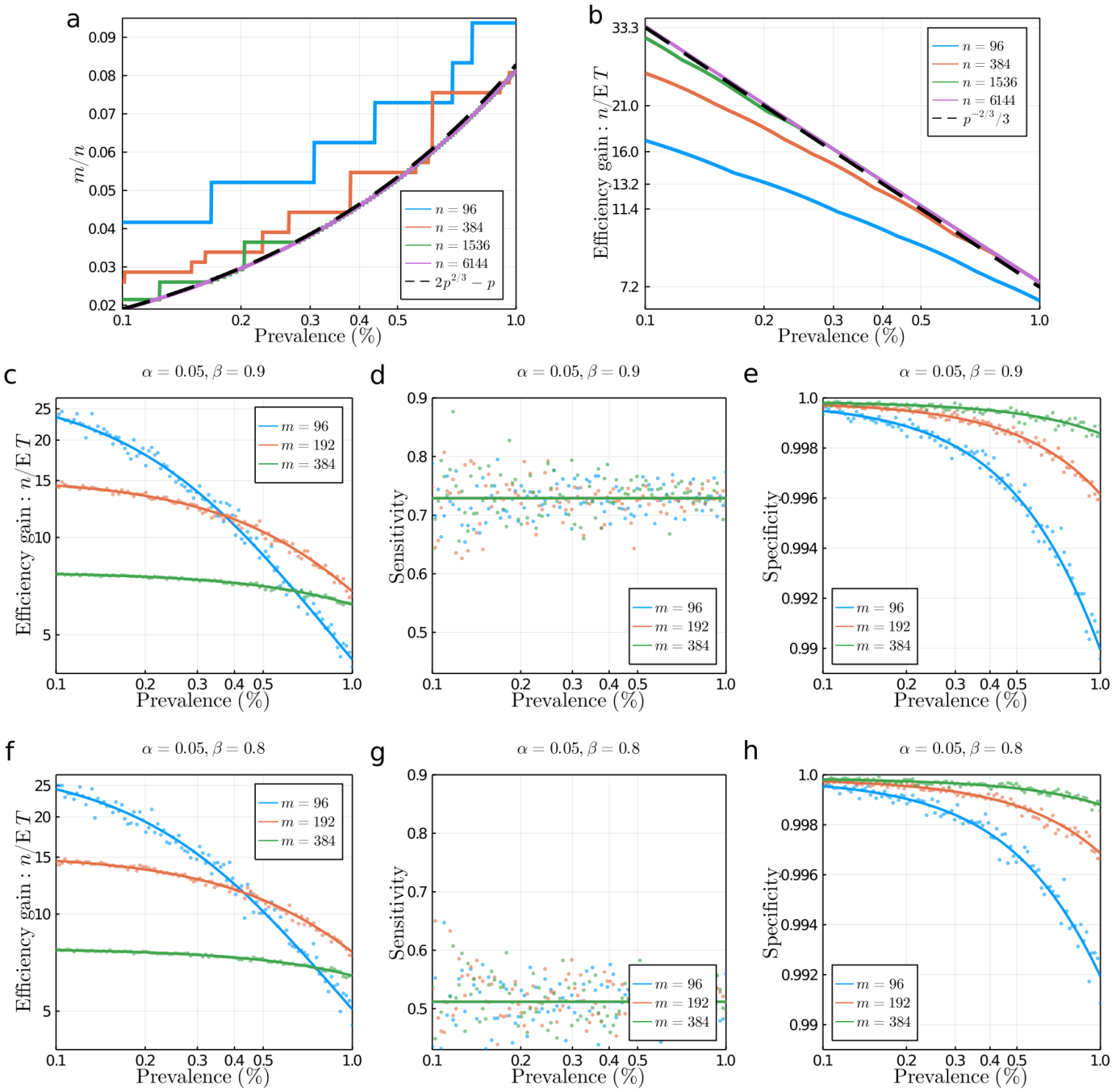
$$\begin{aligned}
P(\hat{H}_j = 0) &= P(\hat{H}_j = 0 | H_j = 0) P(H_j = 0) + P(\hat{H}_j = 0 | H_j = 1) P(H_j = 1) \\
&= (1-\alpha) \cdot P(H_j = 0) + (1-\beta) \cdot [1 - P(H_j = 0)] \\
&= 1 - \beta + (\beta - \alpha) \cdot P(H_j = 0) \\
&= 1 - \beta + (\beta - \alpha) \cdot (1-p)^{|G_j|} \\
&= 1 - \beta + (\beta - \alpha) \cdot (1-p)^{k}.
\end{aligned}$$

In the last line, we have used that n is a multiple of m/q.

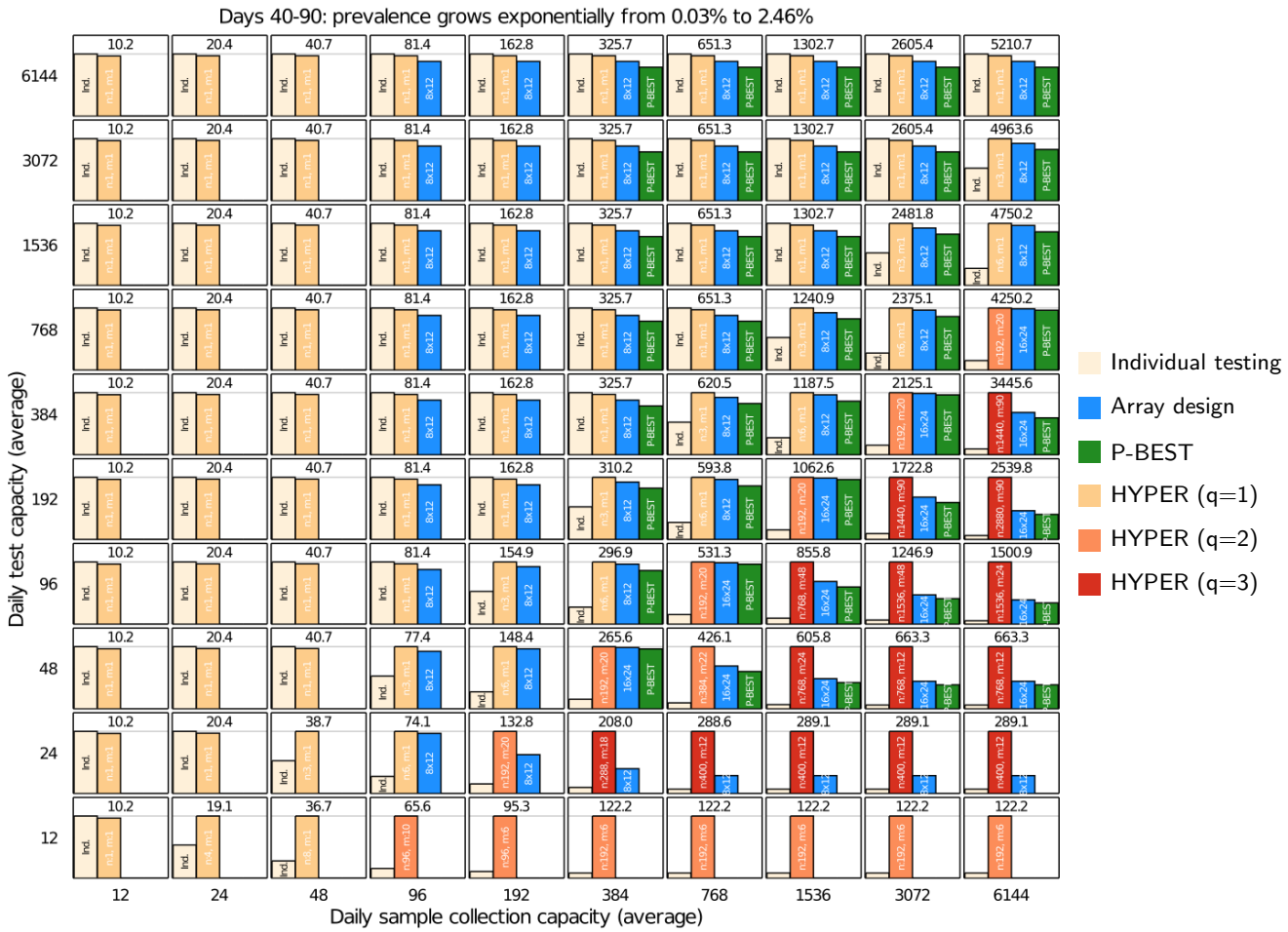
Similarly, we can calculate for $j \neq k$, noting that $|G_j| = |G_g| = k$, and denoting $|G_j \cap G_g| = v$, so that $|G_j \cup G_g| = 2k - v$,

$$\begin{aligned}
P(\hat{H}_j = \hat{H}_g = 0) &= \sum_{a,b=0}^{1} P(\hat{H}_j = \hat{H}_g = 0 | H_j = a, H_g = b) P(H_j = a, H_g = b) \\
&= (1-\alpha)^2 \cdot r^{2k-v} + 2(1-\beta)(1-\alpha) r^k (1 - r^{k-v}) + (1-\beta)^2 (1 - 2r^k + r^{2k-v}) \\
&= [1 - \beta + (\beta - \alpha) \cdot r^k]^2 + (\alpha - \beta)^2 \cdot (r^{2k-u} - r^{2k}).
\end{aligned}$$

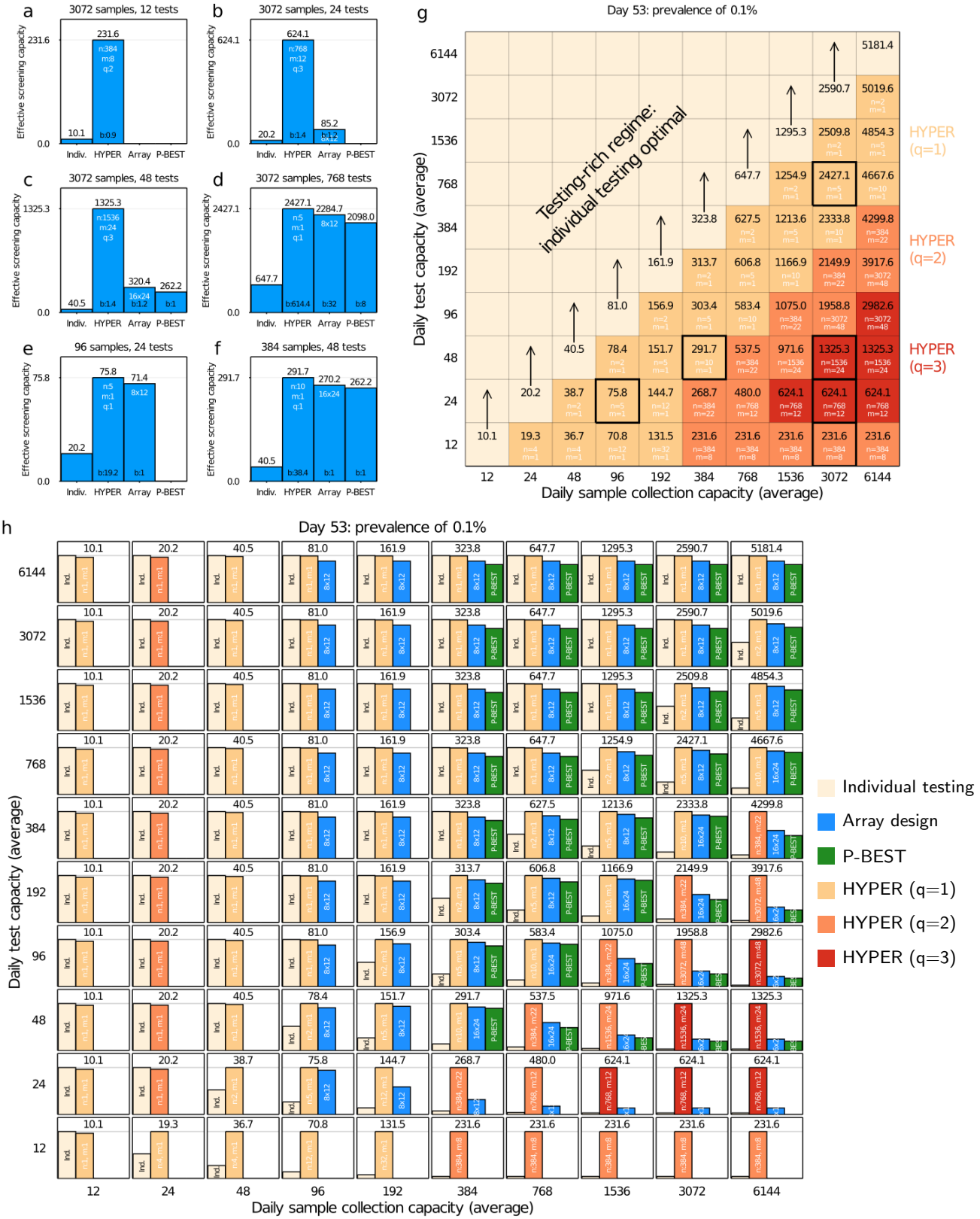
As discussed before, due to the construction of hypergraph factorization designs, we have $v \leq u = \binom{m-2}{q-2} \cdot \lceil n / \binom{m}{q} \rceil$. Since the above expression is monotonically increasing in u, we can also conclude that we can upper bound it by replacing v with u. This finishes the proof. □



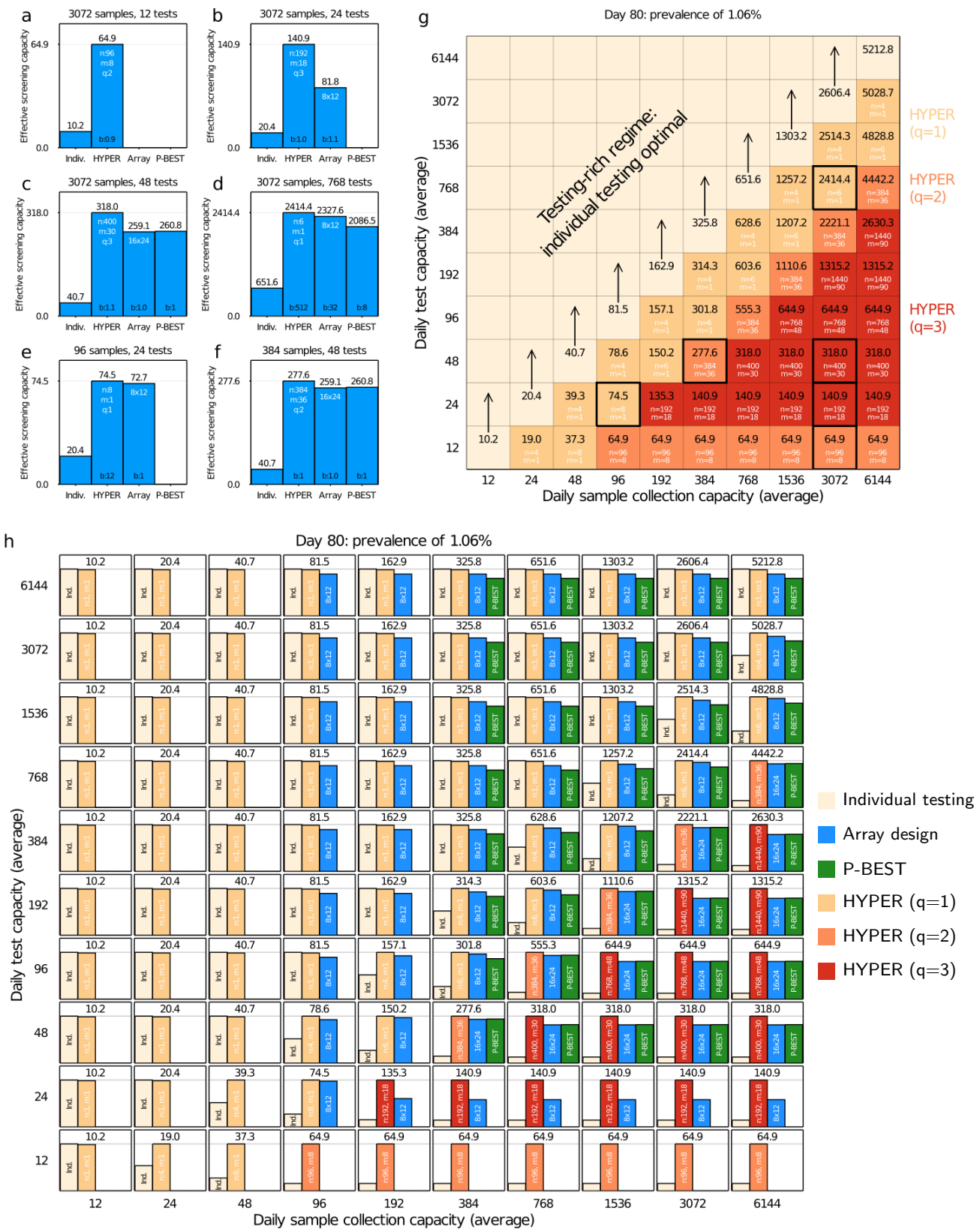
Supplementary Figure 1: Theoretical characterization. We consider the optimal choice (**a**) of m for $q = 2$ in the noiseless large n case (i.e., $\alpha = 0$, $\beta = 1$, and $n \to \infty$) as $p \to 0$. As n grows from $n = 96$ to $n = 6144$, the value of m optimizing the efficiency (found by exhaustive search) approaches the theoretical limit of $m/n \approx 2p^{2/3} - p$ for small prevalence p. Likewise, the corresponding optimal efficiency (**b**) approaches $\mathbb{E}T/n \approx 3p^{2/3}$ for small prevalence p. The approximation improves for increasingly small p as $n \to \infty$. We also consider a noisy case where all tests have sensitivity $\beta = 90\%$ and specificity $1 - \alpha = 95\%$. We consider the overall efficiency (**c**), sensitivity (**d**), and specificity (**e**) of three designs with three choices of m, all with $n = 3072$ and $q = 2$. The theoretical predictions (solid curves) agree well with simulation results (scatter plots). For a lower sensitivity $\beta = 80\%$ (but same specificity $1 - \alpha = 95\%$), the efficiency (**f**) is slightly better (since more positives are missed), the sensitivity (**g**) is lower, and the specificity (**h**) is slightly improved (at larger prevalence).



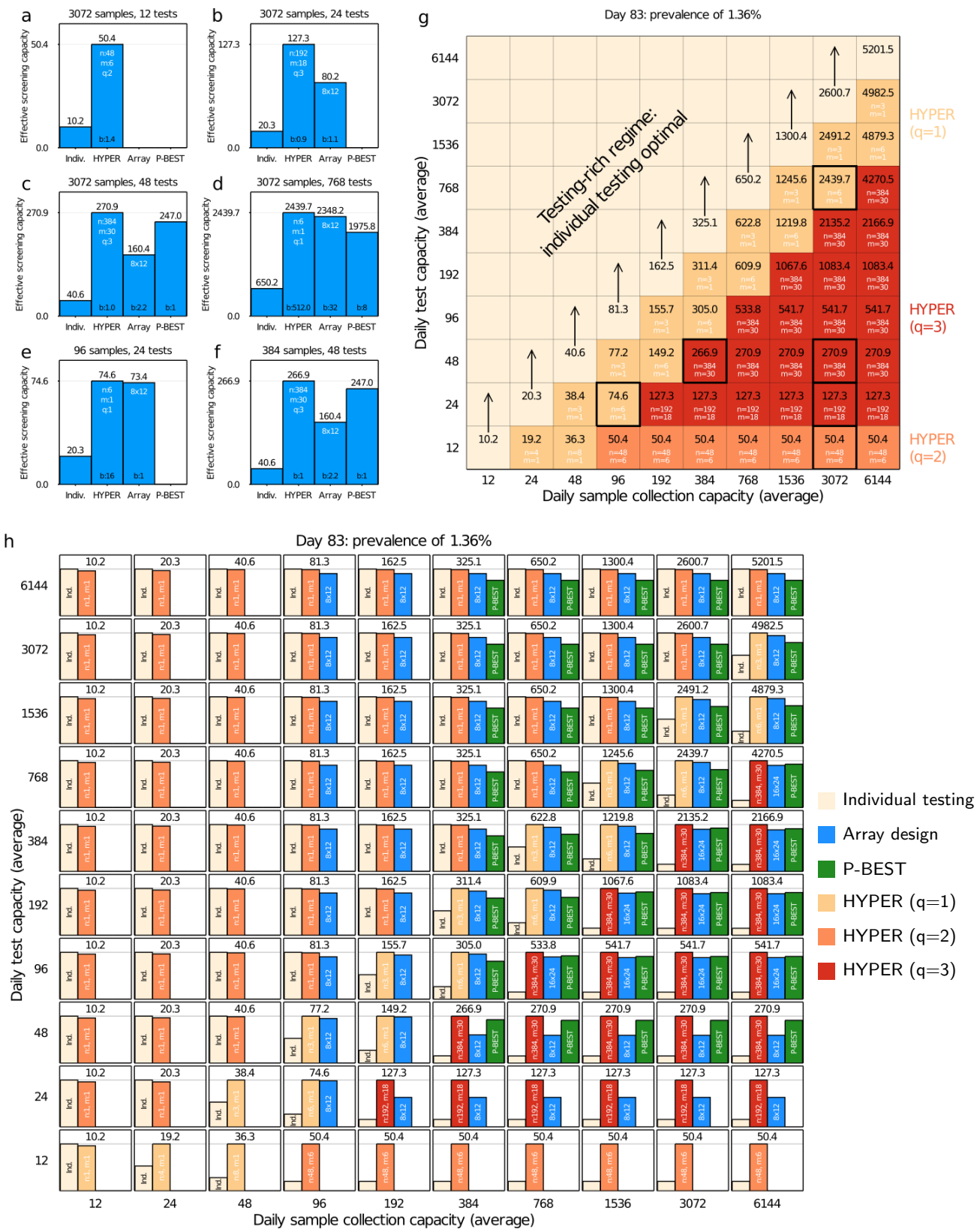
Supplementary Figure 2: Detailed comparison corresponding to Fig. 3. Each cell shows the effective number of individuals screened given constrained sample collection and testing capacities, for individual testing, HYPER (best n and m noted in white with q indicated by bar color), array designs[7] (better design between 8×12 and 16×24 noted in white), and P-BEST[6]. The best effective screening capacity for each cell is shown in black.



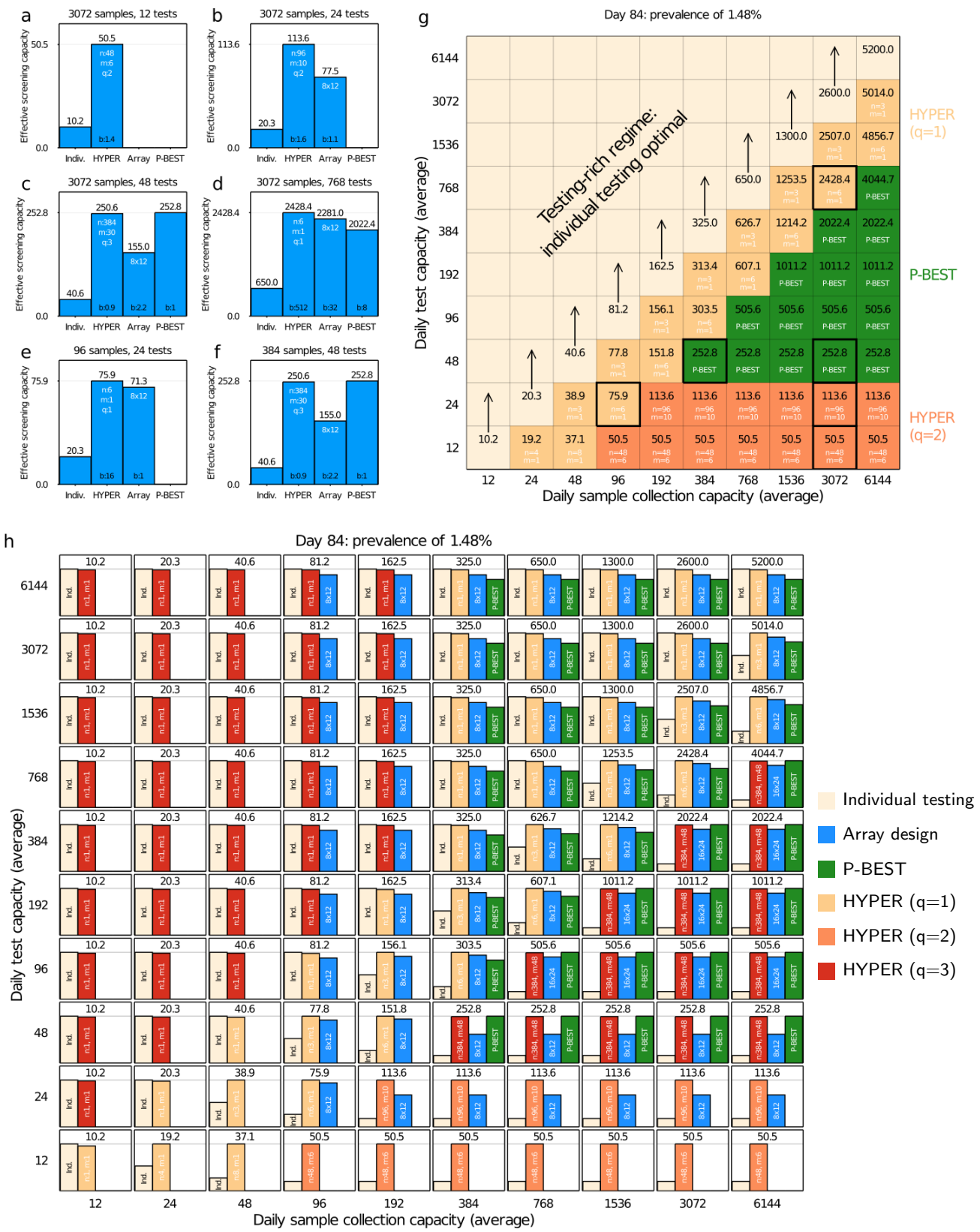
Supplementary Figure 3: Comparison of pooling methods under resource constraints (Fig. 3 and Supplementary Fig. 2) repeated for day 53 (prevalence of 0.1%). At this low prevalence, HYPER is the most effective across the grid of resource constraints and can be significantly so. The low prevalence setting also favors simple designs (low q) across more of the test-constrained regime.



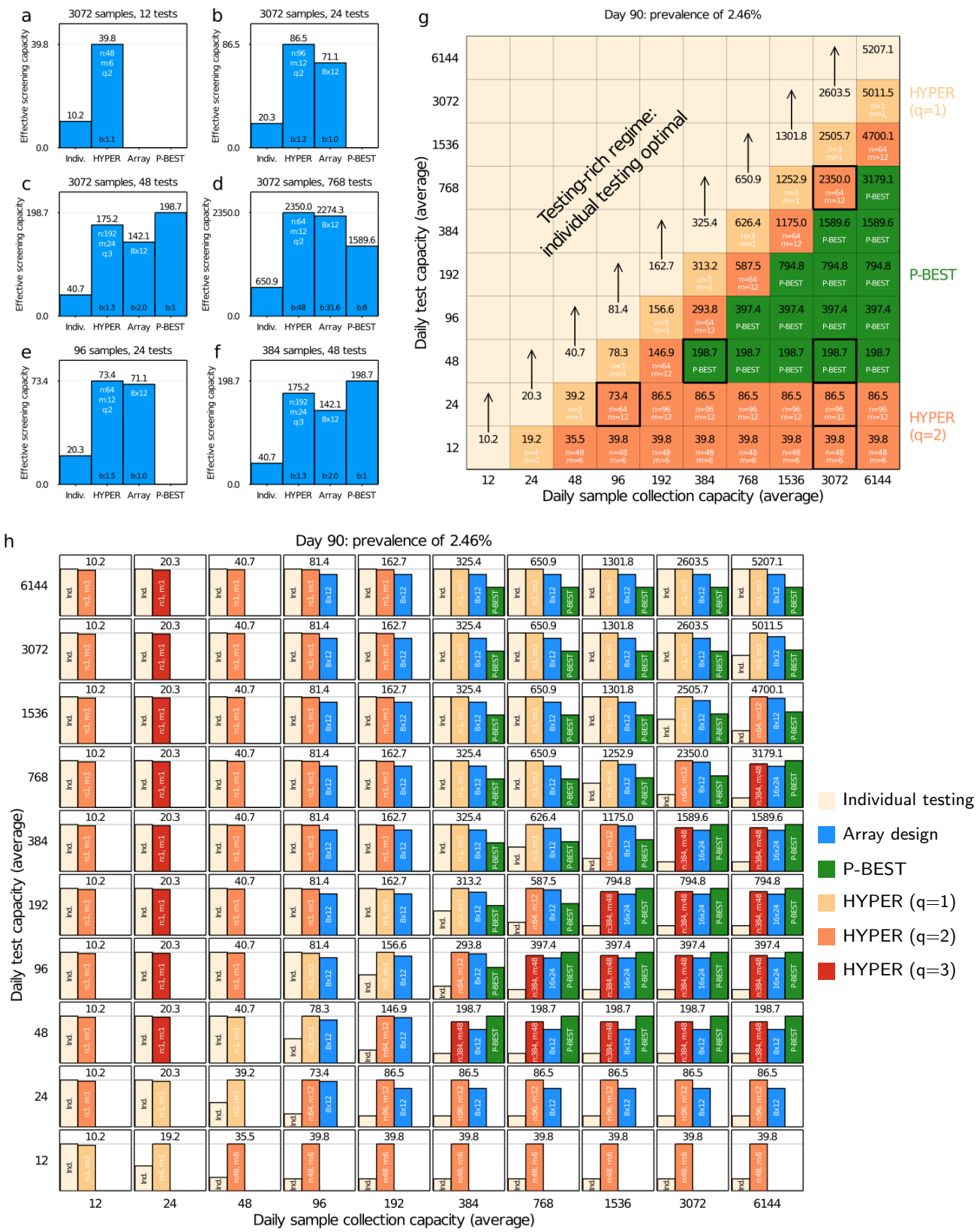
Supplementary Figure 4: Comparison of pooling methods under resource constraints (Fig. 3 and Supplementary Fig. 2) repeated for day 80 (prevalence of 1.06%). At this moderate prevalence, HYPER remains best across all scenarios. Compared to Supplementary Fig. 3, which had lower prevalence, designs with higher q are more often the most effective.



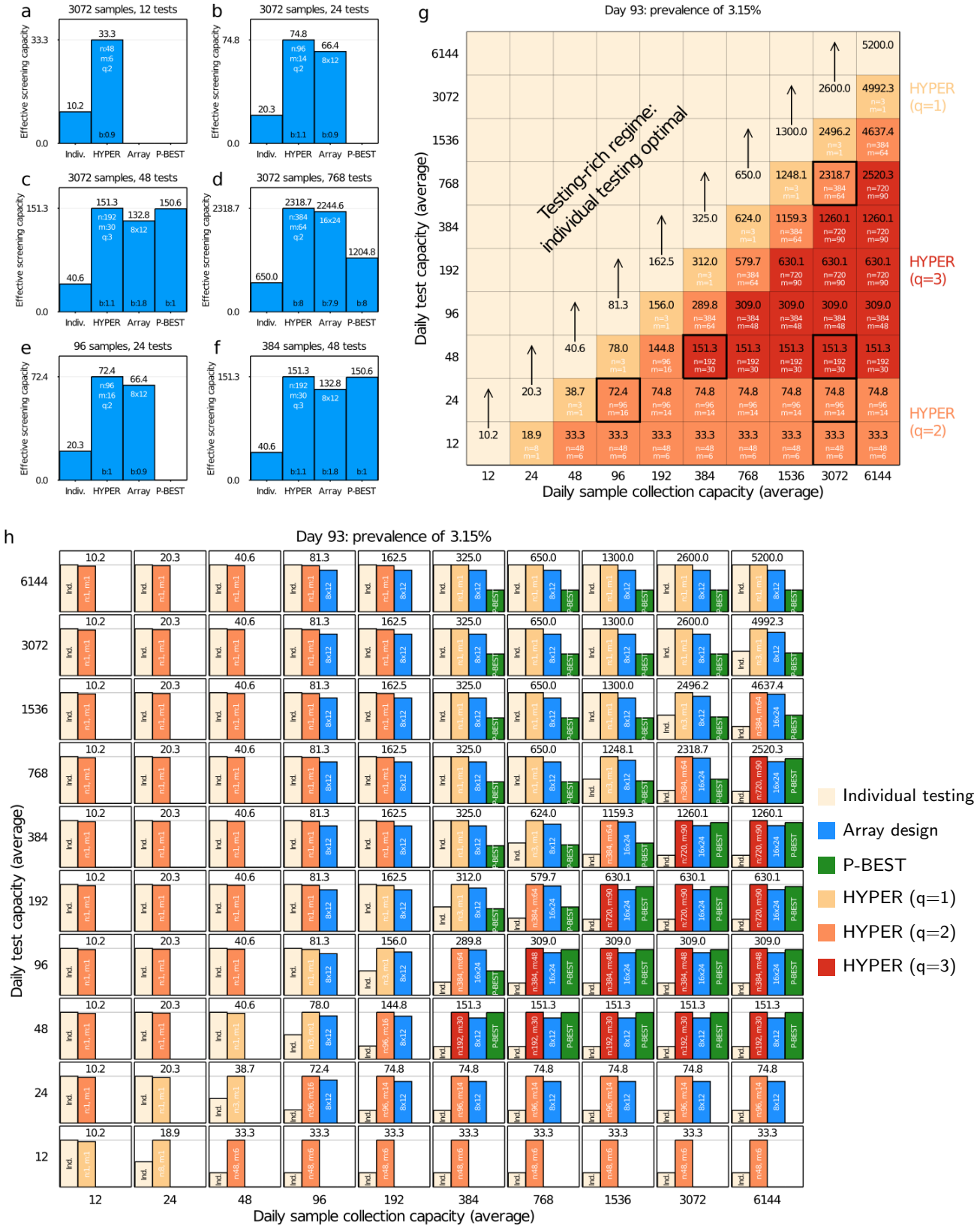
Supplementary Figure 5: Comparison of pooling methods under resource constraints (Fig. 3 and Supplementary Fig. 2) repeated for day 83 (prevalence of 1.36%). At this prevalence, HYPER remains best across all scenarios. Compared to Supplementary Fig. 4, designs with higher q are even more often the most effective.



Supplementary Figure 6: Comparison of pooling methods under resource constraints (Fig. 3 and Supplementary Fig. 2) repeated for day 84 (prevalence of 1.48%). At this intermediate prevalence, there is a subset of scenarios in which P-BEST outperforms HYPER. However, HYPER is still nearly as effective in these scenarios. Outside these settings P-BEST is either not viable or substantially under-performs HYPER.



Supplementary Figure 7: Comparison of pooling methods under resource constraints (Fig. 3 and Supplementary Fig. 2) repeated for day 90 (prevalence of 2.46%). At this prevalence, P-BEST continues to outperform HYPER on a subset of scenarios, with a larger gap than in Supplementary Fig. 6. Outside these settings P-BEST is still either not viable or substantially under-performs HYPER.



Supplementary Figure 8: Comparison of pooling methods under resource constraints (Fig. 3 and Supplementary Fig. 2) repeated for day 93 (prevalence of 3.15%). At this higher prevalence, HYPER again performs best across all scenarios.

Supplementary Table 1: List of HYPER design $\mathbf{H}_{n,m,q}$ considered. We simulated the following HYPER configurations and selected the most effective for each scenario of the constrained-resource analyses shown in Fig. 3 and Supplementary Figs. 2 to 8.

$q = 1$

n	m	n	m
4	1	384	1
8	1	384	2
12	6	384	3
20	10	384	4
40	10	384	6
40	20	384	8
44	11	384	12
44	22	384	16
48	12	384	24
48	16	384	32
48	24	384	48
64	8	384	64
64	16	384	96
64	32	384	128
80	20	384	192
96	1	400	10
96	2	400	20
96	3	400	40
96	4	400	50
96	6	400	80
96	8	400	100
96	12	400	200
96	16		
96	24		
96	32		
96	48		
100	20		
192	12		
192	16		
192	32		
192	48		
192	64		
192	96		
288	12		
288	18		
288	36		
288	48		
288	96		
288	144		
360	40		

$q = 2$

n	m	n	m	n	m
12	6	96	64	384	42
12	12	144	30	384	44
20	12	192	6	384	46
20	20	192	12	384	48
24	6	192	20	384	64
24	12	192	24	384	96
40	12	192	40	384	192
40	20	192	44	384	288
40	40	192	48	400	12
44	12	192	64	400	20
44	20	192	192	400	40
44	40	192	96	400	44
44	44	288	12	400	48
48	6	288	20	400	64
48	12	288	30	400	96
48	20	288	40	400	192
48	24	288	44	400	288
48	40	288	48	400	384
48	44	288	64	768	12
48	48	288	96	768	24
64	12	288	192	768	48
64	20	384	4	768	96
64	40	384	6	1536	24
64	44	384	8	1536	48
64	48	384	10	1536	96
64	64	384	12	3072	48
96	4	384	14	3072	96
96	6	384	16	6144	96
96	8	384	18		
96	10	384	20		
96	12	384	22		
96	14	384	24		
96	16	384	26		
96	18	384	28		
96	20	384	30		
96	22	384	32		
96	24	384	34		
96	40	384	36		
96	44	384	38		
96	48	384	40		

$q = 3$

n	m	n	m	n	m
12	6	96	48	400	48
12	12	96	60	400	60
12	18	96	90	400	90
20	12	144	30	400	192
20	18	180	90	400	282
20	30	192	6	400	384
24	6	192	12	400	390
24	12	192	18	720	90
40	12	192	24	768	12
40	18	192	30	768	24
40	30	192	42	768	48
40	42	192	48	1440	90
40	48	192	60	1536	24
40	60	192	90	1536	48
44	12	192	192	2880	90
44	18	288	12	3072	48
44	30	288	18	5760	90
44	42	288	30		
44	48	288	42		
44	60	288	48		
48	6	288	60		
48	12	288	90		
48	18	288	192		
48	24	288	282		
48	30	360	90		
48	42	384	6		
48	48	384	12		
48	60	384	18		
64	12	384	24		
64	18	384	30		
64	30	384	42		
64	42	384	48		
64	48	384	60		
64	60	384	90		
96	6	384	192		
96	12	384	282		
96	18	400	12		
96	24	400	18		
96	30	400	30		
96	42	400	42		