Bloom Origami Assays: Practical Group Testing

Louis Abraham, Gary Becigneul, Benjamin Coleman, Bernhard Scholkopf, Anshumali Shrivastava, Alexander Smola

▶ **To cite this version:**

Louis Abraham, Gary Becigneul, Benjamin Coleman, Bernhard Scholkopf, Anshumali Shrivastava, et al.. Bloom Origami Assays: Practical Group Testing. 2021. hal-03454803

HAL Id: hal-03454803
https://hal.science/hal-03454803

Preprint submitted on 29 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

arXiv:2008.02641v1 [cs.LG] 21 Jul 2020

Bloom Origami Assays: Practical Group Testing

Louis Abraham
ETH Zurich
Zurich, Switzerland
louis.abraham@yahoo.fr

Gary Bécigneul
ETH Zürich
Zürich, Switzerland
gary.becigneul@inf.ethz.ch

Benjamin Coleman
Rice University
Houston, TX
ben.coleman@rice.edu

Bernhard Schölkopf
MPI for Intelligent Systems
Tübingen, Germany
bs@tuebingen.mpg.de

Anshumali Shrivastava
Rice University
Houston, TX
anshumali@rice.edu

Alexander Smola
Amazon Web Services
Palo Alto, CA
smola@amazon.com

Abstract

We study the problem usually referred to as *group testing* in the context of COVID-19. Given n samples collected from patients, how should we select and test mixtures of samples to maximize information and minimize the number of tests? Group testing is a well-studied problem with several appealing solutions, but recent biological studies impose practical constraints for COVID-19 that are incompatible with traditional methods. Furthermore, existing methods use unnecessarily restrictive solutions, which were devised for settings with more memory and compute constraints than the problem at hand. This results in poor utility. In the new setting, we obtain strong solutions for small values of n using evolutionary strategies. We then develop a new method combining Bloom filters with belief propagation to scale to larger values of n (more than 100) with good empirical results. We also present a more accurate decoding algorithm that is tailored for specific COVID-19 settings. This work demonstrates the practical gap between dedicated algorithms and well-known generic solutions. Our efforts results in a new and practical multiplex method yielding strong empirical performance without mixing more than a chosen number of patients into the same probe. Finally, we briefly discuss adaptive methods, casting them into the framework of adaptive sub-modularity.

1 Introduction

Lacking effective treatments or vaccinations, the most effective way to save lives in an ongoing epidemic is to mitigate and control its spread. This can be done by testing and isolating positive cases early enough to prevent subsequent infections. If done regularly and for a sufficiently large fraction of susceptible individuals, mass testing has the potential to prevent many of the infections a positive case would normally cause. However, a number of factors, such as limits on material and human resources, necessitate economical and efficient use of test resources.

Group testing aims to improve test quality by testing groups of samples simultaneously. We wish to leverage this framework to design practical and efficient COVID-19 tests with limited testing resources. Group testing can be ***adaptive*** or ***non-adaptive***. In the former, tests can be decided one at a time, taking into account previous test results. In the latter, one can run tests in parallel, but also has to select all tests before seeing any lab results.

A popular example of a ***semi-adaptive*** group test is to first split n samples into g groups of (roughly) equal size, pool the samples within the groups and perform g tests on the pooled samples. All samples in negatively tested pools are marked as negative, and all samples in positively tested pools are subsequently tested individually.

Preprint. Under review.

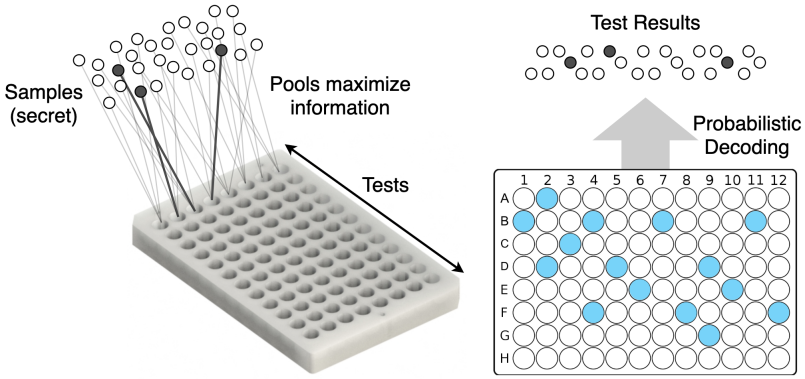


Figure 1: We formulate the group testing problem as a constrained information maximization problem. Samples are grouped into testing pools so that the information gain is maximized while obeying practical constraints (i.e. no more than 64 samples in one group). Here, positive samples are shown in black and positive tests are shown in blue. The tests are decoded with error correcting probabilistic methods.

Practical Constraints for COVID-19. Although group testing is a well-studied problem, the recent COVID-19 pandemic introduces specific constraints. In contrast to seroprevalence antibody tests, PCR tests aim to detect *active* cases, and only successfully do so during part of the disease course [8]). This results in a small **prevalence** (prior probability of population infection; we will assume a **default value** of 10^{-3}), assuming we screen the general population rather than only symptomatic individuals. Group testing has recently been validated for COVID-19 PCR tests [22, 26]. It is facilitated by the fact that PCR is an amplification technique that can detect small virus concentrations. Nevertheless, there are limitations on the number of samples l that can be placed in a group ([26] considers up to 64), and constraints on the number of times a particular sample can be used ([22] uses serial incubation of the same respiratory sample in up to $k = 10$ tubes). Besides, there are practical issues: adaptive testing is time consuming and hard to manage. Complex multiplex designs are prone to human error.

Existing research on non-adaptive group testing is generally concerned with identifying at most k positive samples amongst n total samples, which is referred to as non-adaptive hypergeometric group testing [9]. This assumption yields asymptotic bounds on the number of tests needed to recover the ground truth [13, 10, 4, 3]. However, these are of limited practical relevance when constructive results on small numbers of samples are required. The specific constraints for COVID-19 force us to revisit the general framework of group testing.

Novel Formulation. We formulate the problem based on the principle of information gain: given n people and m testing kits, the characteristics of the test and prior probabilities for each person to be sick, we seek to optimize the way the tests are used by combining several samples. For simplicity, samples are assumed to be independent analogous to [18]. However, we focus on implementable tests, unlike [18] which focuses on asymptotic results that are valid for large n. Figure 1 summarizes our approach.

Optimal Characterization: By leveraging the framework of adaptive sub-modularity initially developed for sensor covering by [6], we prove near-optimality of a simple greedy-strategy for adaptive testing. Despite the simplicity, it turns out that this greedy strategy has exponential running time and becomes infeasible for $n \geq 16$. Fortunately, the near optimally of the greedy-adaptive method points toward a simple and scalable non-adaptive solution leveraging randomization akin to the Bloom Filter structure [20].

Bloom Origami Assays:[1] [19, 2] recently showed that pooling using random hash functions, similar to a Bloom filter structure, can lead to an efficient and straightforward group testing protocol. We will show that such a protocol, based on random hash functions, is unnecessarily restrictive. Bloom filters were designed for streaming data, where there is no choice but to use universal hash functions for

[1]The term *Origami* stems from the idea to use paper folding techniques for test multiplexing, see [25].

pooling. For COVID-19, the computational situation is much simpler. Leveraging our information gain framework, we propose superior but straightforward hashing strategies.

A bigger problem with Bloom filters is the (necessarily) simple decoder. The decoder trades accuracy for efficiency, as it was designed for internet-scale problems where linear time decoding is prohibitive. For COVID-19, we instead propose a message-passing decoder, similar to Counter Braids [16], which is more accurate. Our proposal of connecting probabilistic graphical model (PGM) inference with Bloom filters could be broadly applicable to situations beyond COVID-19 group testing. Since the graphical model is a bipartite graph for which no narrow junction tree can be found, message passing does not necessarily converge to the global optimum. Therefore, we propose a new method for graphical model inference leveraging probabilistic concentration and meet-in-the-middle (MITM) techniques, which may be of independent interest. Our MITM method is particularly useful for the low prevalence scenario. This paper illustrates the power of algorithmic re-design to target practical constraints. We obtain significant gains even on the relatively well-studied topic of group testing.

2 Preliminaries

Notations are progressively introduced throughout but are gathered in the appendix, which also contains the proofs. Denote the number of patient[2] samples by n. As previously mentioned, we consider the group testing task in the particular context of the COVID-19 pandemic. This choice of problem setting naturally introduces new mathematical constraints of a practical nature:

Impracticality of Adaptivity. Adaptive methods require several hours in between each lab result of the adaptive sequence. This inspires us to only consider either non-adaptive methods or semi-adaptive methods with no more than two phases of testing.

Low Concentration and Test Accuracy. Excessive mixing of patient swabs may result in prohibitively low viral concentration with negative consequences for testing. A recent study reports that one can safely mix a patient swab up to 10 times [22]; another relays that mixing up to 32 patient samples into the same probe yields a false negative rate below 10% [26].

There is clearly ambiguity in the limitations of the experimental protocol. For instance, [15] validate double-digit numbers of patients per sample for PCR tests. While dilution effects are relevant for such large pools, they can be partly addressed by incubating respiratory swabs multiple times [22]. Also note that we are only concerned with the accuracy of the *tests* per se rather than the biological sampling protocol (i.e. whether swabs are taken when viral load is detectable in patients). In what follows we consider group sizes of $n = 100$ as a sensible upper limit.

Notations and Reminders Denote the number of tests to run by m. Tests are assumed to be imperfect, with a *true positive rate (or sensitivity)* tpr[3] and *true negative rate (or specificity)* tnr.[4] As simple **default values**, we will use $\mathrm{tpr} = 99\%$ [21] and $\mathrm{tnr} = 90\%$ [26].[5]

Patient sample i is infected with probability $p_i \in [0, 1]$ and we assume statistical independence of infection of patient samples. Denoting by a '1' a positive result (infection), the unknown ground truth is a vector of size n made up of '0's and '1's. This vector describes who is infected and who is not. We call this the *secret*, denoted as $s \in \{0, 1\}^n$. A *design of a test* $d \in \{0, 1\}^n$ to run in the lab is a subset of patient samples to mix together into the same sample, where $d_i = 1$ if patient sample i is mixed into design d and $d_i = 0$ otherwise. Note that the outcome of a perfect design d for a given secret s can simply be obtained as $\mathbf{1}_{\langle d,s\rangle>0}$ where $\langle d, s\rangle := \sum_{i=1}^{n} d_i s_i$. That is, a test result is positive if there is at least one patient i for which $d_i = 1$ (patient i is included in the sample) and $s_i = 1$ (patient i is infected). Figure 1 illustrates the problem setting.

Recall that the secret s is unknown. However, since we assume that patient sample i is infected with probability p_i and that patient samples are independent, we have a *prior* probability distribution over the possible values of s. We hence represent the random value of s as a *random variable (r.v.)*, denoted by S, with probability distribution $p_S(s) := \Pr[S = s]$ over $\{0, 1\}^n$. Let us now recall the

[2]For simplicity, we will refer to all individuals being tested as *patients*.

[3]equivalent terms include *hit rate*, *detection rate* and *recall*.

[4]equivalent terms include *correct rejection rate* and *selectivity*.

[5]This number is affected by selection bias since it heavily depends on the stage of the disease; it is lower if a person is tested too late [8, 21]; our results provide guidance as to how to analyze the samples that were collected rather than the collection timing and protocol itself.

definition of the *entropy* of our random variable,

$$H(S) = -\sum_{s\in\{0,1\}^n} p_S(s)\log_2 p_S(s), \tag{1}$$

The entropy represents *the amount of uncertainty that we have on its outcome*, measured in bits. It is maximized when S follows a uniform distribution, and minimized when S constantly outputs the same value. As we perform tests, we gain additional knowledge about S. For instance, if we group all samples into the same pool and have a negative result, then our *posterior* probability that all patients are healthy goes up. That is, $p_S((0,\dots,0))$ increases according to Bayes' rule of probability theory. More generally, we may perform a sequence of tests of varying composition, updating our posterior after each test. Our goal will be to select designs of tests so as to minimize entropy, resulting in the least amount of uncertainty about the test outcome for all individuals.

3 Solving for Small Number of Patients

Given n people, test characteristics tpr & tnr and a set of prior probabilities of sample infection $(p_i)_{1\le i\le n}$, the best multiset $\mathcal{D}$ of m pool designs is the one maximizing the information gain. The tests are order insensitive, which gives a search space of cardinality $\binom{2^n+m}{m}$. Evaluating the information gain of every multiset separately takes $\mathcal{O}\left(2^{n+m}\right)$ operations.[6] Hence, brute-forcing this search space is prohibitive even for small values of n and m.

We resort to randomized algorithms to find a good enough solution. Our approach is to use Evolutionary Strategies (ES). We apply a variant of the $(1+\lambda)$ ES with optimal restarts [17] to optimize any objective function over individuals (multisets of tests).

Detailed Description. We maintain a population of 1 individual between steps. At every step of the ES, we mutate it in $\lambda \in \mathbb{N}^+$ offsprings. In the standard $(1+\lambda)$ ES, each offspring is mutated from the population, whereas our offsprings are iteratively mutated, each one being the mutation of the previous. These offsprings are added to the population, and the best element of the population is selected as the next generation of the population.

We initialize our population with the "zero" design that doesn't test anyone. Our mutation step is straightforward: flipping one bit d_i of one pool design d, both chosen uniformly at random. We also restrict our search space if needed: the number of 1's in a column must be less than the number of times a given swab can be mixed with others, the number of 1's in a line is constrained not to put too many swabs into the same pool. Our iterative mutation scheme allows us to step out of local optima.

After choosing a basis b proportional to $n \times m$ (which is approximately the logarithm of our search space), we apply restarts according to the Luby sequence: $(b, b, 2b, b, b, 2b, 4b, b, b, 2b, b, b, 2b, 4b, 8b, ...)$. This sequence of restarts is optimal for Las Vegas algorithms [17], and our ES can be viewed as such under two conditions: *(i)* that the population never be stuck in a local optimum, which can be achieved in our algorithm using $\lambda = n \times m$ (note that much smaller constant values are used in practice); *(ii)* the second condition is purely conceptual and consists in defining a success as having a score larger than some threshold. The fact that our algorithm does not use this threshold as an input yields the following result, proved in Appendix C.1:

Theorem 1. *Under condition (i), the evolutionary strategy using the Luby sequence for restarts yields a Las Vegas algorithm that restarts optimally [17] to achieve any target score threshold.*

4 Motivating Greedy Information Maximization

Note that since tests are imperfect, for a given pool design $d \in \{0,1\}^n$ and a given secret $s \in \{0,1\}^n$, the Boolean outcome $T(s,d)$ of the test in the lab is not deterministic. If tests were perfect, we would have $T(s,d) = \mathbf{1}_{\langle d,s\rangle>0}$. To allow for imperfect tests, we model $T(s,d)$ as a r.v. whose distribution is described by $\Pr[T(s,d)=1 \mid \langle d,s\rangle > 0] = \mathrm{tpr}$ and $\Pr[T(s,d)=0 \mid \langle d,s\rangle = 0] = \mathrm{tnr}$.[7] Since the secret s is also unknown (and described by the r.v. S), the outcome $T(S,d)$ has now two sources

[6] We chose to implement a version with complexity $\mathcal{O}\left(m2^{n+m}\right)$, but more cache efficient in practice.

[7] For prior information on whether and how the errors depend on the number of samples mixed into a given pool design (e.g. by dilution effects), we can take this into account by letting tpr and tnr depend on $|d| = \sum_i d_i$.

of randomness: imperfection of tests and unknown secret.[8] In practice, one will not run one test but multiple tests. We now suppose that m tests of pool designs are run and let their designs be represented as a multiset $\mathcal{D} \in (\{0,1\}^n)^m$.

This leads us to the following question: given an initial prior probability distribution p_S over the secret, how should we select pool designs to test in the lab? We want to select it such that once we have its outcome, we have as much information as possible about S, i.e. the entropy (uncertainty) of S has been minimized. Since we cannot know in advance the outcome of the tests, we have to minimize this quantity *in expectation* over the randomness coming from both the imperfect test and unknown secret. This requires the notion of *conditional entropy*.

Conditional Entropy. Given pool designs $\mathcal{D}$, we consider two random variables S (secret) and $T := T(S, \mathcal{D})$ (test results). The conditional entropy of S given T is given by:

$$H(S|T) = -\sum_{s\in\{0,1\}^n, t\in\{0,1\}^m} \Pr[S=s, T=t] \cdot \log_2\left(\frac{\Pr[S=s,T=t]}{\Pr[T=t]}\right) = \mathbb{E}_{t\sim T(S,\mathcal{D})}\left[H(p_{S|T=t})\right] \quad (2)$$

In this formula, the joint probability $\Pr[S = s, T = t]$ has been computed with the conditional probability formula $\Pr[S = s, T = t] = \Pr[S = s] \Pr[T = t|S = s]$, and the posterior distribution is computed using Bayesian updating, i.e.,

$$p_{S|T=t}(s) = \Pr[S = s|T = t] = \Pr[S = s, T = t]/\Pr[T = t], \quad (3)$$

where $\Pr[T = t] = \sum_s \Pr[S = s, T = t]$. It represents the amount of information (measured in bits) needed to describe the outcome of S, given that the result of T is known. The *mutual information* between S and T can equivalently be defined as $I(S, T) := H(S) - H(S|T)$. It quantifies the amount of information obtained about S by observing T.

A well-motivated criterion for test selection. Since $H(S)$ does not depend on d, selecting the pool design d minimizing the conditional entropy of S given the outcome of $\mathcal{D}$ is equivalent to selecting the one maximizing the mutual information between S and $T(S, \mathcal{D})$. We now have a clear criterion for selecting $\mathcal{D}$:

$$\mathcal{D}^* \in \arg\max_{\mathcal{D}} I(S, T(S, \mathcal{D})). \quad (4)$$

This criterion selects the pool designs $\mathcal{D}$ whose outcome will maximize our information about S.

Expected Confidence. We report another evaluation metric of interest called the *expected confidence*. It is the mean average precision of the maximum likelihood outcome. The maximum likelihood outcome it defined by:

$$\mathrm{ML}(t) := \arg\max_s \Pr[S = s|T = t], \quad (5)$$

which yields the following definition of Expected Confidence $\mathrm{Confidence}(S|T) := \Pr[S = \mathrm{ML}(T)]$

$$\Pr[S = \mathrm{ML}(T)] = \sum_{t\in\{0,1\}^m} \Pr[T = t, S = \mathrm{ML}(t)] = \mathbb{E}_{t\sim T(S,\mathcal{D})}\left[\max_s p_{S|T=t}(s)\right] \quad (6)$$

ML is of particular practical interest: given test results t, a physician wants to make a prediction. In this case, it makes sense to use the maximum likelihood predictor. The interpretation of $\mathrm{Confidence}$ is straightforward: it is the probability that the prediction is true (across all possible secrets).

Updating the priors. Both scoring functions described above compute the expectation relative to the test results of a score on the posterior distribution $p_{S|T=t}(s)$. After observing the test results, we are able to replace the prior distribution p_S by the posterior. By the rules of Bayesian computation, this update operation is commutative, i.e., the order in which designs d_1 and d_2 are tested does not matter, and compositional in the sense that we can test $\{d_1, d_2\}$ simultaneously with the same results.

[8] Laboratory errors in composing the pooled designs d could be modeled by correspondingly describing d by a random variable, or by including these errors into the random variable T.

Thus, we can decompose those steps and make different choices as we run tests (see the adaptive method below).

Although searching the space of all possible adaptive strategies would yield a prohibitive complexity of $\Omega(2^{2^m})$, it turns out that a simple adaptive strategy can yield provably near-optimal results. We describe an adaptive scheme in Algorithm 1 which greedily optimizes the criterion defined in Eq. (4).

Algorithm 1: (Greedy-Adaptive)

Input: Numbers n & m, test characteristics tpr & tnr, priors p_i for $i \in \{1, ..., n\}$;
Output: The sequence of tests to adaptively run in the lab;
Initialization: Set $k := m$ and set prior p_S using the p_i's;
while $k > 0$ **do**
    For each pool design d in $\{0,1\}^n$, compute $I(S, T(S, d))$;
    Select any $d^* \in \arg\max_d I(S, T(S, d))$;
    Observe result $T(S, d^*)$ of design d^* in the lab;
    Update p_S accordingly (see Eq. (3)) to the realization of d^* in the lab ;
    Decrease the number of remaining tests k by 1;
end

Leveraging the framework of adaptive sub-modularity [6], and assuming that the criterion defined by Eq. (4) is adaptive sub-modular[9], Algorithm 1 has the guarantee below.

Theorem 2. *Denote by* 'Algo' *an adaptive strategy. Let* $I(\mathrm{Algo})$ *be the expected mutual information obtained at the end of all* m *tests by running* Algo*, the expectation being taken over all* 2^m *outcomes of lab results. Denote by 'Optimal' the best (unknown) adaptive strategy. If we run Algorithm 1 for* m_1 *tests and Optimal for* m_2 *tests, we have:*

$$I(\text{Algorithm 1}) \geq \left(1 - e^{-\frac{m_1}{\alpha m_2}}\right) I(\text{Optimal}), \tag{7}$$

where α *is defined as follows: assume that our priors* p_i *are wrong, in the sense that there exist constants* c, d *with* $cp_i \leq p'_i \leq dp_i$ *for* $i \in \{1, ..., n\}$*, with* $c \leq 1$ *and* $d \geq 1$*, where* p'_i *denotes the true prior: we set* $\alpha := d/c$*.*

Remarks. Accordingly, Algorithm 1 is *(i)* robust to wrong priors and *(ii)* near-optimal in the sense that the ratio of its performance with that of the optimal strategy goes to 1 exponentially in the ratio of the numbers of tests run in each algorithm. For $\alpha = 1$ and $m_1 = m_2$, this yields $1 - e^{-1} \simeq 0.63$.

5 Testing at Scale with Bloom Filters

Our previous methods are effective, but they are prohibitively expensive for $n > 30$ patients. To address this, we present a randomized approach to selecting $\mathcal{D}$ by grouping patients into pools using Bloom filters [1].

Randomized test pooling may be attractive to practitioners because it is straightforward to understand and implement in the laboratory. The simplest method partitions n patients into random groups of equal size. Patients are either re-tested or reported positive if their group tests positive (**Single Pooling**). In [2], the authors propose an extension to this idea that inserts patients into two sets of pools, named **double pooling**, which offers impressive advantages at the same cost. We present a generalization of this idea that uses an array of Bloom filters to improve the error characteristics of the test. While Bloom filters have been considered for the low-prevalence COVID-19 testing problem [19, 12], current methods are based on a simple randomized encoding and decoding process that was designed for internet-scale applications where even linear time was prohibitive and where the keys are not known beforehand. This sacrifices accuracy. We now design an improved algorithm.

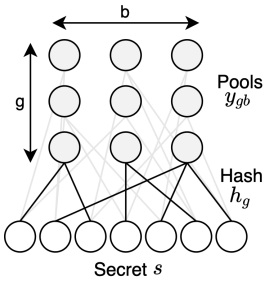


Figure 2: Test design. n people are shuffled g times and divided into b bins.

[9] Empirical validation in Appendix F.

Encoding. Bloom filters use universal random hash functions for load balancing because the streaming algorithm setting does not allow us to control the number of items in each group. Here, we can improve the filter with perfect load balancing. We divide the m tests into g groups of b pools. In each group, we assign the n patient samples to the b pools so that each pool contains n/b patients.[10]. This procedure constrains the multiset $\mathcal{D}$ of possible test designs. With uniform prior probabilities, we implement a perfectly load balanced hash by assigning each patient a number based on a permutation π_j of the integers $\{1, ...n\}$ Thus patient i is assigned to pool $h_j(i) := \pi_j(i) \bmod b$ in group j.

For non-uniform priors, we can resort to a variable load hash to balance total weights into pools. Due to the concavity of the entropy, the information gain is maximized if all pools have the same probability of testing positive. This is maximized for $1/2$, the mode of the binary entropy.

Load balancing implies Information Gain: Load balancing, as exhibited by our encoding, maximizes the information gain for a practical subset of constrained Bloom filter group test problems. Theorem 3 motivates Bloom filters in the context of our information theoretic framework. With a constraint on the number of samples in each pool, our load balancing hash allocation is the optimal pooling strategy provided that $\Pr[t_b = 1]$ is sufficiently small ($\sim 20\%$). We defer a detailed discussion to the appendix.

Theorem 3. *Assuming independent priors, the information gain $I(S, T)$ of the tests $\{t_1, ...t_b\}$, in a single Bloom filter row is maximized by having all the positive pool probabilities $\Pr[t_b = 0] = \prod_{i \in pool\, b} \Pr[s_i = 0]$ equal to a constant that depends only on tpr and tnr.*

Decoding for Perfect Tests. Assuming perfect tests, one can easily decode the pooled test results $t \in \{0, 1\}^{b \times g}$ because all patients in negative pools are healthy. We can then identify positive (and ambiguous) samples by eliminating healthy samples from positive pools, as described in the appendix. In the case where $g = 1$ and $g = 2$, we have the widely-used single pooling method and the recently-proposed double pooling method [2]. Assuming there are no false negative pool results, one can use the decoder to identify all positive samples and derive optimal dimensions $b \times g$ that minimize the number of tests, as shown in the below theorem:

Theorem 4. *Given m perfect tests, n patients and a uniform prior (prevalence) ρ, the decoder correctly identifies all positive samples and mislabels any negative sample with probability $\Pr[\hat{s}_i = 1 | s_i = 0] \leq \left(1 - e^{-\rho \frac{n}{b}}\right)^g$. The bound is minimized for $g = \frac{m}{n\rho} \log 2$ and $b = m/g$.*

The analysis borrows tools from regular Bloom filters and the results shown in [20]. Note that the problem with no test error and $1/2$ prevalence is a #P-complete restriction of #SAT, called monotone CNF [24]. Realistic tests with nontrivial fnr and fpr are technically more interesting. A natural idea is an algorithm dating back to [11] when decoding diseases from the QMR database.

Decoding via Message Passing. Indeed, false negative rates are often as high as 10%. The decoder fails for imperfect tests because even negative pools might contain positive samples. A small number of healthy pools might even test positive for some protocols (e.g. due to spurious contamination).

When viewed as a probabilistic graphical model we can interpret t_{gb} as a corrupted version of the true state y_{gb}. It is our goal to infer the secret s that produced t_{gb}. Belief propagation is a common technique to estimate the posterior distribution $p_{S|T=t}$ for a graphical model. Since our graphical model cannot be rewritten as a junction tree with narrow tree width there are no efficient exact algorithms. Instead, we resort to loopy belief propagation [14].

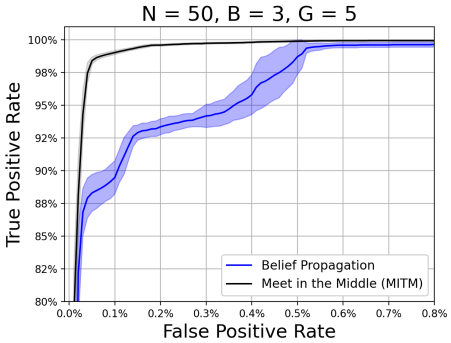


Figure 4: Classification using posteriors from different inference methods.

While inexact (loopy-BP isn't guaranteed to converge to the minimum) the resulting solution can classify samples as positive or negative with reasonable performance. While the degree of each pool

[10] or $\lfloor n/b \rfloor$ and $\lfloor n/b \rfloor + 1$ patients if n is not a multiple of b.

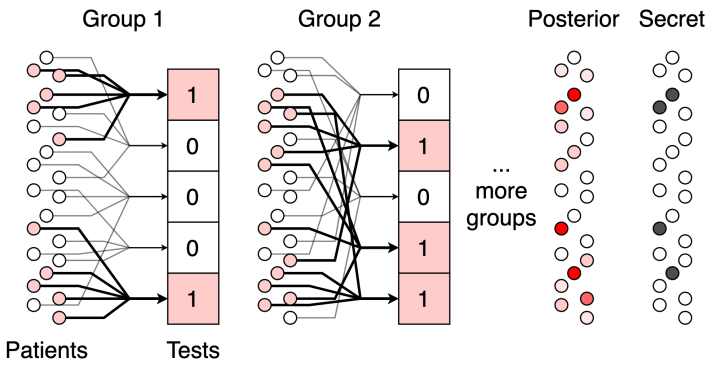


Figure 3: Intuition behind probabilistic decoding. In each group, we suspect that patients in positive pools are positive. If a patient falls within multiple positive pools, the likelihood that their test status is positive increases. Even if a false positive or negative occurs, we may still report the correct diagnosis thanks to information from other groups. This process is known as "error correction" and can be implemented with message passing or our MITM algorithm.

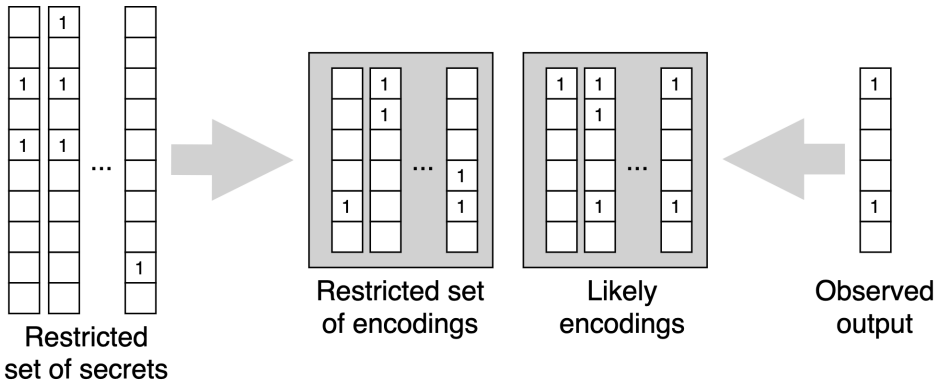


Figure 5: Intuition behind the MITM approach. If the prevalence is low, then we do not need to consider inputs with many positives. This restricts the set of possible secrets and the set of ways we can encode those secrets. The figure shows the inputs for at most 3 positives. Given a (potentially corrupted) output, there are only a small set of true encodings that could have produced that output - it is highly unlikely that every test had a false result. The two conditions "meet in the middle" to produce a small set of states. Our MITM algorithm efficiently approximates the posterior probabilities by summing over this restricted state space.

node is so high that the clique potential would naively involve an intractable number of states, the clique potentials have a simple form that permits an efficient implementation (detail in the appendix).

Decoding for Imperfect Tests: Meet-in-the-Middle (MITM). The structure of the problem also enables an efficient approximation to the exact solution in the (realistic) setting where the tests are fairly accurate and the disease prevalence is low. Low prevalence implies that there are relatively few "likely secrets" $s \in \{0,1\}^n$, because most s_i are 0 with high probability. Thus, we only need to consider secrets with a small number of positive patients.

Since the secrets concentrate in a small subset of $\{0,1\}^n$, we expect to see relatively few Bloom encodings $y \in \{0,1\}^t$ for low-prevalence problems. Furthermore, the output space is likely to be corrupted in relatively few ways. The true state y_{gb} is likely to be the same as the observed output t_{gb}, so we only need to consider states that are similar to the observed output. By restricting our attention to "likely secrets" and "likely outputs", we can reduce the $\mathcal{O}(2^n)$ complexity of the naive brute-force algorithm. This process constitutes a "meet in the middle" approach where we only need to consider

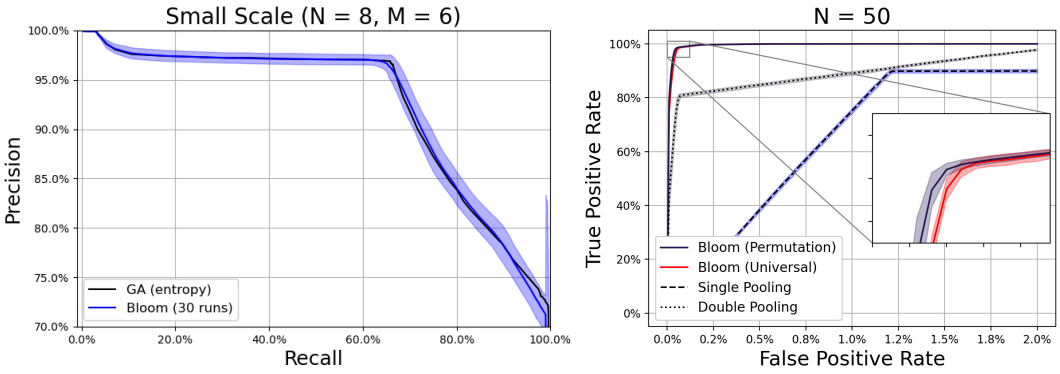


Figure 6: Comparison of group testing designs. We compare brute-force optimization by a genetic algorithm (GA) to our randomized Bloom design on small-scale experiments (left) and our Bloom design against baselines for a larger problem (right). All these experiments used MITM decoding.

a small number of Bloom encodings for inference (Figure 5). We show detailed pseudo-code in Algorithm 2, and prove Theorem 5 in Appendix C.5.

Theorem 5. *Let $\varepsilon > 0$ and consider the smallest k such that $f(k) := \sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{n-j} < \varepsilon$. Define $A(\varepsilon) := \sum_{i=0}^{k-1} \binom{n}{i}$, and $C(\varepsilon)$ the number of different encodings of secrets with less than k infected people. Obviously[11], $C(\varepsilon) < A(\varepsilon)$ and in practice $C(\varepsilon) \ll A(\varepsilon)$. For any test result $t \in \{0,1\}^m$, define $P := \sum_i t_i$ and $N := m - P$. Let $B(\varepsilon) := \sum_{prob[FP][FN] > \varepsilon} \binom{P}{FP}\binom{N}{FN}$, where*

$$prob[FP][FN] := (1-\mathrm{tnr})^{FP}\mathrm{tpr}^{P-FP}(1-\mathrm{tpr})^{FN}\mathrm{tnr}^{N-FN}. \tag{8}$$

Then there exists an algorithm with preprocessing time $\mathcal{O}((n+m)A(\varepsilon))$, space complexity $\mathcal{O}((n+m)C(\varepsilon))$ and query time $\mathcal{O}((n+m)\min(B(\varepsilon), C(\varepsilon)))$ that estimates $P[s_i|t]$ as a fraction $\tilde{P}[s_i \wedge t]/\tilde{P}[t]$ with an error less than $4\varepsilon/P[t] \leq 4\varepsilon/\tilde{P}[t]$, where the $\tilde{P}$ values are our estimation of P.

Illustrative values for MITM decoding. Using Stirling's formula $n! \sim \sqrt{2\pi n}(n/e)^n$, one can easily show that for a fraction $x \in [0,1]$, we have $f(xn) = o((2p^x)^n)$ when $n \to +\infty$. If $k := xn$ is such that $2p^x < 1$, i.e. $x > \log(2)/\log(1/p)$, then $f(k)$ will be exponentially small w.r.t. n. With our default $p = 0.1\%$ we only need to consider secrets with a fraction smaller than $x^* = \log_2(2)/\log_2(1/10^{-3}) \approx 10.03\%$ of infected people to yield negligible error. For $n = 60$, choosing $x = 13\%$ reduces[12] the search space of secrets from $2^{60} \approx 10^{18}$ to $\sum_{i=0}^{\lceil 60*0.13 \rceil - 1} \binom{60}{i} < 6 \cdot 10^7$ with an error $\varepsilon < (2p^{0.13})^{60} < 5 \cdot 10^{-6}$.

6 Numerical Experiments

We ran simulations to compare test designs for a large variety of group testing parameters (n, tpr/tnr, $b \times g$, prevalence) in the appendix. In this section we present results for a practical scenario where $\mathrm{tnr} = 0.9$, $\mathrm{tpr} = 0.99$, and 0.1% prevalence. In Figure 6, we compare the entropy-minimizing solution found by genetic algorithms with several Bloom filter designs. The Bloom filter performance closely resembles the optimal solution, albeit with higher variance. This validates our claim that the load balancing permutation hash implies a good information gain. We also apply our graphical model framework to 3×5 arrays of Bloom filters, single pooling and double pooling designs. We use the MITM technique to compute posteriors for all designs and we compare performance. While MITM provides the best results, computational constraints may demand belief propagation for situations where there are many positive group tests. In the high-prevalence scenario, belief propagation will still provide sufficient error correction for good diagnostic results (Figure 4). The vanilla bloom decoding (single and double pooling) is unnecessarily inaccurate, clearly implying the need for specific tailored algorithms.

[11] Because the code space is the image of the secret space w.r.t. the encoding function.

[12] We actually observe much tighter bounds in practice.

7 Conclusion & Future Work

We have presented a framework for group testing taking into account specifics of the current COVID-19 pandemic. It applies methods of probability and information theory to construct and decode multiplex codes spanning the relevant range of group sizes, establishing an interesting connection to Bloom filters and graphical models inference along the way. Our empirical results, more of which are included in the appendix, show that our methods lead to better codes than randomized pooling and popular approaches such as single pooling and double pooling.

Furthermore, we provide an approximate inference algorithm through Theorem 5 that outperforms the message passing approach for realistic parameter values by pruning the exponential search space. We also prove compute-time bounds on its error, highly useful in practice because they are strict.

We believe that the test multiplexing problem is an ideal opportunity for our community to make a contribution towards addressing the current global crisis. By firmly rooting this problem in learning and inference methods, we provide fertile ground for further development. As more information about test characteristics becomes available, we could take into account dependencies of tpr, tnr on pool size. The framework could be adapted to different objective functions, or linked to decision theory using suitable risk functionals, e.g., taking into account the downstream risk of misdiagnosing an individual with particular characteristics (comorbidities, probability of spreading the disease, etc.). It can be combined with the output of other methods providing individualized estimated of infection probabilities, to optimize pool allocation for non-uniform priors/prevalence. Statistical dependencies (e.g., for family members) could be taken into account. Finally, similar methods also permit addressing the problem of prevalence estimation. Further details as well as some concrete design recommendations derived from our methods are available in the appendix.

Acknowledgments

Gary Bécigneul is funded by the Max Planck ETH Center for Learning Systems. Benjamin Coleman and Anshumali Shrivastava are supported by NSF- 1652131, nsf-bigdata 1838177, AFOSR-YIPFA9550-18- 1-0152, Amazon Research Award, and ONR BRC grant for Randomized Numerical Linear Algebra.

8 Broader Impact

The motivation for this work was to help address the worldwide shortage of testing capacity for Cov-SARS-2. Testing plays a major role in breaking infection chains, monitoring the pandemic, and informing public policy. Countries successful at containing Covid-19 tend to be those that test a lot.[13]

On an individual level, availability of tests allows early and targeted care for high-risk patients. While treatment options are limited, it is believed that antiviral drugs are most effective if administered early on, since medical complications in later stages of the disease are substantially driven by inflammatory processes, rather than by the virus itself [23].

Finally, large-scale testing as enabled by pooling and multiplexing strategies may be a crucial component for opening up our societies and economies. People want to visit their family members in nursing homes, send their children to school, and the economy needs to function in order to secure supply chains and allow people to earn their livelihoods.[14]

However, the present work also poses some ethical challenges, of which we would like to list the below.

The first family concerns the accuracy of the tests. Indeed, when the number of tests and patients are equal, it is natural to compare the $\mathrm{tpr}/\mathrm{tnr}$ of the individual test to the $\mathrm{tpr}/\mathrm{tnr}$ of the individual results in our grouped test framework (obtained by marginalizing the posterior distribution). In some situations with unbalanced priors, the marginal $\mathrm{tpr}/\mathrm{tnr}$ of some people in the group could be lower than the test $\mathrm{tpr}/\mathrm{tnr}$, even if the test will be more successful overall. However, reporting the marginal individual results gives doctors a tool to decide whether further testing should be needed; hence we cannot rule out that individuals might be worse off by being tested in a group. We furthermore show in the appendix that some designs are more fair than others, in that the individual performances are more equally distributed.

The second family of concerns, directly resulting from the first, is the responsibility of the doctor when assigning the people to batches and giving them prior probabilities (using another model). The assignment of people in batches should be dealt with in a future extension of our framework, while the sensitivity of our protocols to priors should be studied in more depth. The adaptive framework may be more robust with respect to the choice of priors than the non-adaptive one.

Finally, the possibility of truly large scale testing may allow countries with sufficient financial resources to perform daily testing of large populations, with significant advantages for economic activity. This, in turn, could exacerbate economic imbalances.

References

[1] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[2] Andrei Z Broder and Ravi Kumar. A note on double pooling tests. *arXiv preprint arXiv:2004.01684*, 2020.

[3] Chun Lam Chan, Sidharth Jaggi, Venkatesh Saligrama, and Samar Agnihotri. Non-adaptive group testing: Explicit bounds and novel algorithms. *IEEE Transactions on Information Theory*, 60(5):3019–3035, 2014.

[4] Mahdi Cheraghchi, Amin Karbasi, Soheil Mohajer, and Venkatesh Saligrama. Graph-constrained group testing. *IEEE Transactions on Information Theory*, 58(1):248–262, 2012.

[5] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[6] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

[13] `https://ourworldindata.org/coronavirus-testing`

[14] `http://www.oecd.org/coronavirus/policy-responses/testing-for-covid-19-a-way-to-lift-confinement-restrictions-89756248/`

[7] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272, 2005.

[8] Xi He, Eric H. Y. Lau, Peng Wu, Xilong Deng, Jian Wang, Xinxin Hao, Yiu Chung Lau, Jessica Y. Wong, Yujuan Guan, Xinghua Tan, Xiaoneng Mo, Yanqing Chen, Baolin Liao, Weilie Chen, Fengyu Hu, Qing Zhang, Mingqiu Zhong, Yanrong Wu, Lingzhai Zhao, Fuchun Zhang, Benjamin J. Cowling, Fang Li, and Gabriel M. Leung. Temporal dynamics in viral shedding and transmissibility of COVID-19. *Nature Medicine*, 26(5):672–675, 2020.

[9] FK Hwang and VT Sós. Non-adaptive hypergeometric group testing. *Studia Sci. Math. Hungar*, 22(1-4):257–263, 1987.

[10] Piotr Indyk, Hung Q Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1126–1142. SIAM, 2010.

[11] Tommi S Jaakkola and Michael I Jordan. Variational probabilistic inference and the qmr-dt network. *Journal of artificial intelligence research*, 10:291–322, 1999.

[12] Tomas Janousek. https://github.com/liskin/covid19-bloom. `https://github.com/liskin/covid19-bloom`, 2020.

[13] Emanuel Knill, William J Bruno, and David C Torney. Non-adaptive group testing in the presence of errors. *Discrete applied mathematics*, 88(1-3):261–290, 1998.

[14] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[15] Stefan Lohse, Thorsten Pfuhl, Barbara Berkó-Göttel, Jürgen Rissland, Tobias Geißler, Barbara Gärtner, Sören L Becker, Sophie Schneitler, and Sigrun Smola. Pooling of samples for testing for SARS-CoV-2 in asymptomatic people. *The Lancet Infectious Diseases*, 2020. https://doi.org/10.1016/S1473-3099(20)30362-5.

[16] Yi Lu, Andrea Montanari, Balaji Prabhakar, Sarang Dharmapurikar, and Abdul Kabbani. Counter braids: a novel counter architecture for per-flow measurement. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):121–132, 2008.

[17] Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47(4):173–180, 1993.

[18] Arya Mazumdar. Nonadaptive group testing with random set of defectives. *IEEE Transactions on Information Theory*, 62(12):7522–7531, 2016.

[19] Monika Mich Cechova. Bloom-filter inspired testing of pooled samples (and splitting of swabs!). April 1, 2020.

[20] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[21] Nikhil S Padhye. Reconstructed diagnostic sensitivity and specificity of the rt-pcr test for covid-19. *medRxiv*, 2020.

[22] Michael Schmidt, Sebastian Hoehl, Annemarie Berger, Heinz Zeichhardt, Kai Hourfar, Sandra Ciesek, and Erhard Seifried. FACT - Frankfurt adjusted COVID-19 testing - a novel method enables high-throughput SARS-CoV-2 screening without loss of sensitivity. *medRxiv*, 2020.

[23] Matthew Zirui Tay, Chek Meng Poh, Laurent Rénia, Paul A. MacAry, and Lisa F. P. Ng. The trinity of COVID-19: immunity, inflammation and intervention. *Nature Reviews Immunology*, 20(6):363–374, 2020.

[24] Radislav Vaisman, Ofer Strichman, and Ilya Gertsbakh. Model counting of monotone cnf formulas with spectra.

[25] Gaolian Xu, Debbie Nolder, Julien Reboud, Mary Oguike, Donelly van Schalkwyk, Colin Sutherland, and Jonathan Cooper. Paper-origami-based multiplexed malaria diagnostics from whole blood. *Angewandte Chemie International Edition*, 55, 08 2016.

[26] Idan Yelin, Noga Aharony, Einat Shaer-Tamar, Amir Argoetti, Esther Messer, Dina Berenbaum, Einat Shafran, Areen Kuzli, Nagam Gandali, Tamar Hashimshony, Yael Mandel-Gutfreund, Michael Halberthal, Yuval Geffen, Moran Szwarcwort-Cohen, and Roy Kishony. Evaluation of COVID-19 RT-qPCR test in multi-sample pools. *Clinical Infectious Diseases*, 2020.

A List of All Notations

We use upper case letters exclusively for random variables (r.v.), except for mutual information I and entropy H.

- n: number of patient samples;
- m: number of tests to run in the lab;
- g: number of groups; b: number of pools; $m = g \cdot b$;
- $s \in \{0,1\}^n$: the *secret* to unveil, with $s_i = 1$ if and only if patient sample i is positive (infected);
- S: r.v. over possible values of s whose law describes the current information we have about s;
- $d \in \{0,1\}^n$: a pool design, with $d_i = 1$ if and only if patient sample i belongs to pool design d;
- $\mathcal{D} \in (\{0,1\}^n)^m$: random multiset describing the pool designs output by the strategy;
- $t \in \{0,1\}^m$: lab result of a list of m tests;
- T: r.v. over possible values of t describing lab results;
- tpr: true positive rate, sensitivity, hit rate, detection rate, recall;
- tnr: true negative rate, specificity, correct rejection rate, selectivity;
- $p_i \in [0,1]$: prior probability of infection of patient sample i;
- $\Pr[A]$: probability of event A to happen;
- $p_S(s) \in [0,1]$: probability of secret $s \in \{0,1\}^n$ to be the correct one, according to the law p_S of r.v. S.

B Future Work & Additional Considerations

B.1 Fairness Considerations

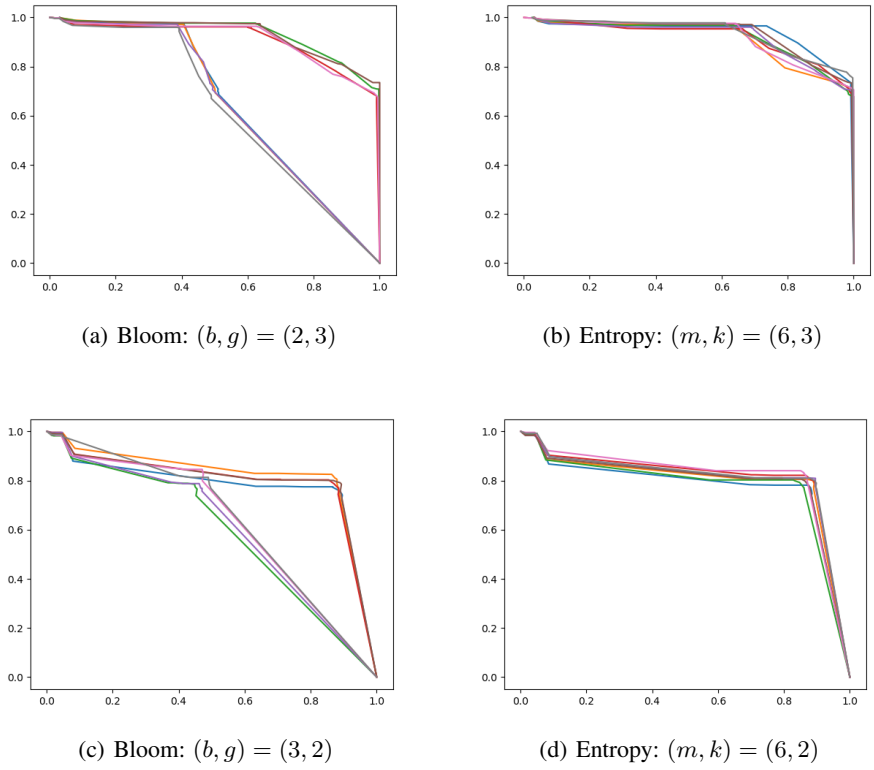
Figure 7 illustrates different Precision-Recall curves for different patients, across different methods/-parameters. In particular, it shows that the Bloom encoder gives more uneven estimation performances across patients, compared to the Entropy encoder.

B.2 Others

Different objective functions. We have used the number of tests and samples as given, and then optimized a conditional entropy. However, from a practical point of view, other quantities are relevant and may need to be included in the objective, *e.g.* the expectation (over a population) of the waiting time before an individual is "cleared" as negative (and can then go to work, visit a nursing home, or perform other actions which may require a confirmation of non-infectiousness).

Semi-adaptive tests. Instead of performing m consecutive tests, one could do them in k batches of respective sizes $m_1, ..., m_k$ satisfying $m_1 + ... + m_k = m$. Adaptivity over the sequence of length k could be handled greedily as in Algorithm 1, except that instead of selecting a single pool design d^*, we would select m_i designs at the i^{th} step. We named this semi-adaptive algorithm the *k-greedy* strategy.

Further practical considerations. A good practical strategy could be to perform one round of pooled tests to disjoint groups every morning as individuals arrive at work, being evaluated during work hours. Those who are in a positive group (adaptively) get assigned to a second pool design tested later, which can consist of a non-adaptive combination of multiple designs, tested over night. They receive the result in the morning before they go to work, and if individually positive, they enter quarantine. If the test is so sensitive that it detects infections even before individuals become contagious (which may be the case for PCR tests), such a strategy could avoid most infections at work.



(a) Bloom: $(b, g) = (2, 3)$

(b) Entropy: $(m, k) = (6, 3)$

(c) Bloom: $(b, g) = (3, 2)$

(d) Entropy: $(m, k) = (6, 2)$

Figure 7: Precision-Recall curves for different patients, for a set of $n = 8$ patients, a prevalence $p = 10^{-3}$ and a total of $m = 6 = b \times g$ tests. Each plot depicts 8 curves: one per patient. Recall that b denotes the number of bins, g the number of groups, and k the maximum number of times one patient swab can be tested. Hence one should compare (a) with (b), and (c) with (d). "Bloom" denotes the use of the Bloom encoding described in Section 5 while "Entropy" denotes the use of the Conditional Entropy / Mutual Information encoder described in Section 4. In both comparisons, we observe that the Entropy encoder yields more similar PR curves across patients, compared to the Bloom encoder.

Dependencies between tpr, tnr and pool size. The reliability of tests may vary with pool size. In our notation, the outcome of the tests is a random variable that need not only depend on whether one person is sick ($\mathbf{1}_{\langle d,s\rangle>0}$) but it may also depend on the number of tested people $|d|$ and the number of sick people $\langle d, s\rangle$ (cf. Footnote 7); it could even assign different values of tpr and tnr to different people. The tpr may in practice be an increasing function of the proportion of sick people $\langle d, s\rangle/|d|$.

Estimating prior infection probabilities. Currently, we start with a factorized prior of infection that not only assumes independence between the tested patients but is also oblivious to the individual characteristics. We could, however, build a simple ML system that estimates the prior probabilities based on a set of features such as: job, number of people living in the same household, number of children, location of home, movement or contact data, etc.[15] Those prior probabilities can then be readily used by our approach to optimize the pool designs, and the ML system can gradually be improved as we gather more test results.

Prevalence estimation. Similar methods can be applied to the question of estimating prevalence. Note that this is an easier problem in the sense that we need not necessarily estimate which individuals are positive, but only how many.

[15] Subject to privacy considerations.

C Proofs

C.1 Theorem 1

Statement: Under the condition that the population never be stuck in a local optimum, the evolutionary strategy using the Luby sequence $(b, b, 2b, b, b, 2b, 4b, b, b, 2b, b, b, 2b, 4b, 8b, ...)$ for restarts yields a Las Vegas algorithm that restarts optimally [17] to achieve any target score threshold.

Proof. Let us remind the main result we use on optimal restarts [17]: the simple Luby sequence of times of restart given by $(b, b, 2b, b, b, 2b, 4b, b, b, 2b, b, b, 2b, 4b, 8b, ...)$ is optimal (up to a log factor) for Las Vegas algorithms (i.e. randomized algorithms that always provide the correct answer when they stop, but may have non-deterministic running time). Our theorem is a direct consequence of conceptually casting our problem as a Las Vegas algorithm: indeed, we seek to optimize a fitness function f. For a given threshold $A > 0$, we can replace the maximization of f by the condition $f > A$. Applying the result of [17] for an exhaustive family of thresholds yields the desired result. □

C.2 Theorem 2

We wish to invoke Theorem 1 of [6]. In order to do so, we need to prove that the conditional entropy which we introduced in Eq. (2) is *adaptive monotone*. Concerning *adaptive sub-modularity*, we make it an assumption upon which our results is conditioned, and validate it numerically with high precision for small values of n (see Appendix F). Direct respective correspondence between our notations and that of [6] is given by:

- Pool designs d : items e;
- Test results T : realizations Φ;
- Set $\mathcal{D}$ of selected designs : set $E(\pi, \Phi)$ of selected items by policy π;
- $H(p_{S|T=t})$: $f(E(\pi, \Phi), \Phi)$;
- $H(S \mid T)$: $f_{avg} := \mathbb{E}[f(E(\pi, \Phi), \Phi)]$.

This allows one to define, following Definition 1 of [6], the conditional expected marginal benefit of a pool design d given results t as:

$$\Delta(d) := -[H(S \mid R(S, d)) - H(S)]. \tag{9}$$

It represents the marginal gain of information obtained, in expectation, by observing the outcome of d at a given stage (this stage being defined by p_S, i.e. after having observed test results t).

Adaptive monotonicity holds if $\Delta(d) \geq 0$ for any d.

Adaptive sub-modularity holds if for any two sets of results t and t' such that t is a *sub-realization*[16] of t', for any pool design d: $\Delta(d \mid t) \geq \Delta(d \mid t')$.

The below lemma concludes the proof.

Lemma. With respect to Δ defined in Eq. (9), adaptive monotonicity holds.
Proof. Adaptive monotonicity is a consequence of the "information-never-hurts" bound $H(X \mid Y) \leq H(X)$ [5].

□

C.3 Theorem 3

We are interested in the information $I(S, T)$ for a single Bloom filter row with B cells. Because each test in the row contains a disjoint set of patients, $I(S, T)$ is the sum of the information for each test

[16] *i.e.* there exist $\mathcal{D}$ and $\mathcal{D}'$ such that $T(S, \mathcal{D}) = t$, $T(S, \mathcal{D}') = t'$ and $\mathcal{D} \subset \mathcal{D}'$.

(i.e. the t_b random variables are independent and there are no cross terms).

$$I(S,T) = H(T) - H(T|S) \tag{10}$$

$$= \sum_{b=1}^{B} H(t_b) - \sum_{b=1}^{B} H(t_b|S_{\text{patients } \in b}) \tag{11}$$

Using the basic definition of $H(t_b)$, we have that

$$H(t_b) = -\sum_{t\in\{0,1\}} \Pr(t_b = t)\log\Pr(t_b = t) \tag{12}$$

We use the fact that

$$\Pr(t_b = t) = \sum_{y\in\{0,1\}} \Pr(t_b = t|y_b = y)\Pr(y_b = y) \tag{13}$$

Since the relationship between the ideal test results y_b and the patient statuses s_i is deterministic, conditioning on s_i is the same as conditioning on y_b. In particular, one can write $\Pr(y_b = 0) = \prod(1 - p_i)$. Observe that $tpr = \Pr(t_b = 0|y_b = 0)$ and $tnr = \Pr(t_b = 1|y_b = 1)$ and let $\rho_b = \Pr(y_b = 1)$. This gives us a simple expression for $\Pr[t_b = t]$ and thus

$$H(t_b) = -\left((1 - 2tnr)\rho_b + tnr\right)\log_2\left((1 - 2tnr)\rho_b + tnr\right) \tag{14}$$

$$-\left((2tpr - 1)\rho_b + 1 - tpr\right)\log_2\left((2tpr - 1)\rho_b + 1 - tpr\right) \tag{15}$$

We approach the second term $H(t_b|S_{\text{patients } \in b})$ the same way.

$$H(t_b|S_{\text{patients } \in b}) = -\sum_{t\in\{0,1\}}\sum_{y\in\{0,1\}} \Pr[t_b = t, y_b = y]\log\Pr[t_b = t|y_b = y] \tag{16}$$

$$= -\sum_{t\in\{0,1\}}\sum_{y\in\{0,1\}} \Pr[t_b = t|y_b = y]\Pr[y_b = y]\log\Pr[t_b = t|y_b = y] \tag{17}$$

$$= -(1 - \rho_b)(tnr\log_2(tnr) + (1 - tnr)\log_2(1 - tnr)) \tag{18}$$

$$-\rho_b(tpr\log_2(tpr) + (1 - tpr)\log_2(1 - tpr)) \tag{19}$$

Put $\beta = (tnr\log_2(tnr)+(1-tnr)\log_2(1-tnr))$ and $\alpha = (tpr\log_2(tpr)+(1-tpr)\log_2(1-tpr))$. Then, the information $I(S,T)$ is equal to

$$I(S,T) = \sum_{b=1}^{B} -\left((1 - 2tnr)\rho_b + tnr\right)\log_2\left((1 - 2tnr)\rho_b + tnr\right) \tag{20}$$

$$-\left((2tpr - 1)\rho_b + 1 - tpr\right)\log_2\left((2tpr - 1)\rho_b + 1 - tpr\right) \tag{21}$$

$$-\left[-(1 - \rho_b)\beta - \rho_b\alpha\right] \tag{22}$$

Information is Concave in ρ: To show that there is a single, constant, and optimal probability for each group test to be positive, we prove that $I(S,T)$ is concave in ρ. It is sufficient to show that each term $I(S,t_b)$ in the sum is concave in ρ_b.

Taking derivatives, we have

$$\frac{d}{d\rho_b}I(S,t_b) = -\frac{1}{\log(2)}(1 - 2tnr)\log((1 - 2tnr)\rho_b + tnr) - \frac{1}{\log(2)}(1 - 2tnr) \tag{23}$$

$$-\frac{1}{\log(2)}(2tpr - 1)\log((2tpr - 1)\rho_b + 1 - tpr) - \frac{1}{\log(2)}(2tpr - 1) \tag{24}$$

$$-\beta + \alpha \tag{25}$$

The second derivative is

$$\frac{d^2}{d\rho_b^2} I(S, t_b) = -\frac{1}{\log(2)} \left(\frac{(1-2tnr)^2}{(1-2tnr)\rho_b + tnr} + \frac{(2tpr-1)^2}{(2tpr-1)\rho_b - tnr + 1} \right) \tag{26}$$

We wish to show that $\frac{d^2}{d\rho_b^2} I(S, t_b) \leq 0$, which we will do by proving that the two fractions are both positive. The squared terms in the numerators are positive, as is the expression $(2tpr-1)\rho_b + 1 - tnr$ because $tnr > 0.5$. This leaves the $(1-2tnr)\rho_b + tnr$ term in the denominator. This term is linear in $\rho_b \in [0, 1]$, with a minimum of 1 - tnr. Thus, $I(S, T)$ is concave.

Optimal Value of ρ_b: Since the information is concave, there is an optimal value of ρ_b that maximizes the information gain from each grouped test. Since $H(t_b)$ depends only on tpr, tnr and ρ_b, it is easy to see that this value is constant and the same for all groups b. This proves the theorem.

However, it is of practical importance to find or approximate the optimal value of ρ_b. If one wanted to load balance a variety of (possibly different) priors into groups that have the optimal probability of testing positive, one needs to know the desired value of ρ_b. We obtain the following equation by setting the derivative to zero:

$$(2tnr-1)\log((1-2tnr)\rho_b + tnr) + (1-2tpr)\log((2tpr-1)\rho_b + 1 - tpr) = c \tag{27}$$

where $c = (1-2tnr) + (2tpr-1) + \log(2)(\beta - \alpha)$. One can obtain the optimal ρ_b by numerically solving this equation. When $tpr = tnr$, we have $c = 0$ and the optimal value of $\rho_b = 0.5$.

C.4 Theorem 4

We prove the theorem using an analysis that is similar to the one for standard Bloom filters. The Bloom filter decoder identifies a sample as positive if all of the pools containing the sample are positive. It is easy to see that the decoder cannot produce false negatives under perfect tests, because each positive sample will always generate a positive pool result. We now analyze the systemic false positives introduced by the pooling operation. Each pool contains either $\frac{N}{B}$ or $\frac{N}{B} - 1$ patients, where the latter situation occurs when B does not perfectly divide N and there are a few "leftover" elements. Thus, any given sample will share a bin with up to $\frac{N}{B} - 1$ other elements, each of which has independent probability ρ of testing positive. To correctly identify a sample as negative, we require that all of these $\frac{N}{B} - 1$ samples also test negative. Hence the probability that our sample will not collide with a positive sample is at least

$$(1-\rho)^{\frac{N}{B}-1} \leq \exp\left(-\rho(\frac{N}{B} - 1)\right) \tag{28}$$

The -1 arises from the fact that the sample cannot collide with itself. This analysis holds for a single Bloom filter row, but we have G independent opportunities to land in a negative pool. The rows are independent because independent random hash functions are used to form the groupings. The probability that we collide with a positive in all G groups is at most

$$(1 - (1-\rho)^{\frac{N}{B}-1})^G \tag{29}$$

This expression gives the probability that we fail to identify the sample correctly. We want to bound the failure probability p_f and choose parameters that minimize the bound. Note that we replaced $\frac{N}{B} - 1$ with $\frac{N}{B}$ - the inequality still holds because $(1-\rho) < 1$.

$$p_f = (1 - (1-\rho)^{\frac{N}{B}-1})^G \leq \left(1 - \exp\left(-\rho\frac{N}{B}\right)\right)^G \tag{30}$$

The optimal dimensions for the Bloom filter come from minimizing the upper bound. We use the relation $M = B \times G$ to put p_f in terms of M and G.

$$p_f \leq \left(1 - \exp\left(-\rho\frac{N}{M}G\right)\right)^G \tag{31}$$

We find that the optimal $G = \frac{M}{N\rho}\log 2$.

C.5 Theorem 5

Notations. we use tilda $\tilde{x}$ to denote the estimation of a quantity x.

Error Bounds and Confidence Levels. Given a test result $t \in \{0,1\}^m$, and a patient $i \in \{1, ..., n\}$, we seek to estimate $P[s_i|t]$, *i.e.* the probability of patient sample s_i being positive. We can rewrite:

$$P[s_i|t] = \frac{P[s_i, t]}{P[s_i, t] + P[\bar{s_i}, t]} =: \frac{\lambda}{\lambda + \mu}, \tag{32}$$

where we defined $\lambda := P[s_i, t]$ and $\mu := P[\bar{s_i}, t]$. Hence, we seek to estimate λ, resp. μ. We use the term "code space" to refer to the space $\{0,1\}^m$ of encodings of secrets $s \in \{0,1\}^n$. We write λ and μ in terms of the joint distribution of secrets s, encodings c, and results t. Summing across the code space yields:

$$\lambda = \sum_c P[s_i, t, c] = \sum_c P[t|s_i, c]P[s_i, c] = \sum_c P[t|c]P[s_i, c], \tag{33}$$

where the last equality comes from conditional independence of t and s_i w.r.t. c. We now seek to estimate $a(c) := P[t \mid c]$ and $b(c) := P[s_i, c]$.

Suppose that we have (under-)estimates $\tilde{a}$ and $\tilde{b}$ such that $0 \leq \max_c(a(c) - \tilde{a}(c)) \leq \varepsilon$ and $0 \leq \sum_c(b(c) - \tilde{b}(c)) \leq \varepsilon$. Later, we will describe how to obtain these estimates. For now, observe that we can (under-)estimate $\lambda = \sum_c a(c)b(c)$ with $\tilde{\lambda} := \sum_c \tilde{a}(c)\tilde{b}(c)$, with the following error bound[17]:

$$\begin{aligned}
0 \leq \lambda - \tilde{\lambda} &= \sum_c a(c)b(c) - \sum_c \tilde{a}(c)\tilde{b}(c) && (34)\\
&= \sum_c a(c)(b(c) - \tilde{b}(c)) + \sum_c (a(c) - \tilde{a}(c))\tilde{b}(c) && (35)\\
&\leq \sum_c (b(c) - \tilde{b}(c)) + \max_c (a(c) - \tilde{a}(c)) && (36)\\
&\leq 2\varepsilon, && (37)
\end{aligned}$$

and similarly $0 \leq \mu - \tilde{\mu} \leq 2\varepsilon$, which would imply $0 \leq (\lambda + \mu) - (\tilde{\lambda} + \tilde{\mu}) \leq 4\varepsilon$; however, we can obtain a tighter upper bound of 3ε by noticing that $\sum_c P[s_i, c] + \sum_c P[\bar{s_i}, c] = \sum_c P[c] \leq 1$, yielding a true $P[s_i|t]$ in the below (arithmetic) interval:

$$P[s_i|t] \in [\tilde{\lambda}, \tilde{\lambda} + 2\varepsilon] / [\tilde{\lambda} + \tilde{\mu}, \tilde{\lambda} + \tilde{\mu} + 3\varepsilon] = \left[\frac{\tilde{\lambda}}{\tilde{\lambda} + \tilde{\mu} + 3\varepsilon}, \frac{\tilde{\lambda} + 2\varepsilon}{\tilde{\lambda} + \tilde{\mu}} \right]. \tag{38}$$

We want to bound the size of this interval to show that our estimate is close to the true $P[s_i|t]$. We do this via a Taylor alternate series:

$$\begin{aligned}
\frac{\tilde{\lambda} + 2\varepsilon}{\tilde{\lambda} + \tilde{\mu}} - \frac{\tilde{\lambda}}{\tilde{\lambda} + \tilde{\mu} + 3\varepsilon} &\leq \frac{2\varepsilon}{\tilde{\lambda} + \tilde{\mu}} + \frac{3\varepsilon\tilde{\lambda}}{(\tilde{\lambda} + \tilde{\mu})^2} && (39)\\
&= \varepsilon \frac{5\tilde{\lambda} + 2\tilde{\mu}}{(\tilde{\lambda} + \tilde{\mu})^2} && (40)\\
&\leq \frac{5\varepsilon}{\tilde{\lambda} + \tilde{\mu}}, && (41)
\end{aligned}$$

which concludes the proof that we can estimate $P[s_i|t]$ with error less than $5\varepsilon/\tilde{P}[t]$, where $\tilde{P}[t] := \tilde{\lambda} + \tilde{\mu}$, given estimates $\tilde{a}$ and $\tilde{b}$. Hence, we only need to construct the under-estimates $\tilde{a}$ and $\tilde{b}$ such that $0 \leq \max_c a(c) - \tilde{a}(c) \leq \varepsilon$ and $0 \leq \sum_c b(c) - \tilde{b}(c) \leq \varepsilon$. To construct $\tilde{b}(c)$, assume we have an integer k such that $f(k) := \sum_{j=k}^n \binom{n}{j} p^j (1-p)^{n-j} < \varepsilon$. Let

$$\tilde{b}(c) := \sum_{\substack{s \in \{0,1\}^n \\ \sum_j s_j < k \\ enc(s) = c}} P[s_i, c]. \tag{42}$$

[17]Note that for any c, we have: $a(c), b(c) \leq 1$, and that $\tilde{a}(c) \leq a(c)$ and $\tilde{b}(c) \leq b(c)$ because they are under-estimates.

Then,

$$\sum_c b(c) - \tilde{b}(c) \leq \sum_c \sum_{\substack{s\in\{0,1\}^n \\ \sum_j s_j \geq k}} P[s_i, c] \tag{43}$$

$$\leq \sum_{\substack{s\in\{0,1\}^n \\ \sum_j s_j \geq k}} P[s_i] \tag{44}$$

$$= f(k) \tag{45}$$

$$\leq \varepsilon. \tag{46}$$

Similarly, let

$$\tilde{a}(c) := \mathbf{1}_{\{prob[FP][FN] > \varepsilon\}} prob[FP][FN] \tag{47}$$

where $\mathbf{1}$ is the indicator function. The $prob[FP][FN]$ term is the probability $P[t|c]$ of getting a particular (corrupted) output t given a (true) code c. This term is defined as follows, where FP, FN, N, P are the number of false positives FP, false negatives FN, total negatives N and total positives P in the output t when compared with c. Note that $N + P = m$ and that $prob[FP][FN] = a(c)$. We use the term $prob[FP][FN]$ only for **notational convenience** to show that $a(c)$ depends on FP, FN, N and P.

$$prob[FP][FN] := (1 - \mathrm{tnr})^{FP} \mathrm{tpr}^{P-FP} (1 - \mathrm{tpr})^{FN} \mathrm{tnr}^{N-FN} = a(c). \tag{48}$$

Then,

$$a(c) - \tilde{a}(c) = prob[FP][FN] - \mathbf{1}_{\{prob[FP][FN] > \varepsilon\}} prob[FP][FN] \tag{49}$$

$$\leq \mathbf{1}_{\{prob[FP][FN] \leq \varepsilon\}} prob[FP][FN] \tag{50}$$

$$\leq \varepsilon. \tag{51}$$

This concludes our presentation of the estimators $\tilde{a}(c)$ and $\tilde{b}(c)$. Note that we presented a confidence interval together with a bound on its size, *i.e.* we showed that the true value $P[s_i|t]$ is within an interval that depends on the observed quantity $\tilde{P}[s_i|t]$. However, we can also provide an interval for the observed quantity as a function of the true value:

$$\tilde{P}[s_i|t] \in [\lambda - 2\varepsilon, \lambda] / [\lambda + \mu - 3\varepsilon, \lambda + \mu] = \left[\frac{\lambda - 2\varepsilon}{\lambda + \mu}, \frac{\lambda}{\lambda + \mu - 3\varepsilon} \right], \tag{52}$$

whose size can be bounded by:

$$\frac{\lambda}{\lambda + \mu - 3\varepsilon} - \frac{\lambda - 2\varepsilon}{\lambda + \mu} = \frac{3\varepsilon\lambda}{(\lambda + \mu)^2} + \frac{2\varepsilon}{\lambda + \mu} + \frac{3\lambda\varepsilon^2}{(\lambda + \mu)^3} \sum_{j=0}^{+\infty} \left(\frac{\varepsilon}{\lambda + \mu} \right)^j \tag{53}$$

$$\leq \frac{5\varepsilon}{\lambda + \mu} + \frac{3\lambda\varepsilon^2}{(\lambda + \mu)^3} \frac{1}{1 - \frac{\varepsilon}{\lambda + \mu}} \tag{54}$$

$$\leq \frac{5\varepsilon}{\lambda + \mu} + \frac{\varepsilon}{\lambda + \mu} \tag{55}$$

$$= \frac{6\varepsilon}{\lambda + \mu} \tag{56}$$

$$= \frac{6\varepsilon}{P[t]}, \tag{57}$$

where we assumed $\varepsilon < \lambda/4$ to justify that $\frac{3\lambda\varepsilon^2}{(\lambda+\mu)^3} \frac{1}{1-\frac{\varepsilon}{\lambda+\mu}} < \frac{\varepsilon}{\lambda+\mu}$. One might also be interested in the error rather than a confidence interval. Recall that $\tilde{P}[t] := \tilde{\lambda} + \tilde{\mu} \leq \lambda + \mu = P[t]$ If we want the error of our estimator, then one can easily show that $|\tilde{P}[s_i|t] - P[s_i|t]| \leq 4\varepsilon / P[t] \leq 4\varepsilon / \tilde{P}[t]$. In practice, $\tilde{P}[t]$ can be computed to get upper bounds on the estimation error and confidence level.

We will now present an algorithm that efficiently computes these estimators. In our algorithm, $A(\epsilon)$ is the number of secrets with at most k nonzeros, $C(\epsilon)$ is number of codes produced by this restricted

set of k-sparse secrets, and $B(\epsilon)$ is a set of probable ideal codes for the potentially-corrupted output t that we observe.

Algorithm 2: (MITM Decoder)

Input : n & m, tpr & tnr, prevalence p, test results $t \in \{0,1\}^m$, precision parameter ε;
Output : Estimates $\tilde{P}[s_i|t]$ for $i \in \{1,...,n\}$ with $|\tilde{P}[s_i|t] - P[s_i|t]| \leq 4\varepsilon/P[t]$;

Preprocessing: (independent of results t)
Compute k such that $f(k) < \varepsilon$ and initialize $\mathcal{C} = \emptyset$;
Enumerate all the codes $c := enc(s)$ for s with less than k positives[a];
Use these codes to approximate $\tilde{P}[s_i, c]$ and $\tilde{P}[\bar{s_i}, c]$ using the formula for $\tilde{b}(c)$ in Eq. (42). Store the results in $\mathcal{C}$;
Query: (dependent upon results t)
Compute $P := \sum_i t_i$, $N := m - P$ and $a(c)$ (see Eq. (48)) for $FP \leq P$, $FN \leq N$;
Compute $B(\varepsilon) := \sum_{a(c)>\varepsilon} \binom{P}{FP}\binom{N}{FN}$;
if $C(\varepsilon) < B(\varepsilon)$ **then**
  Estimate $P[s_i, t]$ (resp. $P[\bar{s_i}, t]$) by iterating over the codes c in $\mathcal{C}$ and reporting $\sum_c a(c)\tilde{P}[s_i, c]$;
else
  Enumerate[b] codes c such that $a(c) > \varepsilon$;
end
Output final estimates $\tilde{P}[s_i|t] := \tilde{P}[s_i, t]/(\tilde{P}[s_i, t] + \tilde{P}[\bar{s_i}, t])$;

[a] This yields a set of size $C(\varepsilon)$ computed in time $A(\varepsilon)$.

[b] We can recursively enumerate these codes in time $B(\varepsilon)$ since $a(c)$ is monotonic w.r.t. both variables, by starting the enumeration at $c := t$, *i.e.* $FP = FN = 0$, and recursively increment FP or FN.

Complexity Analysis. Since the outcome of a test t is conditionally independent to s w.r.t. c, we can pre-compute all encodings $c := enc(s)$ for s belonging to the reduced search space of size $A(\varepsilon) := \sum_{i=0}^{k-1} \binom{n}{i}$. Saving all these resulting encodings with a hashmap or a set structure gives a space of complexity proportional to $C(\varepsilon) \leq A(\varepsilon)$, since the output function image of an input set is always smaller than (or equal to) the size of the input set. Finally, at query time, we seek to estimate $P[s_i|t]$. Note that we have pruned two search spaces: the space of encodings of $A(\varepsilon)$ many secrets, reduced from 2^m to $C(\varepsilon)$, and the space of codes c such that for our given t, $prob[FP][FN] > \varepsilon$, reduced from 2^m to $B(\varepsilon)$. Given a test result t, we can compute N, P in $\mathcal{O}(m)$ operations, which then allows us to compute $B(\varepsilon)$ for this t. Also note that we approximate $P[s_i, t]$ via $\sum_c \tilde{a}(c)\tilde{b}(c)$. Since $\tilde{a}(c) = 0$ for c such that t doesn't belong to the reduced test results space of size $B(\varepsilon)$, we can choose to perform this sum on either this set, or the reduced code space of size $C(\varepsilon)$: whichever is the smallest. This is where the denomination "meet-in-the-middle" comes from.

D Interactive demonstration

The C++ code can be used in the browser through an interactive WebAssembly demo:
https://bloom-origami.github.io/

The following features are implemented:

- Bloom assay generation
- Greedy adaptive strategy simulation
- Design optimization using genetic algorithms
- Posterior decoding using MITM

E Prevalence Estimation

Our designs assume that the prevalence ρ is known, at least approximately. However, we can also use our Bloom filter design to estimate the prevalence in the overall infected population. When we randomly and independently sample an individual from the population, they have probability ρ of

being infected. The prevalence estimation problem is to determine ρ using as few tests as possible. Here, we assume perfect tests to simplify the analysis.

Of course, one could individually test a large number of people from the population and report the fraction of positive test results. The challenge is that if we screen individuals, we end up with a random variable for which the mean to variance ratio is unfavorable. Consider a random variable $X \in \{0, 1\}$ with $\mathbb{E}[X] = \rho$ and variance $\mathrm{var}[X] = \rho - \rho^2 = \rho(1 - \rho)$. The error of the empirical average of m individual tests is

$$\frac{1}{m}\sum_{i=1}^{m} X_i - \mathbb{E}[X] = O\left(\frac{\mathrm{std}[X_u]}{\sqrt{m}}\right).$$

The relative error is $\sqrt{\frac{1-\rho}{\rho}}$. Clearly this is minimized for $\rho = 1$. Unfortunately, this value is entirely useless since it corresponds to the situation where every test returns positive. In practice, we encounter the unfortunate situation of $\rho << 1$ where the relative error diverges. Under a naive random sampling approach to prevalence estimation, a very large number of tests are required. To amend this situation, it is beneficial to increase the probability of a positive test by testing multiple candidates at once. Our pooled tests are no longer positive with probability ρ but with probability $q = 1 - (1 - \rho)^k$, where k is the number of samples combined in a single pool. Knowing q, we can solve for ρ via

$$\rho = 1 - (1 - q)^{\frac{1}{k}}$$

We will use the central limit theorem and the delta method to show that we need fewer Bloom filter pooled tests than random individual tests to estimate the prevalence. The central limit theorem states that

$$\overline{X}_m - \mu \xrightarrow{d} \mathcal{N}\left(0, \frac{\sigma(\mu)^2}{m}\right)$$

where $\overline{X}_m$ is the average of m trials, $\mu = \mathbb{E}[X]$, and $\sigma^2 = \mathrm{var}[X]$. The delta method states that if we have a function $g(x)$ and its derivative $g'(x)$, then

$$g(\overline{X}_m) - g(\mu) \xrightarrow{d} \mathcal{N}\left(0, \frac{[g'(\mu)\sigma(\mu)]^2}{m}\right)$$

E.1 Prevalence Estimation with Random Sampling

Suppose we randomly sample individuals from the population and perform m individual tests. Here, X is the test status of the patient and it is positive with prevalence ρ. We estimate ρ as $\hat{\rho} = \frac{1}{m}\sum_{i=1}^{m} X_i$ Observe that $\mathbb{E}[X] = \mu = \rho$ and $\mathrm{var}[X] = \rho(1 - \rho)$. Use the central limit theorem to observe that

$$\hat{\rho} - \rho \xrightarrow{d} \mathcal{N}\left(0, \frac{\rho(1-\rho)}{m}\right)$$

E.2 Prevalence Estimation with Bloom Filters

Suppose we combine k samples into each bin. Now X is the test status of the bin and it is positive with probability $q = 1 - (1 - \rho)^k$. Hence $\mu = q$ and $\sigma^2 = q(1 - q)$. Use the delta method with

$$g(x) = 1 - (1 - x)^{\frac{1}{k}}$$
$$g'(x) = \frac{1}{k}(1 - x)^{\frac{1}{k}-1}$$

Observe that $g(\mu) = \rho$ and that $\hat{\rho} = g(\overline{X}_n)$. From the delta theorem we have

$$\hat{\rho} - \rho \xrightarrow{d} \mathcal{N}\left(0, \frac{[g'(\mu)\sigma(\mu)]^2}{m}\right)$$

We proceed by analyzing the $g'(\mu)\sigma(\mu)$ term. This term is

$$\sqrt{\mu(1-\mu)}\frac{1}{k}(1-\mu)^{\frac{1}{k}-1}$$

Recall that $\mu = q = 1-(1-\rho)^k$. Substitute this value to get

$$\hat{\rho} - \rho \xrightarrow{d} \mathcal{N}\left(0, \frac{\alpha^2}{m}\right)$$

where

$$\alpha = \frac{1}{k}\frac{\sqrt{1-(1-\rho)^k}}{(1-\rho)^{\frac{k}{2}-1}}$$

E.3 Comparison

We are interested in whether the variance of the Bloom filter estimator is larger than the variance of the random sampling estimator. That is, we want to prove the following inequality.

$$\frac{1}{k}\frac{\sqrt{1-(1-\rho)^k}}{(1-\rho)^{\frac{k}{2}-1}} \leq \sqrt{\rho(1-p)}$$

Rearrange

$$\sqrt{\frac{1}{(1-\rho)^k} - 1}\left(\frac{1-\rho}{k}\right) \leq \sqrt{\rho(1-\rho)}$$

Recall the inequality $1 - x \geq e^{-x\,/\,(1-x)}$ when $0 \geq x < 1$. Applied to our situation, this means that

$$\frac{1}{(1-\rho)^k} < e^{\rho k\,/\,(1-\rho)}$$

Therefore our inequality becomes

$$\sqrt{e^{\frac{\rho k}{1-\rho}} - 1}\left(\frac{1-\rho}{k}\right) \leq \sqrt{\rho(1-\rho)}$$

Put $k = (1-\rho)/\rho$. Then the inequality is true when $\rho \leq 1/e$. Bloom filters are a better way to measure prevalence provided that ρ is smaller than 37% or (using symmetry arguments) greater than 63%.

F Empirical Validation of Adaptive Sub-Modularity

Below the C++ code used to validate the assumption of adaptive sub-modularity relative to Theorem 2, for small values of n.

```
1
2  #include <vector>
3  #include <algorithm>
4  #include <utility>
5  #include <math.h>
6  #include <assert.h>
7  #include <iostream>
8  #include <functional>
9  #include <map>
10 #include <queue>
11
```

```cpp
using namespace std;

using vd =  vector<double>;

// expected entropy of simultaneous tests
double expected_entropy(const double obs01, const double obs11,
                        const vd &prior,
                        const vector<int> &tests) {
    // optimized version with constant memory
    int t = tests.size()   ;
    int N = prior.size();
    double ans = 0;
    for(int m=0; m<1<<t; m++) {
        double prob_m = 0;
        double entropy_m = 0;
        for(int s=0; s<N; s++) {
            double joint_s_m = prior[s];
            // probability of observing joint_s_m
            for(int i=0; i<t; i++) {
                auto p = (s & tests[i]) ? obs11 : obs01;
                joint_s_m *= (m & (1<<i)) ? p : 1-p;
            }
            prob_m += joint_s_m;
            if(joint_s_m)
                entropy_m -= joint_s_m * log2(joint_s_m);
        }
        if(prob_m)
            entropy_m += prob_m * log2(prob_m);
        ans +=  entropy_m;
    }
    return ans;
}



static double drand() {
    return (double)rand() / RAND_MAX;
}

int main() {
    int TESTS = 100000;
    while(TESTS--) {

        int n = 5;
        int N = 1 << n;
        double obs01 = drand() / 2;
        double obs11 = 1 - drand() / 2;

//        vd prob_ill(n);
//        for(auto &v : prob_ill)
//            v = drand();
//        auto prior = factor(prob_ill);
        vd prior(N);
        double s = 0;
        for(auto &v : prior)
            s += v = drand();
        for(auto &v : prior)
            v /= s;

```

```
71
72          int test1 = rand() % (N-1) + 1;
73          int test2 = rand() % (N-1) + 1;
74
75          auto aux = [&](const vector<int> &tests) {
76              return expected_entropy(obs01, obs11, prior, tests);
77          };
78
79          auto delta = aux({test1, test2}) - aux({test1})
80              - aux({test2}) + aux({});
81
82          if(delta < -1e-6) {
83              cout << test1 << ' ' << test2 << endl;
84              cout << obs01 << ' ' << obs11 << endl;
85  //              for(auto x : prob_ill)
86  //                  cout << x << ' ';
87              cout << endl;
88
89              cout << delta << endl;
90              cout << aux({}) << ' ' << aux({test1}) << ' '
91                  << aux({test2}) << ' ' << aux({test1, test2}) << endl;
92          }
93      }
94  }
95  static double drand() {
96      return (double)rand() / RAND_MAX;
97  }
98  int main() {
99      int TESTS = 100000;
100     while(TESTS--) {
101         int n = 5;
102         int N = 1 << n;
103         double obs01 = drand() / 2;
104         double obs11 = 1 - drand() / 2;
105 //        vd prob_ill(n);
106 //        for(auto &v : prob_ill)
107 //            v = drand();
108 //        auto prior = factor(prob_ill);
109         vd prior(N);
110         double s = 0;
111         for(auto &v : prior)
112             s += v = drand();
113         for(auto &v : prior)
114             v /= s;
115
116         int test1 = rand() % (N-1) + 1;
117         int test2 = rand() % (N-1) + 1;
118         auto aux = [&](const vector<int> &tests) {
119             return expected_entropy(obs01, obs11, prior, tests);
120         };
121         auto delta = aux({test1, test2})
122             - aux({test1}) - aux({test2}) + aux({});
123         if(delta < -1e-6) {
124             cout << test1 << ' ' << test2 << endl;
125             cout << obs01 << ' ' << obs11 << endl;
126 //              for(auto x : prob_ill)
127 //                  cout << x << ' ';
128             cout << endl;
129             cout << delta << endl;
```

```
130                 cout << aux({}) << ' ' << aux({test1}) << ' '
131                     << aux({test2}) << ' ' << aux({test1, test2}) << endl;
132             }
133         }
134 }
```