Bloom Origami Assays

Abraham et al.

Outline

- Introduction / Motivation / Setting
- Adaptive vs non-adaptive testing
- Exploration using ES
- The case for greedy
- Bloom filters for test design
- Meet-in-the-middle decoding
- Experimental results
- Unfairness of group testing

Introduction / Motivation / Setting

- **Group testing is great**, how can we study it with a more practical approach?
- Constraints
  - Number of tests
  - Number of samples collected per person
  - Number of samples mixed together
- Real world setting
  - All options don’t have the same probability
  - We don’t know how many people are ill
  - Tests are not perfect
  - Testing takes time

Bayesian 101

- Represent the state of each person with a bit
- Input a prior over the n-bit strings
- Each test updates the prior

Trivia: Bayesian update is

- commutative (order of tests does not matter)
- *compositional* (testing in parallel or sequence does not matter)

Non-adaptive testing

n=4 people, m=2 tests

$$\begin{matrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{matrix}$$

Search space: $\binom{2^n+m}{m}$

(less with constraints on rows and columns)

Adaptive testing

1. Test people
2. Update prior
3. Go back to 1.

Search space: $(2^n - 1)2^m$

Exploration using ES

- Advantages
  - No tricky math
  - Exact computation
  - Choice of objective
  - Choice of constraints
- Disadvantages
  - Expensive
  - *Almost* no guarantee on the solution optimality

ES: how we do it

- Search space: binary strings of size n x m
- **(1 + λ) strategy**: mutate the *best* individual into λ offsprings and repeat
- Constant initialization: simple and best to satisfy constraints
- **Luby's restart** strategy
  - Optimal restart strategy up to a logarithmic constant (Thm 1)
  - Rule-of-thumb: O(n * m) as basis
- Fitness criterion
  - Conditional entropy (= expectation of entropy)
  - Expected confidence (= expectation of mode)
  - Expected number of diagnosed with confidence > threshold
  - ...

ES in practice

- Gives a good intuition
- **Super useful** to give counter-examples
- Scalable up to ~10 people and tests

You can test it!

https://louisabraham.github.io/crackovid/crackovid.html

The case for greedy

- ES is not scalable to optimize adaptive strategies or for large n
- Can we be greedy? **YES**

- **Magic of submodularity**
  1. Find a hard optimization problem
  2. Show that your objective is monotone submodular
  3. Profit
- We apply it to conditional entropy

Conditional entropy

- **Adaptive monotonicity is trivial**: making one more test decreases the conditional entropy (*information never hurts*)
- **Adaptive submodularity is NOT trivial**
- Conditional entropy is not submodular in general
  - Take b1, b2 random bits and b3 = b1 ^ b2
  - H(b3) = H(b3 | b1) = 1
  - H(b3 | b1, b2) = 0
  - So b2 “helps” more when combined with b1 than alone
- But our model only allows for OR operations and independent corruption

Is conditional entropy of tests submodular?

Is conditional entropy of tests submodular?

Probably!

```
51  int main() {
52      int TESTS = 100000;
53      while(TESTS--) {
```

Implications of submodularity

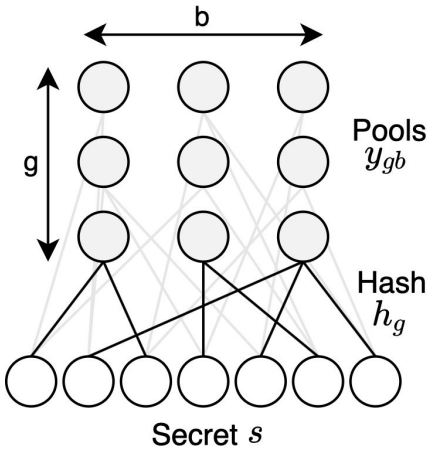
- In short, **approximation ratio 1 - 1/e ≃ 0.63**
- We also get robust bounds when:
  - changing the number of tests
  - setting wrong priors

Bloom filters for test design

- We want to gain a lot of information with a small number of tests
- Bloom filters 101: store sets in a compressed way
  - Take a bit array A of size m and k hash functions h1..hk that go to [|1;m|]
  - For each x, set A[h1(x)] = A[h2(x)] = ... = A[hk(x)] = 1
  - To test whether an element x is in the set, compute A[h1(x)] & A[h2(x)] & ... & A[hk(x)]
  - Nice analysis using Azuma's inequality
- Let's try to ***store*** **the set of ill patients in the results of m tests**
  - k is the number of samples of each patient
  - hi(x) tells where i-th sample goes
  - A[j] is the (ideal) outcome of the j-th test

Improving on Bloom filters

- Bloom filters are meant for online applications that use stream inputs
- Hash functions are just meant for load balancing (don't put many items in the same bin)
- Instead, we can use ***perfect* load balancing**
- Take n = k * b, m = g * b
- For i = 1...g
  - Shuffle [|1, n|]
  - Assign 1...k to batch 1
  - Assign k+1...2k to batch 2
  - etc (b times)
- Total: g rows, b columns



Theory of Bloom Origami Assays

- Recall n = k * b, m = g * b
- How can we choose b? Depends on the prevalence p

  Thm 4: b = n p / log(2)
  (assume perfect tests and maximize probability of perfect decoding)

- If p is not uniform, how can we balance the bins to maximize information gain for a single row?

  Thm 3: Make all bins have a probability equal to a constant depending on tnr and tpr

Posterior decoding

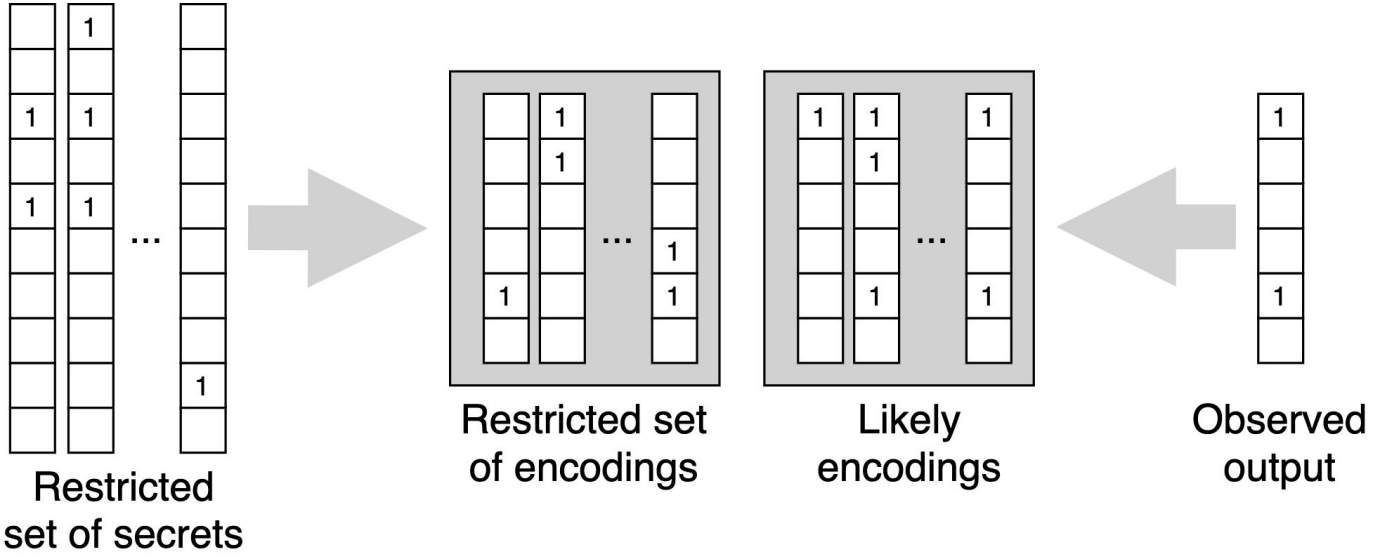
- **Testing has 2 components: test design and posterior decoding**
- We have scalable designs, we need scalable decoding

Remark: we don’t need the distribution over the bitstrings of length n, only n marginals.

- We have a PGM
- Can we apply message passing? We don’t have a polytree, loopy belief propagation comes with no guarantee but gives acceptable performance
- Can we find better?

Performant decoding: MITM

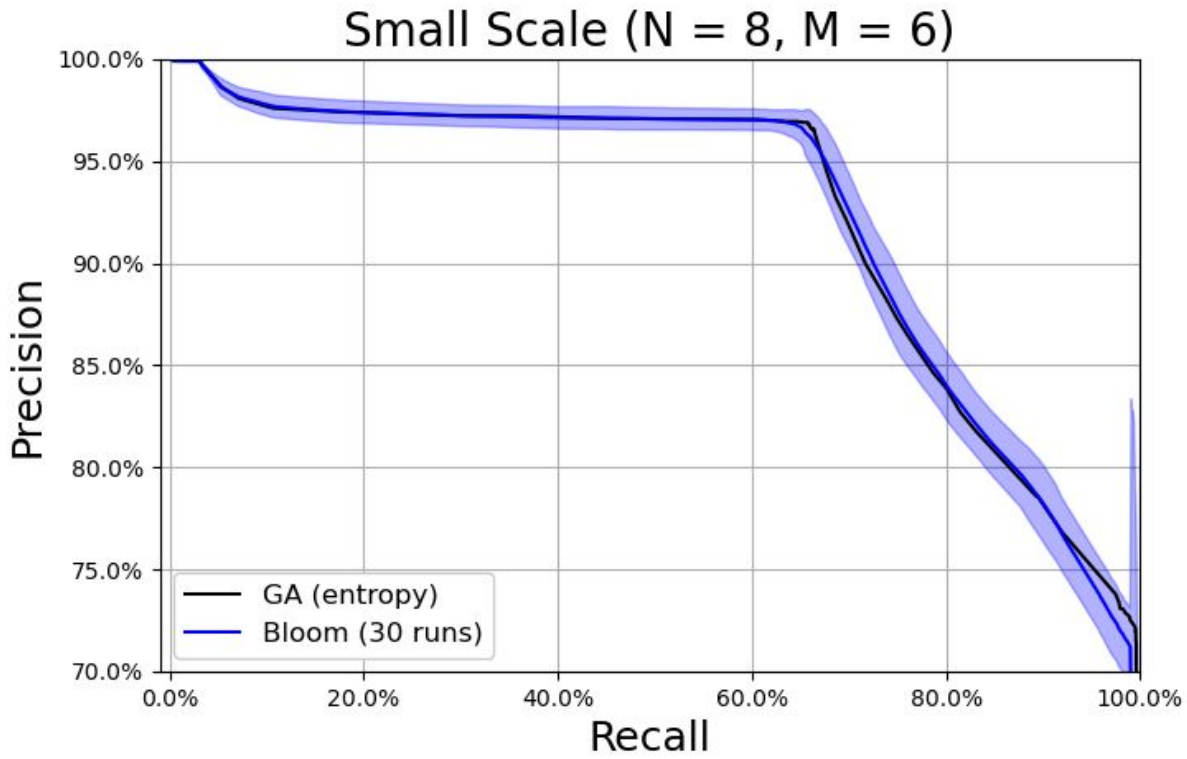
- In practice, p << ½ and most *secrets* never happen
- Likewise, TNR and TPR are close to 1 and most bits are correct
- (if p = ½ and TNR = TPR = 1, equivalent to #SAT)
- Let's bruteforce over a **pruned** search space!



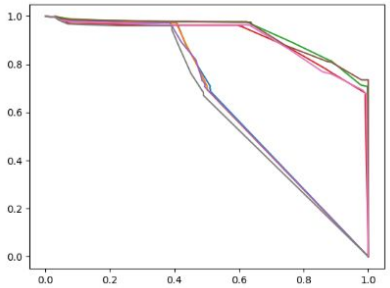
MITM details

- Pre-computation:
  - Cut a portion < Ɛ of the secret space (consider secrets with at most k infected people)
  - Store the encodings of those secrets with the summed marginals
- Inference:
  - Ignore decoding errors with probability < Ɛ
  - Sum the joint marginals for all possible pre-computed encodings
  - Normalize
- **Thm 5**: For any test result t, the above algorithm estimates the posterior P(si | t) with error at most 4Ɛ/P(t) and produces an upper bound on this error

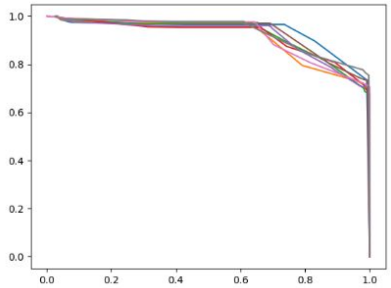
Comparison of test designs (small scale)



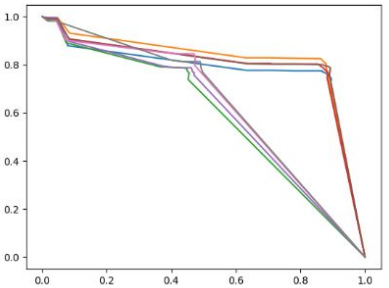
Comparison of test designs (small scale)



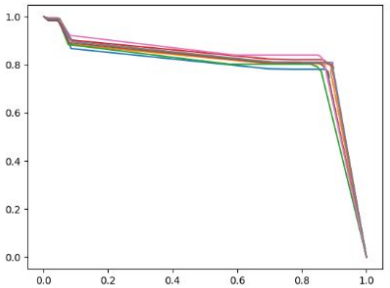
(a) Bloom: $(b, g) = (2, 3)$



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

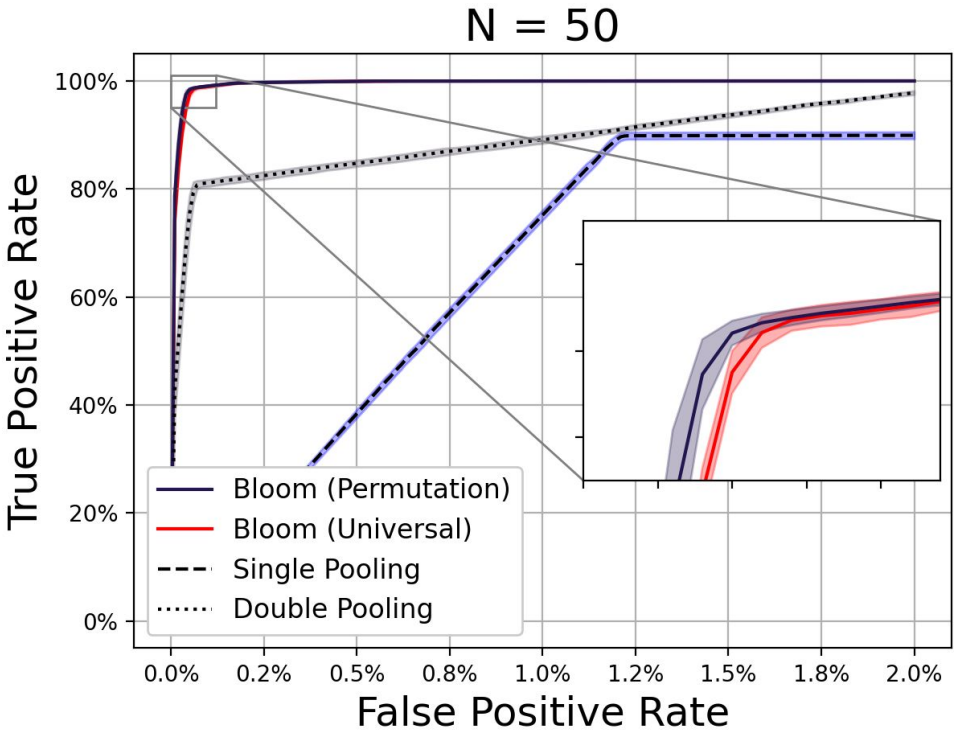


(c) Bloom: $(b, g) = (3, 2)$

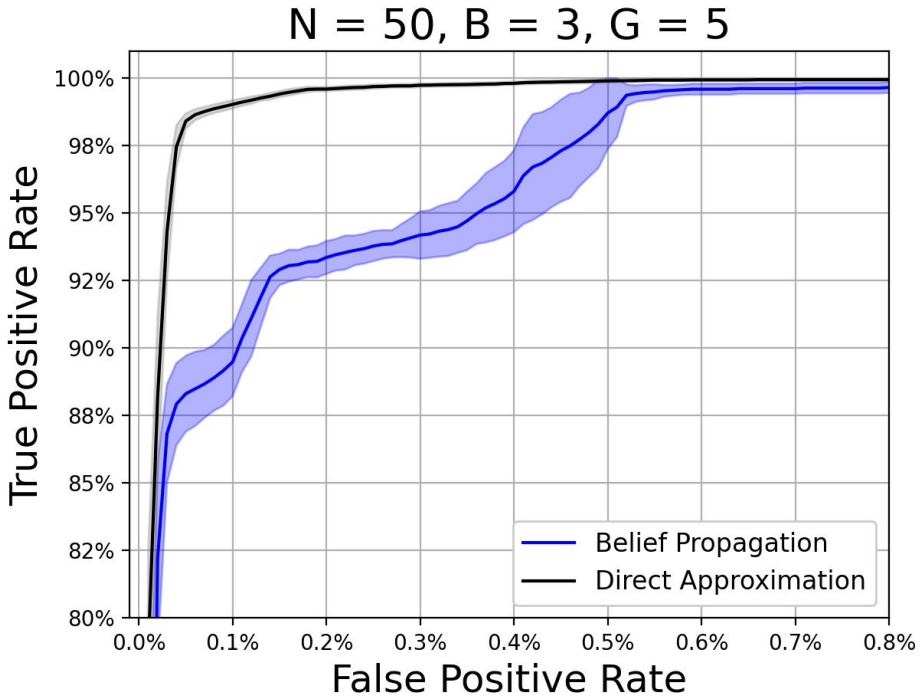


(d) Entropy: $(m, k) = (6, 2)$

Comparison of test designs (large scale)



Comparison of decoding algorithms



Unfairness of group testing

- **Some designs for group testing are *unfair*** on a small scale! (our ES strategy seems fair)
- Origami Bloom Assays are still *randomly fair* when using uniform prevalence
- When priors are not uniform, how does group testing affect the TPR/TNR of the posterior marginals?
- What is the responsibility of the doctor when deciding priors, affecting patients or deciding on a testing scheme?