Combinatorial Group Testing: Motivations and the Matrix Representation

For any positive integer m and any finite set X, let $[m]$ denote the set $\{1, \dots, m\}$, $\binom{X}{m}$ denote the collection of all m-subsets of X, and 2^X denote the power-set of X.

1 The basic group testing problem

The basic group testing problem is to identify the set of "*positives*" from a large population of "*items*" using as few "*tests*" as possible. A test is a subset of items, which returns positive if there is a positive in the subset. The semantics of "positives," "items," and "tests" depend on the applicational context.

Example 1.1 (Blood testing)**.** Group testing as a research area can be traced back to 1943 when Dorfman studied the problem of testing for syphilis in WWII draftees blood samples [12]. In this case, items are blood samples, which are positive if they are infected. A test is a *pool* (group) of blood samples. Testing a group of samples at a time will save resources if the test outcome is negative. On the other hand, if the test outcome is positive then all we know is that at least one sample in the pool is positive but we do not know which one. Since 1943, group testing has found numerous applications in many areas of Mathematics, Computer Science, and Computational Biology. The reader is referred to the standard monograph on the subject by Du and Hwang [13], and several surveys [1, 14, 16, 22]. These expository works are nice but outdated. Results covered in this seminar are mostly new. (We will make use of old techniques and results, however!)

We next describe an application in computational biology.

Example 1.2 (DNA library screening)**.** The basic problem of *DNA library screening* is to determine which clone (a DNA segment) from the library contains which probe from a given collection of probes in an efficient fashion. A clone is said to be positive for a probe if it contains the probe, and negative otherwise. In practice clones are pooled together in some manner to be tested against each probe, since checking each clone-probe pair is expensive and usually only a few clones contain any given probe. An example is when Sequenced-Tagged Site markers (also called STS probes) are used [23]. If the test result for a pool (of clones) is negative, indicating that no clone in the pool contains the probe, then no further tests are needed for the clones in the pool.

It is customary to consider two types of group testing problems: *Combinatorial Group Testing* (CGT) and *Probabilistic Group Testing* (PGT). CGT is the worst case model, where it is assumed that the number of positives among N items is at most d for some fixed positive integer d. In PGT, we impose some probability distribution on the positives. For example, earlier works assumed that each item is positive with some probability p [26, 27]. Thus, CGT is analogous to Hammings adversarial noise model in information theory [17]; and PGT corresponds to Shannon's probabilistic noise model [25]. We are mostly concerned with the combinatorial group testing model in these notes. Hence, unless specically stated, "group testing" means combinatorial group testing from now on.

If the test pools are done in s stages ($s \in \mathbb{N}$), where the pools of a later stage are designed depending on the test results of the earlier stages, then the group testing algorithm is said to be an s-stage algorithm. Group testing strategies can also be either *adaptive* or *non-adaptive*. A group testing algorithm is non-adaptive if all tests must be specified without knowing the outcomes of other tests. Clearly, being nonadaptive is equivalent to being 1-stage. A group testing algorithm is *error tolerant* if it can detect or correct errors in test outcomes. We will have more to say about error-correcting group testing in later lectures.

Exercise 1 (A simple adaptive strategy)**.** Consider the case when there's at most $d = 1$ positive. Design an *adaptive* CGT algorithm with at most $\lceil \log_2 N \rceil$ tests. Show that this number of tests is also necessary.

Exercise 2 (Adaptive strategy for $d = 2$)**.** Repeat the above problem with $d = 2$; use as few tests as you can. You don't need to prove the lower bound.

Example 1.3 (Non-adaptive CGT and traitor tracing)**.** In many applications such as Pay-TV, satellite radio, and the distribution of copyright-protected materials, a content provider needs to broadcast digital information to a specic set of users (e.g., subscribers) who were given key(s) for decrypting the content. One natural requirement for such broadcast system is the ability to *trace traitors*, in the following sense. Some users might collude, build a pirate decoder, and distribute it widely (for a fee or not). Or, a pirate might achieve similar effects via hacking accounts of legitimate users. Either way, such users are called traitors. It is desirable for the system to be able to identify at least one traitor by examining the pirate decoder. This is the *traitor tracing problem* [6].

A simple type of key distribution scheme can be abstracted as follows. Let N be the number of users in the system. Let T be a set of cryptographic keys to be distributed to the users. User j receives a key set $F_j \subseteq T$. User j is able to decrypt the content if the content was encrypted using one of the keys in F_j. At one extreme, we can set $|T| = 1$ and all users get the same key. In this case, the system uses little resources but it cannot trace a single traitor. At another extreme, we can set $|T| = N$ and assign each user a separate key. This is rather inefficient because we will have to encrypt the same content N times for broadcasting. (Technically, we say that the "ciphertext" size is $O(N)$.) However, this scheme can trace any number of traitors.

When examining the decoder, we can try each key in T one by one. Suppose the decoder decodes iff it possesses the key, then each decoder test can be viewed as a group test, where the test turns positive iff one of the traitors possesses the testing key. (In practice, pirate decoders are smarter, but for our purposes we assume a simple decoder.) Traitor tracing is thus a non-adaptive group testing problem. One objective is to minimize the total number of keys $|T|$ needed.

For simplicity lets assume that there is at most $d = 1$ traitor. In this case, it is not hard to see that as long as all the F_j are non-empty and distinct, then we can uniquely identify the traitor by pin-pointing the subset of keys the decoder possesses. Thus $|T| = \lceil \log_2(N + 1) \rceil$ is sufficient. It is not hard to show that this number of tests is also necessary.

2 Non-adaptive combinatorial group testing and the matrix representation

Consider a non-adaptive group testing strategy with t tests on a population of N items. The strategy can be represented by a $t \times N$ binary matrix $\mathbf{M} = (m_{ij})$, where $m_{ij} = 1$ iff item j belongs to test i. Let $\mathbf{M}_i$ and $\mathbf{M}^j$ denote row i and column j of $\mathbf{M}$, respectively. Abusing notation, we will also use $\mathbf{M}_i$ (respectively, $\mathbf{M}^j$) to denote the set of columns (respectively, rows) corresponding to the 1-entries of row i (respectively, column j). More precisely,

$$\begin{aligned}\mathbf{M}_i &= \{j \mid m_{ij} = 1\} \\ \mathbf{M}^j &= \{i \mid m_{ij} = 1\}\end{aligned}$$

In other words, $\mathbf{M}_i$ is the ith pool, and $\mathbf{M}^j$ is the set of pools that item j belongs to. Henceforth, we will also often view a t-dimensional binary vector as a subset of $[t]$ (which is the set of the positions of the 1-entries). Thus, set operations such as unions and intersections are applied to binary vectors in the natural way.

A key question is: "*what properties must the matrix* $\mathbf{M}$ *satisfy in order for the corresponding non-adaptive group testing strategy to be able to uniquely identify the arbitrary set of at most* d *positives?*"

To answer this question, consider an arbitrary subset $D \subseteq [N]$ of positive items, where $|D| \leq d$. Let $\mathbf{y} = (y_i)_{i=1}^t \in \{0,1\}^t$ denote the test outcome vector, i.e. $y_i = 1$ iff the ith test is positive. Then, it is not hard to see that the test outcome vector is precisely the (boolean) union of the positive columns:

$$\mathbf{y} = \bigcup_{j \in D} \mathbf{M}^j.$$

Consequently, to be able to uniquely identify an arbitrary subset D of at most d positives, the test outcome vectors $\mathbf{y}$ have to be distinct. Conversely, if the test outcome vectors are distinct then we can pre-compute a "lookup table" which associates each test outcome vector with its (positive) item set. This observation motivates the following definition.

Definition 2.1 (Separable matrix)**.** A $t \times N$ binary matrix $\mathbf{M}$ is *d-separable* if the unions of up to d columns of $\mathbf{M}$ are all distinct.

Note that in the above definition we also need to take into account the union of *no* column of $\mathbf{M}$. This "union" is empty, and thus no column of $\mathbf{M}$ can be all-zero. This requirement makes sense because if there was no positive item, the test outcome is the all-zero vector. If there was an item j_0 which belongs to no test, then we cannot distinguish between the case when there is no positive and the case when the item j_0 is positive. The following proposition is straightforward from the denition above.

Proposition 2.2. *A non-adaptive combinatorial group testing strategy works for up to* d *positives iff the matrix* $\mathbf{M}$ *representing the strategy is* d*-separable.*

Exercise 3. Let $\mathbf{M}$ be a d-separable matrix with t rows and N columns. Prove that

$$\sum_{k=0}^{d} \binom{N}{k} \leq 2^t.$$

Then, infer that $t = \Omega(d \log(N/d))$.

Identifying the positives given the test outcome vector is also called *decoding*. If we use a separable matrix in a brute-force manner, then either the decoding time is $\Omega(N^d)$ or the space complexity is $\Omega(N^d)$ (with a pre-built lookup table). Is there any way to improve either or both of the decoding time and space complexity?

A very natural decoding algorithm is Algorithm 1, which we will refer to as the *naive decoding algorithm*. Does naive decoding always work for an arbitrary separable matrix? Unfortunately, the answer is no.

Algorithm 1 Naive decoding

```
1: for $j = 1$ to $N$ do
2:     if item $j$ belongs to at least one negative test then
3:         mark $j$ as a negative item
4:     end if
5: end for
6: return $R$, the set of remaining items
```

Exercise 4. Find a d-separable matrix $\mathbf{M}$ such that there is some set S of at most d positives for which the naive decoding algorithm fails to return S.

Exercise 5. Show that if there can be up to $d = N$ negative items, then we must need at least $t \geq N$ tests. Hence, when $d = N$ we can do no better than the "test one item at a time" strategy.

Definition 2.3 (Disjunct matrix)**.** A matrix $\mathbf{M}$ is called *d-disjunct* if it represents a NAGT strategy for which the naive decoding algorithm always returns the correct answer as long as there $\leq d$ positives.

The following equivalent characterization of disjunct matrices is easier technically to understand and handle. The proof is easy and thus omitted.

Proposition 2.4 (Another characterization of disjunct matrices)**.** *Let $d \in [N]$ be integers. A $t \times N$ binary matrix $\mathbf{M}$ is said to be d-disjunct if the union of arbitrary $\leq d$ columns does not contain another column.*

So disjunct matrices allow for linear time decoding (in the size of the matrix, $O(tN)$), which is definitely a vast improvement over the brute-force algorithm for separable matrices. What did we sacrifice for this speedup? Exercise 4 has shown that a d-separable matrix is not necessarily a d-disjunct matrix. The following exercises complete the picture.

Exercise 6. Show that any d-disjunct matrix is d-separable, and any d-separable matrix is $(d-1)$-disjunct.

$$\begin{tikzcd}
d\text{-separable} \arrow[r, Rightarrow, "/" marking] \arrow[d, Rightarrow] & d\text{-disjunct} \arrow[d, Rightarrow] \\
(d-1)\text{-disjunct} & d\text{-separable}
\end{tikzcd}$$

Exercise 7. Let $\mathbf{M}$ be a d-separable matrix. Let $\overline{\mathbf{M}}$ be the complement of $\mathbf{M}$, where each entry is turned from 0 to 1 and vice versa. Show that the stacking of $\mathbf{M}$ and $\overline{\mathbf{M}}$ is d-disjunct. In particular, the optimal number of rows of a d-disjunct matrix is at most twice the optimal number of rows of a d-separable matrix.

Open Problem 2.5. Is there a generic way to "turn" a d-separable matrix into a d-disjunct matrix where the number of tests blowup is less than 2?

3 The heavy hitter problem and four main requirements

There are many recent applications of non-adaptive combinatorial group testing, ranging from drug and DNA library screening [18,19,22,30], multiple access control protocols [3,29], live baiting of DoS attackers [20], data forensics [15], data streams [10], codeword testing [24], pattern matching [7], among others. See also the standard monograph on group testing for more details on some of these applications [13]. In this section, we discuss one application which motivates the studies of efficiently decodable group testing procedures.

Modern Internet core routers have very high capacities, in the order of terabits per second or more [28]. There are literally hundreds of thousands of packets passing through such a router each second. Internet service providers (ISPs) often need to collect various types of statistics about traffic passing through such routers. For example, they want to estimate the number of distinct source IPs [2], the top-k frequent sources [8], or some notion of flow entropies [4]. Because there is a huge amount of data passing through at an extremely high speed, these statistics collecting algorithms are often required to run in sub-linear time (preferably polylog time) using sub-linear space (again, preferably poly-log space). This type of problems belong to the broad umbrella of *data streaming* [21].

Heavy hitter is one of the most fundamental problems in data streaming. We will not define the problem rigorously as there are several variations with subtle differences. The reader is referred to [5, 8, 9, 11] for the descriptions and different formulations of the problem. We will, however, describe informally an approach for solving the (informal version of the) heavy hitter problem using combinatorial group testing.

For concreteness, let us restrict our attention to the networking application, where a router needs to keep track of top d frequent source IPs using low space and time complexity. Think of d as much smaller than N. There are roughly $N = 2^{32}$ potential source IP addresses. Hence, keeping a counter for each source is infeasible. Let $\mathbf{M}$ be a d-disjunct matrix with t rows and N columns. It is known, as we shall see later, that $t = O(d^2 \log N)$ is sufficient for such a matrix to exist. For small values of d, we have $t \ll N$. The idea (first proposed in [10]) is to use only $t + 1$ counters. One counter c counts the total number of packets seen so far. Then there is a counter c_i for each row i of the matrix $\mathbf{M}$. This counter keeps track of all the source IP j for which $m_{ij} = 1$. Now, suppose we define a "heavy hitter" as a source IP which occurs $> 1/(d + 1)$ fraction of time. Then, we call a "test" i positive if the counter c_i has value more than $c/(d + 1)$. If c_i keeps track of a heavy hitter, then certainly the ith "test" is positive. Conversely, if the ith test is positive then it is not necessarily true that c_i keeps track of any heavy hitter, because the non-heavy-hitters might "contaminate" the counter enough to make its value larger than $c/(d + 1)$. Thus, if we know that the total frequency of all the nonheavy-hitters is at most $c/(d + 1)$ then our group testing scheme works. This assumption is called the "small tail" assumption. For $d \ll N$, $t = O(d^2 \log N) \ll N$ which means the total space required is a lot less than the naive one-counter-per-IP scheme.

There are several requirements for group testing algorithms motivated by this data streaming application:

- *Small number of tests.* The number of tests t is proportional to the memory space needed by the streaming algorithm. Minimizing the number of tests is thus an important objective, which was also a main objective in all of the group testing applications discussed above.
- *Strongly explicit construction.* We do not want to store the entire matrix $\mathbf{M}$ because its size is too large. We want a strongly explicit construction of $\mathbf{M}$, which means a column of $\mathbf{M}$ can be computed on the fly quickly. Concretely, $\mathbf{M}$ is *explicitly constructed* if, given two indices $(i, j) \in [t] \times [N]$, we can compute m_{ij} in time poly$(t, logN)$.
- *Sub-linear decoding time.* When the matrix $\mathbf{M}$ is d-disjunct, the naive decoder runs in time $O(tN)$ which is way too high. It is thus desirable to have a disjunct or separable matrix which can be decoded in time sublinear in N, preferably poly-log in N.
- *Error-tolerance.* Another requirement comes from the small tail assumption, which certainly does not always hold. Every time a counter c_i is "contaminated" by non-heavy-hitters, the test outcome is a false positive. We want group testing strategies which can tolerate errors.

These four requirements shall be our guiding force for most of this course.

References

[1] D. J. BALDING, W. J. BRUNO, E. KNILL, AND D. C. TORNEY, *A comparative survey of non-adaptive pooling designs*, in Genetic mapping and DNA sequencing (Minneapolis, MN, 1994), Springer, New York, 1996, pp. 133–154.

[2] Z. BAR-YOSSEF, T. S. JAYRAM, R. KUMAR, D. SIVAKUMAR, AND L. TREVISAN, *Counting distinct elements in a data stream*, in RANDOM, 2002, pp. 1–10.

[3] T. BERGER, N. MEHRAVARI, D. TOWSLEY, AND J. WOLF, *Random multiple-access communications and group testing*, IEEE Trans. Commun., 32 (1984), pp. 769–779.

[4] A. CHAKRABARTI, G. CORMODE, AND A. MCGREGOR, *A near-optimal algorithm for estimating the entropy of a stream*, ACM Transactions on Algorithms, 6 (2010).

[5] M. CHARIKAR, K. CHEN, AND M. FARACH-COLTON, *Finding frequent items in data streams*, in ICALP, 2002, pp. 693–703.

[6] B. CHOR, A. FIAT, AND M. NAOR, *Tracing traitors*, 1994, pp. 257–270.

[7] R. CLIFFORD, K. EFREMENKO, E. PORAT, AND A. ROTHSCHILD, *Pattern matching with don't cares and few errors*, J. Comput. Syst. Sci., 76 (2010), pp. 115–124.

[8] G. CORMODE AND M. HADJIELEFTHERIOU, *Finding the frequent items in streams of data*, Commun. ACM, 52 (2009), pp. 97–105.

[9] G. CORMODE AND S. MUTHUKRISHNAN, *An improved data stream summary: the count-min sketch and its applications*, J. Algorithms, 55 (2005), pp. 58–75.

[10] ———, *What's hot and what's not: tracking most frequent items dynamically*, ACM Trans. Database Syst., 30 (2005), pp. 249–278.

[11] E. D. DEMAINE, A. LÓPEZ-ORTIZ, AND J. I. MUNRO, *Frequency estimation of internet packet streams with limited space*, in ESA, 2002, pp. 348–360.

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

[13] D.-Z. DU AND F. K. HWANG, *Combinatorial group testing and its applications*, vol. 12 of Series on Applied Mathematics, World Scientific Publishing Co. Inc., River Edge, NJ, second ed., 2000.

[14] A. G. D′YACHKOV AND V. V. RYKOV, *A survey of superimposed code theory*, Problems Control Inform. Theory/Problemy Upravlen. Teor. Inform., 12 (1983), pp. 229–242.

[15] M. T. GOODRICH, M. J. ATALLAH, AND R. TAMASSIA, *Indexing information for data forensics*, in Third International Conference on Applied Cryptography and Network Security (ANCS), 2005, pp. 206–221.

[16] S. GYŐRI, *Coding for a multiple access OR channel: a survey*, Discrete Appl. Math., 156 (2008), pp. 1407–1430.

[17] R. W. HAMMING, *Error detecting and error correcting codes*, Bell System Tech. J., 29 (1950), pp. 147–160.

[18] KAINKARYAM, *Pooling in high-throughput drug screening*, Current Opinion in Drug Discovery & Development, 12 (2009), pp. 339–350.

[19] R. KAINKARYAM AND P. WOOLF, *poolhits: A shifted transversal design based pooling strategy for high-throughput drug screening*, BMC Bioinformatics, 9 (2008).

[20] S. M. KHATTAB, S. GOBRIEL, R. G. MELHEM, AND D. MOSSÉ, *Live baiting for service-level dos attackers*, in INFOCOM, 2008, pp. 171–175.

[21] S. MUTHUKRISHNAN, *Data streams: algorithms and applications*, Foundations and Trends in Theoretical Computer Science, 1 (2005).

[22] H. Q. NGO AND D.-Z. DU, *A survey on combinatorial group testing algorithms with applications to dna library screening*, in DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, vol. 55, Providence, RI, 2000, Amer. Math. Soc., pp. 171–182.

[23] M. OLSON, L. HOOD, C. CONTOR, AND D. BOTSTEIN, *A common language for physical mapping of the human genome*, Science, (1989), pp. 1434–1435.

[24] A. RUDRA AND S. UURTAMO, *Data stream algorithms for codeword testing*, in ICALP (1), 2010, pp. 629–640.

[25] C. E. SHANNON, *A mathematical theory of communication*, Bell System Tech. J., 27 (1948), pp. 379–423, 623–656.

[26] M. SOBEL AND P. A. GROLL, *Group testing to eliminate efficiently all defectives in a binomial sample*, Bell System Tech. J., 38 (1959), pp. 1179–1252.

[27] ———, *Binomial group-testing with an unknown proportion of defectives*, Technometrics, 8 (1966), pp. 631–656.

[28] J. WEBSITE, *Juniper M160 internet backbone router*, 2011. http://www.juniper.net/techpubs/qrc/m160-qrc.pdf.

[29] J. K. WOLF, *Born again group testing: multiaccess communications*, IEEE Transaction on Information Theory, IT-31 (1985), pp. 185–191.

[30] X. XIN, J.-F. F. RUAL, T. HIROZANE-KISHIKAWA, D. E. HILL, M. VIDAL, C. BOONE, AND N. THIERRY-MIEG, *Shifted transversal design smart-pooling for high coverage interactome mapping.*, Genome research, 19 (2009), pp. 1262–1269.