TR 00-007

A Survey on Combinatorial Group Testing Algorithms with Applications to DNA Library Screening

Hung Q. Ngo and Ding-zhu Du

February 07, 2000

A Survey on Combinatorial Group Testing Algorithms with Applications to DNA Library Screening

Hung Q. Ngo* Ding-Zhu Du*

Jan 7, 2000

Abstract

In this paper, we give an overview of Combinatorial Group Testing algorithms which are applicable to DNA Library Screening. Our survey focuses on several classes of constructions not previously discussed in the literature, provides a general view on pooling design constructions and poses several open questions arising from this view.

1 Introduction

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 [22]. If the test result for a pool (of clones) is negative,

*Department of Computer Science and Engineering, University of Minnesota, 200 Union street, EE/CS Building, room 4-192, Minneapolis, MN 55455, USA. e-mail: {hngo, dzd}@cs.umn.edu. Support in part by by the National Science Foundation under grant CCR-9530306.

indicating that no clone in the pool contains the probe, then no further tests are needed for the clones in the pool.

This problem is just an instance of the general group testing problem, in which a large population of *items* containing a small set of *defectives* are to be tested to identify the defectives efficiently. We assume some testing mechanism exists which if applied to an arbitrary subset of the population gives a *negative outcome* if the subset contains no defective and *positive outcome* otherwise. Objectives of group testing vary from minimizing the number of tests, limiting number of pools, limiting pool sizes to tolerating a few errors. It is conceivable that these objectives are often contradicting, thus testing strategies are application dependent.

Group testing algorithms can roughly be divided into two categories : *Combinatorial Group Testing* (CGT) and *Probabilistic Group Testing* (PGT). In CGT, it is often assumed that the number of defectives among n items is equal to or at most d for some fixed positive integer d. In PGT, we fix some probability p of having a defective. If the pools are simultaneously tested s times, with later test pools collected based on previous test results, then the CGT 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 non-adaptive is equivalent to being 1-stage. A group testing algorithm is *error tolerant* if it can detect or correct some e errors in test outcomes. Test errors could be either $0 \to 1$, i.e. a negative pool is identified as positive, or $0 \to 1$ in the contrast.

Library screening applications introduce several new constraints to group testing. Firstly, s-stage group testing algorithms with small s (e.g. ≤ 2) are often preferable [3, 2]. The common requirement is to have an adaptive algorithm. Secondly, DNA screening is error prone since the pools have to be purified before probing. Hence, sometime tolerating several errors is desirable [3]. Lastly, assembling pools is costly, so sometime robots are used to assemble the pools. This makes coordinating the pools with some physical arrangement of clones (such as a grid) important.

There are three related surveys previously done in this area. The first was a survey from Dyachkov and Rykov (1983, [10]) done in the context of superimposed codes. The second was a monograph by Du and Hwang (1993, [8]), which gave a nice account of CGT algorithms. The third was an article by Balding et al. (1995, [2]), which comparatively surveyed certain classes of non-adaptive algorithms.

In this paper, we give an overview of Combinatorial Group Testing Algorithms with applications to DNA Library Screening. Our survey focuses on several classes of constructions not previously discussed in the literature, provides a general view on pooling design constructions and poses several open questions arising from this view.

The rest of the paper is organized as follows. Section 2 fixes up basic definitions and notations needed for the rest of the paper. It also gives a taxonomy of non-adaptive group testing algorithms from which later sections are organized. Section 3 discusses deterministic algorithms. Section 4 provides a new general perspective on constructing a class of deterministic pooling designs, from which several open problems popped up naturally. Section 5 presents random algorithms, and section 6 introduces error-tolerance group testing algorithms. Section 7 concludes the paper.

2 Preliminaries

2.1 The Matrix Representation

We first emphasize that we are concerned only with combinatorially non-adaptive group testing strategies, for DNA library screening applications prefer parallel tests as we have mentioned earlier. The "combinatorial" part comes from the assumption that there are at most d defectives in a population of n items.

Consider a $t \times n$ 01-matrix M. Let R_i and C_j denote row i and column j respectively. Abusing notation, we also let R_i (resp. C_j) denote the set of column (resp. row) indices corresponding to the 1-entries. The *weight* of a row or a column is the number of 1's it has. M is said to be *d-disjunct* if the union of any d columns does not contain another. A d-disjunct $t \times n$ matrix M can be used to design a non-adaptive group testing algorithm on n items by associating the columns with the items and the rows with the pools to be tested. If $M_{ij} = 1$ then item j is contained in pool i (and thus test i). If there are no more than d defectives and the test outcomes are error-free, then it is easy to see that the test outcomes uniquely identify the set of defectives. We simply identify the items contained in negative pools as *negatives* (good items) and the rest as *positives* (defected items). Notice that d-disjunct property implies that each set of $\leq d$ defectives corresponds uniquely to a test outcome vector, thus decoding test outcomes involves only a table lookup. The design a d-disjunct matrix is thus also naturally called *non-adaptive pooling design.* We shall use this term interchangeably with the long "non-adaptive combinatorial group testing algorithm".

Let $S(\bar{d}, n)$ denotes the set of all subsets of n items (or columns) with size at most d, called the set of *samples.* For $s \in S(\bar{d}, n)$, let $P(s)$ denote the union of all columns corresponding to s. A pooling design is e-error detecting (correcting) if it can detect (correct) up to e errors in test outcomes. In other words, if a design is e-error detecting then the test outcome vectors form a t-dimensional binary code with minimum Hamming distance at least $e+1$. Similarly, if a design is e-error correcting then the test outcome vectors form a t-dimensional binary code with minimum Hamming distance at least $2e+1$. The following remarks are simple to see, however useful later on.

Remark 1 *Suppose M has the property that for any $s, s' \in S(\bar{d}, n), s \neq s'$, $P(s)$ and $P(s')$ viewed as vectors have Hamming distance $\geq k$. In other words, $|P(s) \oplus P(s')| \geq k$ where $\oplus$ denotes the symmetric difference. Then, M is $(k-1)$-error detecting and $\lfloor \frac{k-1}{2} \rfloor$-error correcting.*

Remark 2 *M being d-disjunct is equivalent to the fact that for any set of $d+1$ distinct columns $C_{j_0}, \dots C_{j_d}$ with one column (say C_{j_0}) designated, C_{j_0} has a 1 in some row where all C_{j_k}'s, $1 \leq k \leq d$ contain 0's.*

An important question to ask is "given n items with at most d defectives, at least how many tests are needed to identify the defectives?" The best asymptotic answer to this question is dated back to Dyachkov and Rykov (1982, [9]) and Dyachkov, Rykov and Rashad (1989, [11]), which can be summarized by the following theorem.

Theorem 1 *Let $t(d, n)$ denote the minimum number of pools needed for the $S(\bar{d}, n)$ problem, then as $n \to \infty$ and $d \to \infty$*

$$\frac{d^2}{2 \log d}(1 + o(1)) \log n \leq t(d, n) \leq d^2 \log e(1 + o(1)) \log n$$

2.2 A Taxonomy of Non-Adaptive Pooling Designs

We give here a tentative taxonomy of non-adaptive pooling design as follows, from which later sections are organized

1. *Deterministic Designs.* This refers to the fact that every pool is deterministically determined. These designs can be further categorized into

   (i) Set-packing designs.

   (ii) Transversal designs.

   (iii) Designs whose d-disjunct matrices are directly constructed.

2. *Random Designs.* In these designs, some or all of entries are randomly determined with parameterized probabilities, which could be optimized based on certain objective function(s). The categories are :

   (i) Random matrices.

   (ii) Random weight-w designs.

   (iii) Random size-k designs.

   (iv) Random designs which come from deterministic designs.

3. *Error Tolerance Designs.* Although these designs are either deterministic or random, they are worth being paid special attention.

3 Deterministic Pooling Designs

3.1 Set Packing Designs

First noted by Kautz and Singleton back in 1964 [16], packing designs with certain parameters can be used to construct disjunct matrices. We first give some basic definitions. A t-(v, k, λ) *packing design* is a collection $\mathcal{F}$ of k-subsets (called *blocks*) of $[v] := \{1, 2 \ldots, v\}$ such that any t-subset of $[v]$ is contained in at most λ members of $\mathcal{F}$. One useful situation for us is when $\lambda = 1$, in which case the packing is called a (v, k, t)-packing. Notice that $\lambda = 1$ means that no two members of $\mathcal{F}$ have t elements in common. Thus, by Remark 2 if $k > d(t-1)$ a d-disjunct matrix M can be constructed from a (v, k, t)-packing by simply indexing M's columns by the blocks and M's rows by members of $[v]$. Moreover, by Remark 1 we see that if $k = d(t-1) + q + 1$ ($q \geq 0$) then M is q-error detecting and $\lfloor \frac{q}{2} \rfloor$-error correcting.

Naturally, the basic problem of packing design is to find the *packing number* $D_\lambda(v, k, t)$, the size of a maximum $t - (v, k, \lambda)$ packing design. We write $D(v,k,t)$ instead of $D_1(v,k,t)$ when $\lambda = 1$. Maximum sized (v,k,t)-packings induce very good pooling designs [2]. Unfortunately, very little is known about optimal packing designs. Most of what we know are for small values of k and t. Mills and Mullin [19] gave a nice account on packing designs. To give the reader a sense of how difficult this problem is, we quote a result on $D(v,k,t)$ as follows. From the theorem, it is conceivable that finding optimal set packing is just as hard as *the main coding theory problem* [24].

Theorem 2 *Let $A(n,d,w)$ denote the size of a maximum constant w-weight binary (n,d)-code, then*

$$D(v,k,t) = A(v, 2k-2t+2, k)$$

Let

$$U_\lambda(v,k,t) = \left\lfloor \frac{v}{k} \left\lfloor \frac{v-1}{k-1} \cdots \left\lfloor \frac{v-t+1}{k-t+1}\lambda \right\rfloor \right\rfloor \right\rfloor$$

then Schőnheim [25] observed that $D_\lambda(v,k,t) \leq U_\lambda(v,k,t)$. When the design is any t-(v,k,λ) design then we have equality. In particular, since we want $\lambda = 1$, Steiner Triple Systems (2-$(v,3,1)$ designs) and Steiner Quadruple Systems (3-$(v,4,1)$ designs) could be used to construct disjunct matrices with small d's. Finite projective planes and affine planes are also t-designs with $\lambda = 1$ but they don't give good pooling designs (too many tests). The only other noticeable result which concerns us is from Brouwer [6], who determines all values of $D(v,4,2)$. For a comprehensive treatment on design theory, the reader is referred to a very nice book by Beth, Jungnickel and Lenz [5].

3.2 Transversal Designs

The simplest form of transversal designs is called the *grid design*. To facilitate the use of robots for pool assembling, the clones can be arranged into rows and columns of a set of $r \times c$ grids, where each row and column contributes a pool. For simplicity, we can assume $rc \mid n$. Clearly, ambiguity can occur if there are more than one positive clone. The simplest example is when there are two positives, say a and b, lying on different rows and columns of a grid G. Obviously, testing G alone is not enough to identify a and b because the two clones c and d collinear with both a and b are also candidates. To resolve ambiguity, we wish to rearrange G into another grid (giving additional pools) so that c and d are not collinear with both a and b anymore. More grids are needed if there are 3 or more positive clones. In fact, if we require a stronger condition that no two clones are collinear twice, called the *unique collinearity condition*, then Hwang [14] showed that the existence of the grids is equivalent to the existence of certain set of mutually orthogonal Latin squares.

Barillot et al. [4] generalized this idea to k-dimensional grids, where each intersection point could be viewed as a vertex of the k-cube. A new grid G' can be obtained from an old grid G by a linear transformation represented by a matrix $A_{d\times d}$. Thus a vertex $x = (x_1, \ldots x_d)^T$ of G is mapped to vertex Ax of G'. A third grid could either be obtained by using A twice (with transformation matrix A^2) or using a different transformation matrix B (with transformation matrix AB). They also extended the 2-dimensional grid to higher dimension. The set of hyperplanes could be taken as pools, however the pool size is usually large. Reducing pool size by taking lower dimension is possible but that increases the number of tests. Pros and cons of this approach have not been studied.

The general case of *transversal design* was mentioned by Balding et al. in [2]. Basically a pooling design is transversal if the pools can be partitioned into parts, each of which is a partition of the clone population. Clearly the hypercube design is a special case of transversal designs. Not much has been studied toward this general direction. Connections of this problem to coding theory is also specified in [2].

3.3 Direct Constructions

Macula [17, 18] gave the following construction of a d-disjunct matrix. Let $\delta(m, d, k)$ be a 01-matrix whose rows are indexed by the d-subsets of $[m]$ and whose columns are indexed by the k-subsets of $[m]$ where $\frac{m}{2} \geq k > d \geq 1$ are integers. $\delta(m, d, k)_{ij} = 1$ iff the i^{th} d-subset is contained in the j^{th} k-subset. It is easy to see that $\delta(m, d, k)$ is d-disjunct with $\binom{m}{d}$ rows and $\binom{m}{k}$ columns.

The number of tests to number of items ratio of $\delta(m, d, k)$'s $(\binom{m}{d}/\binom{m}{d})$ is not so good in terms of the random bound given in Theorem 1. However, Macula showed that with high probability $\delta(m, 2, k)$ could solve the $S(\bar{d}, n)$ problem, effectively converting a deterministic construction to a probabilistic (random) one. This point will be discussed further in a later section. In addition, m and k could be chosen carefully in certain cases to suit one's need.

However, the method of choosing these parameters needs more thorough analysis than just trial and error.

4 On Constructions of d-disjunct Matrices

In set packing designs, the matrix M was row indexed by all elements of a $[v]$ set, and column indexed by selected k-subsets of $[v]$. Looking at this at a different angle, the rows were indexed by all points at rank 1 and columns by sampled points at rank k of the Boolean Algebra lattice B_v (see Figure 1) of all subsets of $[v]$.

On the other hand, Macula's construction involves taking all points at rank d as rows and rank k as columns of our d-disjunct matrix. Macula's design rate wasn't so good because number of points at level d is too large. However, if we pick points at lower levels than d to be the rows, then the matrix is not d-disjunct anymore.

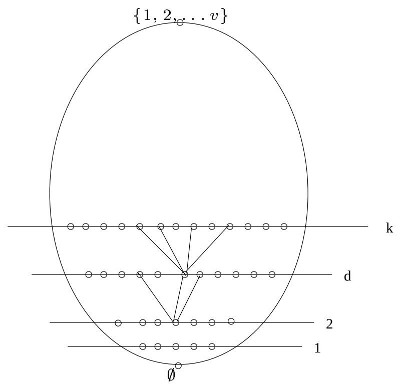

Figure 1: The Boolean Algebra Lattice

Stretching this line of reasoning, one might hope to somehow take sampled points at different ranks of B_v, not taking *all* points. Ngo and Du (1999, [21]) took this approach and gave the following construction. Given integers $m > k > d \geq 1$. A matching of size l in K_v is called an l-matching. Let $M(m, k, d)$ be a 01-matrix whose rows are indexed by the set of all d-matchings on K_{2m}, and whose columns are indexed by the set of all k-matchings on K_{2m}. All matchings are to be ordered lexicographically. $M(m, k, d)$ has a 1 in row i and column j if and only if the i^{th} d-matching is contained in the j^{th} k-matching. The fact that $M(m, k, d)$ is d-disjunct is not difficult to be seen.

Noticing that a k-matching is a k subset of $\left[\binom{2m}{2}\right]$, because the set of edges of K_{2m} is exactly $\left[\binom{2m}{2}\right]$. From the above observation, this construction could be seen as taking from $B_{\left[\binom{2m}{2}\right]}$ sampled points at rank d as rows and sampled points at rank k as columns. Ngo and Du also showed that $M(m,k,d)$ is 3-error detecting and 1-error correcting.

On another dimension, the Boolean Algebra is clearly not the only lattice we have to work on. Some obvious question arising would be

1. Besides the Boolean Algebra B_v, what are other lattices we can use ? For example, one candidate is $C_{v,u}$, the lattice of all v tuples of Z_u, which is a generalization of B_v, since $B_v = C_{v,2}$.

2. Which conditions must hold to pick some two levels of the lattice to construct d-disjunct matrices ? To avoid being too vague and for the ease of analysis, we could restraint ourselves to the lattices with some regularity constraint. An example would be to work on lattices where the number of points covering a point p at rank k and the number of points covered by p depends only on k.

3. In terms of error tolerance properties, can we from the lattice infer some information about the error correcting and detecting capability of the matrix being constructed ?

With respect to question 1, Ngo and Du [21] found that picking points at levels d and k of the lattice of all subspaces of $GF(q)^v$ would also work.

5 Random Pooling Designs

Random designs refer to the designs whose matrices are randomly determined in some manner. The fact that a design is nondeterministic means that it is possible for some positives and negatives not to be identified. Let M be a random $t \times n$ matrix, our algorithm of identifying the defectives is the same as before, namely pointing those items contained in negative tests as negative (these are called *resolved negatives*). Clearly, an item in a positive pool where all others in the pool are resolved negatives must be positive. These positive items are said to be *resolved positives*. Let $\bar{N}$ ($\bar{P}$) denote the number of unresolved negatives (positives). Balding et al. introduced several criteria to compare designs such as $P(\bar{N} = \bar{P} = 0)$, $P(\bar{N} = 0)$, $E(\bar{P})$, and

$E(\bar{N})$, where $P(X = j)$ is the probability of $X = j$ and $E(X)$ is the expected value of a random variable X. We would like the probabilities to be as close to 1 as possible and the expected values to be as small as they can get.

5.1 Random Matrices

Erdős (as usual) and Renyi (1963, [12]) first introduced random methods in search problems. Much later, Sebő (1985, [26]) adopted the idea to group testing. To construct a random disjunct matrix M, we simply assign 1 to an entry of M with some fixed probability p. Given n and d, p and t could be chosen properly so that the probability of M being d-disjunct is higher than some certain tolerable threshold.

Although this method is not used in practice, partially due to its bad performance [2], the idea can be used to obtain very good bounds on the number $t(d, n)$. Theorem 1 is an example of such random bounds.

5.2 Random Weight-w Designs

If a clone is contained in no pool, we don't have any information above the clone. If a clone is contained in every pool and it happens to be positive, all tests turn out to be positive and thus the amount of information we get is also zero. On the same line of reasoning, a design with a clone contained in too many or too less number of tests is not good. Moreover, if the number of pools containing a clone varies, then the analysis would be very tedious if not impossible. Consequently, it is reasonable to attempt constructing random matrices with some constant weight w, where w could be chosen to optimize some of the efficiency criteria. This could be done by assigning the columns randomly to w-subsets of $[t]$. These designs are called *random weight-w designs*. Let the corresponding probabilities and expected values be denoted by $P_w(\cdot)$ and $E_w(\cdot)$ respectively. Let $W(i)$ denote the probability that a particular set of i pools is exactly the set of pools not containing any positive clones. The following formulas were obtained by Bruno et al. [7], and Hwang [15].

$$W(i) = \sum_{h=i}^{t} (-1)^{h-i} \binom{t-i}{h-i} \left[\frac{\binom{t-h}{w}}{\binom{t}{w}} \right]^d \tag{1}$$

$$P_w(\bar{N} = j) = \sum_{i=0}^{t} \binom{t}{i} W(i) \binom{n-d}{j} \left[1 - \frac{\binom{t-i}{w}}{\binom{t}{w}}\right]^{n-d-j} \left[\frac{\binom{t-i}{w}}{\binom{t}{w}}\right]^{j} \tag{2}$$

$$E_w(\bar{N}) = (n-d) \sum_{i=1}^{w} \binom{w}{i} \left[\frac{\binom{t-i}{w}}{\binom{t}{w}}\right]^{d} \tag{3}$$

An open question is to find w so that $E_w(\bar{N})$ is minimized. Notice that these formulas were calculated ignoring the fact that in practice we don't want identical columns in the matrix. The reason being that taking into account this fact makes the calculation really difficult. Bruno et al. [7] also indicated that random weight-w designs perform better than the random design discussed earlier.

5.3 Random Size-k Designs

Dually, instead of reasoning on the columns of M we could do the same on the rows of M. A pool containing too few clones is wasted if these clones are negatives, while a pool containing too many clones gives little information if there is a positive clone in it. Hence, we could as well randomly choose the rows of M with some constant size k uniformly. Similar formulas as those in the last sections were obtained by Hwang (1999, [15]):

$$\begin{aligned} P_k(\bar{N} = n-d) &= \left[1 - \frac{\binom{n-d}{k}}{\binom{n}{k}}\right]^{t} & (4) \\ P_k(\bar{N} = j) &= \sum_{i=0}^{t} \binom{t}{i} \left[\frac{\binom{n-d}{k}}{\binom{n}{k}}\right]^{i} \left[1 - \frac{\binom{n-d}{k}}{\binom{n}{k}}\right]^{t-i} \\ &\quad \cdot \sum_{l=j}^{n-d} (-1)^{l-j} \binom{n-d}{l} \left[\frac{\binom{n-d-l}{k}}{\binom{n-d}{k}}\right]^{i} \\ &\quad \text{for } 0 \le j < n-d & (5) \end{aligned}$$

In the same paper, Hwang also gave formulas to compute $E_x(\bar{P})$ for $x \in \{p, w, k\}$. Here $E_p(X)$ denote the expected value of X when M is constructed using the first random method with probability p.

5.4 Random Designs from Deterministic Designs

Macula [18] showed that his matrix $\delta(m, 2, k)$ could be used to solve the $S(\bar{d}, n)$ problem with high probability of success. Clearly, this is desirable since the test to item ratio of $\delta(m, 2, k)$ is smaller than that of $\delta(m, d, k)$ in general. The probability, denoted by $P_\delta(n, d, k)$, can be shown to be

$$P_\delta(n,d,k) \geq \left[\frac{\sum_{i=1}^{k} (-1)^{i+1} \binom{k}{i} \binom{\binom{n-i}{k}}{d-1}}{\binom{\binom{n}{k}-1}{d-1}} \right]^d$$

For example, when $d = 5$ and $n \approx 1,000,000$ we can pick $\delta(44, 2, 5)$, which has 946 rows (tests), 1,086,008 columns (items), and $p(44, 5, 5) \geq .97107$.

Borrowing this idea, Ngo and Du [21] also showed that $M(m, k, 2)$ could be used to solve $S(\bar{d}, n)$ with probability $P_M(n, k, d)$ of giving the right answer, where

$$P_M(m,k,d) \geq \left[\frac{\sum_{j=1}^{k} (-1)^{j+1} \binom{k}{j} \binom{\sum_{i=0}^{j} (-1)^i \binom{j}{i} g(m-i,k-i)}{d-1}}{\binom{g(m,k)-1}{d-1}} \right]^d$$

Here, $g(m, l) = \binom{2m}{2l} \frac{(2d)!}{2^d d!}$. For example, $p(8, 6, 9) \geq 98.5\%$, with the number of defectives $d = 9$, the number of items $n = g(8, 6) = 18,918,900$ and the number of test $t = g(8, 2) = 5460$.

One can see from these formulas that the efficiency benchmarks to compare pooling designs often involve complicated, hypergeometric types of formulas arising from inclusion exclusion enumerations. This makes the analysis difficult and tedious. Usually, what we can do is to plug in some particular values and do manual comparison, which is clearly not satisfactory theoretically. More work needs to be done in asymptotic analysis of these formulas in order to give any satisfaction result if at all.

6 Error Tolerance Pooling Designs

As we have mentioned earlier, when DNA probing could be error prone, which leads us to the greater challenge of designing pools that could tolerate some number of errors. This problem is the non-adaptive version of the *searching* *game* initiated by Ulam [28] back in 1976. Ulam's problem was to determine a chosen number u out of $[n]$ using the minimum number of questions of the form: Is $u \in S$, $S \subseteq [n]$. Moreover, the responder could lie once or twice. In general, the questions and answers could be q-ary, i.e. each question is a partition of $[n]$ into q parts and each answer points out which part(s) any of d unknowns belong to. Up to e lies is allowed. It is easy to see that our problem is the non-adaptive version of this so-called *q-ary search problem with lies* where $q = 2$. Although quite a lot of research effort has been put on solving this problem, we only have solutions for several special cases where q and e are small.

Adaptively, when $d = 1, q = 2$ Pelc [23] solved the case $e = 1$, Guzicki [13] solved the case $e = 2$, and Spencer [27] provided a nearly optimal solution (up to a constant) for general e. The q-ary case (with $d = 1$) was consider by Aigner [1] and Muthukrishnan [20] with complete solutions.

Non-adaptively, several author have noticed that when $d = 1$, the design is equivalent to an e-error correcting code. Balding and Torney [3] studied several instances of the problem when $d \leq 2$ where an optimal strategy is possible. Macula [18] showed that his construction is error tolerable up to certain calculatable probability. Ngo and Du construction [21] was shown to be 3-error detecting and 1-error correcting in the worst case, but can tolerate more errors on average.

We need deeper results and new breakthroughs in order to improve our present knowledge of the most general case of the problem, especially in the non-adaptive case. For example, we need good bounds similar to those in Theorem 1 given the number of items n, maximum number of defectives d and number of errors e.

7 Conclusions

In this paper, we have given an overview of up-to-date results on Combinatorial Group Testing algorithms which are applicable to DNA library screening. We have been focusing more on new classes of constructions not previously discussed and pointed out directions to generalize existing results. We also have discussed some related open questions popped up in this area.

Finally, we would like to conclude that this is a young and active field with deep connections to Coding Theory, Design Theory. We also would like to point out that the theory Distance Regular Graphs, in particular Association

Schemes, should play an important role in improving our pooling designs.

References

[1] M. Aigner, *Searching with lies*, J. Combin. Theory Ser. A, 74 (1996), pp. 43–56.

[2] 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.

[3] D. J. Balding and D. C. Torney, *Optimal pooling designs with error detection*, J. Combin. Theory Ser. A, 74 (1996), pp. 131–140.

[4] E. Barillot, B. Lacroix, and D. Cohen, *Theoretical analysis of library screening using a n-dimensional pooling strategy*, Nucl. Acids Res., (1991), pp. 6241–6247.

[5] T. Beth, D. Jungnickel, and H. Lenz, *Design theory*, Cambridge University Press, Cambridge, 1986.

[6] A. E. Brouwer, *Optimal packings of K_4's into a K_n*, J. Combin. Theory Ser. A, 26 (1979), pp. 278–297.

[7] W. J. Bruno, E. Knill, D. J. Balding, D. C. Bruce, N. A. Doggett, W. W. Sawhill, R. L. Stallings, C. C. Whittaker, and D. C. Torney, *Efficient pooling designs for library screening*, Genomics, (1995), pp. 21–30.

[8] D. Z. Du and F. K. Hwang, *Combinatorial group testing and its applications*, World Scientific Publishing Co. Inc., River Edge, NJ, 1993.

[9] A. G. Dyachkov and V. V. Rykov, *Bounds on the length of disjunctive codes*, Problemy Peredachi Informatsii, 18 (1982), pp. 7–13.

[10] ———, *A survey of superimposed code theory*, Problems Control Inform. Theory/Problemy Upravlen. Teor. Inform., 12 (1983), pp. 229–242.

[11] A. G. Dyachkov, V. V. Rykov, and A. M. Rashad, *Superimposed distance codes*, Problems Control Inform. Theory/Problemy Upravlen. Teor. Inform., 18 (1989), pp. 237–250.

[12] P. Erdős and A. Rényi, *On two problems of information theory*, Magyar Tud. Akad. Mat. Kutató Int. Közl., 8 (1963), pp. 229–243.

[13] W. Guzicki, *Ulam's searching game with two lies*, J. Combin. Theory Ser. A, 54 (1990), pp. 1–19.

[14] F. K. Hwang, *An isomorphic factorization of the complete graph*, J. Graph Theory, 19 (1995), pp. 333–337.

[15] ———, *Random size-k pool designs with distinct columns*, (1999). preprint.

[16] W. H. Kautz and R. C. Singleton, *Nonrandom binary superimposed codes*, IEEE Trans. Inf. Theory, 10 (1964), pp. 363–377.

[17] A. J. Macula, *A simple construction of d-disjunct matrices with certain constant weights*, Discrete Math., 162 (1996), pp. 311–312.

[18] ———, *Probabilistic nonadaptive group testing in the presence of errors and dna library screening*, Annals of Combinatorics, (1999), pp. 61–69.

[19] W. H. Mills and R. C. Mullin, *Coverings and packings*, in Contemporary design theory, Wiley, New York, 1992, pp. 371–399.

[20] S. Muthukrishnan, *On optimal strategies for searching in presence of errors*, in Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (Arlington, VA, 1994), New York, 1994, ACM, pp. 680–689.

[21] H. Q. Ngo and D. Z. Du, *New constructions of non-adaptive pooling designs*, (1999). preprint.

[22] M. Olson, L. Hood, C. Contor, and D. Botstein, *A common language for physical mapping of the human genome*, Science, (1989), pp. 1434–1435.

[23] A. Pelc, *Solution of Ulam's problem on searching with a lie*, J. Combin. Theory Ser. A, 44 (1987), pp. 129–140.

[24] S. Roman, *Coding and information theory*, Springer-Verlag, New York, 1992.

[25] J. Schönheim, *On maximal systems of k-tuples*, Studia Sci. Math. Hungar, 1 (1966), pp. 363–368.

[26] A. Sebő, *On two random search problems*, J. Statist. Plann. Inference, 11 (1985), pp. 23–31.

[27] J. Spencer, *Ulam's searching game with a fixed number of lies*, Theoret. Comput. Sci., 95 (1992), pp. 307–321.

[28] S. M. Ulam, *Adventures of a mathematician*, Charles Scribner's Sons, New York, 1976.