JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 14, Number 9, 2007

Pp. 1208–1216
DOI: 10.1089/cmb.2007.0052

# Efficient Constructions of Disjunct Matrices with Applications to DNA Library Screening

YONGXI CHENG[1] and DING-ZHU DU[2]

## ABSTRACT

**The study of gene functions requires a DNA library of high quality, such a library is obtained from a large mount of testing and screening. Pooling design is a very helpful tool for reducing the number of tests for DNA library screening. In this paper, we present two Las Vegas algorithms for efficient constructions of $d$-disjunct and $(d; z)$-disjunct matrices respectively. These new constructions can be directly applied to construct error-free and error-tolerant pooling designs.**

**Key words:** disjunct matrices, DNA library screening, Las Vegas algorithm, nonadaptive group testing, pooling designs.

## 1. INTRODUCTION

THE BASIC TASK of DNA library screening is to determine which *clone* (a DNA segment) from the library contains which *probe* from a given collection of probes. If a clone contains a probe, then the clone is said to be *positive* for the probe, otherwise it is said to be *negative* for that probe. Since in practice checking each clone-probe pair is expensive and usually only a few clones in the library contain a given probe, clones are pooled together to be tested against each probe. An example is when Sequenced-Tagged Site markers (also called STS probes) are used (Olson et al., 1989). For a given probe, if the test result for a pool of clones is negative, then no clone in the pool contains the probe, and we do not need any further tests for the clones in the pool. If the test result for a pool is positive, then there exists at least one clone in the pool containing the given probe.

The above problem is an instance of the *combinatorial group testing* problem, in which there are $n$ items each can be either *positive* (used to be called *defective*) or *negative* (used to be called *good*), and the number of positive items is upper bounded by an integer $d$. We assume there exists some testing mechanism which if applied to an arbitrary subset of the items gives a *positive outcome* (denoted by 1) if the subset contains at least one positive item and a *negative outcome* (denoted by 0) otherwise. The goal is to identify all positive items using minimum number of tests.

Group testing algorithms can be either *adaptive* or *nonadaptive*. An adaptive algorithm conducts the tests one by one and allows to design a later test using the outcomes of all previous tests. A nonadaptive

[1]Department of Computer Science, Tsinghua University, Beijing, China.
[2]Department of Computer Science, University of Texas at Dallas, Richardson, Texas.

algorithm specifies all tests simultaneously, forbidding designing tests using the outcome information of other tests. In general, nonadaptive algorithms require more number of tests because we are not allowed to use the outcomes of other tests for more efficient test designs, but require less time since all tests can be performed in paralell. Between completely adaptive and nonadaptive algorithms, there are $s$-stage algorithms in which all tests in each stage must be specified simultaneously, but the tests in different stages can be adaptive.

A group testing algorithm is *error-tolerant* if it can detect or correct some errors in the test outcomes. Previous works on error-tolerant group testing algorithms are those of, among others, Muthukrishnan (1994), Aigner (1996), Balding and Torney (1996), Macula (1997, 1999), and De Bonis et al. (2005).

In the application to DNA library screening, a group testing algorithm is called a *pooling design*, and the composition of each test is called a *pool*. While it is still important to minimize the number of tests, there are two other goals. First, in practice we prefer nonadaptive group testing (NGT) algorithms in which all tests are performed simultaneously, so that the total time spent is tolerable (sometimes a pooling design with $s$ stages for small $s$ is adopted). Second, DNA screening is error prone, so it is desirable to design error-tolerant algorithms. For a comprehensive discussion of this topic, the reader is referred to the book by Du and Hwang (2006).

A NGT algorithm can be represented as a 0-1 matrix $M = (m_{ij})$ in which the columns are associated with the items and the rows are associated with the tests, and $m_{ij} = 1$ indicates that item $j$ is contained in test $i$. The outcomes of the tests can also be represented as a 0-1 vector where 0 indicates a negative outcome and 1 indicates a positive outcome, which is called the *outcome vector*. Denote by $D$ the set of columns associated with positive items, and denote by $U(D)$ the union of these columns (i.e., the bitwise boolean sum of these 0-1 column vectors), it is easy to see that the outcome vector is equal to $U(D)$ if there is no error in the test outcomes. Given the matrix representation of a NGT algorithm and the outcome vector, the process of identifying all the positive items is called *decoding*.

A 0-1 matrix is said to be *$d$-disjunct* if no column is contained in the union of any other $d$ columns. If the matrix represents a NGT algorithm is $d$-disjunct, the number of positive items is no more than $d$ and the test outcomes are error-free, then we have the following easy decoding method: for each column $c$, $c$ corresponds to a positive item if and only if $c$ is contained in the outcome vector $v$. $d$-disjunct matrices form a basis for NGT algorithms, and the design of a $d$-disjunct matrix is also called a *nonadaptive pooling design*.

However, when there are errors in the test outcomes, the above decoding method no longer works. To deal with the case where there are errors in the test outcomes, we require the matrix to be *$(d; z)$-disjunct*. A 0-1 matrix is said to be $(d; z)$-disjunct (D'yachkov et al., 1989; Macula, 1997) if for any column $c$ and any $d$ other columns, $c$ has at least $z$ elements not covered by the union of these $d$ columns. Thus, $d$-disjunct is just $(d; 1)$-disjunct. $(d; z)$-disjunct matrices form a basis for error-tolerant NGT algorithms, the following is a good illustration showing the $\lfloor \frac{z-1}{2} \rfloor$-error-correcting ability of a $(d; z)$-disjunct matrix. If the matrix represents the algorithm is $(d; z)$-disjunct, there are no more than $d$ positive items and at most $\lfloor \frac{z-1}{2} \rfloor$ errors in the test outcomes (i.e., the outcome vector $v$ has no more than $\lfloor \frac{z-1}{2} \rfloor$ error bits), then the following decoding method holds: a column $c$ corresponds to a positive item if and only if $|c \setminus v| \leq \lfloor \frac{z-1}{2} \rfloor$, here $c \setminus v$ denotes the difference of two sets of row indices determined by vectors $c$ and $v$ (i.e., $|c \setminus v|$ is the number of rows having value 1 in $c$ and value 0 in $v$). In this paper, we investigate efficient constructions of both $d$-disjunct and $(d; z)$-disjunct matrices.

### *1.1. Related work*

For asymptotic bounds on disjunct matrices, if we denote $t(d, n)$ to be the minimum number of rows required by a $d$-disjunct matrix with $n$ columns, then $t(d, n) = \Omega(\frac{d^2 \log_2 n}{\log_2 d})$ (D'yachkov and Rykov, 1982; Ruszinkó, 1994; Füredi, 1996). In particular, D'yachkov and Rykov (1982) proved that $t(d, n) \geq \frac{d^2}{2 \log_2 d}(1 + o(1)) \log_2 n$, which is the best lower bound so far. For upper bounds on $t(d, n)$, by using random coding method D'yachkov et al. (1989) proved that for $n$ large, $t(d, n) \leq (\log_2 e)(1 + o(1))d^2 \log_2 n$, which is currently the best. For $(d; z)$-disjunct matrices, let $t(d, n; z)$ denote the minimum number of rows required by a $(d; z)$-disjunct matrix with $n$ columns. For fixed $d$ and $z$, D'yachkov et al. (1989)

studied $\lim \frac{\log_2 n}{t}$ as $n \to \infty$ among others, they proved that $t(d, n; z) \geq C[\frac{d^2 \log_2 n}{\log_2 d} + (z-1)d]$ where $C$ is a constant.

On construction of disjunct matrices, Kautz and Singleton (1964) introduced the construction of disjunct matrices from set packing designs, in the context of superimposed codes. Hwang and Sós (1987) (also cited in Du and Hwang, 2006) gave an explicit construction that results in $t \times n$ $d$-disjunct matrices with $t \leq 16d^2(1 - \log_3 2 + (\log_3 2)\log_2 n) \approx 5.91d^2 + 10.09d^2 \log_2 n$. Other works known on constructing disjunct matrices are those of, among others, Erdös et al. (1985), Macula (1996), D'yachkov et al. (2000), Ngo and Du (2002), Park et al. (2003), Du et al. (2006), and Fu and Hwang (2006).

### *1.2. Our contribution*

We present two Las Vegas algorithms for constructing $d$-disjunct and $(d; z)$-disjunct matrices, respectively. The first algorithm, for given $n$ and $d$, when $d \ln n \gg 1$, constructs a $t \times n$ $d$-disjunct matrix with some pre-specified constant probability in time $O(d^2 n^2 \ln n)$ (can be further improved to expected $O(n^2 \ln n)$ time), with $t = c(1 + O(\frac{1}{\sqrt{d \ln n}}))d^2\ (O(1) + \log_2 n)$, where $c \approx 4.28$ is constant. Compared to the construction of Hwang and Sós (1987), our algorithm reduces $t$ by more than half for $d \ln n$ reasonably large, also from an algorithmic point of view our construction is much more efficient. The second algorithm, for given $n$, $d$ and $z > 1$, when $d \ln n \gg z - 1$, with some pre-specified constant probability and the same running time, constructs a $t \times n$ $(d; z)$-disjunct matrix with $t \approx 4.28(1 + O(\frac{1}{\sqrt{d \ln n}}))d^2$ $(O(1) + \frac{2.30(z-1)}{d} + \log_2 n)$.

To the best of our knowledge, the first algorithm is the first explicit construction of $d$-disjunct matrices that achieves the leading constant $c \approx 4.28$, and runs in time polynomial in both $n$ and $d$ provided $d \ln n \gg 1$; the second algorithm is the first construction of $(d; z)$-disjunct matrices that achieves $t = O(dz + d^2 \log_2 n)$ provided $d \ln n \gg z - 1$.

## 2. A NEW ALGORITHM

In this section we present a Las Vegas algorithm, for given $n$ and $d$, with some constant probability the algorithm returns a $t \times n$ $d$-disjunct matrix, with $t = c(1 + O(\frac{1}{\sqrt{d \ln n}}))d^2(O(1) + \log_2 n)$ when $d \ln n \gg 1$, where $c \approx 4.28$ is constant. This result can be directly applied to construct nonadaptive pooling designs, in which $n$ denotes the number of items and $d$ denotes an upper bound on the number of positives.

The main idea of the algorithm is, similar to those in Kautz and Singleton (1964), Hwang and Sós (1987), and Fu and Hwang (2006), to control the number of intersections of any two columns to guarantee $d$-disjunctness. The major difference is that we also use randomness in our construction, which turns out to be effective. Roughly speaking, first we construct a larger $t \times n_0$ ($n_0 > n$) random binary matrix, then we remove some of its columns so that in the resultant matrix the weight of each column will not be too small and the intersection of any two columns will not be too large.

For given $n$ and $d$, let $c_1, c_2 > 1$ be constants that will be specified later. Define $q = \frac{1}{c_1} + \frac{1}{c_2}$ which is intended to denote the failure probability of the algorithm (we require $c_1, c_2$ to satisfy $q < 1$). First, we set some parameters that will be useful in the algorithm. Define $n_0 = 2qc_2 n$. Set $\epsilon, \delta > 0$ such that $\ln(1 + \epsilon) = \frac{2\epsilon}{1+\epsilon}$ ($\epsilon \approx 3.92$) and $\delta = \sqrt{\frac{2\epsilon \ln c_1}{(1+\epsilon)d \ln(c_2 n_0)}}$ (we require $c_1, c_2$ to satisfy $\delta < 1$). Assign $p = \frac{1-\delta}{(1+\epsilon)d}$ and $t = \frac{\ln(c_2 n_0)}{p^2 \epsilon} = \frac{(1+\epsilon)^2}{\epsilon(1-\delta)^2} d^2 \ln(c_2 n_0)$. For our algorithm for constructing $d$-disjunct matrices, see Algorithm 1.

---

**Algorithm 1.** Constructing $d$-disjunct matrix $M_{t \times n}$, with parameters $c_1$ and $c_2$ ($\frac{1}{c_1} + \frac{1}{c_2} < 1$):

---

**Step 1.** Construct a random binary matrix $M_{t \times n_0}$ with each cell assigned to be 1 independently with probability $p$.

**Step 2.** For any $1 \leq i \leq n_0$, denote by $w_i$ the weight of column $i$, let $\mu_1 = E[w_i] = pt$, mark column $i$ if $w_i \leq (1 - \delta)\mu_1$.

**Step 3.** For any $1 \le i < j \le n_0$, denote by $w_{i,j}$ the number of rows of $M_{t\times n_0}$ that have 1 at both column $i$ and column $j$ (i.e., $w_{i,j}$ denotes the number of intersections of column $i$ and column $j$), let $\mu_2 = E[w_{i,j}] = p^2 t$, create an edge between columns $i$ and $j$ if $w_{i,j} \ge (1+\epsilon)\mu_2$.

**Step 4.** Remove all marked columns from $M_{t\times n_0}$. For each edge between the remaining columns, remove one of its two columns arbitrarily.

**Step 5.** Let $M$ denote the resulting matrix from the above steps. If $M$ has less than $n$ $(= \frac{n_0}{2qc_2})$ columns, exit and the algorithm fail; else return the first $n$ columns of $M$ as the $d$-disjunct matrix $M_{t\times n}$.

---

In the above setting of parameters, for $d \ln n \gg 1$ we have $\delta = \sqrt{\frac{2\epsilon \ln c_1}{(1+\epsilon)d\ln(2qc_2^2 n)}} \ll 1$, and $\delta \approx \sqrt{\frac{2\epsilon \ln c_1}{(1+\epsilon)d\ln n}}$. Thus, $t = \frac{(1+\epsilon)^2}{\epsilon(1-\delta)^2} d^2 \ln(c_2 n_0) = \frac{(1+\epsilon)^2}{\epsilon(1-\delta)^2 \log_2 e} d^2 \log_2(2qc_2^2 n) \approx \frac{(1+\epsilon)^2}{\epsilon \log_2 e}(1+2\delta) d^2 \log_2(2qc_2^2 n)$ $\approx c(1+t_1 \frac{1}{\sqrt{d\ln n}})\, d^2(t_2 + \log_2 n)$, where $c$, $t_1$ and $t_2$ are positive constants, $c = \frac{(1+\epsilon)^2}{\epsilon \log_2 e} \approx 4.28$, $t_1 = \sqrt{\frac{8\epsilon \ln c_1}{1+\epsilon}}$ and $t_2 = \log_2(2qc_2^2)$.

Also, we can see that as long as Algorithm 1 returns a matrix $M_{t\times n}$ (i.e., there are at least $n$ columns left at Step 5), $M_{t\times n}$ is $d$-disjunct. Since from the algorithm, we know that the weight of any column of $M_{t\times n}$ is greater than $(1-\delta)\mu_1$, and the number of intersections between any pair of columns of $M_{t\times n}$ is less than $(1+\epsilon)\mu_2$. Notice we set $p$ such that $(1-\delta)\mu_1 = d(1+\epsilon)\mu_2$, thus for any column $i$, the union of any $d$ other columns can only cover less than $d(1+\epsilon)\mu_2 = (1-\delta)\mu_1$ 1's of column $i$. Therefore, they can not completely cover column $i$ since $w_i > (1-\delta)\mu_1$.

## 3. ANALYSIS OF ALGORITHM 1

In this section we analyze the success probability and running time of Algorithm 1.

### *3.1. Preliminaries*

We first present two lemmas that will be useful later. The first lemma is the Markov inequality (e.g., see Theorem 3.2 in by Motwani and Raghavan, 1995).

**Lemma 3.1 (Markov Inequality).** *Let $Y$ be a random variable assuming only non-negative values, then for all $t > 0$,*

$$\Pr[Y \ge t] \le \frac{E[Y]}{t},$$

*where $E[Y]$ is the expectation of $Y$.*

The second lemma is commonly known as Chernoff's bounds (Theorems 4.1 and 4.2 in Motwani and Raghavan, 1995).

**Lemma 3.2 (Chernoff's Bounds).** *Let $X_1, X_1, \cdots, X_n$ be independent 0-1 random variables, for $1 \le i \le n$, $\Pr[X_i = 1] = p_i$, where $0 < p_i < 1$. Let $X = \sum_{i=1}^n X_i$, and $\mu = E[X] = \sum_{i=1}^n p_i$. Then, for any $\delta > 0$,*

*(1)* $\Pr[X \ge (1+\delta)\mu] \le [\frac{e^\delta}{(1+\delta)^{1+\delta}}]^\mu$,

*(2)* $\Pr[X \le (1-\delta)\mu] \le e^{-\mu\delta^2/2}$.

The Chernoff's bounds in Motwani and Raghavan (1995) are for strict inequalities, but the same bounds also hold for nonstrict inequalities.

### 3.2. Success probability of Algorithm 1

First, we estimate the expectations of the number of marked columns and the number of edges created in Algorithm 1.

**Lemma 3.3.** *Let $Y$ be the random variable that denotes the number of marked columns in Step 2 of Algorithm 1, then $E[Y] \le \frac{n_0}{c_1}$.*

**Proof.** For $1 \le i \le n_0$, in Step 2 of Algorithm 1, let $Y_i$ be the indicator random variable for the event that column $i$ is marked, that is

$$Y_i = \begin{cases} 1 & \text{column } i \text{ is marked, i.e., } w_i \le (1-\delta)\mu_1; \\ 0 & \text{otherwise.} \end{cases}$$

Since $w_i = M(1,i) + M(2,i) + \cdots + M(t,i)$ is the sum of $t$ independent 0-1 random variables, by applying the Chernoff bound, (2) in Lemma 3.2, we obtain that

$$\Pr[Y_i = 1] = \Pr[w_i \le (1-\delta)\mu_1] \le e^{-\mu_1\delta^2/2}.$$

Since $\mu_1 = E[w_i] = pt$, it follows that $-\mu_1\delta^2/2 = -\frac{\delta^2}{2}pt = -\frac{\delta^2}{2}\frac{1-\delta}{(1+\epsilon)d}\frac{(1+\epsilon)^2}{\epsilon(1-\delta)^2}d^2\ln(c_2n_0) = -\frac{(1+\epsilon)\delta^2}{2\epsilon(1-\delta)}\ d\ln(c_2n_0) = -\frac{(1+\epsilon)}{2\epsilon(1-\delta)}\frac{2\epsilon\ln c_1}{(1+\epsilon)d\ln(c_2n_0)}d\ln(c_2n_0) = -\frac{\ln c_1}{1-\delta} \le -\ln c_1$. Therefore, $\Pr[Y_i = 1] \le e^{-\mu_1\delta^2/2} \le \frac{1}{c_1}$. Notice $Y = \sum_{1\le i\le n_0} Y_i$ and all the $Y_i$'s are i.i.d. random variables, we have $E[Y] = n_0\Pr[Y_1 = 1] \le \frac{n_0}{c_1}$. ∎

**Lemma 3.4.** *Let $m$ be the random variable that denotes the number of edges created in Step 3 of Algorithm 1, then $E[m] < \frac{n_0}{2c_2}$.*

**Proof.** For $1 \le i < j \le n_0, 1 \le k \le t$, in Step 3 of Algorithm 1, define random variable $X_k^{i,j} = M(k,i)M(k,j)$, then $w_{i,j} = X_1^{i,j} + X_2^{i,j} + \cdots + X_t^{i,j}$. Let $X^{i,j}$ be the indicator random variable for the event that there is an edge between column $i$ and column $j$:

$$X^{i,j} = \begin{cases} 1 & \text{there is an edge between column } i \text{ and column } j, \text{ i.e., } w_{i,j} \ge (1+\epsilon)\mu_2; \\ 0 & \text{otherwise.} \end{cases}$$

Since $w_{i,j}$ is the sum of $t$ independent 0-1 random variables, the Chernoff bound, (1) in Lemma 3.2, implies that

$$\Pr[X^{i,j} = 1] = \Pr[w_{i,j} \ge (1+\epsilon)\mu_2] \le \left[\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}}\right]^{\mu_2}.$$

Notice that $\mu_2 = E[w_{i,j}] = p^2t = \frac{1}{\epsilon}\ln(c_2n_0)$, we have $(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}})^{\mu_2} = (\frac{e}{(1+\epsilon)^{\frac{1+\epsilon}{\epsilon}}})^{\ln(c_2n_0)}$. Since $\ln(1+\epsilon) = \frac{2\epsilon}{1+\epsilon}$, it follows that $(1+\epsilon)^{\frac{1+\epsilon}{\epsilon}} = (e^{\frac{2\epsilon}{1+\epsilon}})^{\frac{1+\epsilon}{\epsilon}} = e^2$, thus $\Pr[X^{i,j} = 1] \le (\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}})^{\mu_2} = (e^{-1})^{\ln(c_2n_0)} = \frac{1}{c_2n_0}$. Since $m = \sum_{1\le i<j\le n_0} X^{i,j}$ and all the $X^{i,j}$'s are identically distributed, $E[m] = \binom{n_0}{2}\Pr[X^{1,2} = 1] \le \binom{n_0}{2}\frac{1}{c_2n_0} < \frac{n_0}{2c_2}$. ∎

Define random variable $Z = Y + m$, then $Z$ denotes the most number of columns that may be removed at Step 4. Since $E[Z] = E[Y] + E[m] \le (\frac{1}{c_1} + \frac{1}{2c_2})n_0$ and $q = \frac{1}{c_1} + \frac{1}{c_2}$, it follows that $n_0 - n = (1 - \frac{1}{2qc_2})n_0 = \frac{1}{q}(q - \frac{1}{2c_2})n_0 = \frac{1}{q}(\frac{1}{c_1} + \frac{1}{2c_2})n_0 \ge \frac{1}{q}E[Z]$. By applying the Markov inequality (Lemma 3.1), the probability that there are less than $n$ columns left in the matrix at Step 5 is at most $\Pr[Z > n_0 - n] \le \frac{E[Z]}{n_0-n} \le q$. Therefore, we have established the following theorem.

**Theorem 3.5.** *Given $n$ and $d$, Algorithm 1 with parameters $c_1$ and $c_2$ ($\frac{1}{c_1}+\frac{1}{c_2}<1$) successfully returns a $d$-disjunct $t\times n$ matrix with probability at least $1-q$, where $t=\frac{(1+\epsilon)^2}{\epsilon(1-\delta)^2}d^2\ln(c_2n_0)$ and $q=\frac{1}{c_1}+\frac{1}{c_2}$. In addition, when $d\ln n\gg 1$, $t\approx c(1+t_1\frac{1}{\sqrt{d\ln n}})\ d^2(t_2+\log_2 n)$, where $c$, $t_1$ and $t_2$ are positive constants, $c=\frac{(1+\epsilon)^2}{\epsilon\log_2 e}\approx 4.28$, $t_1=\sqrt{\frac{8\epsilon\ln c_1}{1+\epsilon}}$ and $t_2=\log_2(2qc_2^2)$.*

**Remark 3.6.** *In the framework of Algorithm 1, $\epsilon$ is chosen to minimize the leading constant of $t$, see Appendix A. The idea behind choosing parameters $c_1, c_2, q$ such that $q=\frac{1}{c_1}+\frac{1}{c_2}$ is, for pre-specified success probability $1-q$ and fixed $c_1$ (thus fixed $t_1$ in the approximate expression $t\approx 4.28(1+t_1\frac{1}{\sqrt{d\ln n}})\ d^2(t_2+\log_2 n)$), $c_2$ is assigned $\frac{c_1}{qc_1-1}$ to minimize the additive term $t_2$, see Appendix B.*

For instance, if we set the failure probability $q=0.99$ and want $t_1=1$, by solving the equations we can get $c_1\approx 1.17$, $c_2\approx 7.40$ and $t_2\approx 6.76$. If we set $q=0.99$ and $t_1=2$, then $c_1\approx 1.87$, $c_2\approx 2.19$ and $t_2\approx 3.25$. For $q=0.9$, we have the choices $t_1=1$, $c_1\approx 1.17$, $c_2\approx 22.13$ and $t_2\approx 9.78$, or, $t_1=2$, $c_1\approx 1.87$, $c_2\approx 2.73$ and $t_2\approx 3.75$, etc.

### *3.3. Running time of Algorithm 1*

The time required by Algorithm 1 is dominated by Step 2, which is $\binom{n_0}{2}t=O(d^2n^2\ln n)$ by simply counts the number of intersections between all pairs of columns. In fact, we can obtain an expected $O(n^2\ln n)$ running time by counting intersections along the rows.

For $1\le i<j\le n_0$, denote by $n(i,j)$ the number of intersections between column $i$ and column $j$. Initially, we set $n(i,j)=0$ for all $1\le i<j\le n_0$. For each row $r$ of $M_{t\times n_0}$, let $w$ denote the weight of row $r$, and let $i_1,i_2,\cdots,i_w$ denote the indices of $r$ having value 1, we increase the values of $n(i_a,i_b)$ by 1 for all $1\le a<b\le w$. The expected number of such pairs $(i_a,i_b)$ for each row is $E[\binom{w}{2}]$, thus the expected running time of Step 2 is $tE[\binom{w}{2}]$. Since $w$ has the binomial distribution with parameters $n_0$ and $p$, the expected running time can be estimated to be $tE[\binom{w}{2}]=t\times\frac{p^2}{2}(n_0^2-n_0)=O(n^2\ln n)$.

Theoretically by running the algorithm repeatedly with independent random choices, we can make the failure probability arbitrarily small, at the cost of spending more time. Practically, we can just repeat running the algorithm until a $d$-disjunct matrix is successfully constructed. It is easy to see that the expected number of times that we need to run the algorithm so that a $d$-disjunct matrix is successfully constructed is no more than $\frac{1}{1-q}$, and once a $d$-disjunct matrix is constructed, it can be used as a NGT algorithm as many times as needed.

## 4. ERROR-TOLERANCE CASE

In this section, we modify Algorithm 1 such that, for given $n$, $d$, and $z>1$, with some constant probability the modified algorithm constructs a $t\times n$ $(d;z)$-disjunct matrix, with $t\approx 4.28(1+O(\frac{1}{\sqrt{d\ln n}}))d^2$ $(O(1)+\frac{2.30(z-1)}{d}+\log_2 n)$ when $d\ln n\gg z-1$. This construction can be directly applied to construct nonadaptive $\lfloor\frac{z-1}{2}\rfloor$-error-tolerant pooling designs, in which $n$ denotes the number of items and $d$ denotes an upper bound on the number of positives.

For given $n$, $d$, and $z>1$, let $c_1,c_2,q,n_0,\epsilon$ and $\delta$ be as in Algorithm 1. Assign $p=\frac{1-\delta}{(1+\epsilon)d}\ (1+\frac{z-1}{(1+\frac{1}{\epsilon})d\ln(c_2n_0)})^{-1}$ and $t=\frac{\ln(c_2n_0)}{p^2\epsilon}=\frac{(1+\epsilon)^2d^2\ln(c_2n_0)}{\epsilon(1-\delta)^2}\ (1+\frac{z-1}{(1+\frac{1}{\epsilon})d\ln(c_2n_0)})^2$. For our algorithm for constructing $(d;z)$-disjunct matrices, see Algorithm 2.

---

**Algorithm 2.** Constructing $(d;z)$-disjunct matrix $M_{t\times n}$, with parameters $c_1$ and $c_2$ ($\frac{1}{c_1}+\frac{1}{c_2}<1$):

---

Algorithm 2 works in the same way as Algorithm 1, but with $p=\frac{1-\delta}{(1+\epsilon)d}\ (1+\frac{z-1}{(1+\frac{1}{\epsilon})d\ln(c_2n_0)})^{-1}$ and $t=\frac{\ln(c_2n_0)}{p^2\epsilon}$.

---

For the case where $d \ln n \gg z-1$ (thus $d \ln n \gg 1$), $\delta = \sqrt{\frac{2\epsilon \ln c_1}{(1+\epsilon)d \ln(2qc_2^2 n)}} \ll 1$, and $\delta \approx \sqrt{\frac{2\epsilon \ln c_1}{(1+\epsilon)d \ln n}}$. We can estimate $t = \frac{(1+\epsilon)^2 d^2 \ln(c_2 n_0)}{\epsilon(1-\delta)^2}\ (1 + \frac{z-1}{(1+\frac{1}{\epsilon})d \ln(c_2 n_0)})^2 \approx \frac{(1+\epsilon)^2 d^2 \ln(c_2 n_0)}{\epsilon}(1+2\delta)\ (1 + 2\frac{z-1}{(1+\frac{1}{\epsilon})d \ln(c_2 n_0)}) = \frac{(1+\epsilon)^2 d^2}{\epsilon \log_2 e}(1+2\delta)\ (\log_2(c_2 n_0) + \frac{(2\log_2 e)(z-1)}{(1+\frac{1}{\epsilon})d}) \approx c(1 + t_1 \frac{1}{\sqrt{d \ln n}})d^2\ (t_2 + \frac{c'(z-1)}{d} + \log_2 n)$, where $c$, $c'$, $t_1$ and $t_2$ are positive constants, $c = \frac{(1+\epsilon)^2}{\epsilon \log_2 e} \approx 4.28$, $c' = \frac{2\log_2 e}{1+\frac{1}{\epsilon}} \approx 2.30$, $t_1 = \sqrt{\frac{8\epsilon \ln c_1}{1+\epsilon}}$ and $t_2 = \log_2(2qc_2^2)$. Similarly, it is easy to see that when $d \ln n \gg z-1$, Algorithm 2 runs in time $O(d^2 n^2 \ln n)$ in the most straightforward manner, and the running time can be improved to expected $O(n^2 \ln n)$ by counting the column intersections along the rows.

Next we will show that for the above assignment to $p$ and $t$, $(1-\delta)\mu_1 - d(1+\epsilon)\mu_2 = z-1$. Since $\mu_1 = pt$ and $\mu_2 = p^2 t$, it follows that $(1-\delta)\mu_1 - d(1+\epsilon)\mu_2 = (1-\delta-(1+\epsilon)dp)pt = (1-\delta-(1+\epsilon)dp)\frac{\ln(c_2 n_0)}{p\epsilon} = (\frac{1-\delta}{(1+\epsilon)dp} - 1)\frac{(1+\epsilon)d \ln(c_2 n_0)}{\epsilon} = \frac{z-1}{(1+\frac{1}{\epsilon})d \ln(c_2 n_0)} \frac{(1+\epsilon)d \ln(c_2 n_0)}{\epsilon} = z-1$. Thus, by similar arguments if Algorithm 2 successfully returns a matrix $M_{t\times n}$, $M_{t\times n}$ is $(d;z)$-disjunct.

In Algorithm 2, $\mu_1 = E[w_i] = pt$, and $\mu_1 \delta^2/2 = \frac{\delta^2}{2} pt = \frac{\delta^2}{2}\frac{\ln(c_2 n_0)}{p\epsilon} > \frac{\delta^2}{2}\frac{\ln(c_2 n_0)}{\epsilon}\frac{(1+\epsilon)d}{1-\delta} > \frac{\delta^2(1+\epsilon)d \ln(c_2 n_0)}{2\epsilon} = \ln c_1$, thus $e^{-\mu_1\delta^2/2} \le \frac{1}{c_1}$. Also, $\mu_2 = E[w_{i,j}] = p^2 t = \frac{1}{\epsilon}\ln(c_2 n_0)$, thus $(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}})^{\mu_2} = (\frac{e}{(1+\epsilon)^{\frac{1+\epsilon}{\epsilon}}})^{\ln(c_2 n_0)} = (e^{-1})^{\ln(c_2 n_0)} = \frac{1}{c_2 n_0}$. Therefore, if we let $Y^*$ be the random variable denoting the number of marked columns in Step 2 of Algorithm 2, and let $m^*$ be the random variable denoting the number of edges created in Step 3 of Algorithm 2, the same results in Lemma 3.3 and 3.4 also hold here, that is $E[Y^*] \le \frac{n_0}{c_1}$ and $E[m^*] \le \frac{n_0}{2c_2}$.

Let $Z^* = Y^* + m^*$ be the random variable denotes the most number of columns that may be removed at Step 4 in Algorithm 2, similarly the probability that there are less than $n$ columns left in the matrix at Step 5 is at most $\Pr[Z^* > n_0 - n] \le q$, which indicates the following theorem.

**Theorem 4.1.** *Given $n$, $d$, and $z > 1$, Algorithm 2 with parameters $c_1$ and $c_2$ ($\frac{1}{c_1} + \frac{1}{c_2} < 1$) successfully returns a $(d;z)$-disjunct $t \times n$ matrix with probability at least $1-q$, where $t = \frac{(1+\epsilon)^2 d^2 \ln(c_2 n_0)}{\epsilon(1-\delta)^2}\ (1 + \frac{z-1}{(1+\frac{1}{\epsilon})d \ln(c_2 n_0)})^2$ and $q = \frac{1}{c_1} + \frac{1}{c_2}$. In addition, when $d \ln n \gg z-1$, $t \approx c(1 + t_1 \frac{1}{\sqrt{d \ln n}})d^2\ (t_2 + \frac{c'(z-1)}{d} + \log_2 n)$, where $c$, $c'$, $t_1$ and $t_2$ are positive constants, $c = \frac{(1+\epsilon)^2}{\epsilon \log_2 e} \approx 4.28$, $c' = \frac{2\log_2 e}{1+\frac{1}{\epsilon}} \approx 2.30$, $t_1 = \sqrt{\frac{8\epsilon \ln c_1}{1+\epsilon}}$ and $t_2 = \log_2(2qc_2^2)$.*

## 5. DISCUSSION

In this paper, we present two Las Vegas algorithms for efficient constructions of $d$-disjunct and $(d;z)$-disjunct matrices respectively. These new constructions can be directly applied to construct error-free and error-tolerant pooling designs. Because of their importance, it is interesting to investigate better constructions of disjunct matrices. E.g., with even smaller number of rows $t$ for fixed $n$, $d$ and $z$.

## APPENDICES

### A. The idea behind choosing parameters $\epsilon$ and $\delta$

We choose $\epsilon, \delta > 0$ to minimize the leading constant of $t$. In Algorithm 1, we require $(1+\epsilon)\mu_2 \le \frac{(1-\delta)\mu_1}{d}$ (i.e., $p \le \frac{1-\delta}{(1+\epsilon)d}$) to guarantee that the resulting matrix is $d$-disjunct. In addition, at Step 4 of Algorithm 1, we require $E[m] = \binom{n_0}{2}\Pr[X^{1,2} = 1] \le n_0$, which can be guaranteed by satisfying $\binom{n_0}{2}[\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}}]^{\mu_2} \le n_0$, that is, $\mu_2 \ln \frac{(1+\epsilon)^{1+\epsilon}}{e^\epsilon} \ge \ln \frac{n_0 - 1}{2}$. Since $n_0 \ge n$ we have $\mu_2 \ln \frac{(1+\epsilon)^{1+\epsilon}}{e^\epsilon} \ge O(1) + \ln n$. By plugging in $\mu_2 = p^2 t \le \frac{(1-\delta)^2}{(1+\epsilon)^2 d^2} t$ we can get $t \ge \frac{(1+\epsilon)^2}{\ln \frac{(1+\epsilon)^{1+\epsilon}}{e^\epsilon}} \frac{1}{(1-\delta)^2} d^2 (O(1) + \ln n)$. Define $f(\epsilon) = \frac{(1+\epsilon)^2}{\ln \frac{(1+\epsilon)^{1+\epsilon}}{e^\epsilon}} = \frac{(1+\epsilon)^2}{(1+\epsilon)\ln(1+\epsilon) - \epsilon}$. To minimize $f(\epsilon)$ for $\epsilon > 0$, from basic calculus $f'(\epsilon) = 0$ implies that $\ln(1+\epsilon) = \frac{2\epsilon}{1+\epsilon}$. It is easy to verify that this equation has only one positive root, which can be solved by using numerical approximation

techniques to be $\epsilon \approx 3.92$, and $f(\epsilon) = \frac{(1+\epsilon)^2}{(1+\epsilon)\ln(1+\epsilon)-\epsilon} = \frac{(1+\epsilon)^2}{\epsilon}$. Also, in order to control the leading constant of $t$, we require $\delta \ll 1$.

### *B. The idea behind choosing $n_0$ and parameters $c_1, c_2$, and $q$*

We choose $c_1$, $c_2$, and $q$ such that $c_2 = \frac{c_1}{qc_1-1}$ (i.e., $q = \frac{1}{c_1} + \frac{1}{c_2}$) to minimize $t_2$ in the approximate expression $t \approx 4.28(1 + t_1\frac{1}{\sqrt{d \ln n}})\, d^2(t_2 + \log_2 n)$, for pre-specified success probability $1-q$ and fixed $c_1$ (thus fixed $t_1$). In Algorithm 1, we construct a $t \times n$ $d$-disjunct matrix by removing some columns from a larger $t \times n_0$ ($n_0 > n$) random matrix, the algorithm succeeds if and only if after removal, the number of remained columns is at least $n$. From Lemma 3.3 and Lemma 3.4, the expectation of the maximum number of columns that may be removed is $E[Z] = E[Y] + E[m] \le (\frac{1}{c_1} + \frac{1}{2c_2})n_0$. In order to apply the Markov inequality to get the failure probability (i.e., $\Pr[Z > n_0 - n]$) upper bounded by some constant $q < 1$, we require $n_0 - n \ge \frac{1}{q}(\frac{1}{c_1} + \frac{1}{2c_2})n_0 \ge \frac{1}{q}E[Z]$, that is $1 - \frac{1}{q}(\frac{1}{c_1} + \frac{1}{2c_2}) > 0$ and $n_0 \ge \frac{1}{1-\frac{1}{q}(\frac{1}{c_1}+\frac{1}{2c_2})}n$.

For $n$ large, $\delta = \sqrt{\frac{2\epsilon \ln c_1}{(1+\epsilon)d \ln(c_2 n_0)}} \approx \sqrt{\frac{2\epsilon \ln c_1}{(1+\epsilon)d \ln n}}$. Because $\epsilon$ is fixed, we can say that $\delta$ depends only on $c_1$. For fixed $q$ and $c_1$ and thus fixed $\delta \ll 1$, $t = \frac{(1+\epsilon)^2}{\epsilon(1-\delta)^2}d^2 \ln(c_2 n_0) \ge \frac{(1+\epsilon)^2}{\epsilon(1-\delta)^2}d^2 \ln(\frac{c_2}{1-\frac{1}{q}(\frac{1}{c_1}+\frac{1}{2c_2})}n)$. To minimize $g(c_2) = \frac{c_2}{1-\frac{1}{q}(\frac{1}{c_1}+\frac{1}{2c_2})}$ for $c_2 > 0$ and $1 - \frac{1}{q}(\frac{1}{c_1} + \frac{1}{2c_2}) > 0$, we rewrite the expression as $g(c_2) = \frac{2q}{(q-\frac{1}{c_1})^2-(q-\frac{1}{c_1}-\frac{1}{c_2})^2}$. It is easy to see that we should choose $c_2$ such that $q - \frac{1}{c_1} - \frac{1}{c_2} = 0$, and choose $n_0 = \frac{1}{1-\frac{1}{q}(\frac{1}{c_1}+\frac{1}{2c_2})}n = 2qc_2 n$.

## ACKNOWLEDGMENTS

Y. Cheng was supported in part by National Natural Science Foundation of China under grant No. 60553001 and National Basic Research Program of China under grant No. 2007CB807900, 2007CB807901. D.-Z. Du was supported in part by National Science Foundation under grant No. CCF0621829.

## REFERENCES

Aigner, M. 1996. Searching with lies. *J. Combin. Theory Ser. A* 74, 43–56.

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

De Bonis, A., Gasieniec, L., and Vaccaro, U. 2005. Optimal two-stage algorithms for group testing problems. *SIAM J. Comput.* 34, 1253–1270.

Du, D.Z., and Hwang, F.K. 2006. *Pooling Designs and Nonadaptive Group Testing: Important Tools for DNA Sequencing*. World Scientific, New York.

Du, D.Z., Hwang, F.K., Wu, W., et al. 2006. New construction for transversal design. *J. Comput. Biol.* 13, 990–995.

D'yachkov, A.G., Macula, A.J., and Rykov, V.V. 2000. New constructions of superimposed codes. *IEEE Trans. Inform. Theory* 46, 284–290.

D'yachkov, A.G., and Rykov, V.V. 1982. Bounds of the length of disjunct codes. *Probl. Contr. Inform. Theory* 11, 7–13.

D'yachkov, A.G., Rykov, V.V., and Rashad A.M. 1989. Superimposed distance codes. *Probl. Contr. Inform. Theory* 18, 237–250.

Erdös, P., Frankl, P., and Füredi, Z. 1985. Families of finite sets in which no set is covered by the union of r others. *Israel J. Math.* 51, 79–89.

Fu, H.L., and Hwang F.K. 2006. A novel use of t-packings to construct d-disjunct matrices. *Discrete Appl. Math.* 154, 1759–1762.

Füredi, Z. 1996. On $r$-cover-free families. *J. Combin. Theory Ser. A* 73, 172–173.

Hwang, F.K., and Sós, V.T. 1987. Non-adaptive hypergeometric group testing. *Studia Sci. Math. Hungar.* 22, 257–263.

Kautz, W.H., and Singleton, R.C. 1964. Nonrandom binary superimposed codes. *IEEE Trans. Inform. Theory* 10, 363–377.

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

Macula, A.J. 1997. Error-correcting nonadaptive group testing with $d^e$-disjunct matrices. *Discrete Appl. Math.* 80, 217–222.

Macula, A.J. 1999. Probabilistic nonadaptive group testing in the presence of errors and DNA library screening. *Ann. Comb.* 3, 61–69.

Motwani, R., and Raghavan, P. 1995. *Randomized Algorithms*. Cambridge University Press, New York.

Muthukrishnan, S. 1994. On optimal strategies for searching in presence of errors. *Proc. 5th ACM-SIAM Symp. Discrete Algorithms (SODA' 94)* 680–689.

Ngo, H.Q., and Du, D.Z. 2002. New constructions of non-adaptive and error-tolerance pooling designs. *Discrete Math.* 243, 161–170.

Olson, M., Hood, L., Cantor, C., et al. 1989. A common language for physical mapping of the human genome. *Science* 245, 1434–1435.

Park, H., Wu, W., Liu, Z., et al. 2003. DNA screening, pooling design and simplicial complex. *J. Comb. Optim.* 7, 389–394.

Ruszinkó, M. 1994. On the upper bound of the size of the $r$-cover-free families. *J. Combin. Theory Ser. A* 66, 302–310.

Address reprint requests to:
*Dr. Yongxi Cheng*
*Department of Computer Science*
*Tsinghua University*
*Beijing 100084, China*

*E-mail:* cyx@mails.tsinghua.edu.cn