Efficiently Decodable Non-adaptive Group Testing

Piotr Indyk* Hung Q. Ngo† Atri Rudra‡

Abstract

We consider the following "efficiently decodable" non-adaptive group testing problem. There is an unknown string $x \in \{0,1\}^n$ with at most d ones in it. We are allowed to test any subset $S \subseteq [n]$ of the indices. The answer to the test tells whether $x_i = 0$ for all $i \in S$ or not. The objective is to design as few tests as possible (say, t tests) such that x can be identified as fast as possible (say, poly(t)-time). Efficiently decodable non-adaptive group testing has applications in many areas, including data stream algorithms and data forensics.

A non-adaptive group testing strategy can be represented by a $t \times n$ matrix, which is the stacking of all the characteristic vectors of the tests. It is well-known that if this matrix is d-disjunct, then any test outcome corresponds uniquely to an unknown input string. Furthermore, we know how to construct d-disjunct matrices with $t = O(d^2 \log n)$ efficiently. However, these matrices so far only allow for a "decoding" time of $O(nt)$, which can be exponentially larger than poly(t) for relatively small values of d.

This paper presents a randomness efficient construction of d-disjunct matrices with $t = O(d^2 \log n)$ that can be decoded in time $\text{poly}(d) \cdot t \log^2 t + O(t^2)$. To the best of our knowledge, this is the first result that achieves an efficient decoding time *and* matches the best known $O(d^2 \log n)$ bound on the number of tests. We also derandomize the construction, which results in a polynomial time deterministic construction of such matrices when $d = O(\log n / \log \log n)$.

A crucial building block in our construction is the notion of (d, ℓ)-*list disjunct* matrices, which represent the more general "list group testing" problem whose goal is to output less than $d + \ell$ positions in x, including all the (at most d) positions that have a one in them. List disjunct matrices turn out to be interesting objects in their own right and were also considered independently by [Cheraghchi, FCT 2009]. We present connections between list disjunct matrices, expanders, dispersers and disjunct matrices. List disjunct matrices have applications in constructing (d, ℓ)-sparsity separator structures [Ganguly, ISAAC 2008] and in constructing tolerant testers for Reed-Solomon codes in the data stream model.

1 Introduction

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 application. For example, the topic of group testing started in 1943 when Dorfman studied the problem of testing for syphilis in WWII draftees' blood samples [8]. In this case, items are blood samples, which are positive if they are infected, and a test is a pool of samples. Since then, group testing has found numerous applications (see, e.g., the book [9]). Many applications require the non-adaptive variant of group testing, in which all tests are to be performed at once: the outcome of one test cannot be used to adaptively design another test. Non-adaptive group testing (NAGT) has found applications in DNA library screening [22], multiple access control protocols [3, 31], data pattern mining [21], data forensics [15] and data streams [7], among others.

The main research focus thus far has been on designing NAGT strategies minimizing the number of tests. In some applications, however, the speed of the "decoding" procedure to identify the positive items is just as important, as we will elaborate later. In this paper, we consider the following "efficiently decodable" NAGT problem. Given integers $n > d \geq 1$, and an unknown string $x \in \{0,1\}^n$ with at most d ones in it (x is the characteristic vector of the positives), we are allowed to test any subset $S \subseteq [n]$ of the indices. The answer to a test S tells whether $x_i = 0$ for all $i \in S$. The objective is to design as few tests as possible (say t tests) such that we can identify the input string x as efficiently as possible (say, in poly(t)-time).

A NAGT algorithm can be represented as a $t \times n$ matrix M, where each row is the characteristic vector of the subset of $[n]$ to be tested. (The answers to the tests can be thought of as M being multiplied by x, where the addition is logical OR and multiplication is the logical AND.) A well known sufficient condition for such matrices to represent uniquely decodable NAGT algorithms is one of *disjunctiveness*. In particular, a matrix M is said to be d-*disjunct* if and only if the union of at most d columns cannot contain another column. Here, each column is viewed as a characteristic vector on the set of rows. It is known that $t \times n$ d-disjunct matrices can be constructed for $t = O(d^2 \log n)$ [25, 2, 9]. A lower bound of $t = \Omega(\frac{d^2}{\log d} \log n)$ has also been known

*CSAIL, MIT. Email: indyk@mit.edu. Supported in part by David and Lucille Packard Fellowship, MADALGO (Center for Massive Data Algorithmics, funded by the Danish National Research Association) and NSF grant CCF-0728645.

†Department of Computer Science and Engineering, University at Buffalo, SUNY. Email: hungngo@buffalo.edu. Supported in part by NSF grant CCF-0347565.

‡Department of Computer Science and Engineering, University at Buffalo, SUNY. Email: atri@buffalo.edu. Supported by NSF CAREER Award CCF-0844796.

for a long time [10, 11, 13].

In terms of decoding disjunct matrices, not much is known beyond the "naive" decoding algorithm: keep removing items belonging to the tests with negative outcomes. The recent survey by Chen and Hwang [5] lists various naive decoding algorithms under different group testing models (with errors, with inhibitors, and variations). The main reason for the lack of "smart" decoding algorithms is that current decoding algorithms are designed for generic disjunct matrices. Without imposing some structure into the disjunct matrices, fast decoding seems hopeless.

The naive decoding algorithm can easily be implemented in time $O(nt)$. For most traditional applications of group testing, this decoding time is fine. However, in other applications, this running time is prohibitive. This raises a natural question (see e.g., [7, 16]) of whether one can perform the decoding in time that is sub-linear (ideally, polylogarithmic) in n, while keeping the number of tests at the best known $O(d^2 \log n)$.

Our Main Result: In this paper we show that such a decoding algorithm indeed exists. Specifically, we present a randomness efficient construction of d-disjunct matrices with $t = O(d^2 \log n)$ tests that can be decoded in time $\text{poly}(d) \cdot t \log^2 t + O(t^2)$. In particular, we only need $R = O(\log t \cdot \max(\log n, d \log t))$ many random bits to construct such matrices. Further, given these R bits, any entry in the matrix can be constructed in time $\text{poly}(t)$ and space $O(\log n + \log t)$. We also derandomize the construction, which results in a $\text{poly}(n)$ time and $\text{poly}(t)$ space deterministic construction of such matrices when $d = O(\log n / \log \log n)$.

To the best of our knowledge, this is the first result that achieves an efficient decoding time *and* matches the best known $O(d^2 \log n)$ bound on the number of tests. An earlier result due to Guruswami and Indyk gives efficient decoding time but with $O(d^4 \log n)$ tests [16]. A similar result but with $O(d^2 \log^2 n)$ tests is also implicit in that paper.

1.1 Applications

In this section we outline two scenarios where efficiently decodable disjunct matrices play an important role, and how our results improve on the earlier works.

Computing Heavy Hitters. Cormode and Muthukrishnan [7] consider the following problem of determining "hot items" or "heavy-hitters." Given a sequence of m items from $[n]$, an item is considered "hot" if it occurs $> m/(d+1)$ times. Note that there can be at most d hot items. Cormode and Muthukrishnan proposed an algorithm based on NAGT that computes all the hot items as long as the input satisfies the following "small tail" property: all of the non-hot items occur at most $m/(d+1)$ times.

The algorithm works as follows: let M be a d-disjunct $t \times n$ matrix. For each test $i \in [t]$, maintain a counter c_i. When an item $j \in [n]$ arrives (leaves respectively), increment (decrement respectively) all the counters c_i such that $M_{ij} = 1$ (i.e. all counters c_i for which test i contains item j). The algorithm also maintains the total number of items m seen so far. At any point in time, the hot items can be computed as follows. Think of the test i corresponding to counter c_i as having a positive outcome if and only if $c_i > m/(d+1)$. Due to the small tail property, a test's outcome is positive if and only if it contains a hot item. Thus, computing the hot items reduces to decoding the result vector.

When Cormode and Muthukrishnan published their result, the only decoder known for d-disjunct matrices was the $O(nt)$-time naive decoder mentioned earlier. This meant that their algorithm above could not be efficiently implemented. The authors then provided alternate algorithms, inspired by the group testing idea above and left the task of designing an efficiently decodable group testing matrix as an open problem. Our main result answers this open question. This application also requires that the matrix M be *strongly explicit*, i.e. any entry in M can be computed in time (and hence, space) $\text{poly}(t)$. Our result satisfies this requirement as well.

We would like to point out that the solution to the hot items problem using our result is not as good as the best known results for that problem. For example, the paper [7] gives a solution which has a lower space complexity than what one can achieve with efficiently decodable NAGT. Nevertheless, the above application to finding heavy hitters is illustrative of many other applications of NAGT to data stream algorithms, and we expect further results along these lines.

Digital forensics. Data forensics refers to the following problem in data structures and security. Assume that one needs to store a data structure on a semi-trusted place. An adversary can change up to d values (out of a total of n values) in the data structure (but cannot change the "layout" of the data structure). The goal is to "store" extra information into the layout of the data structure so that if up to d values in the data structure are changed then an auditor can quickly pinpoint the locations where data was changed. Goodrich, Atallah and Tamassia [15] introduced the problem and used group testing to solve this problem. In particular, using a randomness efficient construction, they present data forensics schemes on balanced binary trees, skip lists and arrays (and linked lists)

that can handle $O\left(\sqrt[3]{n/\log^2 n}\right)$, $O\left(\sqrt[3]{n/\log^2 n}\right)$ and $O\left(\sqrt[4]{n/\log n}\right)$ many changes respectively. Using our randomness efficient construction, we can improve these bounds to $O\left(\sqrt{n/\log n}\right)$, $O\left(\sqrt{n/\log n}\right)$ and $O\left(\sqrt[3]{n}\right)$ many changes respectively. These latter bounds are the best possible with the techniques of [15]. (For more details see Appendix A.)

1.2 Techniques

Connections to coding theory. Our construction involves concatenated codes. A concatenated binary code has an *outer* code $C_{\text{out}} : [q]^{k_1} \to [q]^{n_1}$ over a large alphabet $q = 2^{k_2}$ and a binary *inner* code $C_{\text{in}} : \{0,1\}^{k_2} \to \{0,1\}^{n_2}$. The encoding of a message in $(\{0,1\}^{k_2})^{k_1}$ is natural. First, it is encoded with C_{out} and then C_{in} is applied to each of the outer codeword symbols. The concatenated code is denoted by $C_{\text{out}} \circ C_{\text{in}}$. (In fact, one can— and we will— use different inner codes C^i_{in} for every $1 \leq i \leq n_1$.)

Concatenated codes have been used to construct d-disjunct matrices at least since the seminal work of Kautz and Singleton [20]. In particular, they picked C_{out} to be a maximum distance separable code of rate $\frac{1}{d+1}$ (e.g., Reed-Solomon code with $n_1 = q$) and the inner code to be the identity code I_q that maps an element $i \in [q]$ to the ith unit vector in $\{0,1\}^q$. The disjunct matrix M corresponding to $C_{\text{out}} \circ C_{\text{in}}$ is obtained by simply putting all the $n = q^{k_1}$ codewords as columns of the matrix. The resulting matrix M is d-disjunct and has $t = O(d^2 \log^2 n)$ many rows. Recently, Porat and Rothschild presented a deterministic polynomial (in fact, $O(nt)$) time construction of d-disjunct matrices with $O(d^2 \log n)$ rows [25]. In their construction, C_{out} is a code over an alphabet size $\Theta(d)$ that lies on the Gilbert-Varshamov bound. Their inner code is also the identity code as used by Kautz and Singleton. Note that since all these constructions result in d-disjunct matrices, they can be decoded in $O(nt)$ time.

Next, we outline how if C_{out} is efficiently *list recoverable* (and C_{in} is the identity code), then M can be decoded in poly(t) time. In particular, consider the following natural decoding procedure for an outcome vector $r \in \{0,1\}^t$. First, think of $r = (r_1, \ldots, r_{n_1})$ as a vector in $(\{0,1\}^{n_2})^{n_1}$, where each symbol in $\{0,1\}^{n_2}$ lies naturally in one of the n_1 positions in an outer codeword. Now consider the following algorithm. For each $i \in [n_1]$, let $L_i \subseteq [q]$ be the set of positions where r_i has a one. Since there are at most d positives, $|L_i| \leq d$, for every $i \in [n_1]$. Furthermore, if the jth item is positive then the jth codeword $(c_1, \ldots, c_{n_1}) \in C_{\text{out}}$ satisfies the following property: for every $1 \leq i \leq n_1$, $c_i \in L_i$. In the Kautz-Singleton construction, it can be checked that the construction of the L_i can be done in poly(t) time. Thus, we would be in business if we can solve the following problem in poly(n_1) time: Given $L_i \subseteq [q]$ for every $1 \leq i \leq n_1$ with $|L_i| \leq d$, output all codewords $(c_1, \ldots, c_{n_1}) \in C_{\text{out}}$ such that $c_i \in L_i$ for every $i \in [n_1]$. This is *precisely* the ("error-free") list recovery problem. It is known that for Reed-Solomon codes, this problem can be solved in poly(n_1) time (cf. [26, Chap. 6]). This observation was made by Guruswami-Indyk [16]. They also presented an efficiently decodable (in fact $O(t)$ time decoding) d disjunct matrix with $t = O(d^4 \log n)$ using an expander based code as C_{out}. However, neither of the constructions in [16] matches the best known bound of $t = O(d^2 \log n)$.

Overview of our construction. Our randomized construction is based on an ensemble of codes that was considered by Thommesen to construct randomized binary codes that lie on the Gilbert-Varshamov bound [30]. C_{out} is the Reed-Solomon code but the inner codes are chosen to be *independent* random binary codes. Unlike [30], where the inner codes are linear, in our case the (non-linear) inner codes are picked as follows: every codeword is chosen to be a random vector in $\{0,1\}^{n_2}$, where each entry is chosen to be 1 with probability $\Theta(1/d)$. We show that this concatenated code with high probability gives rise to a d-disjunct matrix.

The use of random, independently generated codes is crucial for our purpose. This is because the correctness of the aforementioned decoding algorithms crucially uses the fact that C_{in} is d-disjunct. (In fact, for the identity code, it is n-disjunct.) However, this idea hits a barrier. In particular, as mentioned earlier, d-disjunct matrices need to have $\Omega\left(\frac{d^2}{\log d} \log n\right)$ many rows. This implies that if we are using a d-disjunct matrix as an inner code and a $O(1/d)$-rate outer Reed-Solomon code then the resulting matrix will have $\Omega\left(\frac{d^3}{\log d} \log n\right)$ rows. Thus, this approach seems to fall short of obtaining the best known bound of $O(d^2 \log n)$.

However, consider the following idea that is motivated by list decoding algorithms for binary concatenated codes (cf. [17]). Let $M_{C_{\text{in}}}$ be the $n_2 \times q$ matrix obtained by arranging the codewords in C_{in} as columns. Think of $M_{C_{\text{in}}}$ as representing a NAGT strategy for at most d unknown positives. Suppose $M_{C_{\text{in}}}$ has the following weaker property than d-disjunctiveness. Let $y \in \{0,1\}^q$ be a vector with at most d ones in it, indicating the positives, and let $r \in \{0,1\}^{n_2}$ be the test outcomes when applying $M_{C_{\text{in}}}$ to y. Suppose it is the case that any $z \in \{0,1\}^q$ which results in the same test outcome r as y has to satisfy (i) $y_i = 1$ implies $z_i = 1$

and (ii) $|z| \leq 2d$, where $|z|$ denote the number of 1's in a binary vector z. (Note that in the d-disjunct case, we need the stronger property that $z = y$.) Then, if we use such an inner code instead of a d-disjunct matrix, in the two step algorithm mentioned a couple of paragraphs above we will have $|L_i| \leq 2d$ (instead of $|L_i| \leq d$). It turns out that, with this weaker inner codes, one can still perform list recovery on such inputs for Reed-Solomon codes. The only catch is that the two-step algorithm can return up to $O(d^2)$ items including all the (at most d) positives. We can thus run the naive decoding algorithm (which will now take $O(d^2t)$ time) to recover x exactly. Further, this latter step can be done without making another pass over the input.

Perhaps most importantly, the relaxed notion of disjunct matrices mentioned above can be shown to exist with $O(d \log n)$ rows, which paves the way for the final matrix to have $O(d^2 \log n)$ rows as desired. It turns out that these new objects are interesting in their own right and next, we discuss them in more detail.

List Disjunct Matrices. We define (d, ℓ)-list disjunct matrices as follows. A $t \times n$ matrix M is called (d, ℓ)-*list disjunct* if the following holds: for any disjoint subsets $S \subseteq [n]$ and $T \subseteq [n]$ such that $|S| \leq d$ and $|T| \geq \ell$, there exists a row in which there is at least one 1 among the columns in T while all the columns in S have a 0. It can be shown that given an outcome vector $r \in \{0,1\}^t$ obtained by using M on an input $x \in \{0,1\}^n$ with $|x| \leq d$, running the naive decoder returns a vector $y \in \{0,1\}^n$ such that it contains x and $|y| \leq |x| + \ell - 1$. (Hence, the name *list* disjunct matrix.) In the proof of our main result, we use the fact that the random inner codes with high probability are (d, d)-list disjunct.

These objects were independently considered by Cheraghchi [6]. (In fact, he considered more general versions that can handle errors in the test results.) We study these objects in some detail and obtain the following results. First, we show that (d, ℓ)-list disjunct matrices contain d-disjunct column sub-matrices, which allows us to prove a lower bound of $\Omega(d/\log d \log(n/\ell))$ on the number of rows for such objects as long as $d \leq O(\log n/\log\log n)$. (This gives a better lower bound for large ℓ than the bound of $d\log(n/d) - d - \ell$ in [6].) We also show that lossless expanders and dispersers with appropriate parameters imply list disjunct matrices. Using known constructions of expanders and dispersers we construct (d, d)-list disjunct matrices with near optimal $(d \log n)^{1+o(1)}$ number of rows and (d, ℓ)-list disjunct matrices with $O(d \log n)$ rows when ℓ and d are polynomially large in n.

We believe that list disjunct matrices are natural objects and merit study on their own right. To substantiate our belief, we point out three application of list disjunct matrices. First, as pointed out by Cheraghchi [6], (d, d)-list disjunct matrices can be used to design optimal *two-stage group testing* algorithms. In the two-stage group testing problem, one is allowed to make tests into two stages (the tests in the second stage can depend on the answers obtained in the first stage). The algorithm works as follows: in the first round, one uses the (d, d)-list disjunct matrix to obtain a subset $S \subseteq [q]$ of size at most $2d$ that contains all of the defective positions. In the second stage, one can probe each of the columns in S, to compute the defective locations. Note that this algorithm takes $O(d \log n)$ many tests, which is optimal as there are $(n/d)^{O(d)}$ many possible answers. Second, we show that (d, ℓ)-list disjunct matrices immediately imply $(d, d+\ell)$-sparsity separator structures. The latter were used by Ganguly to design deterministic data stream algorithm for the d-sparsity problem [14]. In fact, Ganguly's construction of $(d, 2d)$-sparsity separator uses lossless expander in pretty much the same way we do. (See Section 6 for more details.) Finally, we point out that the recent work of Rudra and Uurtamo [27] uses (d, d)-list disjunct matrices to design data stream algorithms for the following "tolerant testing" of Reed-Solomon codes. Given the received word, the algorithm needs to decide if it is within Hamming distance d of some codeword or is at distance at least $2d$ from every codeword. Their algorithm also works with d-disjunct matrices but then one only gets a sub-linear space data stream algorithm for $d = o(\sqrt{n})$, whereas using list disjunct matrices allows them to design a sub-linear space data stream algorithm for $d = o(n)$. (See Section 6 for more details.)

1.3 Paper Organization

We begin with some preliminaries in Section 2. In Section 3, we define list disjunct matrices and present upper and lower bounds. Section 4 formally presents the connection between list recoverable codes and efficiently decodable disjunct matrices. We present our main construction of efficiently decodable disjunct matrices equaling the best known number of tests in Section 5. Finally we present connections between list disjunct matrices and pseudo-random objects and their applications in Section 6.

2 Preliminaries

For an integer $\ell \geq 1$, let $[\ell]$ to denote the set $\{1, \ldots, \ell\}$. For any $t \times N$ matrix M, we will use M_j to refer to its j'th column and M_{ij} to refer to the i'th entry in M_j. Let $\mathbb{F}_q$ denote the finite field of q elements.

The basic problem of non-adaptive combinatorial group testing can be described as follows. Given a population of N "items" which contains at most d "positive" items, we want to identify the positives as

quickly as possible using t simultaneous "tests." Each test is a subset of items, which returns "positive" if there's at least one positive item in the subset. We want to "decode" uniquely the set of positives given the results of the t simultaneous tests.

It is natural to model the tests as a $t \times N$ binary matrix M where each row represents a test and each column is an item. Set $M_{ij} = 1$ if and only if item j belongs to test i. The test outcome vector is a t-dimensional binary vector $\mathbf{r}$, where a 1 indicates a positive test outcome for the corresponding test.

For the decoding to be unique, it is sufficient for the test matrix M to satisfy a property called *d-disjunctiveness*. We say a $t \times N$ binary matrix M is *d-disjunct* if, for every column M_j in M and every subset of d other columns $M_{j_1}, \ldots, M_{j_d}$ (i.e. $j \notin \{j_1, \ldots, j_d\}$), $M_j \not\subseteq M_{j_1} \cup \cdots \cup M_{j_d}$. Here, we interpret the columns as (characteristic vectors of) subsets of $[t]$. The following lemma gives a sufficient condition for disjunct matrices, and can be shown by a simple counting argument [9].

LEMMA 2.1. *Let M be a binary matrix such that w is the minimum Hamming weight of columns in M and let λ be the size of the largest intersection between any two columns in M. Then M is d-disjunct as long as $\lambda d + 1 \leq w$. In particular, M is $\left\lfloor \frac{w-1}{\lambda} \right\rfloor$-disjunct.*

When M is d-disjunct, it is well-known that the following *naive decoder* works (cf. [9]). Let the outcome vector be $\mathbf{r}$. For every $i \in [t]$ and $j \in [N]$ such that $r_i = 0$ and $M_{ij} = 1$, "eliminate" item j. Once all such indices from $[N]$ are eliminated, we will be left precisely with the set of positives. The naive decoder takes $O(tN)$-time.

A *$(t, k, d)_q$-code* C is a subset $C \subseteq [q]^t$ of size q^k such that any two codewords differ in at least d positions. The parameters t, k, d and q are known as the *block length*, *dimension*, *distance* and *alphabet size* of C. Sometimes we will drop the distance parameter and refer to C as a $(t, k)_q$ code.

Given a $(t, k, d)_q$ code C, let M_C denote the $t \times q^k$ matrix whose columns are the codewords in C. We shall construct disjunct matrices by specifying some binary code C and show that M_C is disjunct. The codes C that we construct will mostly be obtained by "concatenating" an outer code – based on Reed-Solomon codes – with a suitably chosen inner code.

Given an $(n_1, k_1)_{2^{k_2}}$ code C_{out} and an $(n_2, k_2)_2$ code C_{in}, their *concatenation*, denoted by $C_{\text{out}} \circ C_{\text{in}}$ is defined as follows. Consider a message $\mathbf{m} \in \left(\{0,1\}^{k_2}\right)^{k_1}$. Let $C_{\text{out}}(\mathbf{m}) = (x_1, \ldots, x_{n_1})$. Then, $C_{\text{out}} \circ C_{\text{in}}(\mathbf{m}) = \left(C_{\text{in}}(x_1), \ldots, C_{\text{in}}(x_{n_1})\right)$. $C_{\text{out}} \circ C_{\text{in}}$ is an $(n_2 n_1, k_1 k_2)_2$-code.

In general, we can concatenate C_{out} with n_1 different inner codes $C^1_{\text{in}}, \ldots, C^{n_1}_{\text{in}}$, one per position. Denote this general concatenated code as $C_{\text{out}} \circ (C^1_{\text{in}}, \ldots, C^{n_1}_{\text{in}})$. It is defined in the natural way. Given any message $\mathbf{m} \in \left(\{0,1\}^{k_2}\right)^{k_1}$, let $C_{\text{out}}(\mathbf{m}) = (x_1, \ldots, x_{n_1})$. Then,

$$C_{\text{out}} \circ (C^1_{\text{in}}, \ldots, C^{n_1}_{\text{in}})(\mathbf{m}) = \left(C^1_{\text{in}}(x_1), \ldots, C^{n_1}_{\text{in}}(x_{n_1})\right).$$

Let $\ell, L \geq 1$ be integers and let $0 \leq \alpha \leq 1$. A q-ary code C of block length n is called *(α, ℓ, L)-list recoverable* if, for every sequence of subsets $S_1, \ldots, S_n$ such that $|S_i| \leq \ell, \forall i \in [n]$, there are at most L codewords $\mathbf{c} = (c_1, \ldots, c_n)$ for which $c_i \in S_i$ for at least αn positions i. A $(1, \ell, L)$-list recoverable code will be henceforth referred to as *(ℓ, L)-zero error list recoverable*. We will need the following powerful result due to Parvaresh and Vardy[1]:

THEOREM 2.1. ([24]) *For any given integer $s \geq 1$, prime power r, power q of r, and every pair of integers $1 < k \leq n \leq q$, there exists an explicit $\mathbb{F}_r$-linear map $E : \mathbb{F}_q^k \to \mathbb{F}_{q^s}^n$ satisfying the following:*

1. *The image $C \subseteq \mathbb{F}_{q^s}^n$ of E is a code of minimum distance at least $n - k + 1$.*
2. *Provided*

$$\alpha > (s+1)(k/n)^{s/(s+1)} \ell^{1/(s+1)}, \tag{2.1}$$

C is an $(\alpha, \ell, O((rs)^s n\ell/k))$-list recoverable code. Further, a list recovery algorithm exists that runs in $\text{poly}((rs)^s, q, \ell)$-time. For $s = 1$, the $(s+1)$ factor in the right-hand-side of (2.1) can be removed.

In the above, the s'th "order" Parvaresh-Vardy code will be referred to as the PV^s code. PV^1 is the well-known Reed-Solomon codes and will be referred to as the RS code.

The simplest possible binary code is probably the "identity code" $I_q : [q] \to \{0,1\}^q$ defined as follows. For any $i \in [q]$, $I_q(i)$ is the vector in $\{0,1\}^q$ that has a 1 in the i'th position and zero everywhere else. It is easy to see that

LEMMA 2.2. *M_{I_q} is q-disjunct.*

The following result was shown in [25] by concatenating a q-ary code on the GV bound (with $q = \Theta(d)$ and relative distance $1 - 1/d$) with the identity code I_q.

THEOREM 2.2. ([25]) *Given n and $d \geq 1$. There exists a $t \times n$ d-disjunct matrix M_{PR} such that $t = O(d^2 \log n)$. Further, the matrix can be constructed in time (and space) $O(tn)$, and all the columns of M_{PR} have the same Hamming weight.*

[1] This statement of the theorem appears in [18].

3 List Disjunct Matrices and Basic Bounds

We now define a generalization of group testing that is inspired by list decoding of codes and will be useful in our final construction. Let $\ell, d \geq 1$ be integers. We call a $t \times n$ boolean matrix M (d,ℓ)-*list disjunct* if for any two *disjoint* subsets S and T of $[n]$ where $|S| \leq d, |T| \geq \ell$, we have $\bigcup_{j \in T} M_j \not\subseteq \bigcup_{j \in S} M_j$. Note that a d-disjunct matrix is a $(d,1)$-list disjunct matrix and vice versa. Without loss of generality, we assume that $\ell \leq n-1$. When $d + \ell > n$, we can replace d by $d = n - \ell$. If $d + \ell \leq n$, then it is sufficient to replace the condition in the definition by $|S| = d, |T| = \ell$. Thus, we will assume $d + \ell \leq n$ henceforth, and only consider $|S| = d, |T| = \ell$ as the condition in the definition of (d,ℓ)-list disjunct matrices.

We next prove upper and lower bounds for the optimal "rate" of list disjunct matrices. To prove an upper bound for the optimal number of rows of a (d,ℓ)-list disjunct matrix, and to devise a $n^{O(d)}$ time construction for it, the idea is to cast the problem of constructing a (d,ℓ)-list disjunct matrix as a special case of the so-called k-*restriction problem* defined and solved in Alon, Moshkovitz, and Safra [2]. (The proof appears in Appendix B.)

LEMMA 3.1. *Given positive integers $n \geq d+1$, let $\epsilon_0 = \left(\frac{\ell}{d+\ell}\right)\left(\frac{d}{d+\ell}\right)^{d/\ell}$. Then, a $t \times n$ (d,ℓ)-list disjunct matrix can be constructed so that $t = \left\lceil \frac{2(d+\ell)\ln n + \ln\binom{d+\ell}{d}}{\epsilon_0} \right\rceil$ in time poly$\left(\binom{d+\ell}{d}, n^{d+\ell}, 2^{d+\ell}, \frac{1}{\epsilon}\right)$. In particular, if $\ell = \Theta(d)$, then a $t \times n$ (d,ℓ)-list disjunct matrix can be constructed in time $n^{O(d)}$ where $t = O(d \log n)$.*

Later on, we will see constructions of $(d, \Theta(d))$-list disjunct matrices that can be constructed more time-efficiently than Lemma 3.1. However, those constructions need more tests (up to logarithmic factors).

Next we show that any (d,ℓ)-list disjunct matrix M contains a large enough d-disjunct sub-matrix. The proof follows by building a hypergraph from M and then observing that an independent set in the hypergraph corresponds to a d-disjunct matrix. Then a result due to Caro and Tuza [4] completes the proof.

LEMMA 3.2. *Let $d, \ell \geq 1$ and $t, n \geq 1$ be integers. Let M be a $t \times n$ (d,ℓ)-list disjunct matrix. Then there is a $t \times m$ sub-matrix of M that is d-disjunct with $m \geq \Omega\left(\left(\frac{n}{d\ell}\right)^{\frac{1}{d}}\right)$. In particular, any $t \times n$ matrix that is (d,ℓ)-list disjunct (with $d \leq O(\log n / \log\log n)$ and $1 \leq \ell \leq n^{1-\gamma}$ for some constant $\gamma > 0$) needs to satisfy $t = \Omega\left(\frac{d}{\log d} \cdot \log n\right)$.*

Proof. Consider the following hypergraph H: each of the n columns in M forms a vertex and there is a $(d+1)$-hyperedge among $d+1$ vertices in $S \subset [n]$ if there exists a $j \in S$ such that the jth column of M is contained in the union of the columns in $S \setminus \{j\}$. Note that an independent set in H of size m corresponds to a column sub-matrix of M that is d-disjunct. For notational convenience define $\deg(i)$ to be the degree of i in H.

Thus, we need to show that there exists a large enough independent set in H, for which we use a result of Caro and Tuza [4]. In particular, H has an independent set of size at least (cf. [28, Pg. 2]):

$$\sum_{i \in [n]} \frac{1}{f(i)}, \text{ where } f(i) = \Theta\left((1+\deg(i))^{\frac{1}{d}}\right). \tag{3.2}$$

We claim that for at least $n/2$ vertices i,

$$\deg(i) \leq 2(d+1)\binom{n}{d-1}(\ell-1). \tag{3.3}$$

Since $\binom{n}{d-1} \leq n^{d-1}$, (3.3) and (3.2) imply that H has an independent set of size at least $\Omega((n/(d\ell))^{1/d})$, as desired.

Next, we prove (3.3). First, we claim that the number of edges in H is upper bounded by $n\binom{n}{d-1}(\ell-1)$. This implies that the average degree of H is at most $(d+1)\binom{n}{d-1}(\ell-1)$. An averaging argument proves (3.3). Now we upper bound the number of edges in H. To this end, "assign" a vertex i to all its incident edges $S \subseteq [n]$ such that there exists a $j \neq i$ where the jth column is contained in the union of the columns in $S \setminus \{j\}$. For every $T \subseteq [n]$ with $|T| = d$ with $i \in T$, there exists at most $\ell - 1$ many j such that $T \cup \{j\}$ is an edge. (This follows from the fact that M is (d,ℓ)-list disjunct.) Thus, any vertex i is assigned to at most $\binom{n}{d-1}(\ell-1)$ edges. Since every edge is assigned at least one vertex, the total number of edges is bounded by $n\binom{n}{d-1}(\ell-1)$, as desired.

The lower bound on the number of rows in a (d,ℓ)-list disjunct matrix then follows from the lower bound on the number of rows in a d-disjunct matrix. The bound on $d \leq O(\log n / \log\log n)$ is needed to make sure that the lower bound on list disjunct matrix holds. (The latter for a d-disjunct matrix with N columns needs $d \leq \tilde{O}(\sqrt{N})$). □

Cheraghchi proves a lower bound of $d\log(n/d) - \ell$ for (d,ℓ)-list disjunct matrices [6]. This is better than the bound in Lemma 3.2 for moderately large ℓ (e.g. when $\ell = \Theta(d)$).

4 From List Recoverable Codes to Efficiently Decodable Group Tests

As was mentioned in the introduction section, concatenating an efficiently list recoverable outer code with list disjunct inner codes gives rise to efficiently decodable disjunct matrices. The decoding algorithm mimics the standard list decoding algorithm for concatenated codes.

THEOREM 4.1. *Let $d, \ell, L \geq 1$ be integers and $0 < \alpha \leq 1$ be a real number. Let C_{out} be an $(n_1, k_1)_{2^{k_2}}$ code which can be $(\alpha, d+\ell-1, L)$-list recovered in time $T_1(n_1, d, \ell, L, k_1, k_2)$. Let $C_{\text{in}}^1, \ldots, C_{\text{in}}^{n_1}$ be $(n_2, k_2)_2$ codes such that for at least αn_1 values of $1 \leq i \leq n_1$, $M_{C_{\text{in}}^i}$ is (d, ℓ)-list disjunct and can be ("list") decoded in time $T_2(n_2, d, \ell, k_2)$. Suppose the matrix $M \stackrel{def}{=} M_{C_{\text{out}} \circ (C_{\text{in}}^1, \ldots, C_{\text{in}}^{n_1})}$ is d-disjunct. Note that M is a $t \times N$ matrix where $t = n_1 n_2$ and $N = 2^{k_1 k_2}$. Further, suppose that any arbitrary position in any codeword in C_{out} and C_{in}^i can be computed in space $S_1(n_1, d, \ell, L, k_1, k_2)$ and $S_2(n_2, d, \ell, k_2)$, respectively. Then,*

(a) given any outcome vector produced by at most d positives, the positive positions can be recovered in time $n_1 T_2(n_2, d, \ell, k_2) + T_1(n_1, d, \ell, L, k_1, k_2) + O(Lt)$; and

(b) any entry in M can be computed in $O(\log t + \log N) + O(\max\{S_1(n_1, d, \ell, L, k_1, k_2), S_2(n_2, d, \ell, k_2)\})$ space.

Proof. The decoding algorithm for M is the natural one. Let $E \subseteq [N]$ with $|E| \leq d$ be the set of positive items. Further let $\mathbf{r} = (r_1, \ldots, r_{n_1}) \in (\{0,1\}^{n_2})^{n_1}$ be the outcome vector.

In the first step of the algorithm, for each $i \in [n_1]$, run the list decoding algorithm for $M_{C_{\text{in}}^i}$ on outcome (sub-) vector r_i. For each i where $M_{C_{\text{in}}^i}$ is (d, ℓ)-list disjunct we will obtain a set $S_i \subseteq \{0,1\}^{n_2}$ with $|S_i| \leq d + \ell - 1$. There are at least αn_1 such i. For the indices i where $M_{C_{\text{in}}^i}$ is not (d, ℓ)-list disjunct, let S_i be $d + \ell - 1$ arbitrary members of $\{0,1\}^{n_2}$. This step takes time $n_1 T_2(n_2, d, \ell, k_2)$.

In the second step, we run the list recovery algorithm for C_{out} on $S_1, \ldots, S_{n_1}$. This step will output a subset $T \subseteq [N]$ such that $|T| \leq L$. This step takes time $T_1(n_1, d, \ell, L, k_1, k_2)$. Now if we can ensure that $E \subseteq T$, then we can run the naive algorithm on M restricted to the columns in T to recover E in time $O(Lt)$. Thus, the total running time of the algorithm is as claimed.

To complete the argument above, we need to show that $E \subseteq T$. Without loss of generality, assume $E = \{\mathbf{m}^1, \ldots, \mathbf{m}^d\} \subseteq (\{0,1\}^{n_2})^{n_1}$. Denote $\mathbf{m}^j = (m_1^j, \ldots, m_{n_1}^j)$ where $m_i^j \in \{0,1\}^{n_2}$. By definition of $\mathbf{r}$, we have $r_i = m_i^1 \cup \cdots \cup m_i^d$. Consequently, if $M_{C_{\text{in}}^i}$ is (d, ℓ)-list disjunct, then $\{m_i^1, \ldots, m_i^d\} \subseteq S_i$. As at least αn_1 of the $M_{C_{\text{in}}^i}$ are (d, ℓ)-list disjunct, E will be included in the set T of at most L codewords returned by the list recovery algorithm for C_{out}, which is an $(\alpha, d + \ell - 1, L)$-list recoverable code.

Finally, we argue the space complexity. Given $i = (i_1, i_2) \in [n_1] \times [n_2]$ and $j \in (\{0,1\}^{k_2})^{k_1}$, we can compute M_{ij} in the following natural way. First compute $(C_{\text{out}}(j))_{i_1}$ in space $S_1(n_1, d, \ell, L, k_1, k_2)$ and then compute $\left(C_{\text{in}}^{i_1}\left((C_{\text{out}}(j))_{i_1}\right)\right)_{i_2}$ in space $S_2(n_2, d, \ell, k_2)$. We might need the extra $O(\log t + \log N)$ space to store the indices i and j. □

By instantiating the outer and inner codes in Theorem 4.1 with different specific codes, we obtain several constructions balancing some tradeoffs: there exists disjunct matrices that can be decoded in $\text{poly}(t)$ time with $O(d^2 \log^2 N)$, $O(d^3 \log N(\log d + \log\log N))$ and $O(d^3 \log N)$ many tests. Further, any entry in these matrices can be constructed in space $\text{poly}(\log t, \log N)$, $\text{poly}(\log t, \log N)$ and $\text{poly}(t, \log N)$ respectively. Last but not least, we can also design efficiently decodable list group testing by using the PV^s code with $s = 1/\epsilon$ as the outer code. All the proofs are in Appendix C.

LEMMA 4.1. *Let $N > d, s \geq 1$ be integers. Then there exists a constant $c \geq 1$ and a $t \times N$ matrix M with $t = O\left(s^{cs} d^{1+1/s} \log^{1+s} N\right)$ that is $\left(d, s^{cs} d^{1+1/s}\right)$-list disjunct and can be list decoded in $\text{poly}(t)$ time. Further, any entry of the matrix can be computed in $\text{poly}(\log t, \log N, s)$ space.*

The result above achieves better bounds than the efficient decodable constructions of list disjunct matrices in [6] (though the results in [6] can also handle errors).

5 On Constructing Efficiently Decodable Disjunct Matrices

This section contains the main result of the paper. We first show probabilistically that, given d and N, there exists a $t \times N$ efficiently decodable d-disjunct matrix with $t = O(d^2 \log N)$, which matches the best known constructions of disjunct matrices (whether probabilistic or deterministic). We then show that our probabilistic construction can be derandomized with low space or low time complexity.

5.1 Probabilistic Existence of Efficiently Decodable Disjunct Matrices

THEOREM 5.1. *Let d, k_1, k_2 be any given positive integers such that $10dk_1 \leq 2^{k_2}$. Define $n_1 = 10dk_1$,*

$n_2 = 480dk_2$, $t = n_1 n_2$, *and* $N = 2^{k_1 k_2}$. *Note that* $n_1 \leq 2^{k_2}$ *and* $t = O(d^2 \log N)$.

Let C_{out} *be any* $\left(n_1, k_1 \stackrel{def}{=} \frac{n_1}{10d}, n_1\left(1 - \frac{1}{10d}\right)\right)_{2^{k_2}}$*-code. Then, there exist inner codes* $C^1_{\text{in}}, \ldots, C^{n_1}_{\text{in}}$*, each of which is an* $(n_2, k_2)_2$ *code such that the following hold:*

(a) Let $C^* = C_{\text{out}} \circ (C^1_{\text{in}}, \ldots, C^{n_1}_{\text{in}})$*, then* M_{C^*} *is* $t \times N$ *matrix that is* d*-disjunct.*

(b) For every $i \in [n_1]$, $M_{C^i_{\text{in}}}$ *is a* (d, d)*-list disjunct matrix.*

Proof. We will show the existence of the required inner codes by the probabilistic method. In particular, we will refer to each inner code C^i_{in} by its corresponding matrix $M_{C^i_{\text{in}}}$. For every $1 \leq i \leq n_1$, pick $M_{C^i_{\text{in}}}$ to be a random $n_2 \times 2^{k_2}$ binary matrix where each entry is chosen independently at random to be 1 with probability $\frac{1}{10d}$. We stress that the random choices for each of the inner codes are independent of each other.

We first bound the probability that condition (a) holds, i.e. M_{C^*} is d-disjunct. For notational convenience define $q = 2^{k_2}$ and $M^* = M_{C^*}$.

By Lemma 2.1, M^* is d-disjunct if the following two events hold:

(i) Every column has Hamming weight at least $\frac{t}{20d} + 1$.

(ii) Every two distinct columns agree in at most $\frac{t}{20d^2}$ positions.

Consider event (i). Any arbitrary column M^*_j of M^* is simply a random vector in $\{0,1\}^t$ where every bit is 1 independently with probability $\frac{1}{10d}$. Thus, by Chernoff bound the probability that M^*_j has Hamming weight $\leq t/(20d)$ is at most $e^{-t/(120d)}$. Taking a union bound over the N choices of j, we conclude that event (i) does not hold with probability at most $Ne^{-\frac{t}{120d}} \leq N^{-39d}$, where the inequality follows from the fact that $t = 4800d^2 \log_2 N$.

We now turn to event (ii). Pick two distinct columns $i \neq j \in [N]$. By the fact that C_{out} has relative distance at least $1 - \frac{1}{10d}$, the ith and jth codeword in C_{out} disagree in some positions $S \subseteq [n_1]$ with $|S| = (1 - \frac{1}{10d})n_1$. Let $A_1 \in \{0,1\}^{t(1-\frac{1}{10d})}$ and $A_2 \in \{0,1\}^{\frac{t}{10d}}$ denote M^*_i projected down to S and $[N] \setminus S$. Let B_1 and B_2 denote the analogous projections for M^*_j. Note that we need to bound the random variable $|A_1 \cap B_1| + |A_2 \cap B_2|$.

We start with $|A_1 \cap B_1|$. Note that by the definition of A_1, B_1 and the inner codes, $A_1 \cap B_1 \in \{0,1\}^{t(1-\frac{1}{10d})}$, where each bit is 1 independently with probability $\frac{1}{100d^2}$. Thus, by Chernoff bound,

$$\begin{aligned}
&\Pr\left[|A_1 \cap B_1| \geq 2\frac{t}{100d^2}\right] \\
&\leq \Pr\left[|A_1 \cap B_1| \geq \frac{2}{100d^2}t(1 - \frac{1}{10d})\right] \\
&\leq e^{-\frac{1}{300d^2}t(1-\frac{1}{10d})} \leq e^{-t/(600d^2)}.
\end{aligned}$$

Next, we turn to $|A_2 \cap B_2|$. Note that $|A_2 \cap B_2| \leq |A_2|$ and that A_2 is a random vector in $\{0,1\}^{\frac{t}{10d}}$ where each bit is 1 independently with probability $\frac{1}{10d}$. Again, by Chernoff bound the probability that $|A_2| \geq \frac{2t}{100d^2}$ is at most $e^{-t/(300d^2)} \leq e^{-t/(600d^2)}$.

Thus, by the union bound the probability that $|A_1 \cap B_1| + |A_2 \cap B_2| < \frac{4t}{100d^2} < \frac{t}{20d^2}$ is at least $1 - 2e^{-t/(600d^2)} = 1 - 2e^{-8k_1k_2} \geq 1 - 2N^{-11}$. Taking the union bound over all $\binom{N}{2}$ choices of i and j, we conclude that (ii) is violated with probability at most N^{-9}. Overall, condition (a) does not hold with probability at most $1/N^{39d} + 1/N^9 \leq 2/N^9$.

We now bound the probability that condition (b) holds. To this end we will show that for any $1 \leq i \leq n_1$, C^i_{in} is not (d, d)-list disjunct with probability at most q^{-58d}. Then by the union bound and the fact that $n_1 \leq 2^{k_2} = q$, condition (b) does not hold with probability at most q^{-57d}. For the rest of the proof, fix an $1 \leq i \leq n_1$. For notational convenience, we will denote the matrix $M_{C^i_{\text{in}}}$ as M^i. We note that the $n_2 \times q$ matrix M^i is not (d, d)-list disjunct if there exists two disjoint subsets $S_1, S_2 \subseteq [q]$ of size d such that for *every* $j \in S_1$, $M^i_j \subseteq \cup_{\ell \in S_2} M^i_\ell$. We will upper bound the probability of such an event happening. Note that all the entries in M^i are independent random variables. This implies (along with the union bound) that the probability that M^i is not (d, d)-list disjunct is at most

$$\begin{aligned}
&\binom{q}{d}\binom{q-d}{d}\left(1 - \left(1 - \left(1 - \frac{1}{10d}\right)^d\right)\left(1 - \frac{1}{10d}\right)^d\right)^{n_2} \\
&\leq q^{2d}\left(1 - (1 - 1/\sqrt[10]{e}) \cdot \frac{1}{\sqrt[10]{e}}\right)^{n_2} \\
&\leq q^{2d} 2^{-n_2/8} \leq q^{-58d},
\end{aligned}$$

where the last inequality follows from our choice of $n_2 = 480dk_2$.

Thus, with probability at least $1 - \frac{2}{N^9} - \frac{1}{2^{57dk_2}}$ both of the required properties are satisfied. □

We use the theorem above to prove the existence of an efficiently decodable disjunct matrix matching the best upper bound known to date.

COROLLARY 5.1. *Let* $N > d \geq 1$ *be any given integers. There exists a* $t \times N$ d*-disjunct matrix with*

$t = O(d^2 \log N)$ *that can be decoded in time* $\text{poly}(d) \cdot t \log^2 t + O(t^2)$.

Proof. We ignore the issue of integrality for the sake of clarity. First, suppose $N \geq 100d^2$. Set $k_2 = \log_2(10d \log_2 N) \geq 1$ and $k_1 = \frac{\log_2 N}{k_2} \geq 1$. It follows that $10dk_1 \leq 2^{k_2}$. Theorems 5.1, 4.1 and 2.1 complete the proof. The outer code in Theorem 5.1 is RS with rate $\frac{1}{10d}$, which is known to have relative distance $1 - \frac{1}{10d}$. The decoding of the outer RS code is done via the algorithm in [1] that runs in $\text{poly}(d) \cdot t \log^2 t$ time. The naive decoding algorithm is used to decode the inner codes, which takes $O(t^2)$ time.

Next, when $d < N < 100d^2$ we can remove arbitrarily $100d^2 - N$ columns from a $t \times 100d^2$ efficiently decodable d-disjunct matrix to obtain a $t \times N$ efficiently decodable d-disjunct matrix which still satisfies $t = O(d^2 \log N)$. □

5.2 Derandomizing the Proof of Corollary 5.1

Using Nisan's PRG for space bounded computation [23], we can reduce the amount of randomness in Corollary 5.1.

COROLLARY 5.2. *Let $N > d \geq 1$ be any given integer. Then using $O(\log t \cdot \max(\log N, d \log t))$ random bits, with probability $1 - o(1)$, one can construct a $t \times N$ d-disjunct matrix with $t = O(d^2 \log N)$ that can be decoded in time* $\text{poly}(d) \cdot t \log^2 t + O(t^2)$.

Proof. The idea is to apply Nisan's PRG G for space bounded computations. In particular, Nisan's PRG can fool computations on R input bits with space $S \geq \Omega(\log R)$ by using only $O(S \log R)$ pure random bits. (This PRG has low space requirements and is strongly explicit, which useful for data stream algorithms.) The idea is to use G on the computation that checks conditions (a) and (b) in proof of Theorem 5.1. Let's start with the checks needed for condition (a). This has two parts. First, we need to check that all the N columns in M^* has Hamming weight at least $\frac{t}{20d} + 1$. To do this we need $O(\log N)$ bits to keep track of the column index. Then for each column we have to

- Compute the element in the corresponding RS code for each of the n_1 outer positions (we will need $O(\log n_1) = O(k_2)$ space to keep an index for these positions). Each such symbol from $\mathbb{F}_{2^{k_2}}$ can be computed in space $O(k_2)$ (by the fact that the generator matrix of RS codes is strongly explicit);
- For each symbol in RS codeword, we need to count the number of ones in the encoding in the corresponding inner codes. For this we will need $O(\log t)$ space to access an element of the $O(t) \times O(t)$ inner code matrix and another $O(\log t)$ space to maintain the counter.

Thus, overall we will need $O(\log N)$ space. The next part of checking condition (a) will need to us compute the Hamming distance between $\binom{N}{2}$ columns. Using the accounting above, we conclude that this step can also be computed with $O(\log N)$ space.

We now account for the amount of space needed to check condition (b). To do this, we need to check that all of the n_1 inner codes are (d, d)-list disjunct. For each of the inner codes, we need to consider all pairs of disjoint subsets of $[2^{k_2}]$ of size d. (One can keep track of all such subsets in space $O(dk_2) = O(n_2) = O(d \log t)$. For the latter equality we will need to choose $n_1 = \Theta(2^{k_2})$.) Then for each pairs of such subsets of columns, we need to check if for some row, all the columns in one subset have 0 while at least one column in the other subset has a 1. Since there are n_2 rows in the inner code matrix, this check be done in space $O(\log n_2 + \log d) = O(\log n_2)$. Thus, overall to check condition (b), we need $O(d \log t)$ space.

Thus, overall we are dealing with a $S = O(\max(\log N, d \log t))$ space computation on $R = O(t^2)$ inputs. However, there is a catch in that Nisan's PRG works with R unbiased random bits while in our proof we are dealing with R $\frac{1}{10d}$-biased bits. However, it is easy to convert $R' = O(R \log d)$ unbiased random bits to R $\frac{1}{10d}$-biased bits (e.g., by grouping $O(\log d)$ chunks of unbiased bit and declaring a 1 if and only if all the bits in the chunk are 1.) Further, this conversion only needs an additional $O(\log \log d + \log R)$ space. Given this conversion, we can assume that our proof needs unbiased bits. Thus, we can apply G with $S' = S + O(\log \log d + \log t) = O(S)$ and $R' = O(t^2 \log d) = O(t^3)$. Thus, this implies that we can get away with $O(\log t \cdot \max(\log N, d \log t))$ unbiased random bits. □

Corollary 5.2 implies that we can get a deterministic construction algorithm with time complexity $\max\left(N^{O(\log d + \log \log N)}, N^{O(d \log^2 d / \log N)}\right)$, by cycling through all possible random bits. Next, we use the method of conditional expectation to get a faster deterministic construction time.

COROLLARY 5.3. *Let $N > d \geq 1$ be any given integer. There exists a deterministic $O(t^5 N^2) + N^{O(d \log d / \log N)}$-time and $\text{poly}(t)$-space algorithm to construct a $t \times N$ d-disjunct matrix with $t = O(d^2 \log N)$ that can be decoded in time* $\text{poly}(d) \cdot t \log^2 t + O(t^2)$.

Proof. Let us define three sets of random variables. For every $i \in [N]$, let X_i be the indicator variable of the event that the ith column of M^* has Hamming weight

at most $\frac{t}{20d}$. Further, for every pair $i \neq j \in [N]$, define $Y_{i,j}$ to be the indicator variable of the event that the ith and the jth columns of M^* agree in strictly more than $t/(20d^2)$ positions. Finally, for any $i \in [n_1]$ and disjoint subsets $S, T \subseteq [q]$ with $|S| = |T| = d$, define $Z_{i,S,T}$ denote the indicator variable for the event that for the ith inner code (whose matrix is denoted by M^i), $\cup_{j \in S} M^i_j \subseteq \cup_{j \in T} M^i_j$. Define $V = \sum X_j + \sum Y_{i,j} + \sum Z_{i,S,T}$. Note that in the proof of Theorem 5.1, we have shown that over the random choices for the inner codes,

$$\begin{aligned}\mathbb{E}[V] &= \sum_{j \in [N]} \mathbb{E}[X_j] + \sum_{i \neq j \in [N]} \mathbb{E}[Y_{i,j}] \\ &\quad + \sum_{j=1}^{n_1} \sum_{S \subseteq [q], |S| = d} \sum_{\substack{T \subseteq [q], \\ |T| = d, T \cap S = \emptyset}} \mathbb{E}[Z_{j,S,T}] \\ &< \frac{2}{N^9} + \frac{1}{2^{57dk_2}}.\end{aligned}$$

And, since k_2 was chosen to be $\log_2(10d \log_2 N)$ in the proof of Corollary 5.1, we have $\mathbb{E}[V] < \frac{2}{N^9} + \frac{1}{(10d \log_2 N)^{57d}}$.

To derandomize the proof, we will use the standard method of conditional expectation by fixing each of the random bits one at a time (in an arbitrary order) and assigning the bit 0 or 1, whichever minimizes the conditional expectation of V given the corresponding assignment. Recall that the random bits define n_1 inner $n_2 \times q$ matrices, and that each bit is chosen to be one with probability $\frac{1}{10d}$. Note that we can apply the method of conditional expectation if we can compute the following conditional probabilities (where A denotes an assignment to an arbitrary subset of bits in the inner codes):

1. For every $j \in [N]$, $\Pr[X_j = 1|A]$;
2. For every $i \neq j \in [N]$, $\Pr[Y_{i,j} = 1|A]$;
3. For every $i \in [n_1]$ and disjoint subsets $S, T \subseteq [q]$ with $|S| = |T| = d$, $\Pr[Z_{i,S,T} = 1|A]$.

We claim that each of the probabilities above can be computed in time $O(t^3)$ (assuming C_{out} is a strongly explicit linear code, e.g. RS code).[2] This implies that each of the $n_1 n_2 q \leq t^2$ conditional expectation values of V can be computed in time $O(Nt^3) + O(N^2 t^3) + t^{O(d)}$. Thus, the overall running time of the algorithm is

$$O(t^5 N^2) + 2^{O(d(\log d + \log\log N))} = O(t^5 N^2) + N^{O\left(\frac{d \log d}{\log N}\right)}.$$

[2] We do not try to optimize exact polynomial dependence on t here.

Note that the construction time is polynomial in N as long as d is $O(\log N / \log\log N)$.

We conclude this proof by showing how to compute the three kinds of probability. We begin with the $X_{(\cdot)}$ indicator variables. Fix $j \in [N]$. First we need to compute the codeword $C_{\text{out}}(j)$. If C_{out} is a strongly explicit linear code, this can be accomplished with $O(t^2)$ multiplications and additions over $\mathbb{F}_{2^{k_2}}$. Since $2^{k_2} \leq t$, each of these operations can definitely be performed in time $O(t)$. Thus, this entire step takes $O(t^3)$ time. Now to compute $\Pr[X_j|A]$, we need to substitute the ith ($1 \leq i \leq n_1$) symbol in $C_{\text{out}}(j)$ (over $\mathbb{F}_{2^{k_2}}$) with the corresponding column in $M_{C^i_{\text{in}}}$. Note that the resulting vector will have some bits fixed according to A while the rest are independent random bits that take a value of 1 with probability $\frac{1}{10d}$. Thus, we are left with computing the "tail" of a Binomial distribution on $O(t)$ trials each with a success probability of $\frac{1}{10d}$. This can be trivially computed in time $O(t^2)$. Computing $\Pr[Y_{i,j} = 1|A]$ is similar to the computation above (the only difference being we have to deal with a Binomial distribution where the success probability is $\frac{1}{100d^2}$) and thus, also takes $O(t^3)$ time overall.

We finally turn to the $Z_{(\cdot,\cdot,\cdot)}$ indicator variables. Fix $i \in [n_1]$ and disjoint subsets $S, T \subseteq [q]$ with $|S| = |T| = d$. Note that we want to compute $\prod_{j \in [n_2]} p_j$, where p_j denote the probability that if the jth row of $M_{C^i_{\text{in}}}$ has a one in the columns from S then it also has a one in the columns spanned by T. Note that with the partial assignment A, p_j is either 0, 1 or is $1 - \left(1 - \left(1 - \frac{1}{10d}\right)^a\right)\left(1 - \frac{1}{10d}\right)^b$ (where a and b are the number of columns in S and T where the values in the jth row are not yet fixed by A). It is easy to check that each such p_j can be computed in time $O(d^2)$. Thus, $\mathbb{E}[Z_{i,S,T}|A]$ can definitely be computed in time $O(t^2)$.

We conclude by noting that the algorithm above can be implemented in poly(t) space. □

6 More on List Disjunct Matrices and their Applications

We begin with connections between list disjunct matrices and dispersers and expanders. (We also present a connection between expanders and disjunct matrices in Appendix D.) Then we present two applications of list disjunct matrices.

6.1 Connection to Dispersers We now state a simple connection between list disjunct matrices and dispersers. A D-left regular bipartite graph $[N] \times [D] \to [T]$ is an (N, L, T, D, ϵ)-disperser if every subset $S \subset [N]$ of size at least L has a neighborhood of size at least $\epsilon|T|$. Given such a bipartite expander G, consider the $T \times N$ incidence matrix M_G of G. We make the following

simple observation.

PROPOSITION 6.1. *Let G be an $(n, \ell, t, \epsilon t/(d+1), \epsilon)$-disperser. Then M_G is a (d, ℓ)-list disjunct matrix.*

Proof. Consider two disjoint subsets of columns S_1 and S_2 with $|S_1| \geq \ell$ and $|S_2| = d$. To prove the claim, we need to show that $\cup_{i \in S_1} M_i \not\subseteq \cup_{i \in S_2} M_i$. We prove the later inequality by a simple counting argument. By the value of the degree of G, we get that

$$|\cup_{i \in S_2} M_i| \leq \frac{dt\epsilon}{d+1} < \epsilon t.$$

On the other hand, as G is a disperser, $|\cup_{i \in S_1} M_i| \geq \epsilon t$, which along with the above inequality implies that $\cup_{i \in S_1} \not\subseteq \cup_{i \in S_2}$, as desired. □

The best known explicit constructions of dispersers is due to Zuckerman [32], who presents explicit $(N, N^{\delta}, N^{(1-\alpha)\delta}, O(\log N/\log \gamma^{-1}), \gamma)$-dispersers, which by Proposition 6.1 implies the following:

THEOREM 6.1. *Let $\gamma, \epsilon > 0$ be any constants. Then there exists an explicit $t \times N$ (d, ℓ)-list disjunct matrix with $t = O(d \log N)$, $d = N^{\gamma}$ and $\ell = d^{1+\epsilon}$.*

6.2 Connection to Expanders We now state a simple connection between list disjunct matrices and expanders. A W-left regular bipartite graph $[N] \times [W] \to [T]$ is an $(N, W, T, D, (1-\epsilon)W)$ expander if every subset $S \subset [N]$ of size at most D has a neighborhood (denoted by $\Gamma(S)$) of size at least $(1-\epsilon)|S|W$. Given such a bipartite expander G, consider the $T \times N$ incidence matrix M_G of G. We have the following simple observation.

PROPOSITION 6.2. *Let G be a $(n, w, t, 2d, w/2+1)$-expander. Then M_G is a (d, d)-list disjunct matrix.*

Proof. Recall that by definition a matrix M is (d, d)-list disjunct if the following is true: for every two disjoint S_1 and S_2 subsets of columns of size exactly d, both $\cup_{i \in S_1} M_i \not\subseteq \cup_{j \in S_2} M_j$ and $\cup_{j \in S_2} M_j \not\subseteq \cup_{i \in S_1} M_i$ hold. Note that this property for M_G translates to the following for G: $\Gamma_G(S_1) \not\subseteq \Gamma_G(S_2)$ and vice-versa. We now argue that the latter is true if G has an expansion of $w/2+1$. Indeed this follows from the facts that $|\Gamma_G(S_1 \cup S_2)| \geq wd + 2d$ and $|\Gamma_G(S_1)|, |\Gamma_G(S_2)| \leq wd$. □

We remark that in our application, we are not really concerned about the value of W and are just interested in minimizing T. (Generally in expanders one is interested in minimizing both simultaneously.) The probabilistic method shows that there exist $(N, W, T, D, W/2+1)$-expanders with $T = O(D \log N)$ (and $W = O(\log N)$). This by Proposition 6.2 implies that

COROLLARY 6.1. *There exist $t \times N$ matrices that are (d, d)-list disjunct with $t = O(d \log N)$.*

Later on in this section, we show that there exists an explicit $(N, W, T, D, W/2+1)$-expander with $T = (D \log N)^{1+o(1)}$. (We thank Chris Umans for the proof.) This with Proposition 6.2 implies the following:

COROLLARY 6.2. *There is an explicit $t \times N$ matrix that is (d, d)-list disjunct with $t = (d \log N)^{1+o(1)}$.*

Cheraghchi achieves a slightly better construction with $d^{1+o(1)} \log n$ tests.

6.2.1 An Explicit Expander Construction To construct an *explicit* expander for our purposes, we will use the following two constructions.

THEOREM 6.2. ([19]) *Let $\epsilon > 0$. There exists an* explicit *$(N_1, W_1, T_1, D_1, W_1(1-\epsilon))$ expander with $T_1 \leq (4D_1)^{\log W_1}$ and $W_1 \leq 2 \log N_1 \log D_1/\epsilon$.*

THEOREM 6.3. ([29]) *Let $\epsilon > 0$ be a* constant. *Then there exists an* explicit *$(N_2, W_2, T_2, D_2, W_2(1-\epsilon))$-expander with $T_2 = O(D_2 W_2)$ and $W_2 = 2^{O(\log\log N_2 + (\log\log D_2)^3)}$.*

We will combine the above two expanders using the following well known technique.

PROPOSITION 6.3. *Let G_1 be an $(N, W_1, T_1, D, W_1(1-\epsilon))$-expander and G_2 be an $(T_1, W_2, T_2, DW_1, W_2(1-\epsilon))$-expander. Then there exists an $(N, W_1 W_2, T_2, D, W_1 W_2(1-2\epsilon))$-expander G. Further, if G_1 and G_2 are explicit then so is G.*

Proof. The graph G is constructed by "concatenating" G_1 and G_2. In particular, construct the following intermediate tripartite graph G' (on the vertex sets $[N]$, $[T_1]$ and $[T_2]$ respectively), where one identifies $[T_1]$ once as the right vertex set for G_1 and once as the left vertex set of G_2. The final graph G is bipartite graph on $([N], [T_2])$ where there is an edge if and only if there is a corresponding path of length 2 in G'. It is easy to check that G is an $(N, W_1 W_2, T_2, D, W_1 W_2(1-2\epsilon))$-expander. □

Next, we prove the following result by combining all the ingredients above.

THEOREM 6.4. *There exists an explicit $(N, W, T, D, W/2+1)$-expander with $T = O(D \log N \cdot f(D, N))$, where*

$$f(D, N) = 2^{O((\log\log D)^3 + (\log\log\log N)^3)}.$$

Note that $f(D,N) = (D \log N)^{o(1)}$.

By Theorem 6.2, there exists an explicit $(N, W_1, T_1, D, 3W_1/4+1)$-expander, where

$$W_1 \leq 16 \log N \log D,$$

and

$$T_1 \leq (4D)^{4+\log\log N + \log\log D}.$$

By Theorem 6.3, there exists an explicit $(T_1, W_2, T_2, D_2, 3W_2/4+1)$-expander, where

$$D_2 = DW_1 \leq 16D \log N \log D,$$

W_2 is $2^{O(\log\log T_1 + (\log\log D_2)^3)}$, which in turn is at most $2^{O((\log\log D)^3 + (\log\log\log N)^3)}$, and T_2 is

$$\begin{aligned} W_2 D_2 &\leq 16D \log N \log D \cdot 2^{O((\log\log D)^3 + (\log\log\log N)^3)} \\ &\leq D \log N \cdot f(D,N), \end{aligned}$$

as desired.

Next, we move onto applications of list disjunct matrices.

6.3 Sparsity Separator Structure Ganguly in [14] presents a deterministic streaming algorithm for d-sparsity testing, which is defined as follows. Given a stream of m items from the domain $[n]$, let f be the vector of frequencies, i.e., for every $i \in [n]$, f_i denotes the number of occurrences of i in the stream. The d sparsity problem is to determine if f has at most d non-zero entries. [14] presents an algorithm for the special case when $f_i \geq 0$ (otherwise the problem is known to require linear space for deterministic algorithms).

A crucial building block of Ganguly's algorithm is what he calls a (d,ℓ)-sparsity separator structure, which is a data structure that can determine if the frequency vector corresponding to a stream has at most d non-zero entries or it has at least ℓ non-zero entries. We now show how any $(d, \ell-d)$-list disjunct $t \times n$ matrix M can be used to build a (d,ℓ)-sparsity separator structure. The idea is almost the same as the use of disjunct matrices in determining hot items [7]: we maintain t counters, one for each test (i.e. $\{c_j\}_{j\in[t]}$). Whenever an item i arrives/leaves, increment/decrement all counters c_j such that the jth test contains i. At the end convert the counters to a result vector $r \in \{0,1\}^t$ in a straightforward manner: $r_j = 1$ if and only if $c_j > 0$. We decode the result vector r according to M and if the number of ones in the output is at most $\ell - 1$ then declare the sparsity to be at most d otherwise declare the sparsity to be at least ℓ.

We now briefly argue the correctness of the algorithm above. First define a binary vector $x \in \{0,1\}^n$ such that $x_i = 1$ if and only if $f_i > 0$. Since all the frequencies are non-negative, it is easy to see that the result vector r computed above is exactly the same as the one that would result from M acting on x. Now consider the two cases (a) x has Hamming weight at most d. In this case as M is $(d, \ell-d)$-list disjunct, the decoder for M will output a vector with at most $\ell - 1$ ones in it. (b) Now let us consider the case that x has Hamming weight at least ℓ. In this case, note that each $x_i = 1$ will contribute a one to all the r_j such that i is contained in test j. Thus, the decoder for M will output a vector with *at least* ℓ ones in it. This completes the proof of correctness of the algorithm above.

We would like to point out in the above we assumed the following property of the decoder for M: when provided with an input x with Hamming weight more than d, the decoder will output a vector with Hamming weight larger than that of x. It is easy to check that the naive decoding algorithm has this property. However, the decoding time is not longer sub-linear.

We also remark that the the way Ganguly uses lossless expanders to construct $(d, 2d)$-sparsity separator structures in *exactly* the same way we use them to construct (d,d)-list disjunct matrices in Proposition 6.2. However, [14] uses properties of expanders to come up with a more efficient "decoder" than what we have for (d,d)-list disjunct matrices.

6.4 Tolerant Testing of Reed-Solomon Codes Rudra and Uurtamo in [27] consider the classical codeword testing problem in the data stream domain. In particular, they consider one pass data stream algorithms for *tolerant testing* and *error detection* of Reed-Solomon (RS) codes. Informally, in the former problem, given a received word, the tester has to decide whether it is close to some RS codeword or is far from every RS codeword. In the latter problem, one has to decide whether the received word is a codeword or not.

Using a slight modification of the well known fingerprinting technique, they give a poly-log space one pass data stream algorithm to solve the error detection problem for RS codes. Then they reduce the tolerant testing problem to error detection via (list) disjunct matrices. In particular, assume that the tolerant tester wants to distinguish between the case when the received word is at Hamming distance at most d from some RS codeword and the case that it is at a Hamming distance at least ℓ from every RS codeword. The reduction is not trivial, so we just sketch the main idea in the reduction here. (They crucially use the fact that any RS code projected onto a (large enough) subset of positions is also an RS code.) Here is a simple idea that does not quite work. Fix a $(d, \ell-d)$-list disjunct matrix M. Then

for each test, check if the corresponding projected down received word belongs to the corresponding RS code (using the error detection algorithm). Then given the outcome vector use the decoding algorithm to determine whether at most d or at least ℓ errors have occurred as in the previous application. The catch is that the correspondence is not necessarily one to one in the sense that some of the test answers might have false negatives (as a projected down error pattern might be a codeword in the corresponding projected down RS code). In [27], this issue is resolved by appealing to the list decoding properties of RS codes.

Acknowledgments We thank Venkat Guruswami and Chris Umans for helpful discussions. Thanks again to Chris Umans for kindly allowing us to use his proof of Theorem 6.4.

References

[1] M. Alekhnovich. Linear diophantine equations over polynomials and soft decoding of reed-solomon codes. *IEEE Transactions on Information Theory*, 51(7):2257–2265, 2005.

[2] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006.

[3] T. Berger, N. Mehravari, D. Towsley, and J. Wolf. Random multiple-access communications and group testing. *IEEE Trans. Commun.*, 32(7):769–779, 1984.

[4] Y. Caro and Z. Tuza. Improved lower bounds on k-independence. *J. Graph Theory*, 15(1):99–107, 1991.

[5] H.-B. Chen and F. K. Hwang. A survey on nonadaptive group testing algorithms through the angle of decoding. *J. Comb. Optim.*, 15(1):49–59, 2008.

[6] M. Cheraghchi. Noise-resilient group testing: Limitations and constructions. In *Proceedings of 17th International Symposium on Fundamentals of Computation Theory (FCT)*, 2009. To Appear.

[7] G. Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM Trans. Database Syst.*, 30(1):249–278, 2005.

[8] R. Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14(4):436–440, 1943.

[9] D.-Z. Du and F. K. Hwang. *Combinatorial group testing and its applications*, volume 12 of *Series on Applied Mathematics*. World Scientific Publishing Co. Inc., River Edge, NJ, second edition, 2000.

[10] A. G. Dýachkov and V. V. Rykov. Bounds on the length of disjunctive codes. *Problemy Peredachi Informatsii*, 18(3):7–13, 1982.

[11] A. G. Dýachkov, V. V. Rykov, and A. M. Rashad. Superimposed distance codes. *Problems Control Inform. Theory/Problemy Upravlen. Teor. Inform.*, 18(4):237–250, 1989.

[12] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković. Efficient approximation of product distributions. *Random Structures Algorithms*, 13(1):1–16, 1998.

[13] Z. Füredi. On r-cover-free families. *J. Comb. Theory, Ser. A*, 73(1):172–173, 1996.

[14] S. Ganguly. Data stream algorithms via expander graphs. In *19th International Symposium on Algorithms and Computation (ISAAC)*, pages 52–63, 2008.

[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)*, pages 206–221, 2005.

[16] V. Guruswami and P. Indyk. Linear-time list decoding in error-free settings: (extended abstract). In *Proceedings of of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 695–707, 2004.

[17] V. Guruswami and A. Rudra. Concatenated codes can achieve list-decoding capacity. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 258–267, 2008.

[18] V. Guruswami and A. Rudra. Soft decoding, dual BCH codes, and better list-decodable ϵ-biased codes. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity (CCC)*, pages 163–174, 2008.

[19] V. Guruswami, C. Umans, and S. P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. In *Proceedings of the 22nd Annual IEEE Conference on Computational Complexity*, pages 96–108, 2007.

[20] W. H. Kautz and R. C. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10(4):363–377, 1964.

[21] A. J. Macula and L. J. Popyack. A group testing method for finding patterns in data. *Discrete Appl. Math.*, 144(1-2):149–157, 2004.

[22] H. Q. Ngo and D.-Z. Du. A survey on combinatorial group testing algorithms with applications to DNA library screening. In *Discrete mathematical problems with medical applications (New Brunswick, NJ, 1999)*, volume 55 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 171–182. Amer. Math. Soc., Providence, RI, 2000.

[23] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[24] F. Parvaresh and A. Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 285–294, 2005.

[25] E. Porat and A. Rothschild. Explicit non-adaptive combinatorial group testing schemes. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 748–759, 2008.

[26] A. Rudra. *List Decoding and Property Testing of*

Error Correcting Codes. PhD thesis, University of Washington, 2007.

[27] A. Rudra and S. Uurtamo. Data stream algorithms for codeword testing, July 2009. Manuscript.

[28] H. Shachnai and A. Srinivasan. Finding large independent sets in graphs and hypergraphs. *SIAM Journal on Discrete Math*, 18:488–500, 2004.

[29] A. Ta-Shma, C. Umans, and D. Zuckerman. Lossless condensers, unbalanced expanders, and extractors. *Combinatorica*, 27(2):213–240, 2007.

[30] C. Thommesen. The existence of binary linear concatenated codes with Reed-Solomon outer codes which asymptotically meet the Gilbert-Varshamov bound. *IEEE Transactions on Information Theory*, 29(6):850–853, November 1983.

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

[32] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.

A Data Forensics and Group Testing

Goodrich, Atallah and Tamassia [15] present the following application of group testing to data forensics. Data forensics refers to the following problem in data structures and security. Assume that one needs to store a data structure on a semi-trusted place. An adversary can change up to d values (out of a total of n values) in the data structure (but cannot change the "layout" of the data structure). The goal is to "store" extra information into the data structure so that if up to d values in the data structure are changed then an auditor can quickly pinpoint the locations where data was changed. The authors point out that an auditor generally has very few resources and thus one needs to store all the book-keeping information in the data structure itself. The authors consider certain data structures and show that extra information can be encoded into the "layout" of the data structure. For example, given n numbers, there are $n!$ possible layouts and thus one can encode $O(n \log n)$ bits of information into the layout. (They also show how to do this information "hiding" in other data structures such as balanced binary search trees and skip lists.)

The question then is what information should be stored in the layout of the data structure such that the auditor can detect the changes. The authors propose the use of a d-disjunct matrix (where each of the columns correspond to the n items) as follows. For each test in the matrix, concatenate all the values of the items present in the test and store its hash value. This requires $O(t)$ bits of storage. Further, the matrix itself has to be stored. Towards this end, the authors present a randomness efficient construction of a d-disjunct matrix with $t = O(d^2 \log n)$ that needs $O(d^3 \log^2 n)$ random bits. (That is, any entry of the matrix can be computed given these bits and with high probability over the random choices, the resulting matrix is d-disjunct.) Thus, the overall number of bits to be encoded into the layout of the data structure is $O(d^3 \log^2 n)$. Given a possibly altered data structure, the auditor uses the information stored in the layout of the data structure to recompute the hashes according to the current values for each test in the matrix. If the computed hash value differs from the one stored, it amounts to the test being positive. Computing the locations where values were changed now just amount to decoding of the disjunct matrix. Using this framework, the authors prove that for balanced binary trees, skip lists and arrays (as well as linked lists) that store n items, the scheme above allows the auditor to pin-point up to $O(\sqrt[3]{n/\log^2 n})$, $O(\sqrt[3]{n/\log^2 n})$ and $O(\sqrt[4]{n/\log n})$ many changes respectively. (The authors show that for the layout of the data structures can store $\Theta(n)$, $\Theta(n)$ and $\Theta(n \log n)$ bits respectively. The bounds on d are obtained by equating the space requirements for the algorithms to the space available in the data structures.[3])

We remark that in [15], the representation of the matrix takes up more space than the space required to store the hash values. Our randomness efficient construction requires $O(\max(\log n(\log d + \log n \log n), d(\log d + \log\log n)^2)) = O(d \log^2 n)$ many random bits, which is better than the bound in [15]. Using our result in the framework of Goodrich et al., we can improve their results to show that for balanced binary trees, skip lists and arrays (as well as linked lists) that store n items, the auditor can pin-point up to $O(\sqrt[2]{n/\log n})$, $O(\sqrt[2]{n/\log n})$ and $O(\sqrt[3]{n})$ many changes respectively. Note that these bounds are the best possible with the techniques of [15]. In addition, given that our disjunct matrices are efficiently decodable, the auditor can pin-point the changes faster than the algorithm in [15]. We remark that the result of Porat and Rothschild [25] will also provide similar improvements in the number of changes that can be tolerated, though the decoding algorithms will be slower than ours.

B Omitted proofs from Section 3

B.1 Proof of Lemma 3.1 We will only need the binary version of the k-restriction problem, which is defined as follows. The input to the problem is an

[3] For the array/linked list data structures, due to technical reasons, the space requirement of the algorithms has be be increased by a factor of $d + 1$.

alphabet $\Sigma = \{0,1\}$, a length n, and a set of m possible *demands* $f_i : \Sigma^k \to \{0,1\}$, $1 \le i \le m$. For every demand f_i, there is at least one vector $\mathbf{a} = (a_1, \ldots, a_k) \in \Sigma^k$ such that $f_i(\mathbf{a}) = 1$ (in words, f_i "demands" vector $\mathbf{a}$). Thus, every demand f_i "demands" a non-empty subset of vectors from Σ^k. A feasible solution to the problem is a subset $S \subseteq \Sigma^n$ such that: for any choice of k indices $1 \le j_1 < j_2 < \cdots < j_k \le n$, and any demand f_i, there is some vector $\mathbf{v} = (v_1, \ldots, v_n) \in S$ such that the projection of $\mathbf{v}$ onto those k indices satisfies demand f_i; namely, $f_i(v_{j_1}, v_{j_2}, \ldots, v_{j_k}) = 1$. The objective is to find a feasible solution S with minimum cardinality. Alon, Moshkovitz, and Safra gave a couple of algorithmic solutions to the k-restriction problems. In order to describe their results, we need a few more concepts.

Given a distribution $\mathcal{D} : \Sigma^n \to [0,1]$, the *density* of a k-restriction problem with respect to $\mathcal{D}$ is

$$\epsilon := \min_{\substack{1 \le j_1 < \cdots < j_k \le n \\ 1 \le i \le m}} \{\Pr_{\mathbf{v} \leftarrow \mathcal{D}} [f_i(v_{j_1}, v_{j_2}, \ldots, v_{j_k}) = 1]\}$$

In words, for any k positions $j_1, \ldots, j_k$, and any demand f_i, if we pick a vector $\mathbf{v}$ from Σ^n at random according to $\mathcal{D}$ then the projection of $\mathbf{v}$ onto those k positions satisfies f_i with probability at least ϵ. The *restriction* $\mathcal{D}_{j_1,\ldots,j_k}$ of a distribution $\mathcal{D}$ on Σ^n to indices $j_1, \ldots, j_k$ is defined by

$$\mathcal{D}_{j_1,\ldots,j_k}(\mathbf{a}) := \Pr_{\mathbf{v} \leftarrow \mathcal{D}} [v_{j_1} = a_1 \wedge \cdots \wedge v_{j_k} = a_k]$$

Two distribution $\mathcal{P}, \mathcal{Q}$ on Σ^n are said to be *k-wise ϵ-close* if, for any $1 \le j_1 < j_2 < \cdots < j_k \le n$, $\|\mathcal{P}_{j_1,\ldots,j_k} - Q_{j_1,\ldots,j_k}\|_1 < \epsilon$. The *support* of a distribution on Σ^n is the number of members of Σ^n which have positive probabilities. Finally, a distribution $\mathcal{D}$ on Σ^n is said to be *k-wise efficiently approximable* if the support of a distribution $\mathcal{P}$ which is k-wise ϵ-close to $\mathcal{D}$ can be enumerated in time $\text{poly}(m, \frac{1}{\epsilon}, 2^k)$.

One of the main results from [2] is the following. The reader is referred to their paper for the definitions of k-wise efficiently approximable distributions.

THEOREM B.1. (ALON-MOSHKOVITZ-SAFRA [2]) *Fix some k-wise efficiently approximable distribution $\mathcal{D}$ on Σ^n. For any k-restriction problem with density ϵ with respect to $\mathcal{D}$, there is an algorithm that, given an instance of the problem and a constant parameter $0 < \delta < 1$, obtains a solution S of size at most $\lceil \frac{k \ln n + \ln m}{(1-\delta)\epsilon} \rceil$ in time poly$(m, n^k, 2^k, \frac{1}{\epsilon}, \frac{1}{\delta})$.*

A distribution $\mathcal{D}$ on $\Sigma^n = \{0,1\}^n$ is called a *product distribution* if all coordinates are independent Bernoulli variables. (Coordinate i is 1 with probability p_i for some fixed p_i.)

THEOREM B.2. ([12]) *Any product distribution on Σ^n is k-wise efficiently approximable.*

We are now ready to prove Lemma 3.1.

First, we show that constructing (d,ℓ)-list disjunct matrices is a special case of the k-restriction problem. Consider an arbitrary $t \times n$ (d,ℓ)-list disjunct matrix M. The matrix satisfies the property that, for any set of d columns $C_1, \ldots, C_d$, and any disjoint set $C'_1, \ldots, C'_\ell$ of ℓ other columns, there exists a row i of M in which $C'_1(i) = \cdots = C'_d(i) = 0$ and $C_1(i) + \cdots + C_\ell(i) > 0$.

Let $k = d + \ell$. Define $m = \binom{d+\ell}{d}$ "demands" $f_I : \Sigma^k \to \{0,1\}$ as follows. For every d-subset $I \in \binom{[k]}{d}$, $f_I(\mathbf{a}) = 1$ if and only if $a_j = 0, \forall j \in I$ and $\sum_{j \notin I} a_j > 0$. We just set up an instance of the k-restriction problem. A solution S to this instance forms the rows of our $t \times n$ matrix.

Next, let $\mathcal{D}$ be the product distribution on $\{0,1\}^n$ defined by setting each coordinate to be 1 with probability p to be determined. Then, $\mathcal{D}$ is k-wise efficiently approximable by Theorem B.2. Fix $1 \le j_1 < j_2 < \cdots < j_k \le n$ and any demand f_I. Choose any vector $\mathbf{v}$ from $\{0,1\}n$ according to $\mathcal{D}$. The projection of $\mathbf{v}$ onto coordinates $j_1, \ldots, j_k$ "satisfies" f_I with probability

$$\epsilon(p) = \left(1 - (1-p)^\ell\right)(1-p)^d.$$

The density ϵ is maximized at $p_0 = 1 - \left(\frac{d}{d+\ell}\right)^{1/\ell}$. Set $\epsilon_0 = \epsilon(p_0)$, $\delta = 1/2$, and apply Theorem B.1.

C Omitted Proofs from Section 4

By concatenating RS codes with the identity code, we get efficient group testing with $O(d^2 \log^2 N)$-tests.

COROLLARY C.1. *Given any integers $N > d \ge 1$, there exists a $t \times N$ d-disjunct matrix M with $t = O(d^2 \log^2 N)$ that can be decoded in time* poly(t). *Furthermore, each entry of M can be computed in space* poly$(\log t, \log N)$.

Proof. The construction is classic [20]: set $M = M_{\text{RS} \circ I_q}$, where $q = 2^{k_2}$ is some power of 2, I_q is the identity code of order q, and RS is the $(q-1, k_1)_q$ RS code. The numbers k_1 and k_2 are to be chosen based on N and d. Then, we apply Theorem 4.1 with $C_{\text{out}} = \text{RS}$ and $C^i_{\text{in}} = I_q$ for all i which shows that M is an efficiently decodable d-disjunct matrix. In the following, we will ignore the issue of integrality for the sake of clarity.

Set $k_2 = \log(d \log N)$ and $k_1 = \frac{\log N}{k_2}$. It follows that $d(k_1 - 1) + 1 \le q - 1$. Note that every column of M has weight exactly $q-1$. Further, as any two codewords in RS agree in at most $k_1 - 1$ positions, it follows from Lemma 2.1 that M is d-disjunct.

Each $M_{C_{\text{in}}^i}$ is $(d,1)$-list disjunct (i.e. d-disjunct) and can be decoded in $O(dq)$-time. For the outer code, we apply Theorem 2.1 with $\alpha = 1$, $\ell = d$, $r = 2$ and $s = 1$ to show that RS is $(1, d, L)$-list recoverable. Note that (2.1) is satisfied with this set of parameters because $d < (q-1)/k_1$. Thus by Theorem 2.1, C_{out} is $(d, O(qd/k_1))$-zero error list recoverable in time $\text{poly}(d, q)$. Note that $N = 2^{k_1 k_2} = q^{k_1}$, $t = q(q-1) = O(d^2 \log^2 N)$, and $qd/k_1 = d^2 \log(d \log N)$; thus, the decoding time is as claimed.

The claim on the space constructibility of the matrix follows from the fact that any position in an RS codeword can be computed in space $\text{poly}(k_1, k_2)$ and any bit value in any codeword in C_q can be computed in $O(k_2)$ space. □

Similarly, by using the $(q-1, k)_q$ RS code as the outer code and the code from Corollary C.1 as the inner code, we obtain efficient Group Testing with $O\left(d^3 \log N(\log d + \log\log N)\right)$ tests.

Corollary C.2. *Given any integers $N > d \geq 1$, there exists a $t \times N$ d-disjunct matrix M with $t = O(d^3 \log N(\log d + \log\log N))$ that can be decoded in time* $\text{poly}(t)$. *Furthermore, each entry of M can be computed in space* $\text{poly}(\log t, \log N)$.

Proof. Fix $d \geq 1$. Pick q, k and C_{out} as in the proof of Corollary C.1. Then pick C_{in} to be the concatenated code C^* corresponding to Corollary C.1 with $N = \Theta(q)$. Note that every codeword in C^* has the same weight (say w) and for every two codeword in C^*, there are at most w/d positions where both of them have a value 1. In other words, every column of $M_{\text{RS}\circ C^*}$ has Hamming weight wq and every two columns agree in at most $2wq/d$ positions, which by Lemma 2.1 implies that $M_{\text{RS}\circ C^*}$ is also $d/2$-disjunct. Further, by Corollary C.1, C^* is (d,d)-list disjunct and can be "list decoded" in time $\text{poly}(d, \log q)$. Note that for $M_{\text{RS}\circ C^*}$, we have $t = O(d^3 k \log^2 q)$ and $N = q^k$. □

The following corollary presents efficient Group Testing with $O\left(d^3 \log N\right)$ tests and is better than Corollary C.2 in terms of the number of tests, but worse in construction space complexity for non-constant d. To prove the corollary, use the RS code as the outer code and the code from Theorem 2.2 as the inner code.

Corollary C.3. *Let $d \geq 1$. There exists $t \times N$ d-disjunct matrix M with $t = O(d^3 \log N)$ that can be decoded in time* $\text{poly}(t)$. *Further, each entry of M can be computed in space* $\text{poly}(t, \log N)$.

Proof. The idea here is to use RS code as the outer code and the code from Theorem 2.2 as the inner code.

Fix $d \geq 1$. Pick q and k so that d is the largest integer with $d < \frac{q-1}{k}$. Let C_{out} be an $(q-1, k)_q$ RS code. Pick C_{in} to be the concatenated code C^* corresponding to Theorem 2.2 with $N = \Theta(q)$. Every codeword in C^* has the same weight (say w) and further, for any two codewords in C^*, there are at most w/d positions where both have a 1. Then by using the same argument in the proof of Corollary C.2 and by the choice of q and k, $M_{\text{RS}\circ C^*}$ is also $d/2$-disjunct.

Further, by Theorem 2.2, C^* is (d,d)-list disjunct and can be decoded in time $O(qd^2 \log q)$. Note that for $M_{\text{RS}\circ C^*}$, we have $t = O(kd^3 \log q)$ and $N = q^k$.

The claim on the space complexity follows from Theorem 2.2 and the fact that any position in C_{out} can be computed in space $\text{poly}(\log q, k)$. □

C.1 Proof of Lemma 4.1 Pick q and k such that d is the largest integer such smaller than $\left(\frac{q}{k}\right)^s \cdot \frac{1}{(s+1)^{s+1}}$. Let C_{out} be the PV^s code with block length q and dimension k and pick the inner code to be C_{q^s}. The required matrix corresponds to the concatenated code $C_{\text{out}} \circ C_{\text{in}}$. Note that $N = q^k$ and $t = q^{s+1}$. This along with the choice of d, implies the claimed bound on t.

The list decoding of $M_{\text{PV}^s \circ C_{q^s}}$ proceeds along obvious lines: think of the outcome vector as $r = (r_1, \ldots, r_q) \in (\{0,1\}^{q^s})^q$ and decode each r_i for C_{q^s} resulting in a set $S_i \subseteq [q^s]$ of size $|S_i| \leq d$ (the latter is because C_{q^s} is d-disjunct). We will use Theorem 2.1 with $\alpha = 1$, $\ell = d$, $r = 2$ and $n = q$. By the choice of d, (2.1) is satisfied. Thus, Theorem 2.1 implies that $M_{\text{PV}^s \circ C_{q^s}}$ is (d, L)-list disjunct for $L = O(s^{O(s)} qd/k) = O\left(s^{O(s)} d^{1+1/s}\right)$, where the latter equality follows from the choice of parameters. Decoding the inner codes takes time $O(q^{s+1})$ while decoding the outer code takes time $\text{poly}(s^{O(s)}, q, d)$. Thus, by the choice of the parameters, the total decoding time is $\text{poly}(t)$ as required.

Finally, the claim on the low space constructibility of $M_{\text{PV}^s \circ C_{q^s}}$ follows from the fact that any position in an arbitrary codeword in PV^s and C_{q^s} can be computed in space $\text{poly}(s, \log q, k)$ and $O(s \log q)$ respectively. □

D Disjunct Matrices and Expanders

Here is a connection between expanders and disjunct matrices.

Proposition D.1. *Let G be a $(n, w, t, 2, w(1 - \frac{1}{3d}))$-expander. Then M_G is a d-disjunct matrix.*

Proof. First we show that for any two columns in M_G, there are at most $\frac{2w}{3d}$ positions where both the columns have a 1. This condition is equivalent to saying that for any $i \neq j \in [n]$, $|\Gamma_G(\{i\}) \cap \Gamma_G(\{j\})| \leq 2\epsilon w$, where

$\epsilon = \frac{1}{3d}$. This condition in turn implied by the facts that $|\Gamma_G(\{i,j\})| \geq 2w(1-\epsilon)$ and $|\Gamma_G(\{j\})| = w$. Finally, note that every column in M_G has weight w. Lemma 2.1 completes the proof. □

The probabilistic method gives the following:

THEOREM D.1. *There exist* $(n, w, t, K, w(1-\epsilon))$ *expander with* $t = O(K \log N/\epsilon^2)$.

Theorem D.1 implies that there exists a d-disjunct matrix with $O(d^2 \log n)$ rows.

It is known that any d-disjunct matrix needs to have $\Omega(\frac{d^2}{\log d} \log n)$ rows [10, 11, 13]. Thus, Proposition D.1 implies the following lower bound:

COROLLARY D.1. *Let* G *be a* $(n, w, t, 2, w(1-\epsilon))$ *expander. Then* $t = \Omega\left(\frac{\log n}{\epsilon^2 \log(1/\epsilon)}\right)$.