GROUP TESTING IN STATISTICAL SIGNAL RECOVERY

A. C. GILBERT AND M. J. STRAUSS

Abstract. Over the past decade, we have seen a dramatic increase in our ability to collect massive data sets. Concomitantly, our need to process, compress, store, analyize, and summarize these data sets has grown as well. Scientific, engineering, medical, and industrial applications require that we carry out these tasks efficiently and reasonably accurately. Data streams are one type or model of massive data sets. Mathematicians, computer scientists, and statisticians have developed models, algorithms, and methods for processing data streams. One common core to all of the statistical recovery algorithms for data streams is *group testing*. We give an expository discussion of group testing, models for streaming algorithms, and how group testing arises in these algorithms. We also give an overview of three streaming algorithms and a discussion of the role of group testing in each algorithm.

1. Introduction

We are drowning in data. We have seen a dramatic increase in computational power and data-gathering mechanisms, as well as an ever-increasing demand for finer data analysis in scientific, engineering, and industrial applications. Each day, literally millions of large data sets are generated in medical imaging, surveillance, and scientific acquisition. This has led to the creation of enormously large data sets with great detail. Also, the Internet has become a communication medium with vast capacity; monitoring such networks generates data that dwarf most databases that have been considered massive in the past. The usefulness of these data sets rests on our ability to process them efficiently, whether it be for storage, transmission, visual display, fast on-line graphical query, correlation, or registration against data from other modalities.

One model of modern massive data sets is the data stream, which consists of data records or items. Data streams possess three challenging properties. They represent (either explicitly or implicity) huge amounts of data, they are distributed either spatially or temporally, and we receive or observe them at enormously high rates. These data items may be transaction records in communication networks or digital samples from analog waveforms, to name two disparate examples. See [Mut05] for more examples.

In the past ten years, researchers in computer science, mathematics and statistics have recognized that we must significantly increase our capacity to process (compress, denoise, correlate, etc.) large data sets while maintaining their inherent statistical properties. As such, they have developed stream processing computational models and sublinear algorithms to compute statistical quantities from large data sets efficiently. Mathematicians have developed new theories of information extraction to quantify and to bound the resources necessary to process these data sets. See [FKSV99, AMS96, HRR98, BYKS01, BYJKS02, AGMS99] and the references therein for a small sample of the works in this area.

Date: August 7, 2006.

Key words and phrases. Approximation, sublinear algorithms, group testing.

Gilbert is with the Department of Mathematics, The University of Michigan at Ann Arbor, 2074 East Hall, 530 Church St., Ann Arbor, MI 48109-1043. E-mail: `{annacg}@umich.edu`. ACG has been supported in part by NSF DMS 0354600 and DARPA ONR N66001-06-1-2011.

Strauss is with the Department of Mathematics and the Department of Electrical Engineering and Computer Science, The University of Michigan at Ann Arbor. E-mail: `martinjs@umich.edu`. MJS has been supported in part by NSF DMS 0354600 and DARPA ONR N66001-06-1-2011.

Traditionally, group testing is a design problem. The goal is to construct an optimally efficient set of tests of items such that the test results contains enough information to determine a small subset of items of interest. It has its roots in the statistics community and was originally designed for the Selective Service to find and to remove men with syphilis from the draft. It appears in many forms, including coin-weighing problems, experimental designs, and public health. We are interested in both the design of tests *and* the design of an efficient algorithm that works with the tests to determine the group of interest. As it happens, some of the same techniques that are useful for designing tests also help to solve algorithmic problems in analyzing and in recovering statistical quantities from streaming data. We hope that the interplay of traditional statistical techniques in conjunction with modern algorithms to solve crucial statistical problems will be of interest to statisticians generally.

The purpose of this article is twofold: (i) to give an expository description of streaming data and algorithmic models and to explain some of the algorithms for computing statistics on streaming data and (ii) to explain the role of group testing in these streaming algorithms. One of the main mathematical tools in sublinear algorithms is group testing and its role is not frequently highlighted in these algorithms.

We begin our discussion in Section 2 with a review of combinatorial group testing, including several examples and constructions from error correcting codes which are a leitmotif in our exposition. We base this brief discussion upon several chapters in [DH93, MS77] and encourage interested readers to look there for more information. Next, in Section 3, we discuss several models for streaming data, including the algorithmic models for processing such data streams. We connect streaming algorithms and group testing through these algorithmic models. In Section 4, we give an approximate group testing algorithm which is at the core of many streaming algorithms. This algorithm may be of interest for the sake of group testing alone and it may find uses in the many other applications which use group testing, so we present it in its own section. We proceed in Section 5 to explain several streaming or sublinear algorithms which use certain types of constrained group testing designs. Despite (or perhaps because of) these constraints, we obtain extremely efficient and novel data processing algorithms. Finally, in Section 7, we give novel uses of the results of group testing—applications that do not simply identify items of interest in a large group but, instead, allow us to compute statistical quantities of interest. We refer to these problems as statistical signal recovery as we are interested in recovering statistics about an underlying signal from a stream of data records.

2. Group Testing Preliminaries

2.1. **History of Group Testing.** We can trace the origins of group testing to World War II. Two economists, Robert Dorfman and David Rosenblatt, with the Research Division of the Office of Price Administration created a efficient method for detecting draftees with syphilis. Dorfman's original report [Dor43] in the Annals of Mathematical Statistics describes a method for removing men from Selective Service with syphilis. They designed the testing procedure for the United States Public Health Service and the Selective Service System, neither of which wanted men serving in the military who were ill and who might spread to a civilian population a diseast that was untreatable (at that time).

The ideas behind the testing method are relatively simple. The Selective Service System was to draw blood from each draftee. Then they would pool the samples into groups of five. They would use a single Wassermann test [WNB06] on the pooled samples to test for the presence of the syphilitic antigen (which indicates a likely syphilis infection). The efficiency in the design came from two key observations.

(1) If no one in the pool has the antigen, then the pool does not, and the test is negative.

(2) If one or more people in the pool have the antigen, then the pooled samples have the antigen. The test is positive and we can then test all five members of the pool individually to determine who is infected.

If no one in the pool has the antigen, then we save four tests as compared with five tests for testing all five individuals from the outset. If one or more in the pool have the antigen, then we waste a single extra test on the entire group.

The group-testing concept languished for several decades after Dorfman's report. The Selective Service System did not put group testing for syphilis into practice because the Wassermann tests were not sensitive enough to detect antigens in a sample diluted by contributions of more than eight or nine healthy people. The ideas remained an interesting exercise in Feller volume I [Fel50] until Sterret wrote a Ph.D. thesis *On the detection of defective members of large populations* [Ste57] in 1957. Sobel and Groll [SG59] picked up the ideas, again in an industrial epxerimental design application, in 1959. Since then the ideas and procedures have been rediscovered, used, generalized, and expanded upon in a diverse collection of areas, including theoretical computer science, coding theory, graph theory, public health, image coding, statistics, and game theory to name a few.

2.2. Definitions and examples.

We define the basic problem in the simplest form. We have a universe of N items in total. We know that d of them are defective and we call this set $\mathcal{D}$ the defective set. This defective set must be a member or sample of a given family we call the sample space. The sample space $S(d, N)$ might, for example, be all subsets of N items of size d. The goal of combinatorial group testing is to construct a collection of tests (called a *design*) to minimize the number of tests needed to find the defective set for the worst case input. We call a best algorithm under this goal a minimax algorithm. At this level, we do not specify the type of tests. They may be adaptive or non-adaptive and they may be linear or nonlinear. Below we will consider several ways in which randomness enters a generalized definition.

We begin with a simple but fundamental setting in combinatorial group testing, that of binary non-adaptive tests. Let $\boldsymbol{M}$ be a $t \times N$, $\{0,1\}$-valued matrix we call the measurement matrix. Let R_i denote the ith row or ith group of $\boldsymbol{M}$. We regard columns C_j of the $\{0,1\}$-valued matrix $\boldsymbol{M}$ as subsets (of the universe of t tests). The entry $\boldsymbol{M}_{ij} = 1$ if item j is in group i and $\boldsymbol{M}_{ij} = 0$ if item j is not in test i. Let $s \in \{0,1\}^N$ be the characteristic vector for our set of items, so s has exactly d entries that are 1 and $(N - d)$ zeros. We apply $\boldsymbol{M}$ to s and collect *measurements*, or the results of our tests, in a vector v,

$$\boldsymbol{M}s = v,$$

where the arithmetic is *boolean*, that is, multiplication of 0's and 1's is the usual multiplication (which coincides with the logical AND) but addition is replaced by the logical OR. Equivalently, we peform the multiplication $\boldsymbol{M}s$ over $\mathbb{Z}$ and then replace all non-zero entries in the result by 1. This corresponds to *disjunctive* tests (such as for syphilis) where a test on a group i is positive or negative, and is positive if and only if at least one item in group i would, by itself, lead to a positive test.

We have the following definitions.

Definition 1. *The measurement matrix $\boldsymbol{M}$ is d-separable if the ORs (or Boolean sums) of any d columns are all distinct.*

Definition 2. *The measurement matrix $\boldsymbol{M}$ is d-disjunct if the ORs of any d columns do not contain any other column.*

Clearly, the d-disjunct condition is a stronger one than d-separability. If a matrix is d-disjunct, then it is also d-separable.

The group testing algorithms that accompany these two types of binary matrices are also relatively straightforward. Let $S(d, N)$ consist of all subsets of size d of N items total. Then a set of d columns chosen from $\boldsymbol{M}$ correspond to a particular sample s in $S(d, N)$ or a particular d-subset.

The union of these d columns is the set of tests which generate positive outcomes for this d-subset. For a d-separable matrix, we need only match the set of positive tests to samples in $S(d, N)$ and we have identified the set of defective items. A straightforward algorithm, however, keeps a look-up table of size $\binom{N}{d}$, which may be prohibitive. On the other hand, for a d-disjunct matrix, the defective items are easily identified in time $O(Nt)$. Let $\mathcal{S}$ be the set of columns made up by the sample and let

$$P(\mathcal{S}) = \bigcup_{j \in \mathcal{S}} C_j$$

be the set of positive tests. If item j is in some negative test (*i.e.*, item j is in some group R_i for which the i'th test result is negative), then identify j as a good item; otherwise, identify j as a defective item. To see that this procedure is correct, note that, if j is a good item, then, by definition of d-disjunctness, $P(\mathcal{S})$ does not contain C_j, so j is in some negative test. On the other hand, if j is a defective item, then it clearly is in no negative test, since the tests are disjunctive.

Let us give two examples of d-separable and d-disjunct matrices. Both of these examples recur in the sections that lie ahead. We start with d-separable matrices. We recall a few basic definitions of binary error correcting codes as they also recur in the sections ahead. We use a binary code to transmit a block of k binary message symbols $u_1, u_2, \ldots, u_k$ as an encoded binary message $x_1, x_2, \ldots, x_N$ of length N. Each symbol u_i and x_i are either 0 or 1. The codewords $x = x_1 \cdots x_N$ are our binary code. Codes were invented to correct the errors that occur when we transmit messages on noisy channels. When we transmit a symbol $x_i = 0$ on a noisy channel, we usually receive it as a 0 but it might be changed to 1 (and, similarly, a 1 may be changed to a 0). To generate a binary linear code, we employ a generator matrix G.

Definition 3. *Given any binary matrix G, a binary linear code C is the span of the rows of G. That is, codewords $w \in C$ are of the form*

$$w = c_1 g_1 + c_2 g_2 + \cdots + c_k g_k = cG$$

where $c_i \in \mathbb{Z}_2$ and G is a $k \times N$ matrix with rows $g_1, \ldots, g_k$. The dimension of the code over $\mathbb{Z}_2$ is k and the length is N.

To clarify, all of the arithmetic for binary codes is carried out over $\mathbb{Z}_2$. We frequently design codes so that part of the codeword consists of the message and the other symbols are check symbols which we use to detect and to correct errors in the received codeword.

Definition 4. *Given a binary linear code C of length N and dimension k over $\mathbb{Z}_2$, the parity check matrix H of C is an $N \times (N-k)$ matrix whose rows generate the orthogonal complement of C. An element $w \in \mathbb{Z}_2^N$ is in C if and only if*

$$wH = 0.$$

Equivalently, the rows of H generate the nullspace of the generator matrix G.

A binary d error correcting code of length N is a binary code which can detect and correct up to d errors. Equivalently,

Lemma 5. *Sums, modulo two, of up to d columns of the transpose of the parity check matrix of the code are distinct.*

Proof. See [MS77] for a short proof of the fact that for a d error correcting code, every $2d$ columns of the transpose of the parity check matrix are linearly independent. For a binary code, this is equivalent to the claim. □

We can convert this property into a d-separability property for a binary matrix H^{tr} by replacing, in each column, various patterns of 0s and 1s with longer binary patterns. For $d = 2$, for example, replace

$$0 \mapsto \tau(0) = \begin{pmatrix}0\\1\end{pmatrix} \quad \text{and} \quad 1 \mapsto \tau(1) = \begin{pmatrix}1\\0\end{pmatrix}.$$

Observe that the modulo-2 addition table is transformed to the Boolean addition table with the

$\bigoplus_2$	0	1
0	0	1
1	1	0

$\bigvee$	$\begin{pmatrix}0\\1\end{pmatrix}$	$\begin{pmatrix}1\\0\end{pmatrix}$
$\begin{pmatrix}0\\1\end{pmatrix}$	$\begin{pmatrix}0\\1\end{pmatrix}$	$\begin{pmatrix}1\\1\end{pmatrix}$
$\begin{pmatrix}1\\0\end{pmatrix}$	$\begin{pmatrix}1\\1\end{pmatrix}$	$\begin{pmatrix}1\\0\end{pmatrix}$

FIGURE 1. Transformation of mod-2 arithmetic to boolean arithmetic.

replacement of Figure 1. So, if two pairs of columns have different sums modulo two, then they have different Boolean sums after transformation.

Next, we construct a family of d-separable designs using a particular family of binary error correcting codes. BCH codes are binary d-error correcting codes and we can transform their parity check matrices into d-separable matrices. For $d = 2$, we can use the 2-error correcting BCH code and the transformation above. For this code of length $N = 2^m - 1$, the parity check matrix is a $2m \times 2^m - 1$ binary matrix with distinct sums of each pair of columns over $\mathbb{Z}_2$. Upon transformation, we get a 2-separable matrix of size $4m \times 2^m - 1$. Note that only for $m \geq 5$ do we have fewer tests than items.

The simplest example when $d = 1$ of a d-disjunct matrix (and a d-separable matrix) is the analogous transformation of the *bit test matrix*, $\boldsymbol{B_1}$. To form the bit test matrix $\boldsymbol{B_1}$, set the ith column of $\boldsymbol{B_1}$ equal to the binary representation of i, for $i = 0, \ldots, \log_2 N - 1$. In this case $t = \log_2 N$ and this construction achieves the lower bound for the minimum number of tests to locate a single item from among N. For $N = 8$, the bit test matrix $\boldsymbol{B_1}$ has the form

$$\boldsymbol{B_1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

This matrix is 1-separable, since all columns are distinct. It follows that the transformed matrix is 1-disjunct, since, looking at the transform τ, it is clear that if c and c' are unequal columns, then neither of $\tau(c)$ and $\tau(c')$ contains the other.

We note that there is a simpler and faster algorithm for detecting a single defect, using the untransformed matrix $\boldsymbol{B_1}$. For example,

$$\boldsymbol{B_1 s} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0\\0\\1\\0\\0\\0\\0\\0 \end{pmatrix} = \begin{pmatrix} 0\\1\\0 \end{pmatrix}.$$

In general, the bit-test matrix times the signal gives location in binary of the defect.

We observe that all of the tests we describe are *linear* and *non-adaptive*. The tests consist of applying a matrix (hence a linear procedure over $\mathbb{Z}_2$, $\mathbb{C}$, or under Boolean arithmetic, depending on the application) to a collection of items represented by a vector of length N. The tests are nonadaptive in the sense that the membership in each group (row of the matrix) does not depend on the outcome of any other test—indeed, there is no natural order in which to perform the tests. Nonadaptive tests are useful when we prefer to perform all the tests simultaneously or with at least some overlap in time. This is the case if the tests take a long time and/or N is so large that we cannot easily store all N items between tests. Both of the situations arise the case in massive streaming data sets.

2.3. **Randomness.** Randomness traditionally arises in *probabilistic group testing*, as opposed to combinatorial group testing. In this setting, there is a distribution on input signals and a design must identify the defective set with high probability over the inputs. In streaming algorithms and in the application of group testing to massive data sets, randomness plays other roles. It is frequently difficult to obtain an accurate or even an approximate distribution on the input signals—in fact, that is frequently the problem we seek an efficient algorithm for!

We use randomness to construct a design or measurement matrix $\boldsymbol{M}$. Rather than specifying a distribution on the input signals and using randomness to draw an input from that distribution, we use a randomized construction for $\boldsymbol{M}$ and consider its worst-case performance on all possible inputs. In constructing $\boldsymbol{M}$, we ask that it satisfy one of two properties. First, we might ask that, for *each* input s (a set of d defective items from a universe of size N), $\Pr(\boldsymbol{M}$ fails on $s)$ be low. Alternatively, we might ask for a stronger condition, namely, that $\Pr($for some s, design $\boldsymbol{M}$ fails on $s)$ be low. In some contexts, one can achieve the "for-each" guarantee with fewer tests than the latter "for-all" guarantee, or achive the "for-each" guarantee when "for-all" is not possible at all.

The two properties highlight two different roles for randomness in group testing and streaming algorithms. If a "for-each" guarantee holds when a "for-all" guarantee does not, this means an adversary sees the *distribution* on designs $\boldsymbol{M}$ but not the *outcome* $\boldsymbol{M}$ itself, and tries to come up with a bad sample s. In such situations, it is often easy for an adversary given access to the *outcome* $\boldsymbol{M}$ to come up with a bad s. Thus the randomness is used in an essential way to defeat the adversary. By contrast, in a "for-all" guarantee, the randomness is used to construct a design efficiently; equivalently, to prove using a form of the probabilistic method that a design exists. Given enough time, one could exhaustively consider and test each possible design on all samples and find one that works. Frequently, we cannot carry out the procedure to test designs in polynomial time—a straightforward procedure would take time exponential in the matrix size—so a randomized construction is a substantial improvement over the straightforward construction process.

There are two observations that relate the two roles of randomness in group testing and streaming algorithms. These observations also hold in more general contexts of randomized algorithms. First, suppose that we are are looking for a set of d defective items out of N total. Then our sample space $S(d, N)$ consists of all d subsets and we have $\binom{N}{d}$ possible input signals. If the failure probability in a "for-each" guarantee is small enough, *i.e.*, less than $\binom{N}{d}$, then we can take a union bound over all $\binom{N}{d}$ samples and get a non-trival "for-all" guarantee. In other words, an exponentially small "for-each" failure probability is small enough to yield a "for-all" guarantee. The second observation connects randomized design constructions with the more traditional probabilistic group testing. Yao's principle derived from the minimax theorem for two-player games implies that there is a randomized design of size t with a "for-each" guarantee if and only if for each possible distribution on inputs there is a fixed design of size t that works.

3. Streaming Algorithms

There are many situations in which data arrive and are processed in a stream. For example, network service providers collect logs of network usage (telephone calls or IP flows) in great detail from switches and routers and aggregate them in data processing centers. They use this data for billing customers, detecting anomalies or fraud, and managing the network. In most cases, we cannot accumulate and store all of the detailed data. We can archive past data but it is expensive to access these data. We would rather have an approximate, but reasonably accurate, representation of the data stream that can be stored in a small amount of space. It is not realistic to make several passes over the data in the streaming setting. It is crucial that we compute the summary representation on the stream directly, in one pass.

3.1. **Streaming Data Models.** Our input, which we refer to as the *stream*, arrives sequentially, item by item, and describes an underlying *signal*. In the simplest case, the signal $\mathbf{A}$ is a one-dimensional function $\mathbf{A} : [1 \dots N] \to \mathbb{Z}^+$. We assume that the domain is discrete and ordered and that the function $\mathbf{A}$ maps the domain to non-negative integers. For example, a signal is the number of employees in different ages (the domain is the set of ages and the range is the number of employees of particular age), or the number of outgoing call minutes from a telephone number (domain is the set of all telephone numbers and the range is the total number of outgoing minutes). For signals over a continuous domain, we assume that the domain is discretized in a sensible fashion.

The stream may describe the underlying signal in one of many ways, but we will focus on one of the more general models, the *cash register* model. In the cash register model, the items that arrive over time are domain values in an arbitrary order, and the function is represented by implicitly aggregating the number of items with a particular domain value. For example, a stream of telephone call records could be:

$$\begin{gathered}\langle 8008001111, 10\rangle, \langle 8008002222, 15\rangle, \langle 8008003333, 13\rangle, \\ \langle 8008001111, 23\rangle, \langle 8008001111, 3\rangle \dots\end{gathered}$$

where each record contains the phone number and the length of the outgoing telephone call. We construct the underlying signal, namely $\mathbf{A} = \langle 8008001111, 36\rangle$, $\langle 8008002222, 15\rangle$, $\langle 8008003333, 13\rangle$ by aggregating the total number of minutes outgoing from numbers 8008001111, 8008002222, 8008003333, etc. This model is called the cash register model because the flow of sales through a cash register in a store generates a stream in this model. More generally, we may consider transactions that also "subtract" from the underlying data distribution. This arises, for example, in a comparison between today's and yesterday's network traffic, as well as in other applications where subtractions naturally occur.

3.2. **Stream Processing Model.** Next we focus on how to process the input data stream. We read and process each data item as it arrives. We cannot backtrack on the stream nor can we access explicitly arbitrary past items. We are allowed a certain amount of auxiliary memory. We may use this, for example, to store a recent window of items we have read or some arbitrary subset of items in the past, or other summary information about the past data stream. The size of this auxiliary memory is crucial as it determines what computations we can perform on the data stream. For most streaming applications, the auxiliary memory is significantly smaller than the signal domain size. Hence, the signal can only be partially represented in the auxiliary memory as data items continue to arrive. We abstract these practical constraints to a streaming data computational model which we illustrate in Figure 2. Two key performance parameters of streaming algorithms are the time needed to process each item on the stream and the auxiliary storage used. For most applications, our goal is to keep both as small as possible.

3.3. **Streaming algorithms and group testing.** To illustrate the connection between streaming algorithms and group testing, let us start with an example. Suppose that our stream of phone call records includes records of the form

$$\langle \texttt{time.stamp}, \texttt{phone.number}, \texttt{flag}\rangle$$

where $\texttt{flag} = 1$ or -1. This type of record indicates the beginning of a phone call (with $\texttt{flag} = 1$) from a phone number at a certain time. The record with the same phone number but with $\texttt{flag} = -1$ at a later point in time indicates the end of that phone call. Our stream might look like

$$\begin{gathered}\langle 1200, 8008001111, 1\rangle, \langle 1203, 8008002222, 1\rangle, \langle 1204, 8008002222, -1\rangle, \\ \langle 1206, 8008003333, 1\rangle, \langle 1207, 8008001111, -1\rangle, \langle 1208, 8008004444, 1\rangle, \dots\end{gathered}$$

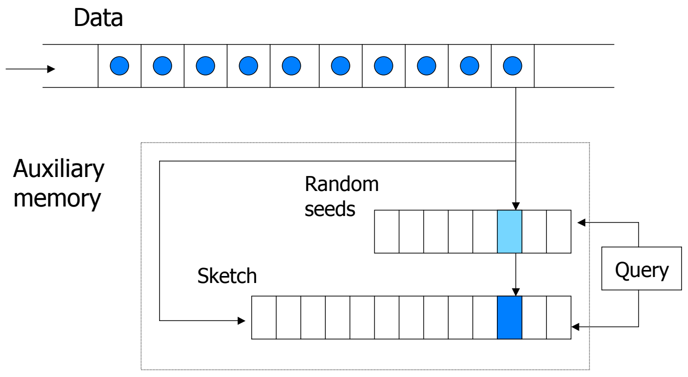

FIGURE 2. Data stream processing computational model.

The underlying signal that we construct from these records is the list of currently active phone numbers (those phone numbers in the midst of a telephone call). After the sixth record in the stream, our signal is

$$\mathbf{A} = \langle 8008001111, 0\rangle, \langle 8008002222, 0\rangle, \langle 8008003333, 1\rangle, \langle 8008004444, 1\rangle;$$

that is, of all the phone numbers seen thus far, two of them are currently making a phone call (signified with a 1), while the other two are not (signified with a 0). More formally, our signal $\mathbf{A}$ is a one-dimensional function with discrete domain equal to the set of all phone numbers and with range $\{0, 1\}$. Let N be the number of all possible phone numbers. Let us further suppose that at the time we analyze the stream, there are d active phone calls. Then our signal $\mathbf{A}$ has d non-zero entries out of N total and the problem of identifying which phone numbers are currently active is equivalent to identifying d defective items out of N total.

Not only is our fundamental problem of identification equivalent to that of group testing, but the severe resource constraints in place in stream processing necessitate as few tests as possible to identify those phone numbers. Ideally, we would make $O(d \log N)$ such tests. As this design problem is at the core of combinatorial group testing, it is natural to mine the previous substantial literature for potential streaming algorithms or subroutines for such algorithms. In fact, these algorithms go much further than simple applications of existing work.

For streaming algorithms, we are especially interested in linear non-adaptive measurements or tests that arise in certain combinatorial group testing models. Suppose that we have a measurement matrix $\boldsymbol{M}$ which we apply to our signal $\mathbf{A}$ to obtain a set of measurements v, $\boldsymbol{M}\mathbf{A} = v$ (sometimes referred to as the *sketch* of the signal $\mathbf{A}$). We use these measurements or sketch v to determine the identity of the d currently active phone numbers. Now, suppose that we see two additional records, consisting of a termination record for a currently active phone call and an initiation record for a new active phone call. The total number of currently active calls remains unchanged at d and we simply update our implicit signal $\mathbf{A}$ as $\mathbf{A} + \Delta\mathbf{A}$ to reflect this change. In this case, $\Delta\mathbf{A}$ decrements to 0 the counter for the terminated call and increments to 1 the counter for the newly initiated call. We can still use the same measurement matrix $\boldsymbol{M}$ to identify the new set of d active phone numbers as the new signal still has only d active phone numbers. What's more, our measurements change from $\boldsymbol{M}(\mathbf{A})$ to $\boldsymbol{M}(\mathbf{A} + \Delta\mathbf{A}) = \boldsymbol{M}(\mathbf{A}) + \boldsymbol{M}(\Delta\mathbf{A})$. Upon receiving two new records, we need only compute $\boldsymbol{M}(\Delta\mathbf{A})$ and update $\boldsymbol{M}(\mathbf{A})$, our previous measurements, accordingly. The additional resource constraints in streaming algorithms dictate how quickly we must compute and update the measurements $\boldsymbol{M}(\Delta\mathbf{A})$ as well as the efficiency of the identification algorithm. We leave these algorithmic discussions for the following sections.

While this example is overly simplistic for the sake of exposition, it does highlight the algorithmic challenges we face in the stream processing model. In order to understand the challenge of maintaining information about currently active phone calls with few resources, let us consider the following example. A transaction stream consists of one million initiations of phone calls followed by 999,996 terminations, leaving $d = 4$ active phone calls. Our goal is to output those four active phone numbers, using an algorithm that is allowed not much more space than needed to store four phone numbers. In other words, we must recover the data set exactly (perhaps only with high probability). A space-efficient algorithm knows very little about a data set of size one million and it does not know which 999,996 points will be deleted. The algorithm cannot retain the memory of 4 special values. Nevertheless, algorithms we present below will be able to determine the four values, and in time polynomial in 4 and $\log(1,000,000)$. Although this is a contrived example, it illustrates the difficulty with maintaining signal statistics in the face of deletes which dramatically change the data distribution.

We note that in our previous discussion our tests were disjunctive, as is common in medical testing. In that case, we converted tests over $\mathbb{Z}$ or $\mathbb{Z}_2$ (e.g., traditional binary error correcting codes) into a matrix under Boolean arithmetic. The simple example above illustrates our need for tests that are linear over $\mathbb{Z}$ (or $\mathbb{C}$, more generally). The first record we see might be $\langle 8008003333, -1\rangle$ which terminates an active phone call. We update $\mathbf{A}$ implicitly by subtracting 1 from index 8008003333. We must update our measurements $\boldsymbol{M}(\mathbf{A})$ to reflect this subtraction and we can do so linearly if our tests are linear over $\mathbb{Z}$; i.e., we carry out our arithmetic over $\mathbb{Z}$ rather than using Boolean arithmetic. For the remainder of our discussion, we use $\mathbb{Z}$- or $\mathbb{C}$-linear measurements.

Let us also observe that we rarely observe signals which are guaranteed to have d nonzero entries, exactly equal to 1, and $N - d$ zeros. Our streaming algorithms must operate on much more general signals while the group testing models are more restrictive. We can, in fact, generalize our group testing models to *additive models* where the ith defective item x_i has defectiveness measure θ_i. In this model, the outcome of a test on a group G is the sum $\sum_{x_i \in G} \theta_i$. Du and Hwang [DH93] observe that not much is known about these models for $d \geq 3$ so our discussion of streaming algorithms may be of independent interest for group testing algorithms. Let us also note that our signals models are richer than those in the additive group testing model as the nondefective items may also have a non-zero measure.

4. Approximate Identificiation

Above, we defined the goal of traditional group testing as identification of a set of d defects using a minimal number of tests. We want to construct a design that works on all sets (or on each set) of d defects. There are many situations in which we may be willing to relax our goals. Rather than identifying all d defects from a single set of measurements or from a single use of those measurements, we may be able to identify all d defective items in several iterations, using the test and measurements repeatedly. Alternatively, we may be satisfied with recovering a large fraction of the defective items. Let us formalize this approximation problem as *ϵ-approximate identification.* That is, for all samples s consisting of d defects, we want our design to output a set of items that contains at least ϵd of the defects. Thus we allow at most $(1 - \epsilon)d$ false negatives. (We do not include the typical bound on the number of allowable false positives as it is usually implied by an upper bound on the runtime of the algorithm.) This definition is useful because it is often much easier to construct a design of prescribed size that *approximately* identifies defects than a design that (strictly) identifies defects.

Let us construct a random matrix $\boldsymbol{R}$ with $t \approx d \log N$ rows and N columns. We place exactly one 1 per column, in a row chosen uniformly at random. This random construction corresponds to assigning each of d defective items at random to one of t groups. The non-defective items are also assigned to groups, but this is irrelevant. In an alternative and essentially equivalent construction, each entry of $\boldsymbol{R}$ is 1 with probability approximately $1/t$ and 0 otherwise, independently. We will give a proof for this construction, rather than the first, as it is easier to analyze.

We say that defective item j is isolated in group i (or by measurement i), if item j and no other defective items are assigned to group i. We say that a matrix $\boldsymbol{R}$ isolates q items if there are q distinct indices j that are isolated in a group given by a row of $\boldsymbol{R}$.

Lemma 6. *Fix parameters d and N. Let $\boldsymbol{R}$ be a matrix with N columns and t rows, such that each entry is one with probability $1/d$ and zero otherwise, independently. For sufficiently large $t \leq O(d\log(N))$, except with probability $\frac{1}{4}$, the matrix $\boldsymbol{R}$ isolates at least ϵd of the defects in any signal s consisting of δ ones (defective items), for $d/2 \leq \delta \leq d$, and $N-\delta$ zeros (nondefective items).*

Proof. Fix a sample with δ defects. In each row, the probability of getting exactly one defect is $p = \delta(1/d)(1-1/d)^{\delta-1}$, so that $p \geq \Omega(1)$. By the Chernoff bound, for some constant c, at least ct of the rows get exactly one defect except with probability $e^{-\Omega(t)} \leq \frac{1}{4}\binom{N}{d}^{-1}$, for sufficiently large $t \leq O(d\log(N))$. By symmetry, the set of isolated defects consists of t independent uniform draws *with replacement* from the set of δ defects. The probability that the collection contains fewer than ϵd *different* items is at most

$$\begin{aligned}
\sum_{j<\epsilon d}\binom{d}{j}(j/\delta)^t &\leq \epsilon d\binom{d}{\epsilon d}(2\epsilon)^t \\
&\leq \epsilon d\frac{d^{\epsilon d}}{(\epsilon d)!}(2\epsilon)^t \\
&\approx \epsilon d\frac{d^{\epsilon d}}{(\epsilon d/e)^{\epsilon d}}(2\epsilon)^t \\
&= \epsilon d\left(\frac{e}{\epsilon}\right)^{\epsilon d}(2\epsilon)^t.
\end{aligned}$$

If we make $\epsilon = e^{-1}$, we get

$$\begin{aligned}
\sum_{j<\epsilon d}\binom{d}{j}(j/d)^t &\leq \epsilon d\left(\frac{e}{\epsilon}\right)^{\epsilon d}\epsilon^t \\
&\leq e^{\ln(d)-1+2d/e-t(1-\ln(2))} \\
&\leq \frac{1}{4d}\binom{N}{d}^{-1},
\end{aligned}$$

provided $t \leq O\left(\log\binom{N}{d}\right) = O(d\log N)$ is sufficiently large. Finally, take a union bound over all at-most-$d\binom{N}{d}$ possible signals. □

Now we extend to "at most d" defects.

Corollary 7. *Fix parameters d and N. For sufficiently large $t \leq O(d\log(N))$, there is a t-by-N matrix $\boldsymbol{M}$ that isolates at least $\epsilon\delta$ of the defects in any sample s having $\delta \leq d$ ones (defects) and $N-\delta$ zeros.*

Proof. By Lemma 6, there exists a matrix with $O(d\log(N))$ rows that works for any δ in the range $d/2 \leq \delta \leq d$. Similarly, for any j, there is a matrix $\boldsymbol{R_j}$ with $O(2^{-j}d\log(N))$ rows that works for any δ in the range $2^{-(j+1)d} \leq \delta \leq 2^{-j}d$. Combine all these matrices. The number of rows is the sum of a geometric series,

$$O(d\log(N))(1+2^{-1}+2^{-2}+\cdots) = O(d\log(N)).$$

□

Suppose that defective item j is isolated in group i. Then no other defective items are in group i but this by itself does not identify j. The signal which consists of all those items in group i does, however, consist of $d' = 1$ defective items only and we can apply the bit-testing matrix $\boldsymbol{B_1}$ to those of the N items that end up group i; this *will* identify j. More precisely, for each row r in the random design $\boldsymbol{R}$ and each row r' of $\boldsymbol{B_1}$, take rr' as a row of our final design, $\boldsymbol{M}$. Thus $\boldsymbol{M}$ has approximately $d\log^2(n)$ rows and we say that $\boldsymbol{M}$ is the row tensor product of B_1 and $\boldsymbol{R}$, $\boldsymbol{M} = \boldsymbol{B_1} \bigotimes_r \boldsymbol{R}$. If the random part of $\boldsymbol{M}$ succeeds, *i.e.*, group r has exactly one defect j, then it is clear that $\boldsymbol{M}$ will identify j. If group r contains more than one item, however, the natural bit-test algorithm may fail and output an arbitrary position; these are false positives. Some items will never appear alone in any group; these are false negatives. Thus the number of false positives is bounded by the number of groups, $O(d\log(N))$. The number of groups is close to the lower bound, since $\log\binom{N}{d} \geq \Omega(d\log(N/d))$. We summarize our above discussion in the following theorem.

Theorem 8. *Suppose there is a Boolean testing procedure that can be applied to any subset of items. Then there is a measurement matrix $\boldsymbol{M}$ with $O(d\log^2 N)$ rows and N columns and non-adaptive algorithm which identifies a list of $O(d\log N)$ items containing at least ϵd defective items from N total,* i.e.*, an ϵ-approximate identification design. The matrix $\boldsymbol{M}$ that is a row tensor product of the bit-test matrix $\boldsymbol{B_1}$ and a random matrix $\boldsymbol{R}$ of $O(d\log N)$ rows, having exactly one non-zero, a 1, in each column is such a matrix, with high probability. Excluding time to make measurements, the algorithm runs in time* $\mathrm{poly}(d\log N)$.

Furthermore, approximate identification is almost as good as strict identification when the tests are adaptive or the tests are non-adaptive and linear. In this section we consider only adaptive tests, but we return to non-adaptive linear tests over the next sections. After tentatively identifying the defects, we can test each one to confirm, remove them from consideration, and test the remaining $N - \epsilon d$ of the items, of which $(1-\epsilon)d$ are defective. By the above discussion, there is an ϵ-approximate identification design for all relevant values of d and N, so we can repeat the process, and recover all the items. We give the following corollary for the noiseless case, but the techniques will actually tolerate considerable noise; *i.e.*, the techniques handle a sample vector s in which supposed "zeros" are actually small-magnitude non-zero numbers.

Lemma 9. *Suppose there is an ϵ-approximate identification scheme with $t \leq O(d\log^2 N)$ tests for d defects out of N items, for all relevant values of ϵ, d, and N. Then there is an adaptive design and algorithm that finds all d defects using $O(t/\epsilon)$ tests and $O(\log(d)/\epsilon)$ rounds of adaptivitiy. Excluding time to make measurements, the algorithm runs in time* $\mathrm{poly}(d\log N)$ *times the time required to call the ϵ-approximate identification algorithm once.*

Proof. Use the hypothesized ϵ-approximate identification scheme on the original (N,d)-sample, test the tentatively identified items, then use the hypothesized ϵ-approximate identification scheme on the residual $(\leq N-\epsilon d, \leq (1-\epsilon)d)$-sample, etc. Note that the total number of tests is the sum of a geometric series,

$$t(1 + (1-\epsilon) + (1-\epsilon)^2 + \cdots) \approx t/\epsilon.$$

The number of terms in the series, which is the number of rounds of adaptivity, is $O(|\log_{(1-\epsilon)} d|) \leq O(\log(d)/\epsilon)$. □

Thus we have, from the above:

Corollary 10. *Suppose there is a Boolean testing procedure that can be applied to any subset of items. Then there is an algorithm that finds all at-most-d defects in a sample of length N using at most $O(d\log^2 N)$ tests. The overall algorithm needs $O(\log(d))$ rounds of adaptivity and succeeds for all samples. Excluding time to make measurements, either algorithm runs in time* $\mathrm{poly}(d\log N)$.

5. Constrained Tests

In this section, we will consider designs whose groups all must be rows of the Hadamard matrix, described below. There are two reasons for considering this. First, one can imagine situations in nature where measurements are naturally constrained to come from a particular set. For example, in Magnetic Resonance Imaging or Synthetic Aperture RADAR, available measurements are dot products with rows of the Discrete Fourier Transform matrix. The Hadamard matrix is a relative of the DFT matrix and many of the techniques we describe for Hadamard extend to DFT, though a complete treatment of DFT is technical and beyond the scope of this paper. Second, as we will discuss below, we can dualize the result—instead of making tests that are Hadamard (or Fourier) rows to identify a set of exactly d defects that are individual items, we can equivalently make tests to a signal x that measure individual positions in x (i.e., sample x) in order to identify a set of exactly d Hadamard (or Fourier) rows that have non-zero dot product with x. In this section and the next, we claim only a "for-each" guarantee of success. Our treatment of this material is an expository presentation of the work in [GL89, KM91, GGI$^+$02b]. We observe that all of these linear measurements are appropriate for streaming data so that we may view this special group testing algorithms as streaming algorithms as well.

The importance of the DFT in modern digital signal processing, computational statistics, and analysis cannot be overstated. It is at the heard of many computational methods. We given an overview of several algorithms which identify significant Fourier (or Hadamard) coefficients. These coefficients capture the major time-invariant or stationary wave-like features of the signal. The largest few Fourier coefficients are useful in data compression, feature extraction, finding approximate periods, and data mining.

5.1. **Background on the Hadamard Matrix.** Fix integer n and let $N = 2^n$. The $N \times N$ Hadamard matrix H_n takes the value $(-1)^{\langle i,\ j\rangle}$ at position (i, j), where we regard i, j, and the inner product over $\mathbb{Z}_2^n$. The Hadamard matrix is a symmetric matrix and, properly normalized, it is unitary, so that H_n^2/N is the identity matrix.

An alternative view of the general Hadamard matrix H_n is the n-fold tensor product of the 2-by-2 matrix H_1 with itself. By associativity of tensor products, we can also write $H_n = H_j \otimes H_{n-j}$ for any j in the range $0 \le j \le n$.

Below, we will be interested in measuring a sample s with a row R_i of the Hadamard matrix. As described above, each such measurement consists of $+1$'s and -1's. We can equivalently make measurements of all 0's and 1's, as is traditional. If we intend to measure x as $\langle R_i,\ x\rangle$ for some row R_i of ± 1's, we replace R_i with $(R_i + \mathbf{1})/2$, which is a vector of 0's and 1's. We also make the measurement $\langle \mathbf{1},\ x\rangle$, where the constant 1 vector $\mathbf{1}$ also a $\{0, 1\}$-valued measurement. At recovery time, we can form $\langle R_i,\ x\rangle = 2\,\langle (R_i + \mathbf{1})/2,\ x\rangle - \langle \mathbf{1},\ x\rangle$.

5.2. **Permute and Assign.** We use a special case of the random hashing of Section 4. We view that section as follows. Instead of assigning the N positions into t groups at random, we equivalently *permute the the N positions at random, then assign to t groups arbitrarily.* Further, as we show, we only need to permute the N positions *pairwise independently* at random, *i.e.*, given distinct positions j and j', a random permutation σ sends the pair (j, j') to a pair $(\sigma(j), \sigma(j'))$ uniformly distributed over all $N(N-1)$ possibilities.

In the remainder of this section, we examine how to accomplish both the permutation and group assignment tasks in a way that is compatible with making only Hadamard measurements. We also need to accomplish bit testing using Hadamard, but that task is similar to the group assignment task.

5.2.1. *Permutation.* If a measurement is a row R_i of the Hadamard matrix, then $\langle R_i,\ x\rangle = \sum_j (-1)^{\langle i,\ j\rangle} x_j$, where the inner product in the exponent is over $\mathbb{Z}_2^n$. We will permute the j's according to $k = \sigma(j) = Pj + \ell$, where P is an invertible n-by-n matrix and ℓ is a vector in $\mathbb{Z}_2^n$. We have:

Lemma 11. *Let R_i be a row of the Hadamard matrix and let $\sigma(j) = Pj + \ell$ be an affine pemutation over $\mathbb{Z}_2^n$. For any signal s, the measurement $\langle R_i,\ s\rangle$ can be written as $f(i,\sigma)\langle R_{i'},\ s\rangle$,* i.e., *a measurement of s by a possibly different Hadamard row $R_{i'}$ multiplied by a function of i and σ (but not of s). In fact, $\langle R_i,\ s\rangle = (-1)^{\langle iP^{-1},\ \ell\rangle}\langle R_{iP^{-1}},\ s\rangle$.*

Proof. Define s'_j by $s'_j = s_{\sigma(j)} = s_{Pj+\ell}$. Then

$$\begin{aligned}
\left\langle R_i,\ s'\right\rangle &= \sum_j (-1)^{\langle i,\ j\rangle} s'_j \\
&= \sum_j (-1)^{\langle i,\ j\rangle} s_{Pj+\ell} \\
&= \sum_k (-1)^{\left\langle i,\ P^{-1}(k+\ell)\right\rangle} s_k \\
&= (-1)^{\left\langle i,\ P^{-1}\ell\right\rangle} \sum_k (-1)^{\left\langle i,\ P^{-1}k\right\rangle} s_k \\
&= (-1)^{\left\langle iP^{-1},\ \ell\right\rangle} \sum_k (-1)^{\left\langle iP^{-1},\ k\right\rangle} s_k \\
&= (-1)^{\left\langle iP^{-1},\ \ell\right\rangle} \left\langle R_{iP^{-1}},\ s\right\rangle .
\end{aligned}$$

□

Thus we want to permute the indices of s by $\sigma(j) = Pj + \ell$ *before* measuring with a Hadamard row, and this can instead be simulated by post-processing *after* measurement of s by some possibly different Hadamard row, whose identity does not depend on s.

5.2.2. *Group Assignment.* Next, we consider the group assignment operation. For fixed integer k, we want to construct an assignment of the N positions to one of 2^k groups. Specifically, we will aim for the structured matrix

$$M_k = \begin{pmatrix} 1 & 1 & 1 & 1 & & & & & \\ & & & & 1 & 1 & 1 & 1 & \\ & & & & & & & & \ddots \end{pmatrix},$$

where there are 2^k rows, N columns, and $N/2^k$ 1's in each row, in positions $iN/2^k$ to $(i+1)N/2^k - 1$ in row i, for $0 \le i < 2^k$. In our context, we want to write $M_k = AR$, where R is a (2^k)-by-N row submatrix of the Hadarmard matrix and A is a (2^k)-by-(2^k) matrix.

We start with an example. If $k = 1$, we want the 2-by-N matrix M_1, which is is

$$M_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & & & & \\ & & & & 1 & 1 & 1 & 1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{pmatrix}.$$

More generally, if we let I_k denote the (2^k)-by-(2^k) identity matrix, we have $M_k = I_k M_k = H_k^2 M_k/2^k = H_k(H_k M_k)/2^k$, as illustrated above for $k = 1$. Thus, in $M_k = AR$, we can take $A = H_k$ provided we can show that the rows of $R = 2^{-k}H_k M_k$ are rows of H_n. This follows by induction from the definition of H_n as $H_n = H_k \otimes H_{n-k}$ and the fact that M_k is of the form $I_k \otimes \mathbf{1}$, where $\mathbf{1}$ here denotes the 1-by-$N/2^k$ matrix of all 1's and is one of the rows of H_{n-k}.

Note that the bit-testing matrix $\boldsymbol{B_1}$ is constructed similarly. One way to see this is by noting that the top row of $\boldsymbol{B_1}$ is equivalent to the row

$$\begin{pmatrix} +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \end{pmatrix},$$

which is a row of R. Other rows of $\boldsymbol{B_1}$ are gotten by applying certain permutation matrices P to the indices j, which, as we have seen above, is compatible with Hadamard measurements. The top row of $\boldsymbol{B_1}$ can be regarded as $(-1)^{\mathrm{msb}(j)}$ and, more generally, row i of $\boldsymbol{B_1}$ is $(-1)^{\text{bit } i \text{ of } j}$. So, if a matrix P acting on j swaps the most significant bit of j with bit i of j, then $(-1)^{\mathrm{msb}(Pj)} = (-1)^{\text{bit } i \text{ of } j}$, as desired.

5.3. **Pairwise Independence.** Next, if the matrix P is chosen at random from amongst invertible matrices and the vector ℓ is chosen at random from $\mathbb{Z}_2^n$, then we claim that the mapping $j \mapsto Pj+\ell$ is pairwise independent.

Lemma 12. *Let $\sigma(j) = Pj + \ell$ be a random permutation where the matrix P is chosen at random from amongst invertible matrices and the vector ℓ is chosen at random from $\mathbb{Z}_2^n$. Then, for any $j \neq j'$ and any $t \neq t'$,* $\Pr(\sigma(j) = t \wedge \sigma(j') = t') = \frac{1}{N(N-1)}$.

Proof. Let $j \neq j'$ be two distinct indices and let $t \neq t'$ be two distinct targets. Even conditioned on the value of P, the distribution on $\sigma(j) = Pj + \ell$ is uniform over $\mathbb{Z}_2^n$, so it suffices to show that $\Pr(\sigma(j') = t' | \sigma(j) = t) = \frac{1}{N-1}$. Conditioned on $Pj + \ell = t$, we have $\sigma(j') = Pj' + \ell = Pj + \ell + P(j' + j) = t + P(j' + j)$, and it suffices to show that $P(j' + j)$ takes on any non-zero value uniformly when $j' + j \neq 0$ and P is random. We may assume that $j = 0$; that is, we need to show that Pj' is uniform over non-zero vectors t' provided $j' \neq 0$.

Given any $j', t' \neq 0$ there is an invertible P with $Pj' = t'$. Furthermore, as P varies at random, Pj' varies uniformly over all non-zero vectors. To see this, fix some non-zero vector k and suppose $P_0 j' = k$. The set $\mathcal{H}$ of invertible matrices that fix k is a multiplicative subgroup of the multiplicative group $\mathcal{G}$ of all invertible matrices. For each $k' \neq 0$ there is, as above, an invertible matrix P with $Pk = k'$. The invertible matrices can be partitioned into multiplicative cosets of $\mathcal{H}$ in $\mathcal{G}$, where a matrix P is in a coset $Q\mathcal{H}$ iff $Pk = Qk$, *i.e.*, iff $PP_0 j' = QP_0 j'$. Since all cosets have the same size, each non-zero vector has an equal chance to be the image of j' under multiplication by a random invertible matrix. The result follows. □

5.3.1. *Using Pairwise Independence.* Finally, we show that pairwise independence suffices. Consider a particular defect, at position j. It is assigned to some group, g. Conditioned on that, each other defect is assigned to g with probability $\frac{2^{-k}N-1}{N-1} \leq 2^{-k}$ by pairwise independence, so the expected number of *other* defects assigned to g is $(d-1)2^{-k}$, which is less than 1/4 provided $2^k > 4d$. It follows by the Markov inequality that the number of other defects assigned is less than 1 with probability at least 3/4 and so each defect is isolated in its own group with probability at least 3/4. Thus the expected number of defects that *fail* to become isolated is at most $d/4$; by the Markov inequality again, the number that fail to be isolated exceeds $3d/4$ only with probability at most 1/3. Thus, with probability at least 2/3, we isolate at least $d/4$ defects. Note that the number of groups (and, so, the number of false positives) is $O(d)$, not $d\log(N)$. We need another factor of $\log(N)$ in the number of groups in order to take the row tensor product with $\boldsymbol{B_1}$ to perform identification. Thus we have

Lemma 13. *There is a randomized algorithm and $\epsilon > 0$ that, for each signal s of length N with d defects, returns a list of at most $O(d)$ items that contains at least ϵd defects. The algorithm takes time* $\mathrm{poly}(d \log N)$*, excluding the time to make measurements.*

5.4. **Duality.** In the above discussion, there's a vector s of length N with d defects (non-zeros). Conceptually, we make t tests of s, where each test includes exactly N/t of the N items and the t tests partition the N items. We can view the collection of test data as $S'H_n s$, where S' has exactly one non-zero per row, a 1. Thus S' picks a random sample. Above we took the view that $S'H_n$ is a collection of tests on our d-defect sample, s. Now we take the view that we have a function, f, given by $f = H_n s$. That is, f consists of the sum of d columns of the Hadamard matrix, or, more generally, a linear combination of d columns. The operator S' samples f in t places and we have an algorithm that recovers s efficiently from $S'H_n s = S'f$. The vector s can be regarded as a compressed representation for the vector $f = H_n s$, so that recovering s is equivalent to recovering f. Thus we have:

Lemma 14. *There is a randomized algorithm and $\epsilon > 0$ that, for each signal s of length N with d non-zero Hadamard coefficients, returns a list of at most $O(d)$ Hadamard rows that contains at least ϵd rows with non-zero dot product to s. The algorithm takes time* $\mathrm{poly}(d \log N)$, including *the time to make the measurements (which are samples to s).*

Recall that our goal is to recover *all* non-zero Hadamard coefficients, analogous to Corollary 10. Unfortunately, this needs to wait until Section 6.1.

5.5. **Fourier.** In this section, we consider replacing the Hadamard matrix with the N-by-N Discrete Fourier Transform matrix, F_N. In this section, let i denote the square root of -1 and index tests by k instead of i. The (k, j) entry of F_N is given by $e^{2\pi ijk/N}/\sqrt{N}$. Our goal is to construct a "partition" of $\mathbb{Z}_N$ into t sets of sizes N/t. The notion of partition here, however, will be relaxed. Given a strict partition, *i.e.*, a collection $\{S_k\}$ of subsets of $\mathbb{Z}_N$, such that each $j \in \mathbb{Z}_N$ is in exactly one S_k, we want a set of vectors r_k such that:

- If $j \in S_k$, then $|r_k(j)| \geq \Omega(1)$. (The vector r_j fractionally contains each $j \in S_j$.)
- For each k, we have $\sum_{j \in S_k} |r_k(j)|^2 \leq O(N/t)$. (The vector r_j fractionally contains no more than $(1/t)$ of all items.)

There are many technical changes to the Hadamard discussion to make all of this go through, which are beyond the scope of this paper. Here we note only that it is possible, and relies on the relaxed notion of partition given above. For more details, see [GGI$^+$02b].

6. Using test results in novel ways

In the previous sections, we focused on approximate versions of the group testing problem and on how to use certain types of tests or designs. In this section, we turn to novel uses of measurements that are not necessarily of a special type. We use these measurements to obtain information not just about special entries in our streaming data (e.g., those portions of the implicit signal that are nonzero) but about other statistical quantities or properties of that signal or distribution. We begin with point queries on a stream, which are a generalization of the identification problem in Section 4. We then use our techniques to answer point queries to answer range queries. Finally, we show how to adapt range queries to extract approximate quantiles for a data set. This exposition follows the work in several papers, including [AGMS99, GKM$^+$05, GGI$^+$02a, CM03].

6.1. **Point Queries.** A point query is the simplest type of query for a data set. Given a signal vector $\mathbf{A}$ and a query position i, we want to estimate the value $\mathbf{A}(i)$. In the streaming context, we have a stream of records which implicity define a signal $\mathbf{A}$ and at some point in the stream of records, we want an approximate answer to the value of the signal at a point. This point i is an arbitrary position and this problem is considerably different from the problem of identifying large or non-zero entries in the signal $\mathbf{A}$. In the streaming telephone records example in Section 4, we are asking whether phone number 8008001111 is currently active, rather than for a list of active phone numbers.

Fast estimates for these types of aggregate queries are useful in database query optimization, approximate query answering and on-line query processing. Typical database systems use selectivity estimates to generate fast estimates for query optimization and approximate query answering. Selectivity estimation stores some summary information about the signal $\mathbf{A}$ and then uses that summary information to provide query estimates. We discuss what type of summary information to use for selectivity estimation (especially for point queries) and how to gather it efficiently on streaming data.

Suppose that we observe a signal $\mathbf{A}$ as a stream of records in the cash register model. Assume that $\mathbf{A} : [1, N] \to \mathbb{R}$ so that its domain U is of length N. Our goal is to estimate the value of $\mathbf{A}$ at index i. We construct a random vector r of length N which consists of ± 1, chosen uniformly at random. Given vector $\mathbf{A}$, we measure it with r, and obtain a single measurement number, $\langle r,\ \mathbf{A}\rangle$. After collecting this measurement, we use it to estimate the value of $\mathbf{A}$ at index i as

$$X = r(i) \langle r,\ \mathbf{A}\rangle .$$

Let $\|\mathbf{A}\|$ denote $\sqrt{\sum_i \mathbf{A}(i)^2}$.

Lemma 15. *With proability at least* $3/4$*, the estimator* X *satisfies* $X = \mathbf{A}(i) \pm 2\sqrt{\sum_{j\neq i} \mathbf{A}(j)^2}$*. If* Z *is the median of* $O(\log(1/\delta))$ *copies of the mean of* $O(1/\epsilon^2)$ *copies of* X*, then, with probability at least* $1 - \delta$*, the estimator* Z *satisfies* $Z = \mathbf{A}(i) \pm \epsilon \|\mathbf{A}\|$*.*

Proof. **Sketch.** Following [AGMS99], we have

$$E[X] = \sum_k E[r(i)\mathbf{A}(k)r(k)] = \mathbf{A}(i).$$

Also,

$$\sigma^2 = \operatorname{var}(X) = E\Big[r(i)^2\Big(\sum_{k\neq i} \mathbf{A}(k)r(k)\Big)^2\Big] = \sum_{k\neq i} \mathbf{A}(k)^2 \le \|\mathbf{A}\|^2 .$$

The Chebychev inequality tells us that $\Pr(|X - E[X]|^2 > 4\sigma^2) < 1/4$, which gives the first statement. To get the second statement, let Y be the mean of $O(1/\epsilon^2)$ copies of X, so $E[Y] = E[X] = \mathbf{A}(i)$ and we can arrange that $E[Y^2] \le \frac{\epsilon^2}{8} \|\mathbf{A}\|^2$. As above, $|Y - \mathbf{A}(i)| \le \epsilon \|\mathbf{A}\|$ except with probability $1/8$. Let Z be the median of $O(\log(1/\delta))$ copies of Y. Then $|Z - \mathbf{A}(i)| > \epsilon \|\mathbf{A}\|$ implies that, for half of the Y's, we have $|Y - \mathbf{A}(i)| > \epsilon \|\mathbf{A}\|$. We expect this to happen for just $1/8$ of the Y's and, by the Chernoff bound, this happens for half of the Y's only with probability exponentially small in the number $O(\log(1/\delta))$ of Y's, which we can arrange to be at most δ. □

Observe that because our measurement process is linear, we can maintain an accurate measurement number $\langle r,\ \mathbf{A}\rangle$ on a stream of updates to the signal $\mathbf{A}$.

This lemma is formulated in terms of random ± 1 vectors. We now give an equivalent formulation in terms of group testing. Assume that G is a random set of distinct values, each drawn from the domain with probability $1/2$. We encode this set G with a random $\{0, 1\}$-valued vector g, as is traditional in group testing. Let $\mathbf{A}_G$ denote $\mathbf{A}$ projected onto the set G (*i.e.*, form the dot product $\langle g,\ \mathbf{A}\rangle$), and let

$$\Sigma_G \mathbf{A} = \sum_{j\in G} \mathbf{A}(j)$$

denote the total sum of $\mathbf{A}$ over G. Also, $\Sigma\mathbf{A}$ is the total sum over the signal. This procedure is equivalent to measuring $\mathbf{A}$ with a random group testing design. For $\mathbf{A}(i)$, consider $E\Big[\Sigma_G \mathbf{A} | i \in G\Big]$. We can estimate this conditional expectation from our counts $\Sigma_G \mathbf{A}$, where G contains i. In addition, we can show that this conditional expectation is equal to $\mathbf{A}(i) + \frac{1}{2}\Sigma_{U\setminus\{i\}}\mathbf{A}$, since the contribution of i is always counted but the contribution of each other point is counted only half the time. Furthermore, we know $\Sigma\mathbf{A}$, so we can estimate $\mathbf{A}(i)$ as

$$\mathbf{A}(i) = 2\Big(\mathbf{A}(i) + \frac{1}{2}\Sigma_{U\setminus\{i\}}\mathbf{A}\Big) - \Sigma\mathbf{A} = 2E\Big[\Sigma_G \mathbf{A} | i \in G\Big] - \Sigma\mathbf{A}.$$

We maintain $\Sigma_G \mathbf{A}$ for each of several random sets G. This is equivalent to forming a random group testing design with several rows. It turns out that this procedure yields an estimate good to within $\epsilon\Sigma\mathbf{A}$ additively if we take an average of $O(1/\epsilon^2)$ repetitions.

Finally, we can state the non-adaptive analog to Theorem 10.

Theorem 16. *Suppose there is a linear testing procedure that can be applied to any subset of items. Then there is a randomized algorithm that finds all at-most-d defects in a sample of length N using at most $d\,\mathrm{polylog}(N)$ tests. The overall algorithm is non-adaptive and succeeds in the sense that there is a distribution $\mathcal{D}$ on measurement matrices $\boldsymbol{M}$, such that, for each signal s, most $\boldsymbol{M} \sim \mathcal{D}$ work on s. Excluding time to make measurements, either algorithm runs in time $\mathrm{poly}(d \log N)$.*

sketch. Use an ϵ-approximate identification scheme to identify ϵd of the d defects. Use Lemma 15 to test each tenatively identified position j or, more generally, to estimate the coefficient c_j. If we let e_j denote the vector that is 1 in position j and 0 elsewhere and let Λ denote the set of identified positions, form the vector $s - \sum_{j\in\Lambda} c_j e_j$ and measure it as

$$\boldsymbol{M}\left(s - \sum_{j\in\Lambda} c_j e_j\right) = \boldsymbol{M}s - \sum_{j\in\Lambda} c_j \boldsymbol{M} e_j,$$

by subtracting $\sum_{j\in\Lambda} c_j \boldsymbol{M} e_j$ from the *original* measurement $\boldsymbol{M}s$, and without adaptively revisiting the original data. Use independent randomness for the various iterations of the algorithm and lower the failure probabilities of each step somewhat so that, taking a union bound, the overall failure probability remains under control. □

6.2. **Dyadic Ranges.** We can, in a similar fashion, estimate the sum of entries $\mathbf{A}(j)$ for all j in any dyadic interval in U, up to $\pm\epsilon\Sigma\mathbf{A}$. By writing any interval as a disjoint union of at most $\log N$ dyadic intervals, we can estimate the number of data set items in any interval.

To sketch our data structure for estimating dyadic ranges, we partition the domain U into dyadic intervals $I_{j,k}$ where $I_{j,k}$ is an interval of the form $[k2^{\ell-j}, (k+1)2^{\ell-j} - 1]$ for integers j and k and $\ell = \log_2 N$. The parameter j of a dyadic interval is its resolution level from coarse, $I_{0,0} = U$, to fine, $I_{\ell,k} = [k]$. For each resolution level j, pick a random subset of the intervals $I_{j,k}$, $k = 0, 1, \ldots, 2^j - 1$. We choose each interval with probability 1/2. Let $\mathbf{A}$ be the union of these intervals and let $\Sigma_S\mathbf{A}$ be the total of the values in the data set that are projected onto S. Or, more formally, $\Sigma_S\mathbf{A} = \sum_{i\in S}\mathbf{A}(i)$. We repeat this process

$$\mathrm{num\_copies} = O(\epsilon^{-2}) \log(\delta^{-1}\log(N)\log(\epsilon^{-2}N))$$

times and generate sets $S_1, \ldots, S_{\mathrm{num\_copies}}$ per resolution level. The counts $\Sigma_{S_l}\mathbf{A}$ for all sets that we have picked comprise our Random Subset Sum (RSS) summary structure. See Figure 3 for an illustration of the RSS data structure. In addition we store and maintain $\Sigma\mathbf{A}$ exactly.

We maintain these RSSs as we stream over the data as follows. For updates with index i, for each resolution level j, we quickly locate the single dyadic interval $I_{j,k}$ into which i falls by examining the high order bits of i in binary. Then, we quickly determine those sets S_l that contain $I_{j,k}$. For each such set we update $\Sigma\mathbf{A}_{S_l}$ appropriately.

6.3. **Quantiles.** Now we turn attention to quantiles. Quantiles are order statistics for a data set. The elementary order statistics include the median (the "halfway point") and the quartiles (the "quarter-way points"). In general, the q-quantiles, for small real-valued $q > 0$, of an ordered sequence of $\Sigma\mathbf{A}$ data items are the values of the data set which have rank $kq\Sigma\mathbf{A}$, for $k = 1, 2, \ldots, 1/q - 1$.

More precisely, there is a universe of item types, which we take to be $1, 2, 3, \ldots, N$. Each type i appears in the data set some non-negative[1] number $\mathbf{A}(i)$ of times. The total number of items in the data set is $\Sigma\mathbf{A} = \sum_i \mathbf{A}(i)$, which may be greater than or less than N. We assume that the data set is presented to us through a series of updates of the form $(i, v(i))$, which means "add

[1]There are additional algorithmic techniques one can use to handle the situation of negative values for $\mathbf{A}$, but both these techniques and the definition of quantile in this case are beyond the scope of this paper.

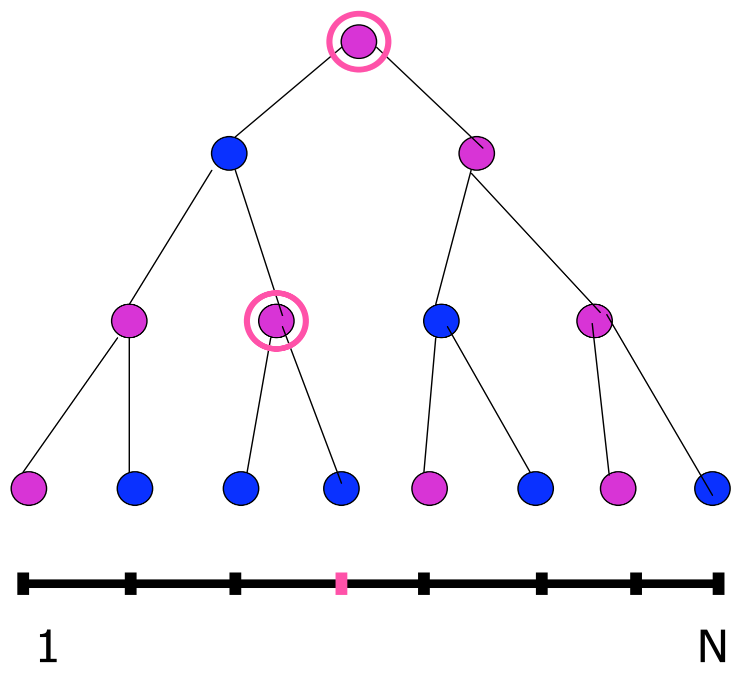

FIGURE 3. At each resolution level j in the dyadic hierarchy, we keep a random subset of intervals over which we maintain frequency counts.

$v(i)$ to $\mathbf{A}(i)$," where $v(i)$ may be negative provided $\mathbf{A}(i)$ remains non-negative. For example, in a rental car agency, i may be a car model year and $\mathbf{A}(i)$ is the number of cars of vintage i on the road. When a car of vintage i is rented, update $(i, +1)$ tells us to add 1 to $\mathbf{A}(i)$; when the car is returned, $(i, -1)$ tells us to subtract 1 from $\mathbf{A}(i)$. The active phone call setup of section 3.3 gives another example.

Most database management systems (DBMSs) maintain order statistics or quantiles on the contents of their database relations as quantiles find multiple uses in databases and in data mining, more generally. Simple statistics such as the mean and variance are both insufficiently descriptive and highly sensitive to data anomalies in real-world data distributions. Quantiles, on the other hand, can summarize massive database relations more robustly.

Our goal is to estimate q quantiles on demand. Because we will be using a small amount of computational resources, we will focus on computing ϵ-approximate q-quantiles rather than on exact q-quantiles. That is, we define a *k'th ϵ-approximate q-quantile* to be any j_k such that

$$\sum_{i<j_k} \mathbf{A}(i) \le kq\Sigma\mathbf{A} \le \epsilon\Sigma\mathbf{A} + \sum_{i\le j_k} \mathbf{A}(i),$$

for $k = 1, \ldots, 1/q - 1$. The set of indices $j_1, \ldots, j_{1/q-1}$ will be q-quantiles approximate up to $\pm\epsilon\Sigma\mathbf{A}$. Note that if $\epsilon = 0$, then we seek the exact quantiles.

We will illustrate the algorithm, using the median as an example quantile. As an overview, let F denote the cumulative function for A, so that $F_j = \sum_{i<j} \mathbf{A}(i)$, and so the median is a point j with $F_j \le \frac{1}{2}\Sigma\mathbf{A} \le F_{j+1}$. We will approximate F_j by $\widetilde{F}_j$, using the techniques of Section 6.2. We assume that we know $\Sigma\mathbf{A}$ exactly, by keeping a single counter. We then define I_j be an indicator

as to whether j is apparently too large to be the median, using *the approximation* $\widetilde{F}$. That is,

$$I_j = \begin{cases} 0, & \widetilde{F}_j \leq \frac{1}{2}\Sigma\mathbf{A}; \\ 1, & \widetilde{F}_j > \frac{1}{2}\Sigma\mathbf{A}. \end{cases}$$

Although F_j increases with j, the approximation $\widetilde{F}_j$ is not necessarily monotonic and, therefore, I_j may not be monotonic. Nevertheless, we can assume $I_0 = 0$ and $I_N = 1$ and so there exists some point j with $I_j = 0$ and $I_{j+1} = 1$ (for example, the largest j for which $I_j = 0$ is such a point). Because $\widetilde{F}_j$ is an approximation to F_j with accuracy guarantees, we can conclude that j is an approximate median.

We now proceed formally. As in Section 6.2, we measure the signal in such a way that, for any dyadic range R, we can estimate $\sum_{i\in R}\mathbf{A}(i)$ to within $\eta\Sigma\mathbf{A}$, additively, where η will be specified below. For any j, the range $0 \leq i < j$ can be partitioned into $O(\log(N))$ dyadic ranges, so we can estimate $F_j = \sum_{0\leq i<j}\mathbf{A}(i)$ to within $O(\log(N)\eta\Sigma\mathbf{A})$, by summing the approximations on the constituent ranges. Let $\widetilde{F}_j$ denote this approximation to F_j, and, by adjusting η and adding the appropriate multiple of $\eta\Sigma\mathbf{A}$, we may assume that $F_j \leq \widetilde{F}_j \leq F_j + O(\log(N)\eta\Sigma\mathbf{A})$. Define I_j from $\widetilde{F}_j$, as above.

Next, we use bisection search to find, in the time to do $O(\log(N))$ estimations of $\widetilde{F}$, a position j such that $I_j = 0$ and $I_{j+1} = 1$. We start with $j_{\text{lo}} = 0$ and $j_{\text{hi}} = N$, so that $I_{j_{\text{lo}}} = 0$ and $I_{j_{\text{hi}}} = 1$. We repeatedly let $j_{\text{mid}} = (j_{\text{lo}} + j_{\text{hi}})/2$ be the midpoint between j_{lo} and j_{hi} and set j_{lo} or j_{hi} to be j_{mid}, depending on $I_{j_{\text{mid}}}$, maintaining the invariant that $I_{j_{\text{lo}}} = 0$ and $I_{j_{\text{hi}}} = 1$. The algorithm terminates when $j_{\text{lo}} + 1 = j_{\text{hi}}$, and outputs $j = j_{\text{lo}}$ as an approximate median.

At this point, we have

$$F_{j_{\text{lo}}} \leq \widetilde{F}_{j_{\text{lo}}} \leq \frac{1}{2}\Sigma\mathbf{A} < \widetilde{F}_{j_{\text{lo}}+1} \leq F_{j_{\text{lo}}+1} + O(\log(N)\eta\Sigma\mathbf{A}),$$

as desired, provided the $O(\log(N)\eta\Sigma\mathbf{A})$ term is at most $\epsilon\Sigma\mathbf{A}$.

6.4. **Computational cost and the return of coding theory.** An important technical detail is how to store and index various S_l's, which are random subsets. The straightforward way would be to store them explicitly, perhaps as a bitmap. This would use space $O(N)$ which we can not afford. For our algorithm, we instead store certain random seeds of size $O(\log N)$ bits and compute a (pseudorandom) function that explicitly shows whether $i \in S_l$ or not. For this, we use the standard 3-wise independent random variable construction shown below, since it works well with our dyadic construction.

We need a generator $G(s,i) = S_i$ that quickly outputs the i'th bit of the set S, given i and a short seed s. In particular, the generator takes a $O(\log N)$-bit seed and can be used to generate sets S of size $O(N)$. The generator G is the extended Hamming code, *e.g.*,

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix},$$

which consists of a row of 1's and then all the columns, in order. So, for each resolution level j, there's a G of size $(j+1)\times 2^j$. Then $G(s,i)$ is the seed s of length $j+1$ dotted with the i'th column of G modulo 2, which is efficient to compute—note that the i'th column of G is a 1 followed by the binary expansion of i.

Thus the algorithms in this section use space, and number of tests, and runtime (excluding the time to make tests), all at most polynomial in $d, \log(N), \log(\Sigma\mathbf{A}), 1/\epsilon, and \log(1/\delta)$.

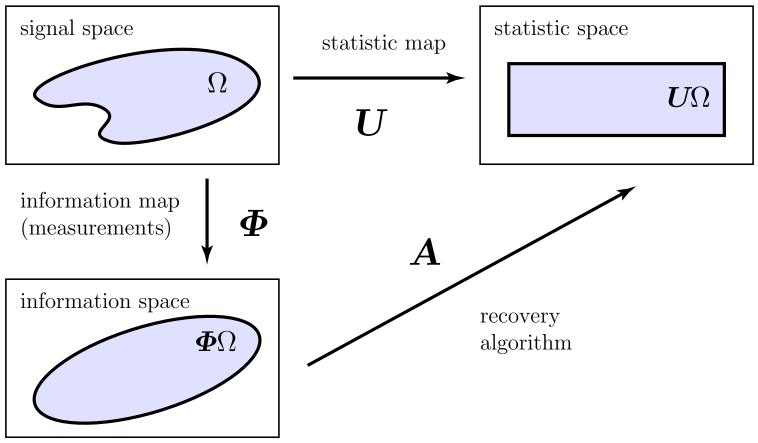

FIGURE 4. Mathematical setting for an optimal recovery problem.

7. CONCLUSIONS

There are many extensions and further developments in streaming algorithms than the few we highlight here. See [Mut05] for a detailed discussion. There are algorithms and software and hardware systems for a large variety of statistical and combinatorial problems, especially for communication networks. A significant number of these results and implementations require some aspect of group testing in order to meet stringent computational constraints. The role of group testing in these algorithms may not be touted but a closer look will reveal its pervasive appearance.

From a mathematical perspective, the streaming algorithms we have discussed are part of a larger mathematical theory of *optimal recovery.* The theory of optimal recovery was introduced in the late 1950s [GW59] and polished to a shine by the mid-1980s [MR77, TW80, Pin86]. We base our discussion on [CL00].

Let Ω be a model for signals of interest, which belong to some ambient signal space. Suppose that we are interested in calculating some (possibly nonlinear) statistic $\boldsymbol{U}f$ of each signal f in Ω. An essential point is that approximating signal statistics may be far easier than recovering the signal completely. We do not have access to the signal itself, but we can obtain some information about the signal via a linear *information* operator $\boldsymbol{\Phi}$. The goal of the recovery problem is to produce an algorithm $\boldsymbol{A}$ that estimates the statistic $\boldsymbol{U}f$ from the information $\boldsymbol{\Phi}f$. Figure 4 illustrates this setting.

If we are concerned primarily with how well the algorithm approximates the statistic *per se*, we could define the *error* as

$$E(\boldsymbol{A}) = \sup_{f \in \Omega} \|\boldsymbol{A}(\boldsymbol{\Phi}f) - \boldsymbol{U}f\|$$

where $\|\cdot\|$ is a norm on the statistic space. The *intrinsic error* in a recovery problem is defined as

$$E_\star = \inf_{\boldsymbol{A}} E(\boldsymbol{A}),$$

the minimal error attainable by any algorithm, computable or not. The intrinsic error is a *fundamental mathematical bound* on how well an algorithm can perform. An algorithm that attains the intrinsic error is called *optimal.* In some cases, it is possible to design the information operator $\boldsymbol{\Phi}$ to minimize the intrinsic error.

We can easily generalize and abstract many of the streaming algorithms succinctly in this framework. The problem domain leads to a signal model Ω and to a relevant set of statistics $\boldsymbol{U}$. Technological and economic limits on possible software and hardware (such as sampling devices) constrain which information operators $\boldsymbol{\Phi}$ are practicable, including the type and number of measurements.

The algorithms $\boldsymbol{A}$ that can be used to approximate the signal statistics are restricted by the computational resources available, such as time and storage. The goal then is to design an information operator and an algorithm that are technologically feasible and still yield an error as small as possible.

This framework allows us to consider a much larger and richer set of mathematical and engineering questions than streaming algorithms alone. We have much greater flexibility to adapt our problem to a particular application. Suppose that our hardware has certain resource constraints but that we are willing to expend more resources on the recovery algorithm. Or, we may have an application where our signal space is especially restricted. Mathematically, we can change our focus from algorithms,per se, to more fundamental resource questions about how many measurements, of what type, are necessary to compute various types of statistics. What fundamental tradeoffs are there in using randomized information maps versus deterministic ones, for example. We are witnessing a flurry of current research activity in this area [VMB02, MV, TG05, CT04b, CT04a, CDS98, Don04b, Don04a, DT05, HN05, CT05] (to mention just a few) but this rich and interesting intersection of mathematics, statistics, and computer science has yet to be fully mined.

References

[AGMS99] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. System Sci.*, 58(1):137–147, 1999.

[AMS96] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. 1996.

[BYJKS02] Z. Bar-Yossef, T. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, pages 209–218, 2002.

[BYKS01] Ziv Bar-Yossef, S. Ravi Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. In *ACM Symposium on Theory of Computing*, pages 266–275, 2001.

[CDS98] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[CL00] W. Cheney and W. Light. *A course in approximation theory.* Brooks-Cole, 2000.

[CM03] G. Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Proc. ACM Principles of Database Systems*, pages 296–306, 2003.

[CT04a] E. Candès and T. Tao. Decoding by linear programming. *IEEE Trans. Information Theory*, 2004. Submitted.

[CT04b] E. Candès and T. Tao. Near optimal signal recovery from random projections and universal encoding strategies. *IEEE Trans. Information Theory*, 2004. Submitted.

[CT05] E. Candès and T. Tao. The Dantzig selector: statistical estimation when p is much larger than n. *Annals of Statistics*, 2005. Submitted.

[DH93] D.-Z. Du and F. K. Hwang. *Combinatorial Group Testing and its Applications.* World Scientific, New Jersey, 1993.

[Don04a] D. Donoho. Compressed sensing. 2004. Preprint.

[Don04b] D. Donoho. For most large underdetermined systems of linear equations, the minimal l1-norm solution is also the sparsest solution. 2004. Preprint.

[Dor43] R. Dorfman. The dection of defective members of large populations. *Ann. Math. Statist.*, 14:436–440, 1943.

[DT05] D. Donoho and J. Tanner. Neighborliness of randomly-projected simplices in high dimensions. 2005. Preprint.

[Fel50] W. Feller. *An introduction to probability theory and its applications*, volume 1. John Wiley and Sons, New York, 1950.

[FKSV99] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An Approximate L^1 Difference Algorithm for Massive Data Streams. In *Proceedings of Foundations of Computer Science (FOCS 1999)*, pages 501–511, 1999.

[GGI$^+$02a] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. Fast, Small-Space Algorithms for Approximate Histogram Maintenance. In *ACM Symposium on Theoretical Computer Science*, 2002.

[GGI$^+$02b] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. J. Strauss. Near-optimal sparse Fourier representations via sampling. In *ACM Symposium on Theoretical Computer Science*, 2002.

[GKM$^+$05] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, M. J. Strauss S. Muthukrishnan, and Martin Strauss. Domain-Driven Data Synopses for Dynamic Quantiles. *IEEE Trans. Knowl. Data Eng.*, 17(7):927–938, 2005.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc. of the ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[GW59] M. Golomb and H. F. Weinberger. Optimal approximation and error bounds. In R. E. Langer, editor, *On Numerical Approximation*, pages 117–190. The University of Wisconsin Press, 1959.

[HN05] J. Haupt and R. Nowak. Signal reconstruction from noisy random projections. *IEEE Trans. Information Theory*, 2005. Submitted.

[HRR98] M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams, 1998.

[KM91] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. In *Proc. of the ACM Symposium on Theory of Computing*, pages 455–464, 1991.

[MR77] C. A. Michelli and T. J. Rivlin. A survey of optimal recovery. In *Optimal estimation in approximation theory.* Plenum Press, 1977.

[MS77] F J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, 1977.

[Mut05] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1, 2005.

[MV] I. Maravić and M. Vetterli. Sampling and reconstruction of signals with finite innovation in the presence of noise. Preprint.

[Pin86] A. Pinkus. n-widths and optimal recovery. In C. de Boor, editor, *Approximation theory*, volume 36 of *Proceedings of Symposia in Applied Mathematics.* American Mathematical Society, 1986.

[SG59] M. Sobel and P. A. Groll. Group testing to eleminate efficiently all defectives in a binomial sample. *Bell System Technical Journal*, 28:1179–1252, 1959.

[Ste57] A. Sterrett. On the detection of defective members of large populations. *Ann. Math. Statist.*, 28:1033–1036, 1957.

[TG05] J. Tropp and A. C. Gilbert. Signal recovery from partial information via orthogonal matching pursuit. 2005. Submitted.

[TW80] J. F. Traub and H. Wozniakowski. *A general theory of optimal algorithms.* Academic Press, 1980.

[VMB02] M. Vetterli, P. Marziliano, and T. Blu. Sampling signals with finite rate of innovation. *IEEE Trans. Signal Proc.*, 50(6), June 2002.

[WNB06] M. Wassermann, A. Neisser, and C. Bruck. Eine serodiagnostische Reaktion bei Syphilis. *Deutsche medicinische Wochenschrift*, 32:745–746, 1906.