Group Testing and Sparse Signal Recovery

Anna C. Gilbert
Department of Mathematics
University of Michigan, Ann Arbor
Email: annacg@umich.edu

Mark A. Iwen
Institute for Mathematics and Its Applications
University of Minnesota, Twin Cities
Email: iwen@ima.umn.edu

Martin J. Strauss
EECS and Mathematics
University of Michigan, Ann Arbor
Email: martinjs@umich.edu

***Abstract*—Traditionally, group testing is a design problem. The goal is to design an optimally efficient set of tests of items such that the test results contain 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 during World War II to remove men with syphilis from the draft [5]. 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 because many of the same techniques that are useful for designing tests are also used to solve algorithmic problems in compressive sensing, as well as to analyze and recover statistical quantities from streaming data. This article is an expository article, with the purpose of examining the relationship between group testing and compressive sensing, along with their applications and connections to sparse function learning.**

I. Introduction

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 an efficient method for detecting draftees with syphilis. 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 on the pooled samples to test for the presence of the syphilitic antigen. If no one in the pool has the antigen, then the pool does not, and the test is negative. 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. These observations provide an efficient testing scheme: if no one in the pool has the antigen, then we save four tests as compared with five individual tests. If one or more in the pool have the antigen, then we waste a single extra test on the entire group (this is unlikely as we assume few in the entire population have syphilis).

The purpose of this article is to examine the role of group testing [6] in streaming algorithms [12], compressive sensing [4], [2], and function learning/interpolation [11]. Recovering an important set of members of a large group from a small number of stored tests is a fundamental task in each of these areas, and general results on group testing are potentially valuable compression, sketching, and learning techniques. As we shall see below, this fact has been exploited in both streaming algorithms and compressed sensing applications. Furthermore, as discussed in the last two sections of this paper, there are a number of new applications in machine learning.

We begin our discussion in Section II with a review of combinatorial group testing. In Section III, we give an approximate group testing algorithm which is at the core of many streaming and highly efficient compressed sensing algorithms [7], [10]. This algorithm is a relaxation of the original formulation of group testing. In the following sections, we connect three applications to group testing through various algorithmic models. Next, in Section IV, we discuss models for streaming data, including the algorithmic models for processing such data streams. We connect streaming algorithms and group testing through these algorithmic models. We proceed in Section V to mention several direct applications of group testing methods to compressive sensing. Section VI relates group testing and compressive sensing to learning functions and proposes several potential applications based on this connection. Finally, in Section VII, we suggest novel uses of the results of group testing which show potential in general experimental design.

II. Group Testing Preliminaries

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. In addition, they may be generated randomly or explicitly.

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

$$\mathbf{v} = \mathcal{M}\mathbf{s},$$

where the arithmetic is *boolean*, that is, multiplication of 0s and 1s is the usual multiplication (which coincides with the logical AND) but addition is replaced by the logical OR. Equivalently, we perform the multiplication $\mathcal{M}s$ over $\mathbb{Z}$ and then replace all non-zero entries in the result by 1. This corresponds to *disjunctive* tests 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 (see Section VI for a simple extension to *conjunctive* tests).

Definition 1: The measurement matrix $\mathcal{M}$ is d-disjunct if the ORs of any d columns do not contain any other column.

The d-disjunct condition is equivalent to the $(d+1)$-strongly selective condition employed elsewhere [3]. There are explicit d-disjunct measurement matrix constructions of size $O(d^2 \log N) \times N$ [13].

The group testing algorithms that accompany these types of binary matrices are 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 $\mathcal{M}$ 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. A straightforward algorithm 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 an example of a d-disjunct matrix. The simplest example when $d = 1$ of a d-disjunct matrix is the *bit test matrix*, $\mathcal{B}$. To form the bit test matrix $\mathcal{B}$, set the ith column of $\mathcal{B}$ 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 $\mathcal{B}$ has the form

$$\mathcal{B} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

This matrix is 1-disjunct since any two columns differ in at least one row.

We note that there is a simpler and faster algorithm than the one given above for detecting a single defect using $\mathcal{B}$. Observe,

$$\mathcal{B}\mathbf{s} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

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 or N is so large that we cannot easily store all N items between tests.

III. Approximate Identificiation

In the previous section, we defined the goal of traditional group testing as the 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.

Let us construct a random matrix $\mathcal{R}$ 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 $\mathcal{R}$ 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 $\mathcal{R}$ isolates q items if there are q distinct indices j that are isolated in a group given by a row of $\mathcal{R}$.

Lemma 1: Fix parameters d and N. Let $\mathcal{R}$ 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 $\mathcal{R}$ 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 1: Fix parameters d and N. For sufficiently large $t \leq O(d\log(N))$, there is a t-by-N matrix $\mathcal{M}$ 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 1, 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 $\mathcal{R}_|$ 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 $\mathcal{B}$ 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 $\mathcal{R}$ and each row r' of $\mathcal{B}$, take rr' as a row of our final design, $\mathcal{M}$. Thus $\mathcal{M}$ has approximately $d\log^2(n)$ rows and we say that $\mathcal{M}$ is the row tensor product of B_1 and $\mathcal{R}$, $\mathcal{M} = \mathcal{B}\bigotimes_r \mathcal{R}$. If the random part of $\mathcal{M}$ succeeds, *i.e.*, group r has exactly one defect j, then it is clear that $\mathcal{M}$ 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 1: Suppose there is a Boolean testing procedure that can be applied to any subset of items. Then there is a measurement matrix $\mathcal{M}$ 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 $\mathcal{M}$ that is a row tensor product of the bit-test matrix $\mathcal{B}$ and a random matrix $\mathcal{R}$ 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 $(d\log N)^{O(1)}$.

Furthermore, approximate identification is almost as good as strict identification when the tests are adaptive or the tests are non-adaptive and linear. 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 $\mathbf{s}$ in which supposed "zeros" are actually small-magnitude non-zero numbers.

Lemma 2: 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 $(d\log N)^{O(1)}$ 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 2: 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 $(d \log N)^{O(1)}$.

IV. Streaming Algorithms

In the next sections, we describe several applications in which group testing plays a fundamental role. We begin with data streams and streaming algorithms. A data stream is a computational model that captures the many situations in which data arrive and are processed in a stream or online. 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. 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.

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 s is a one-dimensional function $s : [1 \dots N] \to \mathbb{Z}^+$. We use standard arithmetic (i.e., *we no longer use boolean arithmetic*). We assume that the domain is discrete and ordered and that the function s 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.

For example, a stream of telephone call records could be:

$$\langle 8008001111, 10\rangle, \langle 8008002222, 15\rangle, \langle 8008003333, 13\rangle,$$
$$\langle 8008001111, 23\rangle, \langle 8008001111, 3\rangle \dots$$

where each record contains the phone number and the length of the outgoing telephone call. We construct the underlying signal, namely $s = \langle 8008001111, 36\rangle$, $\langle 8008002222, 15\rangle$, $\langle 8008003333, 13\rangle$ by aggregating the total number of minutes outgoing from numbers 8008001111, 8008002222, 8008003333, etc. 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.

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

$$\langle 1200, 8008001111, 1\rangle, \langle 1203, 8008002222, 1\rangle,$$
$$\langle 1204, 8008002222, -1\rangle, \langle 1206, 8008003333, 1\rangle,$$
$$\langle 1207, 8008001111, -1\rangle, \langle 1208, 8008004444, 1\rangle, \dots$$

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

$$s = \langle 8008001111, 0\rangle, \langle 8008002222, 0\rangle,$$
$$\langle 8008003333, 1\rangle, \langle 8008004444, 1\rangle;$$

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 s 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 s 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.

V. Compressive Sensing

In the compressive sensing (CS) application, data do not arrive sequentially; rather, we assume that we can quickly obtain (either computationally or physically) a vector of measurements of a signal. More precisely, let $\mathbf{s} \in \mathbb{C}^N$ and an invertible $N \times N$ matrix Ψ be given. Furthermore, assume that $\mathbf{s}$ is sparse with respect to Ψ (i.e., only $d \ll N$ entries of $\Psi \cdot \mathbf{s}$ are significant, or large in magnitude). The primary mathematical problem in compressive sensing is to generate a $t \times N$ measurement matrix, $\mathcal{M}$, with the smallest number of rows possible (i.e., t minimized) so that the d significant entries of $\Psi \cdot \mathbf{s}$ can be well approximated by an efficient recovery algorithm using only the t-element vector result of

$$(\mathcal{M} \cdot \Psi) \cdot \mathbf{s}$$

as input.

Although the arithmetic for most CS problems is performed over $\mathbb{C}$ (as opposed to Boolean arithmetic in traditional group testing), the primary difficulty in CS, as in group testing, is *identifying* the significant signal entries. Once they have been identified, we have a variety of methods to estimate their values. In fact, compressive sensing using binary measurement matrices is essentially equivalent to group testing in a population where every population member is allowed a complex 'defectiveness value,' and group tests yield a sum of the tested subset members' defectiveness values. Given this similarity, it

should not be surprising that d-separable matrices $\mathcal{M}$ can be used to solve the CS problems [3], including extensions to the Fourier sampling case [7], [9]. Near-optimal randomized CS methods using techniques similar to those outlined in Section III also exist [8]. Of greatest interest with these approaches is that, similar to the Corollary 2 method, they run in $(d \log N)^{O(1)}$ time. They exchange the use of slightly more samples than linear programming based CS methods [4], [2] for *sublinear* reconstruction runtimes. However, linear programming reconstruction may also be utilized with binary measurement matrices if the number of tests is the dominant concern [1].

VI. Function Learning

We observe, in this section, that the group testing and compressive sensing problems are analogous to learning learning function of the form

$$f(\mathbf{x}) = \mathbf{a} \cdot \mathbf{x}$$

where $\mathbf{a} \in \{0,1\}^N$ (or $\mathbb{C}^N$) is d-sparse or compressible, unknown, and we employ boolean or complex arithmetic. In either case, the $t \times N$ measurement matrix generates t point evaluations of the function f. We seek a reconstruction algorithm to recover (an approximation to) f from these point evaluations. Hence, recent compressive sensing methods provide near-optimal results concerning exact recovery of such sparse linear functions. Likewise, the group testing variant of Corollary 2 provides a near-optimal adaptive learning method for sparse boolean disjunctions.

As a simple consequence of DeMorgan's Law, group testing algorithms can also be modified to allow the near-optimal recovery of sparse conjunctions (i.e., boolean functions $f : \{0,1\}^N \rightarrow \{0,1\}$ consisting of a conjunction of d of the N input variables). Given a boolean measurement matrix $\mathcal{M}$ we simply (i) flip every $\mathcal{M}_{j,k}$ bit, and then (ii) compute $\neg f(\mathcal{M}_j)$ for each new measurement row j. The original recovery algorithm will then reconstruct f. Thus, group testing methods can be used to detect population members of interest which, although invisible when separated, are detectable when they *all* are present in a test. Potential applications include "blind chemistry" (i.e., the application of group testing to determine which d of N reactants create a particular detectable compound of interest).

Finally, we note that compressed sensing methods' ability to approximate sparse linear functions immediately implies their ability to approximate any (smooth) function with a sparse or compressible gradient. Suppose $f : \mathbb{R}^N \rightarrow \mathbb{R}$ has a sparse (or compressible) gradient at x_0. Then, for any $t \times N$ compressed sensing matrix $\mathcal{M}$'s j^{th} row we have

$$\frac{f(x_0 + h\mathcal{M}_j) - f(x_0)}{h} = \nabla f\big|_{x_0} \cdot \mathcal{M}_j + O(h).$$

Hence, evaluating f at $t+1$ points gives us an estimate of $\mathcal{M}\left(\nabla f\big|_{x_0}\right)$, and $\mathcal{M}$'s associated reconstruction algorithm can then be used to recover (an approximation to) f's gradient. This, in turn, identifies f's most important variables (with the largest magnitude partial derivatives) near x_0.

VII. Conclusions

While the original group testing methods were used in experimental design, the extensions to function learning have potential as well. For example, suppose we are interested in achieving the highest possible fuel efficiency per dollar of cost for a new car design. We can model our car design's miles per gallon-dollar as a function of the design parameters. These parameters might include many tire, body shape, engine, material, etc. specifications and costs. Furthermore, evaluating the fuel efficiency per cost of any specific design may be extremely time consuming. In other words, our design function may be hard to evaluate. Evaluating the fuel efficiency to cost ratio for any design may require computationally intensive air flow simulations etc. Even worse, prototypes may have to be manufactured. Hence, questions such as "Which 5 design parameters most impact my new design's fuel efficiency per cost?" might be extremely difficult and costly to answer. Compressed sensing methods may be helpful in such situations by providing a small number of test points (i.e., car designs) which help reveal the most important variables (i.e., design parameters). Other potential applications include the identification of keystone species in complicated ecosystem models, and the fast runtime optimization of complicated code parameters for large batches of problems.

References

[1] R. Berinde, A. C. Gilbert, P. Indyk, H. Karloff, and M. Strauss. Combining geometry and combinatorics: A unified approach to sparse signal recovery. *preprint*, 2008.

[2] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52:489–509, 2006.

[3] G. Cormode and S. Muthukrishnan. Combinatorial Algorithms for Compressed Sensing. *Conference on Information Sciences and Systems*, March 2006.

[4] D. Donoho. Compressed Sensing. *IEEE Trans. on Information Theory*, 52:1289–1306, 2006.

[5] R. Dorfman. The detection of defective members of large populations. *Ann. Math. Stat.*, 1943.

[6] D. Z. Du and F. K. Hwang. *Combinatorial Group Testing and Its Applications*. World Scientific, 1993.

[7] A. Gilbert, S. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal sparse Fourier representations. *SPIE*, 2005.

[8] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. Algorithmic linear dimension reduction in the l_1 norm for sparse vectors. *submitted*, 2006.

[9] M. A. Iwen. A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. In *SODA*, 2008.

[10] J. Laska, S. Kirolos, Y. Massoud, R. Baraniuk, A. Gilbert, M. Iwen, and M. Strauss. Random sampling for analog-to-information conversion of wideband signals. *Proc. IEEE Dallas Circuits and Sys Conf.*, 2006.

[11] Y. Mansour. Randomized approxmation and interpolation of sparse polynomials. *SIAM Journal on Computing*, 24:2, 1995.

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

[13] E. Porat and A. Rothschild. Explicit non-adaptive combinatorial group testing schemes. *ICALP*, 2008.