Bounds for Nonadaptive Group Tests to Estimate the Amount of Defectives[*]

Peter Damaschke and Azam Sheikh Muhammad
Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
`[ptr,azams]@chalmers.se`

Abstract

Group testing is the problem of finding d defectives in a set of n elements, by asking carefully chosen subsets (pools) whether they contain defectives. Strategies are preferred that use both a small number of tests close to the information-theoretic lower bound $d \log_2 n$, and a small constant number of stages, where tests in every stage are done in parallel, in order to save time. They should even work if d is not known in advance. In fact, one can succeed with $O(d \log n)$ queries in two stages, if certain tests are randomized and a constant failure probability is allowed. An essential ingredient of such strategies is to get an estimate of d within a constant factor. This problem is also interesting in its own right. It can be solved with $O(\log n)$ randomized group tests of a certain type. We prove that $\Omega(\log n)$ tests are also necessary, if elements for the pools are chosen independently. The proof builds upon an analysis of the influence of tests on the searcher's ability to distinguish between any two candidate numbers with a constant ratio. The next challenge is to get optimal constant factors in the $O(\log n)$ test number, depending on the prescribed error probability and the accuracy of d. We give practical methods to derive upper bound tradeoffs and conjecture that they are already close to optimal. One of them uses a linear programming formulation.

Keywords: group testing, combinatorial search, learning by queries, nonadaptive strategy, competitive ratio, randomization, linear program, lower bound
AMS subject classification: 68Q17 Computational difficulty of problems, 68Q32 Computational learning theory, 68Q87 Probability in computer science, 68W20 Randomized algorithms.

1 Introduction

Suppose that d unknown elements in a set of n elements have a property called *defective*, or synonymously, *positive*. Non-defective elements are also called *negative*. A *group test* takes a subset of elements, called a *pool*, and returns 1 if the pool contains at least one defective, and 0 otherwise. We also speak of a *positive pool* and *negative pool*, respectively. The combinatorial

[*]This is an extended version of a paper that appeared in preliminary form in the Proceedings of the *4th International Conference on Combinatorial Optimization and Applications COCOA 2010*, Big Island, Hawaii, *Lecture Notes in Computer Science* (Springer) 6509, pages 117-130. The present manuscript slightly deviates from the journal version, due to small corrections.

group testing problem asks to determine at most d defectives using a minimum number of group tests; we also refer to them as *queries*. We remark that d is usually small compared to n. Queries may be asked in *stages*, where all queries are prepared prior to the stage and then asked in parallel. Queries prepared for the next stage can depend on the outcome of all previous stages. Strategies with one query per stage are also called (fully) *adaptive*, whereas 1-stage strategies are (fully) *nonadaptive*. Group testing strategies with minimal "adaptivity" are preferable for applications where the tests are time-consuming. Next, it is often assumed that d (or at least a good upper bound for d) is known beforehand. A more realistic scenario is that not only the defectives are completely unknown, but also their number d is.

Group testing with its numerous variants is a classical problem in combinatorial search, with a history dating back to year 1943 [10], and it has applications in chemical testing, bioinformatics, communication network protocols, fault diagnosis, information gathering, compression, streaming algorithms, etc., see for instance [3, 4, 5, 6, 8, 9, 11, 13, 14, 15, 16]. In chemical testing applications, the elements are chemical samples, and the defectives are contaminated samples. Instead of testing the samples individually it is often possible to test a whole pool at once, so that a negative result reveals immediately that all samples in the pool are negative.

By the trivial information-theoretic lower bound, essentially at least $d \log n$ queries are necessary, in the worst case, to identify d defectives. (Here and in the following, logarithms are always base 2, if not said otherwise.) A group testing strategy using $O(d \log n)$ queries, despite ignorance of d before the testing process, is called competitive, and the "hidden" constant factor is the competitive ratio. (However, we always assume that the presence of defectives is known, that is, $d \geq 1$.)

To our best knowledge, the currently best competitive ratio for deterministic, adaptive strategies is 1.5 [17]. For 1-stage group testing, at least $\Omega((d^2/\log d) \log n)$ queries are needed even in the case of a known d, see [1]. As opposed to this, already 2 stages are enough to enable an $O(d \log n)$ test strategy, also the competitive ratio has been improved in several steps [8, 12, 2] to currently 1.9 for all d, and asymptotically to 1.44 as d grows [2]. Still these strategies assume d being known in advance.

Apparently we were the first to study group testing strategies that are both minimal adaptive and competitive, i.e., suitable even when nothing about the magnitude of d is known beforehand [7]. Any efficient deterministic competitive group testing strategy needs $\Omega(\log d/\log\log d)$ stages (and $O(\log d)$ stages are sufficient). The picture changes radically when randomization is applied. First we can estimate d; for clarity we define this subproblem formally:

BDNNGT (Bounding the Defective Number by Nonadaptive Group Tests):

Let d denote the unknown number of defectives. A searcher can prepare some number L of pools and perform nonadaptive group tests on them. Let $s = s_0 \ldots s_{L-1}$ be the string of test results, where $s_i = 0$ ($s_i = 1$) if the ith pool is negative (positive). We tolerate some fixed small failure probability $\epsilon > 0$. Based on s, the searcher has to output an alleged defective number d' such that:

- On the one hand, d is underestimated (that is, $d' < d$) with probability at most ϵ.
- On the other hand, the expected ratio d'/d, in the good case $d' \geq d$, is bounded by some constant c independently of d.

As shown in [7], for every fixed failure probability $\epsilon > 0$ there is a randomized strategy for BDNNT using $L = g \log n$ pools (g constant), where c depends on ϵ and g, but not on d and n. Then, we can subsequently apply any 2-stage $O(d \log n)$ strategy for known d (using d' in the role of d), and thus obtain a randomized 3-stage competitive strategy. If we instead append a randomized 1-stage strategy with $O(d \log n)$ queries [2], we even get a competitive group testing strategy that needs only 2 stages.

Determining d exactly would be as hard as combinatorial group testing itself, thus it would require $\Omega((d^2/\log d)\log n)$ nonadaptive queries. But an estimate of d within a constant factor is sufficient (and also necessary) for minimal adaptive competitive group testing. We call the expected ratio of our estimate and the true d a competitive ratio as well; it is always clear from context which competitive ratio is meant.

In fact, a nonadaptive estimator of d as specified above is not hard to obtain [7]. To this end we prepare pools as follows. We fix some probability q and put every element in the pool independently with probability $1 - q$. Clearly, the group test gives the result 0 and 1 with probability q^d and $1 - q^d$, respectively. We prepare $O(\log n)$ of these pools such that the values $1/\log_2(1/q)$ form an exponential sequence of numbers between 1 to n. Note that these values are the defective numbers d for which $q^d = 1/2$. Then, the position in the sequence of pools where the test results switch from 0 to 1 hints to the value of d, subject to some constant factor and with some constant error probability.

Note that the expected competitive ratio of 2-stage or 3-stage group testing is determined by three quantities: the competitive ratio c' of the group testing strategy used in the last stage(s), and both the query number and competitive ratio of the BDNNGT strategy. With the above notations, the expected total query number is $(g/d + c'c)d \log n$. The currently best randomized 2-stage group testing strategy [2] uses $(1.44 + o(1))d \log n$ queries (asymptotically for $n \to \infty$). Moreover, in the successful case $d' \geq d$, the searcher can be sure that actually all defectives have been found, based on properties of this strategy from [2].

In this paper we focus on the challenge of getting a best possible tradeoff between parameters ϵ, c, g in BDNNGT. In the aforementioned $(g/d + c'c)d \log n$ result, obviously the asymptotic competitive ratio for growing d is $c'c$, and the worst-case competitive ratio is $g + c'c$, attained when $d = 1$. We can make c arbitrarily close to 1, at cost of a large g, but for minimizing the worst-case competitive ratio $g + c'c$ we have to balance the number $g \log n$ of pools and the expected ratio $c = d'/d$.

Besides its use in competitive group testing, BDNNGT may also be of independent interest: If samples come from the same source and are not distinguished (for example, water samples from the same lake without particular consideration of the exact place, or food samples from the same shop), we may only be interested in the amount of contaminated samples rather than their identities.

A first obvious question is whether $O(\log n)$ tests are really needed to solve BDNNGT. Intuitively this should be expected, based on the following heuristic argument. “Remote” queries with $1/\log_2(1/q)$ far from d will almost surely have a fixed result (0 or 1), thus they contribute very little information about the precise location of d. Therefore we must have queries with values $1/\log_2(1/q)$ within some constant ratio of every possible d, which would imply an $\Omega(\log n)$ bound. However, the searcher may use the accumulated information from all queries, and even though “unexpected” results of the remote queries have low probabilities, a few such events might together reveal enough useful information about d. Apparently, in order to turn the intuition into a proof we must somehow quantify the influence of remote queries and show that they actually provide too little information in total. To see the challenge, we first remark that the simple information-theoretic argument falls short. Imagine that we divide the interval from 1 to n into exponentially growing segments. Then the problem of estimating d up to a constant factor is in principle (up to some technicalities) equivalent to guessing the segment where d is located. The number of possible outcomes is some $\log n$, thus we need $\Omega(\log\log n)$ queries, which is a very weak lower bound. The next idea that comes into mind is to take the very different probabilities of binary answers into account. The entropy of the distribution of result strings is low, however it is not easy to see how to translate entropy into a measure suited to our problem.

One main result of the present paper is a proof of the $\Omega(\log n)$ query bound for any fixed c and ϵ, at least when pools are constructed "elementwise" as proposed above. A key ingredient is a suitable influence measure for queries. The proof is built on a simpler auxiliary problem that may deserve independent interest: deciding on one of two hypotheses based on probabilistic information, thereby respecting a pair of error bounds. It has to be noticed that our result does not yet prove the non-existence of a randomized $o(\log n)$ query strategy in general. The result only refers to randomized pools constructed in the aforementioned simple way: adding every element to a pool independently with some fixed probability $1-q$. However, the result gives strong support to the conjecture that $\Omega(\log n)$ is also a lower bound for any other randomized pooling design for our problem. Intuitively, randomized pools that treat all elements symmetrically and make independent decisions destroy all possibilities for a malicious adversary to mislead the searcher by some clever placement of defectives. Therefore it is hard to imagine that other constructions could have benefits.

The later part of the paper is devoted to practical methods to make c as small as possible, using a given number $g\log n$ of queries and respecting a given failure probability ϵ. Getting an optimal tradeoff turns out to be a highly nontrivial problem in itself. One approach is a linear programming (LP) formulation. When ϵ and a sequence of pools of fixed sizes are given, minimizing c is an LP. However, the LP formulation works for a given number n of elements. In order to obtain also upper bounds on c for $n \to \infty$ (and fixed g and ϵ) we use an "infinitary extension" of BDNNGT inspired by ideas from the lower-bound proof and come up with a nonlinear constraint optimization problem that can still be treated in practice, using standard software packages. In particular, we use a sequence of pool sizes with some nice invariance property, and we have reason to conjecture that our tradoffs are already nearly optimal.

The rest of the paper is organized as follows. In Section 2 we give a formal problem statement and some useful notation. In Section 3 we study a probabilistic inference problem on two hypotheses, and we define the influence of the random bit contributed by any query. This is used in Section 4 to prove the logarithmic lower bound for estimating the defective number by group tests. Then we derive particular $O(\log n)$ query strategies for estimating the defectives. In Section 5 we propose an LP formulation, and in Section 6 we argue for translation-invariant pool sizes and introduce a method to calculate competitive ratios in the limit, i.e., for $n \to \infty$. Section 7 reports some numerical results, and Section 8 concludes the paper with open theoretical questions.

2 Preliminaries

We study the following problem abstracted from BDNNGT.

Problem 1: Given are positive integers n and L, some positive error probability $\epsilon < 1$, and some $c > 1$ that we call the competitive ratio. Furthermore, an "invisible" number $x \in [1, n]$ is given. A searcher can prepare L nonadaptive queries to an oracle as follows. A query specifies a number $q \in (0, 1)$, and the oracle answers 0 with probability q^x, and 1 with probability $1 - q^x$. Based on the string s of these L binary answers the searcher is supposed to output some number x' such that $\mathbf{Pr}[x' < x] \le \epsilon$ and $\mathbf{E}[x'/x] \le c$ holds for every x.

The actual problem is to place the L queries, and to compute an x' from s, in such a way that the demands are fulfilled. The optimization version asks to minimize c given the other input parameters. We will prove that $L = \Omega(\log n)$ queries are needed, for any fixed ϵ and c. Note that randomness is not only in the oracle answers but possibly also in the rule that decides on x' based on s, and even in the choice of queries.

Symbols $\mathbf{Pr}$ and $\mathbf{E}$ in the definition refer to the resulting probability distribution of x' given x. Note that no distribution of x is assumed, rather, the conditions shall be fulfilled for any fixed x. We might, of course, define similar problem versions, e.g., with two-sided errors or with worst-case (rather than expected) competitive ratio and tail probabilities. However we stick to the above problem formulation, as it came up in this form in competitive 2-stage and 3-stage group testing, and other conceivable variations would behave similarly. The connection to BDNNGT is that an oracle query represents a randomized pool where every element is selected independently with probability $1 - q$, and x is the unknown number of defectives. In Problem 1 we allow real-valued x, which does not change anything asymptotically (for $n \to \infty$) but simplifies several technical issues.

Recall that both the construction of pools and the rule that decides d' having seen s may be randomized. A deterministic strategy for BDNNGT with $O(\log n)$ pools and guaranteed success ($\epsilon = 0$) is impossible:

Theorem 1 *No deterministic nonadaptive group testing strategy with $O(\log n)$ queries can always output a number d' with $d \le d' \le cd$, where d is the true number of defectives and c any prescribed constant.*

Proof. Assume we have such a strategy. Then we can determine d' and, since $d' \ge d$, apply a deterministic 2-stage strategy (e.g., from [2]) to find the d defectives, using $O(d' \log n) = O(cd \log n) = O(d \log n)$ further queries. But this contradicts our result in [7] that no deterministic strategy can find an unknown number d of defectives using $O(d \log n)$ queries in constantly many stages. $\diamond$

This negative result motivates the nonzero failure probability in BDNNGT (and in Problem 1).

It turns out that some coordinate transformations reflect the geometry of Problem 1 better than the variables originating from BDNNGT. We will look at x on the logarithmic axis and reserve symbol y for $y = \ln x$. Note that $y \in [0, \ln n]$. Furthermore, we relate every q to that value y which would make $q^x = q^{e^y}$ some constant "medium" probability, such as $1/e$, the inverse of Euler's constant. (The choice of this constant is arbitrary, but again it will simplify some expressions.) We denote this y value by t, in other words, we want $q^{e^t} = 1/e$, which also means $q = e^{-e^{-t}}$ and $\ln(1/q) = e^{-t}$. Symbol t is reserved for this transformed q. We refer to t as a *query point*. See Figure 1.

3 Probabilistic Inference of one-out-of-two Hypotheses

A well-known story tells that Buridan's ass could not decide on either a stack of hay or a pail of water and thus suffered from both hunger and thirst. The following problem also demands a decision between two alternatives either of which could be wrong, but it also offers a clear rationale for the decision. As usual in inference problems, the term "target" refers to the true hypothesis. The connection to Problem 1 will be made later.

Problem 2: The following items are given: two hypotheses g and h; two nonnegative real numbers $\epsilon, \delta < 1$; furthermore N possible observations that we simply denote by indices $s = 1, \ldots, N$; probabilities p_s to observe s if g is the target, and similarly, probabilities q_s to observe s if h is the target. Clearly, $\sum_{s=1}^N p_s = 1$ and $\sum_{s=1}^N q_s = 1$. Based on the observed s, the searcher can infer g with some probability x_s, and h with probability $1 - x_s$. The searcher's goal is to choose her x_s for all s, so as to limit to ϵ the probability of wrongly inferring h when g is the target, and to limit to δ the probability of wrongly inferring g when h is the target.

We rename the observations so that $p_1/q_1 \leq \ldots \leq p_N/q_N$.

In the optimization version of Problem 2, only one error probability, say ϵ, is fixed, and the searcher wants to determine $x_1, \ldots, x_N$ so as to minimize δ. We denote the optimum by $\delta(\epsilon)$. Problem 2 is easily solved in a greedy fashion:

Lemma 2 *A complete scheme of optimal strategies (one for every ϵ) for Problem 2 is described as follows. Determine u such that $p_1 + \ldots + p_{u-1} \leq \epsilon < p_1 + \ldots + p_u$, and let $f := (\epsilon - p_1 - p_2 - \ldots - p_{u-1})/p_u$. Infer h if $s < u$, infer h with probability f in case $s = u$, and otherwise infer g. Consequently, $\delta(\epsilon) = (1 - f)q_u + q_{u+1} + \ldots + q_N$.*

Proof. We only have to prove optimality. In any given strategy, let us change two consecutive "strategy values" simultaneously by $x_s := x_s - \Delta_s$ and $x_{s+1} := x_{s+1} + \Delta_{s+1}$, for some $\Delta_s, \Delta_{s+1} > 0$. If the target is g, this manipulation changes the probability to wrongly infer h by $p_{s+1}\Delta_{s+1} - p_s\Delta_s$. If the target is h, this manipulation changes the probability to wrongly infer g by $q_{s+1}\Delta_{s+1} - q_s\Delta_s$. We choose our changes so that the first term is zero, that is, $\Delta_s/\Delta_{s+1} = p_{s+1}/p_s$. Now $q_{s+1}/q_s \leq p_{s+1}/p_s = \Delta_s/\Delta_{s+1}$ shows $q_{s+1}\Delta_{s+1} - q_s\Delta_s \leq 0$, hence we only improved the strategy. The manipulation is impossible only if some index u exists with $x_s = 0$ for all $s < u$, and $x_s = 1$ for all $s > u$. Now the lemma follows easily. $\diamond$

Lemma 2 also implies:

Corollary 3 *$\delta(\epsilon)$ is a monotone decreasing and convex (i.e., sub-additive), piecewise linear function with $\delta(0) = 1$ and $\delta(1) = 0$.* $\diamond$

The following technical lemma shows that certain small additive changes in the probability sequences do not change the error function much (which is quite intuitive). In order to avoid heavy notation we give the proof in a geometric language, referring to a coordinate system with abscissa ϵ and ordinate δ. See Figure 2.

Lemma 4 *Consider the following type of rearrangement of a given instance of Problem 2. Replace every p_s with $p_s - \rho_s$, where $\sum_s \rho_s = \rho$. Similarly, replace every q_s with $q_s - \tau_s$, where $\tau_s = \rho_s q_s / p_s$ and $\sum_s \tau_s = \tau$. Then add the removed probability masses, in total ρ and τ, arbitrarily to existing pairs (p_s, q_s) or create new pairs (p_s, q_s), but in such a way that $\sum_s p_s = 1$ and $\sum_s q_s = 1$ are recovered. If such a rearrangement reduces $\delta(\epsilon)$, then the decrease is at most τ.*

Proof. By Corollary 3, the curve of function $\delta(\epsilon)$ is a chain of straight line segments whose slopes $-\delta'(\epsilon)$ get smaller from left to right, and these slopes are the ratios q_s/p_s. The rearrangement has the following effect on the curve: Pieces of the segments are cut out, whose horizontal and vertical projections have total length ρ and τ, respectively. Then their horizontal and vertical lengths may increase again by re-insertions (and all these actions may change the slopes of existing segments), and possibly new segments are created. Finally all segments are assembled to a new chain connecting the points $\delta(0) = 1$ and $\delta(1) = 0$, and having a monotone sequence of slopes again.

Consider a fixed ϵ. Let ρ_0 and τ_0 be the total horizontal and vertical length, respectively, of the pieces cut out to the left of ϵ. Let ρ_1 and τ_1 be defined similarly for the pieces to the right of ϵ. The largest possible reduction of $\delta(\epsilon)$ appears if: (a) some new vertical piece of length τ_1 forms the left end, and (b) some new horizontal piece of length ρ_0 and forms the right end of the modified curve. Note that pieces in (a) were originally located below $\delta(\epsilon)$, and pieces in (b) were originally located to the left of ϵ. This moves the remainder of the original curve (a) down by τ_1 length units, and (b) to the left by ρ_0 length units. The vertical move (a) reduces $\delta(\epsilon)$ by τ_1. The horizontal move (b) causes that the new function value at ϵ is the old function value at $\epsilon + \rho_0$. Since the slopes decrease from left to right, the slope at our fixed ϵ (and to the right of it) can be at most τ_0/ρ_0. Thus, move (b) reduces $\delta(\epsilon)$ by at most $\rho_0\tau_0/\rho_0 = \tau_0$. Finally note that $\tau_1 + \tau_0 = \tau$. $\diamond$

One should not be confused by the fact that ρ does not appear in the decrease bound: As we have chosen to consider δ as a function of ϵ, the setting is not symmetric.

We are particularly interested in the special case of Problem 2 where the $N = 2^L$ observations s are strings of L independent bits.

Problem 3: The following items are given: two hypotheses g and h; two nonnegative real numbers $\epsilon, \delta \leq 1$; and 2^L possible observations described by binary strings $s = s_1 \ldots s_L$. Furthermore, for $k = 1, \ldots, L$, we are given the probability a_k to observe $s_k = 0$ if the target is g, and the probability b_k to observe $s_k = 0$ if the target is h. The s_k are independent. The rest of the problem specification is as in Problem 2.

Clearly, our probabilities p_s and q_s evaluate to

$$p_s = \prod_{k=1}^{L} ((1 - s_k)a_k + s_k(1 - a_k)),$$

$$q_s = \prod_{k=1}^{L} ((1 - s_k)b_k + s_k(1 - b_k)).$$

Since the greedy algorithm in Lemma 2 applies also to Problem 3, a complete set of optimal strategies is described as follows: Infer h for p_s/q_s below some threshold, infer g for p_s/q_s above that threshold, and infer g or h randomized (with some prescribed probability) for p_s/q_s equal to that threshold.

Remark: Since the p_s and q_s are just products of certain probabilities a_k or $1-a_k$, and b_k or $1-b_k$, respectively, taking the logarithm reveals a nice and simple geometric structure of the optimal strategies from Lemma 2: Note that

$$\log(p_s/q_s) = \sum_{k=1}^{L}((1-s_k)(\log a_k - \log b_k) + s_k(\log(1-a_k) - \log(1-b_k))).$$

Since log is a monotone function, comparing the p_s/q_s with some threshold is equivalent to comparing the $\log(p_s/q_s)$ with some threshold. In other words, the decision for g or h is merely a linear threshold predicate. We will not need this remark in our lower-bound proof, still it might be interesting to notice.

In the following we consider any fixed $\epsilon > 0$, and all notations are understood with respect to this fixed error bound. Now think of our L independent bits as $L-1$ bits plus a distinguished one, say the kth bit. We define the *influence* of this kth bit as the decrease of $\delta(\epsilon)$, that is, the difference to the $\delta(\epsilon)$ value accomplished by an optimal strategy when the kth bit is ignored. Trivially, $\delta(\epsilon)$ can only decrease when more information is available.

Lemma 5 *With the above notations for Problem 3, the influence of the kth bit is at most* $\min(\max(a_k, b_k), \max(1-a_k, 1-b_k))$.

Proof. The kth bit splits every old observation s, consisting of the $L-1$ other bits and generated with probabilities p_s, q_s depending on the target, in two new observations. Their new probability pairs are obviously $(p_s a_k, q_s b_k)$ for $s_k = 0$, and $(p_s(1-a_k), q_s(1-b_k))$ for $s_k = 1$. In order to apply Lemma 4 we can view this splitting of observations as cutting out pieces from the segment of slope q_s/p_s of the $\delta(\epsilon)$ curve in the following way. If $q_s/p_s \le b_k/a_k$, a piece of vertical length $q_s b_k$ is cut out. If $q_s/p_s > b_k/a_k$, a piece of horizontal length $p_s a_k$ is cut out, corresponding to a piece of vertical length $p_s a_k q_s/p_s = q_s a_k$. (Note that we must first "cut out enough length" in both directions, therefore this case distinction is needed.) This is done for all old s. Since, of course, the old q_s sum up to 1, we have $\tau \le \max(a_k, b_k)$. The same reasoning applies to $1-a_k, 1-b_k$, thus we have $\tau \le \max(1-a_k, 1-b_k)$ as well. $\diamond$

Note that the influence bound in Lemma 5 is expressed only in terms of the probabilities of the respective bit being 0/1, conditional on the hypothesis. Hence we can independently apply Lemma 5 to each of the bits, no matter in which order they are considered, and simply add the influence bounds of several bits (similarly to a union bound of probabilities).

4 The Logarithmic Lower Bound

We further narrow down our one-out-of-two inference problem to a special case of Problem 3. (Below we reuse symbol q, without risk of confusion.)

Problem 4: The following items are given: two hypotheses r and 1. where $r > 1$ is a fixed real number; two nonnegative real numbers $\epsilon, \delta \leq 1$, furthermore 2^L possible observations described by binary strings $s = s_1 \ldots s_L$. For $k = 1, \ldots, L$, let q_k^x be the probability to observe $s_k = 0$ if the target is x. We also speak of a "query at q_k". The s_k are independent. The rest of the problem specification is as before. In particular, let ϵ be the probability of wrongly inferring 1 although r is the target, and let δ be the probability of wrongly inferring r although 1 is the target.

Now hypothesis $x = r$ generates the string s with probability

$$\prod_{k=1}^{L} ((1 - s_k)q_k^r + s_k(1 - q_k^r)),$$

and hypothesis $x = 1$ generates s with probability

$$\prod_{k=1}^{L} ((1 - s_k)q_k + s_k(1 - q_k)),$$

in other words, $a_k = q_k^r$ and $b_k = q_k$. As earlier we fix some error bound ϵ. From Lemma 5 we get immediately:

Lemma 6 *With the above notations for Problem 4, the influence of a query at q is at most* $\min(q, 1 - q^r)$. $\diamond$

Problem 4 was stated, without loss of generality, for hypotheses r and 1. Similarly we may formulate it for hypotheses rx and x (for any positive x), which merely involves a coordinate transformation. We speak of the "influence of q on x" when we mean the influence of a query at q, with respect to Problem 4 for hypotheses rx and x. Clearly, the influence of q on x equals the influence of q^x on 1. Therefore Lemma 6 generalizes immediately to:

The influence of q on x is at most $\min(q^x, 1 - q^{rx})$.

Remember $y := \ln x$ from Section 2. By a slight abuse of notation, the phrase "influence of q on y" refers to the logarithmic coordinates, and Lemma 6 gets this form:

The influence of q on y is at most $\min(q^{e^y}, 1 - q^{re^y})$.

While q^{e^y} obviously decreases doubly exponentially with growing $y > 0$, it is also useful to have a simple upper bound for $1 - q^{re^y}$ when $y < 0$. Since $1 - e^{-z} \leq z$ for any variable z, we take z with $e^{-z} = q^{re^y}$ to obtain $1 - q^{re^y} \leq z = -\ln q^{re^y} = \ln(1/q)re^y$. Now we have:

The influence of q on y is at most $\min(q^{e^y}, \ln(1/q)re^y)$.

Finally we also transform q into t as introduced in Section 2, and we speak of the "influence of t on y", denoted $I_t(y)$. With $q = e^{-e^{-t}}$ and $\ln(1/q) = e^{-t}$, our influence lemma is in its final shape:

Lemma 7 $I_t(y) \leq \min(e^{-e^{y-t}}, re^{y-t})$. $\diamond$

From this bound we get:

Lemma 8 *For every fixed t we have $\int_0^{\ln n} I_t(y)\,dy = \Theta(\ln r)$.*

Proof. For simplicity we bound the integral over the entire real axis. (Since $I_t(y)$ decreases rapidly on both sides of t, this is not even too generous.) The advantage is that we can assume $t = 0$ without loss of generality. We split the integral in two parts, at $y = -\ln r$. As $I_t(y)$ is a minimum of two functions, we can take either of them as an upper bound. Specifically we get $\int_{-\infty}^{\infty} I_t(y)\,dy < \int_{-\infty}^{-\ln r} re^y\,dy + \int_{-\ln r}^{\infty} e^{-e^y}\,dy = \int_{\ln r}^{\infty} re^{-y}\,d(-y) + \int_{-\ln r}^{\infty} e^{-e^y}\,dy = re^{-\ln r} + \Theta(\ln r) = 1 + \Theta(\ln r)$. The second integral is $\Theta(\ln r)$ since both $e^{-e^{-\ln r}} = e^{-1/r}$ and (for instance) $e^{-e^0} = e^{-1}$ are between some positive constants, the function is monotone decreasing, and $\int_0^{\infty} e^{-e^y}\,dy = \Theta(1)$. $\diamond$

The next lemma connects our "bipolar" number guessing problem to the problem we started from.

Lemma 9 *For every $r > 1$ and $0 < \delta < 1$ we have: Any strategy solving Problem 1 with error probability ϵ and competitive ratio $c := 1 + (r-1)\delta$ yields a strategy solving Problem 4 with hypotheses rx and x, for every $x \le n/r$, with error probabilities ϵ and δ.*

Proof. Imagine a searcher wants to solve an instance of Problem 1, and an adversary tells her that the target is either rx or x. Despite this strong help, in case that rx is the target, the searcher must still guess rx subject to an error probability ϵ, due to the definition of Problem 1. In the other case when the target is x, error probability δ means a competitive ratio of $(1-\delta) + r\delta = 1 + (r-1)\delta$. $\diamond$

We are ready to state the main result of this section:

Theorem 10 *Any strategy for Problem 1, with fixed error probability ϵ and competitive ratio c, needs $\Omega(\ln n/\ln c)$ queries, where the constant factor may depend on ϵ.*

Proof. Fix some $r > c$ and $\delta = (c-1)/(r-1)$, hence $c = 1 + (r-1)\delta$. We choose $r = \Theta(c)$ large enough so that $D := 1 - \epsilon - \delta$ is positive. Due to Lemma 9, the set of queries must be powerful enough to solve Problem 4 with hypotheses rx and x, for every $x \le n/r$, with error probabilities ϵ and δ. In the case of no queries, the error tradeoff at every x would be simply $\delta(\epsilon) = 1 - \epsilon$. Since we need to reduce $\delta(\epsilon)$ down to our fixed δ, all queries together must have an influence at least $1 - \epsilon - \delta$ on x. In transformed coordinates this means $\sum_t I_t(y) \ge D$ for all $0 \le y \le \ln n - \ln r$, where the sum is taken over all t in our query set (multiple occurrences counted). Hence $\int_0^{\ln n - \ln r} \sum_t I_t(y)\,dy \ge D(\ln n - \ln r)$. Since Lemma 8 states $\int_0^{\ln n - \ln r} I_t(y)\,dy = \Theta(\ln r)$ regardless of t, the number of queries is at least $(\ln n - \ln r)D/\Theta(\ln r) = \Omega(\ln n/\ln r)$. $\diamond$

Note that this integration argument also applies if the queries themselves are located according to some probability distribution, that is, Theorem 10 also holds for "fully randomized" strategies.

Theorem 10 only shows that the query number is logarithmic, for any fixed parameter values. But the proof method is not suited for deriving also good lower bounds on the hidden constant factor. For instance, this factor should increase to infinity when ϵ tends to 0. To reflect this behaviour in the lower bound, apparently the previous proof must be combined with some reduction between problem instances with different ϵ. We leave this topic here. Anyways, in practice one would apply some reasonable standard value like $\epsilon = 0.05$ rather than trading much more queries for smaller failure probabilities. A more relevant question, addressed in the remainder of the paper, is which upper bounds we can accomplish.

5 A Linear Program Formulation of BDNNGT

In this section we derive an LP formulation for the inference of d' from the observed string s of test results in BDNNGT, provided that the (randomized) construction of the pools is fixed. For convenience we use symbols i, j instead of d, d' (true and predicted number of defectives) because they will have the role of indices of other variables.

We denote by q_{ki} the conditional probability that the kth pool is negative if the number of defectives is i. Of course, the q_{ki} depend on how the pools are constructed. The conditional probability p_{si} to observe result string s if the number of defectives is i is easy to express in terms of the q_{ki}, due to the independence of pools:

$$p_{si} = \prod_{k=0}^{L-1} ((1 - s_k)q_{ki} + s_k(1 - q_{ki})). \tag{1}$$

Recall that i, j are in the range $1, \dots, n$. Since the string s is the only (one-shot) information that the searcher obtains, a strategy for inferring j is now completely characterized by the system of conditional probabilities x_{js} to choose j if string s was observed. Define $c_{ij} = j/i$. Now BDNNGT for the given pools can be translated, with some care, into an LP with variables x_{js} and c:

$$\min\ c \tag{2}$$

$$\sum_s p_{s1} \sum_j c_{1j} x_{js} \le c \tag{3}$$

$$\forall i > 1 : \ \sum_s p_{si} \sum_{j \ge i} c_{ij} x_{js} \le (1 - \epsilon)r \tag{4}$$

$$\forall i > 1 : \ \sum_s p_{si} \sum_{j < i} x_{js} \le \epsilon \tag{5}$$

$$\forall j, s : \ x_{js} \ge 0 \tag{6}$$

$$\forall s : \ \sum_j x_{js} = 1 \tag{7}$$

In the following we use c_i and ϵ_i as shorthands for the left-hand side of constraint (4) and (5), respectively. The following Proposition basically says that the LP is correct:

Proposition 11 *For any fixed system of pools, e.g., probabilities p_{si}, we have: The system of values x_{js} returned by the LP (2)–(7) describes a strategy for BDNNGT, and the returned objective value c is a valid upper bound on the expected competitive ratio j/i, as defined in the statement of BDNNGT, for any $i = 1, \dots, n$. Moreover, if constraints (5) turn out to be binding, c is exactly the smallest possible competitive ratio for the given system of pools.*

Proof. Constraints (6)–(7) only say that the x_{js} form a probability distribution conditional on s, for each s. Clearly, ϵ_i is the probability to output some $j < i$, with i being the true number of defectives. Notice the special case $\epsilon_1 = 0$. For $i > 1$, the expected ratio j/i, conditional on $j \geq i$, is $c_i/(1-\epsilon_i)$. Our actual objective in BDNNGT is to minimize $\max_i c_i/(1-\epsilon_i)$. But since (5) enforces $\epsilon_i \leq \epsilon$, we have $c_i/(1-\epsilon_i) \leq c_i/(1-\epsilon) \leq c$ due to (4). As for $i = 1$, the expected $j/i = j$ is simply c_1 which is at most c by (3). Altogether, the two assertions follow from these inequalities. $\diamond$

We remark that we cannot simply replace ϵ with ϵ_i in (4), since c is also a variable, and then (4) would no longer be a linear constraint. Finally notice that $\max_i c_i/(1-\epsilon)$ would not be a linear objective function, but by using a standard trick we have replaced, in (3), the $\max_i$ operator with the auxiliary variable c that we minimize in (2). An alternative LP formulation would fix c to a constant, which avoids this "ϵ_i problem", and then use an arbitrary (meaningless) objective function. Then the LP only decides the existence of a feasible solution. The drawback is that we have to run that LP version many times in order to find the optimal c by, e.g., binary search. Thus we stick to the above LP version.

In our empirical results (for pool sizes discussed later on) we consistently observed the following properties of our LP: (i) The c_i are strictly monotone decreasing: $c_1 > \ldots > c_m$. (ii) $\epsilon_i = \epsilon$ for all $i > 1$, that is, the maximum failure probability ϵ is exhausted, for every $i > 1$. (iii) Most strings s return a unique j with $x_{js} = 1$, and for the few strings s that return several possible j (with $x_{js} > 0$), these j are a few consecutive integers. – Although these properties are intuitive, we can *prove* only partial results that hint to them. Since they would be of minor interest, we skip these partial results here and leave proofs (or counterexamples) as an open problem.

6 Translation-Invariant Pooling Designs for BDNNGT

In our LP it remains to fix the q_{ki}. As mentioned in the Introduction, we put each element independently with the same probability in the kth pool. For each k, let q_k be the probability *not* to put an element in the kth pool. Obviously this yields $q_{ki} = q_k^i$. Furthermore, our $\Omega(\log n)$ lower bound proof suggests that query points should divide the logarithmic axis of defective numbers equidistantly; we also speak of translation invariance. Therefore we fix some ratio $b > 1$ and use probabilities such that $q_{k-1} = q_k^b$ for all k. In other words, we choose $q_k = q^{b^{-k}}$, for $k = 0, \ldots, L-1$, and some fixed $q < 1$. Note that, asymptotically, $L = \log n / \log b$, that is, $g = 1/\log b$. We discuss experimental results in a separate section below.

The LP yields an upper bound for the competitive ratio c (given ϵ and L) when also the number n of elements is fixed, and it yields the best c for a given sequence of pool sizes. For trivial reasons c increases with n (when ϵ and $g = L/\log n$ are fixed). Thus it would also be interesting to know the limit for $n \to \infty$. As a consequence of the quickly decreasing influence of queries with pool sizes far from n/d one can show that $\lim_{n\to\infty} c$ is always finite (where c denotes the optimum). However, experiments suggest that the convergence is slow, and the larger n is, the more variables are needed in the LP, so that we cannot reach accurate limit results from the LP. Therefore we also developed an alternative tool to address this question. For this purpose we consider the following "infinite extension" of Problem 1. This has merely formal reasons that will be explained below.

Problem 5: Given are some positive error probability $\epsilon < 1$, some $c > 1$ that we call the competitive ratio, and an "invisible" number x which can be any real number. A searcher can prepare countably infinitely many nonadaptive queries to an oracle as follows. A query specifies a number $q \in (0, 1)$, and the oracle gives answer 0 with probability q^x and answer 1 with probability $1 - q^x$. Based on the infinite string s of the binary answers, the searcher is supposed to output some number x' such that $\mathbf{Pr}[x' < x] \le \epsilon$ and $\mathbf{E}[x'/x] \le c$ holds for every x.

For Problem 5 we naturally consider the *density* of queries, i.e., the number of queries per length unit on the logarithmic axis, corresponding to $L/\ln n$ in Problem 1. We withhold a precise formal definition of density, because for our upper bounds we will only consider translation-invariant strategies, so that the notion of density is straightforward. We want to minimize c, given ϵ and the density.

Remember that $y = \ln x$, and every query, with probability q of responding with 0, is matched to a query point t on the logarithmic axis through $q = e^{-e^{-t}}$. If y is the unknown target value (in logarithmic coordinates), the probability of answer 0 to a query at point t is $q^x = e^{-e^{y-t}}$. We place the query points t equidistantly, at points $t = ju + v$, where u is a fixed space (thus u^{-1} is the density), j loops over all integers, and v is a random shift being uniformly distributed in $[0, u)$. For every two-sided infinite binary sequence s of answers we also specify an y_s such that the output $y' = \ln x'$ is located y_s length units to the right of the point of the leftmost answer 0 in s (see details below). One should not worry about the uncountably infinitely many s; in practice we "cut out a finite segment" of this infinite strategy according to:

Lemma 12 *Any translation-invariant strategy for Problem 5 with bounds ϵ and c and density u^{-1} yields a strategy for the original Problem 1 that has asymptotically, i.e., for $n \to \infty$, the same characteristics as the given strategy: error probability ϵ, competitive ratio c, and $u^{-1} \ln n$ queries.*

Proof. We simply take the query points in the interval $[0, \ln n]$ plus some margins on both sides, whose lengths grow with n but slower than $\ln n$. Since even the total influence of the (infinitely many!) ignored queries on any point y, $0 \le y \le \ln n$, decreases exponentially with the margin length, the resulting finite strategy performs as the original strategy for Problem 5, subject to terms that vanish for $n \to \infty$. $\diamond$

The reason for replacing Problem 1 with Problem 5 is its greater elegance. This way we avoid case distinctions with respect to the position of y. In particular, because of the equal distances and the random shift, we can now assume without loss of generality that $y = 0$, and the searcher does not know the shift of the coordinates. (This is an equivalent view of the real situation where the searcher knows the coordinate system but not y.) We remark that the random shift cannot make the upper bounds worse: If, in any strategy for Problem 5, the query points are first shifted randomly, then the resulting strategy still respects the bounds ϵ and c at every y, if the original strategy did.

Next we describe how to calculate the optimal values y_s for our specific strategy. We need to consider only those two-sided infinite strings s that have a leftmost 0 and a rightmost 1.

We call the segment bounded by these positions the significant segment. Clearly, all other response strings appear with probability 0. We (arbitrarily) index the bits in each s so that $s_0 = 0$ is the leftmost 0, that is, $s_k = 1$ for all $k < 0$. The point on the y-axis where the leftmost query t with answer 0 is located is called the reference point.

The probability density of the event that string s appears, and its reference point is $ju+v$ (j integer, $0 \leq v < u$), is given by

$$f_s(ju+v) := u^{-1} \prod_k \Big((1-s_k) e^{-e^{-(k+j)u-v}} + s_k (1 - e^{-e^{-(k+j)u-v}}) \Big)$$

where k loops over all integers, and the s_k are the bits of s as specified above.

Since, for each s, our strategy returns the point located y_s units to the right of the reference point t, the contribution of string s to the error probability (of having output $y' < 0$) amounts to $\int_{-\infty}^{-y_s} f_s(t)\, dt$. Hence our goal is to minimize $\sum_s \int_{-\infty}^{+\infty} e^{t+y_s} f_s(t)\, dt$ under the constraint $\sum_s \int_{-\infty}^{-y_s} f_s(t)\, dt \leq \epsilon$. This together with Lemma 12 implies:

Proposition 13 *For any fixed u, the solution to the problem of minimizing the function $\sum_s \int_{-\infty}^{+\infty} e^{t+y_s} f_s(t)\, dt$ under the constraint $\sum_s \int_{-\infty}^{-y_s} f_s(t)\, dt \leq \epsilon$ yields an upper bound on the competitive ratio c for Problem 1 when $u^{-1} \ln 2 \cdot \log_2 n$ queries are used.* $\diamond$

Now these bounds can be calculated by standard nonlinear constraint optimization problem solvers. It suffices to consider some finite set of the most likely strings s whose sum of probabilities is close enough to 1.

7 Some Numerical Results and Practical Issues

7.1 Linear Program for BDNNGT

We have implemented our LP using GLPK and run it extensively for many parameter combinations. We present a few numerical results in Tables 1 and 2. They show the best competitive ratios c we could obtain for different g (first column) and $\epsilon = 0.01, \dots 0.05$ (in 0.01 steps) by chosing translation-invariant sequences. Note that g is meant as $1/\log b$ rather than $g = L/\log n$ which holds only asymptotically. Recall from Section 1 that the worst-case competitive ratio for 2-stage group testing is currently $g + 1.9c$. The results suggest that always some g around 2 minimize this objective. Clearly, the LP solutions tell us which estimates j to choose for a given test result string s (not displayed here).

For the following brief discussion we define the notion of a pattern: For any binary string s of test results, let the pattern $[s]$ be the substring of s from the leftmost 0 to the rightmost 1. In the special case $s = 1 \dots 10 \dots 0$ define $[s] = \lambda$ (the empty string). A left (right) *shift* of s is a string with the same pattern as s, having the pattern substring more to the left (right). For example, $s = 1111111101001000000$ has the pattern $[s] = 01001$.

The LP in its naive form becomes problematic for larger n, as the LP would then comprise too many variables. Note that $n2^L \approx n2^{g \log n} = n^{g+1}$ variables x_{js} are needed if we consider all binary strings s of length L. By the observation that long patterns are unlikely we can save variables. We can ignore in the LP the least frequent strings s such that their $\max_i \sum_s p_{si}$ remains below some ϵ', and run the LP for $\epsilon - \epsilon'$ (considering the occurrence of some rare string as failure). It can be shown that we keep only strings s whose patterns $[s]$ have some bounded length depending only on the fixed ϵ but not on n. This reduces the number of variables from $n2^L$ to some $O(n)$. For huge n, even this would be too much. A way to keep the number of variables manageable is to restrict i and j to a mildly exponential sequence of selected values and interpolate.

7.2 Asymptotic Competitive Ratios for BDNNGT

We implemented the method of Section 6 using the Matlab features `fmincon` for optimization and `quadgk` for numerical integration. As an illustration, Table 3 displays the competitive ratios for $\epsilon = 0.01, \dots 0.05$ and $g \log_2 n$ pools, for some g in 0.5 steps. Some c are even slightly smaller than in the corresponding LP results; a possible explanation is that our LP does not use a random shift in the pool sizes.

Of course, the optimizer also outputs the strategy variables y_s. For larger g the calculations become too time-consuming: The denser the query points are, the more strings s have non-negligible probabilities, and the resulting large number of variables leads to slow convergence. However, these technical issues can be resolved by more computational power. One should also bear in mind that a strategy, i.e., table of y_s values, needs to be computed only once for any given pair of input parameters g and ϵ, thus long waiting times are acceptable. Anyways, some optimality criterion for the problem could enable us to find the optimal strategies more efficiently than by naive direct use of an optimizer.

8 Conclusions and Open Questions

Saving expensive tests in chemical analytics and biological testing by combinatorial methods is a rewarding task. This paper is concerned with the special problem of estimating the number of "defective" samples by nonadaptive group tests, formally stated as BDNNGT. A main theoretical result is that $\Omega(\log n)$ queries are needed, if the single pools are formed in a natural way by independent random choices. While this bound is intuitive, it has not been proved before, and quite some technical efforts were needed. It remains open how to confirm (or disprove) this lower bound also for arbitrary pools.

The logarithmic lower bound also suggests that query points should be placed translation-invariant on the logarithmic axis of defective numbers. We gave such a strategy which allows numerical calculation of the predictions and competitive ratios, for any given query density and error bound. One could also think of other translation-invariant strategies, for instance, query points may be chosen by a Poisson process, however this seems worse because then the density of actual query points can accidentally be low around the target value. In summary we conjecture that our strategy in Section 6 is already optimal, with respect to the constant factors and parameters, among all possible randomized strategies.

For fixed pool sizes we also gave an LP formulation, leading to a practical solution method that only requires an LP solver. Still it would be nice to *prove* optimality of translation-invariant pooling designs, and the various structural properties we conjectured for the LP solutions in Section 5.

The interpretation of BDNNGT as a game where an adversary chooses i could also help solve the open problems, since results from the theory of matrix games might be used. Furthermore, it might be possible that the optimal strategies can be obtained purely combinatorially, for example by some greedy algorithm. In that case the LP formulation would only have an auxiliary role in the optimality proof. However, we found that some natural candidates for greedy algorithms are not confirmed by our empirical LP solutions.

Finally, our upper-bound methods are only numerical. A challenging question is whether the dependency of optimal competitive ratio, error probability and query number can be characterized in a closed analytical form.

Acknowledgments

This work has been supported by the Swedish Research Council (Vetenskapsrådet), grants no. 2007-6437, "Combinatorial inference algorithms – parameterization and clustering", and 2010-4661 "Generalized and fast search strategies for parameterized problems". We are indebted to the referees of COCOA 2010 for some helpful editorial remarks, Tadao Takaoka who also spotted some typos and did the presentation at COCOA as we were unable to come, and Vinay Jethava for some technical help.

References

[1] H.B. Chen, F.K. Hwang. Exploring the missing link among d-separable, $\bar{d}$-separable and d-disjunct matrices, *Discr. Appl. Math.* 155 (2007) 662–664

[2] Y. Cheng, D.Z. Du. New constructions of one- and two-stage pooling designs, *J. Comp. Biol.* 15 (2008) 195–205

[3] M. Cheraghchi. Noise-resilient group testing: Limitations and constructions, *17th Int. Symp. on Fundamentals of Computation Theory FCT 2009*, *LNCS* 5699, 62–73

[4] M. Cheraghchi. Improved constructions for non-adaptive threshold group testing, *37th Int. Coll. on Automata, Languages and Progr. ICALP 2010*, *LNCS* 6198, 552–564

[5] A.E.F. Clementi, A. Monti, R. Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks, *12th Symp. on Discr. Algor. SODA 2001*, 709–718

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

[7] P. Damaschke, A. Sheikh Muhammad. Competitive group testing and learning hidden vertex covers with minimum adaptivity, *Discr. Math. Algor. Appl.* 2 (2010), 291–311

[8] A. De Bonis, L. Gasieniec, U. Vaccaro. Optimal two-stage algorithms for group testing problems, *SIAM J. Comp.* 34 (2005), 1253–1270

[9] A. De Bonis, U. Vaccaro. Constructions of generalized superimposed codes with applications to group testing and conflict resolution in multiple access channels, *Theor. Comp. Science* 306 (2003), 223–243

[10] R. Dorfman. The detection of defective members of large populations, *The Annals of Math. Stat.* 14 (1943), 436–440

[11] D.Z. Du, F.K. Hwang, *Pooling Designs and Nonadaptive Group Testing*, World Scientific (2006)

[12] D. Eppstein, M.T. Goodrich, D.S. Hirschberg. Improved combinatorial group testing algorithms for real-world problem sizes, *SIAM J. Comp.* 36 (2007), 1360–1375

[13] A.C. Gilbert, M.A. Iwen, M.J. Strauss. Group testing and sparse signal recovery, *42nd Asilomar Conf. on Signals, Systems, and Computers 2008*, 1059–1063

[14] M.T. Goodrich, D.S. Hirschberg. Improved adaptive group testing algorithms with applications to multiple access channels and dead sensor diagnosis, *J. Comb. Optim.* 15 (2008), 95–121

[15] P. Indyk, H.Q. Ngo, A. Rudra. Efficiently decodable non-adaptive group testing, *20th Symp. on Discr. Algor. SODA 2010*, 1126–1142

[16] A.B. Kahng, S. Reda. New and improved BIST diagnosis methods from combinatorial group testing theory, *IEEE Trans. CAD of Integr. Circuits and Systems* 25 (2006), 533–543

[17] J. Schlaghoff, E. Triesch, Improved results for competitive group testing, *Combinatorics, Probability and Computing* 14 (2005), 191–202

Table 1: Competitive ratios for maximum 8 defectives.

g	e 0.01	e 0.02	e 0.03	e 0.04	e 0.05
0.5	5.81	5.43	5.17	4.87	4.58
1.0	4.14	3.65	3.37	3.14	2.97
1.5	3.17	2.79	2.58	2.43	2.35
2.0	2.68	2.43	2.30	2.20	2.13
2.5	2.47	2.27	2.15	2.06	2.00
3.0	2.41	2.22	2.11	2.02	1.96

Table 2: Competitive ratios for maximum 16 defectives.

g	e 0.01	e 0.02	e 0.03	e 0.04	e 0.05
0.5	8.26	7.26	6.64	6.21	5.86
1.0	4.71	4.07	3.67	3.43	3.28
1.5	3.46	3.07	2.87	2.71	2.59
2.0	3.00	2.67	2.50	2.39	2.30
2.5	2.76	2.47	2.32	2.23	2.15
3.0	2.69	2.36	2.22	2.14	2.07

Table 3: Competitive ratios for large n.

g	e 0.01	e 0.02	e 0.03	e 0.04	e 0.05
0.5	11.86	9.83	8.67	7.87	7.28
1.0	5.20	4.50	4.11	3.82	3.60
1.5	3.69	3.28	3.02	2.86	2.72
2.0	2.99	2.69	2.52	2.39	2.28

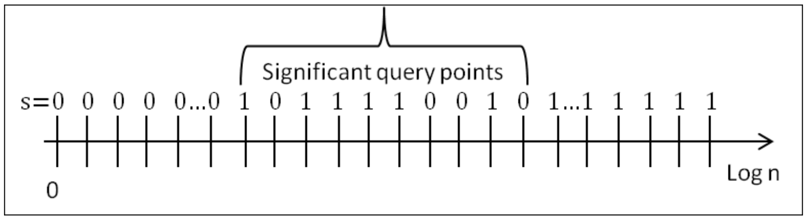

Figure 1: An illustration of the logarithmic axis, with query points and test results. The unknown number is somewhere in the middle of the picture, and only queries around this point give informative answers.

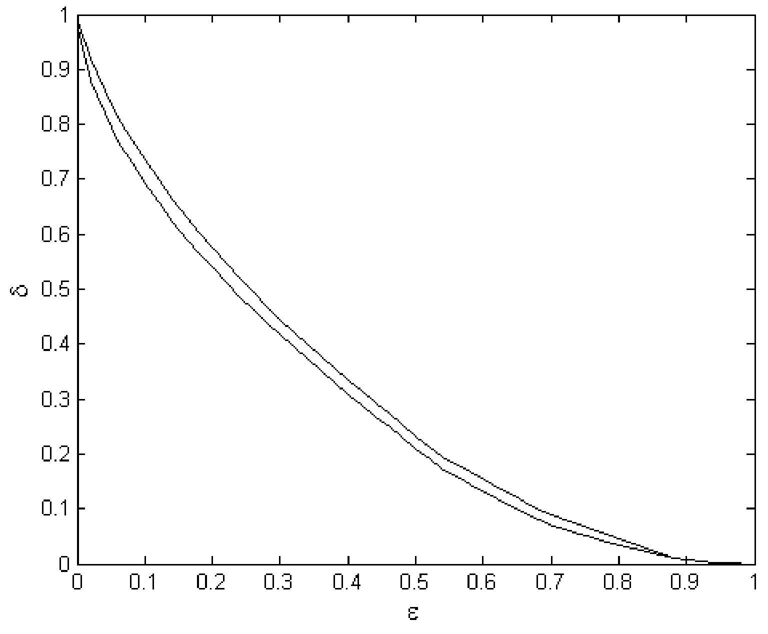

Figure 2: Example of a function $\delta(\epsilon)$ before and after small changes of the probabilities.