Competitive Group Testing and Learning Hidden Vertex Covers with Minimum Adaptivity[*]

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

Abstract

Suppose that we are given a set of n elements d of which have a property called defective. A group test can check for any subset, called a pool, whether it contains a defective. It is known that a nearly optimal number of $O(d\log(n/d))$ pools in 2 stages (where tests within a stage are done in parallel) are sufficient, but then the searcher must know d in advance. Here we explore group testing strategies that use a nearly optimal number of pools and a few stages although d is not known beforehand. We prove a lower bound of $\Omega(\log d/\log\log d)$ stages and a more general pools vs. stages tradeoff. This is almost tight, since $O(\log d)$ stages are sufficient for a strategy with $O(d\log n)$ pools. As opposed to this negative result, we devise a randomized strategy using $O(d\log(n/d))$ pools in 3 stages, with any desired success probability $1-\epsilon$. With some additional measures even 2 stages are enough. Open questions concern the optimal constant factors and practical implications. A related problem motivated by, e.g., biological network analysis is to learn hidden vertex covers of a small size k in unknown graphs by edge group tests. (Does a given subset of vertices contain an edge?) We give a 1-stage strategy using $O(k^3\log n)$ pools, with any parameterized algorithm for vertex cover enumeration as a decoder. During the course of this work we also provide a classification of types of randomized search strategies in general.

Keywords: nonadaptive group testing, competitive group testing, combinatorial search, randomization.
AMS subject classification: 68Q17 Computational difficulty of problems, 68Q32 Computational learning theory, 68Q87 Probability in computer science, 68W20 Randomized algorithms.

1 Background

The *group testing* problem is to find d elements called *positive*, or synonymously, *defective*, in a set X of size n by queries of the following type. The searcher can choose arbitrary subsets $Q\subset X$ called *pools*, and ask whether Q contains at least one defective. Non-defective elements are called *negative*. A *positive pool* is a pool containing some defective, thus responding Yes

[*]This is an extended version of a paper presented at the *17th International Symposium on Fundamentals of Computation Theory FCT 2009*, Wroclaw, *Lecture Notes in Computer Science* (Springer) 5699, pages 84-95.

to a group test. A *negative pool* is a pool without defectives, thus responding No to a group test.

Group testing has several applications, most notably in biological and chemical testing, but also in communication networks, information gathering, compression, streaming algorithms, etc., see for instance [9, 10, 15, 20, 21, 22] and further pointers therein.

Throughout this paper, log means $\log_2$ if no other base is mentioned. By the information-theoretic lower bound, at least $\log \binom{n}{d} \approx d\log(n/d)$ pools are needed to find d defectives even if the number d is known in advance, and it is an easy exercise to devise an adaptive query strategy using $O(d\log(n/d))$ pools. Here, a strategy is called *adaptive* if queries are asked sequentially, that is, every pool can be prepared based on the outcomes of all earlier queries. For many applications however, the time consumption of adaptive strategies is hardly acceptable, and strategies that work in a few *stages* are strongly preferred: The pools for every stage must be prepared in advance, depending on the outcomes of earlier stages, and then they are queried in parallel.

Any 1-stage strategy needs $\Omega(d^2 \log n/\log d)$ pools, as a consequence of the lower bound for $\bar{d}$-separable matrices [6] which are pooling designs that can distinguish between any two possible sets of at most d defectives. The same lower bound was already well known for d-disjunct matrices, i.e., special $\bar{d}$-separable matrices that also allow very simple decoding of the test results [18, 24]. On the other hand, $O(d^2 \log n)$ pools are sufficient. The currently best factor is 4.28; see [8] and the references therein. The first 2-stage strategy using a number of pools within a constant factor of optimum, more precisely $7.54\,d\log(n/d)$, was developed in [14] and later improved to essentially $4\,d\log(n/d)$ [19] and finally $1.9\,d\log(n/d)$, or even $1.44\,d\log(n/d)$ for large enough d [8]. These strategies use stage 1 to find $O(d)$ candidate elements including all defectives, which are then tested individually in stage 2. Such 2-stage strategies are called “trivial” (a misunderstandable but established term); they are of independent interest as an intermediate type of strategies between 1-stage and “truly 2-stage” strategies. Note that a trivial 2-stage strategy requires no subsequent decoding.

The 2-stage strategies still require the knowledge of an upper bound d on the number of defectives, and they guarantee an almost optimal query complexity only relative to this d which can be much larger than the true number of defectives in the particular case. As opposed to this, adaptive strategies with $O(d\log(n/d))$ pools do not need any prior knowledge of d. Beginning with [3, 16, 17], substantial work has been done to minimize the constant factor in $O(d\log(n/d))$, called the *competitive ratio*. The currently best results are in [25]. Our problem with unknown d was also raised in [22], and several batching strategies have been proposed and studied experimentally. To our best knowledge, the present work is the first to establish rigorous results for this question:

Can we take the best of two worlds and perform group testing without prior knowledge of d in a few stages, using a number of pools close to the information-theoretic lower bound? This question is not only of theoretical interest. If the number d of defectives varies a lot between the problem instances, then the conservative policy of assuming some “large enough” d systematically requires unnecessarily many tests, while a strategy with underestimated d even fails to find all defectives.

It is fairly obvious that a 1-stage strategy cannot do better than n individual tests. On the bright side, $O(\log d)$ stages are sufficient to accommodate a strategy with $O(d\log(n/d))$ pools: Simply double the assumed d in every other stage, and apply the best 2-stage strategy repeatedly, including a check if all defectives have been found. In this paper we prove that almost $\log d$ stages are really needed in the worst case. This clearly separates the complexity of the cases with known and unknown d.

In related work [11] we studied query strategies and the computational complexity of learning Boolean functions depending on only a few unknown relevant variables. Group testing is the special case where the Boolean function is already known to be the disjunction of the relevant variables.

One modern application of group testing is the reconstruction of biological networks, e.g., protein interaction networks, by experiments that signal the presence of at least one interaction in a "pool" of proteins. In some experimental setting, a group test signals that one fixed protein, called a *bait*, interacts with some protein in a pool. Hence the problem of finding precisely the interaction partners of a bait is just the group testing problem. Since the degrees d of vertices in interaction networks are very different and tests are time-consuming, we arrive again at the type of problem discussed above. In [13] we studied a similar prolem with more powerful, non-binary queries that return all interaction partners of a bait.

Instead of learning a whole graph, i.e., the neighbors of every vertex, we may want to learn only a small set of vertices that is incident to all edges, that is, a small *vertex cover*. In interaction networks, these vertices can be expected to play a major role, as a small vertex cover represents, e.g., a small group of proteins involved in all interactions [23]. Suppose that an *edge group test* is available that tells us, for a pool Q of vertices, whether some vertices in Q are joined by an edge. This assumption is also known as the complex model of group testing. Then we encounter the problem of *learning a hidden vertex cover*: Given a graph with a known vertex set but an unknown edge set, and a number k, identify a vertex cover of size at most k (or all of them), by using a possibly small number of edge group tests. Learning hidden structures in graphs has been intensively studied for many structures and query models, we refer to [1, 2, 4] for recent results and a survey. Learning a hidden star [1] is a related but quite different problem.

Note that the vertex cover problem, where the graphs is given and known from the beginning, is NP-complete, on the other hand, it is a classical example of a fixed-parameter tractable (FPT) problem: It can be solved in $O(b^k p(n))$ time, with some constant base b and some fixed polynomial p. In a sense we will extend the classical FPT result and show that *hidden* vertex covers can be learned efficiently and nonadaptively if k is small.

2 A Classification of Search Strategies and our Results

We discuss both deterministic and randomized strategies. Traditionally, a randomized algorithm is called Monte Carlo if its output is correct with positive probability (that can be boosted close to 1), and called Las Vegas if its output is always correct but the complexity is randomized. Since we are concerned with queries *and* stages we need a finer classification that may be interesting also for other, similar search problems. Regarding the output reliability we distinghuish the following types of strategies, with increasing demands:

PC (probably correct): The output is correct with a given probability $1-\epsilon$.
PCV (probably correct and verified): Like PC but, additionally, the searcher always knows whether the output is correct or the algorithm has failed this time.
C (correct): The output is always correct.

Note that an attribute CV would be meaningless, since C includes by definition that the output is verified. Regarding complexity bounds we distighuish:

EQ: The given number of queries is an expected number.
GQ: The given number of queries is a guaranteed upper bound.
ES, GS: similarly defined, for the number of stages.

We denote types of strategies by these symbols, separated by hyphens. Deterministic strategies is a special case, with characterization GS-GQ-C. (But conversely, the definition allows also randomized GS-GQ-C strategies.)

For competitive group testing with $O(d \log n)$ queries (i.e., pools), we show the following results in this paper:

- For strategies with strict demands we give a lower bound tradeoff for pools vs. stages. In particular, we show that $\Omega(\log d/ \log\log d)$ stages are needed to achieve $O(d \log n)$ queries. First we prove this for deterministic GS-GQ-C strategies, using a counting argument. (Whereas the technique is fairly standard, the details of the adversary strategy and counting process are not obvious. We explore a hypergraph representation of the query results. Also, there remains a $\log\log d$ gap between our current bounds. We conjecture that our proof can be refined to give a matching $\Omega(\log d)$ lower bound.)

- Then we extend the result to GS-GQ-C strategies. Here the additional idea is to let an adversary speculate on some sequence of pooling designs that appears with positive probability, and apply the "deterministic" proof. It remains open whether the lower bound still holds for GS-EQ-C and ES-GQ-C.

- If we relax some demands, the number of stages goes down to a constant. By estimating the unknown number d of defectives in stage 1 and then using a 2-stage strategy, we obtain a GS-EQ-PCV strategy with 3 stages, or alternatively, an ES-EQ-C strategy with $3 + \delta$ stages, where δ can be made arbitrarily small, at cost of the constant factor in the query number $O(d \log(n/d))$.

- The same idea, but combined with randomized 1-stage group testing, yields a GS-EQ-PCV strategy that works in only 2 stages but has a few disadvantages (see details later on). The complexity of ES-GQ strategies remains open.

- Next we show that competitive group testing with a trivial 2-stage GS-EQ-PCV strategy is impossible (where "trivial" is meant in the above sense).

- If we keep GS-GQ but relax C, we need a non-constant number of stages again, but it is slightly better than the GS-GQ-C lower bound: We give a GS-GQ-PCV strategy with $2 \log d/ \log\log n$ stages, based on random partitions. At first glance it might appear strange to see n in the denominator, but note that a large n can help the stage number because n appears in the query number $O(d \log n)$ as well.

We also highlight some results that came out as byproducts:

- From the hypergraph characterization in our lower bound proof we get that determining the *exact* number d of defectives by group tests is not easier than actually identifying the defectives. (Note that in applications like environmental testing we may only be interested in the amount of contamination of samples, rather than in individual items.) This contrasts to the result that we can approximate d efficiently by randomized strategies (see above).
- As another special case of our pools vs. stages tradeoff, we show that the number of pools in deterministic strategies with constantly many stages cannot be limited to any function $f(d) \log n$.

Our result for finding hidden vertex covers by edge group tests is only loosely tied to the rest and not listed here.

3 A Lower Bound for Adaptivity in Competitive Deterministic Group Testing

In this section we give an adversarial answer strategy that forces a certain minimum number of stages upon a searcher who wants to keep the number of pools restricted. Consider a set X containing an unknown subset of defectives.

Definition 1 *Given a set P of pools, the* response vector *t assigns every positive (negative) pool the value 1 (0). Let P^+ and P^- be the set of positive and negative pools, respectively. The* response hypergraph *$RH(P,t)$ has the vertex set $V := X \setminus \bigcup_{Q \in P^-} Q$, and every $Q \in P^+$ is turned into a hyperedge $Q \cap V$ of $RH(P,t)$.*

The response vector simply describes the outcome of a group testing experiment using the set P of pools. The vertices of $RH(P,t)$ are all elements that appear in no negative pool. The hyperedges of $RH(P,t)$ are the positive pools restricted to these vertices, that is, all elements recognized as negative are removed.

A *hitting set* of a hypergraph is a set of vertices that intersects every hyperedge. Note that a superset of a hitting set is a hitting set, too. From the definitions it follows immediately:

Lemma 2 *Let t be a given response vector. Then a subset of X is a candidate for being the set of defectives if and only if it is a hitting set of $RH(P,t)$.* $\diamond$

Before we state our adversary strategy in detail, we outline the ideas. Consider any deterministic group testing strategy that works in stages. Our adversary answers the queries in every stage in such a way that $RH(P,t)$ has some hitting set that is much smaller than the vertex set. This leaves the searcher uncertain about the status (positive or negative) of all the other vertices in $RH(P,t)$. (Note that an adversary working against a *deterministic* searcher can hide defectives *after* having seen the pools.) Next, we use a standard trick to simplify the analysis: The adversary may cautiously reveal some extra information. Specifically, our adversary tells the searcher a subset of defectives that forms already a hitting set of $RH(P,t)$. The effect is that all hyperedges of $RH(P,t)$ are now "explained" by the revealed defectives, thus $RH(P,t)$ does not contain any further useful information for the searcher. Hence the searcher can totally forget the hypergraph, and the searcher's knowledge is merely represented by two sets: the already known defectives, and the elements whose status is yet unknown. Each of the latter elements can be (independently!) positive or negative. We will play around with the cardinalities of these two sets and make the searcher's life as hard as possible. In the following we specify our adversary strategy in detail.

Let f be any monotone increasing function and d the true number of defectives. Recall that d is unknown to the searcher. Suppose that the searcher is aiming for at most $f(d)\log n$ queries in total. Let us consider the moment prior to any stage. Suppose that k defectives are already known and u elements are yet undecided. As we might have $d=k$, the searcher can prepare a set P of at most $f(k)\log n$ pools for the next stage. (Actually, the number of pools already used up in earlier stages must be subtracted, which makes the limit even lower, but our analysis does not take advantage of this fact.) These queries in P can generate at most $2^{f(k)\log n}=n^{f(k)}$ different response vectors. The adversary chooses some number $h\le u$ and announces that at least h of the u undecided elements are also defective. In particular, there exist $\binom{u}{h}$ possible sets of exactly h further defectives. By the pigeonhole principle, some family T of at least $\binom{u}{h}/n^{f(k)}$ of these candidate sets of size h generate the same consistent response vector t. Now the adversary answers with just this response vector t. Let Y denote the union of all sets in T.

Lemma 3 *Y is entirely in the vertex set of $RH(P,t)$.*

Proof. Assume that some $q\in Y$ is in some pool Q which is negative in t, that means, $t(Q)=0$. By the definition of Y, element q also belongs to some $Z\subseteq Y$ such that response vector t is generated if Z is the actual set of defectives. This contradicts $Q\cap Z=\emptyset$. $\diamond$

Define $y=|Y|$. Finally, the adversary actually names a set H of h new defectives in Y, in compliance with t. By Lemma 2, H is a hitting set in $RH(P,t)$. Since arbitrary supersets of H are hitting sets, too, and Y is included in $RH(P,t)$ by Lemma 3, it follows that H plus any of the $y-h$ elements of $Y\setminus H$ build a hitting set of $RH(P,t)$. Using Lemma 2 again, we conclude that the $y-h$ elements of $Y\setminus H$ are still undecided, that is, they may be defective or not, independently of each other.

Since Y must contain at least $\binom{u}{h}/n^{f(k)}$ different subsets of size h, we get the following chain of inequalities:

$$\frac{y^h}{h!}>\binom{y}{h}\ge\frac{\binom{u}{h}}{n^{f(k)}}>\frac{(u-h)^h}{h!n^{f(k)}}.$$

Multiplication with $h!$ and taking the h-th root yields $y>(u-h)/n^{f(k)/h}$.

In summary, after the considered stage the searcher knows $k+h$ defectives, and at least $y-h>(u-h)/n^{f(k)/h}-h=u((1-h/u)/n^{f(k)/h}-h/u)$ elements remain undecided. Thus we update k,u by $k:=k+h$ and $u:=y-h$. This concludes the discussion of any one stage.

In the following, s denotes the number of stages the adversary wants to enforce, k_i and u_i indicates the value of k and u, respectively, before stage i, and h_i is the value of h in stage i. The adversary will choose the h_i in such a way that u_s is still positive. Her choice of numbers $h_i>f(k_i)$ will be based on $f(k_i)$ only.

Remember that the function f is fixed. This allows us to neglect some minor terms in the expression for the updated u without affecting the asymptotics: Since $k\le d$ holds at any time, the largest possible $f(k)$ depends on d only. Furthermore, since our adversary chooses in every stage $h > f(k)$ depending on $f(k)$ only, the ratios h/u in the above expression for $y - h$ remains bounded by an arbitrarily small constant all the time, when we start at large enough n. Therefore we can assume for simplicity that at least $u/n^{f(k)/h}$ elements are undecided after the considered stage. The relative error becomes arbitrarily small as n grows.

Let $k_1 = 1$, that is, one defective is revealed in the beginning. We let our adversary choose the h_i such that $\sum_i f(k_i)/h_i \leq 1$ after any number $i \leq s$ of stages. Since $u_1 = n - 1$ and $u_{i+1} \geq u_i/n^{f(k_i)/h_i}$, we have that u_i remains positive for $i \leq s$, as desired.

Specifically, let $h_i := sf(k_i)$ for all i, hence $f(k_i)/h_i = 1/s$ and $k_{i+1} = k_i + h_i = k_i + sf(k_i)$. Now we can formulate a somewhat technical but general result:

Theorem 4 *Let f be any monotone increasing function with $f(d) \geq d$ for all d. Any deterministic group testing strategy that uses, for arbitrary combinations d, n, at most $f(d)\log n$ pools for finding a previously unknown number d of defectives out of n elements, needs at least $s(d)$ stages in the worst case, where $s(d)$ is defined as the minimum number with the following property: If the operator $k := k + sf(k)$ is iterated $s := s(d)$ times starting from $k = 1$, then $k \geq d$ is reached.* $\diamond$

Theorem 4 in this general form looks bulky, but for the most important cases of functions f we get some interesting results from it. First we look at $f(d) = d$. Due to the information-theoretic lower bound for group testing, this is the smallest meaningful function f in our context.

Corollary 5 *Any deterministic group testing strategy that uses, for arbitrary combinations d, n, only $O(d \log n)$ pools for finding a previously unknown number d of defectives out of n elements needs $\Omega(\log d/\log\log d)$ stages in the worst case.*

Proof. With the previous denotations, $f(k) = k$ yields $k_{i+1} = (s + 1)k_i$, hence $d = k_{s+1} = (s + 1)^s$, and $s > \log d/\log\log d$. $\diamond$

We remark that raising the number of pools by a constant factor a does not help very much: Since now $f(k) = ak$, our adversary chooses $k := (as + 1)k$ and still achieves $s = \Theta(\log d/\log\log d)$, although with a smaller constant factor.

Next it is interesting to consider $f(d) = d^2$, because this number of pools would allow a 1-stage strategy if d were known beforehand (see Section 1). However, our adversary strategy for *unknown* d essentially squares k in every iteration, leading to $s = \Omega(\log\log d)$. Finally, consider an arbitrary but fixed function f that may be rapidly growing.

Corollary 6 *Let f be any monotone increasing function. No deterministic group testing strategy that uses at most $f(d)\log n$ pools for finding a previously unknown number d of defectives out of n elements can succeed in constantly many stages.*

Proof. It suffices to notice in Theorem 4 that the number s of iterations needed to reach $k \geq d$ depends on d. $\diamond$

This contrasts sharply to our result in the next section where we give randomized strategies with constantly many stages, based on a randomized estimate of d. A simple but interesting observation in this context is that finding the exact number of defectives is not easier than solving the whole group testing problem:

Theorem 7 *Any group testing strategy that exactly determines the previously unknown* number *of defectives must also identify the* set *of defectives.*

Proof. Assume that a searcher has applied any group testing strategy to some set of elements containing d defectives, and afterwards the searcher knows d. Let P be the set of pools ever used, and t the response vector. By Lemma 2, the possible sets of defectives are exactly the hitting sets of $RH(P,t)$. Hence, by assumption, all hitting sets of $RH(P,t)$ have the same cardinality. This is possible only if $RH(P,t)$ has only the trivial hitting set consisting of all vertices. Using Lemma 2 again, it follows that the searcher knows the defectives. $\diamond$

So far we have studied only deterministic strategies. Recall that they are of type GS-GQ-C. Next we show that the general Theorem 4 (and hence the subsequent results) still holds for randomized strategies of this type. Note that the adversary has to be "oblivious", i.e., she must fix the set of defectives before seeing the searcher's random decisions.

Theorem 8 *Let f be any monotone increasing function with $f(d) \geq d$ for all d. Any GS-GQ-C group testing strategy that uses, for arbitrary combinations d, n, at most $f(d)\log n$ pools for finding a previously unknown number d of defectives out of n elements, needs at least $s(d)$ stages, with $s(d)$ defined as in Theorem 4.*

Proof. Consider any randomized strategy. We let our adversary from Theorem 4 play against the strategy. The possible outcomes can be viewed as a tree, in an obvious way: In every tree node the searcher decides on the set of pools for the next round, with probabilities specified by the strategy, and then our adversary reacts as above. Now, if the strategy is GS-GQ-C, it must succeed on any path of the tree, i.e., identify the defectives correctly within the specified bounds for stages and pools.

The above procedure describes an adaptive adversary that can decide on further defectives step by step according to the searcher's choices. However, an oblivious adversary can choose any path p of the tree and speculate that the searcher will behave as on p (which happens with positive probability), and choose the set of defectives according to the result of p. Then the sequence of pools used on p can be seen as a deterministic strategy (applied by the searcher with some positive probability), and Theorem 4 shows that, on p, the searcher cannot work correctly and stick to both complexity bounds at the same time. Hence one of the constraints is violated with positive probability, thus a GS-GQ-C strategy with $f(d)\log n$ pools and less than $s(d)$ stages cannot exist. $\diamond$

We conjecture that the same lower-bound tradeoff still holds for each of the weaker demands GS-EQ-C and ES-GQ-C. It is tempting to argue as follows: If a strategy respects an expected value of one complexity bound (and the other bound is strict), then the concrete value on some tree path p must not exceed the expectation, and Theorem 4 shows again that the strategy cannot work correctly on p. However, the catch is that the expectation in EQ or ES refers to the searcher's behaviour on any set of defectives fixed beforehand, hence the adversary cannot simply speculate on p as in the GS-GQ-C case.

4 Randomized Estimation of the Number of Defectives

In this section we show that a "conservative bound" $\hat{d}$ on the number d of defectives in a set of n elements can be determined by $O(\log n)$ randomized nonadaptive group tests. What the above phrase means is that we get $\hat{d} < d$ with an arbitrarily small prescribed failure probability, but $\hat{d}$ overestimates d only by a constant expected factor. Thus, $\hat{d}$ can be further used in any group testing strategy that needs an upper bound on d: Such a strategy fails only with a small probability due to $\hat{d} < d$, but it is also unlikely to waste too many tests due to a large $\hat{d}/d$.

Let $b > 1$ be a fixed positive real number. We prepare a sequence of pools indexed by integers i as follows. We put every element independently with probability $1 - (1 - 1/n)^{b^i}$ in the ith pool. The ith pool is negative with probability $q_i := (1 - 1/n)^{db^i}$, since this is the probability that all d defectives are outside the pool. The test outcomes of all pools are independent, as we have chosen the elements of the pools independently. Also note that, regardless of the unknown value of d, our sequence q_i is doubly exponential: Every number is the bth power of the previous number and the bth root of the next number. This is a nice invariant that enables a "uniform" analysis for all possible values of d.

Let e denote Euler's number and $k := \lfloor \log_b n \rfloor$. We have $q_0 = (1 - 1/n)^n \approx 1/e$ if $d = n$, and $q_k \approx (1 - 1/n)^n \approx 1/e$ if $d = 1$. That means, q_0 is "away from" 0 even if $d = n$, and q_k is "away from" 1 even if $d = 1$. Now let i range from some constant negative index to k plus some positive constant. Then group testing on this sequence of pools yields, with high probability, some negative pools in the beginning, even if $d = n$, and some positive pools in the end, even if $d = 1$. Notice that we prepare roughly $\log_b n = \log n / \log b$ pools.

We propose a simple algorithm to get $\hat{d}$ from the test outcomes. It is equivalent to estimate the index j of a pool with some fixed value q_j. Our algorithm will "assume" $q_j = 1/2$ for a certain j based on the test outcomes, and output the corresponding number of defectives $\hat{d}$ that would lead to $q_j = 1/2$. In order to distinguish this assumed probability value from the true (and unknown) q_j we denote it $\hat{q}_j$. Our algorithm uses yet another positive parameter s (integer) that we discuss later. The subsequent lemmas are meant with respect to the algorithm. Here it is:

Let i be the largest index of a negative pool; we will refer to i as the *main index*. Then let $\hat{q}_{i-s} := 1/2$ and set $\hat{d}$ accordingly, that is, $\hat{d} := -1/(b^{i-s} \log(1 - 1/n))$.

(Clearly $\hat{d}$ must be rounded to an integer, for simplicity we omit ceiling brackets.) Remember that $b > 1$ is some fixed base. We choose s large enough to make $1/2^{b^s}$ "small", see the details below. With these presumptions we get:

Lemma 9 *The probability of $\hat{d} < d$ is $O(1/2^{b^s} \log b)$.*

Proof. The event $\hat{d} < d$ is equivalent to $q_{i-s} < 1/2$ for the main index i. In this case, i is one of the indices with $q_i < 1/2^{b^{s+j}}$, $j \geq 0$. Hence the probability of this failure is bounded by $\sum_{j \geq 0} 1/2^{b^{s+j}}$. Since $1/2^{b^s}$ is already small due to the choice of s, and every term is the bth power of the previous one, the sequence then decreases rapidly: After every $\log_b 2 = 1/\log b$ indices it is reduced to the square. Thus we get a failure probability as claimed, with a small hidden constant. $\diamond$

In order to express the expected competitive ratio we define a function F with argument $b > 1$ by:

$$F(b) := \max_{0 \leq \theta < 1} \sum_{i=-\infty}^{\infty} \frac{b^{-i+\theta}}{2^{b^{i-\theta}}} \prod_{k=1}^{\infty} \left(1 - \frac{1}{2^{b^{i-\theta+k}}}\right).$$

Although the expression for $F(b)$ looks complicated, it is not hard to prove that $F(b)$ is monotone in b, and to get good simple bounds for $F(b)$. However, we do not further analyze F, as these issues affect only the constant factors in our final results below. Instead, some numerical values may illustrate the behaviour of F:

$\#pools$	$1 \log n$	$2 \log n$	$4 \log n$	$8 \log n$	$16 \log n$	$32 \log n$
b	2.000	1.414	1.190	1.091	1.045	1.022
$F(b)$	1.466	0.830	0.549	0.397	0.307	0.247

Lemma 10 *The expectation of $\hat{d}/d$ is at most $F(b) \cdot b^s$.*

Proof. We (arbitrarily) shift our indices such that $1/2^{b^{-1}} > q_0 \geq 1/2^{b^0}$ and define θ such that $0 \leq \theta < 1$ and $q_0 = 1/2^{b^{-\theta}}$. If the main index is i, the algorithm yields $\hat{q}_{i-s} = 1/2$ whereas $q_{i-s} := 1/2^{b^{i-s-\theta}}$, hence $\hat{d}/d = b^{-i+\theta} \cdot b^s$. From $q_i = 1/2^{b^{i-\theta}}$ and the definition of the main index, the assertion follows. $\diamond$

Altogether we have shown:

Theorem 11 *Consider a set of n elements including an unknown number d of defectives. For any $b > 1$ and any positive integer s, there is a randomized 1-stage group testing strategy with $\log n / \log b$ pools that outputs a number $\hat{d}$ such that $\hat{d} < d$ holds only with probability $O(1/2^{b^s} \log b)$, and $\hat{d}/d < F(b) \cdot b^s$ in expectation.* $\diamond$

5 Randomized Competitive Group Testing in Two or Three Stages

From the result of the previous section we can easily conclude:

Theorem 12 *Consider a set of n elements including an unknown number d of defectives.*
(i) There is a GS-EQ-PCV group testing strategy that finds all defectives in 3 stages using $O(d \log(n/d))$ pools.
(ii) For any fixed $\delta > 0$ there is a ES-EQ-C group testing strategy that finds all defectives in $3 + \delta$ stages using $O(d \log(n/d))$ pools.

Proof. Fix any desired failure probability bound $\epsilon > 0$. By Theorem 11 we can get in stage 1 some $\hat{d}$ that exceeds d with probability at least $1 - \epsilon$, keeping the expected ratio $\hat{d}/d$ constant at the same time, and $O(\log n)$ pools are used for that, where the constant factor depends on ϵ. Then we apply one of the established 2-stage group testing strategies for a maximum number $\hat{d}$ of defectives. As mentioned in Section 1, there are 2-stage strategies that first determine $O(d)$ candidate elements including all defectives, and test them individually in the final stage. We add one more pool in this final stage, consisting of the complement of the candidates. If this extra pool is negative, the searcher knows that all defectives are found, hence the strategy is PCV. (But it can still fail if $\hat{d} < d$.) This shows (i). Assertion (ii) is a simple extension: If failure is detected in stage 3, the strategy is repeated until success. We achieve any desired δ by chosing ϵ small enough. $\diamond$

Some practical remarks are in order here: Since the probability of a too generous $\hat{d}$ decreases rapidly as the ratio $\hat{d}/d$ grows, large deviations from the expected number of pools are very unlikely. On the other hand, note that $\hat{d}/d$ is not necessarily close to 1. The best choice of the method parameters b, s that minimize, e.g., the hidden factor in $O(d \log(n/d))$ depends on ϵ and the largest relevant d. This needs to be further explored. We did some numerical experiments adopting the $1.44\, d \log(n/d)$ bound for 2-stage group testing [8]. With bases $1.2 < b < 1.5$ we get factors from $3 + 1/d \log b$ to $6 + 1/d \log b$, for ϵ from 0.1 down to 0.02.

Next we show that even two stages are enough if we randomize the further tests as well. In Theorem 10 of [8] it is shown that up to d defectives can be identified correctly, with any desired constant probability, in one stage using $O(d \log n)$ pools. Inspection of the proof shows that their random set of pools has, with constant probability, the following property: Let P be the set of positive elements. If actually $|P| \leq d$ then, (i) for every $v \notin P$ there exists a negative pool that contains v but is disjoint to P. The same analysis in [8] also shows that, (ii) for every $v \in P$, there exists a positive pool Q that contains v but is disjoint to $P \setminus \{v\}$. The strategy outputs all elements that appear in positive pools only. The above properties imply that, with constant probability, every negative element v is recognized as negative due to (i), and every positive element v is recognized as positive due to (ii). The latter statement holds since Q is a positive pool but all elements in $Q \setminus \{v\}$ are recognized as negative. In other words, the output set O is exactly P in this event. From the searcher's point of view we have: If O (in the role of P) satisfies (i) and (ii) then the searcher is sure about $O = P$. Hence this strategy is PCV.

Theorem 13 *Consider a set of n elements including an unknown number d of defectives. There is a GS-EQ-PCV group testing strategy that finds all defectives in 2 stages using $O(d \log n)$ pools.*

Proof. As before, fix any desired failure probability bound $\epsilon > 0$ and determine in stage 1 some $\hat{d}$ that is smaller than d with probability at most $\epsilon/2$, and with constant expected ratio $\hat{d}/d$. In stage 2, apply a the 1-stage PCV strategy for $\hat{d}$ defectives and failure probability $\epsilon/2$. Thus the strategy fails with probability at most ϵ, and if it succeeds, the searcher knows that exactly the set of positive elements has been returned. $\diamond$

Thus we have saved one stage, but we conjecture that the hidden constant factor in $O(d \log n)$ will be considerably larger than for 3 stages, mainly since the error probability ϵ must be divided between two stages (whereas in Theorem 12 only stage 1 could potentially fail). We must leave the constant factors, depending on the prescribed ϵ, as an open question. Obviously, the 2-stage strategy also needs a larger amount of randomness.

A clear disadvantage of the 2-stage strategy is in the verification step. While the proposed 3-stage strategy really ends after the tests in stage 3, the 2-stage strategy needs quite some computational postprocessing to confirm the result: It guarantees a correct result only if the output set O has the above properties (i) and (ii), that is, we have to revisit the pool set and find the required pools for every element v. A natural question is whether a trivial 2-stage strategy exists with the same performance. Remember that a 2-stage group testing strategy is called trivial if stage 2 consists of individual tests of candidate positives only, hence no decoding or verification is needed at all. However we can show:

Theorem 14 *There exists no trivial 2-stage GS-EQ-PCV strategy for group testing with previously unknown d, using $O(d \log n)$ pools.*

Proof. Since such a strategy has to test all candidates from stage 1 individually in stage 2, the total number of pools is at least the number of candidates from stage 1.

First we prove the assertion for GS-GQ-PCV strategies. Suppose that the searcher wants to use at most $cd \log n$ pools, for some constant c. Regardless of her random choices the searcher may use only $c \log n$ pools in stage 1, since otherwise the adversary may simply fix $d = 1$. It remains to show that the number of candidates remains too large after these $c \log n$ nonadaptive group tests. As the pooling design may be randomized, we use Yao's technique to derive a lower bound on the expected number of candidates. We say that a negative pool *discards* the elements in this pool. First consider any deterministic, i.e., fixed pooling design in stage 1. Our adversary fixes some number d (we discuss the choice of d below) and decides d defectives at random. Thus, any pool of size xn is negative with probability $(1-x)^d$, that is, the pool discards an expected number of $x(1-x)^d n$ elements. By a standard calculation, this expectation is maximized if $x = 1/(d+1)$, hence the expected number of elements discarded by any one pool is at most $n/e(d+1)$, where e denotes Euler's number. By linearity of expectation, $c \log n$ pools discard an expected number of at most $(cn \log n)/(e(d+1))$ elements. (Actually this number is smaller if the pools overlap.) If d is chosen larger than $(c/e) \log n$, there remain $\Omega(n)$ expected candidates when any deterministic strategy is applied in stage 1. Any randomized pooling design using a fixed number of $c \log n$ pools in stage 1 is a probability distribution on the deterministic pooling designs of that size. Hence, with $d > (c/e) \log n$ random defectives, $\Omega(n)$ expected candidates are left, also if such a randomized design is applied. It follows that some specific set of $d > (c/e) \log n$ defectives exists that fools the given randomized strategy.

Finally we consider GS-EQ-PCV strategies. Now the searcher may also randomize over the number of pools to be used in stage 1, but still their expected number must not exceed $c \log n$, for the same reason as above. For any fixed $\epsilon > 0$, Markov's inequality yields that the strategy must choose the actual number of pools below $(1+\epsilon)c \log n$ with some constant positive probability. Applying the previous result to this event we see that the expected number of candidates remains $\Omega(\epsilon n)$ there. Thus, with constant positive probability the searcher is faced with $\Omega(\epsilon n)$ candidates to test in stage 2, which in total enforces a linear expected number of pools. $\diamond$

Note that this proof uses that possibly $d = \Omega(\log n)$. If d is smaller compared to $\log n$, we can again succeed with $O(d \log n)$ pools in 2 stages, even with a GS-GQ-PCV strategy. However, we prove a more general result below.

Lemma 15 *Consider a set of m elements. There exists a GS-GQ-C strategy using $O(\log m)$ pools in 1 stage, that finds the defective if $d = 1$, and otherwise recognizes that $d = 0$ or $d > 1$.*

Proof. Let n be a power of 2, otherwise we create dummy elements, which doubles m in the worst case. We arrange the elements in a $\log m$ dimensional hypercube and query its $2\log m$ maximal subhypercubes. If actually $d = 1$, the answers obviously determine the defective uniquely. If $d = 0$ then all pools are negative. If $d > 1$ then at least two complementary pools are positive. These cases are recognized by the searcher. $\diamond$

Theorem 16 *Consider a set of n elements including an unknown number d of defectives. There exists a GS-GQ-PCV strategy using $O(d\log n)$ pools in $O(\log d/\log\log n)$ stages.*

Proof. Every stage works as follows: If h is a currently known lower bound for d (where $h = 1$ is assumed in the beginning), we partition the elements randomly into some $\Theta(h\log n)$ disjoint pools of equal size and query them. For the next stage we update h as the number of positive pools.

Using simple standard results from probability theory, the following two things happen with high probability: As long as $h\log n < d$, at least a constant fraction of pools answers positively, thus h is multiplied by $\Theta(\log n)$ in every stage. As soon as $h\log n > d^2$, the defectives are separated, that is, no two defectives get into the same pool.

In the latter case the searcher recognizes that fewer than $\sqrt{h\log n}$ pools are positive and conjectures that the defectives are in fact separated. Finally, in one further round the searcher applies Lemma 15 to each positive pool from the previous stage, to find the defective and confirm that it is the only one.

As for the stage number, note that $s = \log d/\log\log n$ is the solution to $(\log n)^s = d$, and constant factors are captured by the O-notation. $\diamond$

In particular, this algorithm uses only 2 stages if $d = O(\sqrt{\log n})$. It would be interesting to explore the case when $\sqrt{\log n} < d < \log n$.

6 Learning Hidden Vertex Covers by Edge Group Tests

In this section we study the problem of learning all minimal vertex covers of size at most k, in an initially unknown graph, from edge group tests. Note that, in general, this does not uniquely determine the graph, because there may exist further minimal vertex covers with more than k vertices.

Definition 17 *A set of pools is $(2,k)$-disjunct if, for any $k+2$ vertices $w_1,\dots,w_k$ and u,v, there exists a pool that includes u,v and excludes $w_1,\dots,w_k$.*

There exist $(2,k)$-disjunct matrices with $O(k^3\log n)$ pools: The simplest randomized construction is to put every vertex in a pool independently with probability $2/k$. Bounds on the size of a more general type of disjunct matrices can be found in [5].

Let $VC(n,k)$ denote the time for enumerating the minimal vertex covers of size at most k in a (known!) graph of n vertices. The time depends on the state-of-the-art of FPT vertex cover algorithms, which is beyond the scope of this work. However, we have $VC(n,k) = O(b^k p(n))$ where $b < 2$ is some fixed base and p some fixed polynomial, see [12] for more details. If only one vertex cover is sought, one can apply faster algorithms such as the one in [7].

Theorem 18 *We can learn all (minimal) vertex covers of size at most k in one stage using $O(k^3 \log n)$ edge group tests and $O(VC(n,k))$ time for auxiliary computations. The strategy is of type GS-GQ-C.*

Proof. We take a set of pools that forms a $(2,k)$-disjunct matrix. Consider any pair of vertices $\{u,v\}$. Clearly, if $\{u,v\}$ belongs to some negative pool then uv is a non-edge. The other case is that $\{u,v\}$ belongs to positive pools only. Assume that $u,v \notin C$ for some vertex cover C with $|C| \leq k$. Due to $(2,k)$-disjunctness, some pool includes u,v and excludes C. This pool must be positive, as it contains u,v. On the other hand, the pool must be negative, as every edge intersects C. This contradiction shows that every vertex cover C with $|C| \leq k$ contains u or v. Hence, for the purpose of learning these small vertex covers, we may simply assume that uv is an edge if $\{u,v\}$ appears in positive pools only. This might be wrong in the unknown graph, but the family of vertex covers of size at most k is preserved.

This reasoning also yields the following parameterized decoding algorithm that actually generates the family of vertex covers of size at most k from the test outcomes: Construct an auxiliary graph where uv is an edge if and only if $\{u,v\}$ belongs to positive pools only. This can be done in $O(k^3 n^2 \log n)$ time, which is dominated by the time for the final step: Compute the minimal vertex covers of size at most k in this graph. $\diamond$

On the other hand, the trivial information-theoretic lower bound gives:

Proposition 19 *Any strategy (which may even be adaptive) for learning hidden vertex covers of size at most k needs $\Theta(k \log(n/k))$ edge group tests.* $\diamond$

An obvious question is what is the best exponent of k in the query number $O(k^{O(1)} \log n)$, depending on the number of stages and the type of strategy. In Theorem 18 we did not use prior knowledge of the size of a smallest vertex cover. If our k is too small, we just obtain an empty result which is correct. However, if we want *some* vertex cover, we have to determine the minimum size k of a vertex cover first. Here, an $O(\log k)$-stage deterministic strategy or a randomized 1-stage method similar to Section 4 should work. Note that the complements of vertex covers in a graph are exactly the independent sets, which in turn corresponds to negative pools. Hence we could again use a randomized sequence of pools of exponentially growing size to estimate k. We have to leave these questions for further research.

7 Some Discussion and Experimental Findings

Besides the open problems regarding strategy types and asymptotic complexity results (mentioned at several places), more research is needed on the practical side:

Our current lower bounds in Section 3 do not say too much about realistic problem sizes, but we conjecture that they can be further raised. An obvious weakness in the analysis in Section 3 is that the searcher is allowed to use the maximum number of pools in every new stage, not counting the pools used up earlier. Asymptotically this is negligible, but for moderate d our adversary gives away some power by this simplification. One may also think of more sophisticated adversary strategies that exploit more structure of the response hypergraphs.

Our $\hat{d}$ in Section 4 is, for the sake of simplicity, based on the "main index" i only, which is the index of the largest negative pool. This strategy is very sensitive to the position of the main index which may easily be an outlier. In order to get more robust estimates we have empirically tried different rules of combining the responses of pools around the main index. Here we summarize a comparison with two other natural rules that show significantly better performance. The rules differ in the choice of pool j for which we assume $\hat{q}_j := 1/2$.

Rule 1: the simple algorithm analyzed in Section 4, with $j = i - s$ for some previously fixed s that depends on the desired failure probability bound.
Rule 2: like Rule 1, but j is the index of the s-th *negative* pool before i, jumping over the positive pools.
Rule 3: Let k be the index of the smallest positive pool, $m = \lfloor (i+k)/2 \rfloor$, and finally, j be the index of the s-th negative pool before m.

The rationale behind Rule 2 is that it suppresses a too large main index i (an outlier) by straight going to the previous negative pools, and the averaging in Rule 3 should have a smoothing effect as well.

We implemented these rules in Matlab and performed extensive simulations with independent random choices of d defective elements. Recall from Section 4 that $b > 1$ is some fixed base used in the sequence of pool sizes. We considered all values of b in the range 1.1 to 1.9 with an increment of 0.1. For fixed n we selected d as $0.01n$ to $0.10n$ with $0.01n$ increments. Now for different b we separately organized simulation results over the range of s values. Then we computed average values for failure probability, competitive ratio, and variance in competitive ratio values. To compare the performance of the three rules, we have drawn graphs of competitive ratios and variances against failure probability (smaller than 0.1) for each b. For this purpose, in every rule only those s values are considered where the average failure probability lies within the desired range. Here we display competitive ratios (Figure 1–3) and variances (Figure 4–6) for b values 1.1, 1.3, and 1.5. Symbols $\times$,+, and $*$ are used to draw points for rules 1,2, and 3, respectively. Integers besides the points indicate the values of s. The measured points are joined using the logarithmic curve fitting just for better visualization of the behaviour, this is not meant to be interpolation.

The following conclusions can be drawn. Rule 2 has shown consistently better performance both in competitive ratio and variance compared to the other two rules. It decreases the average competitive ratio by 20% and the variance by 50% or more compared to Rule 1, for any given failure probability, whereas a significant improvement over Rule 3 is also achieved. For both Rule 2 and 3, the improvement in the competitive ratio compared to Rule 1 increases as the failure probability approaches zero. A close look at the entire simulation data also reveals that Rule 3 has consistently gained failure probability less than 0.08 with competitive ratio in the range 2 to 3 already for $s = 0$, in contrast to Rule 1 that requires large jumps s to achieve small failure probabilities. Overall these simulations suggest that the new rules offer a better performance. An obvious plan is to analyze and understand them also in theory and to figure out the best hidden constants we can achieve in Theorem 12.

Some other ideas for further research on this topic came up:

We invented Rule 2 and 3 by ad-hoc considerations and then tested them. In principle it is also possible to formulate our problem of minimizing the competitive ratio, for a given failure probability, *approximately* as a linear program that returns probabilities of chosing pool j (see notations above). If this experimental approach is feasible, it may even guide us to a conjecture for the optimal randomized rule.

A 2-stage estimator where stage 1 roughly determines the magnitude of d such that stage 2 can focus on the range of the most likely d may save many pools.

We studied the problem of estimating the unknown $d \leq n$ independently, and then we just applied the result to competitive group testing. For this purpose however, we may restrict d straightaway to $O(n/\log n)$ (since otherwise trivial individual testing is anyhow better), and thus further reduce the total number of pools in Theorem 12 and 13.

Acknowledgments

This article was inspired by the first author's participation in the wonderful Dagstuhl Seminars 08301 "Group Testing in the Life Sciences" (2008) and 09281 "Search Methodologies" (2009). The work has been supported by the Swedish Research Council (Vetenskapsrådet), grant no. 2007-6437, "Combinatorial inference algorithms – parameterization and clustering". We are indebted to an anonymous referee who pointed out a flaw in an earlier version of Section 5.

References

[1] N. Alon, V. Asodi, Learning a hidden subgraph, *SIAM J. Discr. Math.* 18 (2005) 697–712.

[2] D. Angluin, J. Chen, Learning a hidden graph using $O(\log n)$ queries per edge, *J. Computer and System Science* 74 (2008) 546–556.

[3] A. Bar-Noy, F.K. Hwang, H. Kessler, S. Kutten, A new competitive algorithm for group testing, *Discr. Appl. Math.* 52 (1994) 29–38.

[4] M. Bouvel, V. Grebinski, G. Kucherov, Combinatorial search on graphs motivated by bioinformatics applications, in *Proc. 31st Workshop on Graph-Theoretic Concepts in Computer Science WG 2005*, *LNCS* 3787, ed. D. Kratsch (Springer, 2005), pp. 17–27.

[5] H.B. Chen, H.L. Fu, F.K. Hwang, An upper bound on the number of tests in pooling designs for the error-tolerant complex model, *Optim. Letters* 2 (2008) 425–431.

[6] 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.

[7] J. Chen, I.A., Kanj, G., Xia, Improved parameterized upper bounds for vertex cover, *31st Symp. on Math. Foundations of Computer Science MFCS 2006*, *LNCS* 4162, ed. R. Kralovic, P. Urzyczyn (Springer, 2006), pp. 238–249.

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

[9] A.E.F. Clementi, A. Monti, R. Silvestri, Selective families, superimposed codes, and broadcasting on unknown radio networks, *12th Symposium on Discrete Algorithms SODA 2001*, 709–718.

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

[11] P. Damaschke, On parallel attribute-efficient learning, *J. Computer and System Science* 67 (2003) 46–62.

[12] P. Damaschke, Parameterized enumeration, transversals, and imperfect phylogeny reconstruction, *Theor. Computer Science* 351 (2006) 337–350.

[13] P. Damaschke, Finding hidden hubs and dominating sets in sparse graphs by randomized neighborhood queries, *Networks*, to appear.

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

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

[16] D.Z. Du, H. Park, On competitive group testing, *SIAM J. Comp.* 23 (1994) 1019–1025.

[17] D.Z. Du, G. Xue, S.Z. Sun, S.W. Cheng, Modifications of competitive group testing, *SIAM J. Comp.* 23 (1994) 82–96.

[18] A.G. Dyachkov, V.V. Rykov, Bounds on the length of disjunctive codes, *Problems of Information Transmission* (Russian) 18 (1982), 7–13.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] M. Lappe, L. Holm, Unraveling protein interaction networks with near-optimal efficiency, *Nature Biotech.* 22 (2003) 98–103.

[24] M. Ruszinkó, On the upper bound of the size of the r-cover-free families, *J. Combin. Theory A* 66 (1994) 302–310.

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

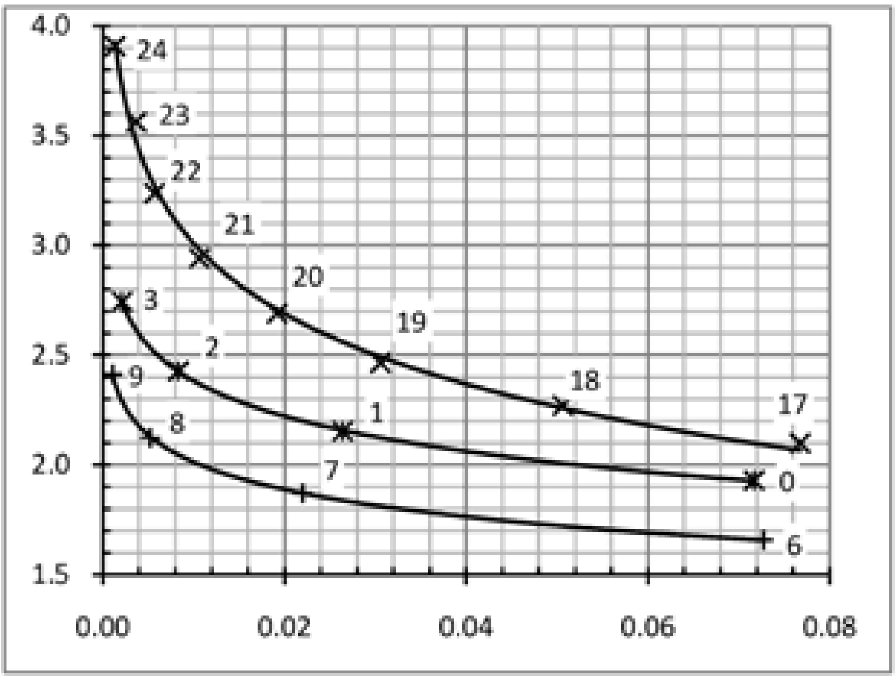

Figure 1: Competitive ratios for b=1.1

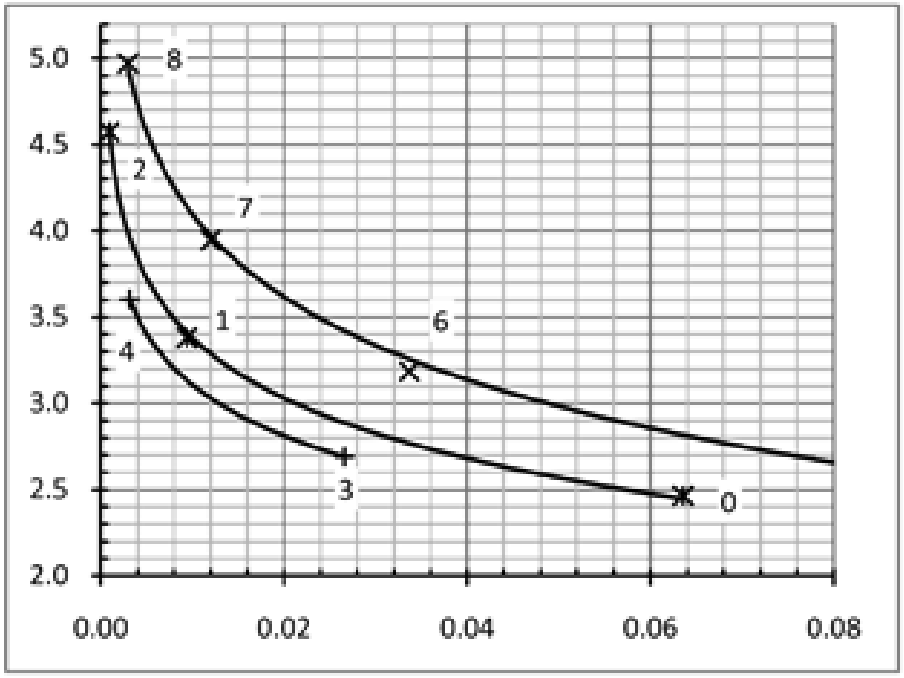

Figure 2: Competitive ratios for b=1.3

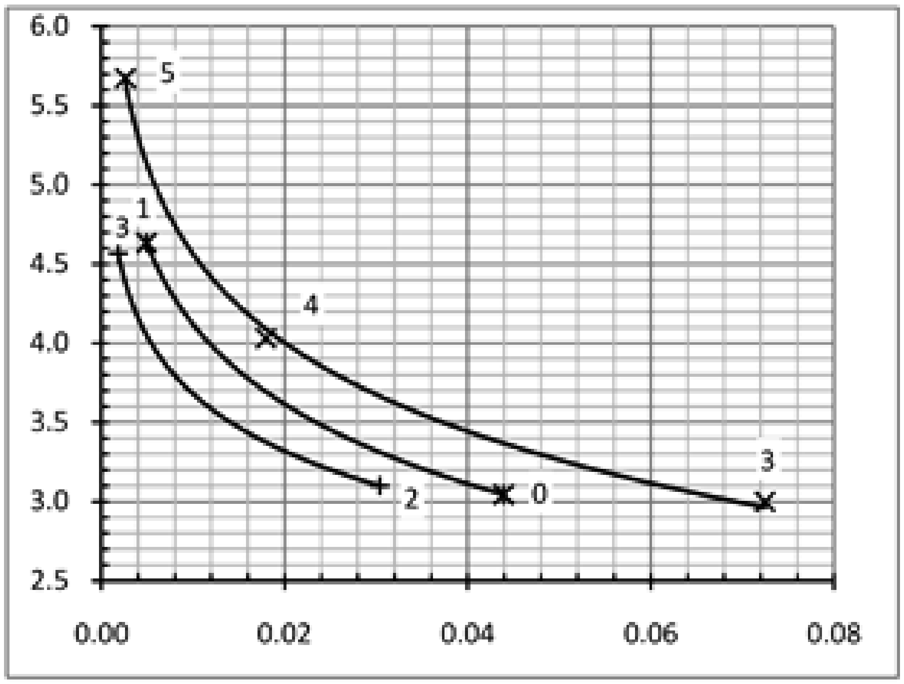

Figure 3: Competitive ratios for b=1.5

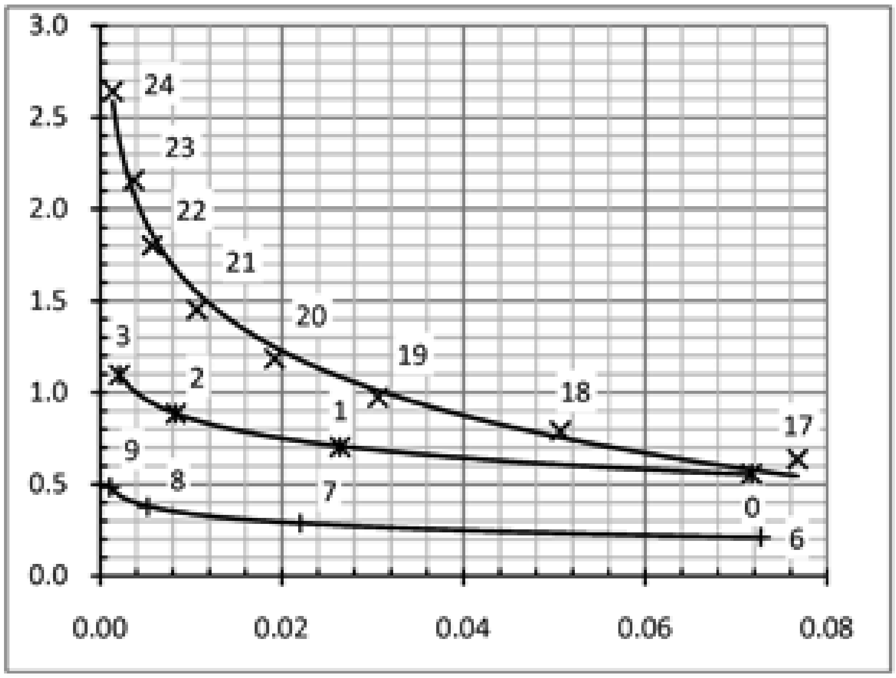

Figure 4: Variances for b=1.1

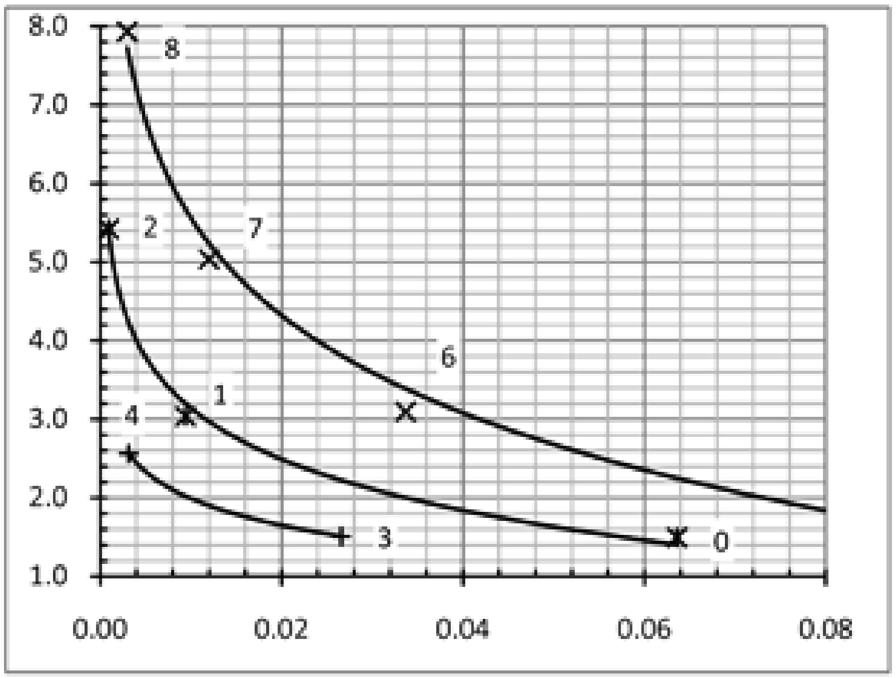

Figure 5: Variances for b=1.3

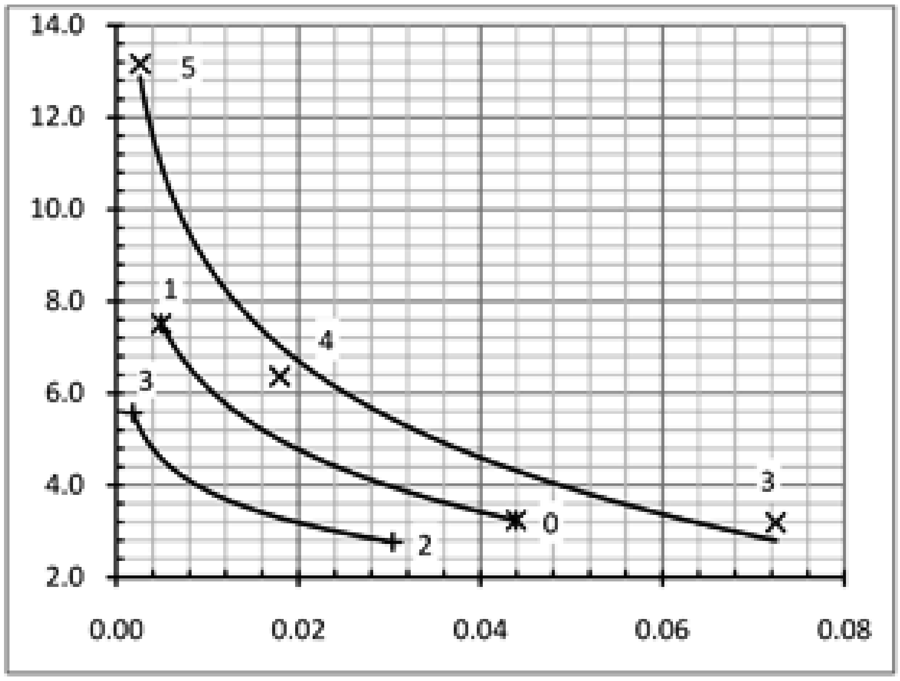

Figure 6: Variances for b=1.5