Threshold Group Testing

Peter Damaschke
School of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
ptr@cs.chalmers.se

Abstract

We introduce a natural generalization of the well-studied group testing problem: A test gives a positive (negative) answer if the pool contains at least u (at most l) positive elements, and an arbitrary answer if the number of positive elements is between these fixed thresholds l and u. We show that the p positive elements can be determined up to a constant number of misclassifications, bounded by the gap between the thresholds. This is in a sense the best possible result. Then we study the number of tests needed to achieve this goal if n elements are given. If the gap is zero, the complexity is, similarly to classical group testing, $O(p \log n)$ for any fixed u. For the general case we propose a two-phase strategy consisting of a DISTILL and a COMPRESS phase. We obtain some tradeoffs between classification accuracy and the number of tests.

1 Introduction

The classical version of the group testing problem is described as follows. In a set of n elements, p elements are *positive* and the other $n - p$ are *negative*. (These terms can stand for any specific property of elements. The "positive" elements are sometimes called "defective" in the group testing literature.) We denote by P the set of positive elements, hence $p = |P|$. Typically p is much smaller than n. A *group test* takes as input any set S of elements, called a *pool*. The test says YES if S contains at least one positive element (that is, $S \cap P \neq \emptyset$) and NO otherwise. The goal is to identify the set P by as few as possible tests.

Group testing is of interest in chemical and biological testing, DNA mapping, and also in several computer science applications. Many aspects of group testing have been studied in depth. One cannot even give an overview of the vast literature. Here we refer only to the book [10] and a few recent papers [7, 11, 12]. Many further pointers can be found there. Group testing also fits in the framework of learning Boolean functions by membership queries, as it is equivalent to the problem of learning a disjunction of p unknown Boolean variables. In [5, 6] we proved a number of complexity results for learning arbitrary Boolean functions with a limited number of relevant variables.

In the present paper we study a generalization of group testing which is quite natural but has not been addressed before, to our best knowledge. Let l and u be nonnegative integers with $l < u$, called the *lower* and *upper threshold*, respectively. Suppose that a group test for pool S says YES if S contains at least u positives, and

NO if at most l positives are present. If the number of positives in S is between l and u, the test can give an arbitrary answer. We suppose that l and u are *constant* and previously known. (If one is not sure about the thresholds, one can conservatively estimate l too low and u too high.) The obvious questions are: What can we figure out about P? How many tests and computations are needed? Can we do better in special cases? We refer to our problem as *threshold group testing*. We call $g := u - l - 1$ the (additive) *gap* between the thresholds. The gap is 0 iff a sharp threshold separates YES and NO, so that all answers are determined. Obviously, the classical case of group testing is $l = 0$, $u = 1$.

It should be observed that the lower sensitivity of threshold group tests compared to the classical case has some consequences that one may find puzzling first. In particular, one cannot simply test single items and thus identify the positives with n trivial tests. Instead one has to examine subsets (details follow in the technical part). However, the fact that an obvious strategy from a special case is no longer applicable is not an objection against the model itself. Anyway, our results will provide sublinear strategies.

In another generalization of group testing, the sensitivity of tests is reduced if too many negatives in S are present, i.e. if the positives are "diluted". This scenario has been studied in [4]. Algorithmically it turned out to be less interesting, in that a straightforward strategy has an asymptotically optimal number of tests. One could think of the following model that combines both aspects. Elements are samples with different concentrations of a substance to detect, and there is an upper threshold for sure detection and a lower threshold under which detection is impossible. Between the thresholds, the outcome of a test is arbitrary. These assumptions seem to be natural in some chemical test scenarios, and the problems are worth studying. This model allows verification of single "positive" elements where concentration is above the upper threshold. It is more difficult to say which other positive elements can be found efficiently (e.g. by "piggybacking" with high-concentration samples), and to what extent unknown concentrations can be estimated by thresholding. Note also that one can create tests with different amounts from different samples, i.e. presence of a sample in a pool is no longer a binary property. Perhaps combinations of techniques from [4] and the present paper are required. These questions are beyond the scope of the present article and are left for future research.

Threshold group testing as introduced here is perhaps not a suitable model for chemical testing, however one can imagine applications of a different nature that justify our model. The n elements may represent distinct "factors" that can be either present or absent: If a number of relevant factors (the positives) are present, some event (YES answer) will be observed. For example, some disorders appear only if various bad factors come together, and they will appear for sure if the bad factors exceed some limit. The relevant factors can have complex interactions and different importance, so that the outcome does not solely depend on their number, but on some monotone property of subsets. (To mention an example, there could be several types of positives, and the result is positive if positive elements of each type are present.) Now, one may wish to learn what all these risk factors are, among the n candidates. If one can create experiments where the factors can be arbitrarily and independently turned on and off (as in some knock-out experiments in cell biology), one can also use these experiments as pools in a search strategy. Thus, threshold

group testing strategies may find interest in such settings, although we studied the problem mainly as a nice extension of group testing, without a concrete application in mind.

Threshold group testing is also related to another search problem, called "guessing secrets", that recently received attention due to a surprising application in Internet routing [1, 9]. Here the adversary knows a set X of secrets and answers YES if all secrets are in the query set S, NO if S and X are disjoint, and arbitrarily in all other cases. Hence, this is threshold group testing in the special case $l = 0$ and $u = p$.

Overview of the paper: In Section 2 we show that the p positives can be found efficiently, subject to at most g wrongly classified items, which is inevitable in the worst case. Still, for $p \gg g$ this means a small relative error. The rest of the paper revolves round the question how many tests and computational steps are needed to achieve this best possible classification. As a first result, the computation time is bounded by $O(p^u n^{g+1})$, with the thresholds in the hidden constant. In Section 3 we discuss the case $g = 0$ and show that the asymptotic test complexity does not exceed that of classical group testing, even if no auxiliary pool with a known number of positives is available. The idea of the strategy is then enriched in Section 4 to treat the general and more realistic case $g > 0$. The main result is that the asymptotic number of tests can be made linear in p times a power of n with an arbitrarily small positive exponent (at cost of the constant factor), and computations need polynomial time. It is hard to say how far this is from optimal, but it means significant progress compared to the trivial bound of $\binom{n}{u}$ tests. Sections 5 and 6 are merely supplements and directed towards further research. We point out some possibilities to modify the algorithms and accelerate them further. Open questions are also mentioned in earlier sections when they arise.

It will become apparent that the algorithms do not need a bound on p, that is, the results hold even if the number of positives is not known in advance.

In this paper we consider only sequential test strategies. In applications with time-consuming tests it can be desirable to work in a limited number of stages where a set of tests is performed nonadaptively, i.e. without waiting for the results of each other. The outcomes of all previous stages can be used to set up the tests for the next stage. For some problems, nonadaptiveness is achievable only at cost of a higher test complexity. These tradeoffs have been investigated for various combinatorial search problems, among them classical group testing [7, 10] and learning relevant variables [5, 6]. Nontrivial results in this direction for threshold group testing would be interesting, too.

Notational remarks: Throughout the paper, a k-set means a set of k elements. A k-subset is a k-set being subset of some other set which is explicitly mentioned or clear from context.

We choosed to present complicated bounds containing binomial coefficients in the form of powers and factorials. One might argue that resolving the binomial coefficientsis not elegant, however one can easier see the dependency on the actual input parameters n and p in this way.

2 Fuzzy Identification of the Positives

First we show that we can always get close to the true set P of positives in a certain sense, using much fewer than 2^n tests (that would be needed for testing all possible pools).

Theorem 1 *If $p \geq u + g$, the seeker can identify a set P' with $|P' \setminus P| \leq g$ and $|P \setminus P'| \leq g$. For $p < u + g$, we still have the error bound $u - 1$ rather than g.*

Proof. Simply test all u-sets. Let P' be any set of alleged positives that is consistent with the answers. That means: For every pool that contained at most l members of P', the test said NO, and for every pool that contained at least u (and thus exactly u) members of P', the test said YES.

Suppose for the moment that P' has cardinality $p' \geq u$. Assume that P' has $g + 1 = u - l$ (or more) elements outside P. Add l other elements from P'. This u-set has at most l elements in P, thus it must return NO. But since it consists of u elements from P', it should also say YES, contradiction.

Similarly, suppose for the moment $p \geq u$, and assume that $g + 1$ (or more) elements of P are not in P'. Take them and add l other elements from P. This u-set is entirely in P and therefore says YES. But since it contains at most l elements from P', it should also say NO, contradiction.

For $p \geq u$ we have seen that $p' \geq p - g$. In particular, $p \geq u + g$ implies $p' \geq u$, and both directions apply. The last assertion is also obvious now. □

Some lower bound on p is necessary to guarantee that the sizes of set differences are bounded by g, even if we test all 2^n subsets. For example, if $p = u - 1$ and the adversary always says NO, every $(u - 1)$-set P' is a consistent solution, even if P' and P are disjoint. But we have $u - 1 > g$ for $l > 0$. In the following we always suppose $p \geq u + g$. Our bound g on the number of misclassified elements on both sides is tight in the following sense:

Proposition 2 *For any fixed set P' with $|P' \setminus P| \leq g$ and $|P \setminus P'| \leq g$, the adversary can answer as if P' were the set of all positives.*

Proof. For pools S with $l < |S \cap P| < u$, the adversary answers arbitrarily, in particular, consistently with P'. It remains to show that the mandatory answers for all other pools S do not rule out P'. If S has at most l positives then the answer must be NO. Since S contains at most $l + g = u - 1$ elements from P', this answer is possible for P' as well. If S has at least u positives then at least $u - g = l + 1$ of them are in P'. Thus, answer YES is also allowed for P'. □

We stress that this result refers only to any single solution P'. It is much more difficult to characterize the families of hypotheses P' that are consistent with some vector of answers. We leave this as an open question. However, we can say at least something about these families:

Proposition 3 *Any two solutions Q and Q' being consistent with the answers of an adversary satisfy $|Q \setminus Q'| \leq g$ and $|Q' \setminus Q| \leq g$.*

Proof. Suppose that some inequality is violated. Then, if $Q = P$, Q' cannot be a candidate. But $Q = P$ is impossible means that Q is not a candidate either. □

Clearly, this statement is not specific to our problem. It holds similarly for any search problem where the consistent hypotheses are close to the target, with respect to some distance measure.

As an illustration we discuss what Proposition 3 means in the case $l = 0$, $u = 2$. Every candidate solution P' can be characterized by the "deviation" $\{x, x'\}$, with unique elements $x \in P \setminus P'$ and $x' \in P' \setminus P$. (One of the set differences may be empty. Then x or x' does not exist, and the deviation is a singleton set.) For deviations $\{x, x'\}$ and $\{y, y'\}$ of any two solutions, not all of x, x', y, y' can exist and be different. Hence the solution space can be represented as a star graph. Either one positive is undetected and several (maybe all) negatives are suspicious to be the only positive that is missing, or one negative is not absolved, but several alleged positives are candidates for the missing negative.

The solution spaces for $u > 2$ could be more complicated. A particularly interesting question is under what conditions a positive or negative element can be recognized as such.

Our next concern is to compute a set P' with the properties mentioned in Theorem 1 from the $\binom{n}{u}$ answers efficiently, that is, to "invert the answers". Let us call a set Q an *affirmative set* if all u-subsets of Q answered YES. Certainly, P is affirmative, as well as every subset of P. On the other hand, an affirmative set Q cannot exceed P very much: $|Q \setminus P| \leq g$, otherwise we get an obvious contradiction. This suggests the strategy to establish increasing affirmative sets, until the second condition $|P \setminus Q| \leq g$ is also satisfied. The somewhat tricky question is how to reach this situation quickly and to recognize it although P is unknown.

Theorem 4 *For $p \geq u + g$ one can find some set P' as in Theorem 1 in time $O(\frac{u^{g+1}}{(u-1)!(g+1)!} p^u n^{g+1})$.*

Proof. As our initial Q we may choose any u-set that answered YES.

Assume that we have an affirmative Q with $|P \setminus Q| > g$. There exists some $(g+1)$-set G with $Q \cap G = \emptyset$, and a set H of size at most g, such that $(Q \cup G) \setminus H$ is affirmative (and by at least one element larger than Q). To see the existence, just notice that any $G \subseteq P \setminus Q$ with $G = |g + 1|$ and $H = (Q \cup G) \setminus P$ would do. Here we use the fact that at most g members of Q are outside P.

In order to find a pair G, H with the desired property, we may try all $\binom{n}{g+1}$ candidates for G. For any fixed G we can find a suitable H, if it exists, by the following observation: H must be a hitting set for the family of u-subsets of $Q \cup G$ that said NO.

We bound the number of such sets. Since Q is affirmative, at least one element must be in G, these are $g + 1$ possibilities. Each of the other $u - 1$ elements can be in Q or be one of g elements of G. An upper bound on $|Q|$ is provided by $|Q \setminus P| \leq g$ which implies $|Q| \leq p+g$. As long as $|P \setminus Q| > g$ (true in every step except the last), this improves to p. Hence the size of our set family is less than $\frac{g+1}{(u-1)!}(p+g)^{u-1}$. By a trivial branching algorithm, some hitting set of cardinality at most g can be found in $O(\frac{u^{g+1}}{(u-1)!}(p+g)^{u-1})$ time.

In the worst case, Q has to be augmented about p times. If, for some Q, every G fails, then at most g positives are not yet in Q, and we can output $P' := Q$. □

There may be room for improving the time bound, mainly because the method above considers all G separately. On the other hand it seems that the number p of augmentation steps cannot be reduced substantially, since we might include up to g new negatives in Q in every step. A more ambitious question is how much computation is needed to output (some concise representation of) the complete solution space for the given answers.

We resume that, for any fixed thresholds, we can approximate P with a polynomial number of tests and in polynomial computation time, subject to a constant number of misclassified elements. But in general we will not be able to identify P exactly. The practical interpretation is: For any element in P' we have evidence that it might be a positive and should be investigated more closely, although there might be a few false positives in P'. Symmetrically, any randomly picked element not in P' is innocent with high probability, although all these elements remain suspicious, in the worst case. This might be already acceptable for applications. Also recall that a less malicious adversary (e.g. with random answers between the thresholds) which is more appropriate to model "nature" in experiments can only give more safety.

3 The Case Without Gap

In this section we study the case $g = 0$, that is, $l + 1 = u$. Theorem 1 implies that we can identify P exactly. Our preliminary bound $\frac{n^u}{u!}$ on the number of queries and the complexity bound in Theorem 4 are however not very satisfactory, because the fact $g = 0$ should help. (However, we will use this preliminary result later, inside a more efficient strategy for general g.)

Suppose that we can create a pool with $u - 1$ positives. Then a simple trick (from e.g. [8]) reduces the problem to classical group testing: Take $u - 1$ positives with you and add them to every pool. Then a pool says YES iff at least one further positive is present.

Still, it is interesting to ask what we can do if such an auxiliary pool is not available. For potential applications it is not clear whether one can always prepare a pool with $u - 1$ known positives right away. Instead, we first have to identify $u - 1$ positives by group tests. Another reason to address the problem is that our procedure for detecting $u - 1$ positives will also provide the idea for attacking the more complicated and more realistic case $g > 0$ in the next section.

Our approach is as follows: First test the whole set. If it says YES, split it in two halves and test them. If at least one half say YES, split it, and continue recursively. (We ignore the other half for the moment.) This way we get smaller and smaller sets that contain at least u positives. This trivial process stops if, at some point, both halves say NO.

In order to continue we need set families with a special covering property: For the moment let t, r, v be any integers with $1 \le t \le r \le v$. On a set of v elements we want a family of r-sets such that every t-set is subset of some member of this family. A lower bound on the size (number of sets of) such families is $\binom{v}{t}/\binom{r}{t}$, and for any

constant t, r and large v it is known that families within a $1 + o(1)$ factor of this bound exist (the Rödl nibble, see e.g. [2]). However we cannot exploit this result here, as we need large r. Therefore we use other, rather simple covering families. More material on this kind of structure can be found e.g. in [3].

Back to a set of, say, m elements that contains at least u positives. We split our set into $u+1$ parts of roughly equal size. Then we test the $u+1$ pools consisting of u of these parts. Clearly, this is a covering family, i.e. we find a pool which contains at least u positives and therefore says YES. Clearly, by iterated application we can reduce m to some constant size depending on u only, using $(u+1)\log_{(1+1/u)} m = \frac{u+1}{\ln(1+1/u)} \ln m = O(u^2 \log m)$ tests. Finally we find u positives by exhaustively testing the u-subsets. It follows:

Lemma 5 *In a set of m elements, at least u of them positives, we can identify u positives by $O(u^2 \log m)$ tests.* □

It would be interesting to close the gap to the information-theoretic lower bound $O(u \log m)$. We remark that covering families may be used right from the beginning in our procedure. Starting with binary search does not improve the worst-case bound, but it is easier as long as we get YES answers. Moreover, we may split the set in halves at random, so that u positives are in one half with high probability, as long as the actual number of positives is considerably bigger than u.

Now, take $u-1$ of the u known positives and use them as auxiliary pool in an ordinary group testing strategy. Since $O(p \log n)$ is a query bound for group testing (see e.g. [10]), we can now formulate the final result of this section.

Theorem 6 *Threshold group testing with $g = 0$ has asymptotic query complexity no worse than $O((p + u^2) \log n)$.* □

4 Gaps are Gulfs

Things become more complicated if there is a gap between the thresholds. We attempt to extend the idea above to the case $g > 0$. For convenience we say that *we (u, a)-cover an m-set* when we test all pools from a family of m/a-sets such that every u-set is contained in some pool. Here a can be any real number greater than 1. Besides the additive gap $g = u - l - 1$ between the thresholds, we also use the multiplicative gap h defined by $h := \lceil u/(l+1) \rceil$.

In the previous section we have used $a = 1 + 1/u$, but in the following we need an a that is large compared to u. By splitting an m-set in au subsets of roughly equal size and testing any combination of u of these subsets, we obviously get an (u, a)-covering of size $\binom{au}{u} = O(e^u a^u)$, where $e = 2.718\ldots$ is Euler's number. Thus we can formulate:

Lemma 7 *Any m-set has an (u, a)-covering of size $O((ea)^u)$, independently of m. Provided that at least u positives are present, the covering can be used to find an m/a-subset with still at least u positives, by $O((ea)^u)$ tests.* □

At least one pool in the covering gives a YES answer, however the catch is that, conversely, a YES implies only that $l+1$ or more positives are in that pool. We

cannot immediately recognize a pool with at least u positives and then continue recursively to narrow down the candidate set. This situation suggests a relaxation of the previous method.

We call a set with at least $l+1$ positives a *heavy* set. Suppose that we have $k \leq h$ disjoint heavy m-sets that together contain at least u positives. (Note that in case $k = h$, the union has guaranteed u positives, but a smaller $k \leq h$ can be sufficient in a concrete case.) In the union of our k heavy m-sets, we find a heavy m/a-set due to Lemma 7, by $O((eka)^u)$ tests. (Just replace m with km, and a with ka.) We put this heavy subset aside and repeat the procedure with the remaining elements, as long as it generates new disjoint heavy m/a-sets. This procedure is limited to sets with $m \geq au$, since the sets in the covering family must have size at least u. We summarize the procedure in

Lemma 8 *From $k \leq h$ heavy m-sets containing a total of at least u positives, we can obtain disjoint heavy m/a-sets and a remainder with fewer than u positives. The number of tests is $O((eha)^u)$ for each of these heavy m/a-sets.* □

This is the building block of an algorithm parameterized by a, that "distills" positive elements in heavy sets of decreasing size. We give a high-level description.

DISTILL:

At any moment, we maintain a collection of disjoint heavy sets of cardinality at most n/a^i, where i can be any non-negative integer, plus the set R of remaining elements, which is not necessarily heavy. A set with at most n/a^i but more than n/a^{i+1} elements is said to be on level $i \geq 1$. Set R is said to be on level 0. For i with $n/a^{i+1} < u \leq n/a^i$ we define the last level $(i+1)$ where only sets of cardinality exactly u are allowed.

The initial collection consists of only one set R on level 0, containing all elements. We apply two kinds of operation:

(i) (u, a)-cover the set on level 0 and move disjoint heavy subsets of it to level 1, as long as possible.

(ii) Take the union U of h heavy sets on level i, apply the procedure from Lemma 8 to U, and move the generated disjoint heavy subsets of U to level $i+1$. Finally move the rest of U back to R. If there are only $k < h$ sets left on level i, apply Lemma 8 to their union.

Clearly, these operations preserve disjointness of the collection, and all sets except perhaps R are heavy. The order of applying these operations is arbitrary. The process stops only when the collection has fewer than u positives on each level, since otherwise (ii) is still applicable. As the number of levels is bounded by $\log_a n$, all but $(u-1)\log_a n$ positives are eventually on the last level, where at least $l+1$ positives are in each u-set. (For the last level of u-sets, the procedure of Lemma 8 can be easily adjusted.)

For the following analysis we rewrite parameter a as $a = n^b$, where $b > 0$.

Theorem 9 DISTILL *needs* $O(\frac{e^u h^u}{l+1} h^{1/b} p n^{ub})$ *tests and packs all positives except* $\frac{u-1}{b}$ *in disjoint* u*-sets, such that at least* $l+1$ *positives are in each of these* u*-sets.*

Proof. It remains to analyze the number of tests. Consider any one of the, at most $\frac{p}{l+1}$, bunches of positives in the final u-sets. It has been extracted from at most h heavy sets on the previous level, by at most $O((eha)^u)$ tests due to Lemma 8. They in turn have been extracted from h^2 heavy sets, etc. Since only $\log_a n$ levels exist, all this has been done by $O(h^{\log_a n})$ applications of (i) and (ii). Hence, after $O(\frac{e^u h^u}{l+1} p a^u h^{\log n/\log a})$ tests, all positives except $(u-1)\log n/\log a$ are in disjoint u-sets. Finally set $a = n^b$. □

Despite the simplicity of this analysis, the bound has a quite pleasant form: The first factor consists of constants, the dependency on p is linear, and we can arbitrarily reduce the exponent of n by choosing a small b, of course at cost of an increasing factor $h^{1/b}$ and the loss of more positives.

More seriously, DISTILL stops with a fraction of positives which is only guaranteed to be at least $1/h$ in the final u-sets. (Recall that we only know of the presence of $l+1$ positives in each u-set.) In order to increase the guaranteed density of positives beyond this limit, we invoke, in a second phase called COMPRESS, the naive algorithm from Theorem 1, but now applied to the heavy u-sets only. Since their total size is at most hp rather than n, the bounds from Theorems 1 and 4 imply immediately:

Corollary 10 *After* $O(\frac{e^u h^u}{l+1} p h^{1/b} n^{ub} + \frac{h^u}{u!} p^u)$ *tests, all positives except* $\frac{u-1}{b} + g$ *are in a set that contains at most* g *negatives. The amount of computation is polynomial.* □

It seems possible to reduce the number of tests to $O(p \log n)$ for fixed thresholds l, u, using $O(\log n)$ levels and a potential function argument in the analysis, but we suspect that any $O(p \log n)$ algorithm would also misclassify $O(\log n)$ positives, which is of no value e.g. if $p = O(\log n)$.

5 Some Refinements

This more informal section will briefly discuss some refinements of the main result (Section 4). The aim is just to illustrate that our techniques have more potential. However, since optimality of the underlying algorithm is not known, we do not give a full treatment of these more sophisticated strategies here, and stress the ideas only.

In the case of large p, the p^u term in Corollary 10 may be prohibitive. An obvious modification of COMPRESS resolves the problem: Apply Theorem 1 to groups of t heavy u-sets, with some parameter t. This yields a tradeoff between final density of positives and test complexity. We can create from any t heavy u-sets a new set that contains all but g of the positives contained in the t heavy sets, and at most g negatives, by $O(\frac{u^u}{u!} t^u)$ tests. The density of positives is now at least $1 - \frac{g}{t(l+1)}$. Application to all our heavy u-sets costs $\frac{u^u}{(l+1)u!} p t^{u-1}$ tests. However we lose up to $\frac{gp}{t(l+1)}$ further positives. We might also iterate the two-phase algorithm above, in

order to catch more positives: The $\frac{gp}{t(l+1)}$ positives that escaped from Compress can be collected in a set of size $(h-1)p$. Then the whole algorithm may be executed (with suitable parameters) on this smaller set, etc.

Next we sketch a method for reducing the exponents in the test complexity bounds from u to $g+1$. The idea is to "shift the thresholds". As already mentioned in Section 3, if we knew already $q \leq l$ positives, we could inject them as an *auxiliary set* in every pool and reduce the threshold group testing problem for l and u to the same problem for thresholds $l-q$ and $u-q$. (Utilizing $q > l$ positives in an auxiliary set is useless, since the adversary may always say Yes, so that the searcher gets no information at all.) Specifically, we propose a modification of Distill with improved test complexity for $l > 0$. To explain the improvement we have to refer to details of Distill.

First we run Distill in a depth-first manner, that is, we move some positives to the last level as quickly as possible. Similarly as in the proof of Theorem 9, it costs at most $h^{u+1/b}n^{ub}$ tests to produce a heavy u-set. Then we try all $\binom{u}{l}$ l-subsets as an auxiliary set. At least one of them contains exactly l positives, so that we can use it to shift the thresholds to 0 and $u-l = g+1$, while running the original version of Distill on the remaining elements. The new upper threshold $g+1$ reduces the exponent of n, on the other hand we only incur another constant factor $O(\frac{u^l}{l!})$. Note that we have at least one positive in each of the final $(g+1)$-sets. Since we cannot see in advance which of the $\binom{u}{l}$ auxiliary sets has really l positives, we must find a good output among the $\binom{u}{l}$ offered results. For this purpose, we may test unions of u of the final $(g+1)$-sets. If there is in fact one positive in every $(g+1)$-set, then all answers must be Yes. Conversely, the Yes answers guarantee that at least $l+1$ positives are among the $u(g+1)$ elements in every such pool, which means a density of positives of at least $\frac{l+1}{u(g+1)}$. Since this is a constant, the resulting set has size $O(p)$.

Finally we apply a Compress phase on this set. By shifting the thresholds, the exponent of p can be reduced to $g+1$ as well. This could work as follows. Apply the algorithm from Theorem 1 to the $O(p)$ size candidate set, but with upper threshold $g+1$, and with help of each of the $\binom{u}{l}$ auxiliary sets. Again we have to find a good P' among the offered solutions. It suffices to rule out sets P' with large $d := |P' \setminus P|$. Then, among the surviving candidates P', a set with largest cardinality is close to P, up to constant differences (since we know that such a P' is among our candidates). Let us call a set Q *dismissive* if all $(g+1)$-subsets of Q answer No, together with a fixed auxiliary set. Note that $P' \setminus P$ is a dismissive set of size d, with respect to any auxiliary set, as they contain at most l positives. On the other hand, there is an auxiliary set (namely one with exactly l positives), such that every dismissive subset of P' can have at most $d+g$ elements. Hence, by computing maximum dismissive subsets of P' from the test results, we can identify one which is close to P, with differences of constant size.

As a consequence, the positive elements can be identified subject to constantly many misclassifications by $O(pn^{(g+1)b} + p^{g+1})$ tests, with an arbitrarily small constant $b > 0$.

6 Logarithmic Distill Phase for Threshold Two

A difficulty in the DISTILL phase in Section 4 is that an uncontrollable number of positives, but fewer than u, remain in U after every application of step (ii). We simply sent them back to level 0. However if we knew more about the number of positives in a heavy set, we might proceed more carefully and avoid returns, using more clever pool sets than just covering families. This section presents a partial result in this direction: a DISTILL method for $l = 0$, $u = 2$ that needs only $O(p \log n)$ tests. Together with threshold shifting this may even lead to improvements for $g = 1$ and general u. We also conjecture that similar test sets can be created for any fixed g.

Proposition 11 *For $l = 0$ and $u = 2$, all positives but one can be collected in a set of size at most $2p$ by $O(\log n)$ tests.*

With some constant $a > 1$, we start with $(2, a)$-covering the whole set. As long as we get at least one YES, we continue recursively on heavy sets of decreasing size. If this process goes through, we eventually obtain a heavy 2-set with one or two positives by $O(\log n)$ tests. Otherwise we know that the heavy set considered last has *exactly* one positive. We repeat the procedure, always starting with all elements that are not yet in the heavy sets.

This is nothing else than depth-first DISTILL with constant parameter a. The new idea comes now: Whenever we have two heavy sets A and B, each containing exactly one positive, we can even identify one of the positives by $O(\log n)$ further tests as follows. Split A and B into sets of roughly half size, denoted A_1, A_2, B_1, B_2. Test the pools $A_i \cup B_j$ for all $i, j \in \{1, 2\}$. If the positives are w.l.o.g. in A_1 and B_1 then $A_1 \cup B_1$ answers YES, and $A_2 \cup B_2$ answers NO. If the other two pools give the same answer, i.e. both YES or both NO, the seeker concludes that the positives must be in A_1 and B_1. If they give different answers, w.l.o.g. $A_1 \cup B_2$ say YES and $A_2 \cup B_1$ says NO, then $A_1 \cup B_2$ is another candidate. In either case, A_1 surely contains a positive, and we can discard A_2. The recursive process stops when one of A and B is a singleton.

We miss at most one positive, in case that a single heavy set is left over. □

Acknowledgments

This work was inspired by and profited a lot from discussions with Ugo Vaccaro, Ferdinando Cicalese, and Annalisa De Bonis during the author's stay at the Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno (Baronissi), Italy. Support from The Swedish Research Council (Vetenskapsrådet), project title "Algorithms for searching and inference in genetics", file no. 621-2002-4574, is also acknowledged. Thanks to the anonymous referees for careful reading and valuable suggestions.

References

[1] N. Alon, V. Guruswami, T. Kaufman, M. Sudan: Guessing secrets efficiently via list decoding, *13th SODA'2002*, 254-262

[2] N. Alon, J. Spencer: *The Probabilistic Method*, Wiley 1992.

[3] C.J. Colbourn, J.H. Dinitz (eds.): *The CRC Handbook of Combinatorial Designs*, CRC Press 1996

[4] P. Damaschke: The algorithmic complexity of chemical threshold testing, *3rd CIAC'97*, *LNCS* 1203, 205-216

[5] P. Damaschke: Adaptive versus nonadaptive attribute-efficient learning, *Machine Learning* 41 (2000), 197-215

[6] P. Damaschke: On parallel attribute-efficient learning, *J. of Computer and System Sciences* 67 (2003), 46-62

[7] A. De Bonis, L. Gasieniec, U. Vaccaro: Generalized framework for selectors with application in optimal group testing, *30th ICALP'2003*, *LNCS* 2719, 81-96

[8] A. De Bonis, U. Vaccaro: Improved algorithms for group testing with inhibitors, *Info. Proc. Letters* 66 (1998), 57-64

[9] F. Chung, R. Graham, F.T. Leighton: Guessing secrets, *Electronic J. on Combinatorics* 8 (2001), 1-25

[10] D.Z. Du, F.K. Hwang: *Combinatorial Group Testing and its Applications*, 2nd edition, World Scientific 2000

[11] E.H. Hong, R.E. Ladner: Group testing for image compression, *IEEE Transactions on Image Proc.* 11 (2002), 901-911

[12] H.Q. Ngo, D.Z. Du: A survey on combinatorial group testing algorithms with applications to DNA library screening, in: *DIMACS Series Discrete Math. and Theor. Computer Science* 55, AMS 2000, 171-182