A Toolbox for Provably Optimal Multistage Strict Group Testing Strategies

Peter Damaschke and Azam Sheikh Muhammad

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

Abstract. Group testing is the problem of identifying up to d defectives in a set of n elements by testing subsets for the presence of defectives. Let $t(n, d, s)$ be the optimal number of tests needed by an s-stage strategy in the strict group testing model where the searcher must also verify that no more than d defectives are present. We develop combinatorial tools that are powerful enough to compute many exact $t(n, d, s)$ values. This extends the work of Huang and Hwang (2001) for $s = 1$ to multistage strategies. The latter are interesting since it is known that asymptotically nearly optimal group testing is possible already in $s = 2$ stages. Besides other tools we generalize d-disjunct matrices to any candidate hypergraphs, which enables us to express optimal test numbers for $s = 2$ as chromatic numbers of certain conflict graphs. As a proof of concept we determine almost all test numbers for $n \leq 10$, and $t(n, 2, 2)$ for some larger n.

1 Introduction

In the *group testing* problem, a set of n elements is given, each being either *defective (positive)* or *non-defective (negative)*. Let P denote the unknown set of positive elements. A *group test* takes any subset Q of elements, called a *pool*. The test (or pool) is positive if $Q \cap P \neq \emptyset$, and negative otherwise. In the latter case, obviously, all elements in Q are recognized as negative. The goal is to identify P using few tests. A group testing strategy may be organized in s *stages*, where all tests within a stage are executed in parallel. In *adaptive* group testing s is not limited, hence tests can be done sequentially. Case $s = 1$ is called *nonadaptive*. Small s are desired in applications where the tests take much time.

It is expected that $|P| \leq d$, for some fixed bound d, with the understanding that $|P| > d$ is unlikely but not impossible. A searcher wants to identify P if $|P| \leq d$, and just report "$|P| > d$" otherwise. This setting is called *strict group testing*, in contrast to *hypergeometric group testing* where $|P| \leq d$ is "promised" to the searcher. It was argued in, e.g., [1] that strict group testing is preferable. It does not rely on the assumption $|P| \leq d$.

In a few lines one cannot possibly give even a cursory overview of the applications of group testing in biology and computer science, and of the main complexity results. We only refer to the books [7, 8] and a few recent papers [2, 5, 11, 16, 19] as entry points to further studies. As is well known [9], $O(d \log n)$ tests are not enough if $s = 1$. The breakthrough result in [6] showed that $O(d \log n)$ tests are sufficient already if $s = 2$, followed by improvements of the hidden constant factor [10, 3]. However, such asymptotic results are designed for optimal behavior as n goes to infinity, but even asymptotically optimal strategies do not necessarily entail the optimal strategy for specific input sizes n. Another motivation of the quest for optimal strategies for specific n is that the pool sizes of asymptotically optimal strategies increase with n, but in some applications, large pools may be infeasible (because of technical obstacles, chemical dilution, etc.). Still we can split an instance into many small instances and solve them independently, now each with optimal efficiency. To mention a practical example, screening blood donations for various infectious diseases is performed at some labs in instances ("minipools") of, e.g., 16 samples [18], and group testing is proposed [20] to reduce the waiting times (can be days) and the considerable costs (for millions of donations annually).

Due to the preceding discussion, we define $t(n, d, s)$ to be the optimal worst-case number of tests needed by a strict group testing strategy for n elements, up to d defectives, and s stages. Some *monotonicity* relations hold trivially: If $n \leq n'$, $d \leq d'$, and $s \geq s'$ then $t(n, d, s) \leq t(n', d', s')$. If $t(n, d, s) = t(n, d, n)$, we write $t(n, d, s+)$ to indicate that more stages would not lower the test number.

The work closest to ours is [15] from which the exact $t(n, d, 1)$ values follow for all $n \leq 14$ and arbitrary d, as well as some test numbers for larger n. The novelty of the present work is that we extend the scope to $s > 1$ stages, which saves tests compared to $s = 1$. To our best knowledge, this terrain has not been systematically explored, with a few exceptions: $t(n, 1, 1) = \log_2 n + o(\log_2 n)$ is the smallest k with $\binom{k}{k/2} \geq n$ due to [17], $t(n, 1, 2+) = \lceil \log_2 n \rceil + 1$ is a side result in [4], and [12] gives partial results on adaptive strategies. The focus in [12, 15] is on the question for which n, d group testing saves tests, compared to trivial individual testing.

While an upper bound on a specific $t(n, d, s)$ is established once a strategy is found, the main challenge is to prove matching lower bounds. We stress that this cannot be done naively by considering all possible pooling designs in every stage, as their number is doubly exponential in n. Instead we have developed several combinatorial tools. Each of our lemmas viewed in isolation is relatively simple, but *together they enable us to determine many $t(n, d, s)$ values exactly*, by a branch-and-bound approach. Yet they should also be of independent interest as structural results. Due to severe space limitations, this paper version demonstrates the use of the theory only for $n \leq 10$ (and all d, s) and for $t(n, 2, 2)$, although even more $t(n, d, s)$ could be managed with our current techniques.

2 Notation

A k-set ($\leq k$-set, $\geq k$-set) is a set with exactly (at most, at least) k elements. We use this notation also for pools, hyperedges, and other sets with special roles. A *hypergraph* is a set of *vertices* equipped with a family of subsets called *hyperedges*.

A hypergraph with only ≤ 2-hyperedges is a *graph* with *edges*. A 1-hyperedge is a *loop*. Note that we allow *parallel* hyperedges, that is, hyperedges being identical as sets may occur multiple times. Two sets are *incomparable* if neither is a subset of the other one. A family of pairwise incomparable sets is an *antichain*. We use standard graph-theoretic symbols: K_n, C_n, $K_{m,n}$ is the clique, cycle, and complete bipartite graph, respectively, with the indicated vertex numbers. A *forest* (union of trees) is a cycle-free graph, and a *leaf* is a vertex incident to only 1 edge.

During the course of applying a group testing strategy, an element which has not appeared in any negative pool so far is called a *candidate element*. A *candidate set* is a set of up to d candidate elements that is consistent with all previous test outcomes. That is, any candidate set is possibly the true set P. The name candidate element reflects the searcher's knowledge: An element is possibly positive until it is discarded, as the member of some negative pool. Therefore we can have candidate elements outside all candidate sets, called *dummy elements*. For example, if $n = 5$ and $d = 2$, and we test 2 disjoint 2-pools with positive outcome, then the 5th element was in no negative pool so far, but any candidate 2-set must take one element from both pools.

In the *candidate hypergraph*, the candidate elements are the vertices and the candidate sets form the hyperedges. For $d = 2$, the candidate hypergraph is just a *candidate graph*, possibly with loops. From the definitions it follows that an instance of the strict group testing problem is solved if and only if the candidate hypergraph has exactly one hyperedge and no dummy elements.

A *pool hypergraph* represents a set of pools in dual form: Vertices $p_1, p_2, p_3, \ldots$ are the pools, and hyperedges are the candidate elements. A vertex belongs to a hyperedge if the corresponding pool contains the corresponding element. Dummy elements are represented as loops at a symbolic *null vertex* p_0.

3 Lower-Bound Tools

The simple *counting bound* says that the number of tests is at least $\log_2$ of number of outcomes. In particular we have $t(n, d, n) \geq \log_2(1 + \sum_{i=0}^{d} \binom{n}{i})$ where the summand 1 accounts for the outcome "$|P| > d$". First we give a more powerful lower bound, which often guarantees one more test.

Lemma 1. *If $m > 2^r$ candidate sets exist which form an antichain, then strict group testing requires at least $r + 2$ tests, even adaptively.*

Proof. It suffices to consider $m = 2^r + 1$. We use induction on r. We first prove that at least 2 tests are required for the base case $r = 0$, that is, $m = 2$. Let C and C' be incomparable candidate sets. Assume that one pool is enough. If it intersects either none or both of C and C', it cannot distinguish between these candidate sets. Thus the pool must intersect C and be disjoint to C' (or vice versa). Now a positive outcome leaves the searcher unsure about the status (defective or not) of the elements in $C' \setminus C$. For the inductive step assume that the claim is true for r, and let $m = 2^{r+1} + 1$. In one test outcome, the majority of candidate sets remain. Therefore, after the first test we would keep at least 2^r+1 candidate sets. Using the inductive hypothesis the claim holds for $r+1$. □

Examples. For $d = 2$ we list, for some pool hypergraphs, the test numbers enforced by Lemma 1 if all pools respond positively. Note that candidate sets of the same size form an antichain. These test numbers will be used to get our specific $t(n, d, s)$ results.

2 loops at a vertex: 2 candidate 1-sets, 2 tests.
2 loops at a pool vertex and 1 loop at p_0: 3 candidate 2-sets, 2 tests.
3 loops at a vertex: 3 candidate 2-sets, 3 tests.
4 loops at a vertex: 6 candidate 2-sets, 4 tests.
2 loops at distance 1 or 2: 3 candidate sets, 3 tests.
C_3: 3 candidate 2-sets, 3 tests.
C_4: 2 candidate 2-sets, 2 tests.

The following lemmas are evident monotonicity observations (proved by a "the searcher gets less information"-argument), but they are extremely useful for limiting the strategies that must be considered in lower-bound proofs. Recall that we consider strict group testing in a prescribed number s of stages.

Lemma 2. *In response to a given deterministic test strategy, consider a test answering strategy A that enforces t tests in the worst case. If the searcher replaces some pool Q, that is negative (positive) in A, with a subset (superset) of Q, then still at least t tests are needed in the worst case.* □

Lemma 3. *Suppose that the outcomes of some pools of a stage are revealed to the searcher, and then she may redesign her other pools from scratch. If t further tests are not sufficient despite redesign, then they are not sufficient for the original problem instance either.* □

Pools can be arranged as a Boolean matrix. It is well known that a pooling design solves *strict* group testing for $s = 1$ if and only if it forms a d-disjunct matrix. We withhold the definition of d-disjunct, as we present a straight generalization to arbitrary candidate hypergraphs. (This should not be confused with the concept in [13] which addresses a different group testing problem.)

Theorem 1. *A nonadaptive strategy solves strict group testing for a given candidate hypergraph if and only if, for every pair of a candidate set C and a candidate element $v \notin C$, it has a pool Q such that $v \in Q$ and $C \cap Q = \emptyset$.*

Proof. To prove necessity, let C be a candidate set and $v \notin C$ a candidate element. If $C = P$ then v must be recognized as negative. Hence some negative pool must contain v, in particular, this pool must be disjoint to C. Next we prove sufficiency. If $P = C$ then clearly all elements outside C will be recognized as negative, and all elements in P are still candidate elements. Suppose $D \subset P$ is also a candidate set. Then the searcher can falsify the assumption $P = D$, since then all elements in $C \setminus D$ would be negative and be recognized as such, too. Hence P is the only remaining candidate set, and no dummy elements remain. The case when P is none of the candidate sets (but $|P| > d$ instead) is also easily recognized by the fact that more than d candidate elements are retained. □

A *partial vector* is a vector where any position either has a *fixed* value 0 or 1, or remains *open*, indicated by the $*$ symbol. We index the candidate elements by $i = 1, 2, 3, \ldots$, and we encode every pair of a candidate set C and a candidate element $v \notin C$ as a partial vector as follows. We assign 0 to all positions of elements of C, and 1 to the position of v. All other positions are open. Two partial vectors *conflict* if one has 0 while the other one has 1 at the same position. Two partial vectors that do not conflict are *compatible*. We translate the candidate hypergraph into a *conflict graph* defined as follows. The vertices represent the partial vectors for all C and $v \notin C$, and two vertices are adjacent if the corresponding partial vectors are in conflict. A pool is naturally represented as its indicator vector, that is, a bit vector with 1 at all positions of elements in the pool, and 0 elsewhere. A pool is said to *cover* a partial vector if all fixed positions in the partial vector have the same value, 1 or 0, as in the pool's indicator vector. The smallest number of colors needed to color a graph G, such that adjacent vertices get distinct colors, is known as chromatic number $\chi(G)$. We refer to $\chi(G)$ of a conflict graph G as *conflict chromatic number*.

Theorem 2. *Solving the strict group testing problem nonadaptively for a given candidate hypergraph is equivalent to coloring the conflict graph. Consequently, the conflict chromatic number equals the number of tests required.*

Proof. The condition in Theorem 1 can be equivalently expressed saying that, for every pair of a candidate set C and candidate element $v \notin C$, some pool must cover its partial vector. Observe that a single pool can cover a set of partial vectors if and only if they are pairwise compatible. In the conflict graph, compatible partial vectors are represented by nonadjacent vertices. Thus, a subset of partial vectors can be covered by a single pool if and only if the vertices form an independent set. Since partitioning a graph into a minimum number of independent sets is equivalent to graph coloring, the assertion follows. □

Of course, the fact that graph coloring is NP-hard does not stop us from computing $\chi(G)$ for specific conflict graphs G needed for our purposes. As a first example, the candidate graph $C_4 = K_{2,2}$ has conflict chromatic number 4, since the partial vectors $[010*]$, $[0*10]$, $[*001]$, $[10*0]$ conflict pairwise. Therefore, if $d = 2$, $s = 2$, and the pool graph used in stage 1 has 2 vertices with 2 loops each, then at least 4 more tests are required in stage 2.

A nonadaptive strategy on the candidate graph $K_{1,k}$ requires $t(k, 1, 1)$ tests, as the central vertex together with either leaf is a candidate set. For instance, if $d = 2$, $s = 2$, and the pool graph used in stage 1 has an edge p_1p_2 and a total of k loops at p_1, p_2, and p_0, then we get this situation. Now let $K_{1,k} + e$ denote the k-star $K_{1,k}$ with one extra edge between two of the leaves. We can prove the necessity of 1 more test using the conflict chromatic number:

Lemma 4. *A nonadaptive strict group testing strategy for the candidate graph $K_{1,k} + e$ requires $t(k, 1, 1) + 1$ tests.*

Proof. Assume that $t(k,1,1)$ tests are enough. Leaving aside the candidate 2-set represented by the extra edge e, we first consider the partial vectors due to the k edges of the k-star. (See definitions prior to Theorem 2.) Let the first position correspond to the center of the k-star, while the further k positions correspond to the leaves. All partial vectors from the k-star have the form $[0\ldots]$, since all edges share the center vertex. The second defective is either of the k leaves. Thus we need already $t(k,1,1)$ pools to cover (or "color") all these partial vectors. Without loss of generality let e be the edge between the 2nd and 3rd vertex. Among the partial vectors due to e, we have in particular $[100*\ldots]$. This conflicts all earlier partial vectors with 0 at the 1st position. Thus, another color (i.e., pool) is needed to cover this partial vector. $\square$

Example. Let $n=4$, $d=2$, $s=2$, let the pool graph in stage 1 be C_3 with 1 loop at 1 vertex, and these pools are positive. Then the candidate graph is $K_{1,k}+e$, hence $t(3,1,1)+1=4$ tests are needed in stage 2.

Let G_1 and G_2 be any two candidate hypergraphs on disjoint sets of vertices (elements). We define their *product* $G_1 \times G_2$ as the candidate hypergraph whose vertices are all elements from G_1 and G_2, with the hyperedges $e_1 \cup e_2$ for any pair of hyperedges e_1 from G_1, and e_2 from G_2. Let t_i be the optimal number of tests for nonadaptive strict group testing when G_i is given ($i=1,2$), that is, the respective conflict chromatic number. Trivially, $G_1 \times G_2$ needs at most t_1+t_2 tests, since we may consider $G_1 \times G_2$ as two independent "parallel" problem instances. A natural conjecture is that t_1+t_2 is also optimal. However the difficulty is that the searcher is free to use pools intersecting both vertex sets, which may cleverly save some tests. In fact, we are able to prove the conjecture in special cases only, yet they are powerful for our purposes. The prototype is the following case that uses similar reasoning as in Lemma 4.

Lemma 5. *Let G_1 be the candidate graph with hyperedges $\{v_1\}$ and $\{v_2\}$ (that is, exactly one of these 2 elements is defective), and G_2 arbitrary. Then nonadaptive strict group testing on $G_1 \times G_2$ needs t_2+2 tests.*

Proof. Let $v_3, v_4, \ldots$ denote the elements of G_2. Pools must cover all partial vectors according to Lemma 1 and Theorem 2. First consider the candidate sets of $G_1 \times G_2$ where v_1 is positive. Their partial vectors have the fixed value 0 at the 1st position. Hence t_2 pools are needed to cover already these partial vectors. Assume that t_2+1 pools are sufficient for $G_1 \times G_2$. Every partial vector of the form $[10\ldots]$, with further 0s at the positions of some candidate set from G_2, conflict with the t_2 former pools. Hence all partial vectors $[10\ldots]$ must be covered by the same last pool. Since all v_i, $i>2$, are candidate elements in G_2 and give rise to $0s$, it follows that this last pool can only be $\{v_1\}$. The key step is that, by symmetry, the pool $\{v_2\}$ must also exist. Now the indicator vectors of these pools, $[100\ldots0]$ and $[010\ldots0]$, conflict with all t_2 pools needed to cover the partial vectors from G_2, since each of these has fixed value 1 at some position v_i, $i>2$. In total we need t_2+2 pools. $\square$

We can similarly prove a more general version of Lemma 5, however with a rather technical condition to G_1. We omit this elaboration. In the reported examples we will only apply the above basic case. One consequence that we use is that the candidate graph $K_{2,n}$ needs $t(n,1,1)+2$ nonadaptive tests. The power of Lemma 5 is also illustrated by the following example: Consider the product of k copies of the above G_1, that is, k disjoint pairs of elements, where one element of each pair is defective. Since 2^k candidate k-sets exist, the counting bound is only k tests, whereas inductive application of Lemma 5 gives the tight lower bound $2k$.

4 Upper-Bound Tools (Sub-Strategies)

Lemma 6. *Any candidate hypergraph of m candidate sets permits a nonadaptive strict group testing strategy with at most m tests.*

Proof. Test all complements of candidate sets. These m pools completely adhere to Theorem 1: For every candidate set C and candidate element $v \notin C$, the complement of C includes v and is disjoint to C. The assertion follows. $\square$

Lemma 7. *Let G be a candidate hypergraph with $n > d$ vertices where all candidate sets are d-sets. Let G' be obtained from G by adding dummy elements. Then the optimal test number on G' (for unchanged d, s) is the same as for G.*

Proof. Add the dummy elements to every pool of an optimal strategy for G. Since $n - d > 0$ elements must be discarded, in every possible application of the strategy (i.e., for any test outcomes), at least one pool is negative, or we recognize $|P| > d$. This negative pool also discards the dummy elements. $\square$

For instance, candidate graph $K_{1,3} + e$ can be solved with 3 tests when we allow 2 stages: Let the candidate 2-sets be the edges v_1v_2, v_1v_3, v_1v_4, v_2v_3. We only test $\{v_2\}$ in stage 1. Either positive or negative, there remain 2 candidate 2-sets for stage 2 and perhaps dummy elements, which requires by Lemma 6 and Lemma 7 only 2 more tests in stage 2.

5 Optimal Strategies for Small Instances

Our aim is now to get exact values $t(n,d,s)$ for as many feasible combinations of n, d, s as possible. Recall that the $t(n,1,1)$, and $t(n,1,2+)$ are already completely known, thus we study $d \geq 2$ only. The methodology can be summarized as follows. In the pool (hyper)graphs in stage 1 we identify certain subsets W of vertices. If all pools in W are positive and the others negative, the candidate elements are precisely the edges in the subgraph induced by W, and the defective edges must cover W. Then we show that the resulting candidate (hyper)graphs enforce too many further tests, by our lower-bound techniques. – This "practical" section is intended to be a proof of concept. Readers may skip any items without losing their thread. Strategy descriptions are highlighted by **(S)**.

$\mathbf{t(3,2,1+)=3}$, $\mathbf{t(4,2,1+)=4}$, $\mathbf{t(5,2,1+)=5}$ hold by the counting bound. For $n=6$ the counting bound gives only $t(6,2,1+)\geq 5$, nevertheless we can prove:

$\mathbf{t(6,2,1+)=6}$. It suffices to consider adaptive strategies. If we begin with a 1-pool and it is negative, we need $t(5,2,1+)=5$ further tests. If we begin with a 2-pool and it is positive, there remain $9>2^3$ candidate 2-sets, hence Lemma 1 requires 5 more tests. Due to Lemma 2 we need not consider more cases.

$\mathbf{t(7,2,3+)=6}$. The lower bound follows from $t(6,2,3)=6$. **(S)** For the upper bound, use 3 mutually disjoint 2-pools in stage 1. If at most 1 of them responds positive, at most 3 candidate elements are left, that can be tested individually in stage 2. If all 3 of them respond positive, we conclude $|P|>2$. If exactly 2 of them respond positive, then, in stage 2, we query separately 1 element from each positive pool (2 tests). A negative outcome means the other element is positive, whereas a positive outcome renders the queried element positive. Thus, one positive element is recognized in both cases. A 6th test in stage 3 on the remaining candidates confirms they are negative, or yields $|P|>2$.

$\mathbf{t(7,2,2)=7}$. We assume for contradiction that $t(7,2,2)\leq 6$ and consider the pool hypergraph of stage 1. Assume that some $\geq$ 3-hyperedge e exists. If e is positive, e explains 3 or more positive pools. Since $t(6,1,2)=4$, the searcher needs 4 more tests for the 2nd defective. Hence the pool hypergraph is merely a graph. Next, let p be a pool vertex with degree 1. Let p be negative and apply Lemma 3. The edge incident to p is negative, hence still 2 defectives among 6 elements must be found, and 1 pool is used up. Thus $t(6,2,2)+1=6+1=7$ tests are needed. Hence the minimum degree is 2. Assume that parallel edges exist, that is, 2 pools share 2 or more elements. Declare these 2 pools positive and and apply Lemma 3 together with 1. Since still at least $11>2^3$ candidate 2-sets remain, and 2 pools are used up, the searcher needs $2+3+2$ pools. Altogether, the pool hypergraph must be a graph of minimum degree 2 without parallel edges. It has at most 5 pool vertices, since 6 pools would forbid a 2nd stage and require $t(7,2,1)=6$, contradicting the known $t(7,2,1)=7$ [15].

Next we can show that cycles C_3, C_4, C_5 together with edges or loops for the other elements always create bad induced subgraphs that enforce too many tests in stage 2 due to our Lemmas. Therefore the pool graph is a forest, perhaps with loops, and all leaves must have loops due to the minimum degree 2. Again, leaves at any distance create bad induced subgraphs, thus no edges other than loops can exist. (Details are omitted due to lack of space.)

It follows that all pool vertices are isolated and have at least 2 loops each. Since 2 such vertices imply already 4 more tests, we can have at most 2 pool vertices and p_0. By the pigeonhole principle, 2 vertices have at least 2 and 3 loops, respectively (or even 1 vertex has 5 loops). Hence the candidate graph contains $K_{2,3}$. Using Lemma 5, at least $t(3,1,1)+2=5$ more tests are required. Thus we can have only one pool vertex p_1. Since at least 2 loops are at p_1, the candidate graph contains $K_{2,5}$, and $t(5,1,1)+2=6$ more tests are needed by Lemma 5.

$\mathbf{t(7,3,1+)=7}$. Follows from the 35 candidate 3-sets by Lemma 1.

$\mathbf{t(8,2,2+)=7}$. First we show that 7 tests are needed even adaptively. If we begin with a 2-pool and it is negative, then $t(6,2,6)=6$ enforces 6 further tests. If we begin with a 3-pool and it is positive, the $18 > 2^4$ candidate 2-sets and Lemma 1 enforce 6 more tests. Due to Lemma 2 we need not consider more cases. **(S)** To manage with 7 tests in 2 stages, we test only one 2-pool in stage 1. If negative, we test the other 6 elements individually in stage 2. If positive, we test the 2 elements in this pool individually, and simultaneously we find up to 1 defective among the other 6 elements using $t(6,1,1)=4$ tests. Note that this strategy also reports if $|P|>2$.

$\mathbf{t(8,2,1)=8}$ is implicit in [15].

$\mathbf{t(8,3,1+)=8}$. Consider the first test of an adaptive strategy. A negative 1-pool enforces 7 more tests since $t(7,3,7)=7$. A positive 2-pool leaves us with $36 > 2^5$ candidate 3-sets. Apply Lemma 1 and finally Lemma 2.

$\mathbf{t(9,2,2+)=7}$. Clearly we only have to show the upper bound for $s=2$. **(S)** Let the pool graph in stage 1 be K_4 (6 edges) plus 3 loops at p_0. It is impossible that exactly 1 pool responds positive. If all pools are negative, then so are the edges in the K_4. If exactly 2 pools are positive, then exactly the edge between them is positive. In the above cases there remain only 3 candidates (loops) in stage 2. If 3 or 4 pools are positive, then we get exactly 3 candidate 2-sets, using edges of the K_4 vertices only. Thus Lemma 6 applies.

$\mathbf{t(9,2,1)=9}$ is known from [15].

$\mathbf{t(9,3,1+)=9}$. We systematically check the tree of all adaptive strategies and give test answers $+$ or $-$ to the searcher's disadvantage. For certain paths in the search tree we find that the searcher is forced to apply too many tests, using earlier bounds and Lemma 1. By Lemma 2 and exploring symmetric cases we can prune most of the tree. Details are omitted due to lack of space, however we remark that our proof has to check just 11 paths, compared to the host of possible strategies and answers.

$\mathbf{t(10,2,3+)=7}$. We have $t(10,2,3) \geq t(8,2,3)=7$. **(S)** For the upper bound we use the following pool graph in stage 1. Take a K_4, but delete the edge p_1p_4 and insert a loop at p_1 and p_4 instead. These 7 elements are complemented with 3 loops at p_0. (Here we particularly emphasize that this pooling design is far from being obvious, we found it after excluding other options with the help of our lower-bound methods. The same remark applies to other cases as well.) As can be quickly checked one by one, all conceivable test outcomes yield one of the following cases (possibly with further dummy elements): at most 3 candidate sets; or 1 recognized defective and at most 4 candidates for a 2nd one; or the candidate graph $K_{1,3}+e$. Using $t(4,1,2)=3$, Lemma 6, and Lemma 7 for the next 2 stages, we can solve all cases in 2 more stages with 3 more tests.

$\mathbf{t(10,2,2)=8}$. **(S)** For $t(10,2,2) \leq 8$ use K_5 as the pool graph in stage 1. It is easy to check that 3 more tests are always enough in stage 2.

Now assume that $t(10,2,2) \leq 7$. In the same way as for $t(7,2,2)$ we can show, due to $t(9,1,2)=5$, $t(7,2,2)=7$, and Lemma 1, that the pool hypergraph in stage 1 is a graph without parallel edges, now with minimum degree 4. The latter implies that at most 5 pool vertices exist. Since a C_3 with loop implies 4 more tests, no further pool vertices can exist. By minimum degree 4, each vertex has at least 2 loops. If all 3 pools are positive, the 9 candidate 2-sets yield 5 more tests by Lemma 1. A C_3 without loop would mean that any vertex of the C_3 has also 2 neighbors outside, leading to 5 pools. But the C_3 requires already 3 more tests. Hence no C_3 can exist. Lemma 5 gives that 2 vertices with 2 loops each require 4 more tests. Thus, in a C_4 at least 3 vertices must be incident to further edges. To avoid C_3, at least 6 pool vertices are needed. Hence no C_4 can exist either. A C_5 cannot exist, since further edges create smaller cycles, and 2 loops per vertex are too many. Hence the pool graph is a forest, with at least 3 loops at every leaf or isolated vertex. Since 2 vertices with 3 and 2 loops imply 5 tests (Lemma 5), at most 2 pool vertices exist. If p_1 and p_2 exist, we choose 2 loops at p_1 and 4 loops at p_2 (or possibly the edge p_1p_2 instead of 1 loop) to get a candidate graph $K_{2,4}$ that needs $t(4,1,1)+2=6$ more tests. If only p_1 exists, we choose 2 loops at p_1 and the 8 other loops from p_1 or p_0 to get a candidate graph $K_{2,8}$ that needs $t(8,1,1)+2=7$ more tests.

$\mathbf{t(10,2,1)=9}$ follows from [15].

$\mathbf{t(10,3,3+)=9}$. The lower bound holds since $t(9,3,9)=9$. **(S)** Our strategy tests 3 disjoint 2-pools in stage 1. If at most 1 pool is positive, the at most 6 candidate elements are tested individually. If 2 pools are positive, 2 tests recognize 2 defectives in stage 2 (as in the $t(7,2,3+)$ strategy). To find a possible 3rd defective among the other 6 elements in stage 3 we use $t(6,1,1)=4$. If all 3 pools are positive, we determine the 3 defectives by 3 tests in stage 2, and 1 final test is used to discard the negative elements or report $|P|>3$.

$\mathbf{t(10,4,1+)=10}$. Consider the first test of an adaptive strategy. A negative 1-pool enforces 9 more tests since $t(9,4,9)=9$. In the case of a positive 2-pool, even revealing a defective means that 3 defectives out of 9 elements must be found, but $t(9,3,9)=9$. By Lemma 2, this case distinction is complete.

6 The Case of Two Defectives and Two Stages

Our $t(9,2,2)$ strategy readily extends to larger n as follows. Let m be the smallest integer with $\binom{m}{2}+3\geq n$. Using K_m plus 3 loops at p_0 (or any subset of this edge set) as the pool graph in stage 1, the same reasoning as for $t(9,2,2)$ yields $t(n,2,2)\leq m+3$. Although this test number grows as $\Theta(\sqrt{n})$, it is optimal (or close to optimal) for surprisingly many n. Below we report some exact results and their lower-bound arguments; note that Lemma 3 is silently used.

$\mathbf{t(8,2,2)=t(13,2,2)=8}$ was already shown.

$\mathbf{t(15,2,2)=t(18,2,2)=9}$. Assume that $t(15,2,2)\leq 8$, and consider the pools in stage 1. A positive ≥ 7-pool allows $70>2^6$ candidate 2-sets, hence by Lemma 1 the remaining 7 tests would not be sufficient. A negative ≤ 5-pool together with $t(10,2,2)=8$ yields 9 tests. Hence only 6-pools can be used. But 2 intersecting positive 6-pools allow at least $25 + 14 = 39 > 2^5$ candidate 2-sets, and 2 disjoint positive 6-pools allow $36 > 2^5$ candidate 2-sets, such that the remaining 6 tests are not sufficient. Hence only one 6-pool may be used. If it responds positive, there remain $69 > 2^6$ candidate 2-sets, implying 8 more tests.

$\mathbf{t(22, 2, 2) = t(24, 2, 2) = 10}$. Assume that $t(22, 2, 2) \leq 9$, and consider the pools in stage 1. A positive $\geq$ 8-pool allows $28 + 8 \cdot 14 = 140 > 2^7$ candidate 2-sets, leading to 10 tests in total. A negative $\leq$ 7-pool together with $t(15, 2, 2) = 9$ yields 10 tests, too. Hence no pool size is usable.

Remarkably, the K_m plus 3 loops strategy misses the simple antichain lower bound of Lemma 1 by at most 1 test up to $n = 31$, and by at most 2 tests up to $n = 58$. However, clearly for large enough n some $O(\log n)$ tests strategy takes over, and it is interesting to ask what constant factor we can achieve.

Theorem 3. *We have $t(n, 2, 2) \leq 2.5 \log_2 n + o(\log_2 n)$, and the trivial lower bound $t(n, 2, 2) \geq 2 \log_2 n - 1$.*

Proof. We encode the n elements as vectors over an alphabet of 4 symbols. The code length is $m = \log_4 n = 0.5 \log_2 n$. In stage 1 we test $4 \cdot 0.5 \log_2 n = 2 \log_2 n$ pools, each consisting of all elements that share a fixed symbol at a fixed position. At most 2 of the 4 pools for every position can be positive, otherwise $|P| > d$ is confirmed. Thus we have at most 2^m candidate elements, and by construction they form disjoint candidate 2-sets. (In the case $|P| = 1$ the only defective is already recognized.) We have $2^m = 2^{0.5 \log_2 n} = \sqrt{n}$. Thus, searching for the 2 defectives is equivalent to searching for 1 defective in a $\leq \sqrt{n}/2$-set. This requires $t(\lceil \sqrt{n}/2 \rceil, 1, 1) = 0.5 \log_2 n + o(\log_2 n)$ tests in stage 2. □

7 Conclusions

We provided methods that make the construction of provably optimal multistage group testing strategies for specific input parameters manageable. The ultimate goal for further research would be a smooth transition from optimal strategies for small n to asymptotically optimal ones. The tools may be further refined to limit the case inspections even more. It would also be helpful to partly automatize the search and leave case inspections to computer programs. However this is not straightforward. To avoid combinatorial explosion we must generate certain set families up to symmetries. Among the open theoretical questions we mention the most intriguing ones: Is the nonadaptive test number additive for the product of candidate hypergraphs (see Lemma 5)? Does there exist, for every d, some s such that $t(n, d, s) = t(n, d, n)$? In [4] we gave an affirmative answer only for $d = 1$, namely $s = 2$.

Acknowledgments. Support came from the Swedish Research Council (Vetenskapsrådet), grant 2010-4661, “Generalized and fast search strategies for parameterized problems”. An early version has been presented at the informal workshop “Search Methodologies III” 2012, organized by Christian Deppe and Ferdinando Cicalese at the Center for Interdisciplinary Research, University of Bielefeld.

References

1. Balding, D.J., Torney, D.C.: Optimal Pooling Designs with Error Detection. J. Comb. Theory A 74, 131–140 (1996)
2. Chen, H.B., Hwang, F.K.: Exploring the Missing Link Among d-Separable, $\bar{d}$-Separable and d-Disjunct Matrices. Discr. Appl. Math. 155, 662–664 (2007)
3. Cheng, Y., Du, D.Z.: New Constructions of One- and Two-Stage Pooling Designs. J. Comp. Biol. 15, 195–205 (2008)
4. Damaschke, P.: Optimal Randomized Group Testing: A Canonical Form and the One-Defective Case. In: Cicalese, F., Porat, E. (eds.) ICALP2011GT (informal proceedings), 55–67, Zürich (2011)
5. De Bonis, A., Di Crescenco, G.: Combinatorial Group Testing for Corruption Localizing Hashing. In: Fu, B., Du, D.Z. (eds.) COCOON 2011. LNCS vol. 6842, pp. 579–591. Springer, Heidelberg (2011)
6. De Bonis, A., Gasieniec, L., Vaccaro, U.: Optimal Two-Stage Algorithms for Group Testing Problems. SIAM J. Comp. 34, 1253–1270 (2005)
7. Du, D.Z., Hwang, F.K.: Combinatorial Group Testing and Its Applications. Series on Appl. Math. vol. 3. World Scientific (2000)
8. Du, D.Z., Hwang, F.K.: Pooling Designs and Nonadaptive Group Testing. Series on Appl. Math. vol. 18. World Scientific (2006)
9. Dyachkov, A.G., Rykov, V.V.: Bounds on the Length of Disjunctive Codes. Problems of Info. Transmission (in Russian) 18, 7–13 (1982)
10. Eppstein, D., Goodrich, M.T., Hirschberg, D.S.: Improved Combinatorial Group Testing Algorithms for Real-World Problem Sizes. SIAM J. Comp. 36, 1360–1375 (2007)
11. Fang, J., Jiang, Z.L., Yiu, S.M., Hui, L.C.K.: An Efficient Scheme for Hard Disk Integrity Check in Digital Forensics by Hashing with Combinatorial Group Testing. Int. J. Digital Content Technol. and its Appl. 5, 300–308 (2011)
12. Fischer, P., Klasner, N,, Wegener, I.: On the Cut-off Point for Combinatorial Group Testing. Discr. Appl. Math. 91, 83–92 (1999)
13. Gao, H., Hwang, F.K., Thai, M.T., Wu, W., Znati, T.: Construction of d(H)-Disjunct Matrix for Group Testing in Hypergraphs. J. Comb. Opt. 12, 297–301 (2006)
14. Goodrich, M.T., Hirschberg, D.S.: Improved Adaptive Group Testing Algorithms with Applications to Multiple Access Channels and Dead Sensor Diagnosis. J. Comb. Optim. 15, 95–121 (2008)
15. Huang, S.H., Hwang, F.K.: When is Individual Testing Optimal for Nonadaptive Group Testing? SIAM J. Discr. Math. 14, 540–548 (2001)
16. Mézard, M., Toninelli, C.: Group Testing With Random Pools: Optimal Two-Stage Algorithms, IEEE Trans. Info. Th. 57, 1736–1745 (2011)
17. Spencer, J.: Minimal Completely Separating Systems. J. Combin. Theory 8, 446–447 (1970)
18. `www.redcrossblood.org/learn-about-blood/what-happens-donated-blood/blood-testing` (version as of Jan. 2013)
19. Xuan, Y., Shin, I., Thai, M.T., Znati, T.: Detecting Application Denial-of-Service Attacks: A Group-Testing-Based Approach. IEEE Trans. Par. Distr.. Syst. 21, 1203–1216 (2010)
20. Zhang, B.: Group Testing Regression Models. Dissertation, Dept. of Statistics, Univ. of Nebraska, Lincoln (2012)

Appendix

$\mathbf{t(7,2,2)=7}$. Omitted details:

A C_3 with loop implies 4 more tests (by the example after Lemma 4). Thus a C_3 implies that only these 3 pool vertices exist, and the other elements are 4 loops at p_0. The latter imply 4 more tests (see Lemma 1). Hence no C_3 can be in the pool graph. Similarly, a C_4 implies 2 more tests, hence only these 4 pool vertices exist. Since a 5th edge would create a C_3, the remaining 3 elements are loops. Loops at distance 1 or 2 imply 3 more tests. Hence the 3 loops are (at most) at one pool vertex p_1 and at p_0. Let p_2 be some neighbor of p_1 in the C_4. If p_1 and p_2 are positive, the edge p_1p_2 with any of the 3 loops is a candidate 2-set, thus Lemma 1 yields 3 more tests. This excludes C_4. Also a C_5 prohibits both further pool vertices and further edges. The remaining 2 loops cannot be at distance 1 or 2, hence they are at one pool vertex p_1 or at p_0. Using a neighbor p_2 similarly as above, Lemma 1 now yields 2 more tests. Altogether, the pool graph has no cycles.

Therefore the pool graph is a forest, perhaps with loops, and all leaves must have loops due to the minimum degree 2. If some leaves have distance 4 (path of 5 pools with 6 elements 2 of which are loops at the leaves), we cannot place the remaining loop without creating a subgraph that incurs at least 2 more tests. Leaves at distance 2 imply 3 more tests, thus no further pool vertices may exist. Now every conceivable placement of the loops implies 4 more tests. If some leaves have distance 3, therefore, only these 3 edges exist, and 4 loops. We cannot add a 5th, isolated, pool vertex, since it must have 2 loops implying 2 more tests. But with only 4 pool vertices, every conceivable placement of the 4 loops implies 3 more tests. Leaves at distance 1 require 3 more tests, thus at most 1 further, isolated, pool vertex may be present. Again this isolated vertex has at least 2 loops, hence the only edge has exactly 1 loop at each end (to avoid 2 vertices with 2 loops each). But now 4 loops are together at the isolated vertex and p_0. This allows at least 5 candidate 2-sets, and Lemma 1 yields 4 more tests. Thus a 3rd pool vertex is ruled out. The remaining case is 1 edge and a total of 6 loops at both ends and possibly p_0. The candidate graph contains $K_{1,6}+e$. Lemma 4 implies $t(6,1,1)+1=5$ more tests.

$\mathbf{t(9,3,1+)=9}$. Omitted details:
$\{v_1\}(-)$: $t(8,3,8)=8$.
$\{v_1,v_2\}(+)$, $\{v_1\}(+)$: $t(8,2,8)=7$.
$\{v_1,v_2\}(+)$, $\{v_3,v_4\}(-)$, $\{v_1\}(+)$: The searcher must find 2 defectives among $v_2,v_5,\dots,v_9$, but $t(6,2,6)=6$.
$\{v_1,v_2\}(+)$, $\{v_3,v_4\}(-)$, $\{v_5\}(-)$, $\{v_1\}(+)$: The searcher must find 2 defectives among $v_2,v_6,\dots,v_9$, but $t(5,2,5)=5$. Hence it suffices to consider a 4th pool avoiding v_1 and v_2.
$\{v_1,v_2\}(+)$, $\{v_3,v_4\}(-)$, $\{v_5\}(-)$, $\{v_6\}(-)$: 9 candidate 3-sets are left.
$\{v_1,v_2\}(+)$, $\{v_3,v_4\}(-)$, $\{v_5\}(-)$, $\{v_6,v_7\}(+)$: 12 candidate 3-sets are left.
$\{v_1,v_2\}(+)$, $\{v_3,v_4\}(-)$, $\{v_5,v_6\}(+)$, $\{v_5\}(+)$: There remain 9 candidate 3-sets. This also rules out any 4th pool with some of v_1,v_2,v_5,v_6.

$\{v_1, v_2\}(+)$, $\{v_3, v_4\}(-)$, $\{v_5, v_6\}(+)$, $\{v_7, v_8\}(-)$, $\{v_6\}(+)$: The $5 > 2^2$ candidate 2-sets for the other 2 defectives enforce 4 more tests. This also rules out any 5th pool with some of v_1, v_2, v_5, v_6. It remains to query v_9.
$\{v_1, v_2\}(+)$, $\{v_3, v_4\}(-)$, $\{v_5, v_6\}(+)$, $\{v_7, v_8\}(-)$, $\{v_9\}(-)$: Here the counting bound enforces 4 more tests.
$\{v_1, v_2\}(+)$, $\{v_3, v_4\}(-)$, $\{v_5, v_6\}(+)$, $\{v_7, v_8, v_9\}(+)$: 12 candidate 3-sets are left.
$\{v_1, v_2\}(+)$, $\{v_3, v_4, v_5\}(+)$: The $33 > 2^5$ candidate 3-sets enforce 7 more tests.