Strict Group Testing and the Set Basis Problem[*]

Peter Damaschke[†], Azam Sheikh Muhammad
Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
[ptr,azams]@chalmers.se
and
Gábor Wiener
Department of Computer Science and Information Theory
Budapest University of Technology and Economics
Budapest 1521, Hungary
wiener@cs.bme.hu

Abstract

Group testing is the problem to identify up to d defectives out 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 at most d defectives are present. We start building a combinatorial theory of strict group testing. We compute many exact $t(n, d, s)$ values, thereby extending known results for $s = 1$ to multistage strategies. These are interesting since asymptotically nearly optimal group testing is possible already in $s = 2$ stages. Besides other combinatorial tools we generalize d-disjunct matrices to any candidate hypergraphs, and we reveal connections to the set basis problem and communication complexity. As a proof of concept we apply our tools to determine almost all test numbers for $n \leq 10$ and some further $t(n, 2, 2)$ values. We also show $t(n, 2, 2) \leq 2.44 \log_2 n + o(\log_2 n)$.

Keywords: group testing, hypergraph, set basis, graph coloring, d-disjunct matrix

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 all elements in Q are recognized as negative. The goal of a *searcher* is to identify P using a minimum number of tests. A group testing strategy may be organized in s *stages*, where the tests applied in a stage may depend on the outcomes of all tests in previous stages, and all tests within a stage are executed in parallel. In *adaptive* strategies 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. The term *pooling design* refers to a set of pools, especially within one stage. A pooling design can be

[*] This is an extended version of a conference paper of the first two authors that appeared in the Proceedings of the *19th International Computing and Combinatorics Conference COCOON 2013*, Hangzhou (China), *Lecture Notes in Computer Science*, Springer, vol. 7936, pp. 446–457.

[†] Corresponding author. Tel. 0046-31-772-5405. Fax 0046-31-772-3663.

written as a binary *matrix* whose rows and columns are the pools and elements, respectively. A matrix entry is 1 if the element is in the pool, and 0 else.

We consider the following scenario. The searcher expects $|P| \leq d$ for some previously known bound d, and $|P| > d$ is unlikely but not impossible. She 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". It was argued in, e.g., [1] that strict group testing is preferable. It does not rely on the assumption $|P| \leq d$, and the searcher is sure about not having missed any defective.

For complexity results and various applications of group testing we refer to [8, 9, 3, 24, 5, 20, 29, 33] as entry points to further studies. For the test number in $s = 1$ stage, a lower bound $\frac{d^2}{2\log_2[e(d+1)/2]}\log_2 n + o(\log_2 n)$ was given in [13] and later refined in [16, 17]. As opposed to that, $O(d \log n)$ tests are sufficient already if $s = 2$, as shown by the random coding upper bound in [14], followed by several improved constructions [15, 10, 6, 18, 17]. However, even asymptotically optimal strategies do not necessarily entail optimal strategies for specific input sizes n. Furthermore, pool sizes increase with n, whereas in some applications large pools may be infeasible. Still we can split an instance into many small instances and solve them independently, each with optimal efficiency. To mention a practical example, screening millions of blood donations for infectious diseases is performed at some labs in instances ("minipools") of, e.g., 16 samples [30], and group testing is proposed [35] to reduce the waiting times and costs. We also refer to [12] for biological applications of 2-stage strategies, with tests in the last stage being individual (whereas we will drop this restriction).

We define $t(n, d, s)$ to be the optimal worst-case number of tests needed for strict group testing for n elements, up to d defectives, and at most 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.

To our best knowledge, the strict group testing model and the construction of optimal strategies for specific problem sizes in the multistage case are under-researched so far. Our ambition is to initiate a combinatorial theory of strict group testing that (a) connects the problem to other fields and (b) enables us to calculate as many $t(n, d, s)$ values as possible, along with the corresponding optimal strategies.

We remark that $t(n, d, 1)$ is the size of a d-disjunct matrix, also known as superimposed code. The first bounds were given in [26]. Exact $t(n, d, 1)$ values follow from [25] for all $n \leq 14$ and some larger n. As we observed in [4], $t(n, 1, 1) = \log_2 n + o(\log_2 n)$ is the smallest k with $\binom{k}{k/2} \geq n$. Namely, by [31] this k is the optimal size of a completely separating set family (see Section 3), and these are exactly the pooling designs for nonadaptive strict group testing when $d = 1$. Although this connection is simple, this seems to be the first case of a combinatorial search problem where completely separating systems really appear. Next, $t(n, 1, 2+) = \lceil \log_2 n \rceil + 1$ is a side result in [4], and [21] gave partial results on adaptive strategies. The $t(n, 2, 2)$ case is particularly interesting. Note that $t(n, 2, 2) \geq 2\log_2 n - 1$ holds trivially, and a result from [28] implied $t(n, 2, 2) \leq 3\log_2 n$. The currently best known upper bounds for $t(n, d, 2)$ for general d can be found in [17]. Here we will improve on $t(n, 2, 2)$. Other work addresses optimal test numbers in other group testing variants [34].

Here we give an overview of the paper structure and the novel contributions.

- We introduce some useful hypergraphs related to strict group testing. In particular, the candidate hypergraph lists the candidate sets for the defective set P (Section 2).

- We show that the last stage of a problem instance, which is nonadaptive strict group testing on a candidate hypergraph, is equivalent to the set basis problem, one of the "oldest" NP-hard optimization problems on hypergraphs, and to the 0-cover number in Communication Complexity (Section 3).

- Furthermore, this is equivalent to optimal coloring of a so-called conflict graph derived from the candidate hypergraph. The set basis formulation enables some elegant proofs of other facts, whereas conflict graph coloring is often more suitable calculating specific test numbers. Our characterizations also naturally generalize d-disjunct matrices to the case of arbitrary hypergraphs (Section 3).

- We study compositions of candidate hypergraphs that are relevant in our calculations. The product formalizes independent problem instances on disjoint sets. We disprove the natural conjecture that the test number is additive, but we prove it in a useful special case and state a slightly weaker conjecture on additivity (Section 4).

- We collect a number of further tools for proving bounds on $t(n, d, s)$ (Section 5 and 6). Using all these preparations we finally manage to give optimal strategies for almost all triples n, d, s with $n \leq 10$ (Section 7). Some of the designs are surprising and would be hard to find without our techniques. In particular, the last stage is often not just individual testing of candidates. (We also stress that matching lower bounds cannot be achieved naively by considering all possible pooling designs in every stage, as their number is doubly exponential in n.) The results extend similar work for $s = 1$ to $s > 1$ stages and thus save tests.

- For $t(n, 2, 2)$ we obtain optimal results also for some larger n. We also show $t(n, 2, 2) \leq 2.44 \log_2 n + o(\log_2 n)$, essentially helped by the set basis characterization (Section 8).

2 Set and Hypergraph 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 in special roles. A *hypergraph* is a set of *vertices* equipped with a family of subsets called *hyperedges*. A *graph* is a hypergraph with only ≤ 2-hyperedges, called *edges*. A *loop* is a 1-hyperedge. We allow *parallel* hyperedges which are identical as sets but occur multiple times. Set A is a subset (superset) of B if $A \subseteq B$ ($A \supseteq B$). Two sets are *incomparable* if neither is a subset of the other one. A family of pairwise incomparable sets is an *antichain* (Sperner family). Standard graph-theoretic symbols K_n, C_n, $K_{m,n}$ denote 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.

Consider any moment after some stages of a group testing strategy. An element that has not yet appeared in negative pools is called a *candidate element*. A *candidate set* is any set of up to d candidate elements that is consistent with all previous test outcomes (and thus, possibly the true set P). The name is justified by a simple observation: An element v is possibly positive until it is discarded, as a member of some negative pool, and before that, the searcher cannot safely conclude that v is negative. Therefore we can even 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.

The *candidate hypergraph* has the candidate elements as vertices and the candidate sets as hyperedges. For $d = 2$ we speak of the *candidate graph*, possibly with loops. The candidate hypergraph completely describes the searcher's instantaneous knowledge. The definitions imply that the possible sets P are exactly the candidate sets and all their supersets. (But note that the candidate sets do not necessarily form an antichain.) An instance of the strict group testing problem is solved if and only if the candidate hypergraph retains exactly one hyperedge and no dummy elements. We also remark that the candidate sets are exactly those $\leq d$-sets of candidate elements which are hitting sets of the family of positive pools so far.

As usual in the field, some proofs use *adversary* arguments, where an adversary answers to the tests in order to enforce a certain worst-case test number.

3 Characterizations of Nonadaptive Strict Pooling Designs

The candidate hypergraph captures the searcher's knowledge after any number of stages, and the question is how the searcher should efficiently finish the search in the remaining stages. This gives rise to the following generalization of the problem, that we call *strict group testing on a candidate hypergraph.* (This should not be confused with the different concept in [23].) A candidate hypergraph is given, and the searcher knows already that the true set P of positive elements is one of the candidate sets or their supersets. If P is among the candidate sets, the searcher must both identify P and verify that all elements outside P are negative. If P is not among the candidate sets, the searcher is only supposed to recognize this fact.

In the following we consider, in particular, the situation prior to the last (s-th) stage. We call a pooling design *strict* if it solves the strict group testing problem on the given candidate hypergraph nonadaptively. Our first characterization of strict designs relates to the algebraic notion of *set basis.* Let us be given a hypergraph whose hyperedges we call *target sets.* A set basis is another hypergraph on the same vertices – we call its hyperedges *basis sets* – such that every target set equals the union of some basis sets.

Theorem 1. *A pooling design is strict if and only if the complement of every candidate set is the union of some pools.*

Proof. Suppose that the pools form a set basis for the complements of candidate sets. Moreover, suppose that P is some candidate set. Then the complement of P is the union of negative pools, hence all candidate elements outside P are discarded. Still P may contain other candidate sets as subsets, let $P' \subset P$ be any of them. The complement of P' is a union of some pools as well. At least one such pool Q intersects P, hence Q is positive. Since also $P' \cap Q = \emptyset$, this test outcome rules out P' as a candidate set. Since this reasoning holds for every $P' \subset P$, it follows that only the candidate set P remains, and no dummy elements, as seen above. If P is not a candidate set, the searcher recognizes this fact as follows. P is a superset of some candidate set P'. Since only candidate elements outside P appear in negative pools, we retain some candidate set together with further elements, unlike the case above. Altogether, the pooling design is strict.

For the converse, assume that the complement of some candidate set P is not a union of pools. Then the negative pools do not exhaust the complement of P, thus, again some candidate set together with further candidate elements remains. Now this means that the pooling design is not strict. □

The proof also shows that decoding is trivial: Discard all elements in negative pools, and what remains is P. Finally check that P is a candidate set. By Theorem 1, solving nonadaptive strict group testing on candidate hypergraphs with a minimum number of pools is equivalent to finding a set basis of minimum size (where the target sets are the complements of the candidate sets). The latter problem NP-hard [32], but, of course, we can still solve it for special hypergraphs of interest.

There is always a trivial set basis where the pools are the complements of all candidate sets, thus we have:

Corollary 2. *Any candidate hypergraph of m candidate sets permits a nonadaptive strict group testing strategy with at most m tests.*

Another consequence is that dummy elements can be ignored for the test number:

Corollary 3. *Let G be a candidate hypergraph obtained from another candidate hypergraph G' by removing all dummy elements. Let t and t' be the optimal number of pools for nonadaptive strict group testing on G and G'. Then $t = t'$.*

Proof. Take an optimal pooling design for G and add the dummy elements to each of the t pools. Since the pooling design for G is a set basis according to Theorem 1, and the dummy elements in G' are in the complements of all candidate sets, this yields a set basis for G'. This shows $t' \leq t$, and the other direction $t \leq t'$ is trivial. □

We add a similar statement for s stages, as we will need it in this form, too.

Lemma 4. *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 the searcher recognizes $|P| > d$. This negative pool also discards the dummy elements. □

Theorem 1 can be rephrased as follows.

Corollary 5. *A pooling design is strict if and only if, for every pair of a candidate set C and a candidate element $v \notin C$, some pool Q exists such that $v \in Q$ and $C \cap Q = \emptyset$.*

Remember that initially, before stage 1, the candidate sets are all $\leq d$-sets. In this special case, Corollary 5 just expresses the well-known condition for d-disjunct pooling designs, hence $t(n, d, 1)$ is the optimal number of rows of a d-disjunct matrix with n columns. For $d = 1$, the condition in Corollary 5 is the definition of completely separating set families: for any two elements u and v, some set Q in the family satisfies $v \in Q$ and $u \notin Q$. Next we give another reformulation which is often more suitable for calculating optimal pool numbers.

In a *partial vector*, 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 encode every pair of a candidate set C and a candidate element $v \notin C$ by 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 they have 0 and 1 at the same position. Partial vectors without conflict are *compatible*. We translate the candidate hypergraph into a *conflict graph* defined as follows. Vertices represent the partial vectors for all C and $v \notin C$, and two vertices are adjacent if the corresponding partial vectors conflict. Next, any pool is represented as its indicator vector: the bit vector with 1 at all positions of elements in the pool, and 0 elsewhere. A pool *covers* a partial vector if every fixed value 1 or 0 in the partial vector equals the value in the pool's indicator vector. The number of colors needed to color a graph G, such that adjacent vertices get distinct colors, is known as its chromatic number $\chi(G)$. We refer to $\chi(G)$ of a conflict graph G as *conflict chromatic number*.

Theorem 6. *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 pools required.*

Proof. The condition in Corollary 5 can also be expressed by saying that, for every pair of a candidate set C and candidate element $v \notin C$, some pool must cover its partial vector. A set of partial vectors can be covered by a single pool, if and only if they are pairwise compatible. In the conflict graph, compatible partial vectors correspond to nonadjacent vertices. Thus, a set 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 graph coloring, the assertion follows. □

Once more, NP-hardness of graph coloring does not stop us from computing $\chi(G)$ for specific conflict graphs G.

Theorem 1 opens even more avenues. First, the set basis problem is equivalent to the minimum biclique edge cover problem in bipartite graphs. A biclique edge cover is a family of complete bipartite subgraphs (bicliques) such that every edge belongs to some of them. The equivalence is seen as follows. Think of the target sets T and elements v as vertices of a bipartite graph, with an edge between T and v if and only if $v \in T$. A minimum set basis can always be made of intersections of the target sets only, and intersections obviously correspond to bicliques. For every v and T, element v must appear in some basis set contained in T, which is exactly the edge cover condition.

Next we forge a bridge to Communication Complexity. Recall that the partial vectors forming the vertex set of the conflict graph are encodings of the pairs of candidate elements v and candidate sets C with $v \notin C$. Now we identify the vertices with these pairs. Consider the incidence matrix M of any candidate hypergraph (that is, rows correspond to the candidate elements, columns to the candidate sets). The vertices of the conflict graph correspond to the 0 entries of M. There is an edge between two 0's if and only if the corresponding vertices have a conflict, that is, the candidate set of one of them contains the candidate element of the other. Let $x = (v, V)$ and $y = (w, W)$ be two vertices of the conflict graph. An edge between x and y exists if and only if the (v, W) entry or the (w, V) entry of M is 1. Hence some vertices of the conflict graph form an independent set if and only if the corresponding 0 entries of M can be covered with a submatrix containing 0's only, also called 0-monochromatic submatrix, and now the conflict chromatic number equals the minimum number of 0-monochromatic submatrices covering all 0 entries of M. This is a well-known measure in Communication Complexity, called the 0-cover number. Its logarithm is the *co-nondeterministic communication complexity* of M (actually, of the corresponding function). Moreover, a clique in the conflict graph is a so-called *fooling set* of M.

The 0-cover number of M equals the minimum number of bicliques covering the edges of the bipartite graph whose adjacency matrix is $E - M$, where E is the all-1 matrix of the same size as M. Due to these relationships, results in Communication Complexity or biclique edge covering may be used to obtain results in strict group testing. It is also possible the other way round; just as an illustration we give a two-sentence proof of the following theorem from [7] that was rediscovered in [22, 2]. Another short proof can be found in [36]. It is well known that the 0-cover number of the $n \times n$ identity matrix I_n is at most $2\lceil \log_2 n \rceil$ (see, e.g., [27]). It is much less known that the exact value is determined precisely.

Theorem 7. *(de Caen et al. [7]) The 0-cover number of I_n is the smallest number k, such that $\binom{k}{k/2} \geq n$.*

Proof. Consider the strict group testing problem on n elements, and $s = 1$, $d = 1$. The candidate hypergraph contains the 1-element sets and the empty set, thus its incidence matrix is I_n plus an all zero column, whose 0-cover number is obviously the same as of I_n, which is, by the above observations, the same as $t(n, 1, 1) = \min(k : \binom{k}{k/2} \geq n)$. □

4 Independent Instances of Nonadaptive Strict Group Testing

A frequent situation is that a candidate hypergraph is composed of two (or more) smaller hypergraphs on disjoint subsets of elements, and the overall candidate sets are all combinations of candidate sets in these "independent" parts. This happens, e.g., when two subsets of earlier positive pools (within disjoint sets) require hitting sets of at least d_1 and d_2 positive elements, respectively, where $d_1 + d_2 = d$.

We formalize this case by a certain product of hypergraphs. Let G_1 and G_2 be any two candidate hypergraphs on disjoint sets of vertices (elements), denoted V_1 and V_2. We define their *product* $G = G_1 \times G_2$ as the candidate hypergraph on the vertex set $V_1 \cup V_2$, whose candidate sets are the unions $C_1 \cup C_2$, for all pairs of candidate sets $C_1 \subseteq V_1$ and $C_2 \subseteq V_2$, from G_1 and G_2, respectively. Let t_i be the optimal number of pools for nonadaptive strict group testing on G_i $(i = 1, 2)$, and t the optimal number for G. We have $t \leq t_1 + t_2$, since the pools from optimal pooling designs for both G_1 and G_2 together form a set basis of G.

One might conjecture additivity: $t = t_1 + t_2$. However, the searcher could use pools with elements from both V_1 and V_2 and save some tests. In fact, additivity does not always hold true. Apparently the smallest counterexample is the candidate graph $K_{3,4}$. Here V_1 and V_2 are the partite sets of the bipartite graph, with 3 and 4 elements, respectively, and exactly 1 element on each side is positive. Then $t_1 = t(3, 1, 1) = 3$ and $t_2 = t(4, 1, 1) = 4$, but for the $K_{3,4}$ we need only $6 = 7 - 1$ pools, as given in the following incidence matrix.

$$\left(\begin{array}{ccc|cccc}
1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0
\end{array}\right)$$

Due to symmetry it suffices to check for one edge that its complement is the union of some pools. (Remarkably, the same design appeared already in [34] in a different context.) This and a few other, similar counterexamples seem to rely on specific numbers. Our current conjecture is that additivity holds with some sporadic exceptions, and $t \geq t_1 + t_2 - 1$ holds in general. Below we prove additivity in special cases that are powerful enough for our purposes.

Theorem 8. *Let G_1 be the candidate hypergraph with hyperedges $\{v_1\}$ and $\{v_2\}$ (that is, exactly one of these 2 elements is defective), and G_2 be an arbitrary candidate hypergraph. Then nonadaptive strict group testing on $G_1 \times G_2$ needs $t_2 + 2$ tests.*

Proof. Due to Corollary 3 we can assume without loss of generality that G_2 has no dummy elements. Let $v_3, v_4, \ldots$ be the elements of G_2. Pools must cover all partial vectors according to Corollary 5 and Theorem 6. First consider those candidate sets of $G_1 \times G_2$ containing v_1. 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, $i > 2$, belong to candidate sets in G_2, thus give rise to 0s, this last pool can only be $\{v_1\}$. Now the key step is that, by symmetry, the pool $\{v_2\}$ must also exist. But 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 a fixed value 1 at some position v_i, $i > 2$. Hence we need $t_2 + 2$ pools in total. □

As one immediate consequence, the candidate graph $K_{2,n}$ needs $t(n, 1, 1) + t(2, 1, 1) = t(n, 1, 1) + 2$ nonadaptive tests. The power of Theorem 8 is also illustrated by the following example. Consider k disjoint pairs of elements, where one element of each pair is defective. Inductive application of Theorem 8 gives the lower bound $2k$ (individual testing is optimal), whereas the information-theoretic lower bound is only k.

We add some small observation that should help attack the additivity conjecture in the future, as it greatly restricts the structure of optimal designs. Consider a pooling design for $G = G_1 \times G_2$. We define a bipartite graph B on a vertex set $X_1 \cup X_2$ as follows. The edges of B represent the pools Q, and the vertices of X_i represent the different intersections $Q \cap V_i$ $(i = 1, 2)$. That is, the vertices for $Q \cap V_1$ and $Q \cap V_2$ are joined by the edge for Q. (Note that the edges of any two pools Q and Q' with $Q \cap V_i = Q \cap V_i$ are incident to the same vertex in X_i. Also, one of the vertices in X_i may represent the empty set.) If all pools are entirely in V_1 or V_2, then B consists of two star graphs, one with center in X_1 and one in X_2, and these center vertices represent the empty set. In particular, this B is a forest. Remarkably, the latter also holds in general:

Proposition 9. *There is always an optimal nonadaptive strict pooling design for $G_1 \times G_2$ whose bipartite graph B is a forest.*

Proof. Assume that B has a cycle. We replace every pool Q corresponding to an edge in this cycle with the pools $Q \cap V_1$ and $Q \cap V_2$. The total number of pools does not increase, and the modified pooling design is still a set basis for the complements of candidate sets in $G_1 \times G_2$. Thus, by Theorem 1 we still have a strict pooling design. For the bipartite graph this means to remove the edges of the cycle and to connect its vertices with the empty-set vertex on the other side of B. By iterating the procedure we destroy all cycles. □

Next we study a second fundamental operation. Let $G_1, \ldots, G_c$ be c candidate hypergraphs on pairwise disjoint vertex sets V_i. We define their *disjoint union* $G_1 + \cdots + G_c$ as the candidate hypergraph on the vertex set $V_1 \cup \cdots \cup V_c$, whose candidate sets are all original candidate sets $C_i \subseteq V_i$ from the G_i. Again, let t_i be the optimal number of pools for nonadaptive strict group testing on G_i.

Theorem 10. *Nonadaptive strict group testing on $G_1 + \cdots + G_c$ can be done with at most $t(c, 1, 1) + \max_{1\le i\le x} t_i$ pools.*

Proof. By Theorem 1 we only need a set basis for the complements of the sets C_i. Our set basis comprises two types of pools. Firstly, since the c sets V_i are pairwise disjoint, we can obviously create a set basis of $t(c, 1, 1)$ pools for their complements. We call them coarse pools. Secondly, we take a set basis (of subsets of V_i) for every G_i and index these pools arbitrarily by $P_{i1}, P_{i2}, P_{i3}, \ldots, P_{i,t_i}$. To this sequence we append empty pools if t_i is not maximum. Then, for every $j \le \max t_i$ we create the pool $P_{1j} \cup \cdots \cup P_{cj}$. We call them fine pools. To see that these pools together form a set basis, consider any fixed candidate set C_i. The complement of V_i is the union of some coarse pools. Furthermore, $V_i \setminus C_i$ is the union of some pools that participate in fine pools. By construction, the "rests" of these fine pools are outside V_i. Hence the complement of C_i is the union of some pools. □

5 Further Tools for Lower Bounds

The simple *counting bound* says that the number of tests is at least $\log_2$ of the 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 11. *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 see that 2 tests are required if $r=0$, that is, $m=2$. If 1 test were enough then we could do it in 1 stage. Thus Theorem 1 applies and, obviously, 2 incomparable sets cannot be unions of pools when only 1 pool is present. 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. By the inductive hypothesis the claim holds for $r+1$. □

The next two lemmas are evident monotonicity observations, therefore the proofs are omitted. Recall that we discuss strict group testing in a prescribed number s of stages.

Lemma 12. *Suppose that the adversary, in response to a given deterministic test strategy, applies 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 13. *Suppose that the adversary reveals the outcomes of some pools of a stage to the searcher, and then allows the searcher to redesign the remaining pools of this stage from scratch. If t further tests are not sufficient despite redesign, then they are not sufficient for the original problem instance either.*

6 Pool Hypergraphs and Some Candidate Hypergraphs

A *pool hypergraph* represents a pooling design, but in dual form, which will turn out to be convenient. Its *pool vertices* $p_1, p_2, p_3, \ldots$ are the pools, and its hyperedges are the candidate elements. A vertex belongs to a hyperedge if the corresponding pool contains the corresponding element. Those elements being in no pools are represented as loops at a symbolic *null vertex* p_0. We may speak of a *pool graph* if all hyperedges are $\leq$ 2-sets.

Recall that our main goal is to get exact values $t(n,d,s)$ for as many triples n,d,s as possible. Since the $t(n,1,1)$, and $t(n,1,2+)$ were completely known before, we study $d\geq 2$ only. We summarize our methodology: In every possible pool (hyper)graph that the searcher may apply in stage 1, an adversary fixes a certain subset W of vertices and renders all pools in W positive and the others negative. The remaining candidate elements are precisely the (hyper)edges in the sub(hyper)graph induced by W, and the defective (hyper)edges must cover W. Then we use our lower-bound techniques to enforce certain test numbers in the resulting candidate (hyper)graphs. Symmetries and the preceding lemmas reduce the amount of case inspections. As a preparation we collect some conclusions from our theorems, as well as special cases that will be used in our $t(n,d,s)$ calculations.

Let $d=2$. The following pool graphs enforce, by Lemma 11, the following minimum numbers of further tests for adaptive strict group testing, provided that all pools respond positively. (Note that candidate sets of the same size form an antichain.)

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

Next we look at some useful special candidate graphs. 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 (which yields the candidate graph C_4 when they respond positively), then at least 4 more tests are required in stage 2.

An observation is that, for $d = 2$, the largest possible clique size in the conflict graph is 5, thus, in general we need stronger arguments than the clique size in order to determine conflict chromatic numbers. We continue with some of them, that we use later.

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 when p_1, p_2 are positive.

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, in a similar way as in Theorem 8:

Lemma 14. *A nonadaptive strict group testing strategy for the candidate graph $K_{1,k} + e$ requires $t(k, 1, 1) + 1$ pools.*

Proof. Assume that $t(k, 1, 1)$ pools 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 the definitions prior to Theorem 6.) Let the first position correspond to the center of the k-star, while the other 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 2nd defective is either of the k leaves. Thus we need already $t(k, 1, 1)$ pools to cover 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. □

Lemma 14 captures some frequent cases, for example: Let $n = 4$, $d = 2$, $s = 2$, let the pool graph in stage 1 be C_3 plus 1 loop at 1 vertex, and let all these pools be positive. Then the resulting candidate graph is $K_{1,3} + e$, hence $t(3, 1, 1) + 1 = 4$ tests are needed in stage 2.

However, the candidate graph $K_{1,3} + e$ can be solved with 3 tests when we allow 2 further 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. No matter whether this pool is positive or negative, there remain 2 candidate 2-sets for stage 2, which requires only 2 more nonadaptive tests by Corollary 2.

7 Optimal Strategies for Small Instances

Now we systematically list the optimal test numbers $t(n, d, s)$ that we found. This section contains only the practical part: we describe the strategies and pooling designs. Their optimality proofs may be tedious to read, therefore we move them to the Appendix. The reader may go through some, to get an idea how the techniques work. Results that trivially follow from the listed ones by monotonicity (see Section 1) are omitted.

$\mathbf{t(3,2,1+)=3}$, $\mathbf{t(4,2,1+)=4}$, $\mathbf{t(5,2,1+)=5}$, $\mathbf{t(6,2,1+)=6}$: by individual testing.

$\mathbf{t(7,2,3+)=6}$. Use 3 mutually disjoint 2-pools in stage 1. If at most 1 of them is positive, at most 3 candidate elements are left, which are tested individually in stage 2. If all 3 of the 2-pools are positive, we conclude $|P|>2$. If exactly 2 are 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, 1 positive element is recognized in both cases. A 6th test in stage 3 on the remaining candidate elements confirms they are negative, or yields $|P|>2$.

$\mathbf{t(7,2,2)=t(7,2,1)=7}$, $\mathbf{t(7,3,1+)=7}$, $\mathbf{t(8,2,1)=t(8,3,1+)=8}$: by individual testing.

$\mathbf{t(9,2,2+)=7}$. 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 positively. 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. Now Corollary 2 applies.

$\mathbf{t(9,2,1)=t(9,3,1+)=9}$: by individual testing.

$\mathbf{t(10,2,3+)=7}$. Here we particularly emphasize that our 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.

We use the following pool graph in stage 1. Take a K_4, but delete one edge, say 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. 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 $t(4,1,2)=3$, Corollary 2, and Lemma 4 for the next 2 stages, we can solve all cases in 2 more stages with 3 more tests.

$\mathbf{t(10,2,2)=8}$. We 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.

$\mathbf{t(10,2,1)=9}$ follows from [25].

$\mathbf{t(10,3,3+)=9}$. 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 the strategy for $t(6,1,1)=4$. 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 to report $|P|>3$.

$\mathbf{t(10,4,1+)=10}$: by individual testing.

8 The Case of Two Defectives and Two Stages

Our $t(9,2,2)=7$ 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 $t(n,2,2)\leq m+3$. (See Table 1.) More formally:

T	8	9	10	11	12	13	14	15	16	17	...	20	...	25
N1	8	12	13	17	20	26	28	42	48	68	...	84	...	293
N2	13	18	24	31	39	48	58	69	91	127	...	254	...	823

Table 1: This table illustrates that more stages save tests, for the example of $d = 2$ defectives. Here, T is a budget of tests, $N1$ is the maximum n that is resolved in 1 stage according to known results [25, 11, 10], and $N2$ is the maximum n that is resolved in 2 stages in the present paper.

Proposition 15. *For $n \le \binom{m}{2} + 3$ we have $t(n, 2, 2) \le m + 3$.*

Although this test number is $\Theta(\sqrt{n})$, it is optimal, or close to optimal, for surprisingly many n. Below we report further exact results for $t(n, 2, 2)$. Once more, the lower-bound arguments are moved to the Appendix.

$\mathbf{t(10, 2, 2) = t(13, 2, 2) = 8}$. Use Proposition 15 with $m = 5$.

$\mathbf{t(15, 2, 2) = t(18, 2, 2) = 9}$. Use Proposition 15 with $m = 6$.

$\mathbf{t(22, 2, 2) = t(24, 2, 2) = 10}$. Use Proposition 15 with $m = 7$.

Remarkably, the K_m plus 3 loops strategy misses the simple antichain lower bound of Lemma 11 by at most 1 test up to $n = 31$, and by at most 2 tests up to $n = 58$. Next, instead of a pool graph being a clique plus some loops at the null vertex, we may similarly use a pool hypergraph being the counterpart of a clique: For some fixed $k > 2$, let the elements be all k-subsets of the pool vertices. The tricky question is how many tests remain for stage 2, and whether such constructions are beneficial for certain ranges of n. In fact, we found that $k = 3$ is a good choice for many n. In detail:

Let m now be the smallest integer with $\binom{m}{3} + 7 \ge n$. In stage 1 we use m pools, let the elements be represented by, up to $\binom{m}{3}$, different 3-sets of pools in the pool hypergraph, and represent up to 7 more elements as loops at the null vertex. We claim that 7 tests are always sufficient in stage 2. Then it follows precisely as above:

Proposition 16. *For $n \le \binom{m}{3} + 7$ we have $t(n, 2, 2) \le m + 7$.*

In fact, this is currently the best upper bound on $t(n, 2, 2)$ that we know for $70 \le n \le 823$. For proving Proposition 16 it remains to prove our claim on 7 tests. If 0 pools in stage 1 are positive, only the 7 loops are candidate elements, and we test them naively. The case of 1 or 2 positive pools is impossible. If we get 3 positive pools, then 1 defective is identified, and again we check the 7 loops individually. In all other cases the candidate 2-sets must be pairs of 3-hyperedges. (It might be interesting to notice that the candidate graph is then a Kneser graph.) At most 6 pools can be positive, otherwise we instantly see $|P| > 2$. If exactly 4 pools are positive, we have just 4 candidate elements. If exactly 6 pools are positive, the candidate 2-sets are 10 disjoint pairs of elements, hence $t(10, 1, 1) = 5 < 7$ further tests suffice. The worst case is when exactly 5 pools are positive. One easily verifies that the candidate graph is then the famous Petersen graph, consisting of 2-disjoint copies of C_5 connected by 5 more edges which form a perfect matching. We do not even need the exact shape of the Petersen graph; the latter property is enough to establish a test number at most 7, due to Theorem 1 and the following:

Lemma 17. *Let G be a graph whose vertex set can be partitioned in U and V, each with exactly k vertices, and with exactly k edges between U and V, which are pairwise not incident. Then the family of the complements of G's edges (viewed as 2-sets of vertices) has a set basis of size at most $k+2$.*

Proof. Our basis sets are simply U, V, and the k edges between them. For brevity we call them cut edges. The complement of any edge e within U is the union of V and of the $k-2$ cut edges that do not touch e. By symmetry, we also serve the edges within V. The complement of any cut edge is the union of the other $k-1$ cut edges. □

This also concludes the proof of Proposition 16. 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. Note that $t(n,2,2) \geq 2\log_2 n - 1$ is a trivial lower bound.

Theorem 18. *Any 2-disjunct matrix of size $z \cdot \log_2 n + o(\log_2 n)$ yields an upper bound $t(n,2,2) \leq (2.5 - \frac{1}{4z-6})\log_2 n + o(\log_2 n)$. In particular we achieve $t(n,2,2) \leq 2.44\log_2 n + o(\log_2 n)$.*

Proof. Let $b > 1$ be some integer, and x, y be positive real numbers with $x + y = 1$. These method parameters will be fixed later. For convenience we omit rounding brackets if in some real expressions only integer values make sense. This does not affect the asymptotic result.

We divide the elements in n^x bags containing n^y elements each, and we encode the n^x bags as vectors over an alphabet of b symbols. The code length is $m := \log_b n^x = x\log_b 2 \cdot \log_2 n$. In stage 1 we test $bx\log_b 2 \cdot \log_2 n$ pools, each consisting of all elements that share a fixed symbol at a fixed position. At most 2 of the b pools for every position can be positive, otherwise $|P| > d = 2$ is confirmed. Let k be the number of positions where we obtain 2 positive answers.

If $k = 0$, then the positive answers identify one bag of n^y elements containing all, up to 2, defectives. This case can be solved with any 2-disjunct matrix in stage 2. It follows that $zy\log_2 n$ further pools in stage 2 are enough.

If $k > 0$, then it is easy to see that the candidate 2-sets form the edges of the disjoint union of 2^{k-1} complete bipartite graphs K_{n^y,n^y}. Clearly, the worst case is $k = m$. Thus, applying Theorem 10 we can solve this case in stage 2 by using fewer than $t(2^m,1,1) + 2t(n^y,1,1)$ pools. (In the bipartite graphs we search for 1 defective on both sides separately, therefore the factor 2.) Since $t(c,1,1) = \log_2 c + o(\log_2 c)$ as mentioned earlier, this pool number is $x\log_b 2 \cdot \log_2 n + 2y \cdot \log_2 n + o(\log_2 n)$.

Now we can bound the factor of $\log_2 n$ in the total pool number in both stages. We obtained $bx\log_b 2 + zy$ and $(b+1)x\log_b 2 + 2y$ in case $k = 0$ and $k > 0$, respectively. By trying the values of b and using the monotonicity of $bx\log_b 2$ for $b > 4$ one can figure out that $b := 4$ gives the optimal performance. Also recall that $x = 1 - y$. Then the two bounds evaluate to $2x + zy = 2 + (z-2)y$ and $2.5x + 2y = 2.5 - 0.5y$. Clearly their maximum is minimized if they are equal, that is, $2 + (z-2)y = 2.5 - 0.5y$, hence $y = \frac{1}{2z-3}$, which yields the assertion. To our best knowledge, the best published value of z is still 5.482 due to [19], which yields the second assertion. (We remark that $x := 1$ and $y := 0$ gives a basic strategy with factor 2.5.) □

We highlight that our strategy does not require extra time for computations, e.g., for decoding the test results: Excluding the non-candidate elements after stage 1 is trivial, the candidate graph is completely described by the partite sets of the complete bipartite subgraphs (we do not list the quadratically many edges), and stage 2 works with either a 2-disjunct matrix or, through Theorem 10, with some composition of 1-disjunct matrices that simultaneously identify the right bipartite graph and the right pair of vertices therein. (Details become evident from the construction in Theorem 10.) Finally, it is well known that d-disjunct matrices are easy to decode.

9 Further Research

Certainly, our tools can be further refined to limit the case inspections even more, and capture more n, d, s. 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. Already the construction of smallest d-disjunct matrices is notoriously difficult. Since the number of rows (pools) is $O(d^2 \log n)$, naive exhaustive search would need $2^{O(d^2 n \log n)} = n^{O(d^2 n)}$ time. But in the realm of exact and parameterized algorithms, one could turn structural theorems, as in [25] and in the present paper, into branching rules that fill the matrix entries more efficiently. The same question arises for general candidate hypergraphs, i.e., for the NP-complete set basis problem.

Some other open questions are intriguing, too: Regarding the product of candidate hypergraphs, in which cases is the nonadaptive test number additive, and how much better than the sum can it be else? Similarly, how can we optimally deal with disjoint sums of candidate hypergraphs? Such results would be useful for studying multistage pooling designs that first separate the candidates for defectives into disjoint subsets (as we have seen in some examples). Does there exist, for every d, some s such that $t(n, d, s) = t(n, d, n)$? In [4] we got an affirmative answer only for $d = 1$, namely $s = 2$.

Acknowledgments

Research of the first two authors has been supported by the Swedish Research Council (Vetenskapsrådet) through the grant 2010-4661, “Generalized and fast search strategies for parameterized problems”. Research of the third author was partially supported by grant no. OTKA 108947 of the Hungarian Scientific Research Fund and by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. A few more examples of non-additivity, as in Section 4, have been found by Endre Csóka at the 5th Emléktábla Workshop “Combinatorial Search” in Budapest and Erdőtarcsa (Hungary) 2013, organized by Dömötör Pálvölgyi and Balázs Patkós. We also thank the referees for their careful and encouraging comments.

References

[1] D.J. Balding, D.C. Torney, Optimal pooling designs with error detection, J. Comb. Theory A 74 (1996) 131–140

[2] S. Bezrukov, D. Fronček, S.J. Rosenberg, P. Kovář, On biclique coverings, Discr. Math. 308 (2008) 319–323

[3] 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

[4] P. Damaschke, A. Sheikh Muhammad, E. Triesch, Two new perspectives on multi-stage group testing, Algorithmica 67 (2013) 324–354

[5] A. De Bonis, G. Di Crescenco, Combinatorial group testing for corruption localizing hashing, in: B. Fu, D.Z. Du (Eds.) COCOON 2011, LNCS vol. 6842, Springer, Heidelberg, 2011, pp. 579–591.

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

[7] D. de Caen, D.A. Gregory, N.J. Pullman, The boolean rank of zero-one matrices, in: Proc. 3rd Caribbean Conf. on Combinatorics and Computing, 1981, pp. 169–173.

[8] D.Z. Du, F.K. Hwang, Combinatorial Group Testing and Its Applications, Series on Appl. Math. vol. 3, World Scientific, 2000

[9] D.Z. Du, F.K. Hwang, Pooling Designs and Nonadaptive Group Testing, Series on Appl. Math. vol. 18, World Scientific, 2006

[10] A.G. D'yachkov, Lectures on designing screening experiments, in: Lecture Note Series 10, Comb. and Comp. Math. Center, Pohang Univ. of Science and Technol. (2003)

[11] A.G. D'yachkov, A.J. Macula, V.V. Rykov, New constructions of superimposed codes, IEEE Trans. Info. Theory 46 (2000), 284–290

[12] A.G. D'yachkov, A.J. Macula, V.V. Rykov, New applications and results of superimposed code theory arising from the potentialities of molecular biology, in: Numbers, Information and Complexity, pp. 265–282, Kluwer, 2000

[13] A.G. D'yachkov, V.V. Rykov, Bounds on the length of disjunctive codes, Problems of Info. Transmission 18 (1982) 166–171

[14] A.G. D'yachkov, V.V. Rykov, A survey on superimposed code theory, Problems of Control and Info. Theory 12 (1983) 229–242

[15] A.G. D'yachkov, V.V. Rykov, A.M. Rashad, Superimposed distance codes, Problems of Control and Info. Theory 18 (1989) 237–250

[16] A.G. D'yachkov, P. Vilenkin, V.V. Rykov, D. Torney, Families of finite sets in which no intersection of ℓ sets is covered by the union of s others, J. Comb. Theory A 99 (2002) 195–218

[17] A.G. D'yachkov, I.V. Vorobyev, N.A. Polyanskii, V.Y. Shchukin, Bounds on the rate of superimposed codes, Preprint, arXiv 1401.0050 [cs.IT] (2014)

[18] 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

[19] P. Erdős, P. Frankl, Z. Füredi, Families of finite sets in which no set is covered by the union of two others, J. Comb. Theory A 33 (1982) 158–166

[20] J. Fang, Z.L. Jiang, S.M. Yiu, L.C.K. Hui, An efficient scheme for hard disk integrity check in digital forensics by hashing with combinatorial group testing, Int. J. Digital Content Technol. Appl. 5 (2011) 300–308

[21] P. Fischer, N. Klasner, I. Wegener, On the cut-off point for combinatorial group testing, Discr. Appl. Math. 91 (1999) 83–92

[22] D. Frondček, J. Jerebic, S. Klavžar, P. Kovář, Strong isometric dimension, biclique coverings, and Sperner's theorem, Comb. Probab. Comput. 16 (2007) 271–275

[23] H. Gao, F.K. Hwang, M.T. Thai, W. Wu, T. Znati, Construction of d(H)-disjunct matrix for group testing in hypergraphs, J. Comb. Optim. 12 (2006) 297–301

[24] 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

[25] S.H. Huang, F.K. Hwang, When is individual testing optimal for nonadaptive group testing? SIAM J. Discr. Math. 14 (2001) 540–548

[26] W.H. Kautz, R.C. Singleton, Nonrandom binary superimposed codes, IEEE Trans. Info. Theory 10 (1964) 363–377

[27] E. Kushilevitz, N. Nisan, Communication Complexity, Cambridge Univ. Press, 1997

[28] A.J. Macula, G.R. Reuter, Simplified searching for two defects, J. Statist. Planning Infer. 66 (1998) 77–82

[29] M. Mézard, C. Toninelli, Group testing with random pools: Optimal two-stage algorithms, IEEE Trans. Info. Theory 57 (2011) 1736–1745

[30] `www.redcrossblood.org/learn-about-blood/what-happens-donated-blood/blood-testing` (version as of Jan. 2013)

[31] J. Spencer, Minimal completely separating systems. J. Comb. Theory 8 (1970) 446–447

[32] L.J. Stockmeyer, The set basis problem is NP-complete, Tech. Report RC-5431, IBM, 1975

[33] Y. Xuan, I. Shin, M.T. Thai, T. Znati, Detecting application denial-of-service attacks: A group-testing-based approach, IEEE Trans. Par. Distr. Syst. 21 (2010) 1203–1216

[34] C.A. Weideman, D. Raghavarao, Some optimum non-adaptive hypergeometric group testing designs for identifying two defectives, J. Statist. Planning Infer. 16 (1987) 55–61

[35] B. Zhang, Group testing regression models, Dissertation, Dept. of Statistics, Univ. of Nebraska, Lincoln, 2012

[36] G. Zhang, L. Cui, A set coverage problem, Info. Proc. Letters 110 (2010) 158–159

Appendix: Optimality Proofs for Small Instances

$\mathbf{t(3,2,1+)=3}$, $\mathbf{t(4,2,1+)=4}$, $\mathbf{t(5,2,1+)=5}$ hold by the counting bound.

$\mathbf{t(6,2,1+)=6}$. 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 11 requires 5 more tests. Due to Lemma 12 we need not consider more cases.

$\mathbf{t(7,2,3+)=6}$ follows from $t(6,2,1+)=6$.

$\mathbf{t(7,2,2)=7}$. This is the first complicated case that needs the whole machinery prepared in this paper.

We assume for contradiction that $t(7,2,2)\leq 6$ and consider the pool hypergraph of stage 1. Assume that some ≥ 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 13. 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 apply Lemma 13 together with Lemma 11. 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 result $t(7,2,1)=7$ [25].

Next we show that cycles C_3,C_4,C_5 in the pool graph, together with edges or loops for the other elements, always create bad induced subgraphs that enforce too many tests in stage 2. In detail:

A C_3 with loop implies 4 more tests (by the example after Lemma 14). 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 Section 6). 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 (see Section 6). 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 11 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 11 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 (with loops) 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 11 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 14 implies $t(6,1,1)+1=5$ more tests.

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 Theorem 8, 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 $t(5,1,1)+2=4+2=6$ more tests are needed by Theorem 8.

$\mathbf{t(7,3,1+)=7}$. Follows from the 35 candidate 3-sets by Lemma 11.

$\mathbf{t(8,2,2+)=7}$. It suffices to consider adaptive strategies. 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 11 enforce 6 more tests. Due to Lemma 12 we need not consider more cases.

$\mathbf{t(8,2,1)=8}$ is implicit in [25].

$\mathbf{t(8,3,1+)=8}$. 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 11 and finally Lemma 12.

$\mathbf{t(9,2,1)=9}$ is known from [25].

$\mathbf{t(9,3,1+)=9}$. 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 11. By Lemma 12 and exploring symmetric cases we can prune most of the tree. We remark that our proof has to check just 11 paths, compared to the host of possible strategies and answers. They are presented here:
$\{v_1\}(-)$: $t(8,3,8)=8$.
$\{v_1,v_2\}(+)$, $\{v_1\}(+)$: $t(8,2,8)=7$.
$\{v_1,v_2\}(+)$, $\{v_3,v_4\}(-)$, $\{v_1\}(+)$: 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 both 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 2-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.

$\mathbf{t(10,2,3+) = 7}$ holds since $t(10,2,3) \geq t(8,2,3) = 7$.

$\mathbf{t(10,2,2) = 8}$. 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 11, 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 11. 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. Thereom 8 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 (Theorem 8), 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, again due to Theorem 8.

$\mathbf{t(10,2,1) = 9}$ follows from [25].

$\mathbf{t(10,3,3+) = 9}$ holds since already $t(9,3,1+) = 9$.

$\mathbf{t(10,4,1+) = 10}$. 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 are still to be found, but we have $t(9,3,9) = 9$. By Lemma 12, this case distinction is complete.

$\mathbf{t(15,2,2) = 9}$. The value of $t(14,2,2)$ remains open. In the following we use Lemma 11 and Lemma 13. Assume that $t(15,2,2) \leq 8$, and consider the pools in stage 1. A positive $\geq$ 7-pool allows $70 > 2^6$ candidate 2-sets, hence the remaining 7 tests would not be sufficient. A negative $\leq$ 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 positively, there remain $69 > 2^6$ candidate 2-sets, implying 8 more tests.

$\mathbf{t(22,2,2) = 10}$. Here, the exact $t(n,2,2)$ values for $n = 19, 20, 21$ remain open. 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.