Combinatorial Search in Two and More Rounds

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

Abstract

In a combinatorial search problem we wish to identify an unknown element by binary tests, where the edges of a hypergraph specify the available tests. We show that, for rather general cases of this problem, the worst-case minimum number of tests, even if adaptive testing is permitted, can already be achieved in a small number of rounds of parallel tests. In particular, the maximum number of necessary rounds grows only as the square root of the number of elements, and two rounds are enough if, e.g., the test number is close to the number of elements, or the hypergraph is a graph. We also provide polynomial-time, hardness, and parameterized results on the computational complexity of finding optimal strategies for some cases, including graphs and tree hypergraphs.

Keywords: combinatorial search, test cover, adaptive strategy, matching

1 Introduction

1.1 Problem Statement

A general combinatorial search problem where tests are binary, deterministic, independent and not destructive can be formalized as a *hypergraph* $\mathcal{H}$, which consists of a set of n *vertices* and a family of subsets called *edges*. We also refer to the edges as *tests* and to the hypergraph as a *test set*. These terms are used interchangeably; we prefer the hypergraph terminology for purely combinatorial statements, and the term test set when it comes to the search problem: One unknown vertex u is the *target*. A test T is positive if $u \in T$ and negative otherwise. The goal is to identify u efficiently, by using tests from $\mathcal{H}$. More precisely, we want strategies that minimize the worst-case number of tests needed to identify every possible target.

Tests may be performed in *rounds* (also called stages), where tests in a round are executed simultaneously, but the tests selected for a round may depend on all previous outcomes. In *adaptive* testing, only one test is done per round. The main focus of this paper is on the relation between tests and rounds. Adaptive strategies can take too much time, and we would also like to limit the number of rounds. But parallelization of tests may (or may not) increase the necessary total number of tests. This gives rise to the following notion.

[*]Tel. 0046-31-772-5405. Fax 0046-31-772-3663.

Definition 1. *A test set $\mathcal{H}$ on n vertices is (n,t,r) if it satisfies: (i) There exists a strategy A such that, for every possible target u, this A identifies u using a total of t tests in r rounds. (ii) The pair (t,r) is optimal, in the sense that no strategy can identify every target u by t tests in $r-1$ rounds, and no strategy can identify every target u in r rounds by $t-1$ tests. Furthermore, let $r(\mathcal{H})$ denote the largest r for which there exists t such that $\mathcal{H}$ is an (n,t,r) test set.*

In other words, one cannot improve t or r without sacrificing the other measure. However, note that an (n,t,r) test set may still admit strategies with strictly larger t and smaller r, or vice versa. Number $r(\mathcal{H})$ is particularly interesting: The corresponding test number t cannot be improved to some $t' < t$, since then $\mathcal{H}$ would also be (n,t',r') for some $r' > r(\mathcal{H})$, contradicting the maximality of $r(\mathcal{H})$. Hence this t is the optimal worst-case test number for $\mathcal{H}$, even if adaptive testing is permitted. Thus $\mathcal{H}$ allows a test-optimal strategy using only $r(\mathcal{H})$ rounds. In other words, it is never meaningful to spend more than $r(\mathcal{H})$ rounds on searching, at least not from the worst-case test number point of view. The major type of results in our paper is that $r(\mathcal{H})$ is "small"; see details below.

1.2 Examples and Related Work

Combinatorial search, or learning an object by queries, subsumes disparate classic problems where the vertices are structured and test sets are implicitly described (we give a few illustrations below). Minimizing the number of rounds is a core topic in this field, so it appears only natural to look into this matter also in the *fully general* search problem, that is, on *arbitrary* hypergraphs, in order to delimit what is possible or not in any special search problems.

In the *group testing* problem, at most d of m items are defective, arbitrary subsets of items can be tested for the presence of defectives, and the goal is to find all defectives. We can model group testing in our framework in an obvious way: We create $n = O(m^d)$ elements, one for each subset of at most d items. The target is the unique element representing the exact set of defectives, and every group test is modelled by an edge containing those elements which represent subsets of items that remain candidates for the defective set if the test is positive. The worst-case number of tests in adaptive group testing strategies is $O(d \log m)$, and remarkably, this test number cannot be achieved in one round due to the lower bound $\Omega(\frac{d^2}{\log d} \log m)$ in [14], whereas already two rounds allow strategies with $O(d \log m)$ tests [15, 11, 17, 16], which was a major insight in group testing and is desired in its applications. The number of rounds in another group testing variant is addressed in [6].

Sorting a set of m distinct numbers by comparisons is a combinatorial search problem, too. Here the vertices are the $n = m!$ permutations, and a comparison is obviously a binary test. Thus we can think of sorting networks with t comparators and depth r as a special type of strategies that search for the correct permutation using a total of t tests in r rounds. For classic and recent complexity results on size versus depth see [1, 27, 5, 21].

Back to general test sets, the case of one round, that is, recognizing whether a test set is $(n,t,1)$, is known as the *test cover* problem. It has direct applications, e.g., in biological testing. This problem is NP-hard and has been intensively studied from the approximation

[12, 10], heuristic [13, 18], and parameterized complexity angle [2, 8, 9, 22]. Much less is known for $r > 1$, actually, the only major result we are aware of is a sufficient condition for test sets that cannot save tests in $r \geq 3$ rounds compared to $r = 1$ [28].

In some applications, the tests have sizes limited by a constant, so this case has received special attention [12, 8]. Even edges with only two vertices (that is, the given hypergraph is a graph) are of interest. There, an optimal one-round strategy tests a maximum packing of k vertex-disjoint 2-paths (pairs of incident edges) and adds a trivial test cover for the remaining graph [12]. However, finding a maximum packing of 2-paths is NP-hard; an $O^*(2.482^{3k})$ time parameterized algorithm is also known [19]. Moreover, an optimal test cover in a graph can be approximated within a factor 4/3, and the problem is APX-hard [12].

Another important special case with several applications is tree hypergraphs where test sets correspond to subtrees obtained by removing single edges in a tree. (See also Section 2 for the definition.) Adaptive testing on tree hypergraphs is equivalent to edge ranking, which has been extensively studied in many directions, see, e.g., [7] for an informal survey and open problems. An optimal adaptive strategy can be computed in $O(n)$ time [25].

Some papers [24, 20] study search strategies in two and more rounds, however for the problem of finding one of several targets, and with arbitrary subsets as queries.

1.3 Our Contributions

Section 2 defines more terminology and reviews some known or simple facts on $(n, t, 1)$ test sets, as context for our results on $r \geq 2$ rounds. Our proof methods in the following sections are elementary, but this appears to be the first systematic exploration of tests versus rounds in arbitrary test sets.

A natural question to start with is how large $r(\mathcal{H})$ can ever be. As a first main result of the paper, Section 3 shows that, somewhat surprisingly, $r(\mathcal{H}) \leq \sqrt{2n+4}$ is a general upper bound. In other words, the minimum number of tests can always be implemented in as few as $\sqrt{2n+4}$ rounds. Moreover, this bound is the best possible, subject to an additive constant, since we give an infinite sequence of hypergraphs $\mathcal{H}$ with $r(\mathcal{H}) = \sqrt{2n} - \Theta(1)$.

Section 4 provides a combinatorial concept that we will use to characterize, in many cases, the optimal test number for $r \geq 2$ rounds. Based on this concept, Sections 5 and 6 show that $r(\mathcal{H})$ is as small as 2 for many hypergraphs: The main result is that $r(\mathcal{H}) \leq 2$, if $\mathcal{H}$ has an adaptive test complexity $\Omega(n)$. If $\mathcal{H}$ is a graph (every edge has size at most 2), we can compute an optimal two-round strategy in polynomial time. Notice that it is not our consideration that is limited to two rounds, rather, it comes out as a *result* that two rounds are sufficient to implement an optimal test number in instances that need many tests.

The "take-away message" from the above results is that $r(\mathcal{H})$ is far below n in general, and even constant for a wide range of test sets. It may meet the expectations that tests can be highly parallelized in instances requiring many tests. However, from this intuition alone it is not easy to guess how many rounds are actually needed to accommodate a strategy with an optimal test number. Actually, the lower bound in Section 3 reveals that many rounds can be necessary in some cases, too, and the $\sqrt{2n}$ borderline is not obvious. (See also the open problem concerning the entire trade-off in Section 3.)

Besides the combinatorial complexity, i.e., number of tests versus rounds, we also study the computational complexity of constructing optimal strategies. Section 7 deals with computing the optimal test number t for two rounds. Already this sub-problem turns out to be NP-hard, therefore only special cases can be efficiently solved. In particular, we show that the problem belongs to FPT with $n-t$ as the parameter. We mention that the problem for one round is in FPT, too [9, 2]. Moreover, by our aforementioned result $r(\mathcal{H}) \leq 2$ for small $n-t$, more than two rounds are not helpful if the parameter $n-t$ is small. Therefore, the problem in this parameter is now settled completely.

Deciding whether $\log_2 n + k$ tests suffice is W[2]-hard, in the parameter k, for any fixed r. Previously this was known for $r=1$ only [9]. We solve the case of graphs, for $r \geq 2$ rounds, in polynomial time already in Section 6. Although not being particularly difficult, we think that this is another main result, as it contrasts nicely to hardness of the graph case for $r=1$. In Section 8 we consider tree hypergraphs. Searching them in one round is trivial, as it always needs $n-1$ tests. We give an $O(n \log^2 n)$ time algorithm to compute an optimal strategy for two rounds.

2 Preliminaries

Some of the notation in this and the next section might appear involved at first glance, but it compacts the formulations, both here and later on.

- If no confusion arises, symbol $\mathcal{H}$ denotes a hypergraph or just the family of its edges.
- For a subset V of vertices, the hypergraph $\mathcal{H}$ *restricted* to V, denoted $\mathcal{H} \downarrow V$, is obtained from $\mathcal{H}$ by deleting all vertices outside V from the edges. Thus, edges having the same intersection with V become identical on $\mathcal{H} \downarrow V$, and edges disjoint to V become empty. In these cases we remove all copies of identical edges but one, and we remove empty edges.
- For a family $\mathcal{E}$ of edges, we denote by $\bigcup \mathcal{E}$ the set of all vertices in the union of all edges in $\mathcal{E}$.
- A k-set ($\leq k$-set, $\geq k$-set) is a set with exactly (at most, at least) k vertices. We use this notation also for sets in special roles, e.g., we speak of k-edges.
- The test set must distinguish all vertices by binary properties. Accordingly, a hypergraph is called a *separating system*, or just *separating*, if for any two vertices v, w there exists an edge containing exactly one of v, w.
- With respect to a fixed hypergraph, a *class* means a maximal set of vertices that belong to precisely the same edges.
- Thus a hypergraph divides its vertices into classes. We say that it *induces* these classes. Note that a hypergraph is separating if and only if all induced classes are singletons.

- An edge T and its complement $\overline{T}$ are obviously equivalent tests. A test set is *complement-free* if it contains no pair of tests T and $\overline{T}$. Thus we silently assume, unless said otherwise, that any given test set is separating and complement-free, and all tests are $\leq n/2$-edges.

- We refer to a test cover with $t = n - 1$ tests in one round as a *trivial test cover*.

Let $\mathcal{T}$ be a given test set. The test set $\mathcal{R} \subseteq \mathcal{T}$ used in a round identifies which class C induced by $\mathcal{R}$ contains the target, and the remaining search problem is specified by $\mathcal{T} \downarrow C$, which is separating if $\mathcal{T}$ was.

Obviously, the worst-case number t of tests always satisfies $t \geq \log_2 n$, and $t \geq r$ for a strategy using r rounds. From the definition of separating systems it also follows easily that $t \leq n-1$ tests in one round are always enough: Starting from an empty $\mathcal{R}$ we can successively add tests from $\mathcal{T}$ to $\mathcal{R}$ in order to partition classes that still have more than one element, and after $n-1$ steps all classes are singleton sets. Then, test $\mathcal{R}$ in one round. Clearly, also optimal r-round strategies are built of at most $n-1$ different tests, which in turn implies that the set of triples $(\mathcal{T}, t, r)$, where $\mathcal{T}$ admits a strategy with t tests in r rounds, belongs to NP.

A natural question is: When does $t \leq n - k$ hold, for a prescribed k? The following definition and theorem, essentially taken from [9], answers this question for $r = 1$. (The reason for the condition $s \leq 2k$ is not obvious, see [9] for details.)

Definition 2. *A test set on n vertices is an (n,k)-mini test cover, (n,k)-MTC for short, if it has $s \leq 2k$ edges and induces $s + k$ or more classes.*

Theorem 3. *A test set $\mathcal{T}$ allows to identify every target with $n - k$ tests in one round if and only if $\mathcal{T}$ contains an (n,k)-MTC. Thus, $\mathcal{T}$ is $(n, n-k, 1)$ if and only if $\mathcal{T}$ contains an (n,k)-MTC but no $(n, k+1)$-MTC.*

- We define the notion of *crossing* edges by saying that two edges cross if they induce four classes.

- A *tree hypergraph* is a hypergraph without crossing edges.

Any single edge is an $(n,1)$-MTC but not an $(n,2)$-MTC. Note that any two edges form an $(n,2)$-MTC if and only if they cross. Tree hypergraphs can be recognized straightforwardly in linear time: View the edges in any order and check that every edge induces at most one class more.

Since a tree hypergraph with t edges induces at most $t+1$ classes, it does not contain any $(n,2)$-MTC (see Definition 2). From these observations and Theorem 3 we see:

Corollary 4. *A test set is $(n, n-1, 1)$ if and only if it is a tree hypergraph. Moreover, the only minimal $(n,2)$-MTC are the crossing edge pairs.*

3 Tight Upper Bound on Adaptivity

We give the first main result that limits the number of meaningful rounds in arbitrary test sets.

Theorem 5. *For (n,t,r) test sets we have $r \leq \sqrt{2n+4}$. In other words, $r(\mathcal{T}) \leq \sqrt{2n+4}$ holds for all test sets $\mathcal{T}$ on n vertices.*

Proof. A test strategy can be naturally viewed as a rooted tree, where every non-leaf node q_i represents a set $\mathcal{T}_i$ of t_i edges tested in a round. Then one child of the tested node is reached depending on all test outcomes. In every leaf of the tree, the target is identified and no further testing takes place. The depth d of a tree is the number of arcs on a longest path from the root to some leaf. Depth equals the maximum number of rounds. We can assume that a strategy never performs redundant tests (that do not induce new classes).

Let us study the tree of any strategy A, with depth $d > 2$. Let q_1 denote the root, and assume that $\mathcal{T}_1$ induces some $\geq (n-d+1)$-class C. Let q_2 be the child of q_1 that we reach if the target is in C. We construct another strategy A' as follows. We test $\mathcal{T}_1 \cup \mathcal{T}_2$ already in round 1. If the target is in C, we continue as in strategy A below node q_2. In all other cases we apply a trivial test cover to the induced class in $\overline{C}$ containing the target. The tree of A' is easy to describe: Copy the subtree rooted at q_2, make q_2 the root, and place under the new root also new subtrees of depth 1 for the trivial test covers on the other classes. Obviously the depth has strictly decreased.

Let τ be the worst-case number of tests done by A. If the target is in C, then strategy A' does not run more than τ tests, since it has done the same tests as A and continues like A. In all other cases there remains some $\leq (d-t_1)$-class containing the target, due to the following argument. $\overline{C}$ has at most $d-1$ vertices, and A has already done t_1 tests in round 1. Since no test is redundant, every test (except perhaps one that equals $\overline{C}$) splits off a new class in $\overline{C}$. Even in the worst case, the largest class outside C retains at most $(d-1)-(t_1-1) = d-t_1$ vertices.

Hence, if the target is in $\overline{C}$, then the two rounds of A' together never exceed $(t_1+t_2)+(d-t_1)-1 = d-1+t_2 \leq d-2+t_1+t_2$ tests. We show that $d-2+t_1+t_2 \leq \tau$, which then implies that A' executes at most τ tests. To see this inequality, consider a path in A with the maximum depth d. Since only $\leq (d-1)$-classes exist outside C, any path not going through q_2 does at most $d-2$ further tests, thus its total length is below d. That is, some path through q_2 in A attains the maximum depth d, hence it performs t_1+t_2 tests at q_1 and q_2, and at least $d-2$ tests in the subsequent $d-2$ rounds.

Thus we have shown: Either we can decrease the depth without increasing the worst-case number of tests, or the edges tested at the root induce only $\leq (n-d)$-classes. In simpler words, in the latter case all possible test outcomes at the root discard at least d vertices. This suggests the following auxiliary concept:

We call a strategy (and its tree) *eager* if the first round always discards at least d vertices.

Now consider an (n,t,r) test set and a corresponding strategy. The tree T is eager, otherwise the test set is $(n,t,r-1)$ or better. Let q be any node in T, and let d_q be the depth of the subtree $T(q)$ rooted at q. Next we make all subtrees of depth $d_q > 2$ eager. (Note that d_q has to be used in the role of d.) If $T(q)$ is not eager, we rearrange $T(q)$ as above. Clearly this does not increase the depth of T. Moreover, root-leaf paths which avoid q are not affected by the rearrangement. For any root-leaf path through q, let s be the number of tests done on this path before it enters q. The strategy in $T(q)$, before rearrangement, performed at most $t-s$ tests on every path, since the test number of T was at most t. Since our rearrangement does not increase the test number in $T(q)$, it is still bounded by $t-s$, hence the total test number is still bounded by t.

Iterating this procedure we get rid of the non-eager subtrees of depth larger than 2, unless the process runs into a cycle and never terminates. To rule out this possibility it suffices to identify some integer variable $D \geq 0$ that strictly decreases in every step. For instance, we may define it by $D := \sum_q (d_q - 1)$, where the sum is taken over all non-leaf nodes q. Every transformation step strictly decreases some terms in D and does not create new positive terms, since we only move some subtree upwards, while each newly created subtree appended to its root has depth 1. Hence D strictly decreases.

Altogether this shows the existence of a strategy for the (n,t,r) test set where all subtrees of depth larger than 2 are eager. Note that the tree has some path of length exactly r. If the test outcomes follow this path, then round i discards at least $r-i+1$ vertices. But this is possible all the way down only if $\sum_{i=1}^{r-2}(r-i+1) \leq n-1$. (Summation has to stop at $i=r-2$ since we had to assume depth at least 3.) This is equivalent to $\sum_{j=3}^{r} j = r(r+1)/2 - 3 \leq n-1$, hence $r \leq \sqrt{2n+4}$. $\square$

A refined analysis might further bring down the small additive constant. We remark that the proof efficiently transforms any strategy A into a strategy A' with the claimed maximum number of rounds which, however, can be larger than $r(\mathcal{T})$ for the given test set $\mathcal{T}$. (For instance, we may add to $\mathcal{T}$ an optimal test cover for one round. Then A' remains a valid strategy but uses at least two rounds whereas $r(\mathcal{T})=1$. But, interestingly enough, our general upper bound is essentially the best possible:

Theorem 6. *For arbitrarily large n there exist $(n, r+1, r)$ test sets $\mathcal{T}$ with $r = \sqrt{2n} - \Theta(1)$. That is, the bound $r(\mathcal{T}) \leq \sqrt{2n+4}$ is tight up to a constant summand.*

Proof. Our example is a test set $\mathcal{T}$ with $n = k(k+1)/2$ vertices and pairwise disjoint edges of decreasing sizes $k, k-1, k-2, \ldots, 2, 1$ plus all singletons. A simple strategy tests the edges by descending sizes, and when the ith test is the first positive one, the strategy tests all vertices but one, in this positive edge. This needs $i + (k-i+1) - 1 = k$ tests for any target. We show that k is the optimal test number, and $k-1$ rounds are needed in the worst case to guarantee these k tests. Together this means that $\mathcal{T}$ is $(n, k, k-1)$, and that $r(\mathcal{T}) = k-1$.

Let K denote the k-edge, and consider any strategy. In the event that K hosts the target, we must have eventually tested $k-1$ of the singletons in K, since otherwise the target could be among the two or more untested vertices, and we cannot distinguish them. In other words, for any strategy there exists a target in K for which the strategy needs $k-1$ tests. With this in mind, assume that some of the edges tested in round 1 is disjoint to K. Then, if the target is in K, we still have to test $k-1$ singletons in K, these are together k tests. If K itself is tested as well, we apply too many tests. If K is not tested, consider an untested $v \in K$. In order to decide whether v is the target, we must cover all vertices outside K by tests, which requires even more tests. This shows that round 1 can only test edges $E \subseteq K$, or we can be forced to exceed k tests. But then, if the tests are negative, we are left with a hypergraph of the same structure, with $k-1$ in the role of k (and perhaps further untested vertices from K). By an inductive argument, this hypergraph is $(n-k, k-1, k-2)$. For the induction base $k=2$ we have a $(3,2,1)$ test set. With $r=k-1$ we obtain $(r+1)(r+2)=k(k+1)=2n$, hence $r=\sqrt{2n}-\Theta(1)$. □

The example used in the proof has $\sqrt{2n}-\Theta(1)$ edges, thus, of course, more rounds are of no use there. The next natural question is whether some test sets with more that $\sqrt{2n}$ edges still need this worst-case number of rounds. In the following sections we will see that $r(\mathcal{H})$ gets down to 2 when the optimal number of tests is close to n. More ambitiously, it would be interesting to figure out the optimal t versus r trade-off for the full range $\sqrt{2n}<t<n$; we must leave this question open.

4 Gain of a Hypergraph

The intuition behind the following concept of gain is as follows. It is desirable to find an edge containing the target early, because then the remaining search can be confined to this edge. This costs one negative test for every edge where the target is not found, but every such test also excludes every vertex in the edge as possible target. Thus it appears to be a good idea to cover a maximum number of distinct vertices by some number s of edges. Detailed investigation shows that, essentially, the number of covered vertices minus s is in fact the right measure for our purpose. Merely for formal reasons we add an offset 1.

Definition 7. *Let $\mathcal{H}$ be a hypergraph on n vertices. The* gain *of $\mathcal{H}$ is the largest possible number $g=|\bigcup\mathcal{S}|-s+1$, where $\mathcal{S}\subseteq\mathcal{H}$ is a selection of s edges with $|\bigcup\mathcal{S}|<n$, that is, not covering all vertices. Such $\mathcal{S}$ is a* gain certificate *for the gain g if $|\bigcup\mathcal{S}|=s+g-1$, and $\mathcal{S}$ is a* minimal gain certificate *if no proper subset of $\mathcal{S}$ is a gain certificate.*

Note that, equivalently, the gain is also the smallest number g such that $|\bigcup\mathcal{S}|\le s+g-1$ holds for every s and every selection $\mathcal{S}\subseteq\mathcal{H}$ of s edges with $|\bigcup\mathcal{S}|<n$.

Definition 8. *In a permutation (linear order) of a set $\mathcal{S}$ of edges, the ith* expansion *x_i is the number of vertices in the ith edge that occur in no earlier edge. A permutation of $\mathcal{S}$ is* L-maximal *(L stands for "lexicographically") if x_1 is maximum among all permutations of $\mathcal{S}$, x_2 is maximum among all permutations of $\mathcal{S}$ that maximize x_1, and so on.*

Lemma 9. *Consider any hypergraph $\mathcal{H}$ with gain g.*
(a) Every set of edges has an L-maximal permutation, with monotone non-increasing expansions.
(b) If $g>k$ then $\mathcal{H}$ contains a selection $\mathcal{S}$ of $s\le k$ edges with $|\bigcup\mathcal{S}|\ge s+k$ and $|\bigcup\mathcal{S}|<n$.
(c) $\mathcal{H}$ contains a (minimal) gain certificate with expansions $x_1\ge\ldots\ge x_s\ge 2$, where $s<g$.

Proof. For (a) consider an L-maximal permutation. If $x_i<x_{i+1}$ for some i, then we switch the two edges and obtain a sequence with increased x_i, which contradicts L-maximality.

For (b) note that a hypergraph of gain larger than k contains, by definition, a selection $\mathcal{S}$ of s edges with $n>|\bigcup\mathcal{S}|\ge s+k$. We show that $s\le k$ can be accomplished. We take an L-maximal permutation of $\mathcal{S}$ as in (a). Let $\mathcal{S}' \subseteq \mathcal{S}$ be the prefix of edges with expansions larger than 1, and let $s' := |\mathcal{S}'|$. From $|\bigcup \mathcal{S}| \geq s + k$ we get $|\bigcup \mathcal{S}'| \geq s' + k$. If $s' \leq k$ then we are done, since S' (in the role of S) is the required selection. If $s' > k$, then consider the L-maximal permutation of $\mathcal{S}'$ that is a prefix of the L-maximal permutation of $\mathcal{S}$ considered above. Clearly, all expansions therein are at least 2. Therefore, the first k edges cover at least $2k$ vertices, and the assertion holds with k in the role of s.

For (c), apply the reasoning from (b) with $k = g - 1$ and note that $s + g - 1 = s + k \leq |\bigcup \mathcal{S}| \leq s + g - 1$. Thus the selection from (b) is a gain certificate, all expansions in the L-maximal permutation are at least 2, and therefore the gain certificate is also minimal. □

The connections between gain and multi-round testing are established by the following bounds.

Theorem 10. *In a test set with gain g, even any adaptive strategy needs at least $n - g$ tests in the worst case.*

Proof. Consider any adaptive strategy and the event that all tests are negative. Let t be their number. Since the gain is g, these t negative edges cover at most $t + g - 1$ elements. But they must cover $n - 1$ elements, otherwise the target is not yet identified. Thus $t + g - 1 \geq n - 1$, that is, $t \geq n - g$. □

Theorem 11. *Any test set $\mathcal{T}$ on n vertices and with gain g admits a test strategy with at most $\max\{g, n - g\}$ tests in two rounds. Specifically, it tests a minimal gain certificate with $s < g$ edges in round 1 and applies a trivial test cover in round 2.*

Proof. By Lemma 9 (b) there exists a selection $\mathcal{S}$ of $s \leq g - 1$ edges with $|\bigcup \mathcal{S}| = s + g - 1$. Let us test all edges of $\mathcal{S}$ in round 1.

If all these tests are negative, there remain $n - (s + g - 1)$ vertices. A trivial test cover finds the target among them by using $n - (s + g - 1) - 1$ further tests in round 2. Hence, in total $s + n - (s + g - 1) - 1 = n - g$ tests are sufficient in this case.

If some test in $\mathcal{S}$ is positive, let j denote the smallest index of such a positive test, in an L-maximal permutation of $\mathcal{S}$. Then a trivial test cover finds the target by using $x_j - 1$ further tests in round 2. In the worst case this is $x_1 - 1$, and a total of at most $s + x_1 - 1$ tests are made. By Lemma 9 (c), $\mathcal{S}$ is a minimal gain certificate, with expansions $x_i \geq 2$ for all i. It follows

$$g = \sum_{i=1}^{s} x_i - s + 1 = x_1 + \sum_{i=2}^{s} x_i - (s - 1) = x_1 + \sum_{i=2}^{s} (x_i - 1) \geq x_1 + s - 1,$$

thus at most g test are made in this case. □

For gains $g \geq \lceil n/2 \rceil$ we can sharpen this upper bound. The proof is similar though.

Theorem 12. *Any test set $\mathcal{T}$ on n vertices and with gain $g \geq \lceil n/2 \rceil$ admits a test strategy with at most $\lceil n/2 \rceil$ tests in two rounds.*

Proof. For convenience define $m := \lceil n/2 \rceil$. Take an L-maximal permutation of a gain certificate $\mathcal{S}$ of s edges, and let x_i denote their expansions. Observe that $|\bigcup \mathcal{S}| = s+g-1 \geq s+m-1$. Let $\mathcal{P}$ be the shortest prefix of the ordered sequence $\mathcal{S}$ that still satisfies $|\bigcup \mathcal{P}| \geq p+m-1$, where $p := |\mathcal{P}|$. We test the p edges of $\mathcal{P}$ in round 1.

If all tests are negative, then round 2 needs at most $n-(p+m-1)-1 = n-m-p \leq m-p$ tests. If some tests are positive, the worst case is to have $x_1 - 1$ tests in round 2. If $p = 1$ then $\mathcal{P}$ is a single m-edge, and the assertion of the Theorem is obvious. For $p \geq 2$, let $\mathcal{P}'$ be the set $\mathcal{P}$ without its last edge in the considered order. Then $|\bigcup \mathcal{P}'| \leq (p+m-1)-2 = (p-1)+m-2$ holds due to the minimality of p. Since $x_i \geq 2$ for all expansions, this also implies $x_1 + 2(p-2) \leq (p-1)+m-2$, thus $p + x_1 - 1 \leq m$. □

The bound is tight up to a constant summand: A test set consisting of two disjoint edges of size almost $n/2$ (to avoid complements) and all singletons actually needs almost $n/2$ tests in two rounds despite a gain nearly n. The same example shows that a large gain is not always helpful in one round, as this is a tree hypergraph and thus an $(n, n-1, 1)$ test set. As another side remark, Theorem 12 says that a large gain guarantees a global upper bound on the number t of tests in two rounds, in analogy to the global upper bound $t \leq n-1$ for one round.

5 Two and More Rounds

Now we use the concept of gain and the preceding results to determine the number of tests needed in two or more rounds, especially if it is larger than $\lceil n/2 \rceil$. The following point (b) gives a complete characterization in this case.

Theorem 13. *Let $\mathcal{T}$ be a test set on n vertices. For each $k < \lfloor n/2 \rfloor$ we have:*
(a) $\mathcal{T}$ has gain k if and only if the optimal worst-case number of tests needed in two rounds equals $n-k$.
(b) $\mathcal{T}$ is $(n, n-k, 2)$ if and only if: $\mathcal{T}$ contains no (n,k)-MTC, and $\mathcal{T}$ has gain k. In the affirmative case, $\mathcal{T}$ contains a selection $\mathcal{S}$ of $s < k$ edges with $|\cup \mathcal{S}| = s+k-1 < n$, and testing $\mathcal{S}$ in round 1 followed by a trivial test cover in round 2 guarantees at most $n-k$ tests.

Proof. For (a), let g denote the gain of $\mathcal{T}$. First suppose that $g = k$. Since $g < \lfloor n/2 \rfloor \leq n/2$, according to Theorem 11 we need at most $n-g$ tests in two rounds, and by Theorem 10 this is optimal (even for adaptive strategies). Conversely, suppose that the optimal worst-case number of tests needed in two rounds is $n-k$. Theorem 10 yields $n-k \geq n-g$, hence $k \leq g$. Assume $g \geq \lceil n/2 \rceil$ for the moment. Due to Theorem 12 this would imply $n-k \leq \lceil n/2 \rceil$, contradicting $k < \lfloor n/2 \rfloor$. Thus $g < \lceil n/2 \rceil$, which also means $g \leq n/2$. Applying Theorem 11 again this yields $n-k \leq n-g$, hence $k \geq g$, altogether $g = k$.

For (b), we put the previous results together. Suppose that $\mathcal{T}$ is $(n, n-k, 2)$. No (n,k)-MTC is present, otherwise the test set is $(n, n-k, 1)$ or better; see Theorem 3. The gain is k, as already seen in (a), and Theorem 9 yields that some $s < k$ edges have an $(s+k-1)$-set as their union. Conversely, suppose that $\mathcal{T}$ satisfies the listed conditions. Since the gain is k, we have that $n-k$ tests are optimal in two rounds, hence we cannot save tests in two rounds.

Since no (n, k)-MTC is present, we cannot save a round either. Altogether $\mathcal{T}$ is $(n, n-k, 2)$. A suitable test set for round 1 is also provided by Theorem 11. $\square$

The following corollary also addresses the case of $r > 2$ rounds. Apart from the technical formulation, the result says that two rounds are sufficient to accommodate the optimal worst-case number of tests, if this number is a large enough fraction of n.

Corollary 14. *No $(n, n-k, r)$ test set exists for $k < \lfloor n/2 \rfloor$ and $r > 2$. Consequently, if a test set $\mathcal{T}$ requires $t > \lceil n/2 \rceil$ adaptive tests, then $r(\mathcal{T}) \leq 2$.*

Proof. Assume that $\mathcal{T}$ is $(n, n-k, r)$. Then we need at least $n-k$ tests in two rounds. Theorem 13 implies that $\mathcal{T}$ has gain $g \leq k$. From Theorem 10 we know that at least $n-g$ tests are necessary also in r rounds. Thus $k \leq g$, implying $g = k$. Applying Theorem 13 again we see that exactly $n-k$ tests are needed in two rounds. This contradicts $r > 2$. The last assertion follows by setting $k = n-t$. $\square$

6 The Case of Graphs

In this small section we will see that, in test sets $\mathcal{T}$ with only ≤ 2-edges, optimal test strategies are easy to characterize and to compute for any $r \geq 2$. This is worth emphasizing, as the graph case found already special attention for $r = 1$, and the following result contrasts nicely to the known hardness for $r = 1$.

Theorem 15. *In a test set $\mathcal{T}$ of only ≤ 2-edges, an optimal two-round strategy works as follows. If the graph has a perfect matching, then test all its edges, except one, in round 1. Otherwise, test some maximum matching in round 1. In both cases, apply a trivial test cover in round 2. Hence an optimal strategy can be computed in polynomial time. Furthermore we have $r(\mathcal{T}) \leq 2$.*

Proof. Let n be the number of elements, and let μ be the size of a maximum matching M in the graph G of 2-edges. By Lemma 9, the gain is $g = \mu$ if G has a perfect matching, and $g = \mu + 1$ otherwise. (In the former case note that, by definition, a minimal gain certificate does not cover all elements, hence it contains all edges of a perfect matching except one, such that $g = \mu - 1 + 1 = \mu$.)

If M is a perfect matching, then the proposed strategy needs at most $(\mu - 1) + 1 = \mu = n - \mu = n - g$ tests in two rounds. If M is not a perfect matching and $n > 2\mu + 1$, then the proposed strategy needs at most $\mu + (n - 2\mu - 1) = n - g$ tests in two rounds as well. By Theorem 10, this test number is optimal. If $n = 2\mu + 1$, then the proposed strategy needs at most $\mu + 1$ tests in two rounds, as in the worst case some test in round 1 is positive. Therefore, in this case the general lower bound from Theorem 10 is slightly too weak, but we show optimality separately: Consider any adaptive strategy. The first $\mu - 1$ tests cover at most $2\mu - 2$ elements, hence at least 3 elements are left. If the next test covers two new elements and is positive, or covers only one new element and is negative, then still two candidates for the target exist, and another test is needed; these are together $\mu + 1$.

A maximum matching is found in polynomial time [26], and $r(\mathcal{T}) \leq 2$ holds since the optimal number of tests is achieved in at most two rounds. □

We also add a small corollary for arbitrary test sets and $k = n - t = 2$. Explicit characterizations of $(n, n-k, r)$ test sets become more complicated for $k \geq 3$.

Corollary 16. *For $n \geq 4$ elements, a test set is $(n, n-2, 2)$ if and only if it has exactly one 2-edge and the rest are 1-edges.*

Proof. Due to Theorem 13, an $(n, n-2, 2)$ test set has gain 2. Hence all tests are $\leq$ 2-edges, and some 2-edge exists. Any two disjoint 2-edges exceed gain 2, but any two crossing 2-edges form an $(n, 2)$-MTC, which is excluded by Theorem 13. Thus an $(n, n-2, 2)$ test set contains exactly one 2-edge. □

7 Computational Complexity of (n,n-k,2) Test Sets

Recognizing (n, t, r) test sets is NP-hard for any fixed r, due to a simple reduction from the test cover problem: Given an $(n, t, 1)$ test set $\mathcal{T}$, we construct an (n^r, rt, r) test set denoted $\mathcal{T}^r$ recursively as follows. $\mathcal{T}^1$ is a copy of $\mathcal{T}$, and a test set $\mathcal{T}^i$, $i > 1$, is a copy of $\mathcal{T}$ where each vertex is replaced with a copy of $\mathcal{T}^{i-1}$. The same reduction also lifts W[2]-hardness of $(n, \log_2 n + k, 1)$ [9] to any fixed r. In this section, however, we focus on $(n, n-k, 2)$ test sets. As we have seen, the gain is of central importance for recognizing them, but as for the complexity we have to make another negative observation:

Theorem 17. *Computing the gain of a hypergraph is NP-hard. Consequently, the problem of recognizing whether a test set is $(n, n-k, 2)$ is NP-hard, too.*

Proof. We give a simple reduction from the Set Cover problem (covering all vertices by a minimum number of edges), which is one of the classic NP-hard problems in [23]. Take a hypergraph $\mathcal{H}_1$ as an instance of Set Cover. Without loss of generality, each of the n vertices of $\mathcal{H}_1$ belongs to some edge. Let $\mathcal{H}_2$ be the hypergraph obtained from $\mathcal{H}_1$ by doubling the vertices: for every vertex v we create a copy v' that belongs to precisely the same edges as v. Furthermore, we add another vertex x to $\mathcal{H}_2$.

We show that $\mathcal{H}_1$ has a set cover with at most s sets if and only if $\mathcal{H}_2$ has a gain at least g, where $g + s = 2n + 1$. Once this equivalence is established, the last assertion follows from the characterization in Theorem 13.

Consider any set cover in $\mathcal{H}_1$, consisting of s sets. After doubling the vertices, the corresponding set family in $\mathcal{H}_1$ covers $2n$ vertices (all but the extra vertex x). Due to Definition 7, $\mathcal{H}_2$ has a gain $2n - s + 1$ or larger. The extra vertex is only needed to avoid covering all vertices of $\mathcal{H}_2$.

Conversely, let g be the gain of $\mathcal{H}_2$, and let $\mathcal{S}$ be any gain certificate, that is, a set of edges with $|\bigcup \mathcal{S}| = s + g - 1$, where s is the number of edges in $\mathcal{S}$. Define $C := \bigcup \mathcal{S}$. If C leaves out some v and v', then we can add one edge to $\mathcal{S}$ that contains v and v', which contradicts the gain g. Hence C is the entire set of vertices except x, and the corresponding set family in $\mathcal{H}_1$ is a set cover with s sets. Since $\mathcal{S}$ is a gain certificate, we have $2n = s + g - 1$, that is, $g + s = 2n + 1$. □

Due to NP-hardness it is meaningful to aim for a parameterized algorithm to compute the gain g. The problem looks related to the k-Partial Set Cover problem which asks to cover at least k vertices by a minimum number of edges and is solvable in $2^{O(k)}nm \log n$ time in hypergraphs with n vertices and m edges [4]. However, an attempt to apply this result to the gain computation fails for subtle reasons. Instead we get fixed-parameter tractability directly, by kernelization techniques. The following theorem addresses not only the gain but the combined problem to be solved according to Theorem 13. We use the standard O^* notation that suppresses polynomial factors and displays only the dependency of the time bound on the parameter.

Theorem 18. *The problem of recognizing whether a test set $\mathcal{T}$ is $(n, n-k, 2)$ is fixed-parameter tractable in the parameter k. More specifically, we can decide it, and in the affirmative case we can also construct a two-round strategy, both in $O^*(2^{4k^2+2k\log_2(2k)+2k})$ time.*

Proof. We check whether $\mathcal{T}$ satisfies the conditions from Theorem 13. Let g be the gain of $\mathcal{T}$ (yet unknown to the algorithm).

First we construct some $\mathcal{C} \subseteq \mathcal{T}$ and $C = \bigcup \mathcal{C}$ as follows. Starting from $\mathcal{C} := \emptyset$ and $C := \emptyset$ we add some edge to $\mathcal{C}$ that increases $c = |C|$ by at least 2, as long as possible. Since g is the gain, we can do this at most $g - 1$ times, and $c \leq 2g - 2$. For the final C we have $|E \setminus C| \leq 1$ for all edges E. This property also implies $g \leq c + 1$.

Next we extract a subset $\mathcal{T}' \subseteq \mathcal{T}$ as follows. Let D be any fixed subset $D \subseteq C$. We consider all edges E of the form $E = D \cup \{v\}$, $v \notin C$. If at most $2c$ such edges exist, we keep them in $\mathcal{T}'$, otherwise we keep only $2c$ of them in $\mathcal{T}'$, arbitrarily chosen.

Let $\mathcal{S}$ be a minimal certificate for the gain g of $\mathcal{T}$, with s edges. Remember $|\bigcup \mathcal{S}| = s+g-1$. Lemma 9 gives that $s \leq g-1 \leq c$. Thus $|\bigcup \mathcal{S}| = s+g-1 \leq 2c$. Assume that $\mathcal{S}$ contains some edge $E = D \cup \{v\}$ which was thrown out from $\mathcal{T}'$. By construction, there are still $2c$ edges of the form $D \cup \{u\}$ (with our fixed $D \subseteq C$) in $\mathcal{T}'$. Since $|\bigcup \mathcal{S}| \leq 2c$, at most $2c - 1$ of the vertices u belong to other edges of $\mathcal{S}$. We take an unused vertex u and replace $D \cup \{v\}$ with $D \cup \{u\}$ in $\mathcal{S}$. Note that still $|\bigcup \mathcal{S}| = s + g - 1$ after the exchange. Iterating this procedure we eventually get some $\mathcal{S} \subseteq \mathcal{T}'$ with $|\bigcup \mathcal{S}| = s+g-1$ (and $s \leq c$). It follows that $\mathcal{T}'$ has gain g. We perform this procedure for all subsets $D \subset C$ in any order.

Thus, in order to compute g, it suffices to compute the gain of the smaller hypergraph $\mathcal{T}'$, and for this we only have to examine sets $\mathcal{S}$ of at most c edges and take the largest $|\bigcup \mathcal{S}|-s+1$. The hypergraph $\mathcal{T}'$ is constructed in $O^*(2^c)$ time and retains at most $2c \cdot 2^c < 2^{c+\log_2 c+1}$ edges, thus brute-force examination can be done in $O^*(2^{c^2+c\log_2 c+c})$ time. With $s \leq c < 2g$ we get the time $O^*(2^{4g^2+2g\log_2(2g)+2g})$.

Suppose in the following that $g = k$, that is, the gain equals our prescribed k. Since the brute-force step examines all combinations of at most c edges in $\mathcal{T}'$, and $k = g \leq c + 1$ (see above), we also find among them a selection $\mathcal{S}$ of $s \leq k - 1$ edges whose union is an $(s+k-1)$-set. Due to Theorem 13 such an $\mathcal{S}$ exists, and the test strategy can test $\mathcal{S}$ in round 1, followed by a trivial test cover in round 2.

It remains to check that $\mathcal{T}$ is free of (n,k)-MTC. We do not use the algorithm from [9] whose time bound grows very fast with k. In our case we leverage the strong additional information that the gain is k. Since an (n,k)-MTC is a set $\mathcal{S}$ of $s \le 2k$ edges inducing $s+k$ or more classes, and the gain is k, the only possibility is that $|\bigcup\mathcal{S}| = s+k-1$, and each vertex in $\bigcup\mathcal{S}$ is a class of its own. (The $(s+k)$-th class is $\overline{\bigcup\mathcal{S}}$.)

We extract another subset $\mathcal{T}'' \subseteq \mathcal{T}$ similarly to $\mathcal{T}'$, but this time we keep $2k$ (rather than $2c$) edges that intersect C in the same set $D \subseteq C$, for each D. Now let $\mathcal{S}$ be any set of $s \le 2k$ edges of $\mathcal{T}$ with the aforementioned properties of an (n,k)-MTC. Using the same transformation as before, we can make $\mathcal{S}$ a subset of $\mathcal{T}''$, moreover, the transformed hypergraph $\mathcal{S}$ is isomorphic to the original $\mathcal{S}$. (Note in particular that $|\bigcup\mathcal{S}|$ cannot grow by our edge exchanges, since the gain is only k.) Due to isomorphism, the transformation also preserves the property that each vertex in $\bigcup\mathcal{S}$ is a class of its own. This shows: If $\mathcal{T}$ has an (n,k)-MTC, then already $\mathcal{T}''$ has some.

Thus it suffices to examine all sets of $s \le 2k$ edges of $\mathcal{T}''$. Similarly as above, $\mathcal{T}''$ is constructed in $O^*(2^c)$ time and has at most $2k \cdot 2^c < 2^{c+\log_2 k+1}$ edges. Brute-force search therefore needs $O^*(2^{2kc+2k\log_2 k+2k})$ time, and by $c < 2k$ this is again $O^*(2^{4k^2+2k\log_2(2k)+2k})$. □

Actually we can compute the gain g much faster. The following algorithm is, however, result-wise weaker: It only determines the optimal test number t for two rounds, but it does not check whether one round would be sufficient for t tests. It remains open whether we can also accelerate the exclusion of (n,g)-MTC.

Theorem 19. *The gain g of a hypergraph (and hence the optimal test number $n-g$ for two rounds, if $n \ge 3g-2$), can be computed in $O^*(2^{g^2+(g/2)\log_2(2g)+2g})$ time.*

Proof. In the given test set $\mathcal{T}$ we construct $\mathcal{T}' \subseteq \mathcal{T}$ as before, but we search for a gain certificate more economically. According to Lemma 9 there exists a minimal certificate $\mathcal{S}$, and it has an L-maximal permutation where all expansions are at least 2. Let $\mathcal{P}$ be the "prefix" consisting of all edges of the ordered $\mathcal{S}$ with expansion at least 3. For brevity let $\mathcal{R} := \mathcal{T} \downarrow \overline{\bigcup\mathcal{P}}$. Then $\mathcal{R}$ has only ≤ 2-edges, since any ≥ 3-edge allows to append another edge with expansion at least 3 to $\mathcal{P}$, contradicting L-maximality of the ordered $\mathcal{S}$. Since the expansions of all further edges in $\mathcal{S}$ after $\mathcal{P}$ are 2, the corresponding 2-edges in $\mathcal{R}$ form a matching. Moreover, this matching is maximum, otherwise $\mathcal{S}$ would not reach the gain.

Thus we can limit brute-force search to the prefixes $\mathcal{P}$, and then compute any maximum matching in $\mathcal{R}$ for each $\mathcal{P}$ (using the above notations). Since $\mathcal{T}$ has gain g, the size of every $\mathcal{P}$ is obviously at most $g/2$. We do not know g in advance (the aim of the algorithm is to compute this number!), but we can loop through $j = 1,2,3\dots$ and enumerate all possible prefixes $\mathcal{P}$ of length j. For every prefix $\mathcal{P}$ that adheres to the above structure (expansions are at least 3 and $\mathcal{R}$ has only ≤ 2-edges), we also compute a maximum matching. As soon as the next j does not yield new prefixes, further increase of j cannot yield new prefixes either. Thus we will stop at some $j \le g/2$, and a gain certificate is any obtained set $\mathcal{S}$ with largest $|\bigcup\mathcal{S}| - s + 1$.

As is well known, maximum matchings can be computed in polynomial time [26]. By the earlier analysis, $\mathcal{T}'$ is constructed in $O^*(2^c)$ time where $c < 2g$, and has at most $2^{c+\log_2 c+1}$ edges. The last exponent is now multiplied with only $g/2$, which gives the assertion. □

8 Tree Hypergraphs

Due to Corollary 4, tree hypergraphs are the worst test sets for one round, as they need exactly $n-1$ tests. Before we discuss testing in two rounds, we relate tree hypergraphs to usual trees. To avoid confusion with vertices and edges of hypergraphs we speak of *nodes* and *arcs* in trees.

Consider any separating system being a tree hypergraph. Recall that any two edges are either disjoint or in subset relation. Let $E_1, \ldots, E_m$ denote the edges that are maximal with respect to the subset relation. Clearly, they have the following properties: The E_i's are pairwise disjoint, at most one vertex v is outside $E_1 \cup \ldots \cup E_m$ (since we have a separating system), and every further edge is a subset of exactly one of the E_i. This defines a partitioning of the edge set, and we can recursively partition the edge set of the hypergraph restricted to each E_i in the same way.

This recursive partitioning can also be described as a rooted tree, recursively constructed as follows. We create a root, and for each E_i append an arc to it. If a vertex v as specified above exists, then we assign it to the root node. (Otherwise, no vertex is assigned to the root node.) For $i = 1, \ldots, m$, the other node of the i-th arc becomes the root of the tree for the hypergraph restricted to E_i.

Note that the vertices are assigned to distinct nodes, but there may be (non-leaf) nodes without assigned vertices. We call this tree the *separating tree*, due to the following property: Every arc represents an edge, i.e., a test. If the test is positive, the remaining candidates for the target are the vertices assigned to nodes of the subtree below this arc, and if the test is negative, the remaining candidates for the target are the vertices assigned to the rest of the tree. These are exactly the two connected components of the tree after removal of the arc.

Theorem 20. *An optimal two-round strategy for a given tree hypergraph can be computed in $O(n \log^2 n)$ time, provided that its separating tree is given.*

Proof. Let T denote the separating tree, and let F be the set of arcs of T tested in round 1. The test outcomes specify the connected component C of $T - F$ where the target is hidden. Moreover, the remaining problem instance for round 2 is described by the subtree C of T. Recall that the test number in one round is just the number of vertices minus 1. Thus, we have to minimize $|F| + b$, where b is the maximum number of vertices in a connected component of $T - F$.

For any node v let $T(v)$ denote the subtree rooted at v. Let us fix b, the budget for the maximum number of tests permitted in round 2. Assume that we have already decided which arcs of $T(v)$ we put in F; this is vacuously true for the leaves v of T. Then let $h(v)$ be the number of vertices in the connected component of $T(v) - F$ at the root v, called the root component. Note that $h(v) \leq b + 1$. Now let v be any node where the arcs in F are already decided in $T(u)$ for all children u of v. It remains to decide which arcs vu be added to F. Every such arc finalizes a connected component of $T-F$. For the children u with $vu \notin F$, the root components of the $T(u)$, node v, and the arcs vu together form the root component of $T(v)$, and $h(v)$ is the sum of their $h(u)$, plus 1 if v represents a vertex. We must put enough new arcs in F to achieve $h(v) \leq b+1$. Clearly, the number of new arcs in F is minimized if we decide $vu \in F$ successively for children u with the largest $h(u)$, until $h(v) \leq b+1$ is reached. An exchange argument implies the correctness of this greedy procedure: The only reason for adding more arcs vu to F (and making $h(v)$ even smaller) is to prevent the current root component from exceeding the budget later on. But instead of adding any further arc vu to F we can add the arc between v and the parent of v later on.

The time is $O(n \log n)$ for every budget b, where the $\log n$ factor is added only for sorting the children by their $h(u)$ values. Let $t(b)$ be the optimal total number of tests when the budget is b. Clearly, $t(b)$ is a monotone non-increasing function. Consider the smallest b with $t(b) \leq b$. Assume $t(b) < b$ for this b. Then we have found a solution that uses at most $b-1$ tests in total, hence we obtain this value of t already with budget $b-1$ or smaller. Thus $t(b-1) \leq b-1$, contradicting the minimality of b. This shows $t(b) = b$. Furthermore, if a solution actually uses a larger budget for round 2, it needs more than b tests is total and cannot be optimal. We conclude that $t(b)$ is the optimal test number overall. We find this specific b by binary search on the budget values. $\square$

Open problems include an improved time bound for $r=2$ and the time complexity for any fixed $r > 2$.

9 Further Research

We state some open problems in addition to those mentioned in the paper. First of all: What are the possible ranges of round numbers r (and test numbers t) for which a given hypergraph $\mathcal{H}$) is (n,t,r)? That is, which subsets of numbers r can appear in the sequence $1,2,\ldots,r(\mathcal{H})$?

Another, more technical question concerns a worst-case time bound (presumably exponential in n) for solving arbitrary instances $\mathcal{T}$ on n elements exactly. A promising approach is dynamic programming on the partitionings. Here we sketch the ideas. We create a directed graph D whose nodes represent partitionings of the n-set (the induced classes plus marks at the already discarded elements). At every node p of D we loop through the edges of $\mathcal{T}$ and decide either to add some test to the current round and refine the partitioning, or to ask the current edges and learn the induced class containing the target. We insert arcs from p to the corresponding nodes. Then the nodes of D are evaluated bottom-up and annotated with the smallest t for each r, obtained by simple min-max calculations. The number of partitionings of an n-set is the nth Bell number. Bounds smaller than n^n are known, such as $(0.792n/\ln(n+1))^n$ [3]. Already for $r=1$ this approach is more efficient than naive examination of all combinations of $t \geq \log_2 n$ edges, and it also works for all r. Furthermore, in general not all partitionings appear in D.

Acknowledgment

The author would like to thank the anonymous reviewers for their very careful and constructive comments.

References

[1] Ajtai, M., Komlós, J., Szemerédi, E.: An $O(n \log n)$ sorting network. In: Johnson, D.S. et al. (eds.) STOC 1983. ACM, pp. 1–9 (1983)

[2] Basavaraju, M., Francis, M.C., Ramanujan, M.S., Saurabh, S.: Partially polynomial kernels for set cover and test cover. In: Seth, A., Vishnoi, N.K. (eds.) FSTTCS 2013. LIPIcs, vol. 24, pp. 67–78, Dagstuhl (2013)

[3] Berend, D., Tassa, T.: Improved bounds on Bell numbers and on moments of sums of random Vvariables. Prob. Math. Stat. 30, 185–205 (2010)

[4] Bläser, M.: Computing small partial coverings. Info. Proc. Letters 85, 327–331 (2003)

[5] Bundala, D., Závodný, J.: Optimal sorting networks. In: Horia Dediu, A. et al. (eds.) LATA 2014. LNCS, vol. 8370, pp. 236–247, Springer, Heidelberg (2014)

[6] Cheng, Y., Du, D.Z., Zheng, F.: A new strongly competitive group testing algorithm with small sequentiality. Annals Oper. Res. 229, 265–286 (2015)

[7] Cicalese, F.: Tree search problems. In: Problem Booklet of the 5th Emléktábla Workshop on Combinatorial Search 2013, available on `http://www.renyi.hu/∼emlektab/`

[8] Crowston, R., Gutin, G., Jones, M., Muciaccia, G., Yeo, A.: Parameterizations of test cover with bounded test sizes. Algorithmica 74, 367–384 (2016)

[9] Crowston, R., Gutin, G., Jones, M., Saurabh, S., Yeo, A.: Parameterized study of the test cover problem. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) MFCS 2012. LNCS, vol. 7464, pp. 283–295, Springer, Heidelberg (2012)

[10] Cui, P.: A Tighter analysis of set cover greedy algorithm for test set. In: Chen, B., Paterson, M., Zhang, G. (eds.) ESCAPE 2007. LNCS, vol. 4614, pp. 24–35, Springer, Heidelberg (2007)

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

[12] de Bontridder, K.M.J., Halldórsson, B.V., Halldórsson, M.M., Hurkens, C.A.J., Lenstra, J.K., Ravi, R., Stougie, L.: Approximation algorithms for the test cover problem. Math. Progr. Series B 98, 477–491 (2003)

[13] de Bontridder, K.M.J., Lageweg, B.J., Lenstra, J.K., Orlin, J.B., Stougie, L.: Branch-and-bound algorithms for the test cover problem. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 223–233, Springer, Heidelberg (2002)

[14] D'yachkov, A.G., Rykov, V.V.: Bounds on the length of disjunctive codes. Probl. Info. Transm. 18, 166–171 (1982)

[15] D'yachkov, A.G., Rykov, V.V.: A Survey on superimposed code theory. Probl. Control Info. Theory 12, 229–242 (1983)

[16] D'yachkov, A.G., Vorobyev, I.V., Polyanskii, N.A., Shchukin, V.Y.: Bounds on the rate of superimposed codes. In: Høst-Madsen, A. A, Kavcic, A., Veeravalli, V.V. (eds.) IEEE Int. Symp. Info. Theory, pp. 2341–2345, IEEE (2014)

[17] 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)

[18] Fahle, T., Tiemann, K.: A faster branch-and-bound algorithm for the test-cover problem based on set-covering techniques. ACM J. Experim. Algor. 11 (2006)

[19] Fernau, H., Raible, D.: A parameterized perspective on packing paths of length two. J. Comb. Optim. 18, 319–341 (2009)

[20] Gerbner, D., Vizer, M.: Rounds in a combinatorial search problem, arXiv 1611.10133 (2016)

[21] Goodrich, M.: Zig-zag Sort: A simple deterministic data-oblivious sorting algorithm running in O(n log n) time. In: Shmoys, D.B. (ed.) ACM STOC 2014, pp. 694–693, ACM (2014)

[22] Gutin, G., Muciaccia, G., Yeo A.: (Non-)existence of polynomial kernels for the test cover problem. Info. Proc. Letters 113, 123–126 (2013)

[23] Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computation, The IBM Research Symp. Series, pp. 85–103, Plenum Press (1972)

[24] Katona, G.O.H.: Finding at least one excellent element in two rounds. J. Statist. Planning and Inference, 141, 2946–2952 (2011)

[25] Lam, T.W., Yue, F.L.: Optimal edge ranking of trees in linear time. In: Karloff, H.J. (ed.) SODA 1998, pp. 436–445, ACM-SIAM (1998)

[26] Micali, S., Vazirani, V.V.: An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. FOCS 1980, pp. 17–27, IEEE (1980)

[27] Paterson, M.S.: Improved sorting networks with O(log N) depth. Algorithmica 5, 75–92 (1990)

[28] Wiener, G.: Rounds in combinatorial search. Algorithmica 67, 315–323 (2013)