Randomized Adaptive Test Cover

Peter Damaschke

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

Abstract. In a general combinatorial search problem with binary tests we are given a set of elements and a hypergraph of possible tests, and the goal is to find an unknown target element using a minimum number of tests. We explore the expected test number of randomized strategies. We obtain several general results on the ratio of the expected and worst-case deterministic test number, as well as complexity results for hypergraphs of small rank, and we state some open problems.

Keywords: combinatorial search, randomization, game theory, LP duality, fractional graph theory

1 Introduction

A hypergraph $\mathcal{H}$ is a set U of n elements (vertices) equipped with a family of subsets called the *edges*. We consider a search problem on hypergraphs, therefore we also call the edges *tests*: One unknown element $u \in U$ is the *target*. A searcher can apply any tests T from $\mathcal{H}$. The test answers *positive* if $u \in T$, and *negative* else. The searcher aims to identify u efficiently from the outcomes of some tests from $\mathcal{H}$. The primary goal is to minimize the number of tests. We will silently assume that $\mathcal{H}$ is *separating*, that is, no two targets cause the same outcomes of all tests (otherwise it would be impossible to distinguish them).

Combinatorial search has applications, e.g., in biological testing [2, 3]. A number of more specific, classic combinatorial search problems can be formulated in the above way, perhaps the foremost example is the group testing problem [7] which found various applications. Note that even problems like sorting by comparisons fit in this framework. (There the target is a sorted sequence of numbers, the elements are all permutations, and a test finds out the relation between two numbers.) One may also think of $\mathcal{H}$ as a system of binary attributes of objects, and then an efficient test strategy is also a compact *classification system*.

A search strategy can work in *rounds*, where all tests in a round are done in parallel, without waiting for each other's outcomes. An *adaptive* strategy performs only one test per round. In a *deterministic* strategy, the choice of tests for each round is uniquely determined by the outcomes of earlier tests. In a *randomized* strategy, the choice of tests for each round can, additionally, depend on random decisions. We stress that the tests still behave deterministically; here

we assume neither random errors nor target probability distributions, only the searcher's choice of tests may be randomized. Also note that deterministic strategies are by definition a special case of randomized strategies, they just do not use the possibility to take random decisions. A deterministic strategy is *optimal* among all deterministic strategies if it minimizes the worst-case number of tests, and a randomized strategy is optimal if it minimizes the worst-case *expected* number of tests, where the worst case refers to maximization over all targets. For clarity we give formal definitions of the test numbers.

Definition 1. *For a separating hypergraph $\mathcal{H}$ let $Det(\mathcal{H})$ and $Rand(\mathcal{H})$ denote the set of all deterministic and randomized search strategies, respectively, on $\mathcal{H}$. Note that $Det(\mathcal{H}) \subset Rand(\mathcal{H})$. For $A \in Rand(\mathcal{H})$ and an element $u \in U$, let $t_A(\mathcal{H}, u)$ denote the expected number of tests done by A if u is the target. (If $A \in Det(\mathcal{H})$, this "expected" number is simply the deterministic number of tests.) The test number of A is $t_A(\mathcal{H}) := \max_{u \in U} t_A(\mathcal{H}, u)$. We define the optimum test numbers as $t_{rand}(\mathcal{H}) := \min_{A \in Rand(\mathcal{H})} t_A(\mathcal{H})$ and $t_{det}(\mathcal{H}) := \min_{A \in Det(\mathcal{H})} t_A(\mathcal{H})$. We may also limit the number r of rounds of strategies A, in which case we write $t_{rand,r}(\mathcal{H})$ and $t_{det,r}(\mathcal{H})$.*

Trivially we have $t_{rand,r}(\mathcal{H}) \leq t_{det,r}(\mathcal{H})$. As we will see, the randomized test number can be significantly smaller, however, note that $t_{rand,1}(\mathcal{H}) = t_{det,1}(\mathcal{H})$, since $t_A(\mathcal{H}, u)$ is the same for all $u \in U$ if A has only one round. In other words, meaningful randomized strategies need at least two rounds.

The case $r = 1$ (thus restricted to the deterministic setting) is well known as the *test cover* problem and can be rephrased as follows: Given a separating hypergraph, find a smallest subset of the edges of $\mathcal{H}$ that still form a separating hypergraph. The complexity of the test cover problem has been intensively studied [1, 3, 5, 6, 8, 10], whereas very little is known for $r > 1$; see [12] for some combinatorial results. The *rank* of a hypergraph $\mathcal{H}$ is the maximum size of its edges. The test cover problem with fixed rank found independent interest, since already this restricted case has practical relevance [2, 4, 9].

To the best of our knowledge, there is no study of the search problem *in full generality and in the randomized setting* so far, although randomization has well been used in many specific search problems. E.g., Quicksort is a famous sorting algorithm, and randomized constructions have been extensively studied for group testing (however we cannot possibly give a survey here).

Contributions. We focus on adaptive testing. An obvious question is how much the test number can benefit from randomization. That is, we study the ratio $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H})$. As a preparation we express the search problem as a matrix game and linear program, which implies some simple lower bounds on $t_{rand}(\mathcal{H})$. We use them to show that $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H})$ is, essentially, at least 2 for large hypergraphs of fixed rank, and the ratio can be up to 4 in certain hypergraphs. By a certain composition of small hypergraphs we get examples where the ratio is away from 1 even when the test number is logarithmic, i.e., close to the information-theoretic lower bound. The largest possible ratio remains an open

problem. For rank-2 hypergraphs we also relate the search problem to some standard graph problems. They do not exactly correspond to the search problem but enable a $\frac{7}{6}$-approximation algorithm for the randomized test number. This approximation is based on the primal-dual method and certain half-integral solutions to the minimum fractional edge cover problem. While the actual complexity of computing $t_{rand}(\mathcal{H})$ remains open even in the rank-2 case, we show that finding optimal deterministic strategies is NP-hard already for rank 3.

2 Game-Theoretic Interpretation and Lower Bounds

Any randomized search strategy on a hypergraph $\mathcal{H}$ can be viewed as a probability distribution on the (finite) set of the deterministic strategies on $\mathcal{H}$, also called a *mixed strategy*. This allows us to apply the classic theory of zero-sum games. The searcher applies a mixed strategy as above, and an adversary plays a mixed strategy, too, which is a probability distribution on U specifying the probability of each element being the target. We also refer to it as a *target distribution*. By von Neumann's minimax theorem, a special case of LP strong duality, we get:

Proposition 1. *There exists a pair of optimal mixed strategies such that the player does an expected number of at most $t_{rand}(\mathcal{H})$ tests whatever the target is, and every deterministic strategy needs an expected number of at least $t_{rand}(\mathcal{H})$ tests. Only deterministic strategies that attain this expected value can be involved with nonzero probability in an optimal mixed search strategy.*

The optimal strategies can be computed by a linear program (LP), with the caveat that the number of deterministic strategies to consider may not be polynomial in n. In the following note that any tests T and $U \setminus T$ are equivalent.

Proposition 2. *The following lower bounds hold.*
Averaging bound: $t_{rand}(\mathcal{H}) \geq \min_{A \in Det(\mathcal{H})} \frac{1}{n} \sum_{u \in U} t_A(\mathcal{H}, u)$*; in words: the best average deterministic test number lower-bounds the randomized test number.*
Set cover lower bound: *For any fixed $u \in U$, replace all tests $T \ni u$ with their complements. Then $t_{rand}(\mathcal{H})$ is at least the size of a smallest set cover of $U \setminus \{u\}$.*

Proof. By Proposition 1, any target distribution yields a lower bound on $t_{rand}(\mathcal{H})$. The uniform target distribution yields the averaging bound. The target distribution that concentrates probability 1 on some fixed $u \in U$ yields the set cover lower bound, since every non-target must occur in some negative test. □

Definition 2. *For some arbitrary but fixed order of U, we call the vector of the n values $t_A(\mathcal{H}, u)$ the* test number vector *of strategy A on $\mathcal{H}$.*

Hence the test number vector of any randomized strategy is a convex linear combination of test number vectors of some deterministic strategies.

A deterministic strategy can be viewed as a *strategy tree* with n leaves for the n elements. Every inner node is marked with the set of tests applied in a round. Depending on their outcomes, the searcher is sent to a child node for the next round. The depth of the tree is the maximum number of rounds.

Proposition 3. *Let $\mathcal{H}$ be a hypergraph on $n = 2^k + m$ elements, where $2^k \leq n$ is the largest power of 2 not exceeding n. Then we have:*
$t_{rand}(\mathcal{H}) \geq \frac{1}{n}((n-2m)k + 2m(k+1)) = k + \frac{2m}{n}$.

Proof. (Sketch.) Since parallel tests could also be done sequentially, the test number vector of any deterministic strategy is the vector of distances of the n leaves to the root in some *binary* tree. Due to simple exchange arguments, among the binary trees with a fixed number n of leaves, the average test number is minimized if $n-2m$ numbers are k and $2m$ numbers are $k+1$. Together with the averaging bound in Proposition 2 this yields the assertion. □

This bound is particularly useful in practical computations of optimal strategies for small instances $\mathcal{H}$: First we check whether $\mathcal{H}$ allows deterministic strategies A with $k + \frac{2m}{n}$ tests on average, since (by Proposition 1) only such A can build a randomized strategy with $k + \frac{2m}{n}$ expected tests. For small $\mathcal{H}$ there are not so many possible strategy trees. We enumerate them and solve the LP that balances the expected test numbers for all targets. If the LP has no balanced solution, then we know $t_{rand}(\mathcal{H}) > k + \frac{2m}{n}$. Thus we would next include deterministic strategies A where $\frac{1}{n}\sum_{u \in U} t_A(\mathcal{H}, u)$ is by $\frac{1}{n}$ larger, solve the extended LP, and so on. (Calculation examples are omitted due to space limitations.)

3 Some Structural Lemmas

Lemma 1. *Let $\mathcal{H}$ be a hypergraph and u an element such that $\{u\}$ is not an edge. Let $\mathcal{H}_u$ be the hypergraph $\mathcal{H}$ with the edge $\{u\}$ inserted. Then $t_{det}(\mathcal{H}_u) = t_{det}(\mathcal{H})$.*

Proof. Trivially, $t_{det}(\mathcal{H}_u) \leq t_{det}(\mathcal{H})$. To show that $t_{det}(\mathcal{H})$ is not strictly larger, we consider an optimal deterministic strategy A_u for $\mathcal{H}_u$. Let A be the strategy for $\mathcal{H}$ that mimics A_u, and whenever A_u wants to test $\{u\}$, strategy A skips this non-existing test and continues as A_u would do if this test were negative. Strategy A can be incomplete in the sense that it may not find the target. Below we discuss how to complete it, using $t_{det}(\mathcal{H}_u)$ tests in the worst case.

Consider any leaf ℓ of the strategy tree of A. If the strategy has not skipped $\{u\}$ on the path to ℓ, then A has behaved as A_u, thus a target is identified. The other case is that A has skipped $\{u\}$ on the path to ℓ. The path to ℓ is identical to the corresponding path in A_u, except that test $\{u\}$ is not done and a negative outcome assumed. Moreover, A_u would have identified some target v on this path. The only missing information is now that u is in fact negative. It follows that at most two candidates, u and v, remain at ℓ. Since $\mathcal{H}$ is separating, we can append any test to distinguish u and v. Since one test was skipped, the path to ℓ in A is strictly shorter than the one in A_u, hence adding a test at the end does not increase the test number compared to A_u. □

Lemma 2. *Let $\mathcal{H}$ be a hypergraph containing an edge $\{u\}$. Against any deterministic searcher, there is an optimal adversary strategy (enforcing at least $t_{det}(\mathcal{H})$ tests) that gives a negative answer whenever $\{u\}$ is tested.*

Proof. When the searcher tests $\{u\}$ and the answer is positive, then u is identified as the target, and $\{u\}$ was the last test. Hence the adversary cannot miss a longest path in the strategy tree by giving a negative answer instead. □

Definition 3. *We define the* composition $\mathcal{H} \triangleright \mathcal{J}$ *of hypergraphs* $\mathcal{H}$ *and* $\mathcal{J}$ *as follows. Let* $U = \{u_1, \ldots, u_n\}$ *and* V *be the vertex sets of* $\mathcal{H}$ *and* $\mathcal{J}$*, respectively. We replace every element* $u_i \in U$ *with a copy of* $\mathcal{J}$*, denoted by* $\mathcal{J}_i$*, on a vertex set* V_i*. Every edge* E *of* $\mathcal{H}$ *is kept in* $\mathcal{H} \triangleright \mathcal{J}$*, by replacing the elements: We define* E *in* $\mathcal{H} \triangleright \mathcal{J}$ *as the union of those* V_i *with* $u_i \in E$ *in* $\mathcal{H}$*.*

Informally, we substitute $\mathcal{J}$ into every element of $\mathcal{H}$, or $\mathcal{H} \triangleright \mathcal{J}$ models a hierarchical classification where $\mathcal{H}$ gives a coarse classification refined by $\mathcal{J}$. While the following Lemma is not too surprising, the proof becomes somewhat tricky. Besides the previous Lemmas it uses a technique that often simplifies lower-bound proofs for search problems: An adversary may reveal more information than the searcher has asked for. Since this only helps the searcher, any lower bound obtained in this way is also a lower bound for the original search problem.

Lemma 3. *The deterministic test number of the composition of any two hypergraphs is additive:* $t_{det}(\mathcal{H} \triangleright \mathcal{J}) = t_{det}(\mathcal{H}) + t_{det}(\mathcal{J})$.

Proof. Subadditivity is easy to see: In order to search $\mathcal{H} \triangleright \mathcal{J}$ one can first run an optimal strategy on $\mathcal{H}$ to determine the copy of $\mathcal{J}$ containing the target, followed by an optimal strategy for $\mathcal{J}$ applied to this copy. The reverse $t_{det}(\mathcal{H} \triangleright \mathcal{J}) \geq t_{det}(\mathcal{H}) + t_{det}(\mathcal{J})$ is much less obvious, as the searcher may interleave tests in $\mathcal{H}$ and in the copies of $\mathcal{J}$. Define $\mathcal{H}_1$ as the hypergraph obtained from $\mathcal{H}$ by inserting all singleton edges $\{u\}$ that are not yet in $\mathcal{H}$. We claim:
$t_{det}(\mathcal{H} \triangleright \mathcal{J}) \geq t_{det}(\mathcal{H}_1 \triangleright \mathcal{J}) \geq t_{det}(\mathcal{H}_1) + t_{det}(\mathcal{J}) = t_{det}(\mathcal{H}) + t_{det}(\mathcal{J})$.
The first inequality is trivial, and the equation holds due to Lemma 1. To show $t_{det}(\mathcal{H}_1 \triangleright \mathcal{J}) \geq t_{det}(\mathcal{H}_1) + t_{det}(\mathcal{J})$ we describe an adversary on $\mathcal{H}_1 \triangleright \mathcal{J}$ forcing the searcher to do at least $t_{det}(\mathcal{H}_1) + t_{det}(\mathcal{J})$ tests. We follow an optimal adversary strategy on $\mathcal{H}_1$ with the property from Lemma 2, as long as the searcher keeps on testing edges of $\mathcal{H}_1$. Whenever the searcher tests an edge from a copy of $\mathcal{J}$, say $\mathcal{J}_i$, although the copy containing the target is not yet identified, we answer negatively and also reveal that the target is not in V_i at all. This gives the searcher the same information as if she had tested V_i, which equals the edge $\{u_i\}$ of $\mathcal{H}_1$, and received a negative answer. Moreover, this still complies with our optimal adversary strategy on $\mathcal{H}_1$. Thus we have "indirectly forced" the searcher to test only edges of $\mathcal{H}_1$ until the set V_k with the target is identified. Since the adversary has run an optimal strategy, at least $t_{det}(\mathcal{H}_1)$ tests have been done so far. At this moment there still remains the set V_k of candidates, and no tests have been performed in $\mathcal{J}_k$. From this it is clear that the searcher needs $t_{det}(\mathcal{J})$ further tests in the worst case. □

4 The Case of Singleton Tests

The *rank* of a hypergraph $\mathcal{H}$ is the maximum number of elements in an edge. First we briefly settle the case of rank 1. Since $\mathcal{H}$ is separating, at least $n-1$ of its n elements must be singleton edges.

Proposition 4. *Let $\mathcal{H}$ be a hypergraph of rank 1. If $\mathcal{H}$ has exactly $n-1$ edges then $t_{rand}(\mathcal{H}) = n-1$. If $\mathcal{H}$ has n edges then $t_{rand}(\mathcal{H}) = \frac{1}{n}(\sum_{i=1}^{n} i-1) = \frac{n+1}{2} - \frac{1}{n}$.*

Proof. The assertion for $n-1$ edges follows from the set cover lower bound in Proposition 2 and the trivial fact that $n-1$ tests suffice. If $\mathcal{H}$ has n edges then the available test number vectors are all permutations of the following vector: $(1, 2, 3, \ldots, n-3.n-2, n-1.n-1)$. We take one of them and its cyclic shifts, each with probability $\frac{1}{n}$. This yields the claimed test number. Optimality is seen by assigning the target probability $\frac{1}{n}$ to every element. □

The above strategy needs $\log_2 n$ random bits. Using only one random bit we can combine the test number vector $(1, 2, 3, \ldots, n-3.n-2, n-1.n-1)$ and its reverse, each with probability $\frac{1}{2}$. Then the balance is not perfect, still the inner elements have an expected test number $\frac{n+1}{2}$ which is only slightly worse. Note that $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H})$ tends to 2 as n grows. This raises the question is how large $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H})$ can ever be for general hypergraphs. We will address it later. Similarly we ask how large $t_{det,r}(\mathcal{H})/t_{rand,r}(\mathcal{H})$ can ever be.

Back to rank 1, next suppose that only r rounds are permitted. For ease of presentation we state only the asymptotic result. In particular, we assume that r divides n, and the nth test must be done even if $n-1$ tests were negative.

Proposition 5. *Let $\mathcal{H}$ be a hypergraph of rank 1, with all n edges. Then we have $t_{rand}(\mathcal{H}) = \frac{r(r+1)}{2} \cdot \frac{n}{r} \cdot \frac{1}{r} = \frac{n}{2}(1 + \frac{1}{r})$ subject to lower-order terms.*

Proof. (Sketch.) We divide U into r bins of $\frac{n}{r}$ elements, arrange the bins in a cycle, and in each round we test all elements of one bin, following the cyclic order and starting at a random bin. Then every element is tested in each round with the same probability $1/r$, and the expected number of tests is as claimed. To show optimality we take again the uniform target distribution. The expected test number of any deterministic r-round strategy is then uniquely determined by the numbers x_i of elements tested in rounds $i = 1, \ldots, r$, and it amounts to $\frac{1}{n}\sum_{j=1}^{r} x_j(\sum_{i=1}^{j} x_i)$, where $\sum_{i=1}^{r} x_i = n$. By standard methods for multivariate extremal problems with constraints, this expression is minimized if all x_i equal $\frac{n}{r}$, and then it becomes $\frac{r(r+1)}{2} \cdot \frac{n^2}{r^2} \cdot \frac{1}{n} = \frac{n}{2}(1 + \frac{1}{r})$ again. □

5 Deterministic vs. Randomized Test Number

The rank-1 case suggests that $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H})$ might typically be around 2 if the rank is small compared to n. In the following we study this question. We consider hypergraphs $\mathcal{H}$ of arbitrary but fixed rank. Let $\nu := \nu(\mathcal{H})$ be the size of

a minimum *edge cover* in $\mathcal{H}$, that is, a subset of edges that covers all elements in U. Since $\mathcal{H}$ is separating, at most one element $u \in U$ is in no edge, and if such u exists, we define ν to be the size of a minimum edge cover of $U \setminus \{u\}$.

Theorem 1. *Consider all hypergraphs $\mathcal{H}$ of any fixed rank h. There we have $t_{det}(\mathcal{H}) = \nu \pm O(1)$. If the edges of $\mathcal{H}$ cover all elements, then we also have $t_{rand}(\mathcal{H}) < \frac{\nu}{2} + O(1)$. Thus, for any $\epsilon > 0$, all large enough $\mathcal{H}$ without uncovered elements satisfy $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H}) > 2 - \epsilon$.*

Proof. We can test the edges of any fixed minimum edge cover $\mathcal{E}$ in one round. For any test outcomes there remain at most h candidates for the target, and we can trivially distinguish them by fewer than h more tests in a second round. This shows $t_{det,2}(\mathcal{H}) < \nu + h$. Next we argue that even an adaptive deterministic strategy cannot be essentially better in the worst case: All test outcomes could be negative until the tested edges cover all elements. Then we have to do at least $\nu - 1$ tests. In short, $t_{det}(\mathcal{H}) \geq \nu - 1$. From the set cover lower bound in Proposition 2 we also get $t_{rand}(\mathcal{H}) \geq \nu$ if some element is in no edge. Thus, in the following we consider only hypergraphs where the edges cover all elements.

Now, an obvious randomized strategy is to test the edges of $\mathcal{E}$ in random order. More precisely, we may take all cyclic shifts of some fixed order or two reverse orders, with equal probability, just as in the case of rank 1. We argue that every fixed target is found after $\frac{\nu}{2} + h$ expected tests. We assign every element u arbitrarily to some edge $E \in \mathcal{E}$ with $u \in E$, called the designated edge of u. Due to the random order of $\mathcal{E}$, the designated edge of any target u appears after at most $\frac{\nu}{2} + 1$ expected tests, and then fewer than further tests identify u as the target. This yields the assertions and finishes the proof. □

By modifying the proof for r rounds and using the randomized strategy from Section 4, we similarly obtain $t_{det,r}(\mathcal{H})/t_{rand,r}(\mathcal{H}) > \frac{2r}{r+1} - \epsilon$.

One might conjecture that shuffling disjoint edges is already the best use of randomization, which would imply that ratio 2 is the best. But in the end of the proof of Theorem 1 we notice that element u may appear earlier, in some non-designated edge, hence $t_{rand}(\mathcal{H})$ might be smaller in a specific hypergraph. The following example demonstrates that, indeed, the ratio can be up to 4.

Proposition 6. *For any fixed h and $\epsilon > 0$ there exist hypergraphs $\mathcal{H}$ of rank h where $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H}) > \frac{4(h+1)}{(h+3)} - \epsilon$.*

Proof. An h-simplex is the set of all $h + 1$ possible edges of size h in a set of $h + 1$ elements Let $\mathcal{H}$ consist of k disjoint h-simplices. Clearly, $\nu = 2k$. Now we arrange the h-simplices in random order and first test one random edge from every h-simplex. With probability $\frac{h}{h+1}$ we hit the target, and fewer than h further tests identify it. In this case we do an expected number of $\frac{k}{2} + O(1)$ tests (remember that h is fixed). With probability $\frac{1}{h+1}$ we miss the target, but there remains only one candidate in every h-simplex. In this case we test a second edge from every h-simplex to find the target. Thus we do an expected number of $k + \frac{k}{2} + O(1)$ tests in total. Hence the overall expected test number is

$\frac{h}{h+1} \cdot \frac{k}{2} + \frac{1}{h+1} \cdot \frac{3k}{2} + O(1) = \frac{h+3}{2(h+1)}k + O(1)$. Our ratio tends to $\frac{2k \cdot 2(h+1)}{(h+3)k} = \frac{4(h+1)}{(h+3)}$ for large k. □

Problem: Is $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H}) < 4$ for all $\mathcal{H}$? Is it bounded by any constant?

Next we look into a different direction. One may conjecture that randomization significantly helps "bad" instances only, where linearly many tests are needed. However we show that $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H})$ can be away from 1 even for "good" hypergraphs where $O(\log n)$ tests are enough.

Theorem 2. *Let $\mathcal{H}$ be any hypergraph with g elements, such that $t_{det}(\mathcal{H}) = t$, and $t_{det}(\mathcal{H})/t_{rand}(\mathcal{H}) \geq c$. Then there exist hypergraphs with arbitrarily large numbers n of elements, such that $t_{det}(\mathcal{L})/t_{rand}(\mathcal{L}) \geq c$ as well, and $t_{det}(\mathcal{L}) = \frac{t}{\log_2 g} \log_2 n$.*

Proof. Let $\mathcal{L}$ be the k-fold composition of $\mathcal{H}$ with itself: $\mathcal{L} := \mathcal{H} \triangleright \ldots \triangleright \mathcal{H}$ (k terms). One may think of $\mathcal{L}$ as a "tree of hypergraphs $\mathcal{H}$" with degree g and depth k. Clearly $\mathcal{L}$ has $n := g^k$ elements. An obvious randomized strategy that goes recursively down this tree needs at most $k \cdot t_{rand}(\mathcal{H})$ expected tests for every target, by linearity of expectation. Hence $t_{rand}(\mathcal{L}) \leq k \cdot t_{rand}(\mathcal{H})$. Lemma 3 inductively applied to k factors yields $t_{det}(\mathcal{L}) = k \cdot t_{det}(\mathcal{H})$. Furthermore, note that $k = \frac{\log_2 n}{\log_2 g}$. Together these bounds imply the assertions. □

To give an example, let $\mathcal{H}$ consist of three elements and singleton tests. Then $g = 3$, $t = t_{det}(\mathcal{H}) = 2$, $t_{rand}(\mathcal{H}) = \frac{5}{3}$, thus we get hypergraphs $\mathcal{L}$ with ratio $c = \frac{6}{5}$ and $t_{det}(\mathcal{L}) = \frac{3}{\log_2 5 - \log_2 3} \log_2 n$. It would be interesting to figure out the largest possible ratio c for any given factor of $\log_2 n$.

6 The Case of Graphs

Next we consider hypergraphs $\mathcal{H}$ of rank 2. For convenience we look at usual graphs only, where all edges have exactly two elements. This is not a severe restriction, due to the following reasoning. Small instances could be solved exhaustively, and for large instances, where also t_{det} is large, we do not have to care about one test more or less. Now, let $\{u\}$ be any singleton edge. Unless u is isolated, it also belongs to some edge $\{u, v\}$. Any given strategy can be changed in the way that it tests $\{u, v\}$ rather than $\{u\}$. In the negative case this is even more efficient. In the positive case, either one further test of an edge $\{u, w\}$ or $\{v, w\}$ can distinguish u and v, or u and v form a connected component. In the latter case we modify the instance by removing v and keeping the edge $\{u\}$ only. Altogether, subject to differences of at most 1 in the test numbers, it suffices to study usual graphs $G = (V, E)$ with edges of size 2, except that isolated vertices are also considered as edges.

A *fractional independent set* assigns a non-negative weight to every vertex, such that every edge has a total weight at most 1. In particular, isolated vertices can get the weight 1.

Proposition 7. *If $\mathcal{H}$ has a fractional independent set of total weight α, then $t_{rand}(\mathcal{H}) \geq \frac{\alpha}{2} - O(1)$.*

Proof. We normalize the assumed fractional independent set, that is, divide all weights by α, such that the weights sum up to 1 and hence form a probability distribution on V. It suffices to show that every deterministic strategy A needs an expected number of at least $\frac{\alpha}{2} - O(1)$ tests on this target distribution. As long as all tests are negative, strategy A tests the edges in some specific order. Since every edge has a probability mass of at most $\frac{1}{\alpha}$, the first i tests cover elements of total probability mass at most $\frac{i}{\alpha}$, for every i. By a simple exchange argument, the expected number of tests until A hits the target is minimized if every tested edge covers new elements whose total probability mass is exactly $\frac{1}{\alpha}$. This yields the assertion. □

A *fractional edge cover* assigns a non-negative weight x_e to every edge $e \in E$, such that every vertex is incident to edges with a total weight at least 1. In particular, isolated vertices (recall that we count them as edges) must get the weight 1. Finding a maximum-weight fractional independent set and a minimum-weight fractional edge cover are natural LP relaxations of the corresponding integral optimization problems, and they form a well-known pair of dual LPs, hence due to LP strong duality they achieve the same optimal value on a given graph.

A function with range $[0, 1]$ is called *half-integral* if it attains only the values $0, \frac{1}{2}, 1$. Although half-integrality is well studied for several optimization problems and is a tool for approximation algorithms (e.g., in [11]), we are not aware of an earlier proof of the following structural result that we will use to approximate the randomized test number.

Theorem 3. *In every graph, the fractional edge cover problem has an optimal solution that is half-integral, with the additional property that the edges e with $x_e = \frac{1}{2}$ form vertex-disjoint odd cycles. Moreover, this solution can be obtained in polynomial time.*

Proof. Consider any optimal fractional edge cover. Observe $x_e \leq 1$ for all edges, since otherwise we could reduce x_e to 1 and get a better valid solution. We call an edge e fractional if $0 < x_e < 1$. A tour is an sequence of edges that starts and ends in the same vertex; note that edges may appear several times in a tour and be traversed in arbitrary directions. The length of a tour is the number of edges, where repeatedly traversed edges are counted again.

Let C be a tour of fractional edges, and assume that C has even length. We may change the weights of the edges alternatingly by some amount $+\epsilon$ and $-\epsilon$. This changes neither the sum of weights incident to any vertex, nor the total weight. As for multiple edges e in C, note that the net effect on x_e is the sum of all changes applied there, and it may be zero. In the latter case we call e an inert edge. Suppose that C has some non-inert edge. Then, by increasing ϵ (starting from $\epsilon := 0$) we reach a value where some x_e becomes 0 or 1 while all other x_e are still in $[0, 1]$. Altogether we get an optimal solution with one fractional edge

less. Iterating the procedure we can destroy all even tours that do not consist of inert edges only.

Now, in the graph F induced by the fractional edges, every connected component has at most one cycle, and it has odd length. Namely, any even cycle is obviously an even tour of non-inert edges, any two intersecting odd cycles also contain an even cycle, and any two disjoint odd cycles connected by a path can be merged into one even tour where not all edges are inert. Next, F cannot have any vertex of degree 1, since the only incident edge would have weight 1 and thus not be fractional. It follows that every connected component of F is merely an odd cycle. Finally, any cycle of length k has a fractional edge cover size of at least $\frac{k}{2}$ (just sum up all constraints for the k vertices), and giving each edge in the cycle the weight $\frac{1}{2}$ is a valid solution.

An optimal fractional vertex cover is obtained by solving an LP, and the above transformations of x_e values in even tours can be done in polynomial time. □

Now we devise an algorithm to compute a randomized search strategy on a given graph G. Note that, in the case of graphs, a search strategy is completely specified by a (random) sequence of edges to be tested until the first positive test occurs. If two target candidates remain at this moment, then one more test identifies the target. Hence, for any target u, the expected test number equals (up to one test) the expected position of the first edge where u occurs in the sequence.

First we compute a fractional edge cover according to Theorem 3. From every odd cycle we randomly choose one vertex, called the left-over vertex. Let M be the set of all edges e with $x_e = 1$ (note that this includes all isolated vertices) plus disjoint edges that cover the odd cycles except their left-over vertices. (Clearly, any path with an even number of vertices is covered by a unique set of disjoint edges.) Let L be a further set of edges, one for each left-over vertex. That is, every edge in L contains a left-over vertex and some arbitrary neighbor. It is important to notice that $M \cup L$ is still an edge cover, thus every vertex occurs in some edge of $M \cup L$.

Let α be the total weight of our fractional edge cover, and let $\ell := |L|$. Observe that $|M| = \alpha - \frac{\ell}{2}$. Let c be the length of a shortest odd cycle C in our fractional edge cover. Our algorithm chooses among two strategies as follows.

Strategy 1 tests the edges of $M \cup L$ in random order. Since $|M| + |L| = \alpha + \frac{\ell}{2}$, the expected test number is at most $\frac{\alpha}{2} + \frac{\ell}{4}$.

Strategy 2 tests the edges of M in random order, followed by the edges of L in random order. Then any fixed vertex occurs already in M with probability at least $\frac{c-1}{c}$, and it occurs only in L with probability at most $\frac{1}{c}$. Hence the expected position of its first occurence is at most

$$\frac{c-1}{c} \cdot \frac{|M|}{2} + \frac{1}{c} \cdot \left(|M| + \frac{\ell}{2} \right) = \frac{c-1}{c} \cdot \left(\frac{\alpha}{2} - \frac{\ell}{4} \right) + \frac{1}{c} \cdot \alpha = \frac{c+1}{2c} \cdot \alpha - \frac{c-1}{c} \cdot \frac{\ell}{4}.$$

For any fixed ℓ this is maximized when c has the smallest possible value, $c = 3$, and then it becomes $\frac{2}{3}(\alpha - \frac{\ell}{4})$. (Actually we have already used the idea of Strategy 2 in the example that proves Proposition 6.)

Numbers α and ℓ are known from the fractional edge cover. We choose Strategy 1 or 2 with the smaller bound on the expected test number. The cut-off point is $\ell = \frac{\alpha}{3}$, hence the bound is at most $\frac{7}{12}\alpha$. Together with the lower bound $\frac{1}{2}\alpha$ from Proposition 7 we finally obtain:

Theorem 4. *For hypergraphs $\mathcal{H}$ with rank 2 we can approximate $t_{rand}(\mathcal{H})$ within a factor $\frac{7}{6}$ in polynomial time.*

7 The P-NP Borderline

The algorithm from Theorem 4 could be refined, e.g., by treating odd cycles of different length differently and analyzing the expecting target position more carefully. The lower bound from Proposition 7 can be raised, too. However, our main point in Theorem 4 was to achieve a rather good approximation ratio. Before making more efforts, one would like to know the real complexity status of computing $t_{rand}(\mathcal{H})$ for hypergraphs of rank 2. As we have seen, the problem is closely related to some standard fractional optimization problems. Moreover, we can solve it in polynomial time for special graphs. For instance, for graphs possessing a perfect matching it is not hard to show that it is optimal to test the matching edges in random order. The difficulty in general graphs is to schedule the tests that cover further, single vertices. Still it seems quite possible that the problem is polynomial-time solvable in graphs. We state this as an open problem. For rank 3 we have the following hardness result.

Theorem 5. *The problem of computing $t_{det}(\mathcal{H})$ for hypergraphs $\mathcal{H}$ of rank 3 is NP-complete.*

Proof. We give a reduction from the NP-complete Exact 3-Cover problem. An instance of Exact 3-Cover consists of a set U of $3k$ elements and a family of triples, i.e., subsets of U with 3 elements. The problem is to cover U by k of them. We construct a hypergraph $\mathcal{H}$ on $U \cup \{x, y, z\}$ where x, y, z are new elements. The edges of U are the given triples and all singletons. Trivially, the construction is polynomial. We claim that the Exact 3-Cover instance is affirmative if and only if $t_{det}(\mathcal{H}) \leq k + 2$.

If we can cover U by k triples, then we can test these k edges and are left with three target candidates, for any possible test outcomes. Then two further tests are sufficient (and necessary) to spot the target. Hence $t_{det}(\mathcal{H}) \leq k + 2$.

For the reverse direction, suppose that we cannot cover U by k triples. Then any k edges within U cover at most $3k-1$ elements. It follows that any $k-1$ edges within U cover at most $3k-2$ elements. Consider any deterministic strategy and an adversary giving negative answers to the first $k-1$ edges in U that are tested. These edges do not cover x, y, z and at least two more elements of U. Let $e \subset U$ be the kth test in U. Note that e still leaves some $u \in U$ uncovered. Regardless

when and in which order $\{x\}$, $\{y\}$, $\{z\}$ and e are tested, after the first three of them we still have at least two target candidates. Namely, if e is among the first three tests, then some elements both in U and outside U are yet uncovered, and if e is the last of these four tests, then two elements in U are yet uncovered. Altogether, $(k-1)+3 = k+2$ tests do not determine the target. □

It appears natural to conjecture that computing $t_{rand}(\mathcal{H})$ for hypergraphs $\mathcal{H}$ of rank 3 is NP-complete as well. An attempt would be to combine the same reduction idea with the averaging bound. However this looks more challenging. We remark that $t_{det,2}(\mathcal{H})$ is computable in polynomial time if $\mathcal{H}$ has rank 2: It is optimal to test some maximum matching in round 1. (Details are omitted due to space limitations.) So we conclude with the following

Problem: What is the complexity status of computing $t_{rand}(\mathcal{H})$ for hypergraphs $\mathcal{H}$ of any fixed rank?

References

1. 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)
2. 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)
3. 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: Epstein, L., Ferragina, P. (Eds.) ESA 2002. LNCS, vol. 2461, pp. 223–233, Springer, Heidelberg (2002)
4. Crowston, R., Gutin, G., Jones, M., Muciaccia, G., Yeo, A.: Parameterizations of Test Cover with Bounded Test Sizes. CoRR abs/1209.6528 (2012)
5. Crowston, R., Gutin, G., Jones, M., Saurabh, S., Yeo, A.: Parameterized Study of the Test Cover Problem. In: Rovan, B., Sassone, V., P Widmayer, P. (Eds.) MFCS 2012. LNCS, vol. 7464, pp. 283–295, Springer, Heidelberg (2012)
6. 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)
7. Du, D.Z., Hwang, F.K.: Combinatorial Group Testing and Its Applications. Series on Appl. Math. vol. 3. World Scientific (2000)
8. 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)
9. Fernau, H., Raible, D.: A Parameterized Perspective on Packing Paths of Length Two. In: Yang, B., Du, D.Z., Wang, C.A. (Eds.) COCOA 2008. LNCS, vol. 5165, pp. 54–63, Springer, Heidelberg (2008)
10. Gutin, G., Muciaccia, G., Yeo, A.: (Non-)existence of Polynomial Kernels for the Test Cover Problem. Info. Proc. Letters 113, 123–126 (2013)
11. Hochbaum, D.S.: Solving Integer Programs over Monotone Inequalities in Three Variables: A Framework for Half Integrality and Good Approximations. Eur. J. Operational Res. 140, 291–321 (2002)
12. Wiener, G.: Rounds in Combinatorial Search. Algorithmica 67, 315–323 (2013)