Testing Local Properties of Arrays

Omri Ben-Eliezer
Tel Aviv University, Tel Aviv 69978, Israel
omrib@mail.tau.ac.il

Abstract

We study testing of local properties in one-dimensional and multi-dimensional arrays. A property of d-dimensional arrays $f : [n]^d \to \Sigma$ is *k-local* if it can be defined by a family of $k \times \ldots \times k$ *forbidden consecutive patterns*. This definition captures numerous interesting properties. For example, monotonicity, Lipschitz continuity and submodularity are 2-local; convexity is (usually) 3-local; and many typical problems in computational biology and computer vision involve $o(n)$-local properties.

In this work, we present a generic approach to test all local properties of arrays over any finite (and not necessarily bounded size) alphabet. We show that *any* k-local property of d-dimensional arrays is testable by a simple canonical one-sided error non-adaptive ϵ-test, whose query complexity is $O(\epsilon^{-1}k \log \frac{\epsilon n}{k})$ for $d = 1$ and $O(c_d \epsilon^{-1/d} k \cdot n^{d-1})$ for $d > 1$. The queries made by the canonical test constitute sphere-like structures of varying sizes, and are completely independent of the property and the alphabet Σ. The query complexity is optimal for a wide range of parameters: For $d = 1$, this matches the query complexity of many previously investigated local properties, while for $d > 1$ we design and analyze new constructions of k-local properties whose one-sided non-adaptive query complexity matches our upper bounds. For some previously studied properties, our method provides the first known sublinear upper bound on the query complexity.

2012 ACM Subject Classification Theory of computation → Sketching and sampling

Keywords and phrases Property Testing, Local Properties, Monotonicity Testing, Hypergrid, Pattern Matching

Digital Object Identifier 10.4230/LIPIcs.ITCS.2019.11

Related Version A full version of the paper is available at [5], `https://eccc.weizmann.ac.il/report/2018/196/`.

Acknowledgements I would like to thank Frederik Benzing, Eric Blais, Eldar Fischer, Sofya Raskhodnikova, Daniel Reichman and C. Seshadhri for stimulating discussions, and the anonymous reviewers for helpful suggestions.

1 Introduction

Property testing [22, 33] is devoted to understanding how much information one needs to extract from an object in order to determine whether it satisfies a given property or is *far* from satisfying the property. This active research area has seen many developments through the last two decades; see the recent book of Goldreich [21] for a good introduction. The property testing notation we use here is standard, see Subsection 1.7 for the relevant definitions.

In this paper we focus on testing of local properties in structured data. The objects we consider are *d-dimensional arrays*, where d is a positive integer, viewed as a constant. A d-dimensional *array* of *width* n, or an *$[n]^d$-array* in short, is a function $A\colon [n]^d \to \Sigma$ from the hypergrid $[n]^d$ to the *alphabet* Σ, where the alphabet Σ is allowed to be any (arbitrarily large)

finite set; we stress that the size of Σ is usually *not required* to be bounded as a function of the other parameters. For example, a *string* is an $[n]^1$-array, and the commonly used RGB representation of images is basically an $[n]^2$-array over $\{0, 1, \ldots, 255\}^3$, where the three values corresponding to each pixel represent the intensity of red, green and blue in it.

1.1 Local properties

Korman, Reichman, and the author [7] recently investigated testing of the property of *consecutive pattern freeness*, i.e., not containing a copy of some (predefined) "forbidden" consecutive subarray. Here, a $[k]^d$-array S is a (consecutive) *subarray* of an $[n]^d$-array A in *location* $(i_1, \ldots, i_d) \in [n-k+1]^d$ if $A(i_1 + j_1 - 1, \ldots, i_d + j_d - 1) = S(j_1, \ldots, j_d)$ for any $j_1, \ldots, j_d \in [k]$.

Naturally, a more general follow-up question raised in [7] was the following: what can be said about testing of properties defined by a *family of forbidden consecutive patterns*? As we shall see soon, many interesting properties of arrays (including a large fraction of the array properties that were previously investigated in the literature) can be characterized this way.

With this in mind, we call a property *local* if it can be characterized by a family of small forbidden consecutive patterns. Formally, a property $\mathcal{P}$ of $[n]^d$-arrays over an alphabet Σ is *k-local* (for $2 \leq k \leq n$) if there exists a family $\mathcal{F}$ of $[k]^d$-arrays over Σ so that the following holds for any $[n]^d$-array A over Σ:

$$A \text{ satisfies } \mathcal{P} \iff \text{None of the (consecutive) subarrays of } A \text{ is in } \mathcal{F}.$$

For $\mathcal{P}$ as above, we sometimes write $\mathcal{P} = \mathcal{P}(\mathcal{F})$ to denote that $\mathcal{P}$ is defined by the forbidden family $\mathcal{F}$.

The main contribution of this work is a generic one-sided error non-adaptive framework to test k-local properties. In some cases, our method either matches or beats the best known upper bounds on the query complexity (although the running time might be far from optimal in general). We show the optimality of our method by proving a matching lower bound for non-adaptive one-sided tests, as well as a (weaker) lower bound for two-sided tests.

In order to demonstrate the wide range of properties captured by the above definition, we now present various examples of properties that are k-local for small k, including some of the most widely investigated properties in the property testing literature, as well as properties from areas of computer science that were not systematically studied in the context of property testing. In what follows, the sum of two tuples $x = (x_1, \ldots, x_d), y = (y_1, \ldots, y_d)$ is defined as the tuple $(x_1 + y_1, \ldots, x_d + y_d)$; additionally, e^i denotes the i-th unit vector in d dimensions.

Monotonicity. Perhaps the most thoroughly investigated property in the testing literature: see e.g. the entries related to monotonicity testing in the Encyclopedia of Algorithms [14, 32] and the references within. An $[n]^d$-array A over an ordered alphabet Σ is *monotone* (non-decreasing) if $A(x) \leq A(y)$ for any $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ satisfying $x_i \leq y_i$ for any i. Monotonicity is 2-local: an array A is monotone if and only if there is no pair $x, x + e^i \in [n]^d$ so that $A(x) > A(x + e^i)$.

Lipschitz continuity. Another well-investigated property with connections to differential privacy [3, 10, 15, 24], an $[n]^d$-array A is *c-Lipschitz continuous* if $|A(x) - A(y)| \leq c \sum_{i=1}^{d} |y_i - x_i|$ for any $x, y \in [n]^d$. This condition holds iff $|A(x) - A(x + e^i)| \leq c$ for any $x, x + e^i \in [n]^d$, and thus Lipschitz continuity is also 2-local.

Convexity. Discrete convexity is an important geometric property with connections to optimization and other areas [9, 8, 13, 18, 30, 31]. A one-dimensional array A is *convex* if $\lambda A(x) + (1 - \lambda)A(y) \geq A(\lambda x + (1 - \lambda)y)$ for any $x, y \in [n]$ and $0 < \lambda < 1$ satisfying

$\lambda x + (1-\lambda)y \in [n]$. Convexity is 3-local for the case $d = 1$: an array $A : [n] \to \Sigma$ is convex if and only $A[x] - 2A[x+1] + A[x+2] \geq 0$ for any $x \in [n-2]$. In higher dimensions, several different notions of discrete convexity have been used in the literature – see e.g. the introductory sections of the book of Murota on discrete convex analysis [27]. Two of the commonly used definitions, $M^\natural$-convexity and $L^\natural$-convexity, are 3-local and 4-local, respectively: see Theorems 4.1 and 4.2 in [26], where it is shown that both notions can be defined locally using slight variants of the Hessian matrix consisting of the partial discrete derivatives. Another common definition that is a natural variant of the continuous case states that convexity is equivalent to the positive semi-definiteness of the Hessian matrix; under this definition, convexity is 3-local. A strictly weaker notion of convexity, called *separate convexity* [13], is defined as follows: an $[n]^d$-array A is separately convex if it is convex along each of the axes. Similarly to one-dimensional convexity, separate convexity is 3-local for any d.

Properties of higher order derivatives. More generally, *any* property of arrays that can be characterized by "forbidden pointwise behavior" of the first k discrete derivatives [13] is $(k+1)$-local. Monotonicity (for $k = 1$), Lipschitz continuity ($k = 1$) and convexity ($k = 2$) are special cases of such properties.

Submodularity. An important property closely related to convexity [10, 11, 30, 34]. Given $x = (x_1, \ldots, x_d), y = (y_1, \ldots, y_d) \in [n]^d$, define $x \wedge y = (\min(x_1, y_1), \ldots, \min(x_d, y_d))$ and $x \vee y = (\max(x_1, y_1), \ldots, \max(x_d, y_d))$. An $[n]^d$-array is *submodular* if $A[x \wedge y] + A[x \vee y] \leq A[x] + A[y]$ for any $x, y \in [n]^d$. Submodularity is 2-local: it is not hard to verify that submodularity is equivalent to the condition that $A(x) + A(x+e^i+e^j) \leq A(x+e^i) + A(x+e^j)$ for all x.

Pattern matching and computer vision. Tasks involving pattern matching under some limitations – such as noise in the image, obstructed view, or rotation of elements in the image – are at the core of computer vision and its applications. For example, the local property of not containing a good enough ℓ_1-approximation of a given forbidden pattern is of practical importance in computer vision. Sublinear approaches closely related to property testing are known to be effective for problems of this type, see e.g. [25].

Computational biology. Many problems in computational biology are closely related to one-dimensional pattern matching. As an example, a defensive mechanism of the human body against RNA-based viruses involves "cutting" a suspicious RNA fragment, if it finds one of a (small) family of short forbidden consecutive patterns in it, indicating that this RNA might belong to a virus. Thus, in order to generate fragments of RNA that are not destroyed by such defensive mechanisms (which is a basic task in computational biology), understanding the process of "repairing" a fragment so that it will not contain any of the forbidden patterns is an interesting problem related to property testing.

1.2 Previous results on local properties

One-dimensional arrays

A seminal result of Ergün et al. [19] shows that for constant ϵ, monotonicity is ϵ-testable over the line (that is, for one-dimensional arrays) using $O(\log n)$ queries over general alphabets. The non-adaptive one-sided error test proposed in [19] is based, roughly speaking, on imitating a binary search non-adaptively. It was shown by Fischer [20] that the above is tight even for two-sided error adaptive tests, proving a matching $\Omega(\log n)$ lower bound. Later on, Parnas, Ron and Rubinfeld [30] and Jha and Raskhodnikova [24] showed that the $O(\log n)$ upper bound on the non-adaptive one-sided query complexity also holds for convexity and Lipschitz

continuity, respectively. For general ϵ, the upper bound in [24] is of the type $O(\epsilon^{-1} \log n)$; the same work also presents a matching lower bound of $\Omega(\log n)$ for the one-sided non-adaptive case, while $\Omega(\log n)$ lower bounds for two-sided non-adaptive tests of convexity, and more generally, monotonicity of the ℓ-th derivative, are proved by Blais, Raskhodnikova and Yaroslavtsev [13] using a communication complexity based approach [12]. Finally, a recent result of Belovs [4] refines the one-sided non-adaptive query complexity of monotonicity to $O(\epsilon^{-1} \log \epsilon n)$.

When the alphabet is binary (of size two), general positive results are known regarding the testability of local properties in one dimension. It follows from the testability of regular languages, established by Alon et al. [2], that any k-local property is testable in $O(c(\mathcal{F})\epsilon^{-1}(\log^3(\epsilon^{-1})))$ queries, where $c(\mathcal{F})$ depends only on the family $\mathcal{F}$ of forbidden consecutive length-k patterns defining the property. However, $c(\mathcal{F})$ can be exponential in k in general.

Multi-dimensional arrays

Chakrabarty and Seshadhri [16] extended some of the above results to hypergrids, showing that a general class of so-called "bounded derivative" properties (all of which are 2-local), including monotonicity and Lipschitz continuity as special cases, are all testable over $[n]^d$-arrays with $O(\epsilon^{-1} d \log n)$ queries. Another work by the same authors [17] shows a matching lower bound of $\Omega(\epsilon^{-1} d \log \epsilon n)$ for monotonicity, that holds even for two-sided adaptive tests, while the communication complexity approach of [13] gives a (non-adaptive, two-sided) $\Omega(d \log n)$ lower bound for convexity, separate convexity and Lipschitz.

Submodularity is testable for $d = 2$ with $O(\log^2 n)$ queries [30]; However, no non-trivial upper bound on the query complexity is known for submodularity in the case $d > 2$ and convexity in the case $d > 1$ under the Hamming distance and over general alphabets (although [9] proves constant-query testability for 2D convexity over a binary alphabet). Under L_1-distance and for any d, it was shown in [10] that convexity in $[n]^d$-arrays is testable with number of queries that depends only on d.

Pattern freeness

In [7], it was shown that the property of (consecutive) pattern freeness, for a single forbidden pattern, is testable with $O(c_d/\epsilon)$ queries for any d. The proof, however, requires multiple sophisticated combinatorial observations and does not seem to translate to the case of a family of forbidden patterns discussed here.

1.3 Our results

In this work, we present a generic approach to test *all* k-local properties of $[n]^d$-arrays. Among other consequences, a simple special case of our result in the one-dimensional regime shows that the abundance of properties whose query complexity is $\Theta(\log n)$ is not a coincidence: in fact, *any* $O(1)$-local property of one-dimensional arrays is testable with $O(\log n)$ queries, using a canonical binary search like querying scheme.

On the other hand, we prove a lower bound for testing local properties in $d > 1$ dimensions, showing that the query complexity of our test is optimal (for fixed d) among non-adaptive one-sided tests, even when restricted to alphabets of size that is polynomial in n^d. We also prove a lower bound for non-adaptive two-sided tests.

1.3.1 Upper bounds

Our first main result is an upper bound on the number of queries required to test any k-local property of $[n]^d$-arrays non-adaptively with one-sided error. The test is *canonical* in a strong sense: The queries it makes depend on n, d, k, and (relatively weakly) on ϵ; they do not depend on $\mathcal{P}$ or the alphabet Σ. In other words, it makes the same type of queries for all k-local properties of $[n]^d$-arrays over any finite (and not necessarily bounded-size) alphabet.

▶ **Theorem 1.** *Let $2 \leq k \leq n$ and $d \geq 1$ be integers, and let $\epsilon > 0$. Any k-local property $\mathcal{P}$ of $[n]^d$-arrays over any finite (and not necessarily bounded size) alphabet has a one-sided error non-adaptive ϵ-test whose number of queries is*

- $O(\frac{k}{\epsilon} \cdot \log \frac{\epsilon n}{k})$ *for $d = 1$.*
- $O(c^d \frac{k}{\epsilon^{1/d}} \cdot n^{d-1})$ *for $d > 1$.*

Here, $c > 0$ is an absolute constant. The test chooses which queries to make based only on the values of n, d, k, ϵ, and independently of the property $\mathcal{P}$ and the alphabet Σ.

Note that we are interested here in the domain where n is large and d is considered a constant. Thus, we did not try to optimize the c^d term in the second bullet, seeing that it is negligible compared to n^{d-1} anyway.

Running time

The main drawback of our approach is the running time of the test, which is high in general. After making all of its queries, our test runs an *inference* step, where it tries to evaluate (by enumerating over all relevant possibilities) whether a violation of the property must occur in view of the queries made, and reject if this is the case.

Without applying any property-specific considerations, the running time of the inference step is of order $|\Sigma|^{O(n^d)}$. However, for various specific properties of interest, such as monotonicity and 1D-convexity, it is not hard to make the running time of the inference step of the same order of magnitude as the query complexity. Moreover, in one dimension we can use dynamic programming to achieve running time that is significantly better than the naive one, but still much higher than the query complexity in general: $O(|\Sigma|^{O(k)} n)$. This works for any k-local property in one dimension; see the last part of Subsection 1.4.1 for more details.

Proximity oblivious test

Interestingly, the behavior of the test depends quite minimally on ϵ, and it can be modified very slightly to create a proximity oblivious test (POT) for any k-local property. The useful notion of a POT, originally defined by Goldreich and Ron [23], refers to a test that does *not* receive ϵ as an input, and whose success probability for an input not satisfying the property is a function of the Hamming distance of the input from the property.

▶ **Theorem 2.** *Fix $d > 0$. Any k-local property $\mathcal{P}$ of $[n]^d$-arrays over any finite (but not necessarily bounded size) alphabet has a one-sided error non-adaptive proximity oblivious test whose number of queries is $O(k \log(n/k))$ if $d = 1$ and $O(kn^{d-1})$ if $d > 1$. For any input A not satisfying $\mathcal{P}$, the rejection probability of A is linear (for fixed d) in the Hamming distance of A from $\mathcal{P}$.*

One can run $O(c_d/\epsilon)$ iterations of the POT to obtain a standard one-sided error non-adaptive test. The query complexity is $O(k\epsilon^{-1} \log(n/k))$ for $d = 1$ and $O(c_d k \epsilon^{-1} n^{d-1})$ for $d > 1$, where $c_d > 0$ depends only on d. Thus, the POT-based test is sometimes as good as the test of Theorem 1 (specifically, for $d = 1$ it matches the above bounds for almost the whole range of ϵ and k). In any case, the multiplicative overhead of the POT-based test is sublinear in $1/\epsilon$ across the whole range.

Type of queries

In one dimension, many of the previously discussed properties, including, for example, monotonicity and Lipschitz continuity, are testable in $O(\log n)$ queries. Previously known tests for monotonicity and Lipschitz continuity make queries that resemble a binary search: these tests query pairs of entries of distance 2^i for multiple choices of $0 \leq i \leq \log n$.

Our test continues the line of works using querying schemes roughly inspired by binary search. The test queries structures that can be viewed, intuitively, as L_∞-*spheres* of different sizes in $[n]^d$. For this purpose, an L_∞-sphere with radius r and width ℓ in $[n]^d$ is a set $X_1 \times X_2 \times \ldots \times X_d \subseteq [n]^d$, where each X_i is a union of intervals of the form $[a_i, a_i+1, \ldots, a_i+\ell-2, a_i+\ell-1] \cup [b_i-\ell+2, b_i-\ell+3, \ldots, b_i-1, b_i]$, and $b_i - a_i \in \{r, r+1\}$ for any $i \in [d]$. More specifically, our test for k-local properties queries spheres with width $k-1$ and radius of order 2^i for different values of i. In the simple special case where $d = 1$ and $k = 2$, this is very similar to the querying scheme mentioned in the previous paragraph.

Implications

In one dimension, the query complexity of the test matches the best known upper bounds (and, in some regimes, refines the dependence on ϵ) for several previously investigated properties including monotonicity, Lipschitz continuity and convexity. For monotonicity of k-th order derivatives, which is $(k+1)$-local, it proves the first sublinear upper bound on the query complexity: $O(k \log n)$; in comparison, the best known lower bound [13] is $\Omega(\log n)$.

For pattern matching type properties in 1D arrays (including applications in computational biology and other areas), our approach gives a property- and alphabet-independent upper bound of $O(k \log n)$ on the query complexity, with essentially optimal dependence on ϵ as well. Previously known approaches for testing such properties, like the regular languages testing approach [2], yield tests whose query complexity is dependent on the family of forbidden patterns considered, whose size might be exponential in the locality parameter k. Our approach, on the other hand, requires an $O(\log n)$ "overhead", but its query complexity is independent of the size of the forbidden family discussed. Instead, the dependence in k is linear.

In multiple dimensions, our approach is far from tight for well-understood properties such as monotonicity and Lipschitz continuity, whose query complexity is known to be $\Theta(d \log n)$ (in comparison, our approach yields an $O(n^{d-1})$ type bound). However, for testing of other properties like convexity (for $d > 1$) and submodularity (for $d > 2$) in $[n]^d$-arrays, no non-trivial upper bounds on the query complexity are known over general alphabets, so our upper bound of $O(n^{d-1})$ is the first such bound. While we do not believe this bound is tight in general, this might be a first step towards the development of new tools for efficiently testing such properties.

Sketching for testing

The fact that the queries made are completely independent of the property suggests the following sketching technique allowing for "testing in retrospect": Given ϵ and k in advance, we make all queries of the generic ϵ-test for k-local properties in "real time", and store them for postprocessing. This is suitable, for example, in cases where we have limited access to a large input for a limited amount of time (e.g. when reading the input requires specialized expensive machinery), but the postprocessing time is not an issue. Note that for this approach we do not need to know the property of interest in advance.

1.3.2 Lower bounds

Our next main result is a tight lower bound for non-adaptive one-sided error testing of local properties, that applies for any d, and is tight for any fixed d satisfying $d > 1$.

▶ **Theorem 3** (One-sided tests). *Let $d \geq 1$ and $n \geq k \geq 2$ be integers, and let $d/n < \epsilon < 1$. There exists a k-local property $\mathcal{P}$ of $[n]^d$-arrays over an alphabet Σ of size $n^{O(d)}$, so that any non-adaptive one-sided error ϵ-test for $\mathcal{P}$ requires $\Omega\left(\min\left\{\frac{k}{d\epsilon^{1/d}} \cdot n^{d-1}, n^d\right\}\right)$ queries.*

Note that the size of the alphabet in Theorem 3 is only polynomial in the input size. We also prove a lower bound for non-adaptive two-sided tests; here the dependence in $|\Sigma|$ is exponential.

▶ **Theorem 4** (Two-sided tests). *Let $d \geq 1$ and $n \geq k \geq 2$ be integers, and let $d/n < \epsilon < 1$. There exists a k-local property $\mathcal{P}$ of $[n]^d$-arrays over an alphabet Σ of size $2^{O(n^d)}$, so that any non-adaptive two-sided error ϵ-test for $\mathcal{P}$ requires $\Omega\left(\min\left\{\frac{\sqrt{k}}{d\epsilon^{(d+1)/2d}} \cdot n^{(d-1)/2}, n^d\right\}\right)$ queries.*

For fixed $d > 1$, the lower bound for one-sided tests matches the upper bound from Theorem 1 across the whole range of ϵ and k. For $d = 1$ the bound obtained here is $\Omega(k/\epsilon)$, which is tight up to a $\log n$ factor. Note the threshold behavior occurring at $k/\epsilon^{1/d} = \Theta(n)$: When $k/\epsilon^{1/d} = o(n)$, the upper bound of Theorem 1 implies that any k-local property is ϵ-testable with a sublinear number of queries, while for $k/\epsilon^{1/d} = \Omega(n)$, the property of Theorem 3 requires $\Omega(n^d)$ non-adaptive one-sided queries to test.

From Theorem 4 we conclude that the improvement in query complexity obtained by two-sided error non-adaptive tests is at most quadratic in the worst case; specifically, there exists a 2-local property requiring $n^{\Omega(d-1)}$ queries to test by two-sided non-adaptive tests.

1.4 Proof ideas and techniques

Here we present the main ideas of our proofs in an informal way, starting with the upper bound. For simplicity, we stick to the one-dimensional case, and assume that ϵ is fixed and $k = o(n)$.

1.4.1 Upper bound for 1D

Suppose that $\mathcal{P} = \mathcal{P}(\mathcal{F})$ is a k-local property of $[n]^1$-arrays A over an alphabet Σ, defined by the forbidden family $\mathcal{F}$. Let S be a consecutive subarray of A of length at least $2k - 2$. The *boundary* of S consists of the first $k - 1$ elements and the last $k - 1$ elements of S, and all other elements of S are its *interior*. We call S *unrepairable* if one cannot make the array S satisfy the property $\mathcal{P}$ without changing the value of at least one element in its boundary. Otherwise, S is *repairable*. Observe the following simple facts.

- It suffices to only query the boundary elements of S in order to determine whether S is unrepairable.
- If S is unrepairable, then A does not satisfy $\mathcal{P}$.
- If S is repairable, then we can delete all forbidden patterns from S by modifying only entries in its interior, without creating any new copies of forbidden patterns in A.

We call the process of understanding whether S is unrepairable using only its boundary elements *inference*. Note that the inference step does not make any additional queries.

A simple sublinear test

A first attempt at a generic test for local properties is the following: we query $\Theta(\sqrt{n})$ intervals in $[n]$, each containing exactly $k-1$ consecutive elements, including the intervals $\{1,\ldots,k-1\}$ and $\{n-k+2,\ldots,n\}$, where the distance between each two neighboring intervals is $\Theta(\sqrt{n})$. A *block* is a subarray consisting of all elements in a pair of neighboring intervals and all elements between them. The crucial observation is that at least one of the following must be true, for any array A that is ϵ-far from $\mathcal{P}$ (recall that ϵ is fixed).

- At least one of the blocks is unrepairable.
- At least $\Omega(\sqrt{n})$ of the blocks do not satisfy $\mathcal{P}$.

Indeed, if the first condition does not hold, then one can make A satisfy $\mathcal{P}$ by only changing elements in the interiors of blocks that do not satisfy $\mathcal{P}$. Seeing that A is ϵ-far from $\mathcal{P}$ and that we do not need to modify elements in the interiors of blocks that satisfy $\mathcal{P}$, this implies that at least $\Omega(\sqrt{n})$ of the blocks do not satisfy $\mathcal{P}$.

Now we are ready to present the test: We query all $O(k\sqrt{n})$ elements of all intervals, and additionally, all $O(\sqrt{n})$ elements of $O(1)$ blocks. Querying all elements of all intervals suffices to determine (with probability 1) whether one of the blocks is unrepairable. If A is ϵ-far from $\mathcal{P}$ and does not contain unrepairable blocks, querying $O(1)$ full blocks will catch at least one block not satisfying $\mathcal{P}$ with constant probability, as desired.

For more details, see Section 3 and the preliminary Section 2 that prepares the required infrastructure.

The optimal test

Improving the query complexity requires us to construct a system of *grids* – which are merely subsets of $[n]$ – inspired by the behavior of binary search. In comparison, the approach of the previous test is essentially to work with a single grid.

The first (and coarsest) grid contains only the first $k-1$ elements and the last $k-1$ elements of $[n]$. In other words, it is equal to $\{1,\ldots,k-1,n-k+2,\ldots,n\}$. The second grid refines the first grid – that is, it contains all elements of the first grid – and additionally, it contains $k-1$ consecutive elements whose center is $n/2$ (whenever needed, rounding can be done rather arbitrarily). We continue with the construction of grids recursively: To construct grid number $i+1$, we take grid number i and add $k-1$ elements in the middle of each block of grid i (blocks are defined as before). Note that the length of blocks is roughly halved with each iteration. We stop the recursive construction when the length of all of the blocks becomes no bigger than ck, where $c \geq 2$ is an absolute constant.

For each block B in grid number $i > 1$, we define its *parent*, denoted $\mathrm{Par}(B)$, as the unique block in interval $i-1$ containing it. A block B in the system of grid is *maximally unrepairable* if it is unrepairable, and all blocks (of all grids in the system) strictly containing it are repairable. It is not hard to see that different maximally unrepairable blocks have disjoint interiors.

The main observation now is that in order to make A satisfy $\mathcal{P}$, it suffices to only modify entries in the interiors of parents of maximally unrepairable blocks. If A is ϵ-far from $\mathcal{P}$, then the total length of these parents must therefore be $\Omega(n)$ (for constant ϵ). However, since the length of $\mathrm{Par}(B)$ is roughly twice the length of B, we conclude that the total length of all maximally unrepairable blocks is $\Omega(n)$.

With this in hand, it can be verified that the following test has constant success probability. For each grid in the system, we pick one block of the grid uniformly at random, and query all entries of its boundary. Additionally, for the finest grid (whose block length is $O(k)$), we also query all interior elements of the picked block.

For more details, see Section 4 (which builds on the infrastructure of Section 2).

Running time in 1D

We now show that the running time of the inference step for a block of length m is $m|\Sigma|^{O(k)}$. Summing over all block lengths, this would imply that the total running time of the test is $n|\Sigma|^{O(k)}$.

The proof uses dynamic programming. Let S be an array of length m over Σ, and assume that $S(1), S(2), \ldots, S(k-1)$ and $S(m-k+2), \ldots, S(m)$ are all known. For each "level" from 1 to $m-k+1$, we keep a boolean predicate for each of the $|\Sigma|^k$ possible patterns of length k over Σ. These predicates are calculated as follows.

- In the first level, the predicate of $\sigma = (\sigma_1, \ldots, \sigma_k)$ evaluates to TRUE if $S(1) = \sigma_1, \ldots, S(k-1) = \sigma_{k-1}$, and additionally, $\sigma \notin \mathcal{F}$, that is, σ is not a forbidden pattern. Otherwise, the predicate of σ is set to FALSE.
- For $i = 2$ to $m-k+1$, the predicate of $\sigma = (\sigma_1, \ldots, \sigma_k)$ in level i evaluates to TRUE if and only if
  1. $\sigma \notin \mathcal{F}$.
  2. there exists $\sigma' = (\sigma'_0, \sigma_1, \ldots, \sigma_{k-1})$ that evaluates to TRUE in level $i-1$.
- Finally, the predicates in level $m-k+1$ are modified as follows: for all $\sigma = (\sigma_1, \ldots, \sigma_k)$ so that $\sigma_j \neq S(m-k+j)$ for some $j \geq 2$, we set the predicate of σ to FALSE.

It is not hard to see that S is unrepairable if and only if all predicates at level $m-k+1$ are FALSE. The running time is $O(m|\Sigma|^{cd})$ for a suitable constant $c > 0$.

Generalization to higher dimensions

The generalization to higher dimensions is relatively straightforward; the main difference is that the boundary of blocks now is much larger: blocks of size $m \times \ldots \times m$ have boundary of size $O(kdm^{d-1})$. Thus, essentially the same proof as above (with suitable adaptations of the definitions) yields a test with query complexity $O(kdn^{d-1})$ for constant ϵ. For the running time, we can no longer use dynamic programming; using the naive approach of enumerating over all possible interior elements of a block, we get that the inference time for a block of size $m \times \ldots \times m$ is $|\Sigma|^{O(m^d)}$, making the total running time of the test $|\Sigma|^{O(n^d)}$.

1.4.2 Lower bound

The property $\mathcal{P}$ underlying our lower bound construction consists of $[n]^d$-arrays A over Σ satisfying all of the following properties. Here we provide a construction over alphabet size $2^{O(n^d)}$, but in Section 6 of the full version of this paper [5] we show how a simple modification of the property can be conducted in order to decrease the size of the required alphabet to $n^{O(d)}$ for the proof of Theorem 3 (unfortunately, for the proof of Theorem 4 this modification does not work).

- The alphabet Σ is of the form $[n]^d \times [n]^d \times 2^{[2n^{d-1}]}$, where 2^X is the *power set* of a set X. The value of A in entry $x \in [n]^d$ is represented as a tuple $A_1(x), A_2(x), A_3(x)$. We view $A_1(x), A_2(x)$ as *pointers* emanating from x.
- For every $x \in [n]^d$ we require $A_1(x) = x$. That is, A_1 points to the location of the element itself.
- There exists a special location $\ell = (\ell_1, \ldots, \ell_d) \in [n]^d$ so that all $x \in [n]^d$ point to ℓ with their second pointer, that is, $A_2(x) = \ell$. We call this location the *lower center of gravity*. We define an *upper center of gravity* as $u = (\ell_1 + 1, \ell_2, \ldots, \ell_d)$.

- A *floor* entry $x = (x_1, \ldots, x_d) \in [n]^d$ satisfies $x_1 = 1$ and a *ceiling* entry satisfies $x_1 = n$. For each such floor or ceiling entry x, we pick $A_3(x)$ to be a singleton (i.e., a set with one element).
- For each floor element x, there exists a path Γ_x from x to the lower center of gravity ℓ. Similarly, for each ceiling element y, there is a path Γ_y directed towards the upper center of gravity, u. In both cases, the path is of length $O(nd)$. The structure of the path depends only on its start and end points (so depends only on x and ℓ in the first case, and y and u in the second case).
- The A_3-data "flows" to the center of gravity through paths. Formally, the A_3-set of each entry y that is not a floor or ceiling entry is required to be equal to the union $\bigcup_{x:\ y \in \Gamma_x} A_3(x)$. In other words, the data in each location in $[n]^d$ is an "aggregation" of the data flowing in all paths that intersect it.
- Finally, we require that $A_3(u) = A_3(\ell)$.

While $\mathcal{P}$ was defined above in global terms, we show that it is actually a 2-local property, that is, all conditions specified here can be written in a 2-local way.

To prove the lower bound, we follow Yao's minimax principle [35], defining a distribution of arrays satisfying $\mathcal{P}$, and a distribution of arrays which are $\Omega(1)$-far from satisfying $\mathcal{P}$, so that a large number of queries is required to distinguish between the distributions.

As positive examples, we take a collection of arrays A satisfying the property, and require that all the singletons in the floor are pairwise disjoint. For negative examples (that are 1/4-far from $\mathcal{P}$), we consider a collection of arrays satisfying all of the above requirements other than the last. Instead, all singletons in the floor and the ceiling are pairwise disjoint (so in particular, $A_3(\ell) \cap A_3(u) = \emptyset$).

We show that for any given $x \in [n]^d$, the expected size of $A_3(x)$ over each of the distributions is $O(d)$. For the one-sided error case, it is shown that one needs to know the values of at least $\Omega(n^{d-1})$ singletons to be able to distinguish between positive and negative examples with one sided error, implying that $\Omega(n^{d-1}/d)$ queries are required to reject negative examples with constant probability.

For the two sided error case, the argument is inspired by the birthday paradox. Very loosely speaking, it follows from the fact that, given two unknown unordered sets A and B of size n, one has to make $\Omega(\sqrt{n})$ queries to distinguish between the case that $A = B$ and the case that $A \cap B = \emptyset$.

Due to space considerations, the full proofs of Theorems 3 and 4 are relegated to the full version of this paper [5]; see Sections 5 and 6 there.

1.5 Other related work

This subsection complements Subsection 1.2, presenting other related previous works that were not mentioned above.

General results in property testing

This paper adds to the growing list of general characterization results in property testing of strings, images, and multi-dimensional arrays; see [1, 6] and the references within for characterization-type results in these domains, mostly over a fixed size alphabet. In particular, for strings, it was shown by Alon et al. [2] that any local property over a fixed size alphabet is constant-query testable, and this paper shows that an overhead of at most $O(\log n)$ is required when the alphabet size is unbounded.

Hyperfiniteness

A graph is hyperfinite if, roughly speaking, it can be decomposed into constant size connected components by deleting only a small constant fraction of the edges. Newman and Sohler [28] investigated the problem of testing in hyperfinite graphs, showing that any property of hyperfinite bounded degree graphs is testable with a constant number of queries. While the graph with which we (implicitly) work – the hypergrid graph, whose vertices are in $[n]^d$ and two vertices are neighbors if they differ by 1 in one coordinate – is a hyperfinite bounded degree graph (for constant d), the results of [28] are incomparable to ours. Indeed, in our case the vertices are inherently ordered, and it does not make sense to allow adding edges between vertices that are not neighbors (as entries of $[n]^d$), unlike the case in [28], where one may add or remove edges arbitrarily between any two vertices. Still, the hyperfiniteness of our graph seems to serve as a major reason that local properties have sublinear tests.

Block tests for image properties

The works of Berman, Murzabulatov and Raskhodnikova [9] and Korman, Reichman, and the author [7] on testing of image properties (that is, on visual properties of 2D arrays) show that tests based on querying large consecutive blocks are useful for image property testing. In this work, the general queries we make are quite different: we query the *boundaries* of blocks of different sizes, so the queries are *spherical*, in the sense that a block can be seen as a ball in the L_∞-metric on vectors in $[n]^d$, while its boundary can be be seen as the (width-k) sphere surrounding this ball. This introduces a new type of queries shown to be useful for image property testing.

1.6 Discussion

Small alphabets

The results in this work are alphabet independent, and in particular, they work for alphabets over any size. An intriguing direction of research is to understand whether one can obtain more efficient general testability results for local properties of multi-dimensional arrays over smaller alphabets; this line of research has been conducted for specific properties of interest, like monotonicity and convexity [4, 29]. Note that the one-sided non-adaptive lower bound we prove here can be adapted to yield a $|\Sigma|^{\Omega(1)}$ lower bound for testing local properties over alphabets Σ of size smaller than n^d.

The most interesting special case is that of constant-sized (and in particular, binary) alphabets. Here, no lower bounds that depend on n are known. For the case $d = 1$, it is known that all $O(1)$-local properties are constant query testable; this follows from a result of Alon et al. [2], who showed that any regular language is constant-query testable. However, it is not known whether an analogous statement holds in higher dimensions. That is, for any $d > 1$, the question whether all k-local properties of $[n]^d$-arrays over $\{0, 1\}$ are ϵ-testable with query complexity that depends only on d, k, and ϵ, first raised in [7] (see also [1]), remains an intriguing open question. We believe that positive results in this front might also shed light on the question of obtaining more efficient inference for large classes of properties, especially over small alphabets.

Does adaptivity help?

This work does not provide any lower bounds for adaptive tests, and it will be interesting to do so; previously investigated properties likes monotonicity yield an $\Omega(d \log n)$ lower bound [13, 17], and we believe that "data flow" type properties, somewhat similar to our lower

bound constructions, can provide instances of 2-local properties that require at least n^c queries, for some constant $c \leq 1$, for the adaptive two-sided case.

However, it is not clear whether better lower bounds (even bounds of the type $\Omega(n^{1+c})$) exist. It will be very interesting to prove better upper and lower bounds for testing local properties. Our conjecture is that *any* 2-local property is testable in $n^{1+o(1)}g(d)$ queries (where $g(d)$ depends only on d), but proving a statement of this type might be very difficult.

Using the unrepairability framework in other contexts

In this work we show that the concept of unrepairability allows to unify and reprove many property testing results on one-dimensional arrays. What about multi-dimensional arrays? for example, can one generalize the currently known proofs for "bounded derivative" properties (including monotonicity and Lipschitz continuity) in d dimensions to a larger class of local properties?

Inference

As mentioned in Subsection 1.4.1, our test queries boundaries of block-like structures, and later infers whether each block is *unrepairable* (recall the definition from Subsection 1.4.1). The inference takes place without making any additional queries, and is based only on the property $\mathcal{P}$, the alphabet Σ, and the values of A in the boundary of the block.

The running time of the inference step is very large in general (although, as we have seen, in the 1D case it can be significantly improved using dynamic programming). The naive way to run the inference is by enumerating over all possible ways to fill the interior of the block, and checking whether each such possibility is indeed $\mathcal{F}$-free. The running time of this method is of order $|\Sigma|^{O(n^d)}$ in general for $d > 1$, and is exponential in n even if $|\Sigma| = 2$.

However, for many natural properties, inference can be done much more efficiently. For example, in monotonicity testing, the inference amounts to checking that no pair of boundary entries violates the monotonicity. Our lower bound constructions depict other properties where inference is efficient: it is not hard to show that the running time of inference in both cases is $O(k\epsilon^{-1/d}n^{d-1})$, which is sublinear in n^d for a wide range of parameters.

Thus, we believe that understanding inference better – including tasks such as characterizing properties in which inference can be done efficiently, and understanding the inference time of specific properties of interest – would be an interesting direction for future research.

1.7 Property testing notation

The property testing notation we use along the paper is standard. Given a property $\mathcal{P}$ of $[n]^d$-arrays over Σ, a proximity parameter $\epsilon > 0$, and query access to an unknown $[n]^d$-array A, a *two-sided error ϵ-test* must accept A with probability at least $2/3$ if A satisfies $\mathcal{P}$, and reject with probability $2/3$ if A is *ϵ-far* from $\mathcal{P}$ (meaning that the relative *Hamming distance* of A from $\mathcal{P}$ is at least ϵ, that is, we need to modify at least ϵn^d values in A to make it satisfy $\mathcal{P}$). A *one-sided error test* is defined similarly, but it must accept if A satisfies $\mathcal{P}$. A test is *non-adaptive* if it makes all of its queries in advance (prior to receiving any of the queried values), and *adaptive* otherwise.

Organization

In Sections 2, 3 and 4 we prove the upper bounds: Section 2 is devoted to the infrastructure needed for the proof, Section 3 presents a simple but non-optimal test, and finally, Section 4 presents the optimal test and proves Theorems 1 and 2.

Due to space considerations, the proofs of the lower bounds are given in Sections 5 and 6 of the full version of this paper [5].

2 The grid structure

In this section we present the grid-like structure in $[n]^d$ that we utilize for our tests.

▶ **Definition 5** (Interval partition)**.** A subset $I \subseteq [n]$ is an *interval* if its elements are consecutive, that is, if $I = \{x, x+1, \ldots, x+y\}$ for some $x \in [n]$ and $y \geq 0$. For any $\ell \geq 0$, we denote the set of the smallest ℓ elements of I by $I[:\ell]$ and also define $I[\ell+1:] = I \setminus I[:\ell]$. In the degenerate case that $|I| < \ell$, we define $I[:\ell]$ to be equal to I.

For $1 \leq w \leq n$, an (n,w)-*interval partition* is a partition of $[n]$ into a collection of disjoint intervals $\mathcal{I} = (I_1, \ldots, I_t)$ where the number of elements in each interval I_i is either w or $w+1$, and for any $i < j$, all elements of I_i are smaller than those in I_j.

▶ **Lemma 6.** *For any positive integer n and $0 \leq i \leq \log n$, there exists an $(n, \lfloor n/2^i \rfloor)$-interval partition $\mathcal{I}_i$ containing exactly 2^i intervals, so that the family $\{\mathcal{I}\}_{i=0}^{\lfloor \log n \rfloor}$ satisfies the following. For any $i > j$ and interval $I \in \mathcal{I}_i$, there exists an interval $I' \in \mathcal{I}_j$ satisfying $I \subseteq I'$.*

Proof. For any i define $n_i = \lfloor n/2^i \rfloor$; observe that $n_0 = n$ and $n_{i+1} = \lfloor n_i/2 \rfloor$ for any i. We prove the lemma by induction on i, starting by defining $\mathcal{I}_0 = ([n])$. Given $\mathcal{I}_i = (I_1^i, \ldots, I_{2^i}^i)$ in which all intervals are of length n_i or $n_i + 1$, we define $\mathcal{I}_{i+1}$ as follows. Each $I_j^i \in \mathcal{I}_i$, is decomposed into two intervals $I_{2j-1}^{i+1}, I_{2j}^{i+1}$ where $|I_{2j-1}^{i+1}|, |I_{2j}^{i+1}| \in \{n_{i+1}, n_{i+1}+1\}$, and all elements of I_{2j-1}^{i+1} are smaller than all elements of I_{2j}^{i+1}; observe that such a decomposition is indeed always possible. Now define $\mathcal{I}_{i+1} = (I_1^{i+1}, \ldots, I_{2^{i+1}}^{i+1})$. Clearly, the intervals of $\mathcal{I}_{i+1}$ satisfy the last condition of the lemma. ◀

In particular, we conclude that for any positive integer w and any $n \geq w$ there exists an integer $w/2 \leq w' \leq w$ for which an (n, w')-interval partition exists.

▶ **Definition 7** ((n,d,k,w)-grid)**.** Let $2 \leq w \leq n$ be integers for which an (n,w)-interval partition $\mathcal{I} = (I_1, \ldots, I_t)$ exists. For integers $2 \leq k \leq w$ and $d \geq 1$, the (d-dimensional) (n,d,k,w)-*grid* induced by $\mathcal{I}$ is the set

$$G = \left\{ (x_1, \ldots, x_d) \in [n]^d \;\middle|\; \exists i \in [d] \;\text{ such that } x_i \in \bigcup_{j=1}^{t} I_j[:k-1] \right\}.$$

We denote the family of all (n,d,k,w)-grids by $\mathcal{G}(n,d,k,w)$. As we have seen in Lemma 6, the family $\mathcal{G}(n,d,k,w)$ is non-empty for any $w = \lfloor n/2^i \rfloor$ satisfying $w \geq k$.

▶ **Definition 8** (G-block, Boundary, Closure)**.** Two tuples $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ in $[n]^d$ are considered *neighbors* if $\sum_{i=1}^{d} |x_i - y_i| = 1$. Given a grid $G \in \mathcal{G}(n,d,k,w)$, consider the neighborhood graph of non-grid entries, i.e., the graph whose set of vertices is $V = [n]^d \setminus G$ and two entries are connected if they are neighbors. A G-*block* B is a connected component of this graph, and the *closure* of B is

$$\overline{B} = \left\{ (x_1, \ldots, x_d) \in [n]^d \;\middle|\; \exists (y_1, \ldots, y_d) \in B \text{ such that } \forall i \in [d]\ |x_i - y_i| < k \right\}.$$

Note that $B \subseteq \overline{B}$. Define the *boundary* of the block B as $\partial B = \overline{B} \setminus B$.

The above notions can naturally be defined with Cartesian products. Recall that the Cartesian product of sets $X_1, \ldots, X_d$, denoted $\prod_{j=1}^{d} X_j$ or $X_1 \times \ldots \times X_j$, is the set of all tuples $(x_1, \ldots, x_d)$ with $x_j \in X_j$ for any $j \in [d]$. Let $G \in \mathcal{G}(n, d, k, w)$ be the grid induced by the interval partition $\mathcal{I} = (I_1, \ldots, I_t)$. It is not difficult to verify that any G-block B can be defined as a Cartesian product $B = \prod_{j=1}^{d} I_{i_j}[k\colon]$ for some intervals $I_{i_1}, \ldots, I_{i_d} \in \mathcal{I}$ (not necessarily different).

$\overline{B}$ and ∂B can also be defined accordingly, as we detail next. For k as above, define $\overline{I_i} = I_i \cup I_{i+1}[\colon k-1]$ for any $1 \leq i \leq t$, where we take $I_{t+1} = \emptyset$ for consistency. Also define $\partial I_i = \overline{I_i} \setminus I_i[k\colon] = I_i[\colon k-1] \cup I_{i+1}[\colon k-1]$. With these in hand, we have

$$B = \prod_{j=1}^{d} I_{i_j}[k\colon] \; ; \quad \overline{B} = \prod_{j=1}^{d} \overline{I_{i_j}} \; ; \quad \partial B = \bigcup_{j=1}^{d} \overline{I_{i_1}} \times \ldots \times \overline{I_{i_{j-1}}} \times \partial I_{i_j} \times \overline{I_{i_{j+1}}} \times \ldots \times \overline{I_{i_d}} \tag{1}$$

Recall that $|I_{i_j}| \in \{w, w+1\}$ for any j, implying that $\left|I_{i_j}[k\colon]\right| \leq w+2-k$ and $\left|\overline{I_{i_j}}\right| \leq w+k$. Also note that $\left|\partial I_{i_j}\right| \leq 2(k-1)$. Thus,

$$|B| \leq (w+2-k)^d \; ; \qquad |\overline{B}| \leq (w+k)^d \; ; \qquad |\partial B| \leq 2d(k-1) \cdot (w+k)^{d-1} \; , \tag{2}$$

where the inequality on $|\partial B|$ holds since each set in the union expression in (1) is of size at most $(2k-2)(w+k)^{d-1}$.

The following observation is a direct consequence of (1).

▶ **Observation 9.** *Let $G \in \mathcal{G}(n, d, k, w)$. The boundary of any G-block is contained in G.*

▶ **Lemma 10.** *For any $G \in \mathcal{G}(n, d, k, w)$, any width-$k$ subarray of an $[n]^d$-array intersects exactly one G-block B. Moreover, the subarray is contained in $\overline{B}$.*

Proof. Let $\mathcal{I} = (I_1, \ldots, I_t)$ be the interval partition inducing G. Suppose that the subarray S is in location $(a_1, \ldots, a_d)$ where $a_j \in I_{i_j}$ for some $i_1, \ldots, i_d$ not necessarily distinct. In other words, the set of entries in S is $\prod_{j=1}^{d} S_j$ where $S_j = \{a_j, a_j+1, \ldots, a_j+k-1\}$ for any $j \in [d]$. We argue that S is contained in $\overline{B}$, where $B = I_{i_1}[k\colon] \times \ldots \times I_{i_d}[k\colon]$: The fact that $a_j \in I_{i_j}$ implies that $a_j+1, \ldots, a_j+k-1 \in I_{i_j} \cup I_{i_j+1}[\colon k-1]$. It follows from (1) that $S \subseteq \overline{B}$. From Observation 9 we conclude that S does not intersect any block other than B, and it remains to show that S intersects B. Indeed, for any $1 \leq j \leq d$, the fact that $a_j \in I_{i_j}$ implies that one of the elements $a_j, \ldots, a_j+k-1$ must be contained in $I_{i_j}[k\colon]$. Denoting this element by b_j, we conclude that $(b_1, \ldots, b_j) \in S \cap B$. ◀

3 Testing with grid queries

In this section we prove the following upper bound for all k-local properties; its proof serves as a warm-up towards proving the main upper bound of Theorem 1.

▶ **Theorem 11.** *Any k-local property of $[n]^d$-arrays over any alphabet is ϵ-testable with one-sided error using no more than $2(d+1)n^{d-\frac{d}{d+1}}k^{\frac{d}{d+1}}\epsilon^{-\frac{1}{d+1}}$ non-adaptive queries.*

The upper bound of Theorem 11 is sublinear in the size of the array as long as $k/\epsilon^{1/d} = o(n)$. The rest of the section is dedicated to the proof of the theorem. We may assume that $k \leq \epsilon^{1/d} n/4$, as otherwise the expression in the statement of the theorem is larger than n^d and the proof follows trivially by querying all $[n]^d$ entries of the given input array. Under this assumption, it holds that $2k \leq n^{d/(d+1)}k^{1/(d+1)}\epsilon^{1/(d+1)}$.

▶ **Definition 12** (Unrepairable block)**.** Let A be an $[n]^d$-array over Σ, and let $G \in \mathcal{G}(n,d,k,w)$. A G-block B is $(\mathcal{P},A)$-*unrepairable* (or simply *unrepairable*, if $\mathcal{P}$ and A are clear from context) if any $[n]^d$-array A' over Σ that satisfies $A'(x) = A(x)$ for any $x \in \partial B$, including the case $A' = A$, contains an $\mathcal{F}$-copy in $\overline{B}$. Otherwise, the block B is said to be $(\mathcal{P},A)$-*repairable*.

Note that the (un)repairability of a block B is determined solely by the values of A on ∂B, and that an unrepairable block always contains an $\mathcal{F}$-copy. These two facts inspire the following lemma, which serves as the conceptual core behind the test of Theorem 11.

▶ **Lemma 13.** *Suppose that A is an $[n]^d$-array that is ϵ-far from satisfying a k-local property $\mathcal{P}(\mathcal{F})$, and let $G \in \mathcal{G}(n,d,k,w)$ where $w \geq k$. Then at least one of the following holds.*

- *There exists a $(\mathcal{P},A)$-unrepairable G-block.*
- *For at least an ϵ-fraction of the G-blocks B, there is an $\mathcal{F}$-copy in $\overline{B}$.*

Proof. Suppose that the first condition does not hold, that is, all G-blocks are $(\mathcal{P},A)$-repairable. By Lemma 10, every $\mathcal{F}$-copy is contained in the closure of some G-block.

Let $\mathcal{C}$ denote the collection of all G-blocks B such that A contains an $\mathcal{F}$-copy in $\overline{B}$. By the repairability, the values of A in each block $B \in \mathcal{C}$ can be modified so that after the modification, A will not contain an $\mathcal{F}$-copy in $\overline{B}$. We stress that the modifications for each block B appear only in B itself and do not modify entries on the grid, so by Lemma 10, they cannot create new $\mathcal{F}$-copies in the closure of other blocks.

After applying all of the above modifications to A, we get an $\mathcal{F}$-free array, i.e., an array that satisfies $\mathcal{P}$. A was initially ϵ-far from $\mathcal{P}$, and the number of entries in each block is bounded by $(w+2-k)^d \leq w^d$, implying that at least an ϵ-fraction of the blocks belong to $\mathcal{C}$. ◀

Proof of Theorem 11. We may assume that $k^d/\epsilon \leq n^d/2$, otherwise our test may trivially query all n^d entries of A. Our (non-adaptive) test T picks $W = \lfloor n^{d/(d+1)} k^{1/(d+1)} \epsilon^{1/(d+1)} \rfloor \geq 2k$, and an integer w satisfying $k \leq W/2 \leq w \leq W$, for which an (n,w)-interval partition exists. T now makes the following queries.

1. T queries all entries of an arbitrarily chosen grid $G \in \mathcal{G}(n,d,k,w)$. The number of entries in any grid is at most $dn^d(k-1)/w \leq 2dn^{d-\frac{d}{d+1}} k^{\frac{d}{d+1}} \epsilon^{-\frac{1}{d+1}}$.
2. T chooses a collection $\mathcal{B}$ of $2/\epsilon$ G-blocks uniformly at random and queries all entries in these blocks. Since each block contains at most $(w+2-k)^d \leq W^d$ entries, the total number of queries is bounded by $2W^d/\epsilon \leq 2n^{d-\frac{d}{d+1}} k^{\frac{d}{d+1}} \epsilon^{-\frac{1}{d+1}}$. Note that the boundaries of all blocks are queried in the first step (since they are contained in the grid). Thus, for any block $B \in \mathcal{B}$, the test queries all entries of $\overline{B}$.

The total number of queries in the above two steps is $2(d+1)n^{d-\frac{d}{d+1}} k^{\frac{d}{d+1}} \epsilon^{-\frac{1}{d+1}}$.

After querying all entries of the grid (and in particular, the whole boundaries of all of the blocks), T can determine for every G-block B whether it is $(\mathcal{P},A)$-unrepairable or not. T rejects if at least one of the blocks is unrepairable or if it found an $\mathcal{F}$-copy in $\overline{B}$ for some $B \in \mathcal{B}$, and accepts otherwise. The test has one-sided error, since an unrepairable block must contain an $\mathcal{F}$-copy. In view of Lemma 13, T rejects arrays A that are ϵ-far from $\mathcal{P}$ with probability at least $2/3$: If A satisfies the first condition of Lemma 13, then T always rejects. If the second condition holds, the probability that none of the $2/\epsilon$ closures $\overline{B}$ for $B \in \mathcal{B}$ contains an $\mathcal{F}$-copy is bounded by $(1-\epsilon)^{2/\epsilon} < e^{-2}$, so T rejects with probability at least $1 - e^{-2} > 2/3$. ◀

4 Systems of grids and testing with spherical queries

In this section we prove Theorems 1 and 2. We do so by considering a system of grids with varying block sizes, defined as follows.

▶ **Definition 14.** Let $d > 0$ and $2 \leq k \leq w \leq n$ be integers. An (n, d, k, w)-*system of grids* is an $(r+1)$-tuple $(G_0, G_1, \ldots, G_r)$ of grids, for $r(n, w) = \lfloor \log(n/w) \rfloor$, so that

- $G_i \in \mathcal{G}(n, d, k, \lfloor n/2^{r-i} \rfloor)$ for any $0 \leq i \leq r$.
- $G_0 \supseteq G_1 \supseteq \ldots \supseteq G_r$ (as subsets of $[n]^d$). In particular, for any $i < j \leq r$, any G_i-block B is contained in a G_j-block B', and we say that B' is an *ancestor* of B. Specifically, the G_{i+1}-block containing B is called the *parent* of B and denoted by $\mathrm{Par}(B)$. For the only G_r-block, B_r, we define $\mathrm{Par}(B_r)$ as the whole domain $[n]^d$.

$r(n, w)$ was chosen so that $w \leq n/2^r < 2w$, making G_0 a $\mathcal{G}(n, d, k, w')$-grid for $w \leq w' < 2w$. As we shall see, when working with such a system, unrepairability of blocks can be handled in a query-efficient way. The following lemma asserts that such a system of grids exists for any suitable choice of parameters.

▶ **Lemma 15.** *An (n, d, k, w)-system of grids exists for all $d > 0$ and $2 \leq k \leq w \leq n$.*

Proof. Consider the family of interval partitions $\mathcal{I}_0, \ldots, \mathcal{I}_{\lfloor \log n \rfloor}$ obtained by Lemma 6. For each $0 \leq i \leq r(n, w)$ define G_i as the $(n, d, k, \lfloor n/2^{r-i} \rfloor)$-grid induced by $\mathcal{I}_{r-i}$. It is not hard to verify that $(G_0, \ldots, G_r)$ satisfies all requirements of an (n, d, k, w)-system of grids. ◀

For the rest of the section, fix a k-local property $\mathcal{P}(\mathcal{F})$ of $[n]^d$-arrays over Σ, as well as an $[n]^d$-array A over Σ. Consider an (n, d, k, w)-system of grids $(G_0, \ldots, G_r)$ constructed as described in the proof of Lemma 15, where w will be determined later. (For now it suffices to require, as usual, that $2 \leq k \leq w \leq n$.)

We say that a G_i-block B is a $(\mathcal{P}, A)$-*witness* if one of the following holds.

- $i = 0$ and the array A contains an $\mathcal{F}$-copy in the closure $\overline{B}$.
- $i > 0$ and B is $(\mathcal{P}, A)$-unrepairable.

Recall that the closure of unrepairable blocks cannot be $\mathcal{F}$-free, so the closure of any witness block contains an $\mathcal{F}$-copy. We say that a witness block B is *maximal* if all of its ancestors are not witnesses, that is, they are repairable.

▶ **Observation 16.** *Any $(\mathcal{P}, A)$-witness is contained in a maximal $(\mathcal{P}, A)$-witness.*

We define the *maximal witness family* $\mathcal{W}$ as the set of all maximal $(\mathcal{P}, A)$-witness blocks. Obviously, the blocks in $\mathcal{W}$ might come from different G_i's

▶ **Observation 17.** *$B_1 \cap B_2 = \emptyset$ for any two blocks $B_1, B_2 \in \mathcal{W}$.*

▶ **Lemma 18.** *All $\mathcal{F}$-copies in A are fully contained in $\bigcup_{B \in \mathcal{W}} \overline{B}$.*

Proof. Let F be an $\mathcal{F}$-copy in A. By Lemma 10, F is contained in the closure of a unique G_0-block B_F; hence, B_F is a $(\mathcal{P}, A)$-witness. From Observation 16 we have $B_F \subseteq B'$ for some maximal $(\mathcal{P}, A)$-witness B'. We conclude that $F \in \overline{B_F} \subseteq \overline{B'}$. ◀

▶ **Lemma 19.** *One can make A satisfy $\mathcal{P}$ by only modifying entries of A in $\bigcup_{B \in \mathcal{W}} \mathrm{Par}(B)$.*

Proof. Fix $B \in \mathcal{W}$. B is a maximal $(\mathcal{P}, A)$-witness, so $\mathrm{Par}(B)$ is repairable.[1] Thus, One can make $\overline{\mathrm{Par}(B)}$ $\mathcal{F}$-free by only modifying entries inside $\mathrm{Par}(B)$. By Lemma 10,

[1] Note that when $B = B_r$ is the maximal witness considered, $\mathrm{Par}(B)$ is $[n]^d$; the latter is repairable for any non-empty property.

width-k subarrays that are not fully contained in $\overline{\mathrm{Par}(B)}$ are left unchanged. Therefore, this modification does not create any new $\mathcal{F}$-copies in A. Seeing that all $\mathcal{F}$-copies in A are originally contained in $\bigcup_{B\in\mathcal{W}} \overline{B} \subseteq \bigcup_{B\in\mathcal{W}} \overline{\mathrm{Par}(B)}$, applying these modifications for all $B \in \mathcal{W}$ deletes all $\mathcal{F}$-copies in A without creating new ones, so in the end of the process A satisfies $\mathcal{P}$. ◀

We may assume that $k \leq \epsilon^{1/d}n/10$, as otherwise the expression in the theorem is $\Omega(n^d)$. We choose $w = 2k$, working with an $(n, d, k, 2k)$-system of grids from now on. A very useful consequence of this choice of w is that here the parent of a block B cannot be much larger than B itself.

▶ **Lemma 20.** *Let $(G_0, G_1, \ldots, G_r)$ be an $(n, d, k, 2k)$-system of grids. Then for any $0 \leq i \leq r$ and any G_i-block B it holds that $|Par(B)|/|B| < 3^d$.*

Proof. For $i = r$ this is trivial. Now fix $i < r$ and let B be a G_i-block. Recall that, following (1), one can write $B = \prod_{j=1}^{d} I_{i_j}[k\colon]$ where each interval I_{i_j} (for $j \in [d]$) is of size at least $2k \geq 4$. On the other hand, we can also write $\mathrm{Par}(B) = \prod_{j=1}^{d} I'_{i'_j}[k\colon]$ where $I'_{i'_j} \supseteq I_{i_j}$ for any $j \in [d]$. It is not hard to verify that $|I'_{i'_j}| \leq 2|I_{i_j}| + 1$ most hold, and so

$$\frac{|\mathrm{Par}B|}{|B|} = \prod_{j=1}^{d} \frac{|I'_{i'_j}| - (k-1)}{|I_{i_j}| - (k-1)} \leq \prod_{j=1}^{d} \frac{2|I_{i_j}| + 1 - (k-1)}{|I_{i_j}| - (k-1)} \leq \left(\frac{2\cdot 2k - k + 2}{2k - k + 1}\right)^d < 3^d$$

where the second inequality holds since $|I_{i_j}| \geq 2k$ for any j. ◀

The next corollary follows immediately from Lemmas 19 and 20.

▶ **Corollary 21.** *Suppose that A is ϵ-far from $\mathcal{P}$. Then the total number of entries in the blocks of $\mathcal{W}$ is at least $\epsilon(n/3)^d$.*

We are now ready for the proof of the main upper bound of this paper, Theorem 1.

4.1 Proof of Theorem 1

As before, we may assume that $k \leq \epsilon^{1/d}n/10$. For larger k, the expression in the theorem dominates n^d and thus becomes trivial. Consider the $(n, d, k, 2k)$-system of grids $(G_0, G_1, \ldots, G_r)$ mentioned above. For any $0 \leq i \leq r$, define $\delta_i = |\mathcal{B}_i \cap \mathcal{W}|/|\mathcal{B}_i|$, where $\mathcal{B}_i$ is the set of all G_i-blocks. In other words, δ_i is the fraction of maximal witnesses among the G_i-blocks. By Corollary 21, if A is ϵ-far from $\mathcal{P}$ then $\sum_{i=0}^{r} \delta_i \geq \epsilon/3^d$. Define $r' = \lfloor \log(\epsilon^{1/d}n/k) \rfloor \geq 1$, noting that $G_{r'} \in \mathcal{G}(n, d, k, w_{r'})$ with $w_{r'} \geq 2k \cdot 2^{r'} \geq \epsilon^{1/d}n$. Thus, the total number of blocks in $\mathcal{B}_{r'}$ is bounded by $(n/w_{r'})^d \leq 1/\epsilon$.

The test

We iterate the following basic step $2 \cdot 3^d/\epsilon$ times.

1. Pick $B \in \mathcal{B}_0$ uniformly at random and query all entries of $\overline{B}$.
2. For any $1 \leq i \leq r'$, pick $B \in \mathcal{B}_i$ uniformly at random and query all entries of $\bigcup_{B\in\mathcal{Q}_0} \partial B$.

Finally, the test rejects if and only if at least one of the blocks B picked during the process is a $(\mathcal{P}, A)$-witness. (Recall that querying all boundary entries of a G_i-block for $i > 0$ suffices to determine whether it is unrepairable, and thus a witness.)

The test is clearly non-adaptive, and has one-sided error: It only rejects if it finds a witness. As we have seen earlier, all witnesses contain an $\mathcal{F}$-copy.

The test is *canonical* in the following sense. The choice of queries in every basic step depends only on n, d, k, and (weakly) on ϵ, and is independent of the property $\mathcal{P}$ or the alphabet Σ. To determine which entries constitute a block, it suffices to know the parameters of the block, that depend only on n, d, k; the dependence in ϵ is only taken into account in the choice of r'. The test only considers $\mathcal{P}$ in order to determine whether each queried block is a witness.

Analysis

Suppose that A is ϵ-far from $\mathcal{P}$. If $\delta_i > 0$ for some $i > r'$ then it must hold that $\delta_{r'} > 0$ as well (since any unrepairable G_i-block most contain an unrepairable $G_{i'}$-block for any $i' < i$). By the choice of r', we must have $\delta_{r'} \geq 1/|\mathcal{B}_{r'}| \geq \epsilon$ in this case. If the above doesn't hold, then $\delta_i = 0$ for any $i \geq r'$, implying that $\sum_{i=0}^{r'-1} \delta_i \geq \epsilon/3^d$. Therefore, in both cases, we have $\sum_{i=0}^{r'} \delta_i \geq \epsilon/3^d$.

The probability that a random $\mathcal{B}_i$-block is a witness is at least δ_i, and therefore the probability that a single basic step leads to a rejection of A is at least $\sum_{i=0}^{r'} \delta_i \geq \epsilon/3^d$. Running $2 \cdot 3^d/\epsilon$ independent iterations of the basic step ensures that the test will accept A with probability at most $(1 - \epsilon/3^d)^{2 \cdot 3^d/\epsilon} \leq e^{-2} < 2/3$, as desired.

Query complexity

For $d = 1$, the query complexity of each basic step is $O(kr')$: The test queries $\overline{B}$ for a single block $B \in \mathcal{B}_0$, and the boundaries of r' larger blocks. Considering the parameters of our system of grids, we have $|B| \leq 4k$ and so $|\overline{B}| < 6k$. On the other hand, the boundary of each of the larger blocks is of size at most $2k - 2$. Therefore, the total query complexity for the 1D test is $O(kr'/\epsilon) = O\left(\frac{k}{\epsilon} \log(\epsilon n/k)\right)$ as desired.

For $d > 1$, consider a single basic step, and for any $0 \leq i \leq r'$ let $B_i \in \mathcal{B}_i$ be the $\mathcal{G}_i$-block picked in this step. From (2) we have $|\overline{B_0}| \leq (6k)^d$, while for any $i > 0$ we have $|\partial B_i| \leq 2d(k-1)(4k \cdot 2^i + k)^{d-1} = O(d \cdot (4k)^d \cdot 2^{(d-1)i})$. Note that the last expression grows exponentially with $(d-1)i$, so the total number of queries in a single basic step is $O((6k)^d + d \cdot (4k)^d 2^{(d-1)r'})$. Plugging in r', we have $2^{(d-1)r'} = \frac{\epsilon^{(d-1)/d}}{k^{d-1}} n^{d-1}$. As the test runs $O(3^d/\epsilon)$ iterations of the basic step, we conclude that the total query complexity is bounded by $c^d k \epsilon^{-1/d} n^{d-1}$ for an absolute constant $c > 0$, completing the proof of Theorem 1.

4.2 Proximity oblivious test

The proof of Theorem 2 follows by a very simple modification of the proof of Theorem 1. The desired proximity oblivious test (POT) is the so called "basic step" from the above test, with r replacing r' (since r' depends on ϵ). The POT rejects if it infers that one of the blocks queried is a witness, like the above test. Its query complexity is $O(kr) = O(k \log n/k)$ for $d = 1$. In the case $d > 1$, the query complexity is dominated by the size of ∂B_r, which is bounded by $O(dkn^{d-1})$.

Clearly this POT has one-sided error, and its queries do not depend on the property $\mathcal{P}$ and the alphabet Σ (on the other hand, they do depend on n, d, k). Using the notation of the previous subsection and denoting by ϵ_A the Hamming distance of a given input A from $\mathcal{P}$, we get (exactly as in the beginning of Subsection 4.1) a rejection probability of at least $\sum_{i=0}^{r} \delta_i \geq \epsilon_A/3^d$ for A, which is linear in ϵ_A for fixed d. This concludes the proof.

References

1. N. Alon, O. Ben-Eliezer, and E. Fischer. Testing Hereditary Properties of Ordered Graphs and Matrices. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 848–858, 2017.
2. N. Alon, M. Krivelevich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. *SIAM Journal on Computing*, 30:1842–1862, 2001.
3. P. Awasthi, M. Jha, M. Molinaro, and S. Raskhodnikova. Testing Lipschitz functions on hypergrid domains. *Algorithmica*, 74:1055–1081, 2016.
4. A. Belovs. Adaptive Lower Bound for Testing Monotonicity on the Line. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 31:1–31:10, 2018.
5. O. Ben-Eliezer. Testing local properties of arrays. *Electronic Colloquium on Computational Complexity (ECCC)*, 2018. full version. URL: `https://eccc.weizmann.ac.il/report/2018/196/`.
6. O. Ben-Eliezer and E. Fischer. Earthmover Resilience and Testing in Ordered Structures. In *Proceedings of the 33rd Conference on Computational Complexity (CCC)*, pages 18:1–18:35, 2018.
7. O. Ben-Eliezer, S. Korman, and D. Reichman. Deleting and Testing Forbidden Patterns in Multi-Dimensional Arrays. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 9:1–9:14, 2017.
8. P. Berman, M. Murzabulatov, and S. Raskhodnikova. Testing convexity of figures under the uniform distribution. In *Proceedings of the 32nd International Symposium on Computational Geometry (SoCG)*, pages 17:1–17:15, 2016.
9. P. Berman, M. Murzabulatov, and S. Raskhodnikova. Tolerant testers of image properties. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 90:1–90:14, 2016.
10. P. Berman, S. Raskhodnikova, and G. Yaroslavtsev. L_p-testing. In *Proceedings of the 46th ACM Symposium on the Theory of Computing (STOC)*, pages 164–173, 2014.
11. E. Blais and A. Bommireddi. Testing submodularity and other properties of valuation functions. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 33:1–33:17, 2017.
12. E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21:311–358, 2012.
13. E. Blais, S. Raskhodnikova, and G. Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *Proceedings of the 29th Conference on Computational Complexity (CCC)*, pages 309–320, 2014.
14. D. Chakrabarty. Monotonicity testing. *Encyclopedia of Algorithms*, pages 1352–1356, 2016.
15. D. Chakrabarty, K. Dixit, M. Jha, and C. Seshadhri. Property testing on Product Distributions: Optimal Testers for Bounded Derivative Properties. *ACM Transactions on Algorithms*, 13:20:1–20:30, 2017.
16. D. Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proceedings of the 45th ACM Symposium on the Theory of Computing (STOC)*, pages 419–428, 2013.
17. D. Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014.
18. X. Chen, A. Freilich, R. Servedio, and T. Sun. Sample-based high-dimensional convexity testing. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 37:1–37:20, 2017.
19. F. Ergün, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60:717–751, 2000.

20 E. Fischer. On the strength of comparisons in property testing. *Information and Computation*, 25:107–116, 2004.
21 O. Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.
22 O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998.
23 O. Goldreich and D. Ron. On proximity-oblivious testing. *SIAM Journal on Computing*, 40:534–566, 2011.
24 M. Jha and S. Raskhodnikova. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM Journal on Computing*, 42:700–731, 2013.
25 S. Korman, D. Reichman, G. Tsur, and S. Avidan. Fast-Match: Fast Affine Template Matching. *International Journal of Computer Vision*, 121:111–125, 2017.
26 S. Moriguchi and K. Murota. On discrete Hessian matrix and convex extendability. *Journal of the Operations Research Society of Japan*, 55:48–62, 2012.
27 K. Murota. *Discrete Convex Analysis*. Society for Industrial and Applied Mathematics, 2003.
28 I. Newman and C. Sohler. Every property of hyperfinite graphs is testable. *SIAM Journal on Computing*, 42:1095–1112, 2013.
29 R. Pallavoor, S. Raskhodnikova, and N. Varma. Parameterized property testing of functions. *ACM Transactions on Computation Theory*, 9:17:1–17:19, 2018.
30 M. Parnas, D. Ron, and R. Rubinfeld. On testing convexity and submodularity. *SIAM Journal on Computing*, 32:1158–184, 2003.
31 L. Rademacher and S. Vempala. Testing geometric convexity. In *Foundations of Software Technology and Theoretical Computer Science Conference (FSTTCS)*, pages 469–480, 2004.
32 S. Raskhodnikova. Testing if an array is sorted. *Encyclopedia of Algorithms*, pages 2219–2222, 2016.
33 R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25:252–271, 1996.
34 C. Seshadhri and Jan Vondrák. Is Submodularity Testable? *Algorithmica*, 69:1–25, 2014.
35 A. C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.