The art of uninformed decisions
A primer to property testing

Eldar Fischer [*]

Abstract

Property testing is a new field in computational theory, that deals with the information that can be deduced from the input where the number of allowable queries (reads from the input) is significantly smaller than the input size. This survey provides an introduction and reference to this exciting field.

Dedicated to the memory of the victims of terror, September 11, 2001.

1 Motivation and introduction

With the recent advances in technology we are faced with the need to process increasingly larger amounts of data in faster times. As was observed by Papadimitriou [47, Page 357], common notions of the efficiency of algorithms tend to gravitate towards stricter time restrictions as computer sciences progress. At first a problem was considered computable if there was an algorithm that could decide it in a finite time given any input instance. Afterwords came the notion of polynomial time computations, and later the possibility of making a computation faster for certain problems through use of parallel machines was also investigated.

In all of the classes considered above, however, the algorithms involved still face the obvious obstacle of having to read the entire input prior to its assessment (in the parallel setting it is assumed that the input can also be read in parallel, which is not always a realistic assumption).

There are practical situations in which the input is so large, that even taking a linear time in its size to provide an answer is too much. Many modern databases have overwhelming sizes. There

[*]NEC Research Institute, 4 Independence Way, Princeton NJ 08540, USA; and DIMACS.
Current address: Faculty of Computer Science, Technion – Israel institute of technology, Technion City, Haifa 32000, Israel. Email: eldar@cs.technion.ac.il

are also other instances where the input is not easily accessible – it could be stored for example on a server at the other end of a slow communication line. And it could also be the case that an explicit representation of the input in fact does not exist, and an "oracle" procedure that calculates its values in the requested locations is given instead.

This survey deals with algorithms that are designed to make a decision concerning the input after reading only a small portion thereof. In the general setting it is assumed that an algorithm has random access to the input, that is, the algorithm is told in advance what is the input size, and is supplied with an oracle that for a query 'i' yields the value v_i of the input in this location. The goal is to minimize the number of calls to this oracle, and in the ideal case to make it a constant independent of the actual input size.

One necessary feature of such algorithms, that is immediately apparent, is that they must be probabilistic. The intuition is that even when not the whole input is actually read, all of the input must still be "considered" by the algorithm in order to give an output relevant to its entirety, and thus every component of the input must be read with some positive probability.

But even random algorithms cannot be expected to give accurate results. Consider for example the simple property of the input being all zeros. No algorithm reading only a small fraction of the input can distinguish with high probability between an input satisfying this property, and an input containing a single '1' in a randomly chosen location. In fact, there are some well known conjectures and results stating that most properties are "evasive", requiring a linear number of queries to decide them, even using a probabilistic algorithm (evasiveness was studied extensively for graph properties; for the interested reader [45] provides an introduction to this subject).

It follows that in order to describe what an algorithm can do without reading the whole input, it is required to formulate a new notion of an approximated version of the original decision problem. As a guide we look at what can be done with the above property of the input being all zeros – while even this property cannot be decided by such an algorithm, it is very easy, using a constant time randomized algorithm, to distinguish between the case that the input is all zeros, and the case that at least an ϵ fraction of the places in the input contain ones (where ϵ is any constant independent of the input size). The algorithm needs only to take $O(\epsilon^{-1})$ uniformly random samples of the input, and reject the input if it finds a '1' in any of them.

The following definition of a property's approximation suffices for most instances of testing problems. It deals with a fixed input size – since testing deals primarily with informational bounds it is usually sufficient to analyze every fixed input size separately (and apart from rarely occurring uniformity issues, it is almost the same as saying that the testing algorithm is given the size of the input in advance).

Definition 1 *An input, given as a function $f : \mathcal{D} \to F$, is said to be* ϵ-close to satisfying a property *P, if there exists a function $f' : \mathcal{D} \to F$ that differs from f in no more than $\epsilon|\mathcal{D}|$ places, and satisfies P. An input which is not ϵ-close to satisfying P is called* ϵ-far from satisfying P.

There are some testability questions that deal with different notions of closeness, such as these defined using the edit distance between inputs when considered as strings; such a case appears in Section 6.

The notion of closeness leads to the following definition of what it means to ϵ-test a property. Remember that in this definition a read from the input, also called a *query*, is allowed from any location (in other words, the algorithm is granted random access to the input).

Definition 2 *Let P be a property, and let us fix the input size n. An* ϵ-test with $q = q(\epsilon, n)$ queries *for P is a probabilistic algorithm that reads the input in up to q places, and with probability at least $\frac{2}{3}$ distinguishes between the case that the input satisfies P, and the case that the input is ϵ-far from satisfying P.*

The '$\frac{2}{3}$' term in the definition is arbitrary of course, and can be amplified to any fixed probability smaller than 1, at the price of a linear increase in q, by running the algorithm a constant number of times and taking the majority vote.

Note that the above definition claims nothing about the running time of the algorithm, but deals only with the amount of data that is allowed to be read from the input. This makes property testing more related to the branches of computer science dealing with informational complexity, rather than to those dealing with computational complexity. However, in most cases the algorithms making decisions based on a small portion of the input happen to have a small running time as well, making them also suitable for providing fast approximations of the input qualities.

A property that has an ϵ-test as above is called *ϵ-testable with q queries.* The best that can be hoped for with regards to large inputs is the situation where q depends only on ϵ and not on n. If this occurs for every fixed $\epsilon > 0$, then the property P is usually simply referred to as *testable.* Lowering the dependency on ϵ is also important, in which case the best that can be hoped for is a number of queries that is linear in ϵ^{-1}, as is the case with the property of the input being all zeros.

Sometimes lower bounds on the number of queries can be proven as well. There is no "official" standard as to what makes a property "totally non-testable". In many cases involving inputs over a binary alphabet, any non-constant lower bound for a fixed ϵ is said to make the property non-testable. However, sometimes testing with $O(\log n)$ queries (for a fixed ϵ) is also considered a sufficient test, especially in cases, such as most problems dealing with inputs over infinite alphabets, where there are some known $\Omega(\log n)$ lower bounds.

A lower bound of the type n^c (for some positive constant c and ϵ) is sometimes considered the final say, but in several instances even testing algorithms that read this many queries are considered interesting, especially if they have a sublinear running time to match.

Sometimes, special types of testing algorithms are sought for. For example, by the definition above the algorithm is allowed to have a 2-sided error probability. A 1-*sided* testing algorithm is an algorithm that in addition accepts with probability 1 any input that satisfies the property to be tested.

In other cases the adaptivity of the algorithm is also considered. A *non-adaptive* test for a property is a testing algorithm that specifies all queries in advance (according to some distribution), and only then is allowed to obtain the input values in the corresponding locations, based on which the algorithm accepts or rejects the input. In other words, a non-adaptive test cannot use the results of previous queries in the decision as to where to make the next query, but only in the final decision as to whether to accept or reject the input.

The rest of this survey is organized as follows. Section 2 deals with the testing of a property defined in terms of polynomial functions, in the context of the works that have started this field. Section 3 deals with the testing of languages, motivated by an attempt to correlate the notion of testing to other notions of low complexity (as it turns out there is little obvious correlation, and some surprising results come into play). Section 4 and Section 5 deal with one model of graph property testing, and another angle of the attempt to give a logical characterization of the testable languages, and Section 6 deals with another graph testing model. Section 7 deals with the testing of properties motivated by the notion of monotonicity. Finally, Section 8 deals with some techniques for proving lower bounds on testability, and Section 9 deals with some recent trends and future research directions, as well as the relation of property testing to other old and new topics in computer science.

There is no hope to give complete proofs for all the results presented in a survey such as this. The strategy used in the following is to give proofs for some of the easier results, and to give proofs of special cases or proof outlines for some of the harder results. It is hoped that through this the reader can get a taste of the techniques used in the field.

Although in general older results tend to appear earlier in this survey, there is no serious attempt to fully reflect the chronological development of this field. For example, many of the results presented in Section 4 have appeared earlier than those presented in Section 3.

2 A first offering – testing for polynomiality

The first results in property testing were motivated by program testing (Blum, Luby and Rubinfeld [16], and Rubinfeld and Sudan [54]), and by probabilistically checkable proofs (Arora and Safra [8], and Arora, Lund, Motwani, Sudan and Szegedy [7]).

In the context of program testing, the idea is that the "input" to be tested is not explicitly given. Instead, a program is given; this program is considered as a function from the set of its possible inputs to the set of its possible outputs, and we have to test for properties of this function. The practical implication is clear – one cannot hope to explicitly write down the entire function, but a "query" into it can be answered in a reasonable time by running the given program on the input that corresponds to the queried "location". Thus testing can ensure that this program will perform within the expected parameters for most inputs.

Probabilistically checkable proofs deal with proving protocols that can be probabilistically verified by reading only a small portion of a proposed proof. Formulating the property of being an input that satisfies the protocol, in a way that allows testing for this, is the connection here. Both the context of program checking and the context of probabilistically checkable proofs place additional requirements on the procedure beyond testability (that happen to be achievable for the example below); their exact definition is outside the scope of this testing survey.

The first result regarding testability per-se is probably that of Blum, Luby and Rubinfeld [16] concerning linearity, and the first explicit definition of property testing was given by Rubinfeld and Sudan in [54] (in its turn it is based on some of the results of [55] and [34]). The first results dealt mainly with the algebraic notion of being a low degree polynomial.

Suppose that for some finite field $\mathcal{F}$ the input is given as a function f from $\mathcal{F}$ to $\mathcal{F}$, that a query consists of finding the value of f at a specified location $x \in \mathcal{F}$, and that the distance is measured by the number of values of f that have to be changed in order to make it satisfy the property to be tested, divided by the input size $|\mathcal{F}|$.

In this setting, it is easy to test using a few queries for the property that f is a polynomial whose degree is bounded by some fixed constant k.

Proposition 2.1 *It takes $O(k + \epsilon^{-1})$ queries to ϵ-test that a function f from $\mathcal{F}$ to $\mathcal{F}$ (where $\mathcal{F}$ is a finite field of arbitrary size) is a polynomial of degree at most k*

Proof: The algorithm first queries f in $k+1$ arbitrary different locations. If f is indeed a polynomial of degree at most k, then by the interpolation property of polynomials these $k+1$ queried values determine all the other values of f.

The algorithm then queries f in $O(\epsilon^{-1})$ additional locations chosen uniformly and independently at random, and for each of these checks that the value of f indeed agrees with the value of the polynomial interpolated from the first $k+1$ queried values. ■

In cases where functions from $\mathcal{F}^d$ to $\mathcal{F}$ are needed to be tested for the property of being low-degree (multivariate) polynomials, interpolation by itself is not satisfactory. The main results of [54] are another test for the 1-dimensional case that satisfies some additional conditions mandated by the context of program checking, as well as an efficient test for the multidimensional case (see also [8] and [7] for related results and applications). The way to handle functions from $\mathcal{F}^d$ to $\mathcal{F}$ is by considering two random intersecting lines in $\mathcal{F}^d$, and testing that the restrictions of the input function to these give degree k polynomials that agree on the intersection point.

Theorem 2.2 ([54]) *The property of a function $f : \mathcal{F}^d \to \mathcal{F}$ being a polynomial of a total degree bounded by k can be ϵ-tested using $O(k^2 + k\epsilon^{-1})$ queries (independently of $|\mathcal{F}|$ and d).*

There was extensive research in the topic of testing a function for being a low degree polynomial, and related questions. The interested reader is referred to [9], that describes some additional results.

3 Testing of languages

As the possibility of testing an input for a property using few queries becomes apparent, the next natural step is to correlate the traditional categorization of properties into computational complexity classes with the new testing scheme.

An ultimate goal would be to try finding a logical characterization of all properties testable with a number of queries independent of the input size, in much the same manner as, say, that the properties recognizable by a finite automaton have a characterization as all properties expressible in the appropriate monadic second order logic language.

This goal is far from achieved. A more humble goal is to identify whole classes, and not just individual properties, that are efficiently testable. Such classes, sometimes identified by the structure of their expression in an appropriate first order logic language, are given special attention in this survey.

Results are sometimes surprising. On one hand, there exist properties that are in AC_0 and are not testable even in a number of queries that is some fixed (small) power of the input size. On the other hand, there exist NP-complete properties, such as graph 3-colorability, that are testable with a number of queries that depends only on the approximation parameter ϵ. A reason for this is that up to changing a small portion of the input, even simple first order logic expressions can capture surprisingly large classes of properties. For example, up to changing a small number of edges, 3-colorability is in fact indistinguishable from some very simple AC_0 property. More on this in Section 5.

For a potential source of classes of properties that are testable we turn first to the well-known classes of languages with low computational complexity, starting with the lowest – the languages recognizable by a finite automaton. These were investigated by Alon, Krivelevich, Newman and Szegedy [5] and were found to be testable. However, in general one cannot prove similar testability results for complexity classes substantially broader than this, as there are already context free languages that are not testable even with relatively many queries – for example, the language of all possible concatenations of two palindromes is such a language.

Theorem 3.1 ([5]) *Any regular language is testable with a number of queries linear in $\epsilon^{-1}\log^3(\epsilon^{-1})$ and independent of the input size. On the other hand, there exist context-free languages that are not testable in a number of queries that is less than some fixed power of the input size.*

To give a taste of how such a testability result can be proven, a testability proof for a special case is given here.

Proposition 3.2 *Let S be any fixed binary string. The property of a binary input string not containing S as a (not necessarily continuous) substring is ϵ-testable, with a number of queries linear in ϵ^{-1} and independent of the input size.*

Proof: We construct by induction on the length of S a test whose error probability is bounded by $\sum_{i=1}^{|S|} 2^{-i-2} < \frac{1}{4}$. When S is of size 1 the test is trivial (it just performs a random sampling of the input and rejects if it finds an instance of the single character that comprises S).

Assume without loss of generality that the first character in S is '1', and let S' be the result of removing this first character of S. Let $v_1 v_2 \ldots v_n$ be an input of size n. The algorithm then chooses uniformly and independently $O(\epsilon^{-1})$ random locations in the input string (the 'O' notation hides a coefficient dependent only on $|S|$), queries them, and notes the smallest queried location index for which the input is '1'. Denote it by i_1.

In the case that no '1' was found in the previous stage, the algorithm accepts the input. This will happen only with a small probability (which can be made smaller than $2^{-|S|-2}$), unless the input is ϵ-close to being all '0', in which case it is clearly ϵ-close to not containing S as a substring.

With high probability, if a '1' was found in the preceding stage, then i_1 is such that $v_1 \ldots v_{i_1-1}$ contains no more than $\frac{1}{2}\epsilon n$ instances '1'. To see this, denote by $\{j_1, \ldots, j_{\epsilon n/2}\} \subset \{1, \ldots, n\}$ the set of the first $\frac{1}{2}\epsilon n$ instances of '1' in the input (if such a set does not exist and yet i_1 was found then it clearly satisfies the above condition). It is easy to see that $O(\epsilon^{-1})$ queries will be sufficient to hit a member of this set with probability $1 - 2^{-|S|-2}$.

The algorithm now $\frac{1}{2}\epsilon$-tests the string $v_{i_1+1} \ldots v_n$ for the property of not containing S' as a substring (with error probability $\sum_{i=1}^{|S|-1} 2^{-i-2}$), and accepts the input if and only if this test accepts. By the induction hypothesis, the number of queries in this stage is also linear in ϵ^{-1}.

For the correctness proof, it is quite clear that if the input does not contain S as a substring then the algorithm will accept with probability 1, because this necessarily implies that $v_{i_1+1} \ldots v_n$ does not contain S' as a substring.

On the other hand, if the algorithm accepts $v_1 \ldots v_n$ with probability more than $\sum_{i=1}^{|S|} 2^{-i-2}$, then we can define a string $w_1 \ldots w_n$ that is ϵ-close to $v_1 \ldots v_n$ and does not contain S as follows: $w_1, \ldots, w_{i_1-1}$ are all set to '0' (this makes no more than $\frac{1}{2}\epsilon n$ changes from $v_1, \ldots, v_n$), $w_{i_1} = v_{i_1} = 1$, and $w_{i_1+1}, \ldots, w_n$ are set to a string that is $\frac{1}{2}\epsilon$-close to $v_{i_1+1} \ldots v_n$ and contains no instance of S'.

It is clear that $w_1 \ldots w_n$ is ϵ-close to $v_1 \ldots v_n$. Also, $w_1 \ldots w_n$ does not contain an instance of S that starts before i_1 as there are no occurrences of '1' in that range, and on the other hand does not contain an instance of S starting at i_1 or later because $w_{i_1+1} \ldots w_n$ does not contain an instance of S'. This completes the proof of the properties of $v_1 \ldots v_n$ in the case that the algorithm accepts, and thus the correctness proof of the test. ∎

It is still not known whether the testing of all context free languages can be bounded from *above* by a power of n that is smaller than 1. An investigation into this question is given in [49].

In [46], Newman found a generalization of the positive part of Theorem 3.1, by considering a non-uniform counterpart of the notion of a finite automaton.

Theorem 3.3 ([46]) *Every property identifiable by a read-once fixed-width oblivious branching program is testable with a number of queries polynomial in ϵ^{-1} (and independent of the input size).*

However, we close this section by noting that this result too is the best possible of its kind in some sense.

Theorem 3.4 ([30]) *There exists a property identifiable by a read-twice oblivious branching program of width three, that is not testable with a number of queries that is less than some fixed power of the input size.*

4 Graph testing in the dense context

Graph theory is a point of contact between pure mathematics and computer science. Because of the major role it plays in the theory of computational complexity, it is only natural that problems that arise from graph theory, such as (proper) vertex colorability, will also be investigated from the point of view of property testing.

The first investigation into the testing of graph properties was performed by Goldreich, Goldwasser and Ron in [37]. It was motivated by the idea of property testing serving as a new notion of approximation, and by some related questions that arise in computational learning theory. Here the input is assumed to be an adjacency matrix of a graph G with n vertices. This means that a query consists of obtaining an answer as to whether a pair v_iv_j of vertices is an edge of G or not, and that G is considered to be ϵ-far from satisfying a property P if one has to add and remove more than a total of $\epsilon\binom{n}{2}$ edges in G to make it satisfy P.

This also means that we restrict ourselves to properties that relate to the graph G underlying the input, such as vertex colorability or the existence of a large clique. Formally, it means that we investigate only properties that are invariant with respect to graph isomorphisms – all permutations of the input that result from permuting the vertex set of the underlying graph.

This invariance of graph properties has some useful implications for testing. For example, it implies that if two graph properties P and Q satisfy that for every ϵ there exist only a finite number of graphs that satisfy one of them but are ϵ-far from satisfying the other, then either both of P and Q are testable (for every ϵ) with a number of queries that is independent of n, or neither of them is thus testable. In this setting P and Q are said to be *indistinguishable* from one another; the exact statement and proof of this observation are found in [4]. The above invariance has also strong implications concerning the type of testers that exist for these properties, such as those described in [40].

Lemma 4.1 ([40]) *If there exists an ϵ-tester for a graph property that makes q queries, then there exists such a tester makes its queries by uniformly and randomly choosing a set of $2q$ vertices and querying all their pairs. In particular, it is a non-adaptive ϵ-tester making $\binom{2q}{2}$ queries.*

Proof sketch: We define a *vertex uncovering* tester to be a testing algorithm that in each stage picks a new vertex v (possibly based on previous queries), and queries all the edges between v and the vertices that were picked in the previous stages. This process is called the *uncovering* of v. Any ϵ-tester making up to q queries can be transformed to a vertex uncovering tester that uncovers exactly $2q$ vertices in the following manner: Whenever the original algorithm queries an edge, the new algorithm uncovers one by one its two incident vertices, if they were not already uncovered in earlier stages. In the end, if the original algorithm accepts or rejects the output before $2q$ vertices were uncovered, the new algorithm first uncovers more arbitrarily chosen vertices until exactly $2q$ vertices are uncovered; after the uncovering is completed, the acceptance criterion of the original algorithm is used.

The new algorithm that uncovers $2q$ vertices may still be adaptive, but now instead of running it over the input graph G, we run it through a uniformly random permutation of the vertices of G. It is now not hard to see that the result of this modification will be an algorithm satisfying the statement of the lemma. ■

As it turns out, for every fixed given k the property of a graph being k-colorable can be ϵ-tested with a number of queries depending only on ϵ (and k). Informally, this means that the NP-hardness of k-colorability for $k \geq 3$ essentially follows from constructions that are not k-colorable but can be made k-colorable by removing only a small portion of the edges.

To illustrate the proof given in [37] for the testability of k-colorability, we give here the proof for the special case $k = 2$.

Proposition 4.2 ([37]) *There exists an ϵ-test for the property of a graph being bipartite, that makes a number of queries which is polynomial in ϵ (and is independent of the input size).*

To prove this proposition, we first note that it is quite trivial to test for bipartiteness if one is already given the supposed proper 2-coloring along with the input graph.

Lemma 4.3 *There exists an algorithm, that given a graph G and a coloring of some of its vertices, with the possibility also to query for a given vertex whether it is colored and with what color, can distinguish (with high probability) using a number of queries that is linear in ϵ^{-1}, between the case* *that G contains no edges whose incident vertices are colored with the same color, and the case that G contains more than $\epsilon\binom{n}{2}$ such edges.*

Proof: There are a total of $\binom{n}{2}$ vertex pairs in G, and we need to distinguish between the case that none of these pairs is an edge connecting two identically colored vertices, and the case that more than $\epsilon\binom{n}{2}$ of the pairs are such edges. This is done by a simple sampling of the pairs, where for every pair in the sample we query the colors of its vertices, and whether it is an edge of G. ■

Proof sketch of Proposition 4.2: The test first picks a set S of $O(\epsilon^{-1}\log(\epsilon^{-1}))$ vertices of the input graph G independently and uniformly at random; with high probability it has the property that all but at most $\frac{1}{5}\epsilon n$ of the vertices that have degree at least $\frac{1}{5}\epsilon n$ in G have also a neighbor in S.

We now look at all possible partitions of S into S_1 and S_2, and define for each of these a partial 2-coloring of G: Every vertex that has a neighbor in S_1 is colored by 2, and every vertex that has a neighbor in S_2 but not in S_1 is colored by 1. With this definition, finding out whether a given vertex is colored and with what color requires $|S| = O(\epsilon^{-1}\log(\epsilon^{-1}))$ queries of G.

For every possible partition of S, we can use the algorithm from Lemma 4.3 to find out whether the graph is $\frac{1}{5}\epsilon$-far from containing no monochromatic edges according to the corresponding coloring. However, the number of possible partitions is exponential in ϵ. In order to check all possible partitions of S at once, we first note that with a simple amplification technique, the error probability of the algorithm in Lemma 4.3 can be made exponentially small in $|S|$; the resulting algorithm still picks a number of vertices that is polynomial in ϵ, and queries only for edges between these vertices and between them and S. Thus we can pick a uniformly random set U consisting of a polynomial number of vertices of G, and then for every partition of S run the algorithm from Lemma 4.3 on this same set, reusing the same queries. With high probability the above algorithm will not make an error concerning any of the $2^{|S|}$ partitions of S.

To end the test, we accept the input if there exists a partition of S for which G is not far from having no monochromatic edges in the corresponding coloring, and otherwise we reject the input. The algorithm will never reject an input which is in fact bipartite, because a proper 2-coloring of the input graph induces for every S a partition for which the corresponding coloring contains no monochromatic edges.

On the other hand, with high probability it is the case that if the algorithm accepts then the graph is ϵ-close to being bipartite. To prove this we look at the appropriate partition of S and its corresponding partial 2-coloring. For every vertex that is not colored by the coloring, we just remove all of its incident edges and color it arbitrarily; we know that with high probability this will change no more than $\frac{4}{5}\epsilon\binom{n}{2}$ edges (no more than $\frac{2}{5}\binom{n\epsilon}{2}$ edges incident with vertices of degree no more than $\frac{1}{5}\epsilon n$, and no more than another $\frac{2}{5}\epsilon\binom{n}{2}$ edges for up to $\frac{1}{5}\epsilon n$ additional vertices with higher degrees that were not colored on account of having no neighbors in S). We also know that the graph may contain up to $\frac{1}{5}\epsilon\binom{n}{2}$ more edges that violate the original partial coloring; by removing these as well we are done. ■

The description of the calculation above is deceptively complex. In fact, one can easily show that instead of picking a set of vertices S, and then checking all its partitions against a second set of vertices U, one could just pick a set that serves as "$S \cup U$", and check whether the subgraph of G induced on this set is bipartite. Thus the above proof also yields a graph-theoretic result – a graph that is ϵ-far from being bipartite contains at least $\frac{2}{3}\binom{n}{k}$ induced subgraphs with k vertices that are not bipartite, where k depends only on ϵ^{-1} and is in fact polynomial in it.

This combinatorial corollary is not a coincidence, as the following result states that a similar phenomenon occurs for any testable graph property that is closed under the taking of induced subgraphs.

Proposition 4.4 (N. Alon, see Appendix D of [40]) *If P is a graph property that is closed under the taking of induced subgraphs, and there exists an ϵ-test for P that always makes q queries independently of the number of vertices of the input G, then there exists a ϵ-test for P that works by uniformly choosing $f(q)$ vertices of the graph G, querying all the pairs, and checking that the induced subgraph satisfies P. Moreover, $f(q)$ is a global polynomial in q, independent of the other parameters.*

We conclude by referring the reader to [37] concerning other partition-related graph properties, such as k-colorability (for a fixed k), having a given maximal cut, or having a large clique. Aside from the testability proofs of these, [37] contains their generalization to the following grand theorem.

Theorem 4.5 ([37]) *For a fixed k, let P be the property of a graph G with n vertices having a partition $V_1, \ldots, V_k$ of its vertex set, with $\alpha_i n \leq |V_i| \leq \alpha'_i n$ for every i (for fixed, given $\alpha_i < \alpha'_i$), and such that for every $1 \leq i \leq j \leq k$ the number of edges between V_i and V_j (or within V_i if $i = j$) is between $\beta_{i,j} n^2$ and $\beta'_{i,j} n^2$ (for fixed, given $\beta_{i,j} < \beta'_{i,j}$).*

Property P is testable, with a number of queries that is polynomial in ϵ.

5 Dense graphs continued – use of the Regularity Lemma

This section is about the use of Szemerédi's Regularity Lemma for property testing, and its application to proving testability for a class of graph properties that are defined by certain first order expressions.

To formulate logical expressions for properties, we shall use a language that includes variables that range over vertices, first order quantifiers over these variables, the adjacency relation "u is a neighbor of v", the equality relation "$u = v$", and boolean connectives. All properties thus expressible are called *first order graph properties.* We further classify them by the first order quantifiers that they use. The best way to illustrate this is by giving some examples:

- "The graph G contains a triangle". This is a first order graph property of type '$\exists$', as it has an expression using only this quantifier: $\exists u_1 u_2 u_3 (u_1 \sim u_2 \wedge u_1 \sim u_3 \wedge u_2 \sim u_3)$, where $u \sim v$ is the relation stating that u and v are neighbors (and in particular are not the same vertex).
- "The graph G does not contain an induced square". This is a first order graph property of type '$\forall$', as it states that for every set of four distinct vertices, certain adjacency configurations do not occur.
- "The graph G contains three distinct vertices u_1, u_2, u_3 such that every other vertex has exactly one of them as a neighbor, and furthermore there exists no triangle containing any of these three vertices". This is a first order graph property of type '$\exists\forall$'.

It is not hard to show that the third example above is related to the property of 3-colorability: Every graph satisfying this property is not far from being 3-colorable, as the vertices u_1, u_2, u_3 determine a proper 3-coloring of the rest of the graph, and conversely every proper 3-coloring of a graph can be used to find a small modification of the graph that satisfies the property above. In other words, the above first order property and the property of being 3-colorable are indistinguishable, as per the definition outlined in Section 4. When it comes to property testing, first order expressions have more expressive power than they do in the context of traditional complexity theory.

Alon, Fischer, Krivelevich and Szegedy have shown in [4] what classes of first order expressions are wholly testable.

Theorem 5.1 ([4]) *All first order graph properties of type '$\exists\forall$' are testable with a number of queries independent of n. On the other hand, there exists a '$\forall\exists$' property that is not thus testable.*

The proof of the positive part of this theorem relies heavily on the existence of regular pairs of sets, as stated in Szemerédi's Regularity Lemma [56]. As for the negative part, the main idea of its proof is given in Section 8.

For a good exposition of Szemerédi's Regularity Lemma (stated below) and its proof, the reader is referred to [21, Chapter 7], and for a full survey regarding the many combinatorial applications of regular pairs and the Regularity Lemma the reader is referred to [44]. In the following we give a brief introduction to this useful tool, starting with the definition of regular pairs.

Definition 3 (density and regularity of set pairs) *For two nonempty disjoint vertex sets A and B of a graph G, we define $e(A,B)$ to be the number of edges of G between A and B. The* edge density *of the pair is defined by $d(A,B) = \frac{e(A,B)}{|A||B|}$. We say that the pair A,B is γ*-regular, *if for any two subsets A' of A and B' of B, satisfying $|A'| \geq \gamma|A|$ and $|B'| \geq \gamma|B|$, the edge density satisfies $|d(A',B') - d(A,B)| < \gamma$.*

The main strength of the definition of regularity is in the characteristics that regular pairs share with random bipartite graphs of the same density. Consider for example the following.

Lemma 5.2 *For every $\eta > 0$ there exist $\gamma = \gamma(\eta)$ and $\delta = \delta(\eta)$ such that if A, B and C are disjoint vertex sets of a graph G where each pair is γ-regular and has density at least η, then G contains at least $\delta|A||B||C|$ distinct triangles with a vertex from A, a vertex from B, and a vertex from C respectively.*

The proof is not very hard, but we shall omit it here and refer the reader to [44], or to [4] for proofs that are more specific to the topic of testing.

The following lemma is a strong tool in graph theory, as it states that in essence most of the vertex-pairs of any given graph can be neatly fitted into a structure of regular pairs. This structure is given by an *equipartition* of G – a partition of its vertex set into sets whose sizes differ from each other by no more than 1.

Lemma 5.3 (Szemerédi's Regularity Lemma [56]) *For every m and $\epsilon > 0$ there exists some $T = T(m,\epsilon)$ with the following property.*

If G is a graph with $n \geq T$ vertices, and $\mathcal{A}$ is an equipartition of the vertex set of G into m sets, then there exists an equipartition $\mathcal{B}$ that is a refinement of $\mathcal{A}$ with k sets, where $m \leq k \leq T$, for which all set pairs but at most $\epsilon\binom{k}{2}$ of them are ϵ-regular.

By using the Regularity Lemma, one could re-prove for example that proper k-colorability is testable – in fact, a combinatorial statement to this effect was implicitly given in [51], before the concept of algorithmic testing came into being.

Another typical use is when testing for the non-existence of fixed size cliques; see for example the following combinatorial proposition.

Proposition 5.4 (N. Alon, private communication) *For every ϵ there exists $\delta = \delta(\epsilon)$ such that any graph with n vertices, which is ϵ-far from being triangle-free, contains at least $\delta\binom{n}{3}$ distinct triangles.*

Proof sketch: To prove this we consider an equipartition $\{V_1, \ldots, V_k\}$ of the vertices of the graph G according to Lemma 5.3, where $5\epsilon^{-1} \leq k \leq T(5\epsilon^{-1}, \epsilon')$ (we make it a refinement of an arbitrary equipartition into $5\epsilon^{-1}$ sets), in which all but at most $\epsilon'\binom{k}{2}$ of the pairs are ϵ'-regular. The parameter ϵ' is set to $\min\{\frac{1}{5}\epsilon, \gamma(\frac{1}{5}\epsilon)\}$, where γ is the function appearing in Lemma 5.2.

We now consider a subgraph G' of G, obtained by deleting all edges of G that are internal to V_i for any i, or are between V_i and V_j for any pair V_i, V_j which is not ϵ'-regular, or any pair whose density is less than $\frac{1}{5}\epsilon$.

Noting that G' was obtained from G by deleting less than $\epsilon\binom{n}{2}$ edges, it follows that G' contains a triangle, since G is ϵ-far from being triangle-free. By the construction of G', this is necessarily a triangle with a vertex from V_i, a vertex from V_j and a vertex V_k, for some V_i, V_j, V_k which are all distinct and for which all pairs are in particular $\gamma(\frac{1}{5}\epsilon)$-regular (with respect to G) and with density at least $\frac{1}{5}\epsilon$. The existence of such V_i, V_j, V_k implies by Lemma 5.2 the existence of at least $\delta'\binom{n}{3}$ distinct triangles in G, where $\delta' = 6\delta(\frac{1}{5}\epsilon)(T(5\epsilon^{-1}, \epsilon'))^{-3}$ with δ denoting here the corresponding function from Lemma 5.2. ■

Proposition 5.4 implies that the following is a 1-sided ϵ-test for being triangle-free: Choose independently and uniformly $O(\delta(\epsilon)^{-1})$ random vertex triplets (where δ is the function of the proposition), and check whether any of them is a triangle. If a triangle was found then reject the input; otherwise accept it.

There is one serious drawback to results proven using the Regularity Lemma: The dependency of $T(m, \epsilon)$ on its parameters is very severe, a tower in a polynomial of $m\epsilon^{-1}$, and by [41] this dependency cannot be significantly improved as long as the full strength of the Regularity Lemma is employed. This makes the tests thus derived of a theoretical importance rather than a practical one.

Unlike the situation with k-colorability, currently there is no known proof for a test (with a constant number of queries) for the property of being triangle-free that has a better dependency on ϵ. However, it is known that the number of queries of a 1-sided test for this property cannot be polynomial in ϵ, using a bound from [14], and by Proposition 4.4 above this follows for 2-sided tests as well.

When one tries to generalize the above result to properties defined as not containing a given fixed induced subgraph, other than a clique, another problem arises (remember that an induced subgraph is a subgraph obtained from the original graph by deleting vertices and their incident edges, but deleting no other edges). In the above proof implicit use was made of the fact that removing edges cannot *add* new triangles to the graph; but in the case of induced squares, for example, removing an edge can actually make its pair part of a new copy of the forbidden subgraph.

In proving the Regularity Lemma one cannot do away completely with the existence of a small fraction of non-regular pairs, which are the core of the problem in proving similar results for induced subgraphs because of the above mentioned inability to safely remove edges. The workaround in [4] to this problem is in proving a variant of the Regularity Lemma, that enables one to find a large induced subgraph of the original graph, for which there exist an equipartition with all the pairs being regular, and that has the additional property of being able to "model" the original graph, in the sense that if the graph G is far from satisfying the property then so is this subgraph.

The full details are outside the scope of this survey; the reader is referred to [4], where this method is used for proving that all '$\exists\forall$' first order graph properties are testable, and to [25], where testability is proven for some additional classes of properties that further generalize the '$\exists\forall$' class.

As a final note, we mention some instances in which using the full version of the Regularity Lemma can be avoided, making for a better dependency of the number of queries on ϵ. It still remains to be seen whether such a use can be avoided in the general case, or even for the property of a graph being triangle-free.

In the case where the property is that of not containing a fixed bipartite graph as a not necessarily induced subgraph, this can be tested with a number of queries polynomial in ϵ. The proof uses a Zarankiewicz type theorem, that states in essence that the only graphs satisfying such a property are almost edge-less. On the other hand, there exists no test making a number of queries that is polynomial in ϵ^{-1}, for any property defined as not containing a fixed non-bipartite graph; this was proven in [1]. There is also an investigation into what properties defined by a forbidden induced subgraph have a test with a polynomial number of queries, in [6].

Suppose now that a property is defined by not containing a fixed *induced* subgraph, and that we wish to test only bipartite graphs for satisfying it. The following is achieved using a rather restricted variant of the Regularity Lemma, proven for this purpose.

Theorem 5.5 ([29] with an improvement by N. Alon) *Any property of bipartite graphs, that is defined as not containing a given fixed induced subgraph, has a* 2*-sided test using a number of queries exponential in* ϵ*, and a* 1*-sided test using a number of queries doubly-exponential in* ϵ*.*

It is not known at present whether testing of such properties of bipartite graphs can be made polynomial in ϵ^{-1} or not.

6 Testing of sparse graphs

As far as inputs given in terms of the adjacency matrix of a graph are concerned, all graphs with $o(n^2)$ edges are indistinguishable from null (edge-less) graphs. This makes testing of properties such as not containing a cycle trivial – the algorithm just needs to sample and distinguish between the case that G has more than $\epsilon\binom{n}{2}$ edges, and the case that G has less than $\frac{1}{2}\epsilon\binom{n}{2}$ edges. In the first case the algorithm rejects G, and in the second case it accepts G.

However, there is still the interesting (and non-trivial) question of whether one could, for an input graph G with n vertices and m edges, distinguish between the case that G is acyclic, and the case that no removal of up to $\epsilon(n+m)$ edges from G (rather than $\epsilon\binom{n}{2}$) can make it acyclic. In [38] Goldreich and Ron have defined a different model for encoding input graphs, in which such a testing notion makes sense (in the above $\epsilon(n+m)$ is used, and not ϵm, because of further technical complications arising in graphs with $o(n)$ edges).

Under the model discussed here, a graph is not encoded by its adjacency matrix, but instead the list of neighbors is given for every vertex. It is assumed that the input is indeed an encoding of a graph, so if u is in the list of neighbors of v then v is also in the list of neighbors of u. Note that the input is not a sequence of bits, but a sequence of integers, some of which serve as 'pointers' to the different lists. Thus a query in this model consists of either finding out the number of neighbors of a vertex v_i, or finding the identity of the j'th neighbor of v_i according to its adjacency list.

In the case where inputs are limited to graphs with a fixed maximum degree Δ, and the allowable modifications of the graphs must also satisfy this restriction, one may assume that all the adjacency lists are of this fixed size, with the possibility of some members in the lists being 'null' for vertices with degree less than Δ. For this reason we may also assume that all queries are of the second type described above. In this model the difference between graphs in relation to $n+m$ is linearly proportional to the hamming distance between inputs, so we can just use the latter.

When the maximum degree is not bounded, the notion of the distance is more complex because the sizes of the adjacency lists can vary. The distance between inputs is actually a function of the edit distances between the corresponding adjacency lists; there are in addition models in which it is also allowed to add and remove vertices from the graph (in removing a vertex it is natural to take into account also the removal of its incident edges), but we shall not discuss these models in detail here.

In a work by Goldreich and Ron [38] several properties (some of which, such as connectivity and acyclicity, are trivial to test in the dense graph model but are not trivial in this one) are proven to be testable. On the other hand, some other properties that are easily testable in the dense context are not testable in the sparse one. The following is a partial summary of their results.

Theorem 6.1 ([38]) *The following properties are testable with a number of queries not depending on n, for the model of testing graphs with a bounded degree Δ given by their adjacency lists: k-edge connectivity for any fixed given k, not having a given fixed graph H as a (not necessarily induced) subgraph, and being acyclic.*

On the other hand, ϵ-testing of bipartiteness for some fixed ϵ, and $\Delta = 3$, requires at least $\frac{1}{3}\sqrt{n}$ queries.

It turns out that in the case of directed graphs, also the property of being acyclic requires many queries to test for in the sparse model [15].

In the sparse context a testing algorithm typically consists of performing a breadth first search from randomly chosen vertices, and then accepting or rejecting the input based on the resulting subgraphs. For example, testing for connectedness can be done using the following simple lemma (one can show that this lemma is also translatable to the model where the input graph and its modifications are subject to a bound on their maximum degree).

Lemma 6.2 *If a graph G with n vertices requires the addition of at least ϵn edges to make it connected, then at least $\frac{1}{2}\epsilon n$ of the vertices of G are in connected components of size at most $2\epsilon^{-1}$.*

Proof: It is easy to see that a graph G as above contains more than ϵn connected components. Each of them contains at least one vertex, and on the other hand it is clear that less than half of them can be of size more than $2\epsilon^{-1}$. ∎

The test now consists of picking $O(\epsilon^{-1})$ random vertices, and performing a breadth first search from each to them to check whether it is part of a connected component with less than $2\epsilon^{-1}$ vertices (this requires $O(\epsilon^{-1})$ queries per vertex). If one of the chosen vertices is in such a component then the algorithm rejects, and otherwise it accepts.

The above test takes $O(\epsilon^{-2})$ queries, and works for both the bounded degree model and the unbounded degree one. Note however that [38] contains an improved analysis, that in particular requires only $O(\epsilon^{-1})$ queries for the unbounded degree model, and slightly more than that for the bounded degree one.

7 Testing posets for monotonicity

Keeping lists in a sorted order is a common practice, and for well known reasons. This naturally leads to the question of how many queries it takes to ensure that a sequence of n integers is mostly sorted, that is, to distinguish between the case that it is monotone nondecreasing, and the case that no subsequence of $(1-\epsilon)n$ integers from it is monotone nondecreasing.

This is equivalent to the question of ϵ-testing a sequence of n integers for this property, where the distance is given by the fraction of integers that need to be altered in order to make the sequence monotone. Ergün, Kannan, Kumar, Rubinfeld and Viswanathan have shown in [23] that for a fixed ϵ it is enough to make $O(\log n)$ queries. Although this is dependent on n, it is still a much better alternative than that of reading the whole input.

Theorem 7.1 ([23]) *There exists an ϵ-test that makes $O(\epsilon^{-1}\log n)$ queries, for the property of a sequence of n integers being monotone nondecreasing.*

Proof sketch: For any monotone increasing sequence of size n, one can check using $O(\log n)$ queries whether a given number appears in this sequence, by the well known binary search method.

We now look at our input, denote it by $v_1v_2\dots v_n$. We can safely assume that all values are distinct, as otherwise we can consider the sequence $v'_1v'_2\dots v'_n$ defined by $v'_i = nv_i + i$ instead. We call v_i *well positioned* if the binary search procedure above, when applied to v_i over $v_1v_2\dots v_n$, indeed finds the position i.

It takes $O(\log n)$ queries to check that a given v_i is well positioned. Our algorithm makes $O(\epsilon^{-1})$ iterations of this procedure for random i's chosen uniformly and independently, to distinguish between the case that all values are well positioned, and the case that at least ϵn of the values are not well positioned. In the first case the algorithm accepts the input, and in the case where any values that are not well positioned were found the algorithm rejects.

If the input is in fact monotone, then clearly all of its values are well positioned, and so the algorithm accepts. On the other hand, if the algorithm accepts (with high enough probability), then it means that more than $(1-\epsilon)n$ of the values are well positioned. To end the proof, we show that the subsequence of all well positioned values is monotone; such a subsequence can easily be completed to a monotone nondecreasing sequence of size n by putting new values in the places where the original values are not well positioned.

Assuming that $i < j$, and that v_i and v_j are both well positioned, we now show that $v_i \leq v_j$. For this it is sufficient to find an index k satisfying $i \leq k \leq j$ such that v_k was visited by the binary searches for both values, because this would imply that $v_i \leq v_k$ and $v_k \leq v_j$. To find it, we just choose k to be the last location that was queried in both binary searches, the one for v_i and the one for v_j. Common locations in the two searches exist because in particular the location $\lfloor \frac{n}{2} \rfloor$ is common to all binary searches. ■

One should note here that the above algorithm only looks adaptive. In fact, the algorithm can supply all the query locations in advance – to check that a value is well positioned, it is enough to unconditionally query the binary search sequence that would arise if it were well positioned, since in order to know that a value is not well positioned it is enough to know that the actual binary search sequence diverges at some point from the queried "expected" sequence.

As it turns out, the above property cannot be tested using a constant number of queries. An $\Omega(\log n)$ lower bound (for some constant ϵ) is given in [23] for non-adaptive algorithms that are restricted to making decisions based on comparisons. In [26] it is proven that for this particular property better general algorithms do not exist, by showing first that for a certain class of properties, that includes monotonicity, there are comparison-based algorithms that are optimal.

It is also of interest to investigate testing for natural notions of monotonicity that differ from the simple notion of a sequence being nondecreasing. The next step is to consider functions whose domain is not endowed with a linear order as $\{1,\ldots,n\}$ is, but with other types of partial orders instead. The first work in this direction is by Goldreich, Goldwasser, Lehman, Ron and Samorodnitsky [36], that have proven the following result about the monotonicity of functions from $\{0,1\}^d$ (with the usual product order over the hypercube) to $\{0,1\}$.

Theorem 7.2 ([36]) *ϵ-testing that a function from $\{0,1\}^d$ to $\{0,1\}$ is monotone nondecreasing can be done using $O(\epsilon^{-1}d)$ queries.*

The testing algorithm for this consists of choosing a random pair of points in $\{0,1\}^d$ that differ in exactly one coordinate, comparing the values of the input function over these, and repeating this procedure $O(\epsilon^{-1}d)$ times. For the correctness proof the reader is referred to [36].

A generalization of the above result to d-dimensional matrices of any size (and not only $\{0,1\}^d$) is given in [22] (some generalizations in this direction were already proven in [36]).

Theorem 7.3 ([22]) *A function from $\{1,\ldots,n\}^d$ to $\{1,\ldots,m\}$ can be ϵ-tested for monotonicity using $O(\epsilon^{-1}d\log n\log m)$ queries.*

Dealing with d-dimensional matrices with integer values (not restricted to some $\{1,\ldots,m\}$) is also covered by [22] – a simple analysis of the algorithm that is given there implies that the 'm' term in the expression for the number of queries can be replaced by the actual size of the image of the input function, which is bounded by n^d. This gives an $O(\epsilon^{-1}d^2\log^2 n)$ upper bound on the required number of queries in this case. As for binary functions over $\{1,\ldots,n\}^d$ where d is fixed, in [22] it is shown that in this case the number of queries does not need to depend on n at all.

Other partially ordered sets can also serve as domains for functions to be tested for monotonicity. For example, rooted trees. Monotonicity testing in the general context is treated in [28]. There are positive results for some partially ordered sets, such as the possibility of testing a binary function over a rooted tree for monotonicity with a number of queries that depends only on the approximation parameter ϵ. On the other hand, [28] also contains some non-trivial lower bounds, including one for the case of a function from $\{0,1\}^d$ to $\{0,1\}$; in that case the number of queries has to depend on d. Narrowing the gap between the upper and lower bounds on the number of queries in this setting still remains an open (and interesting) question.

8 Proving non-testability results

The methods presented thus far deal with proving upper bounds for property testing, through means of presenting efficient testing algorithms. As with other areas of computer science, lower bounds play an essential role in the field of property testing as well, and the first example of a property that is hard to test (one that belongs to NP in the traditional complexity framework) was given in [37].

This section presents what is currently the most efficient tool for proving lower bounds for property testing, Yao's method [57]. The basic method is the following. Instead of trying to find a worst case input for every possible (randomized) ϵ-testing algorithm, we find a probability distribution over the inputs, such that every *deterministic* algorithm will fail with probability greater than $\frac{1}{3}$ on an input taken according to this distribution. From this it follows that every randomized algorithm will also fail with at least the same probability over this input distribution, and so for every fixed randomized algorithm there will be at least one ("non-randomized") input that demonstrates at least the same average failure rate.

In our context we usually deal with two distributions, one on inputs satisfying the property and one on inputs that are ϵ-far (for some constant ϵ) from satisfying it, such that in some sense it is hard for a (deterministic) algorithm making few queries to distinguish between them. For simplifying the presentation we will only consider binary inputs in the following. We first need some definitions.

Definition 4 (restrictions, variation distance) *For a distribution D over inputs, where each input is a function $f : \mathcal{D} \to \{0,1\}$, and for a subset $\mathcal{Q}$ of the domain $\mathcal{D}$, we define the* restriction *$D|_{\mathcal{Q}}$ of D to $\mathcal{Q}$ to be the distribution over functions of the type $g : \mathcal{Q} \to \{0,1\}$, that results from choosing a function $f : \mathcal{D} \to \{0,1\}$ according to D, and setting g to be $f|_{\mathcal{Q}}$, the restriction of f to $\mathcal{Q}$.*

Given two distributions D_1 and D_2 of binary functions from $\mathcal{Q}$, we define the variation distance *between D_1 and D_2 as follows: $d(D_1, D_2) = \frac{1}{2}\sum_{g:\mathcal{Q}\to\{0,1\}} |\Pr_{D_1}(g) - \Pr_{D_2}(g)|$, where $\Pr_D(g)$ denotes the probability that a random function chosen according to D is identical to g.*

It is a well known fact that if a deterministic algorithm is given an input chosen according to either a distribution D_1 or a distribution D_2, then its acceptance probabilities for the two cases do not differ by more than $d(D_1, D_2)$. The most common application in our case is the following folklore lemma.

Lemma 8.1 *Suppose that there exists a distribution D_P on inputs over $\mathcal{D}$ that satisfy a given property P, and a distribution D_N on inputs that are ϵ-far from satisfying the property, and suppose further that for any $\mathcal{Q} \subset \mathcal{D}$ of size q, the variation distance between $D_P|_{\mathcal{Q}}$ and $D_N|_{\mathcal{Q}}$ is less than $\frac{1}{3}$. Then it is not possible for a non-adaptive algorithm making q (or less) queries to ϵ-test for P.*

Proof: We construct a distribution D over inputs by deciding with probability $\frac{1}{2}$ to choose an input according to D_P, and with probability $\frac{1}{2}$ to choose an input according to D_N. We now consider deterministic algorithms – every deterministic non-adaptive testing algorithm with q queries consists of querying the input f over a fixed set Q of size q, and then making an acceptance or a rejection decision based on $f|_Q$.

Let a_P be the acceptance probability of our algorithm when given a random input chosen according to D_P, and let a_N be its acceptance probability for an input chosen according to D_N. The probability of making an error is clearly $\frac{1}{2}(1 - a_P) + \frac{1}{2}a_N$, and by the assumptions on the distributions and the discussion above we know that $|a_P - a_N| < \frac{1}{3}$. Putting these together shows that the error probability for every such algorithm is more than $\frac{1}{3}$. It thus follows from Yao's principal that there is no (probabilistic) non-adaptive ϵ-testing algorithm for the property P that makes at most q queries and whose error probability is at most $\frac{1}{3}$. ■

In general, we would like to place lower bounds on all testing algorithms, including adaptive ones, so we now turn our attention to those. In cases where binary inputs are involved, a lower bound of q queries on non-adaptive algorithms implies a lower bound of $\log q$ queries on adaptive ones (see the end of Section 9 for more information on this), which suffices if it is only required to prove that a constant number of queries is insufficient for testing a particular property.

In some cases the gap between adaptive and non-adaptive testing is indeed exponential, but sometimes the gap between adaptive and non-adaptive properties can be narrowed further. In some cases this is accomplished using some feature of the particular hard to test property, such as Lemma 4.1 that makes this gap no more than quadratic for properties formulated in the dense graph model. In other cases it can be proven that the input distributions satisfy certain additional conditions, ensuring stronger lower bounds for adaptive algorithms; more about this below. There are also many properties for which the true gap between adaptive and non-adaptive testing is not yet known.

As the main example here for using Yao's method, we consider the following problem: Given as input the adjacency matrices of two graphs with n vertices, test for the property of these two graphs being isomorphic.

This property was shown in [4] to be non-testable, and this was then used to show that there exist non-testable '$\forall\exists$' type first order graph properties. The original exposition of the proof in [4] uses counting instead of Yao's method; here we give a proof using Yao's method that can serve as an example for proving lower bounds.

Proposition 8.2 ([4]) *The property of two graphs (given by their adjacency matrices) being isomorphic is not $\frac{1}{8}$-testable with any constant number of queries.*

Proof: We show that for every q there exists n, such that q edge queries are insufficient for distinguishing between the case that the two graphs with n vertices are isomorphic, and the case that all vertex permutations of the first graph differ from the second graph in more than $\frac{1}{4}\binom{n}{2}$ places (remember that the total size of the input is $2\binom{n}{2}$).

For this we construct two distributions over inputs. The distribution D_P is constructed by letting the input consist of a random graph G, with each edge being independently chosen with probability $\frac{1}{2}$, and a second graph that is a uniformly random permutation of G. The distribution D_N is constructed by letting the input consist of a random graph G as before, and another, independently chosen, random graph G'.

It is clear that an input chosen according to D_P satisfies the property with probability 1. On the other hand, by a standard large deviation argument, in an input chosen according to D_N any fixed vertex permutation of G will with probability at least $1 - 2^{-n^2/50}$ differ in more than $\frac{1}{4}\binom{n}{2}$ places from G'. Thus, with probability $1 - o(1)$, all vertex permutations of G will have at least this difference from G'. While D_N does not exactly satisfy the requirements of Lemma 8.1 (as it requires that an input chosen according to D_N will be far from satisfying the property with probability 1), we will replace it with D'_N, the distribution that results from conditioning D_N on the event that all permutations of G indeed differ in more than $\frac{1}{4}\binom{n}{2}$ places from G'.

We now consider any fixed set $\mathcal{Q} = p_1, \ldots, p_q$ of vertex pairs, some inside the first graph, some inside the second one. For an input chosen according to D_N, the values of these pairs will be q uniformly and independently chosen random values in $\{0, 1\}$. Since D'_N is the conditioning of D_N on an event that occurs with probability $1 - o(1)$, the variation distance between the restriction of D'_N to $\mathcal{Q}$ and the uniform distribution over boolean functions from $\mathcal{Q}$ is $o(1)$.

We now show that the distance between the restriction of D_P to $\mathcal{Q}$ and the uniform distribution over $\mathcal{Q}$ is also $o(1)$. For this, let $u_1, \ldots, u_k$ be all vertices of the first graph that appear in the pairs $p_1, \ldots, p_q$, and let $v_1, \ldots, v_l$ be all such vertices of the second graph. It is clear that $k \le 2q$ and $l \le 2q$. Now let $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ be the permutation used in D_P to choose the second graph according to the first one. Let E denote the event that, for this permutation, $\sigma(u_i) \neq v_j$ for every $1 \le i \le k$ and $1 \le j \le l$. It is clear that, conditioned on the case that E occurs, $p_1, \ldots, p_q$ will again be q uniformly and independently chosen values of $\{0, 1\}$. But since for every fixed q the event E occurs with probability $1 - o(1)$ (for large enough n), this means that the restriction of the (unconditioned) distribution D_P to $\mathcal{Q}$ will have variation distance $o(1)$ from the uniform distribution over boolean functions from $\mathcal{Q}$.

To conclude, we use Lemma 8.1 for the distributions D_P and D'_N, where n is chosen to be large enough so that the restriction of any of these to any set of q queries will have variation distance less than $\frac{1}{6}$ from the corresponding uniform distribution. ■

The proof of the negative part of Theorem 5.1 in [4] was done by showing that there exists a '$\forall\exists$' first order graph property that is indistinguishable from a graph isomorphism property similar to the one above.

As for a quantified lower bound, it can be shown that the distributions of $p_1, \ldots, p_k$ in the two cases become close enough for our purposes when $n = \Omega(q^2)$. Together with Lemma 4.1, this means that the above property cannot be tested with a number of queries that is less than some power of $2\binom{n}{2}$, the input size.

For proving lower bounds on non-adaptive 1-sided algorithms, the problem is usually much easier than the case for 2-sided algorithms. It is enough in this case to find a distribution over inputs that are ϵ-far from satisfying the property, but in which for any fixed $\mathcal{Q} \subset \mathcal{D}$ of size q, with high probability the restriction of the input f to $\mathcal{Q}$ is extensible to some possible input (different from f) that satisfies the property.

A concrete example can help explain this – in the case of testing a graph for being triangle free, a lower bound that is super-polynomial in ϵ^{-1} (but in this case is of course independent of the input size) is given by finding a graph that is far from being triangle free, but still does not contain too many distinct triangles, so that q queries will not capture a triangle with high probability. Such graphs can be constructed using the number-theoretic construction in [14], and by Proposition 4.4 this actually implies a lower bound for 2-sided testing as well. The construction details and a generalization thereof to other properties defined in terms of not containing a fixed (not necessarily induced) subgraph are found in [1], and further investigation in the direction of forbidden induced subgraphs is found in [6].

We now turn to some cases where the gap between the adaptive and the non-adaptive lower bounds can be narrowed without relying on a symmetric nature of the property (such as the case with graph properties in the dense model). Two examples of such cases are [38] (indeed, non-adaptive lower bounds are meaningless in the context of sparse graphs) and [30]. We conclude by proving the following lemma, which mimics the implicit argument used in [30]. A final word of caution – when using the following lemma additional care has to be taken in constructing the distributions D_P and D_N; for example, it is quite important that an input chosen according to D_N will be ϵ-far from satisfying the property with probability 1, not just $1 - o(1)$.

Lemma 8.3 *Suppose that there exists a distribution D_P on inputs over $\mathcal{D}$ that satisfy a given property P, and a distribution D_N on inputs that are ϵ-far from satisfying the property. Suppose further that for any $\mathcal{Q} \subset \mathcal{D}$ of size q and any $g : \mathcal{Q} \to \{0,1\}$, we have $\frac{2}{3}\mathrm{Pr}_{D_P|_\mathcal{Q}}(g) < \mathrm{Pr}_{D_N|_\mathcal{Q}}(g) < \frac{3}{2}\mathrm{Pr}_{D_P|_\mathcal{Q}}(g)$. Then it is not possible for* any *algorithm making q (or less) queries to ϵ-test for P.*

Proof: As in the proof of Lemma 8.1, we construct D by deciding with probability $\frac{1}{2}$ to choose an input according to D_P, and with probability $\frac{1}{2}$ to choose an input according to D_N. Every (possibly adaptive) deterministic algorithm with q queries takes the shape of a decision tree, which is a complete binary tree of height q, where every non-leaf node corresponds to a query location with its two children being labeled according to the two possible outcomes of the query, and every leaf node corresponds to an acceptance or a rejection decision.

Now let a_P denote the probability that the algorithm accepts an input chosen according to D_P, and let a_N denote the probability that the algorithm accepts an input according to D_N. It is enough to prove now that $|a_P - a_N| < \frac{1}{3}$, because then the proof here can be concluded in an identical manner to that of the proof of Lemma 8.1.

Let L be the set of leaves of the decision tree that accept the input. Every $u \in L$ corresponds to a set of queries $\mathcal{Q}_u$ and their answers $g_u : \mathcal{Q}_u \to \{0,1\}$. Moreover, for every input f, the decision tree of the algorithm will reach the leaf u if and only if $f|_{\mathcal{Q}_u} = g_u$, implying $a_P = \sum_{u\in L} \mathrm{Pr}_{D_P|_{\mathcal{Q}_u}}(g_u)$ and similarly $a_N = \sum_{u \in L} \mathrm{Pr}_{D_N|_{\mathcal{Q}_u}}(g_u)$. Finally, using the conditions on D_P and D_N we obtain $a_P - a_N = \sum_{u\in L}(\mathrm{Pr}_{D_P|_{\mathcal{Q}_u}}(g_u) - \mathrm{Pr}_{D_N|_{\mathcal{Q}_u}}(g_u)) < \frac{1}{3}\sum_{u\in L}\mathrm{Pr}_{D_P|_{\mathcal{Q}_u}}(g_u) = \frac{1}{3}a_P \le \frac{1}{3}$, and similarly $a_N - a_P < \frac{1}{3}$, concluding the proof. ∎

9 Related research and future directions

Property testing is a young field, that currently enjoys rapid growth. The purpose of this section is to give a glimpse into some topics not covered by the rest of this survey. Some of them were left uncovered due to size or scope considerations, while others were left out simply because they are still waiting to be investigated. The interested reader is also referred to other surveys of this and related fields, such as [52], [43] and [35].

9.1 Other works concerning testability

As is the case with all surveys, this one also cannot cover every work in this field. The following (also partial) list is given in acknowledgment of this fact.

Other algebraically motivated properties: The question of whether an input function over a vector space forms a bounded degree polynomial is just one of several algebraically motivated properties. For example, one could ask whether a 'multiplication table' given as input forms a (finite) group. In [23] some upper and lower bounds are given for this. For a supposed group with n elements there is an ϵ-test using $O(n^{3/2} \log^c n)$ queries, for some global c; this is a rather large quantity, but it is still much smaller then the size of the input, n^2. One should also note that in the context of prover assisted testing (see below), this quantity can be further reduced; related properties play a role in quantum property testing (see below) as well.

Another topic, somewhat related to testing polynomiality, is testing whether a given function satisfies a property defined by a functional equation, for example "$f(x+y)+f(x-y) = 2f(x)f(y)$". Some properties of this type are investigated for functions defined over certain subsets of $\mathcal{R}$ (the field of real numbers), in [53].

Testing of matrix poset properties: Another way to generalize the result about testing binary matrices for monotonicity is in extending it to other properties of matrices, defined in terms of first order expressions using the partial order that underlies a matrix (that is, the product order of each location's coordinates). Such properties are investigated in [29].

Geometrically motivated properties: In this context the input is not a binary sequence or a sequence of integers, but a sequence of real numbers. As a first example, it is shown in [23] how to test, using $O(\log n)$ queries, a sequence of n points from $\mathcal{R}^2$ for the property of being the nodes of a convex polygon given in a clockwise order. In [20] testing for several geometric properties of points from $\mathcal{R}^d$, such as being the nodes of a convex polygon given in *any* order, is considered. For most of these matching upper and lower bounds are given; these are rather large (most are fixed powers of n), but are still sublinear in the input size.

Another example, given a fixed d, is testing for the property of a given set of vectors in $\mathcal{R}^d$ being (k, b)-clusterable. Usually in such a context a more relaxed notion of testing needs to be used; here it is possible to efficiently distinguish between the case that the input is (k, b)-clusterable, and the case that even removing an ϵ fraction of the points will not make the rest $(k, (1+\epsilon)b)$-clusterable. The testing procedure is given in [2].

A geometrically motivated problem in a different direction is that of testing whether a given matrix of real numbers has certain properties, that are expected from a distance function of a discrete metric space. Properties of this type are investigated in [48].

Testing of boolean functions: In relatively recent years, discrete harmonic analysis is proving itself useful for computational issues. In a body of work that started in [50] and continued in [27] this was applied to property testing. The main result of [27] is a test of a general boolean function with n variables for the property of depending on only k on them, where the number of queries depends only on k and the approximation parameter ϵ. Although in a future version of [27] harmonic analysis will play a lesser role, it still remains the source of many insights about the nature of boolean functions.

Recent general results: With the advancement in the understanding of what properties are testable comes advancement in the understanding of how a property tester works. For graph properties in the dense model we already have some knowledge of how a property tester looks like in general, as exemplified in [40], despite still being far from classifying all the testable properties in the dense graph setting.

In [19] a general method for obtaining positive property testing results is developed. Although only time will tell if this new method becomes widely used, it looks promising in view of the fact that until now the methods used for proving upper bounds were usually narrower in scope (the most general method in this respect is probably the use of Szemerédi's Regularity Lemma for dense graph properties; for proving negative results there is a very general method in the form of Yao's principal).

9.2 Some topics related to testability

Property testing is already making itself felt in other areas of computer science. The following are some areas of interaction.

Low-rank approximations: There are many applications to finding a low-rank matrix that approximates a given input matrix. In [33] it is shown how to construct a low-rank approximation for a given matrix (in the form of a fast oracle), in time that depends only on the approximation parameters and not on the size of the approximated matrix. While strictly speaking this is not a testability result, it does share some of the methods. Other results regarding fast approximation of matrices are given in [32].

Testing with a prover: This topic deals with accepting or rejecting an input based on communicating with a computationally unbounded prover. It is somewhat like interactive proofs, only the operating requirements of the verifier here are the ones used in the field of property testing.

The general problem is to distinguish between the case that the input satisfies a property, and the case that the input is ϵ-far from satisfying the property. In general the prover is allowed to supply a long (usually polynomial size) proof, but the verifier is still allowed only a small number of queries from the input as well as from the supplied proof.

Testing with a prover is investigated in [24] and [11]; in these the running time of the verifier is restricted as well, and this poses some restrictions on the size of the proof through the size of the allowable address-space. As expected, there are problems that are hard to test in the usual context but are easy to test with a prover. For example, it takes $\Omega(\sqrt{n})$ queries to test that a function $f : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ is a bijection, but if a prover provides a function g that is supposedly the inverse of f, then it takes a constant number of queries to test that f and g are indeed close to being inverses of each other.

Statistical deductions: In this context one would like to check certain properties of a distribution (say, the distribution of the heights of NBA basketball players), where closeness is defined by the L_1 norm $|A - B| = \sum_{i \in R} |A(i) - B(i)|$ (or the variation distance as defined in Section 8, which is half this norm). In here, instead of querying, one can only obtain a sample of values that were independently chosen using this distribution (in the above example, one can only measure the heights of randomly selected NBA players).

When the distribution is defined over a set of n elements, it takes many samples to 'ϵ-test' (under this new definition) for a property, but in many cases the number of required samples is still much less than the $\Omega(n \log n)$ samples needed to actually write down an approximation of the whole distribution. For example, based on the methods of [39], one can prove that $O(\sqrt{n} \log^c n)$ queries (for some global constant c) are sufficient to ϵ-test for the property of being the uniform distribution over the whole set. On the other hand, it was proven that at least $\Omega(\sqrt{n})$ samples are required for this. Further results in this direction include those about testing that two distributions (also given by samples) are close, and that a joint distribution (with two 'coordinates') is independent. These are given in [13] and [12] respectively.

Quantum testability: Just as other notions of computational and informational complexity have their quantum counterparts, a quantum counterpart of the notion of property testing is being developed as well. The first results are in [17], showing that in some cases quantum computers can efficiently test properties for which there exist no efficient classical test, while some other properties are equally hard to test in the quantum context. The investigation into quantum property testing is taken further in [31], where quantum testers for some algebraically motivated properties (which are hard to test classically) are constructed.

9.3 A few more questions

Let us conclude with a few possible future directions in property testing, that some of the people in the field find interesting. The following list is not meant to be exhaustive or representative, and was chosen based mainly on personal preference.

Weighted testing: In most of the cases described above, the distance used in the definition of being ϵ-far from satisfying a property is the hamming distance between inputs. It is interesting to investigate also counterparts of the above testability questions in which closeness is measured by a weighted hamming distance.

In the most lenient model it is assumed that the algorithm is given all the weights in advance; thus the weights are not considered as part of the input, but as part of the description of the parameters within which the algorithm is supposed to operate. For some of the results, such as the result in [46] about the testing of languages recognizable by a read-once bounded width oblivious branching program, the proof extends naturally to the weighted case. In other instances, such as that of testing that a graph is triangle free, the question of the weighted case seems very hard (though some special cases, such as that of a weight function generated by giving weights to the vertices and using their multiples for the edges, can still be easily deduced from the unweighted case).

While it seems very probable that for many properties testability in fact does not extend to the weighted case, there are some recent positive advances. In [42] an even stricter model is considered, that of distribution-free testing. In it, the algorithm is not given the weights in advance, but is only given the ability to obtain independent samples of the distribution corresponding to the weight function. This model in a sense connects property testing with the topic of statistical deductions described above, and is motivated by a similar model from learning theory. One of the main results in [42] supplies some non-trivial upper bounds on testing d-dimensional matrices for monotonicity (on the other hand, these are still far from the bounds known for the unweighted case).

Additional kinds of queries: In certain settings it may be interesting to investigate what can be done if we allow queries that are more powerful than the usual "what is the input value at location i". Of course allowing queries that are too powerful can easily lead to a trivializing of the problem, but there may still be instances in which allowing certain types of queries is both natural and feasible in some real-life scenarios. There is already an investigation in [18] into allowing 'range queries' in the geometric context, i.e. queries as to whether there exist input points in a specified cube, and which are expected to provide one such point (chosen arbitrarily) if any such point exists.

Connections to traditional complexity: There are properties that are hard to decide but are easy to test (such as graph 3-colorability), and properties that are easy to decide (even AC_0 ones) but are hard to test. It would be interesting to take the approximating feature of testing into account, and formulate some connections between testable properties (subject to some uniformity assumptions on the testing algorithms) and easy to decide properties that approximate them, apart from the nearly obvious connection to Promise-BPP (for graph properties it can also be made into a connection with P, using the algorithmic version of Szemerédi's Regularity Lemma from [3]).

Another somewhat related question is this: What is the expressive power of first order graph expressions when approximations are taken into account? The results of [4] suggest that it may be stronger then expected.

Adaptive versus non-adaptive testing: A reminder – a non-adaptive testing algorithm is an algorithm that has to supply all the queries in advance, and only then is given the corresponding input values, based on which it accepts or rejects the input. In general, an adaptive ϵ-test for a property of binary sequences using q queries can be converted to a non-adaptive one using up to $2^q - 1$ queries – one just goes over the whole decision tree of the adaptive algorithm, makes all queries appearing there in advance, and then navigates the tree using the query results in order to accept or reject the input. There are known properties for which this gap is essential, for example properties derived from the sparse graph model investigated in [38]; the discussion in [26] contains some examples and details.

For graph properties in the dense graph model, in [40] it is proven that an adaptive algorithm making q queries can be converted to a non-adaptive one making only $\binom{2q}{2}$ queries (this is stated above as Lemma 4.1). A look at the simple proof reveals that it is based on the fact that graph properties are invariant with respect to graph isomorphisms. Examining other properties, such as counting properties that are invariant with respect to any permutation of the input (in [10] arguments related to such invariance are used for proving lower bounds for certain approximation problems), one gets the impression that a larger invariance group of a property implies a smaller gap between its adaptive and non-adaptive testing possibilities. It would be interesting to formalize and prove this observation.

References

[1] N. Alon, Testing subgraphs in large graphs, *Random Structures and Algorithms* 21 (2002), 359-370.

[2] N. Alon, S. Dar, M. Parnas and D. Ron, Testing of clustering, *Proceedings of the* 41st *IEEE FOCS* (2000), 240–250.

[3] N. Alon, R. A. Duke, H. Lefmann, V. Rödl and R. Yuster, The algorithmic aspects of the Regularity Lemma, *Journal of Algorithms* 16 (1994), 80–109.

[4] N. Alon, E. Fischer, M. Krivelevich and M. Szegedy, Efficient testing of large graphs, *Combinatorica* 20 (2000), 451–476.

[5] N. Alon, M. Krivelevich, I. Newman and M. Szegedy, Regular languages are testable with a constant number of queries, *SIAM Journal on Computing* 30 (2001), 1842–1862.

[6] N. Alon and A. Shapira, A characterization of easily testable induced subgraphs, manuscript.

[7] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, Proof verification and the hardness of approximation problems, *Journal of the ACM* 45 (1998), 501–555 (a preliminary version appeared in Proc. 33rd FOCS, 1992).

[8] S. Arora and S. Safra, Probabilistic checking of proofs: A new characterization of NP, *Journal of the ACM* 45 (1998), 70–122 (a preliminary version appeared in Proc. 33rd FOCS, 1992).

[9] S. Arora and M. Sudan, Improved low-degree testing and its applications, *Proceedings of the* 29th *ACM STOC* (1997), 485–495.

[10] Z. Bar-Yossef, R. Kumar and D. Sivakumar, Sampling algorithms: Lower bounds and applications, *Proceedings of the* 33rd *ACM STOC* (2001), 266–275.

[11] T. Batu, R. Rubinfeld and P. White, Fast approximation PCPs for multidimensional bin-packing problems, *Proceedings of the* 3rd *International Workshop on Randomization and Approximation Techniques in Computer Science* (1999), LNCS volume 1671, 246–256.

[12] T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld and P. White, Testing random variables for independence and identity, *Proceedings of the* 42nd *IEEE FOCS* (2001), 442–451.

[13] T. Batu, L. Fortnow, R. Rubinfeld, W. Smith and P. White, Testing that distributions are close, *Proceedings of the* 41st *IEEE FOCS* (2000), 259–269.

[14] F. A. Behrend, On sets of integers which contain no three terms in arithmetic progression, *Proceedings of the National Academy of Sciences of the United States of America* 32 (1946), 331–332.

[15] M. A. Bender and D. Ron, Testing properties of directed graphs: Acyclicity and connectivity *Random Structures and Algorithms* 20 (2002), 184–205.

[16] M. Blum, M. Luby and R. Rubinfeld, Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences* 47 (1993), 549–595 (a preliminary version appeared in Proc. 22nd STOC, 1990).

[17] H. Buhrman, L. Fortnow, I. Newman and H. Röhrig, Quantum property testing, *Proceedings of the* 14th *ACM-SIAM SODA* (2003), 480–488.

[18] A. Czumaj and C. Sohler, Property testing with geometric queries, *Proceedings of the* 9th *Annual European Symposium on Algorithms (ESA)* (2001), LNCS volume 2161, 266–277.

[19] A. Czumaj and C. Sohler, Abstract combinatorial programs and efficient property testers, *Proceedings of the* 43rd *IEEE FOCS* (2002), 83–92.

[20] A. Czumaj, C. Sohler and M. Ziegler, Property testing in computational geometry, *Proceedings of the 8th Annual European Symposium on Algorithms (ESA)* (2000), LNCS volume 1879, 155–166.

[21] R. Diestel, *Graph Theory* (2nd edition), Springer (2000).

[22] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron and A. Samorodnitsky, Improved testing algorithms for monotonicity, *Proceedings of the* 3rd *International Workshop on Randomization and Approximation Techniques in Computer Science* (1999), 97–108.

[23] F. Ergün, S. Kannan, R. Kumar, R. Rubinfeld and M. Viswanathan, Spot checkers, *Journal of Computer and System Science* 60 (2000), 717–751 (a preliminary version appeared in Proc. 30th STOC, 1998).

[24] F. Ergün, R. Kumar and R. Rubinfeld, Fast approximate PCPs, *Proceedings of the* 31st *ACM STOC* (1999), 41–50.

[25] E. Fischer, Testing graphs for colorability properties, *Proceedings of the* 12th *ACM-SIAM SODA* (2001), 873–882.

[26] E. Fischer, On the strength of comparisons in property testing, *Information and Computation*, in press (also available as ECCC TR00-083).

[27] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky, Testing juntas, *Proceedings of the* 43rd *IEEE FOCS* (2002), 103–112.

[28] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld and A. Samorodnitsky, Monotonicity testing over general poset domains, *Proceedings of the* 34th *ACM STOC* (2002), 474–483.

[29] E. Fischer and I. Newman, Testing of matrix properties, *Proceedings of the* 33rd *ACM STOC* (2001), 286–295.

[30] E. Fischer and I. Newman, Functions that have read-twice constant width branching programs are not necessarily testable, *The* 17th *IEEE Conference on Computational Complexity* (2002), 73–79; an extended version together with J. Sgall will appear in *Random Structures and Algorithms.*

[31] K. Friedl, F. Magniez, M. Santha and P. Sen, Quantum testers for hidden group properties, *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science* (2003).

[32] A. Frieze and R. Kannan, Quick approximation to matrices and applications, *Combinatorica* 19 (1999), 175-220.

[33] A. Frieze, R. Kannan and S. Vempala, Fast Monte-Carlo algorithms for finding low-rank approximations, *Proceedings of the* 39th *IEEE FOCS* (1998), 370–378.

[34] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan and A. Wigderson, Self-testing/correcting for polynomials and for approximate functions, *Proceedings of the* 23rd *ACM STOC* (1991), 32–42.

[35] O. Goldreich, Combinatorial property testing – a survey, In: *Randomization Methods in Algorithm Design* (P. Pardalos, S. Rajasekaran and J. Rolim eds.), AMS-DIMACS (1998), 45–60.

[36] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron and A. Samorodnitsky, Testing monotonicity, *Combinatorica* 20 (2000), 301–337 (a preliminary version written by the first four authors appeared in Proc. 39^{th} FOCS, 1998).

[37] O. Goldreich, S. Goldwasser and D. Ron, Property testing and its connection to learning and approximation, *Journal of the ACM* 45 (1998), 653–750 (a preliminary version appeared in Proc. 37^{th} FOCS, 1996).

[38] O. Goldreich and D. Ron, Property testing in bounded degree graphs, *Algorithmica* 32 (2002), 302–343 (a preliminary version appeared in Proc. 29^{th} STOC, 1997).

[39] O. Goldreich and D. Ron, On testing expansion in bounded-degree graphs, manuscript (available as ECCC TR00-20).

[40] O. Goldreich and L. Trevisan, Three theorems regarding testing graph properties, *Random Structures and Algorithms* 23 (2003), 23–57.

[41] W. T. Gowers, Lower bounds of tower type for Szemerédi's Uniformity Lemma, *Geometric and Functional Analysis* 7 (1997), 322–337.

[42] S. Halevi and E. Kushilevitz, Distribution-free monotonicity testing *Proceedings of the* 7^{th} *RANDOM + 6th APPROX* (2003).

[43] M. Kiwi, F. Magniez and M. Santha, Exact and approximate testing/correcting of algebraic functions: A survey, *Proceedings of* 1^{st} *Summer School on Theoretical Aspects of Computer Science* (2000), LNCS volume 1770, 302–313 (also available as ECCC TR01-014).

[44] J. Komlós and M. Simonovits, Szemerédi's Regularity Lemma and its applications in graph theory, In: *Combinatorics, Paul Erdös is Eighty*, Vol II (D. Miklós, V. T. Sós, T. Szönyi eds.), János Bolyai Math. Soc., Budapest (1996), 295–352.

[45] L. Lovász and N. Young, Lecture notes on evasiveness of graph properties, Technical report, Princeton University, 1994. Available at:
`http://www.uni-paderborn.de/fachbereich/AG/agmadh/WWW/english/scripts.html`

[46] I. Newman, Testing membership in languages that have small width branching programs, *SIAM Journal on Computing* 31 (2002), 1557–1570.

[47] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley (1994).

[48] M. Parnas and D. Ron, Testing metric properties, *Proceedings of the* 33rd *ACM STOC* (2001), 276–285.

[49] M. Parnas, D. Ron and R. Rubinfeld, Testing parenthesis languages, *Proceedings of the* 5th *International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)* (2001), 261–272.

[50] M. Parnas, D. Ron, and A. Samorodnitsky, Testing basic boolean formulae, *SIAM Journal on Discrete Mathematics* 16 (2002), 20–46.

[51] V. Rödl and R. Duke, On graphs with small subgraphs of large chromatic number, *Graphs and Combinatorics* 1 (1985), 91–96.

[52] D. Ron, Property testing (a tutorial), In: *Handbook of Randomized Computing* (S. Rajasekaran, P. M. Pardalos, J. H. Reif and J. D. P. Rolim eds), Kluwer Press (2001).

[53] R. Rubinfeld, On the robustness of functional equations, *SIAM Journal on Computing* 28 (1999), 1972–1997.

[54] R. Rubinfeld and M. Sudan, Robust characterization of polynomials with applications to program testing, *SIAM Journal of Computing* 25 (1996), 252–271 (first appeared as a technical report, Cornell University, 1993).

[55] R. Rubinfeld and M. Sudan, Testing polynomial functions efficiently and over rational domains, *Proceedings of the* 3rd *Annual ACM-SIAM Symposium on Discrete Algorithms* (1992), 23–43.

[56] E. Szemerédi, Regular partitions of graphs, In: *Proc. Colloque Inter. CNRS* No. 260 (J. C. Bermond, J. C. Fournier, M. Las Vergnas and D. Sotteau eds.), 1978, 399–401.

[57] A. C. Yao, Probabilistic computation, towards a unified measure of complexity. *Proceedings of the* 18th *IEEE FOCS* (1977), 222–227.