Introduction to Testing Graph Properties

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.
E-mail: oded@wisdom.weizmann.ac.il.

March 21, 2010

Abstract

The aim of this article is to introduce the reader to the study of testing graph properties, while focusing on the main models and issues involved. No attempt is made at providing a comprehensive survey of this study, and specific results are often mentioned merely as illustrations of general themes.

Keywords: Property Testing, Graph Properties,

Contents

1 The General Context

In general, property testing is concerned with super-fast (probabilistic) algorithms for deciding whether a given object has a predetermined property or is *far* from any object having this property. Such algorithms, called testers, obtain local views of the object by making adequate queries; that is, the object is seen as a function and the tester gets oracle access to this function, and thus may be expected to work in time that is sub-linear in the size of the object.

Looking at the foregoing formulation, we first note that property testing is concerned with promise problems (cf. [26, 29]), rather than with standard decision problems. Specifically, objects that neither have the property nor are far from having the property are being discarded. The exact formulation of these promise problems refers to a *distance measure* defined on the set of all relevant objects (i.e., this distance measure coupled with a distance parameter determine the set of objects that are far from the property). Thus, the choice of natural distance measures is crucial to the study of property testing. Secondly, we note that the requirement that the algorithms operate in sub-linear time (i.e., without reading their entire input) calls for a specification of the *type of queries* that these algorithms can make to their input. Thus, the choice of natural query types is also crucial to the study of property testing. These two general considerations will become concrete when we get to the actual subject matter (i.e., testing graph properties).

1.1 Why Graphs?

Let us start with an empirical observation, taken from Shimon Even's book *Graph Algorithms* [25] (published in 1979):

> Graph theory has long become recognized as one of the more useful mathematical subjects for the computer science student to master. The approach which is natural in computer science is the algorithmic one; our interest is not so much in existence proofs or enumeration techniques, as it is in finding efficient algorithms for solving relevant problems, or alternatively showing evidence that no such algorithms exist. Although algorithmic graph theory was started by Euler, if not earlier, its development in the last ten years has been dramatic and revolutionary.

Meditating on these facts, one may ask what is the source of this ubiquitous use of graphs in computer science. The most common answer is that graphs arise naturally as a model (or an abstraction) of numerous natural and artificial objects. Another answer is that graphs visualize binary relations over finite sets. These two different answers correspond to two types of models of testing graph properties that will be discussed below.

1.2 Why Testing?

Suppose we are given a huge graph representing some binary relation over a huge data-set (see below), and we need to determine whether the graph (equivalently, the relation) has some predetermined property. Since the graph is huge, we cannot or do not want to even scan all of it (let alone process all of it). The question is whether it is possible to make meaningful statements about the entire graph based only on a "small portion" of it. Of course, such statements will at best be approximations. But in many settings approximations are good enough.

As a motivation, let us consider a well-known example in which fast approximations are possible and useful. Suppose that some cost function is defined over a huge set, and that one wants to obtain the average cost of an element in the set. To be more specific, let $\mu : S \to [0,1]$ be a cost function, and suppose we want to estimate $\overline{\mu} \stackrel{\rm def}{=} \frac{1}{|S|}\sum_{x\in S}\mu(x)$. Then, uniformly (and independently) selecting $m \stackrel{\rm def}{=} O(\epsilon^{-2}\log(1/\delta))$ sample points, $x_1,...,x_m$, in S we obtain with probability at least $1-\delta$ an estimate of $\overline{\mu}$ within $\pm\epsilon$. That is,

$$\mathbf{Pr}_{x_1,...,x_m\in S}\left[\left|\frac{1}{m}\sum_{i=1}^{m}\mu(x_i) \;-\; \overline{\mu}\right| \;>\; \epsilon\right] \;<\; \delta\,. \tag{1}$$

Graphs capture more complex features of data sets; that is, relations among pairs of elements (rather then functions of single elements). Specifically, a symmetric binary relation $R \subseteq S\times S$ is represented by a graph $G=(S,R)$, where the elements of S are viewed as vertices and the elements in R are viewed as edges.

The study of testing graph properties reveals that sampling a huge data set may be useful not only towards approximating various statistics regarding a function defined over the set but also towards approximating various properties regarding a binary relation defined on this set. As we shall see, in many cases, the sampling method used (or at least its analysis) is significantly more sophisticated than the one employed in gathering statistics of the former type. But before doing so, we wish to further discuss the potential benefit in the notion of approximation underlining the definition of property testing.

Firstly, being close to a graph that has the property is a notion of approximation that, in certain applications, may be of direct value. Furthermore, in some cases, being close to a graph having the property translates to a standard notion of approximation (see Section 2.2). In other cases, it translates to a notion of "dual approximation" (see, again, Section 2.2).

Secondly, in some cases, we may be forced to take action without having time to run a decision procedure, while given the option of modifying the graph in the future, at a cost proportional to the number of added/omitted edges. For example, suppose we are given a graph that represents some suggested design, where bipartite graphs correspond to good designs and changes in the design correspond to edge additions/omissions. Using a `Bipartiteness` tester, we may (with high probability) accept any good design, while rejecting designs that will cost a lot to modify. That is, we may still accept designs that are not good, but only such that are close to being good and thus will not cost too much to modify later.

Thirdly, we may use the property tester as a preliminary stage before running a slower exact decision procedure. In case the graph is far from having the property, with high probability, we obtain an indication towards this fact, and save the time we might have used running the decision procedure. Furthermore, if the tester has one-sided error (i.e., it always accepts a graph having the property) and the tester has rejected, then we have obtained an absolutely correct answer without running the slower decision procedure at all. The saving provided by using a property tester as a preliminary stage may be very substantial in many natural settings where *typical* graphs either have the property or are very far from having the property. Furthermore, *if* it is *guaranteed* that graphs either have the property or are very far from having it *then* we may not even need to run the decision procedure at all.

1.3 Three Models of Testing Graph Properties

A graph property is a set of graphs closed under graph isomorphism (renaming of vertices). Let Π be such a property. A Π-tester is a *randomized* algorithm that is given oracle access to a graph, $G=(V,E)$, and has to determine whether the graph is in Π or is far from being in Π. The type of oracle (equiv., the type of queries allowed) and distance-measure depend on the model, and we focus on three such models:

1. The adjacency predicate model [31]: Here the Π-tester is given oracle access to a symmetric function $g : V \times V \to \{0,1\}$ that represents the adjacency predicate of the graph G; that is $g(u,v) = 1$ if and only if $(u,v) \in E$. In this model distances between graphs are measured according to their representation; that is, if the graphs G and G' are represented by the functions g and g', then their relative distance is the fraction of pairs (u,v) such that $g(u,v) \neq g'(u,v)$.

   Note that saying that $G = ([N], E)$ is ϵ-far from the graph property Π means that for every $G' \in \Pi$ it holds that G is ϵ-far from G'. Since Π is closed under graph isomorphism, this means that G is ϵ-far from any isomorphic copy of $G' = ([N], E')$; that is, for every permutation π over $[N]$, it holds that $|\{(u,v) : g(u,v) \neq g'(\pi(u), \pi(v))\}| > \epsilon N^2$, where g and g' are as above.

   Finally, note that this notion of distance between graphs is most meaningful in the case that the graph is dense (since in this case fractions of the number of possible vertex pairs are closely related to fractions of the actual number of edges). Thus, this model is often called the dense graph model.

2. The incidence function model [33]: Here, for some fixed upper bound d (on the degrees of vertices in G), the Π-tester is given oracle access to a function $g : V \times [d] \to V \cup \{\perp\}$ that represents the graph $G = (V, E)$ such that $g(u,i) = v$ if v is the i^{th} vertex incident at u and $g(u,i) = \perp$ if u has less than i neighbors. That is, $E = \{(u,v) : \exists i \ f(u,i) = v\}$, where we always assume that $g(u,i) = v$ if and only if there exists a $j \in [d]$ such that $g(v,j) = u$.

   Indeed, only graphs of degree at most d can be represented in this model, which is called the bounded-degree graph model.

   In this model too, distances between graphs are measured according to their representation, but here the representation is different and so the distances are different. Specifically, if the graphs G and G' are represented by the functions g and g', then their relative distance is the fraction of pairs (u,i) such that $g(u,i) \neq g'(u,i)$. Again, saying that $G = ([N], E)$ is ϵ-far from the graph property Π means that for every $G' \in \Pi$ it holds that G is ϵ-far from G'. Since Π is closed under graph isomorphism (and the ordering of the vertices incident at each vertex is arbitrary), this means that for every permutation π over $[N]$, it holds that

   $$\sum_{u \in V} |\{v : \exists i \ g(u,i) = v\} \triangle \{v : \exists i \ g'(\pi(u), i) = \pi(v)\}| \ > \ \epsilon d N ,$$

   where g and g' are as above, and $\triangle$ denotes the symmetric difference (i.e., $A \triangle B = (A \cup B) \setminus (A \cap B)$).

3. The general graph model [51, 45]: In contrast to the foregoing two models in which the oracle queries and the distances between graphs are linked to the representation of graphs as functions, in the following model the representation is blurred and the query types and distance measure are decoupled.

   The relative distance between the graphs $G = ([N], E)$ and $G' = ([N], E)$ is usually defined as $\frac{|E \triangle E'|}{\max(|E|,|E'|)}$; that is, the absolute distance is normalized by the actual number of edges rather than by an absolute upper bound (on the number of edges) such as $N^2/2$ or $dN/2$.

   The types of queries typically considered are the two types of queries considered in the previous two models. That is, the algorithm may ask whether two vertices are adjacent in the graph and may also ask for a specific neighbor of a specific vertex.

Needless to say, the general graph model is the most general one, and it is indeed closest to actual algorithmic applications. The fact that this model has so far received relatively little attention merely reflects the fact that its study is overly complex. Given that current studies of the other models still face formidable difficulties (and that these models offer a host of interesting open problems), it is natural that researchers shy away from yet another level of complication.

The current focus on query complexity. Although property testing is motivated by referring to super-fast algorithms, research in the area tends to focus on the *query complexity* of testing various properties. This focus should be viewed as providing an initial estimate to the actual complexity of the testing problems involved; certainly, query complexity lower bounds imply corresponding bounds on the time complexity, whereas the latter is typically at most exponential in the query complexity. Furthermore, in many cases, the time complexity is polynomial in the query complexity and this fact is typically stated. Thus, we will follow the practice of focusing on the query complexity of testing, but also mention time complexity upper bounds whenever they are of interest.

1.4 Organization

The following three sections are devoted to the three models discussed above: We start with the dense graph model (Section 2), then move to the bounded-degree model (Section 3), and finally get to the general graph model (Section 4). In each model we review the definition of testing (when specialized to that model), provide a taste of the known results, and demonstrate some of the ideas involved (by focusing on testing Bipartiteness, which seems a good benchmark).

We conclude this article with a discussion of a few issues that are relevant to all models; these include the treatment of directed graphs (Section 5.1), the related notions of tolerant testing and distance approximation (Section 5.2), and the notion of proximity oblivious testing (Section 5.3).

The appendix presents three observations that occurred to us in the process of writing this article. These refer to testing (degree) regularity in the dense graph model (Appendix A.1), non-adaptive testers in the bounded-degree graph model (Appendix A.2), and testing strong connectivity with forward queries only (Appendix A.3).

2 The Dense Graph Model

In the adjacency matrix model (a.k.a the dense graph model), an N-vertex graph $G = ([N], E)$ is represented by the Boolean function $g : [N] \times [N] \to \{0,1\}$ such that $g(u,v) = 1$ if and only if u and v are adjacent in G (i.e., $\{u,v\} \in E$). Distance between graphs is measured in terms of their aforementioned representation (i.e., as the fraction of (the number of) different matrix entries (over N^2)), but occasionally one uses the more intuitive notion of the fraction of (the number of) unordered vertex pairs over $\binom{N}{2}$.[1] Recall that we are interested in *graph properties*, which are sets of graphs that are closed under isomorphism; that is, Π is a graph property if for every graph $G = ([N], E)$ and every permutation π of $[N]$ it holds that $G \in \Pi$ if and only if $\pi(G) \in \Pi$, where $\pi(G) \stackrel{\rm def}{=} ([N], \{\{\pi(u), \pi(v)\} : \{u,v\} \in E\}$. We now spell out the meaning of property testing in this model.

Definition 2.1 (testing graph properties in the adjacency matrix model): *A* tester *for a graph property Π is a probabilistic oracle machine that, on input parameters N and ϵ and access to* (the

[1]Indeed, there is a tiny discrepancy between these two measures, but it is immaterial in all discussions.

adjacency predicate of) *an N-vertex graph $G = ([N], E)$, outputs a binary verdict that satisfies the following two conditions.*

1. *If $G \in \Pi$ then the tester accepts with probability at least 2/3.*
2. *If G is ϵ-far from Π then the tester accepts with probability at most 1/3, where G is* ϵ-far *from Π if for every N-vertex graph $G' = ([N], E') \in \Pi$ it holds that the symmetric difference between E and E' has cardinality that is greater than $\epsilon \cdot \binom{N}{2}$.*

If the tester accepts every graph in Π with probability 1, then we say that it has one-sided error. *A tester is called* non-adaptive *if it determines all its queries based solely on its internal coin tosses* (and the parameters N and ϵ)*; otherwise it is called* adaptive.

The query complexity of a tester is the number of queries it makes to any N-vertex graph, as a function of the parameters N and ϵ. We say that a tester is efficient if it runs in time that is polynomial in its query complexity, where basic operations on elements of $[N]$ (and in particular, uniformly selecting an element in $[N]$) are counted at unit cost.

We stress that testers are defined as uniform algorithms that are given the size parameter N and the distance (or proximity) parameter ϵ as explicit inputs. This uniformity (over the values of the distance parameter) makes the positive results stronger and more appealing (especially in light of a separation result shown in [10]). In contrast, negative results typically refer to a fixed value of the distance parameter.

The study of property testing in the dense graph model was initiated by Goldreich, Goldwasser, and Ron [31], as a concrete and yet general framework for the study of property testing at large. From that perspective, it was most natural to represent graphs as Boolean functions, and the adjacency matrix representation was the obvious choice. This dictated the choice of the type of queries as well as the distance measure. In retrospect, the dense graph model seems most natural when graphs are viewed as representing generic (symmetric) binary relations (cf. the second motivation to the study of graphs mentioned in Section 1.1 as well as the discussion of sampling in Section 1.2).

2.1 A Taste of the Known Results

We first mention that graph properties of arbitrary query complexity are known: Specifically, in this model, graph properties (even those in $\mathcal{P}$) may have query complexity ranging from $O(1/\epsilon)$ to $\Omega(N^2)$, and the same holds also for monotone graph properties in $\mathcal{NP}$ (cf. [32]).[2] In this overview, we focus on properties that can be tested within query complexity that only depends on the proximity parameter (i.e., ϵ). Interestingly, there is much to say about this class of properties. Let us start with a brief summary, and provide more details later.

1. A celebrated result of Alon, Fischer, Newman, and Shapira [3] provides a combinatorial characterization of the class of properties that can be tested within query complexity that only depends on the proximity parameter. This class contains natural properties that are not testable in query complexity poly$(1/\epsilon)$; see [1].

[2] We mention that a full query complexity hierarchy is established in [32] by using unnatural graph properties, starting from the $\Omega(N^2)$ lower bound of [31], which also uses an unnatural graph property. In contrast, the $\Omega(N)$ lower bound established in [27] (following [2]) refers to the natural property of testing whether an N-vertex graph consists of two isomorphic copies of some $N/2$-vertex graph.

2. The prior work of Goldreich, Goldwasser, and Ron [31] provides a natural class of properties that can be tested within query complexity $\mathrm{poly}(1/\epsilon)$. This class consists of so-called "partition problems" and includes sets such as k-colorability, for any fixed $k \geq 2$, and graphs containing a clique for density ρ, for any fixed $\rho > 0$.

3. A relatively recent work of Goldreich and Ron [37] initiates a study of the class of properties that can be tested within query complexity $\widetilde{O}(1/\epsilon)$.

Before providing more details on the foregoing results, we mention that, when disregarding a possible quadratic blow-up in the query complexity, we may assume that the tester in canonical in the following sense.

Theorem 2.2 (canonical testers [39, Thm 2]): *Let Π be any graph property. If there exists a tester with query complexity $q(N,\epsilon)$ for Π, then there exists a tester for Π that uniformly selects a set of $O(q(N,\epsilon))$ vertices and accepts iff the induced subgraph has property Π', where Π' is a fixed graph property that may depend on N as well as on Π. Furthermore, if the original tester has one-sided error, then so does the new tester, and a sample of $2q(N,\epsilon)$ vertices suffices*

Indeed, the resulting tester is called canonical. We warn that Π' need not equal Π (let alone that Π' may depend on N), and that the time complexity of the canonical tester may be significantly larger than the time complexity of the original tester.

2.1.1 Testability in $q(\epsilon)$ queries, for any function q

As stated above, a celebrated result of Alon *et al.* [3] provides a combinatorial characterization of the class of properties that can be tested within query complexity that only depends on the proximity parameter. This characterization refers to the notion of a *regularity instance*, where regularity is in the sense of Szemerédi's Regularity Lemma [56]. The result essentially asserts that a graph property can be tested in query complexity that only depends on ϵ if and only if it can be characterized in terms of a constant number of regularity instances. The lesson from this characterization is that, when ignoring the specific dependency on ϵ, *testing graph properties in query complexity that only depends on ϵ reduces to graph regularity.* This lesson makes more concrete the feeling already raised by Theorem 2.2 that testing in this model reduces to combinatorics.

The downside of the algorithms that emerge from this characterization is that their query complexity is related to the proximity parameter via a function that grows tremendously fast. Specifically, the query complexity is only upper bounded by $\mathtt{tf}(\mathtt{tf}(\mathrm{poly}(1/\epsilon)))$, where $\mathtt{tf}$ is the tower function defined inductively by $\mathtt{tf}(n) = \exp(\mathtt{tf}(n-1))$ with $\mathtt{tf}(1) = 2$. (Indeed, $\mathtt{tf}$ is the inverse of the $\log^*$ function.)

Interestingly, it is known that a super-polynomial dependence on the proximity parameter is inherent to the above result. Actually, as shown by Alon [1], such a dependence is essential even for testing *triangle freeness*. Indeed, this fact provides a nice demonstration of the non-triviality of testing graph properties. *One might have guessed that $O(1/\epsilon)$ or $O(1/\epsilon^3)$ queries would have sufficed to detect a triangle in any graph that is ϵ-far from being triangle freeness, but Alon's result asserts that this guess is wrong*, whereas (in contrast) $\mathtt{tf}(\mathrm{poly}(1/\epsilon))$ queries do suffice [1].

Perspective. It is indeed an amazing fact that many properties can be tested within (query) complexity that only depends on the proximity parameter (rather than also on the size of the object being tested). This amazing statement seems to put in shadow the question of what is the form of the aforementioned dependence, and blurs the difference between a reasonable dependence (e.g., a polynomial relation) and a prohibiting one (e.g., a tower-function relation). We beg to disagree with this sentiment and claim that, as in the context of standard approximation problems (cf. [43]), *the dependence of the complexity on the approximation* (or proximity) *parameter is a key issue.*

We wish to stress that we do value the impressive results of [2, 7, 8, 28] (let alone [3]), which refer to graph property testers having query complexity that is independent of the graph size but depend prohibitively on the proximity parameter. We view such results as an impressive first step, which called for further investigation directed at determining the actual dependency of the complexity on the proximity parameter.

While it is conceivable that there exist (natural) graph properties that can be tested in $\exp(1/\epsilon)$ queries but not in $\text{poly}(1/\epsilon)$ queries, we are not aware of such a property.[3] We thus move directly from complexities of the form $\texttt{tf}(1/\epsilon)$ (and more) to complexities of the form $\text{poly}(1/\epsilon)$.

2.1.2 Testability in $\text{poly}(1/\epsilon)$ queries

Testers of query complexity $\text{poly}(1/\epsilon)$ are known for several natural graph properties [31].

- k-`Colorability`, for any fixed $k \geq 2$. The query-complexity is $\text{poly}(k/\epsilon)$. For $k = 2$ the running-time is $\widetilde{O}(1/\epsilon^3)$, whereas for $k > 2$ the running-time is $\exp(\text{poly}(1/\epsilon))$ (and running-time polynomial in $1/\epsilon$ is unlikely, since k-`Colorability` is NP-complete, for $k \geq 3$).

  The k-`Colorability` tester has one-sided error; that is, in case the graph is k-colorable, the tester always accepts. Furthermore, when rejecting a graph, this tester always supplies a small counterexample (i.e., a $\text{poly}(1/\epsilon)$-size subgraph that is not k-colorable).

  The `2-Colorability` (equivalently, `Bipartiteness`) Tester is presented in §2.3. An improved analysis has been obtained by Alon and Krivelevich [4].

- ρ-`Clique`, for any fixed $\rho > 0$, where ρ-`Clique` is the set of graphs that have a clique of density ρ (i.e., N-vertex graphs having a clique of size ρN).

- ρ-`CUT`, for any fixed $\rho > 0$, where ρ-`CUT` is the set of graphs that have a cut of density at least ρ (compared to N^2).

  A generalization to k-way cuts has query-complexity $\text{poly}((\log k)/\epsilon)$.

- ρ-`Bisection`, for any fixed $\rho > 0$, where ρ-`Bisection` is the set of graphs that have a bisection of density at most ρ (i.e., an N-vertex graph is in ρ-Bisection if its vertex set can be partitioned into two equal parts with at most ρN^2 edges going between them).

Except for k-`Colorability`, all the other testers have two-sided error, and this is unavoidable for any tester of $o(N)$ query complexity for any of these properties.

All the above property testing problems are special cases of the General Graph Partition Testing Problem, which is parameterized by a set of lower and upper bounds. In this problem one needs to determine whether there exists a k-partition of the vertices so that the number of vertices in each part as well as the number of edges between each pair of parts falls between the corresponding lower and upper bounds (in the set of parameters). For example, ρ-clique is expressible as a 2-partition in which one part has ρN vertices, and the number of edges in this part is $\binom{\rho N}{2}$. A tester for the general problem also appears in [31]: The tester uses $\widetilde{O}(k^2/\epsilon)^{2k+O(1)}$ queries, and runs in time exponential in its query-complexity.

[3] Needless to say, demonstrating the existence of such (natural) properties is an interesting open problem.

From testing to searching. Interestingly, the testers for (all cases of) the General Graph Partition Problem can be modified into algorithms that find an (implicit representation of an) approximately adequate partition whenever it exists. That is, if the graph has the desired (partitioning) property, then the testing algorithm may actually output auxiliary information that allows to reconstruct, in $\text{poly}(1/\epsilon) \cdot N$-time, a partition that approximately obeys the property. For example, for ρ-CUT, we can construct a partition with at least $(\rho - \epsilon) \cdot N^2$ crossing edges. We comment that this notion of an implicit representation of an adequate structure may be relevant for other sets in $\mathcal{NP}$, where this structure corresponds to an NP-witness.

Back to testing graph properties. Although many natural graph properties can be formulated as partition problems, many other properties that can be tested with $\text{poly}(1/\epsilon)$ queries cannot be formulated as such problems. The list include the set of regular graphs, connected graphs, planar graphs, and more. We identify three classes of such natural properties:

1. Properties that only depends on the vertex degree distribution (e.g., degree regularity and average degree). For example, for any fixed $\rho > 0$, the set of N-vertex graphs having ρN^2 edges can be tested using $O(1/\epsilon^2)$ queries, which is the best result possible.[4] The same holds with respect to testing degree regularity, where the $\Omega(1/\epsilon^2)$ queries lower bound follows by reduction to estimating the average value of Boolean functions and a corresponding upper bound can be obtained by building on the $\widetilde{O}(1/\epsilon^3)$-query algorithm presented in the proof of [31, Prop. 10.2.1.3].[5]

2. Properties that are satisfied only by sparse graphs (i.e., N-vertex graphs having $O(N)$ edges)[6] such as `Cycle-freeness` and `Planarity`. These properties can be tested by rejecting any graph that is not sufficiently sparse (see [31, Prop. 10.2.1.2]).

3. Properties that are almost trivial in the sense that, for some constant $c > 0$ and every $\epsilon > N^{-c}$, all N-vertex graphs are ϵ-close to the property. For example, every N-vertex graph is N^{-1}-close to being connected (or being Hamiltonian or Eulerian). These properties can be tested by accepting any N-vertex graph if $\epsilon > N^{-c}$ (without making any query), and inspecting the entire graph otherwise (where, in this case $\binom{N}{2} = \text{poly}(1/\epsilon)$). (See [31, Prop. 10.2.1.1].)

In view of all of the foregoing, we believe that characterizing the class of graph properties that can be tested in $\text{poly}(1/\epsilon)$ queries may be very challenging. We mention that the special case of induced subgraph freeness properties was resolved in [9].

2.1.3 Testability in $\widetilde{O}(1/\epsilon)$ queries

While Theorem 2.2 may be interpreted as suggesting that testing in the dense graph model leaves no room for algorithmic design, this conclusion is valid only if one ignores a possible quadratic blow-up in the query complexity (and also disregards the time complexity). As advocated by Goldreich and Ron [37], a finer examination of the model, which takes into account the exact query complexity (i.e., cares about a quadratic blow-up), reveals the role of algorithmic design. In particular, the

[4] Both upper and lower bounds can be proved by reduction to the problem of estimating the average value of Boolean functions (cf. [22]).

[5] For the lower bound, consider the problem of distinguishing between a random N-vertex graph in which each vertex has degree either $(0.5+\epsilon)N$ or $(0.5-\epsilon)N$ and a random $(N/2)$-regular N-vertex graph. For the upper bound, see Appendix A.1.

[6] Actually, this class can be extended by considering a more relaxed notion of sparseness that includes N-vertex graphs having $O(N^{2-\Omega(1)})$ edges.

results in [37] distinguish adaptive testers from non-adaptive ones, and distinguish the latter from canonical testers. These results refer to testability in $\widetilde{O}(1/\epsilon)$ queries. In particular, it is shown that:

- Testing every "non-trivial for testing" graph property requires $\Omega(1/\epsilon)$ queries, even when adaptive testers are allowed. Furthermore, any canonical tester for such a property requires $\Omega(1/\epsilon^2)$ queries.
- There exists a natural graph property that can be tested by $\widetilde{O}(1/\epsilon)$ adaptive queries, requires $\Omega(\epsilon^{-4/3})$ non-adaptive queries, and is actually testable by $O(\epsilon^{-4/3})$ non-adaptive queries.
- There exists a natural graph property that can be tested by $\widetilde{O}(1/\epsilon)$ adaptive queries but requires $\Omega(\epsilon^{-3/2})$ non-adaptive queries.
- There exist an infinite class of natural graph properties that can be tested by $\widetilde{O}(1/\epsilon)$ non-adaptive queries.

All the above testers have one-sided error probability and are efficient, whereas the lower bounds hold also for two-sided error testers (regardless of efficiency).

The foregoing results seem to indicate that even at this low complexity level (i.e., testing in $\widetilde{O}(1/\epsilon)$ adaptive queries) there is a lot of structure and much to be understood. In particular, it is conjectured in [37] that, for every $t \geq 4$, there exists graph properties that can be tested by $\widetilde{O}(1/\epsilon)$ adaptive queries and have non-adaptive query complexity $\Theta(\epsilon^{-2+\frac{t}{2}})$.

2.1.4 Reflections

Let us reflect about some issues that arise from the foregoing exposition.

Adaptive testers versus non-adaptive ones. Recall that Theorem 2.2 asserts that canonical testers (which are in particular non-adaptive) have query complexity that is at most quadratic in the query complexity of general (possibly adaptive) testers. Still the results surveyed in §2.1.3 indicate that such a gap may exist. An interesting question, raised by Michael Krivelevich, is whether such a gap exists also for properties having query complexity that is significantly larger than $\widetilde{O}(1/\epsilon)$. In particular, we mention that testing `Bipartiteness`, which has non-adaptive query complexity $\widetilde{\Theta}(\epsilon^{-2})$ (cf. [4, 21])[7] and requires $\Omega(\epsilon^{-3/2})$ adaptive queries [21], may be testable in $o(\epsilon^{-2})$ adaptive queries (cf. [40]).

One-sided versus two-sided error probability. As noted above, for many natural properties there is a significant gap between the complexity of one-sided and two-sided error testers. For example, ρ-`CUT` has a two-sided error tester of query complexity poly$(1/\epsilon)$, but no one-sided error tester of query complexity $o(N^2)$. In general, the interested reader may contrast the characterization of two-sided error testers in [3] with the results in [8].

A contrast to recognizing graph properties. The notion of testing a graph property Π is a *relaxation* of the classical notion of *recognizing the graph property* Π, which has received much attention since the early 1970's (cf. [46]). In the classical (recognition) problem there are no margins of error; that is, one is required to accept all graphs having property Π and reject all graphs that lack

[7] The $\widetilde{O}(\epsilon^{-2})$ upper bound is due to [4], improving over [31], whereas the $\Omega(\epsilon^{-2})$ lower bound is due to [21].

property Π. In 1975, Rivest and Vuillemin resolved the Aanderaa–Rosenberg Conjecture, showing that any deterministic procedure for deciding any non-trivial monotone N-vertex graph property must examine $\Omega(N^2)$ entries in the adjacency matrix representing the graph. The query complexity of randomized decision procedures was conjectured by Yao to be $\Omega(N^2)$, and the currently best lower bound is $\Omega(N^{4/3})$. This stands in striking contrast to the aforementioned results regarding testing graph properties that establish that many natural (non-trivial) monotone graph properties can be *tested* by examining a constant number of locations in the matrix (where this constant depends on the constant value of the proximity parameter).

Graph properties are poor codes. We note that with the exception of two properties, which each contains a single N-vertex graph, the adjacency matrix representation of any property Π_N of N-vertex graphs yields a code over $\{0,1\}^{\binom{N}{2}}$ with relative distance at most $O(1/N)$. Specifically, if Π_N neither consists of the N-vertex clique nor of the N-vertex independent set, then Π_N contains a graph $G = ([N], E)$ that contains two vertices $u, v \in [N]$ that have different neighborhoods in G. Consider a permutation π that transposes u and v, while leaving the rest of $[N]$ intact, and let $G' = ([N], \{\pi(a), \pi(b) : (a,b) \in E\})$. Then $G' \in \Pi_N$, but G' is $\frac{2N}{\binom{N}{2}}$-close to G.

2.2 Testing versus other forms of Approximation

We shortly discuss the relation of the notion of approximation underlying the definition of testing graph properties to more traditional notions of approximation.

Application to the standard notion of approximation: The relation of testing graph properties to standard notions of approximation is best illustrated in the case of `Max-CUT`. Any tester for the set ρ-`CUT`, working in time $T(\epsilon, N)$, yields an algorithm for approximating the size of the maximum cut in an N-vertex graph, up to additive error ϵN^2, in time $\frac{1}{\epsilon} \cdot T(\epsilon, N)$. Thus, for any constant $\epsilon > 0$, using the above tester of [31], we can approximate the size of the max-cut to within ϵN^2 in constant time. This yields a constant time approximation scheme (i.e., to within any constant relative error) for dense graphs, which improves over previous work of Arora *et al.* [12] and de la Vega [24] who solved this problem in polynomial-time (i.e., in $O(N^{1/\epsilon^2})$–time and $\exp(\widetilde{O}(1/\epsilon^2)) \cdot N^2$–time, respectively). In the latter works the problem is solved by actually finding approximate max-cuts. Finding an approximate max-cut does not seem to follow from the mere existence of a tester for ρ-cut; yet, as mentioned above, the tester in [31] can be used to find such a cut in time linear in N.

Relation to "dual approximation" (cf. [43, Chap. 3]): To illustrate this relation, we consider the aforementioned ρ-`Clique` Tester. The traditional notion of approximating `Max-Clique` corresponds to distinguishing the case in which the max-clique has size at least ρN from, say, the case in which the max-clique has size at most $\rho N/2$. On the other hand, when we talk of testing ρ-`Clique`, the task is to distinguish the case in which an N-vertex graph has a clique of size ρN from the case in which it is ϵ-far from the class of N-vertex graphs having a clique of size ρN. This is equivalent to the "dual approximation" task of distinguishing the case in which an N-vertex graph has a clique of size ρN from the case in which any ρN subset of the vertices misses at least ϵN^2 edges. To demonstrate that these two tasks are vastly different we mention that whereas the former task is NP-Hard, for $\rho < 1/4$ (see [15, 41]), the latter task can be solved in $\exp(O(1/\epsilon^2))$-time, for any $\rho, \epsilon > 0$. We believe that there is no absolute sense in which one of these approximation tasks is more important than the other: Each of these tasks may be relevant in some applications and irrelevant in others.

2.3 A Benchmark: Testing Bipartiteness

The `Bipartite` tester is extremely simple: It selects a tiny, random set of vertices and checks whether the induced subgraph is bipartite.

Algorithm 2.3 (Bipartite Tester in the Dense Graph Model [31]): *On input N, d, ϵ and oracle access to an adjacency predicate of an N-vertex graph,* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$*:*

1. *Uniformly select a subset of $\widetilde{O}(1/\epsilon^2)$ vertices of* V.
2. *Accept if and only if the subgraph induced by this subset is bipartite.*

Step (2) amounts to querying the predicate on all pairs of vertices in the subset selected at Step (1), and testing whether the induced graph is bipartite (e.g., by running BFS). As will become clear from the analysis, it actually suffice to query only $\widetilde{O}(1/\epsilon^3)$ of these pairs. We comment that a more complex analysis due to Alon and Krivelevich [4] implies that the Algorithm 2.3 is a `Bipartite` Tester even if one selects only $\widetilde{O}(1/\epsilon)$ vertices (rather than $\widetilde{O}(1/\epsilon^2)$) in Step (1).

Theorem 2.4 [31]: *Algorithm 2.3 is a* `Bipartite` *Tester* (in the dense graph model)*. Furthermore, the algorithm always accepts a bipartite graph, and in case of rejection it provides a witness of length* $\mathrm{poly}(1/\epsilon)$ (that the graph is not bipartite).

Proof: Let R be the subset selected in Step (1), and $\mathrm{G_R}$ the subgraph of G induced by R. Clearly, if G is bipartite then so is $\mathrm{G_R}$, for any R. The point is to prove that if G is ϵ-far from bipartite then the probability that $\mathrm{G_R}$ is bipartite is at most $1/3$. Thus, from this point on we assume that at least ϵN^2 edges have to be omitted from G to make it bipartite.

We view R as a union of two disjoint sets U and S, where $t \stackrel{\mathrm{def}}{=} |\mathrm{U}| = O(\epsilon^{-1} \cdot \log(1/\epsilon))$ and $m \stackrel{\mathrm{def}}{=} |\mathrm{S}| = O(t/\epsilon)$. We will consider all possible partitions of U, and associate a partial partition of V with each such partition of U. The idea is that in order to be consistent with a given partition, $(\mathrm{U_1}, \mathrm{U_2})$, of U, all neighbors of $\mathrm{U_1}$ (respectively, $\mathrm{U_2}$) must be placed opposite to $\mathrm{U_1}$ (respectively, $\mathrm{U_2}$). We will show that, with high probability, most high-degree vertices in V do neighbor U and so are forced by its partition. Since there are relatively few edges incident to vertices that do not neighbor U, it follows that, with very high probability, each such partition of U is detected as illegal by $\mathrm{G_R}$. Details follow, but before we proceed let us stress the key observation: *It suffices to rule out relatively few* (partial) *partitions of* V (i.e., these induced by partitions of U), rather than all possible partitions of V.

We use the notations $\Gamma(v) \stackrel{\mathrm{def}}{=} \{u : (u, v) \in \mathrm{E}\}$ and $\Gamma(\mathrm{X}) \stackrel{\mathrm{def}}{=} \cup_{v \in \mathrm{X}} \Gamma(v)$. Given a partition $(\mathrm{U_1}, \mathrm{U_2})$ of U, we define a (possibly partial) partition, $(\mathrm{V_1}, \mathrm{V_2})$, of V so that $\mathrm{V_1} \stackrel{\mathrm{def}}{=} \Gamma(\mathrm{U_2})$ and $\mathrm{V_2} \stackrel{\mathrm{def}}{=} \Gamma(\mathrm{U_1})$ (assume, for simplicity that $\mathrm{V_1} \cap \mathrm{V_2}$ is indeed empty). As suggested above, if one claims that G can be "bi-partitioned" with $\mathrm{U_1}$ and $\mathrm{U_2}$ on different sides, then $\mathrm{V_1} = \Gamma(\mathrm{U_1})$ must be on the opposite side to $\mathrm{U_2}$ (and $\Gamma(\mathrm{U_1})$ opposite to $\mathrm{U_1}$). Note that the partition of U places no restriction on vertices that have no neighbor in U. Thus, we first ensure that *almost all* "influential" (i.e., "high-degree") vertices in V have a neighbor in U.

Technical Definition 2.4.1 (high-degree vertices and good sets): *We say that a vertex $v \in \mathrm{V}$ is of* high-degree *if it has degree at least $\frac{\epsilon}{3}N$. We call* U good *if all but at most $\frac{\epsilon}{3}N$ of the high-degree vertices in* V *have a neighbor in* U.

We comment that NOT insisting that a good set U neighbors *all* high-degree vertices allows us to show that, with high probability, a random U of size unrelated to the size of the graph is good. (In contrast, if we were to insist that a good U neighbors *all* high-degree vertices, then we would have had to use $|\mathrm{U}| = \Omega(\log N)$.)

Claim 2.4.2 *With probability at least* $5/6$*, a uniformly chosen set* U *of size* t *is good.*

Proof: For any high-degree vertex v, the probability that v does not have any neighbor in a uniformly chosen U is at most $(1-\epsilon/3)^t < \frac{\epsilon}{18}$ (since $t = \Omega(\epsilon^{-1}\log(1/\epsilon))$). Hence, the expected number of high-degree vertices that do not have a neighbor in a random set U is less than $\frac{\epsilon}{18} \cdot N$, and the claim follows by Markov's Inequality. □

Technical Definition 2.4.3 (disturbing a partition of U): *We say that an edge* disturbs *a partition* $(\mathrm{U}_1, \mathrm{U}_2)$ *of* U *if both its end-points are in the same* $\Gamma(\mathrm{U}_i)$*, for some* $i \in \{1,2\}$.

Claim 2.4.4 *For any good set* U *and any partition of* U*, at least* $\frac{\epsilon}{3}N^2$ *edges disturb the partition.*

Proof: Each partition of V has at least ϵN^2 violating edges (i.e., edges with both end-points on the same side). We upper bound the number of these edges that are not disturbing. Actually, we upper bound the number of edges that have an end-point that is not in $\Gamma(\mathrm{U})$.

- The number of edges incident to high-degree vertices that do not neighbor U is bounded by $\frac{\epsilon}{3}N \cdot N$ (since there are at most $\frac{\epsilon}{3}N$ such vertices).
- The number of edges incident to vertices that are not of high-degree vertices is bounded by $N \cdot \frac{\epsilon}{3}N$ (since each such vertex has at most $\frac{\epsilon}{3}N$ incident edges).

This leaves us with at least $\frac{\epsilon}{3}N^2$ violating edges connecting vertices in $\Gamma(\mathrm{U})$ (i.e., edges disturbing the partition of U). □

The theorem follows by observing that $\mathrm{G_R}$ is bipartite only if either (1) the set U is not good; or (2) the set U is good and there exists a partition of U so that none of the disturbing edges occurs in $\mathrm{G_R}$. Using Claim 2.4.2 the probability of event (1) is bounded by 1/6, whereas by Claim 2.4.4 the probability of event (2) is bounded by the probability that there exists a partition of U so that none of the corresponding $\geq \frac{\epsilon}{3}N^2$ disturbing edges has both edge-point in the second sample S. Actually, we pair the m vertices of S, and consider the probability that none of these pairs is a disturbing edge for a partition of U. Thus the probability of event (2) is bounded by

$$2^{|\mathrm{U}|} \cdot \left(1 - \frac{\epsilon}{3}\right)^{m/2} < \frac{1}{6}$$

where the inequality holds since $m = \Omega(t/\epsilon)$. The theorem follows. ■

Comment: The procedure employed in the proof yields a $\mathrm{poly}(1/\epsilon) \cdot N$-time algorithm for 2-partitioning a bipartite graph so that at most ϵN^2 edges lie within the same side. This is done by running the tester, determining a partition of U (defined as in the proof) that is consistent with the bipartite partition of R, and partitioning V as done in the proof (with vertices that do not neighbor U, or neighbor both $\mathrm{U}_1, \mathrm{U}_2$, placed arbitrarily). Thus, the placement of each vertex is determined by inspecting at most $\widetilde{O}(1/\epsilon)$ entries of the adjacency matrix. Furthermore, the aforementioned partition of U constitutes a succinct representation of the 2-partition of the entire graph. All this is a typical consequence of the fact that the analysis of the tester follows the "enforce-and-test" paradigm (see [54, Sec. 4]).

3 The Bounded-Degree Graph Model

The bounded-degree model refers to a fixed degree bound, denoted $d \geq 2$. An N-vertex graph $G = ([N], E)$ (of maximum degree d) is represented in this model by a function $g : [N] \times [d] \to \{0, 1, ..., N\}$ such that $g(v, i) = u \in [N]$ if u is the i^{th} neighbor of v and $g(v, i) = 0$ if v has less than i neighbors.[8] Distance between graphs is measured in terms of their aforementioned representation (i.e., as the fraction of (the number of) different array entries (over dN)), but occasionally we shall use the more intuitive notion of the fraction of (the number of) edges over $dN/2$. We now spell out the meaning of property testing in this model.

Definition 3.1 (testing graph properties in the bounded-degree model): *For a fixed d, a* tester *for a graph property Π is a probabilistic oracle machine that, on input parameters N and ϵ and access to* (the incidence function of) *an N-vertex graph $G = ([N], E)$ of maximum degree d, outputs a binary verdict that satisfies the following two conditions.*

1. *If $G \in \Pi$ then the tester accepts with probability at least $2/3$.*
2. *If G is ϵ-far from Π then the tester accepts with probability at most $1/3$, where G is* ϵ-far *from Π if for every N-vertex graph $G' = ([N], E') \in \Pi$ of maximum degree d it holds that the symmetric difference between E and E' has cardinality that is greater than $\epsilon \cdot dN/2$.*

One-sided testers and non-adaptive testers are defined as in Definition 2.1.

The query complexity of a tester is defined as in Section 2; ditto for its efficiency.

The study of property testing in the bounded-degree graph model was initiated by Goldreich and Ron [33], with the aim of allowing the consideration of sparse graphs, which appear in numerous applications (cf. the first motivation to the study of graphs mentioned in Section 1.1). The point was that the dense graph model seems irrelevant to sparse graphs, both because the distance measure that underlies it deems all sparse graphs as close to one another, and because adjacency queries seems unsuitable for sparse graphs. Sticking to the paradigm of representing graphs as functions, where both the distance measure and the type of queries are determined by the representation, the aforementioned representation seemed the most natural choice. Indeed, a conscious decision was (and is) made not to capture, at this point (and in this model), sparse graphs that do not have constant (or low) maximum degree.

3.1 A Taste of the Known Results

We first mention that, also in this model, graph properties of arbitrary query complexity are known: Specifically, in this model, graph properties (in $\mathcal{NP}$) may have query complexity ranging from $O(1/\epsilon)$ to $\Omega(N)$, and furthermore such properties are monotone and natural (cf. [32], which builds over [20]). In particular, testing 3-Colorability requires $\Omega(N)$ queries, whereas testing 2-Colorability (i.e., Bipartiteness) requires $\Omega(\sqrt{N})$ queries [33] and can be done using $\widetilde{O}(\sqrt{N}) \cdot \text{poly}(1/\epsilon)$ queries [34]. We also mention that many natural properties are testable in query complexity that only depends on the proximity parameter (i.e., ϵ). A partial list includes k-edge connectivity, for every fixed k, and Planarity (cf. [33] and [18], respectively). Details follow.

[8] For simplicity, we assume here that the neighbors of v appear in an arbitrary order in the sequence $g(v, 1), ..., g(v, \deg(v))$, where $\deg(v) \stackrel{\text{def}}{=} |\{i : g(v, i) \neq 0\}|$.

3.1.1 Testability in $q(\epsilon)$ queries, for any function q

We first mention, that with the exception of properties that only depend on the degree distribution, adaptive testers are essential for obtaining query complexity that only depends on ϵ (cf. [53]).[9] Still, as observed in [38], at the cost of an exponentially blow-up in the query complexity, we may assume that the tester's adaptivity is confined to performing (full, BFS-like) searches of a predetermined depth from several randomly selected vertices. However, the best testing results are typically obtained by testers that either perform more adaptive searchers or perform DFS-like rather than BFS-like searchers. A few examples follow, where all testers are efficient (i.e., their running time is polynomial in their query complexity).

Testing connectivity. Graph connectivity can be tested in $\widetilde{O}(1/\epsilon)$ queries [33]. Essentially, the tester starts a search (e.g., a BFS) from a few randomly selected vertices, but each such search is terminated after a predetermined number of vertices is encountered (rather than after visiting all vertices that are at a predetermined distance from the start vertex). This tester rejects if and only if it detects a small connected component, and thus it has one-sided error. The result essentially extended to k-edge connectivity, for any $k \geq 2$, but the query complexity is $\widetilde{O}(k^3/\epsilon^c)$, where $c = \min(k-1, 3)$ (cf. [33]).

Testing cycle-freeness. Cycle-freeness can be tested in $\widetilde{O}(\epsilon^{-3})$ queries, by a tester having two-sided error [33]. Essentially, the tester compares the number of edges to the number of connected components, while fully exploring any small connected components that it happens to visit. The two-sided error is unavoidable by any tester that has query complexity $o(\sqrt{N})$ (cf. [33, Prop. 4.3]). Viewing cycle-free graphs as graphs that have no K_3-minor, leads us to the following general result of Benjamini, Schramm, and Shapira [18], which refers to graph minors (to be briefly recalled next).

The graph H is a minor of the graph G, if H can be obtained from G by a sequence of edge removal, vertex removal, and edge contraction operations. We say that G is H-minor free if H is not a minor of G. Thus, a graph is cycle-free if and only if it is K_3-minor free, where K_k denotes the k-vertex clique. (The notion of minor freeness extends to sets of graphs; that is, for a set of graphs $\mathcal{H}$, the graph G is $\mathcal{H}$-minor free if no element of $\mathcal{H}$ is a minor of G.) Lastly, a graph property is minor-closed if it is closed under removal of edges, removal of vertices, and edge contraction. Note that, for every finite sets of graphs $\mathcal{H}$, the property of being $\mathcal{H}$-minor free (e.g., Planarity) is minor-closed.

Theorem 3.2 ([42], improving over [18]):[10] *Any minor-closed property can be tested in query complexity* $\exp(\text{poly}(1/\epsilon))$.

We mention that this tester has two-sided error, which is unavoidable for any tester of query complexity $o(\sqrt{N})$, except for the case that the forbidden minors are all cycle-free.

3.1.2 Testability in $\widetilde{O}(N^{1/2}) \cdot \text{poly}(1/\epsilon)$ queries

The query complexity of testing two natural properties is $\widetilde{\Theta}(N^{1/2}) \cdot \text{poly}(1/\epsilon)$, and in both cases the time complexity has the same form. The properties are `Bipartiteness` and `Expansion`. In

[9] Actually, the result extends to query complexity of the form $o(\sqrt{N} \cdot q(\epsilon))$, for any function q. In contrast, note that triangle-freeness can be tested by $O(\sqrt{N/\epsilon})$ non-adaptive queries; see Appendix A.2.

[10] The query complexity obtained in [18] is triple-exponential in $1/\epsilon$.

both cases, the algorithm is based on taking many (i.e., $\widetilde{O}(N^{1/2}) \cdot \mathrm{poly}(1/\epsilon)$) *random walks* from a few randomly selected vertices, where each walk has length $\mathrm{poly}(\epsilon^{-1} \log N)$.

The foregoing algorithmic approach originates in [34], where it was applied to testing `Bipartiteness`; for further details see §3.2.2. This approach was also suggested for testing `Expansion` [35], but the analysis was successfully completed only in [44, 49]. We mention that the `Bipartite` tester has one-sided error, and whenever it rejects it may also output a short proof that the graph is not bipartite (i.e., an odd cycle of length $\mathrm{poly}(\epsilon^{-1} \log N)$).

The $\Omega(N^{1/2})$ lower bound on the query complexity of testing each of the aforementioned properties was proved in [33]; for details see §3.2.1. We note that the lower bound for testing Bipartiteness stands in sharp contrast to the situation in the dense graph model, where Bipartite testing is possible in $\mathrm{poly}(1/\epsilon)$-time.

An application to the study of the dense graph model. We mention that the Bipartiteness tester of the bounded-degree model was used in order to derive an alternative Bipartite tester for the dense graph model [40]. In the case that almost all vertices in the N-vertex graph have degree $O(\epsilon^{0.99}N)$, this tester improves over the ones presented in [31, 4]. Essentially, this dense-graph model tester invokes the bounded-degree model tester on the subgraph induced by a sample S of $\widetilde{O}(1/\epsilon)$ random vertices (and emulates neighbor queries regarding a vertex $v \in S$ by making adjacency queries of the form (v, w) for every $w \in S$).

3.1.3 Reflections

The fact that the bounded-degree model is closer (than the dense graph model) to standard algorithmic research offers greater interaction at the technical level. Indeed, techniques such as local search and random walks are quite basic in both domains, and the relationship becomes even tighter when we shall move to the general graph model (in Section 4). At the current point, we mention that the idea underlying the cycle-freeness tester (outlined in §3.1.1) was employed to the design of an algorithm for approximating the minimum spanning tree weight in sub-linear time [23].

We also mention that the idea underlying the expansion tester has become quite pivotal in the contents of testing distributions, which emerged with [13].

3.2 A Benchmark: Testing Bipartiteness

Both the following lower and upper bounds reflect the fact that being far from `Bipartiteness` does not require having constant size cycles of odd length. We comment that a simplified version of the upper bound implies that odd cycles of logarithmic length must exist (cf. [34, Prop. 1]).

3.2.1 A lower bound

In contrast to Theorem 2.4, under the incidence function representation, there exists no `Bipartite` tester of complexity that is independent of the graph size.

Theorem 3.3 [33]: *Testing* `Bipartiteness` (with constant ϵ and d) *requires* $\Omega(\sqrt{N})$ *queries* (in the incidence function model).

Proof Idea: For any (even) N, we consider the following two families of graphs:

1. The first family, denoted $\mathcal{G}_1^N$, consists of all degree-3 graphs that are composed of the union of a Hamiltonian cycle and a perfect matching. That is, there are N edges connecting the vertices in a cycle, and the other $N/2$ edges are a perfect matching.

2. The second family, denoted $\mathcal{G}_2^N$, is the same as the first *except* that the perfect matchings allowed are restricted as follows: the distance on the cycle between every two vertices that are connected by a perfect matching edge must be odd.

Clearly, all graphs in $\mathcal{G}_2^N$ are bipartite. It can be shown that almost all graphs in $\mathcal{G}_1^N$ are far from being bipartite. On the other hand, one can prove that a testing algorithm that performs $o(\sqrt{N})$ queries cannot distinguish between a graph chosen randomly from $\mathcal{G}_2^N$ (which is always bipartite) and a graph chosen randomly from $\mathcal{G}_1^N$ (which with high probability is far from bipartite). Loosely speaking, this is the case since in both cases the algorithm is unlikely to encounter a cycle (among the vertices that it has inspected). ■

3.2.2 An algorithm

The lower bound of Theorem 3.3 is essentially tight. Furthermore, the following natural algorithm constitutes a `Bipartite` tester of running time $\text{poly}((\log N)/\epsilon) \cdot \sqrt{N}$.

Algorithm 3.4 (Bipartite Tester in the Bounded-Degree Model [34]): *On input N, d, ϵ and oracle access to an incidence function for an N-vertex graph,* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, *of degree bound d, repeat* $T \stackrel{\text{def}}{=} \Theta(\frac{1}{\epsilon})$ *times:*

1. *Uniformly select s in* V.
2. (Try to find an odd cycle through vertex s):
   (a) *Perform* $K \stackrel{\text{def}}{=} \text{poly}((\log N)/\epsilon) \cdot \sqrt{N}$ *random walks starting from s, each of length* $L \stackrel{\text{def}}{=} \text{poly}((\log N)/\epsilon)$.
   (b) *Let* R_0 (respectively, R_1) *denote the vertices set reached from s in an even* (respectively, odd) *number of steps in any of these walks.*
   (c) *If* $R_0 \cap R_1$ *is not empty then* reject.

If the algorithm did not reject in any of the foregoing T iterations, then it accepts.

Theorem 3.5 [34]: *Algorithm 3.4 is a* `Bipartite` *Tester* (in the incidence function model). *Furthermore, the algorithm always accepts a bipartite graph, and in case of rejection it provides a witness of length* $\text{poly}((\log N)/\epsilon)$ (that the graph is not bipartite).

Motivation – the special case of rapid mixing graphs. The proof of Theorem 3.5 is quite involved. As a motivation, we consider the special case where the graph has a "rapid mixing" feature. It is convenient to modify the random walks so that at each step each neighbor is selected with probability $1/2d$, and otherwise (with probability at least $1/2$) the walk remains in the present vertex. Furthermore, we will consider a single execution of Step (2) starting from an arbitrary vertex, s, which is fixed in the rest of the discussion. The rapid mixing feature we assume is that, for every vertex v, a (modified) random walk of length L starting at s reaches v with probability approximately $1/N$ (say, up-to a factor of 2). Note that if the graph is an expander then this is certainly the case (since $L = \omega(\log N)$).

The key quantities in the analysis are the following probabilities, referring to the *parity* of the length of a path obtained from the random walk by omitting the self-loops (transitions that remain at current vertex). Let $p^0(v)$ (respectively, $p^1(v)$) denote the probability that a (modified) random walk of length L, starting at s, reaches v while making an even (respectively, odd) number of real (i.e., non-self-loop) steps. By the rapid mixing assumption (for every $v \in \mathrm{V}$), it holds that

$$\frac{1}{2N} \;<\; p^0(v) + p^1(v) \;<\; \frac{2}{N}\,. \tag{2}$$

We consider two cases regarding the sum $\sum_{v\in\mathrm{V}} p^0(v)p^1(v)$: If the sum is (relatively) "small", we show that V can be 2-partitioned so that there are relatively few edges between vertices that are placed in the same side, which implies that G is close to being bipartite. Otherwise (i.e., when the sum is not "small"), we show that with significant probability, when Step (2) is started at vertex s it is completed by rejecting G. These two cases are analyzed in the following two (corresponding) claims.

Claim 3.5.1 *Suppose* $\sum_{v\in\mathrm{V}} p^0(v)p^1(v) \;\le\; \epsilon/50N$. *Let* $\mathrm{V}_1 \stackrel{\rm def}{=} \{v \in \mathrm{V} : p^0(v) < p^1(v)\}$ *and* $\mathrm{V}_2 = \mathrm{V} \setminus \mathrm{V}_1$. *Then, the number of edges with both end-points in the same* V_σ *is bounded above by* ϵdN.

Proof Sketch: Consider an edge (u, v) where, without loss of generality, both u and v are in V_1. Then, both $p^1(v)$ and $p^1(u)$ are greater than $\frac{1}{2}\cdot\frac{1}{2N}$. However, one can show that $p^0(v) > \frac{1}{3d}\cdot p^1(u)$: Observe that an $(L-1)$-step walk of path-parity 1 ending at u is almost as likely as an L-step walk of path-parity 1 ending at u, and that once an $(L-1)$-step walk reaches u, with probability exactly $1/2d$, it continues to v in the next step. Thus, the edge (u, v) contributes at least $\frac{(1/4N)^2}{3d}$ to the sum $\sum_{w\in\mathrm{V}} p^0(w)p^1(w)$. It follows that we can have at most $(\epsilon/50N)/(1/48dN^2)$ such edges, and the claim follows. □

Claim 3.5.2 *Suppose* $\sum_{v\in\mathrm{V}} p^0(v)p^1(v) \;\ge\; \epsilon/50N$, *and that Step (2) is started with vertex* s. *Then, with probability at least* $2/3$, *the set* $R_0 \cap R_1$ *is not empty* (and rejection follows).

Proof Sketch: Consider the probability space defined by an execution of Step (2) with start vertex s. We define random variables $\zeta_{i,j}$ representing the event that the vertex encountered in the $L^{\rm th}$ step of the $i^{\rm th}$ walk equals the vertex encountered in the $L^{\rm th}$ step of the $j^{\rm th}$ walk, and that the $i^{\rm th}$ walk corresponds to an even-path whereas the $j^{\rm th}$ to an odd-path. Then

$$\begin{aligned}
\mathbf{E}[|R_0 \cap R_1|] \;&>\; \sum_{i\neq j} \mathbf{E}[\zeta_{i,j}] \\
&=\; K(K-1)\cdot \sum_{v\in\mathrm{V}} p^0(v)p^1(v) \\
&>\; \frac{500N}{\epsilon}\cdot \sum_{v\in\mathrm{V}} p^0(v)p^1(v) \\
&\ge\; 10
\end{aligned}$$

where the second inequality is due to the setting of K, and the third to the claim's hypothesis. Intuitively, with high probability, it should hold that $|R_0 \cap R_1| > 0$. This is indeed the case, but proving it is less straightforward than it seems; the problem being that the $\zeta_{i,j}$'s are not pairwise independent. Yet, since the sum of the covariances of the dependent $\zeta_{i,j}$'s is quite small, Chebyshev's Inequality is still very useful (cf. [11, Sec. 4.3]). Specifically, letting $\mu \stackrel{\rm def}{=} \sum_{v\in\mathrm{V}} p^0(v)p^1(v)$ $(= \mathbf{E}[\zeta_{i,j}])$, and $\overline{\zeta}_{i,j} \stackrel{\rm def}{=} \zeta_{i,j} - \mu$, we get:

$$\mathbf{Pr}\left[\sum_{i\neq j}\zeta_{i,j} = 0\right] \;<\; \frac{\mathbf{Var}\left[\sum_{i\neq j}\zeta_{i,j}\right]}{(K^2\mu)^2}$$

$$
\begin{aligned}
&= \frac{1}{K^4\mu^2} \cdot \left(\sum_{i,j} \mathbf{E}\left[\overline{\zeta}_{i,j}^2\right] + 2\sum_{i,j,k} \mathbf{E}\left[\overline{\zeta}_{i,j}\overline{\zeta}_{i,k}\right] \right) \\
&< \frac{1}{K^2\mu} + \frac{2}{K\mu^2} \cdot \mathbf{E}[\zeta_{1,2}\zeta_{1,3}]
\end{aligned}
$$

For the second term, we observe that $\mathbf{Pr}[\zeta_{1,2}=\zeta_{1,3}=1]$ is upper bounded by $\mathbf{Pr}[\zeta_{1,2}=1] = \mu$ times the probability that the L^{th} vertex of the first walk appears as the L^{th} vertex of the third path. Using the rapid mixing hypothesis, we upper bound the latter probability by $2/N$, and obtain

$$
\begin{aligned}
\mathbf{Pr}[|R_0 \cap R_1| = 0] &< \frac{1}{K^2\mu} + \frac{2}{K\mu^2} \cdot \mu \cdot \frac{2}{N} \\
&< \frac{1}{3}
\end{aligned}
$$

where the last inequality uses $\mu \geq \epsilon/50N$ and $K^2 \geq 6 \cdot 50N/\epsilon$ (along with $\epsilon > 5000/N$). The claim follows. $\square$

Beyond rapid mixing graphs. The proof in [34] refers to a more general sum of products; that is, $\sum_{u\in \mathrm{U}} p_{\mathrm{odd}}(u)p_{\mathrm{even}}(u)$, where $\mathrm{U} \subseteq \mathrm{V}$ is an appropriate set of vertices, and $p_{\mathrm{odd}}(v)$ (respectively, $p_{\mathrm{even}}(v)$) is essentially the probability that an L-step random walk (starting at s) passes through v after more than $L/2$ steps and the corresponding path to v has odd (respectively, even) parity. Much of the analysis in [34] goes into selecting the appropriate U (and an appropriate starting vertex s), and pasting together many such U's to cover all of V. Loosely speaking, U and s are selected so that there are few edges from U and the rest of the graph, and $p_{\mathrm{odd}}(u)+p_{\mathrm{even}}(u) \approx 1/\sqrt{|\mathrm{V}|\cdot|\mathrm{U}|}$, for every $u \in \mathrm{U}$. The selection is based on the "combinatorial treatment of expansion" of Mihail [48]. Specifically, we use the contrapositive of the standard analysis, which asserts that rapid mixing occurs when all cuts are relatively large, to assert the existence of small cuts which partition the graph so that vertices reached with relatively high probability (in a short random walk) are on one side and the rest of the graph on the other. The first set corresponds to the aforementioned U and the cut is relatively small with respect to U. A start vertex s for which the corresponding sum is big is shown to cause Step (2) to reject (when started with this s), whereas a small corresponding sum enables to 2-partition U while having few violating edges among the vertices in each part of U.

The actual argument of [34] proceeds in iterations. In each iteration a vertex s for which Step (2) accepts with high probability is fixed, and an appropriate set of remaining vertices, U, is found. The set U is then 2-partitioned so that there are few violating edges inside U. Since we want to paste all these partitions together, U may not contain vertices treated in previous iterations. This complicates the analysis, since it must refer to the part of G, denoted H, not treated in previous iterations. We consider walks over an (imaginary) Markov Chain representing the H-part of the walks performed by the algorithm on G. Statements about rapid mixing are made with respect to the Markov Chain, and linked to what happens in random walks performed on G. In particular, a subset U of H is determined so that the vertices in U are reached with probability $\approx 1/\sqrt{|\mathrm{V}|\cdot|\mathrm{U}|}$ (in the chain) and the cut between U and the rest of H is small. Linking the sum of products defined for the chain with the actual walks performed by the algorithm, we infer that U may be partitioned with few violating edges inside it. Edges to previously treated parts of the graphs are charged to these parts, and edges to the rest of $\mathrm{H} \setminus \mathrm{U}$ are accounted for by using the fact that this cut is small (relative to the size of U).

4 The General Graph Model

In contrast to the foregoing two models in which the oracle queries and the distances between graphs are linked to the representation of graphs as functions, in the following model the representation is blurred and the query types and distance measure are decoupled. This decoupling makes the current model closer in spirit to standard studies in graph algorithms.

Giving up on the representation as a yardstick for the relative distance between graphs, leaves us with no absolute point of reference. Instead, we just define the relative distance between graphs in relation to the actual number of edges in these graphs; specifically, the relative distance between the graphs $G = ([N], E)$ and $G' = ([N], E)$ may be defined as $\frac{|E \triangle E'|}{\max(|E|,|E'|)}$ (or, alternatively, as $\frac{|E \triangle E'|}{(|E|+|E'|)/2}$).[11]

Turning to the question of query types, we again need to make a choice, which is now free from representation considerations. The most natural choice is to allow both *adjacency queries* and *incidence queries* (i.e., the two types of queries that were each allowed in one of the previous queries).[12] However, other choices has been considered too (cf. [17]). We note that, typically, adjacency queries become more useful as the graph becomes more dense, whereas incidence queries (a.k.a neighbor queries) become more useful as the graph becomes more sparse (cf. [17]).

Definition 4.1 (testing graph properties in the general model): *A* tester *for a graph property* Π *is a probabilistic oracle machine that, on input parameters* N *and* ϵ *and access to a function answering adjacency queries and incidence queries regarding an* N*-vertex graph* $G = ([N], E)$*, outputs a binary verdict that satisfies the following two conditions.*

1. *If* $G \in \Pi$ *then the tester accepts with probability at least* $2/3$.
2. *If* G *is* ϵ*-far from* Π *then the tester accepts with probability at most* $1/3$*, where* G *is* ϵ-far *from* Π *if for every* N*-vertex graph* $G' = ([N], E') \in \Pi$ *it holds that the symmetric difference between* E *and* E' *has cardinality that is greater than* $\epsilon \cdot \max(|E|, |E'|)$.

One-sided testers and non-adaptive testers are defined as in Definition 2.1.

The query complexity of a tester is defined as in Section 2; ditto for its efficiency.

The study of property testing in the general graph model was initiated by Parnas and Ron [51], who only considered incidence queries, and extended by Kaufman, Krivelevich, and Ron [45], who considered both types of queries. Needless to say, the aim of these works was to allow the consideration of arbitrary graphs and so strengthen the relation between property testing and standard algorithmic studies. However, forsaking the paradigm of representing graphs as functions means that the connection to the rest of property testing is a bit weakened (or at least becomes more cumbersome). Still, we believe that the trade-off is worthwhile.

4.1 A Taste of the Known Results

It is natural to attempt to extend testers designed for the bounded-degree model to the general graph model. Such extensions face two potential difficulties, which refer to two ways in which the general graph model extends the bounded-degree model:

[11] Needless to say, these two definitions may not yield the same result, but they are related by a factor of at most 2.

[12] Recall that the incidence query (u, i) is answered with 0 if u has less than i neighbors. Thus, the incidence queries allow to emulate degree queries at logarithmic cost.

1. Firstly, the maximum degree of vertices in the graph may no longer be constant, and the question is how does the performance of the tester depends on the degree bound, d. Formally, one should think of the degree bound d as a variable, and analyze the tester accordingly.

   Note that when d increases, relative distances decrease and so testing may become easier. On the other hand, we can no longer scan all neighbors of a given vertex at constant cost.

2. Treating the maximum degree as a variable, raises the question of what happens when there is a significant discrepancy among the degrees of the various vertices. Such a situation can break the balance between the aforementioned positive and negative effects of increasing the maximum degree. Specifically, the algorithmic operations may becomes more costly when the maximum degree increases, but when using the distance measure of Definition 4.1 the distances no longer vary with the maximum degree (i.e., d) but rather vary with the average degree. Thus, we may be in trouble if the maximum degree is significantly larger than the average degree.

The effect of the foregoing issues is tester-dependent. For example, the operation of the Connectivity tester (outlined in §3.1.1) is not affected by the possible discrepancies in the vertex degrees, and so this tester (as is) applies also to the general graph model (cf. [51]). In contrast, the `Bipartiteness` tester presented in Algorithm 3.4 should be modified to the current setting. Details follow.

4.2 A Benchmark: Testing Bipartiteness

Firstly, it was shown in [45] that the algorithm's performance does not deteriorate when d increases. Next, an algorithm for the general graph model was obtained by emulating Algorithm 3.4 on an imaginary graph that is obtained by replacing vertices of high degree by adequate gadgets. Specifically, a vertex having degree that is t times larger than the average degree is replaced by a t-by-t bipartite expander graph, while connecting the original neighbors to vertices on one of the sides of the expander (such that no vertex has degree greater than twice the average degree). This replacement preserves the distance to `Bipartiteness` (up to a constant factor). We warn that implementing the emulation (of Algorithm 3.4 on this imaginary graph) is not straightforward. In particular, it seems to require a procedure for sampling edges in the actual graph such that almost all edges are sampled with probability that is approximately (up to a constant factor) the uniform one.[13] For details, see [45].

As evident from the above description, the extension of a tester from the bounded-degree model to the general graph model may require ideas that are specific to the property at hand. For example, the gadgets used above should preserve `Bipartiteness` (as well as distance to `Bipartiteness`).

Another issue that arises is that one may hope to perform better when the degree bound d (whether maximum or average) is large. Indeed, we know that in case of `Bipartiteness`, dense graphs can be tested with much fewer queries than sparse graphs (recall Algorithm 2.3). Thus, an optimal tester for the general graph model should be able to match the result of the dense graph model whenever the actual graph happens to be dense. Such a result is indeed provided by [45], who show a `Bipartiteness` tester (for the general graph model) that is optimal for all possible edge densities.

Theorem 4.2 (Testing Bipartiteness in the General Graph Model [45]): *Ignoring factors that are polynomial in* $\epsilon^{-1}\log N$, *the query* (and time) *complexity of testing* `Bipartiteness` *is* $\min(\sqrt{N}, N^2/M)$, *where* M *denotes the number of edges in the input graph.*

[13] A more accurate sampling procedure is implicit in the subsequent work of [36].

Note that dealing with $M \gg N^{3/2}$ requires some deviation from the aforementioned emulation (of Algorithm 3.4). Indeed, in such a case the tester of [45] behaves quite differently. Specifically, it takes $K = \sqrt{N^2/M}$ random walks (rather than N^2/M random walks), from each random start vertex, and checks for collisions among the endpoints these K walks by using $\binom{K}{2}$ adjacency queries. We mention that the use of adjacency queries is necessary for an $o(\sqrt{N})$ query tester of `Bipartiteness`.

An opposite behavior. In contrast to the case of testing `Bipartiteness`, where the complexity improves with the edge density, in the case of testing triangle-freeness we see the opposite behavior [5].[14] Furthermore, in contrast to testing `Bipartiteness`, there is a gap between the complexity of testing triangle-freeness in the bounded-degree model and the corresponding complexity in the general graph model even when the graph is sparse (i.e., $M = O(N)$). For example, in the general graph model, the complexity is $\Omega(N^{1/3})$ as long as $M = N^{2-o(1)}$ [5].

4.3 Reflections

The bulk of algorithmic research regarding graphs refers to general graphs. Of special interest are graphs that are neither very dense nor have a bounded degree. In contrast, research in testing properties of graphs started (in [31]) with the study of dense graphs, proceeded to the study of bounded-degree graphs (in [33]), and reached general graphs only in [51, 45]. This evolution has historical reasons to be reviewed first.

Testing graph properties was initially conceived (in [31]) as a special case of the framework of testing properties of functions. Thus, graphs had to be represented by functions, and two standard representations of graphs (indeed, the two reviewed in Sections 2 and 3) seemed most fitting in this context. We stress that both models were formulated in a way that identifies the graphs with a specific functional representation, which in turn defines the type of queries allowed to the tester as well as the notion of fractional distance (which underlies the performance guarantee).

The identification of graphs with any specific functional representation was abandoned by Parnas and Ron [51] who developed a more general model by decoupling the type of queries allowed to the tester from the distance measure: Whatever is the mechanism of accessing the graph, the distance between graphs is defined as the number of edges in their symmetric difference (rather than the number of different entries with respect to some specific functional representation). Furthermore, the relative distance may be defined as the size of the symmetric difference divided by the actual (total) number of edges in both graphs (rather than divided by some (possibly non-tight) upper-bound on the latter quantity). Also, as advocated by Kaufman *et al.* [45], it is reasonable to allow the tester to perform both adjacency and neighbor queries (and indeed each type of query may be useful in a different range of edge densities). Needless to say, this model seems adequate for the study of testing properties of arbitrary graphs, and it strictly generalizes the positive aspects of the two prior models (i.e., the models based on the adjacency matrix and bounded-degree incidence list representations).

We wish to advocate further study of the latter model. We believe that this model, which allows for a meaningful treatment of property testing of general graphs, is the one that is most relevant to computer science applications. Furthermore, it seems that designing testers in this model requires the development of algorithmic techniques that may be applicable also in other areas of algorithmic research. As an example, we mention that techniques in [45] underly the

[14] This is to be expected in light of the fact that testing triangle-freeness has complexity $O(d/\epsilon)$ in the bounded-degree model [33], whereas in the dense graph model testing triangle-freeness requitres more than $\mathrm{poly}(1/\epsilon)$ queries [1].

average degree approximation of [36]. (Likewise techniques of [33] underly the minimum spanning tree weight approximation of [23]; indeed, as noted next, the bounded-degree incidence list model is also more algorithmic oriented than the adjacency matrix model.)

Let us focus on the algorithmic contents of property testing of graphs. Recall that, when ignoring a quadratic blow-up in the query complexity, property testing in the adjacency matrix representation reduces to sheer combinatorics (as reflected in the notion of canonical testers, see Theorem 2.2). Indeed, as shown in [37], a finer look (which does not allow for ignoring quadratic blow-ups in complexity) reveals the role of algorithmic design also in this model. But still property testing in the incidence list representation seems to require more sophisticated algorithms. Testers in the general graph models seem to require even more algorithmic ideas (cf. [45]).

To summarize, we advocate further study of the model of [51, 45] for two reasons. The first reason is that we believe in the greater relevance of this model to computer science applications. The second reason is that we believe in the greater potential of this model to have cross fertilization with other branches of algorithmic research. Nevertheless, this advocation is not meant to undermine the study of the dense graph and bounded-degree models. The latter have their own merits and also offer a host of interesting open problems, which are potentially relevant to computer science at large.

5 Additional Issues

In this section we discuss three issues that are relevant to each of the three models discussed in the prior corresponding three sections.

5.1 Directed Graphs

So far our discussion was confined to undirected graphs. Nevertheless, the three models extend naturally to the case of directed graphs. Actually, when considering incidence queries, two different sub-models emerge (cf. [16]): In the first model the tester may only query for edges in the forward direction (resp., backward direction), whereas in the second model both forward and backward directions are allowed. That is, in the second model, the directed graph $G = ([N], E)$ is represented by two functions, $g_{\rm out}$ and $g_{\rm in}$, such that $g_{\rm out}(u, i) = v$ (resp., $g_{\rm in}(u, i) = v$) if the $i^{\rm th}$ out-going edge of u leads to v (resp., the $i^{\rm th}$ in-coming edge of u arrives from v).

The gap between these two query models was demonstrated by Bender and Ron, who initiated the study of testing properties of directed graphs [16]. In particular, they showed that while strong connectivity in bounded-degree directed graphs can be tested by $\widetilde{O}(1/\epsilon)$ forward and backward queries [16, Sec. 5.1], when only forward (resp., backward) queries are allowed no tester can work with $o(\sqrt{N})$ queries (even when allowing two-sided error [16, Sec. 5.2]).[15]

Another task studied in [16] is testing whether a given directed graph is acyclic (i.e., has no directed cycles). They presented an `Acyclicity` tester of poly$(1/\epsilon)$ complexity in the adjacency predicate model, and showed that in the incidence list model no `Acyclicity` tester can work with $o(N^{1/3})$ queries (even when both forward and backward queries are allowed). The question of whether `Acyclicity` can be tested with $o(N)$ queries (in the bounded-degree digraph model) remains open. In general, it seems that the study of this model deserves more attention than it has received so far. (We mention that testing directed graphs in the dense digraph model was further studied in [6, 50].)

[15]The lower bound can be strengthened to $\Omega(N)$ when considering only one-sided error testers. In the case of two-sided error, some improvements are possible; see Appendix A.3.

5.2 Tolerant Testing and Distance Approximation

Recall that property testing calls for distinguishing objects having a predetermined property from object that are far from any objects that has this property (i.e., are far from the property). A more "tolerant" notion requires distinguishing objects that are close to having the property from objects that are far from this property. Such a distinguisher is called a tolerant tester, and is a special case of a distance approximator that given any object is required to approximate its distance to the property. The study of these related notions was initiated by Parnas, Ron, and Rubinfeld [52].

Definition 5.1 (sketch for the generic case): *Let Π be a set of functions over a finite set Ω. A* distance approximator *for Π is a probabilistic oracle machine T that on input an approximation parameter ϵ and access to any function f outputs with probability at least $2/3$ a value that approximates the relative distance of f to Π up to an additive term of ϵ; that is, $\mathbf{Pr}[|T^f - \delta_\Pi(f)| \leq \epsilon] \geq 2/3$, where $\delta_\Pi(f) \stackrel{\text{def}}{=} \min_{g\in\Pi}\{\delta(f,g)\}$ and $\delta(f,g) \stackrel{\text{def}}{=} \mathbf{Pr}_{x\in\Omega}[f(x) \neq g(x)]$.*

A simple observation is that any tester that makes uniformly distributed queries offers some level of tolerance. Specifically, if a tester makes $q(\epsilon)$ queries and each query is uniformly distributed, then this tester distinguishes between objects that are ϵ-far from the property and objects that are $(\epsilon/10q(\epsilon))$-close to the property. Needless to say, the challenge is to provide stronger relations between property testing and distance approximators. Such a result was provided by Fischer and Newman [28]: They showed that, *in the dense graph model, testability in a number of queries that only depends on ϵ implies distance approximator in a number of queries that only depends on ϵ*. In the the bounded-degree model, many of the known testers were extended to yield distance approximators (cf. [47]).

5.3 Proximity Oblivious Testing

Note that in order to satisfy the property testing requirement, any tester (of a reasonable property) must obtain the proximity parameter as auxiliary input and determine its actions accordingly. The question, addressed here, is what does the tester do with this parameter (or how does the parameter affect the actions of the tester). A very minimal effect is exhibited by testers that, based on the value of the proximity parameter, determine the number of times that a basic test is invoked, where the basic test is oblivious of the proximity parameter. For example, the celebrated linearity tester of [19] repeats a basic test that consists of selecting two random points, x and y, and probing the value of the function at the points x, y, and $x+y$. This basic test is repeated for a number of times that is inversely proportional to the proximity parameter.

Our focus here is on such basic tests (i.e., basic tests that are oblivious of the proximity parameter), called proximity oblivious testers. Although proximity oblivious testers were implicit in prior works (see, e.g., [19, 2, 3]), their general study was initiated by Goldreich and Ron [38].

Definition 5.2 (sketch for the generic case): *Let Π be a set of functions over a finite set Ω. A* proximity-oblivious tester *for Π is a probabilistic oracle machine T that, when given oracle access to any function f over Ω, satisfies the following two conditions:*

1. *The machine T accepts each function in Π with probability 1.*
2. *For some* (monotone) *function $\rho : (0,1] \to (0,1]$, each function $f \notin \Pi$ is rejected by T with probability at least $\rho(\delta_\Pi(f))$, where $\delta_\Pi(f)$ is as in Definition 5.1.*

The function ρ is called the detection probability of the tester *T.*

Indeed, we require that $\rho(\epsilon) > 0$ for every $\epsilon > 0$, whereas extending Item 2 to $f \in \Pi$ (while avoiding contradiction with Item 1) mandates extending ρ so that $\rho(0) = 0$. The requirement that ρ is monotone (i.e., monotonically increasing) does not rule out cases where the tight lower-bound is non-monotone (e.g., [14]), because ρ is not required to be tight.

Indeed, using a proximity-oblivious tester T, we can obtain a standard (one-sided error) tester (of error probability at most $1/3$). Specifically, given the proximity parameter ϵ, the standard tester invokes T for $\Theta(1/\rho(\epsilon))$ times, and accepts if and only if all these invocations accept. Two natural questions regarding proximity oblivious testers are:

1. Which properties have proximity oblivious tests (of small query complexity)?
2. How does the detection probability of such tests grow as a function of the distance of the object from the property, and how does this relate to the query complexity of the best (standard) tester for the corresponding property.

Goldreich and Ron [38] provide a mix of positive and negative results regarding the foregoing questions. In particular, they provide a characterizations of the graph properties that have constant-query proximity-oblivious testers in the two main models discussed in this article (i.e., the dense graphs model and the bounded-degree graph model). It follows that constant-query proximity-oblivious testers do not exist for many easily testable properties (e.g., `Bipartiteness` in the dense graph model). Also, even when proximity-oblivious testers exist, repeating them does not necessarily yield the best standard testers for the corresponding property (e.g., `Clique Collection` in the dense graph model).

Acknowledgments

We are grateful to Tali Kaufman, Michael Krivelevich, and Dana Ron for useful discussions.

References

[1] N. Alon. Testing subgraphs of large graphs. *Random Structures and Algorithms*, Vol. 21, pages 359–370, 2002.

[2] N. Alon, E. Fischer, M. Krivelevich and M. Szegedy. Efficient Testing of Large Graphs. *Combinatorica*, Vol. 20, pages 451–476, 2000.

[3] N. Alon, E. Fischer, I. Newman, and A. Shapira. A Combinatorial Characterization of the Testable Graph Properties: It's All About Regularity. In *38th STOC*, pages 251–260, 2006.

[4] N. Alon and M. Krivelevich. Testing k-Colorability. *SIAM Journal on Disc. Math.*, Vol. 15 (2), pages 211-227, 2002.

[5] N. Alon, T. Kaufman, M. Krivelevich, and D. Ron. Testing triangle freeness in general graphs. In *17th SODA*, pages 279–288, 2006.

[6] N. Alon and A. Shapira. Testing subgraphs in directed graphs. *JCSS*, Vol. 69, pages 354–482, 2004.

[7] N. Alon and A. Shapira. Every Monotone Graph Property is Testable. In *37th STOC*, pages 128–137, 2005.

[8] N. Alon and A. Shapira. A Characterization of the (natural) Graph Properties Testable with One-Sided. In *46th FOCS*, pages 429–438, 2005.

[9] N. Alon and A. Shapira. A Characterization of Easily Testable Induced Subgraphs. *Combinatorics Probability and Computing*, Vol. 15, pages 791–805, 2006.

[10] N. Alon and A. Shapira. A Separation Theorem in Property Testing. *Combinatorica*, Vol. 28 (3), pages 261–281, 2008.

[11] N. Alon and J.H. Spencer, *The Probabilistic Method*, John Wiley & Sons, Inc., 1992.

[12] S. Arora, D. Karger, and M Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *JCSS*, Vol. 58 (1), pages 193–210, 1999.

[13] T. Batu, L. Fortnow, R. Rubinfeld, W.D. Smith and P. White. Testing that Distributions are Close. In *41st FOCS*, pages 259–269, 2000.

[14] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. In *the 36th FOCS*, pages 432–441, 1995.

[15] M. Bellare, O. Goldreich, and M. Sudan. Free Bits, PCPs and Non-approximability – Towards Tight Results. *SIAM Journal on Computing*, Vol. 27, No. 3, pages 804–915, June 1998.

[16] M. Bender and D. Ron. Testing acyclicity of directed graphs in sublinear time. *Random Structures and Algorithms*, pages 184–205, 2002.

[17] I. Ben-Eliezer, T. Kaufman, M. Krivelevich, and D. Ron. Comparing the strength of query types in property testing: the case of testing k-colorability. In *19th SODA*, 2008.

[18] I. Benjamini, O. Schramm, and A. Shapira. Every Minor-Closed Property of Sparse Graphs is Testable. In *40th STOC*, pages 393–402, 2008.

[19] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *JCSS*, Vol. 47, No. 3, pages 549–595, 1993.

[20] A. Bogdanov, K. Obata, and L. Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *43rd FOCS*, pages 93–102, 2002.

[21] A. Bogdanov and L. Trevisan. Lower Bounds for Testing Bipartiteness in Dense Graphs. In *IEEE Conference on Computational Complexity*, pages 75–81, 2004.

[22] R. Canetti, G. Even and O. Goldreich. Lower Bounds for Sampling Algorithms for Estimating the Average. *IPL*, Vol. 53, pages 17–25, 1995.

[23] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. In *19th ICALP*, pages 190–200, 2001.

[24] W.F. de la Vega. MAX-CUT has a randomized approximation scheme in dense graphs. *Random Structures and Algorithms*, Vol. 8 (3), pages 187–198, 1996.

[25] S. Even. *Graph Algorithms.* Computer Science Press, 1979.

[26] S. Even, A.L. Selman, and Y. Yacobi. The Complexity of Promise Problems with Applications to Public-Key Cryptography. *Inform. and Control*, Vol. 61, pages 159–173, 1984.

[27] E. Fischer and A. Matsliah. Testing graph isomorphism. In *17th SODA*, pages 299–308, 2006.

[28] E. Fischer and I. Newman. Testing versus estimation of graph properties. In *37th STOC*, pages 138–146, 2005.

[29] O. Goldreich. On Promise Problems: In memory of Shimon Even (1935–2004). *ECCC*, TR05-018, January 2005. See also in *Theoretical Computer Science: Essays in Memory of Shimon Even*, Springer, LNCS Festschrift, Vol. 3895, March 2006.

[30] O. Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008.

[31] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, pages 653–750, July 1998. Extended abstract in *37th FOCS*, 1996.

[32] O. Goldreich, M. Krivelevich, I. Newman, and E. Rozenberg. Hierarchy Theorems for Property Testing. *ECCC*, TR08-097, 2008. Extended abstract in the proceedings of *RANDOM'09*.

[33] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, pages 302–343, 2002.

[34] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, Vol. 19 (3), pages 335–373, 1999.

[35] O. Goldreich and D. Ron. On Testing Expansion in Bounded-Degree Graphs. *ECCC*, TR00-020, March 2000.

[36] O. Goldreich and D. Ron. Approximating Average Parameters of Graphs. *Random Structures and Algorithms*, Vol. 32 (3), pages 473–493, 2008.

[37] O. Goldreich and D. Ron. Algorithmic Aspects of Property Testing in the Dense Graphs Model. *ECCC*, TR08-039, 2008.

[38] O. Goldreich and D. Ron. On Proximity Oblivious Testing. *ECCC*, TR08-041, 2008. Also in the proceedings of the *41st STOC*, 2009.

[39] O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. *Random Structures and Algorithms*, Vol. 23 (1), pages 23–57, August 2003.

[40] M. Gonen and D. Ron. On the Benefit of Adaptivity in Property Testing of Dense Graphs. In *Proc. of RANDOM'07*, LNCS Vol. 4627, pages 525–539, 2007. To appear in *Algorithmica* (special issue of RANDOM and APPROX 2007).

[41] Håstad, J. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, Vol. 182, pages 105–142, 1999. (Preliminary Version in *28th STOC*, 1996 and *37th FOCS*, 1997.)

[42] A. Hassidim, J. Kelner, H. Nguyen, and K. Onak. Local Graph Partitions for Approximation and Testing. In *50th FOCS*, pages 22–31, 2009.

[43] D. Hochbaum (ed.). *Approximation Algorithms for NP-Hard Problems*. PWS, 1996.

[44] S. Kale and C. Seshadhri. Testing expansion in bounded degree graphs. In *35th ICALP*, pages 527–538, 2008. (Preliminary version appeared as TR07-076, *ECCC*, 2007.)

[45] T. Kaufman, M. Krivelevich, and D. Ron. Tight Bounds for Testing Bipartiteness in General Graphs. *SIAM Journal on Computing*, Vol. 33 (6), pages 1441–1483, 2004.

[46] L. Lovász and N. Young. Lecture notes on evasiveness of graph properties. Technical Report TR–317–91, Princeton University, Computer Science Department, 1991.

[47] S. Marko and D. Ron. Distance approximation in bounded-degree and general sparse graphs. *Transactions on Algorithms*, 5(2), 2009. Article number 22.

[48] M. Mihail. Conductance and convergence of Markov chains– A combinatorial treatment of expanders. In *30th FOCS*, pages 526–531, 1989.

[49] A. Nachmias and A. Shapira. Testing the expansion of a graph. TR07-118, *ECCC*, 2007.

[50] Y. Orenstein. Testing properties of directed graphs. Master's thesis, School of Electrical Engineering, 2010.

[51] M. Parnas and D. Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, Vol. 20 (2), pages 165–183, 2002.

[52] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant Property Testing and Distance Approximation. *Journal of Computer and System Sciences*, Vol. 72 (6), pages 1012–1042, 2006.

[53] S. Raskhodnikova and A. Smith. A note on adaptivity in testing properties of bounded-degree graphs. *ECCC*, TR06-089, 2006.

[54] D. Ron. Algorithmic and Analysis Techniques in Property Testing. *Foundations and Trends in TCS*, to appear.

[55] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2), pages 252–271, 1996.

[56] E. Szemerédi. Regular partitions of graphs. In *Proceedings, Colloque Inter. CNRS*, pages 399–401, 1978.

Appendix: In Passing – Three Unrelated Observations

The following three observations occurred to us in the process of writing this article.

A.1 Testing Degree Regularity in the Dense Graph Model

We improve the $\widetilde{O}(\epsilon^{-3})$ query upper bound of [31, Prop. 10.2.1.3] to an optimal quadratic bound.

Proposition A.1 *In the dense graph model, degree regularity can be tested in $O(\epsilon^{-2})$ non-adaptive queries.*

Proof: We start by reviewing the $\widetilde{O}(\epsilon^{-3})$-query tester presented in the proof of [31, Prop. 10.2.1.3]. This tester selects $O(1/\epsilon)$ random vertices, and estimates the degree of each of them up to $\pm\epsilon N/100$ using a sample of $s = \widetilde{O}(1/\epsilon^2)$ random vertices (and making the corresponding s queries). This tester accepts if and only if all these estimates are at most $\epsilon N/20$ apart. The analysis is based on the observation that if the tester accepts with high probability, then all but $\epsilon' N$ vertices have degree that is within $\pm\epsilon' N$ units of some value, where $\epsilon' = \epsilon/13$. By omitting and adding at most $\epsilon' N^2$ vertices (i.e., from/to the exceptional vertices), we reach a situation in which all vertices have degrees that at most $D \stackrel{\rm def}{=} 4\epsilon' N$ units apart. At this point, we are done by applying a theorem of Noga Alon (cf. [31, Apdx. D]) that asserts that such a graph is $((3D/N) + o(1))$-close to being regular.

We improve the foregoing upper bound as follows. For a sufficiently large constant c, let $\ell \stackrel{\rm def}{=} \log_2(c/\epsilon)$, and consider an algorithm that, for every $i \in [\ell]$, proceeds as follows:

1. The algorithm selects uniformly $c \cdot 2^i$ vertices, and estimates the degree of each of these vertices up to $\pm 2^{4i/5}\epsilon \cdot N/c$ units by using a sample of $s_i \stackrel{\rm def}{=} c^3 \cdot 2^{-3i/2}\epsilon^{-2}$ random vertices.

   Note that with probability at least

   $$\begin{aligned} 1 - c \cdot 2^i \cdot \exp(-2s_i \cdot (2^{4i/5}\epsilon/c)^2) &= 1 - c \cdot 2^i \cdot \exp(-2c \cdot 2^{i/10}) \\ &> 1 - 2^{-i-c} \end{aligned}$$

   all these estimates are as desired.

2. If two of these estimates are more than $2^{1+(4i/5)}\epsilon \cdot N$ units apart, then the algorithm rejects.

(The algorithm accepts if and only if it does not reject in any of these ℓ iterations.) The query complexity of this algorithm is $\sum_{i\in[\ell]} c2^i \cdot c^3 2^{-3i/2}\epsilon^{-2} = O(\epsilon^{-2})$, and it accepts each regular graph with high probability (i.e., whenever all the foregoing degree estimates are adequate).

On the other hand, if a graph is accepted with high probability, then, for every $i \in [\ell]$, it holds that all but at most a 2^{-i} fraction of the vertices have degree that is within $2^{1+4i/5}\epsilon \cdot N/c$ of the average degree, denoted ρ. For each value of $i \in [\ell]$, let us denote the set of deviating vertices by B_i; that is, each vertex in $[N] \setminus B_i$ has degree $(\rho \pm 2^{1+4i/5}\epsilon/c) \cdot N$. Thus (dealing separately with each $B_i \setminus B_{i+1}$ as well as with B_ℓ and $[N] \setminus B_1$), we may omit at most $40\epsilon N^2/c$ edges from the graph, and obtain a graph in which every vertex has degree at most $(\rho + 2\epsilon/c)N$. Next, by adding at most $42\epsilon N^2/c$ edges to the graph, we can obtain a graph in which every vertex has degree at least $(\rho - 2\epsilon/c)N$, and if we add these edges uniformly (among the vertices) then each vertex in the resulting graph has degree $(\rho \pm 44\epsilon/c)N$. At this point we can apply the result of aforementioned result of Noga Alon, and be done. ■

A.2 Non-Adaptive Testers in the Bounded-Degree Graph Model

Recall that, for any function q, if a property can be tested in $o(\sqrt{N} \cdot q(\epsilon))$ non-adaptive queries in the bounded-degree graph model, then it depends only on the vertex degree distribution [53]. In contrast, we show that triangle-freeness can be tested by $O(\sqrt{N/\epsilon})$ non-adaptive queries (in the same model).

The tester selects at random $O(\sqrt{N/\epsilon})$ vertices, queries for the neighbors of each of them, and accepts if and only if the subgraph discovered contains no triangles. Note that if the input graph is ϵ-far from triangle-freeness, then it contains $\Omega(\epsilon N)$ triangles, whereas a random sample of $O(\sqrt{N/\epsilon})$ vertices is likely to hit two vertices of such a triangle.

The argument can be extended to testing H-freeness, for any fixed H, with $O((N/\epsilon)^{1-\frac{1}{\beta(H)}})$ non-adaptive queries, where $\beta(H)$ denotes the minimum vertex cover of H. In this case, if the input graph is ϵ-far from being H-free, then a sample of $O((N/\epsilon)^{1-\frac{1}{\beta(H)}})$ random vertices is likely to hit all vertices in a vertex cover of one of the copies of H. A more general statement, with weaker quantitative bounds, follows.

Proposition A.2 *Let Π be a graph property having a q-query proximity-oblivious tester of detection probability ρ, in the bounded-degree model. Then, in this model, Π can be tested by $O(N^{\frac{q-1}{q}}/\rho(\epsilon))$ non-adaptive queries.*

Actually, Proposition A.2 holds also when q is an upper bound on the number of different vertices that appear in the queries of the proximity-oblivious tester.

Proof: The main observation is that a sample of $O(N^{1-(1/q)})$ vertices (along with the neighbor queries that correspond to each vertex) is likely to allow for the emulation of a random execution of the proximity-oblivious tester (POT). Specifically, given a q-query POT, we consider the following non-adaptive POT:

1. Select a random sample of $O(N^{1-(1/q)})$ vertices, denoted S, and query the neighborhood of each vertex in S. For every $(v, i) \in S \times [d]$, denote the oracle answer by $\Gamma_i(v)$.

   These are all the queries made by the new POT, and the following steps only involve computations (and no actual queries).

2. Select and fix random coins for T, deriving a residual deterministic oracle machine T'.

3. Let $S = \{s_1, ..., s_{|S|}\}$, and $\overline{S} \stackrel{\text{def}}{=} \{(s_{(i-1)q+1}, ..., s_{iq}) : i \in [|S|/2q]\}$; that is, $\overline{S}$ consists of q-sequences of elements in S such that no element appears twice.

   For every $(v_1, ..., v_q) \in \overline{S}$, try to emulate an execution of T using the information obtained in Step 1. For $j = 1, ..., q$, proceed as follows, where initially the permutation $\pi : [N] \to [N]$ is totally undetermined.

   (a) Obtain the j^{th} query of T', denoted (u_j, i_j).
   If π is undetermined on u_j, then determine $\pi(u_j) = v_j$.
   If π is determined on u_j and $\pi(u_j) \notin S$, then this emulation is terminated.
   Thus, the algorithm proceeds to Step 3b only if $\pi(u_j) \in S$, whereas in this case the value of $\Gamma_{i_j}(\pi(u_j))$ is known.

(b) Let $a_j = \Gamma_{i_j}(\pi(u_j))$, and suppose that $a_j \in [N]$ (as otherwise we provide a_j as the oracle answer to T', and proceed to the next iteration).[16] If π^{-1} is undetermined on a_j, then select at random a vertex u such that π is undetermined on u, and determine $\pi(u) = a_j$. Provide u as the oracle answer to T', and proceed to the next iteration.

Note that it is quite likely that $a_j \notin S$, and in this case if T' subsequently issues a query of the form $(u, \cdot)$ then the emulation will be terminated (in the corresponding execution of Step 3a).

If the current emulation is successfully completed, then halt and output the corresponding verdict of T'. Otherwise, proceed to the next $(v_1, ..., v_q) \in \overline{S}$, while resetting π to be totally undetermined.

4. If no emulation is successfully completed, then halt and output the verdict 1 (i.e., accept).

Each execution of Step 3b may yield a value $a_j \notin S$, with probability at least $1-(|S|/N)$. However, with probability at least $|S|/2N$, it holds that $a_j \in S$. Thus, for each $(v_1, ..., v_q) \in \overline{S}$, we complete an emulation of T' (in Step 3) with probability at least $(|S|/2N)^{q-1} \gg 1/|\overline{S}|$. Furthermore, such an emulation correspond to the execution of T' on a random isomorphic copy of the input graph.

To see that, with high probability, at least one of the $|\overline{S}|$ emulations is completed, we consider all $|\overline{S}|$ emulations simultaneously. Let $u_1^{(i)}, ..., u_q^{(i)}$ denote the sequence of vertices that occur in the i^{th} emulation, and let $\pi^{(i)}$ denote the corresponding permutation. We partition the $|S|/2$ samples that do not appear in $\overline{S}$ into q equal sets, denoted $S_1, ..., S_q$, and terminate the i^{th} emulation in iteration $j < q$ if $a_j^{(i)} \notin S_j$. (Indeed, this only makes early termination more likely; cf. Step 3b.) Still, on can show by induction on j, that with high probability the number of emulations that are not terminated by iteration j exceeds $|\overline{S}| \cdot (|S|/4qN)^j$. Furthermore, the queries issued in the $j+1^{\text{st}}$ iteration are mostly different, because they are determined based on different sequences in $\overline{S}$. Using $|\overline{S}| \cdot (|S|/4qN)^{q-1} > 1$, we conclude that, with high probability, there exists an emulation that does not terminate before the last iteration.

It follows that the foregoing non-adaptive POT has detection probability at least $\rho/2$. Applying this POT for $O(1/\rho(\epsilon))$ times, we obtain a non-adaptive tester of query complexity $O(N^{1-(1/q)}/\rho(\epsilon))$. ■

Conclusion. Recall that all subgraph-freeness properties do have a proximity-oblivious testers of constant-query complexity in the bounded-degree graph model. Our conclusion is that non-adaptive testers are not totally useless in that model.

A.3 Testing Strong Connectivity with Forward Queries Only

We show that, for any constant $\epsilon > 0$, strong connectivity in bounded-degree digraphs can be tested by using $N^{1-\Omega(1)}$ forward queries (and no backward queries). Needless to say, the same holds for using only backward queries, and in both cases the tester has two-sided error (which is unavoidable).[17]

Proposition A.3 *In the directed bounded-degree model where only forward queries are allowed, strong connectivity can be tested in query complexity* $\exp(1/\epsilon) \cdot N^{1-\frac{1}{t}}$*, where* $t = \lceil 4/\epsilon d \rceil \cdot d < d + (1/\epsilon)$ *and* d *is the in-degree and out-degree bound.*

[16] Recall that in this case a_j is a fixed indication that the relevant vertex has less than i_j neighbors.

[17] The distributions used in [16, Sec. 5.2] can be used to prove an $\Omega(N)$ query bound for one-sided error. The point is that we can find no direct evidence to the fact that a vertex has in-degree zero.

Proof Sketch: Our starting point is the observation that if a graph is ϵ-far from being strongly connected, then it contains at least $\epsilon dN/4$ source and sink components each containing at most $\lceil 4/\epsilon d \rceil$ vertices (cf. [16, Cor. 9]).[18] The easy case is when the graph contains at least $\epsilon dN/8$ small sink component, since these are easy to detect by forward queries. The problematic case is the one in which the graph contains $\epsilon dN/8$ source components, and we start by considering the simple case in which each of these source components consists of a single vertex.

In the latter case we can estimate the number of vertices having in-degree zero, by estimating the number of vertices having in-degree d, $d-1$, all through 1. To estimate the number of vertices having in-degree $i > 1$, we estimate the number of i-way collisions at the head of randomly selected[19] directed edges, and use the information we already gathered regarding in-degree j for every $j > i$. The number of vertices having in-degree 1 is estimated by estimating the collisions between a uniformly selected vertex and the vertex at the head of a uniformly selected random edge. Note that, for every $i \geq 2$, the number of i-way collisions can be estimated by a sample of size $O(N^{1-\frac{1}{i}})$.

In the foregoing, we have relied on the fact that a vertex has zero in-degree if and only if it is a source vertex, and on the hypothesis that many source vertices exist. But, in general, we only know that there are many small source components. So the intuitive idea is to "contract" all small components, and consider in-coming edges at the component level. One small difficulty is that we cannot determine the components of the input graph, and so the following modification is used.

For every vertex v, we let C_v denote the set of vertices u such that v and u reside on a directed cycle of size at most $s \stackrel{\text{def}}{=} \lceil 4/\epsilon d \rceil$. We say that v is good if for every $u \in C_v$ it holds that $C_u = C_v$. Note that, given a vertex v, we can determine C_v as well as whether v is good by using d^s queries. Also note that every vertex that resides in a small source component is good. We now emulate the foregoing procedure on the directed graph in which for every good v the set C_v is contracted to a new vertex, and note that a vertex has in-degree zero in the resulting graph if and only if it represents a small source of G. Noting that the maximum degree in this graph is $s \cdot s$, the claim follows. ■

Conclusion. Our lesson is that some non-trivial testing can be carried out also in the model that allows forward queries only.

[18]Throughout this proof, the word component means a strongly connected component, and source (resp., sink) components are components that have no in-coming (resp., out-going) edges.

[19]We may select a random directed edge by selecting a vertex uniformly, and selecting each of its out-going edges with probability $1/d$.