Property Testing in Massive Graphs

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

July 16, 1999

Abstract

We consider the task of evaluating properties of graphs that are too big to be even scanned. Thus, the input graph is given in form of an oracle which answers questions of the form *is there an edge between vertices u and v*, or *who is the i^{th} neighbor of v*. Our task is to determine whether a given input graph has a predetermined property or is "relatively far" from any graph having the property. Distance between graphs is measured as the fraction of the possible queries on which the corresponding oracles, representing the two graphs, differ. We show that randomized algorithms of running-time substantially smaller than the size of the input graph may reach (with high probability) a correct verdict regarding whether the graph has some predetermined property (such as being bipartite) or is far from having it.

1 Introduction

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 (equiv., 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 processing 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 $\mu : S \mapsto [0,1]$ 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 $\overline{\mu}$ up-to $\pm\epsilon$. That is,

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

Graphs capture more complex features of a data-set; 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 $\mathrm{G} = (S,R)$, where the elements of S are called vertices and the elements in R are called edges. In this survey, we focus on two types of graphs:

1. Dense graphs: Such graphs have many edges; specifically, $|R| = \Omega(|S|^2)$ (say $|R| > 0.1 \cdot |S|^2$). A natural representation of such graphs is by an oracle which on query a pair $(u,v) \in S \times S$ responds with a bit indicating whether $(u,v) \in R$ or not.

   Such an oracle may be either a look-up table to which we have "direct access" (i.e., we can obtain the (u,v)-entry in unit cost) or a fast procedure which allows us to determine whether $(u,v) \in R$.

2. Bounded-degree graphs: Such graphs have few edges, and furthermore each vertex in them has few neighbors; that is, for some small d (say $d = 10$), $|\{v : (u,v) \in R\}| \leq d$ holds for every $u \in S$. A natural representation of such graphs (of degree bound d) is by an oracle which on query a pair $(u,i) \in S \times \{1,...,d\}$ responds with the name of the $i^{\rm th}$ neighbor of u (or with a special symbol if u has less than i neighbors). Again, such an oracle may be either a look-up table or a fast procedure.

Each of these two types of graphs gives rise to a natural notion of inspecting a portion of the graph: Asking queries to the corresponding oracle (defined above). Also, each of these types of graphs gives rise to a natural notion of distance (among graphs): Specifically, two dense graphs (over vertex set S) are considered relatively close if they differ on $o(|S|^2)$ edges, whereas two bounded-degree graphs are considered relatively close if they differ on $o(|S|)$ edges. A natural notion of approximation emerges: *A graph approximately has a predetermined property if it is relatively close to a graph having this property.* This leads to the following definition.

Testing graph properties – informal definition: A graph property is a set of graphs closed under graph isomorphism (renaming of vertices). Let $\mathcal{P}$ be such a property. A $\mathcal{P}$-tester is a *randomized* algorithm that is given oracle access to a graph, and has to determine whether the graph is in $\mathcal{P}$ or is far from being in $\mathcal{P}$. The type of oracle and distance-measure depend on the model, and we focus on two such models:

1. The adjacency predicate model: Here the $\mathcal{P}$-tester is given oracle access to a function $f : S \times S \mapsto \{0,1\}$ which represents the graph $\mathrm{G}_f = (S, f^{-1}(1))$, where

$$f^{-1}(1) \stackrel{\rm def}{=} \{(u,v) \in S \times S : f(u,v) = 1\}\,.$$

   The tester is also given a distance parameter $\epsilon > 0$, and is required to accept f if $\mathrm{G}_f \in \mathcal{P}$ and reject f if G_f *differs in more than* $\epsilon|S|^2$ *edges* from any graph in $\mathcal{P}$. (The tester is allowed to err, in each of these cases, with some small probability.)

2. The incidence function model: Here the $\mathcal{P}$-tester is given oracle access to a function $f : S \times \{1,...,d\} \mapsto S \cup \{\perp\}$ which represents the graph $\mathrm{G}_f = (S, \{(u,v) : u \in S\,,\, v \in \Gamma_f(u)\})$, where

$$\Gamma_f(u) \stackrel{\rm def}{=} \{v : \exists i\, f(u,i) = v\}\,.$$

   The tester is also given a distance parameter $\epsilon > 0$, and is required to accept f if $\mathrm{G}_f \in \mathcal{P}$ and reject f if G_f *differs in more than* $\epsilon d|S|$ *edges* from any graph in $\mathcal{P}$. (Again, a small error probability is allowed.)

Thus, in both cases, rejection is required only in case the corresponding representation is ϵ-far from having the property $\mathcal{P}$. In both cases ϵ-far refers to difference of more than ϵ fraction of the entries in the oracle. The adjacency predicate model is most adequate for testing of dense graphs, whereas the incidence function model is adequate for testing bounded-degree graphs.

The definition of property testing is a relaxation of the standard definition of a decision task (where one is required to decide correctly on ALL graphs): The tester is allowed arbitrary behavior when the graph does not have the property, and yet is ϵ-close to a graph having the property. Thus, a property tester may be far more efficient than a standard decision procedure (for the same property). We shall see that this is indeed the case for a variety of graph properties. But before doing so we wish to further discuss the notion of approximation underlining the above definition.

Firstly, being close to a graph which has the property is a notion of approximation which, in certain applications, may be of great value. Furthermore, in some cases, being close to a graph having the property translates to a standard notion of approximation (see Section 3.2). In other cases, it translates to a notion of "dual approximation" (see, again, Section 3.2).

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 which represents some design problem, where Bipartite graphs corresponds 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 which will cost a lot to modify. That is, we may still accept bad designs, but only such which are close to being good and thus will not cost too much to later modify.

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, in case the tester has one-sided error (i.e., it always accepts a graph having the property), we have obtain 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.

Organization: In Section 2, we redefine the two models of testing graph properties discussed informally above. In Section 3 (resp., Section 4) we survey results in the first (resp., second) model. We will contrast the two models by considering the complexity of testing bipartiteness in both of them. Additional models for testing graph properties are discussed in Section 5. A wider perspective on property testing is provided in Section 6.

2 Testing Graph Properties – Two Models

We assume some familiarity with basic notions of graph theory and graph algorithms (cf., [18]). We switch to more standard notations for graphs (i.e., denote the vertex and edge sets V and E, respectively). Furthermore, we typically assume that the vertex set V equals $\{1, ..., |\mathrm{V}|\}$. Here we discuss undirected graphs, and so both representations presented below are redundant (since each edge appears twice).

Two natural representations of a graph are offered by its adjacency matrix and by its incidence list. Correspondingly, we consider two representations of graphs by functions.

1. An N-vertex graph, $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, can be represented by the *adjacency predicate*, $f : \mathrm{V} \times \mathrm{V} \mapsto \{0, 1\}$, so that $(u, v) \in \mathrm{E}$ if and only if $f(u, v) = 1$.
2. An N-vertex graph of degree bound d, $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, can be represented by the *incidence function*, $g : \mathrm{V} \times \{1, ..., d\} \mapsto \mathrm{V} \cup \{0\}$, so that $g(u, i) = v$ if v is the i^{th} vertex incident at u, and $g(u, i) = 0 \notin \mathrm{V}$ if u has less than i neighbors.

As usual, the choice of representation has a fundamental impact on the potential algorithm. Here the impact is even more dramatic since we seek algorithms which only inspect a relatively small fraction of the object (graph represented by a function). Furthermore, there is another fundamental impact of the choice of representation on the task of property testing. This has to do with our definition of distance, which is relative to the size of the domain of the function. In particular, distance ϵ in the first representation means a symmetric difference of $2\epsilon \cdot N^2$ edges, whereas in the second representation this means a symmetric difference of $2\epsilon \cdot dN$ edges. (In both cases, the extra factor 2 is due to the redundant representation which is adopted for sake of simplicity.)

Definition 1 (testing graph properties): *For any* $m \in \mathbb{N}$, *let* $[m] \stackrel{\mathrm{def}}{=} \{1, 2, ..., m\}$. *Let* $\mathcal{P}$ *be a graph property.*

1. *A* $\mathcal{P}$-tester for the adjacency predicate model *is a randomized algorithm that on input a* size parameter $N \in \mathbb{N}$, distance parameter $\epsilon > 0$, *and oracle access to a function* $f : [N] \times [N] \mapsto \{0, 1\}$, *with probability at least* $2/3$, *accepts if* f *represents a graph in* $\mathcal{P}$ *and rejects if* f *is* ϵ*-far from any graph in* $\mathcal{P}$. *That is,*
   - *If* $\mathrm{G} = ([N], f^{-1}(1))$ *is in* $\mathcal{P}$ *then the algorithm accepts with probability at least* $2/3$.
   - *If for every* $h : [N] \times [N] \mapsto \{0, 1\}$ *such that* $([N], h^{-1}(1)) \in \mathcal{P}$, *the functions* f *and* h *differ in more than* $\epsilon \cdot N^2$ *entries then the algorithm rejects with probability at least* $2/3$.

2. *A $\mathcal{P}$-tester for d-bounded incidence function model is a randomized algorithm that on input a size parameter $N \in \mathbb{N}$, distance parameter $\epsilon > 0$, and oracle access to a function $f : [N] \times [d] \mapsto \{0\} \cup [N]$ with probability at least 2/3 accepts if f represents a graph in $\mathcal{P}$ and rejects if f is ϵ-far from any graph in $\mathcal{P}$. That is, letting $\Gamma_g(u) \stackrel{\text{def}}{=} \{v : \exists i\, g(u,i) = v\}$,*
   - *If $G = ([N], \{(u,v) \in [N] \times [N] : v \in \Gamma_f(u)\})$ is in $\mathcal{P}$ then the algorithm accepts with probability at least 2/3.*
   - *If for every $h : [N] \times [d] \mapsto \{0\} \cup [N]$ such that $([N], \{(u,v) \in [N] \times [N] : v \in \Gamma_h(u)\}) \in \mathcal{P}$, the functions f and h differ in more than $\epsilon \cdot dN$ entries then the algorithm rejects with probability at least 2/3.*

As usual, the error probability may be decreased by successive applications of the tester.

The first model (i.e., adjacency predicate) is most appropriate for dense graphs (i.e., $|E| = \Omega(|V|^2)$), whereas the second model (i.e., d-bounded incidence function) is applicable and most appropriate for graphs of degree bound d. Below, we demonstrate the difference between the two representations by considering the task of testing whether a graph is Bipartite.

We will be focus on the query and time complexity of such testers, as a function of N and ϵ. By query complexity we mean the number of oracle queries made by the algorithm. In evaluating the running time, we count each query at unit cost.[1]

We stress that the testing algorithms are allowed to be randomized, and that this is of key importance for achieving query complexity which is significantly lower than the size of the graph. A deterministic tester for any "non-degenerate" property needs to query the oracle on a constant fraction of its domain, and so is of little interest in our context.

3 The First Model (Adjacency Predicates)

In this section we consider the representation of N-vertex graphs by adjacency predicates mapping pairs $\{1,2,...,N\} \times \{1,2,...,N\}$ to $\{0,1\}$. Recall that in this model, distance between graphs refers to the fraction of different edges over $N^2/2$.

3.1 Some Known Results

Testers of complexity which depends only on the distance parameter, ϵ, are known for several natural graph properties [22, 1]. In particular, the following properties can be tested in query-complexity $\mathrm{poly}(1/\epsilon)$ and time complexity $\exp(\mathrm{poly}(1/\epsilon))$ (cf., Goldreich, Goldwasser and Ron [22]):[2]

- k-Colorability[3], for any fixed $k \geq 2$. The query-complexity is $\mathrm{poly}(k/\epsilon)$, and for $k = 2$ the running-time is $\widetilde{O}(1/\epsilon^3)$, where by $\widetilde{O}(m)$ we mean $O(m \cdot \mathrm{poly}(\log m))$. In case the graph is k-colorable the tester always accepts, whereas in case the graph is ϵ-far from k-colorable the tester rejects with probability at least $\frac{2}{3}$ and furthermore supplies a small counterexample (in the form of a small subgraph which is not k-colorable).

  The 2-Colorability (equiv., Bipartite) Tester is presented in Subsection 3.3. An improved analysis has been recently obtained by Alon and Krivelevich [2].

[1] Alternatively, one may consider a RAM model of computation, in which trivial manipulation of vertices (e.g., reading/writing a vertex and comparing vertices) is performed at unit cost.

[2] Except for Bipartite testing, running-time of $\mathrm{poly}(1/\epsilon)$ is unlikely, as it will imply $\mathcal{NP} \subseteq \mathcal{BPP}$.

[3] A graph is k-colorable if its vertices can be partitioned into k parts so that there are no edges among vertices residing in the same part.

- ρ-Clique, for any $\rho > 0$. That is, does the N-vertex graph have a clique (i.e., a set of vertices with edges among each pair in it) of size ρN.
- ρ-CUT, for any $\rho > 0$. That is, does the N-vertex graph have a cut of size at least ρN^2. A generalization to k-way cuts works within query-complexity $\text{poly}((\log k)/\epsilon)$.
- ρ-Bisection, for any $\rho > 0$. That is, can the vertices of the N-vertex graph be partitioned into two equal parts with at most ρN^2 edges going between them.

All the above property testing problems are special cases of the General Graph Partition Testing Problem, 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 has been presented in [22] too: The tester uses $\widetilde{O}(k^2/\epsilon)^{2k+O(1)}$ queries, and runs in time exponential in its query-complexity.

Going beyond the General Graph Partition Problem: Although many natural graph properties can be formulated as partition problems, many more cannot. Among these we mention a few classes which certainly do not exhaust all graph properties.

- Many natural graph properties are very easy to test in the adjacency predicate model. A very partial list includes Connectivity, Hamiltonicity, Cycle-freeness and Planarity (cf., [22]). The reason being that for these properties either every N-vertex graph is at distance at most $O(1/N)$ from a graph having the desired property (and so for $\epsilon = \Omega(1/N)$ the trivial algorithm which always accepts will do), or the property holds only for sparse graphs (and so for $\epsilon = \Omega(1/N)$ one may reject any non-sparse graph).
- On the other hand, there are ("unnatural") graph properties in $\mathcal{NP}$ which are extremely hard to test; namely, any testing algorithm must inspect at least $\Omega(N^2)$ of the vertex pairs [22].
- Alon *et. al.* [1] have recently suggested to study graph properties through the type (or "logical complexity") of a formula defining the property. Specifically, they considered graph properties expressible by first order formulae. They showed that every graph property expressible by such formula of the form $\exists^*\forall^*$ is testable in complexity depending only on ϵ (alas the dependency many be terrible; e.g., a tower of $\text{poly}(1/\epsilon)$ exponentiations). In contrast, they also showed that there exists a (natural) graph property expressible by a first order formula of the form $\forall^+\exists^+$ which cannot be tested within complexity depending only on ϵ.

In view of the above, we believe that providing a characterization of graph properties, according to the complexity of testing them, may be very challenging.

From Testing to Searching: Most graph properties discussed above are in $\mathcal{NP}$. Furthermore, in these cases the NP-witness for G having property $\mathcal{P}$ is a natural structure in the graph; for example, in case of the General Graph Partition Problems the witness is merely an adequate partition of the vertices. Interestingly, our testers for (all cases of) the General Graph Partition Problem, can be modified into algorithms which provide such approximate NP-witnesses. That is, if the graph has the desired (partitioning) property, then the testing algorithm may actually output auxiliary information that allows to construct, in poly$(1/\epsilon) \cdot N$-time, a partition which approximately obeys the property. For example, for ρ-CUT, we can construct a partition with at least $(\rho - \epsilon) \cdot N^2$ crossing edges.

One-sided error probability: The k-Colorability tester has one-sided error: It always accepts k-colorable graphs. Furthermore, when rejecting a graph, this tester always supplies a poly$(1/\epsilon)$-size subgraph which is not k-colorable. The other algorithms for all the other cases of the General Graph Partition Problem discussed above, have two-sided error. This is unavoidable within $o(N)$ query-complexity.

3.2 Testing vs Deciding and Other Forms of Approximation

We shortly discuss the relationship of the notion of approximation underlying the definition of testing graph properties to more traditional notions. The latter include exact decision as well as other notions of approximation.

Relation to recognizing graph properties: Our notion of testing a graph property $\mathcal{P}$ is a *relaxation* of the notion of *deciding the graph property* $\mathcal{P}$ which has received much attention in the last three decades [35]. In the classical problem there are no margins of error – one is required to accept all graphs having property $\mathcal{P}$ and reject all graphs which lack it. In 1975 Rivest and Vuillemin [40] resolved the Aanderaa–Rosenberg Conjecture [38], 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 $\Omega(N^2)$. Progress towards this goal was made by Yao [43], King [32] and Hajnal [25] culminating in an $\Omega(N^{4/3})$ lower bound. This stands in striking contrast to the testing results of [22] mentioned above, by which some non-trivial monotone graph properties can be *tested* by examining a constant number of locations in the matrix.

Application to the standard notion of approximation: The relation of testing graph properties to the standard notions of approximation is best illustrated in the case of Max-CUT. Any tester for the class ρ-cut, working in time $T(\epsilon, N)$, yields an algorithm for approximating 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 [22], 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, improving over previous work of Arora *et. al.* [4] and de la Vega [15] 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 constructing 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 [22] can be used to find such a cut in time linear in N.

Relation to "dual approximation" (cf., [29, 30])**:** To illustrate this relation, we consider the ρ-Clique Tester mentioned above. 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-Cliqueness", 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 [12, 26, 27]), 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.

3.3 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 1 (Bipartite Tester in the first model [22]):
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; see [18]). 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 [2] implies that the above algorithm 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 1 (Goldreich, Goldwasser and Ron [22]): *Algorithm 1 is a Bipartite Tester* (in the adjacency predicate 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 induced by it. 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 U_1 (resp., U_2) must be placed opposite to U_1 (resp., 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 which 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 $\mathrm{V}_1 \cap \mathrm{V}_1$ is indeed empty). As suggested above, if one claims that G can be "bipartited" with U_1 and U_2 on different sides then $\mathrm{V}_1 = \Gamma(\mathrm{U}_1)$ must be on the opposite side to U_2 (and $\Gamma(\mathrm{U}_1)$ opposite to U_1). Note that the partition of U places no restriction on vertices which have no neighbor in U. Thus, we first ensure that *most* "influential" (i.e., "high-degree") vertices in V have a neighbor in U.

Definition 3.1 (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 *for* V *if all but at most $\frac{\epsilon}{3}N$ of the high-degree vertices in* V *have a neighbor in* U.

Note that NOT insisting that U neighbors *all* high-degree vertices allows us to show that a random U of size unrelated to the size of the graph has this feature. (If we were to insist that U neighbors *all* high-degree vertices then we would have had to use $|\mathrm{U}| = \Omega(\log N)$.)

Claim 3.2 *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 which 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. $\Box$

Definition 3.3 (disturbing a partition of U): *We say that an edge* disturbs *a partition* $(\mathrm{U}_1, \mathrm{U}_2)$ *of* U *if both is end-points are in the same* $\Gamma(\mathrm{U}_i)$*, for some* $i \in \{1,2\}$.

Claim 3.4 *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 which are not disturbing. Actually, we upper bound the number of edges which have an end-point not in $\Gamma(\mathrm{U})$.

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

The theorem follows by observing that 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 G_R. Using Claim 3.2 the probability of event (1) is bounded by $1/6$; and using Claim 3.4 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 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 is due to $m = \Omega(t/\epsilon)$. The theorem follows. $\blacksquare$

Comment: The procedure employed in the proof yields a $\text{poly}(1/\epsilon) \cdot N$-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) which is consistent with the bipartite partition of R, and partitioning V as done in the proof (with vertices which 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.

4 The Second Model (Incidence Functions)

In this section we consider the representation of N-vertex graphs of degree bound d by incidence functions mapping pairs $\{1,2,...,N\} \times \{1,2,...,d\}$ to $\{0,1,2,...,N\}$. Recall that in this model, distance between graphs refers to the fraction of different edges over $dN/2$.

4.1 Some Known Results

Testers of complexity which depends only on the distance parameter, ϵ, are known for several natural graph properties (cf., Goldreich and Ron [23]). In particular, the following properties can be tested in time (and thus query-complexity) $\text{poly}(d/\epsilon)$:

- *Connectivity.* The tester runs in time $\widetilde{O}(1/\epsilon)$. In case the graph is connected the tester always accepts, whereas in case the graph is ϵ-far from being connected the tester rejects with probability at least $\frac{2}{3}$ and furthermore supplies a small counter-example to connectivity (in the form of an induced subgraph which is disconnected from the rest of the graph).
- *k-edge-connectivity.* The tester runs in time $\widetilde{O}(k^3 \cdot \epsilon^{-3})$. For $k = 2,3$ improved testers have running-times $\widetilde{O}(\epsilon^{-1})$ and $\widetilde{O}(\epsilon^{-2})$, respectively. Again, k-edge-connected graphs are always accepted, and rejection is accompanied by a counter-example.
- *k-vertex-connectivity, for $k = 2,3$.* The testers run in time $\widetilde{O}(\epsilon^{-k})$.
- *Planarity.* The tester runs in time $\widetilde{O}(d^4 \cdot \epsilon^{-1})$. Planar graphs are always accepted, and rejection is accompanied by a counter-example (in the form of a subgraph homomorphic to $K_{3,3}$ or to K_5).
- *Cycle-Freeness.* The tester runs in time $\widetilde{O}(\epsilon^{-3})$. Unlike all other algorithms, this tester has two-sided error probability, which is unavoidable for testing this property within $o(\sqrt{N})$ queries.

The complexity of Bipartiteness testing is considered in Subsections 4.2 and 4.3: We survey an $\Omega(\sqrt{N})$ lower bound on the query complexity of any tester [23], and a matching upper bound [24]. The lower bound stands in sharp contrast to the situation in the first model (i.e., representation by adjacency predicates), where Bipartite testing is possible in $\text{poly}(1/\epsilon)$-time. The query complexity upper bound (for the incidence function representation) is obtained by a natural Bipartite-tester of running time (and query complexity) $\widetilde{O}(\text{poly}(1/\epsilon) \cdot \sqrt{N})$. We stress that, for $\epsilon > N^{-\Omega(1)}$, the testers in both models are faster than the decision procedure.

4.2 A Lower Bound on the Complexity of Testing Bipartiteness

In contrast to Theorem 1, under the incidence function representation there exists no Bipartite tester of complexity independent of the graph size.

Theorem 2 (Goldreich and Ron [23]): *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 which are composed by 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 which are connected by an perfect matching edge must be odd.

Clearly, all graphs in $\mathcal{G}_2^N$ are bipartite. One first proves that almost all graphs in $\mathcal{G}_1^N$ are far from being bipartite. Afterwards, one proves that a testing algorithm that performs less than $\alpha\sqrt{N}$ queries (for some constant $\alpha < 1$) is not able to 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 will be far from bipartite). Loosely speaking, this is done by showing that in both cases the algorithm is unlikely to encounter a cycle (among the vertices it has inspected). ■

4.3 An Algorithm for Testing Bipartiteness

The lower bound of Theorem 2 is essentially tight. Furthermore, the following natural algorithm constitutes a Bipartite tester of running time $\text{poly}((\log N)/\epsilon) \cdot \sqrt{N}$.

Algorithm 2 (Bipartite Tester in the second model [24]):
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* (resp., R_1) *denote the vertices set reached from s in an even* (resp., 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 one of the above T iterations, then it accepts.

Theorem 3 (Goldreich and Ron [24]): *Algorithm 2 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 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, 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 $L \geq O(\log N)$).

The key quantities is 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 which remain at current vertex). Let $p^0(v)$ (resp., $p^1(v)$) denote the probability that a (modified) random walk of length L starting at s reaches v while making an even (resp., odd) number of real (i.e., non-self-loop) steps. By the rapid mixing assumption we have (for every $v \in \mathrm{V}$)

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

We consider two cases regarding the sum $\sum_{v\in\mathrm{V}} p^0(v)p^1(v)$ – In case the sum is (relatively) "small", we show that V can be 2-partitioned so that there are relatively few edges between vertices placed in the same side, which implies that G is close to be 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. The two cases are presented in greater detail in the following (corresponding) two claims.

Claim 4.1 *Suppose* $\sum_{v\in\mathrm{V}} p^0(v)p^1(v) \;\leq\; \epsilon/50N$. *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 a walk of length $L-1$ with path-parity 1 ending at u is almost as likely as such a walk having length L, and that once such a walk reaches u it continues to v in the next step with probability exactly $1/2d$. Thus, such an edge contributes at least $\frac{(1/4N)^2}{3d}$ to the sum $\sum_{v\in\mathrm{V}} p^0(v)p^1(v)$. The claim follows. ■

Claim 4.2 *Suppose* $\sum_{v\in\mathrm{V}} p^0(v)p^1(v) \;\geq\; \epsilon/50N$, *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

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

where the second inequality is due to the setting of K, and the third to the claim's hypothesis. Intuitively, we expect that with high probability $|R_0 \cap R_1| > 0$. 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., [3, Sec. 4.3]). Specifically, letting $\mu \stackrel{\text{def}}{=} \sum_{v \in \mathrm{V}} p^0(v) p^1(v)$ $(= \mathbf{E}(\zeta_{i,j}))$, and $\overline{\zeta}_{i,j} \stackrel{\text{def}}{=} \zeta_{i,j} - \mu$, we get:

$$\begin{aligned}
\mathbf{Pr}\left(\sum_{i \neq j} \zeta_{i,j} = 0\right) &< \frac{\mathbf{V}(\sum_{i \neq j} \zeta_{i,j})}{(K^2\mu)^2} \\
&= \frac{1}{K^4\mu^2} \cdot \left(\sum_{i,j} \mathbf{E}(\overline{\zeta}_{i,j}^2) + 2 \sum_{i,j,k} \mathbf{E}(\overline{\zeta}_{i,j}\overline{\zeta}_{i,k})\right) \\
&< \frac{1}{K^2\mu} + \frac{2}{K\mu^2} \cdot \mathbf{E}(\zeta_{1,2}\zeta_{1,3})
\end{aligned}$$

For the second term, we observe that $\mathbf{Pr}(\zeta_{1,2} = \zeta_{2,3} = 1)$ is upper bounded by the probability that $\zeta_{1,2} = 1$ 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 $K < N/4$, $\mu \geq \epsilon/50N$ and $K^2 \geq 6 \cdot 50N/\epsilon$. The claim follows. ■

Beyond rapid mixing graphs. The proof in [24] refers to a more general sum of products; that is, $\sum_{v \in \mathrm{V}} p_{\text{odd}}(v) p_{\text{even}}(v)$, where $\mathrm{U} \subseteq \mathrm{V}$ is an appropriate set of vertices, and $p_{\text{odd}}(v)$ (resp., $p_{\text{even}}(v)$) is the probability that a random walk (starting at s) passes through v after more than $L/2$ steps and the corresponding path to v has odd (resp., even) parity. Much of the analysis in [24] 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 $p_{\text{odd}}(u) + p_{\text{even}}(u) \approx 1/\sqrt{|\mathrm{V}| \cdot |\mathrm{U}|}$, for every $u \in \mathrm{U}$. The selection is based on the "combinatorial treatment of expansion" of Mihail [36]. Specifically, we use the counterpositive 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 U above 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 [24] 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 $\approx 1/\sqrt{|V| \cdot |U|}$ (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 $H \setminus U$ are accounted for by using the fact that this cut is small (relative to the size of U).

5 Two Other Models

So far our discussion was confined to undirected graphs. Yet, the two models (above) extend naturally to the case of directed graphs. Some of the results for the undirected case extend naturally to the directed case (e.g., testing directed connectivity). A natural task in the directed graph models is testing whether a given directed graph is acyclic (i.e., has no directed cycles). Bender and Ron have presented a Acyclicity-tester of $\text{poly}(1/\epsilon)$ complexity in the adjacency predicate model, and showed that no such tester may exist in the incidence list model [8].

In our discussion above (as well as in the next section) we have linked the issue of representation to the distance measure. That is, when representing a graph by an oracle from some domain D to a range R, we have considered the relative distance of graphs as the fraction of different entries in their representation divided by the size of the domain (i.e., $|D|$). This (quite natural) convention is abandoned by Parnas and Ron who developed a more general model by decoupling the representation of the graph from the distance measure [37]: Whatever is the mechanism of accessing the graph, the distance between graphs is defined as the number of edges in their symmetric difference. (The relative distance may be defined as the latter quantity divided by the total number of edges in both graphs.) The new model allows to treat well the case of sparse graphs which are not of bounded-degree – a case that was not treated in a satisfactory manner in either of the previous two models. Many of the testers for bounded-degree graphs can be extended to the case of sparse graphs in the new model. Furthermore, the following problem of estimating the diameter of a graph is shown to be solvable in this model within complexity $\text{poly}(1/\epsilon)$: Given a diameter parameter D and a distance parameter ϵ, determine whether the graph has diameter at most D or is ϵ-far from any graph of diameter at most $2D+2$ (i.e., one has to add more than $\epsilon \cdot |E|$ edges in order to obtain from the input graph $G = (V, E)$ a graph of diameter at most $2D + 2$). (We comment that in the "bounded-degree model" and for $\epsilon > 1/D$ this task can be easily reduced to testing connectivity.)

6 A Wider Perspective

Our formulation of testing graph properties (in both the adjacency predicate and incident list models) is a special case of property testing of arbitrary functions.

Definition 2 (property tester [39])*: Let S be a finite set, and $\mathcal{P}$ a subset of functions mapping S to $\{0,1\}^*$. A* (property) tester for *$\mathcal{P}$ is a probabilistic oracle machine, M, which given a* distance parameter $\epsilon > 0$ *and oracle access to an arbitrary function $f : S \mapsto \{0,1\}^*$ satisfies the following two conditions:*

1. (the tester accepts every f in $\mathcal{P}$): *If $f \in \mathcal{P}$ then*

$$\mathbf{Pr}(M^f(\epsilon)=1) \geq \frac{2}{3}$$

2. (the tester rejects every f that is far from $\mathcal{P}$): *If* $|\{x \in S : f(x) \neq g(x)\}| > \epsilon \cdot |S|$, *for every* $g \in \mathcal{P}$, *then*

$$\mathbf{Pr}(M^f(\epsilon) = 1) \leq \frac{1}{3}$$

Property testing (as just defined) emerges naturally in the context of program checking [13, 34, 20, 39] and probabilistically checkable proofs (PCP) [9, 10, 19, 6, 5, 11, 14, 7, 12, 26, 28]. Specifically, in the context of program checking, one may choose to test that the program satisfies certain properties before checking that it computes a specified function. This paradigm has been followed both in the theory of program checking [13, 39], and in practice where often programmers first test their programs by verifying that the programs satisfy properties that are known to be satisfied by the function they compute. In the context of probabilistically checkable proofs, the property tested is being a codeword with respect to a specific code. This paradigm, explicitly introduced in [10], has shifted from testing codes defined by low-degree polynomials [9, 10, 19, 6, 5] to testing Hadamard codes [5, 11, 14, 7, 33, 41], and recently to testing the "long code" [12, 26, 28, 41].

Much of the work cited above deals with the development and analysis of testers for algebraic properties; specifically, linearity, multi-linearity, and low-degree polynomials [13, 34, 9, 10, 19, 20, 39, 6, 5, 11, 14, 7]. The study of property testing as applied to combinatorial properties was initiated by Goldreich, Goldwasser and Ron [22]. Specifically, they initiated the study of property testing in the adjacency predicate model. The study of property testing in the incidence function model was latter initiated in [23]. We comment that testing of combinatorial properties other than ones related to graphs has been considered in [17, 21, 16]: Specifically, these works consider the task of testing whether a function $f : \Sigma^n \mapsto R$ is monotone (with respect to orderings of both Σ and R).[4]

Further generalization: We mention that an even more general formulation of property testing was suggested in [22] –

> Let $\mathcal{P}$ be a fixed property of functions, and f be an unknown function. The goal is to determine (possibly probabilistically) if f has property $\mathcal{P}$ or if it is far from any function which has property $\mathcal{P}$, where distance between functions is measured with respect to some distribution D on the domain of f. Towards this end, one is given examples of the form $(x, f(x))$, where x is distributed according to D. One may also be allowed to query f on instances of one's choice.

The above formulation is inspired by the PAC learning model [42]. In fact, property testing is related to variants of PAC learning as has been shown in [22] and [31] (the results in [22] are generic and [31] aims at obtaining better results for properties which are related to concept classes extensively investigated in the machine learning literature). The general formulation above allows the consideration of arbitrary distributions (rather than uniform ones), and of testers which utilize only randomly chosen instances (rather than being able to query instances of their own choice).

References

[1] N. Alon, E. Fischer, M. Krivelevich and M. Szegedy. Efficient Testing of Large Graphs. In *40th FOCS*, to appear, 1999.

[4] That is, if $x_i \leq_\Sigma y_i$, for every i, then a monotone f should satisfy $f(x_1 \cdots x_n) \leq_R f(y_1 \cdots y_n)$. Ergun *et. al.* [17] deal only with the case $n = 1$, Goldreich *et. al.* [21] focuses on the case $\Sigma = R = \{0, 1\}$, whereas Dodis *et. al.* [16] deal with the general case. Specifically, Dodis *et. al.* provide a monotonicity tester with complexity $O(\frac{n}{\epsilon} \cdot (\log |\Sigma|) \cdot (\log |R|))$.

[2] N. Alon, and M. Krivelevich. Testing k-Colorability. In preparation, 1999.

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

[4] S. Arora, D. Karger, and M Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 284–293, 1995.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *JACM*, Vol. 45, pages 501–555, 1998.

[6] S. Arora and S. Safra. Probabilistic checkable proofs: A new characterization of NP. *JACM*, Vol. 45, pages 70–122, 1998.

[7] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. In *Proceedings of the Thirty-Sixth Annual Symposium on Foundations of Computer Science*, pages 432–441, 1995.

[8] M. Bender and D. Ron. Testing Acyclicity. Preprint, 1999.

[9] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[10] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[11] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 294–304, 1993.

[12] M. Bellare, O. Goldreich, and M. Sudan. Free bits, pcps and non-approximability – towards tight results. *SICOMP*, Vol. 27, No. 3, pp. 804–915, June 1998.

[13] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:549–595, 1993.

[14] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 184–193, 1994.

[15] W. F. de la Vega. MAX-CUT has a randomized approximation scheme in dense graphs. To appear in *Random Structures and Algorithms*, 1994.

[16] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron and A. Samorodnitsky. Improved Testing Algorithms for Monotonicity. In *Random99*, to appear.

[17] F. Ergun, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. In *Proceedings of 30th STOC*, pages 259–268, 1998.

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

[19] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. *JACM*, Vol. 43, pages 268–292, 1996.

[20] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 32–42, 1991.

[21] O. Goldreich, S. Goldwasser, E. Lehman and D. Ron. Testing Monotinicity. In *39th FOCS*, pages 426–435, 1998.

[22] 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.

[23] O. Goldreich and D. Ron. Property Testing in Bounded Degree Graphs. In *Proc. of the 29th ACM Symp. on Theory of Computing*, pages 406–415, 1997.

[24] O. Goldreich and D. Ron. A sublinear Bipartite Tester for Bounded Degree Graphs. *Combinatorica*, Vol. 19 (2), pages 1–39, 1999.

[25] P. Hajnal. An $\Omega(n^{4/3})$ lower bound on the randomized complexity of graph properties. *Combinatorica*, 11(2):131–144, 1991.

[26] J. Håstad. Testing of the long code and hardness for clique. In *Proc. of the 28th ACM Symp. on Theory of Computing*, pages 11–19, 1996.

[27] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proc. of the 37th IEEE Symp. on Foundation of Computer Science*, pages 627–636, 1996.

[28] J. Håstad. Getting optimal in-approximability results. In *Proc. of the 29th ACM Symp. on Theory of Computing*, pages 1–10, 1997.

[29] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the Association for Computing Machinery*, 34(1):144–162, January 1987.

[30] D. S. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for machine scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[31] M. Kearns and D. Ron. Testing problems with sub-learning sample complexity. In *Proc. of 11the COLT*, 1998.

[32] V. King. An $\Omega(n^{5/4})$ lower bound on the randomized complexity of graph properties. *Combinatorica*, 11(1):23–32, 1991.

[33] M. Kiwi. *Probabilistically Checkable Proofs and the Testing of Hadamard-like Codes.* PhD thesis, Massachusetts Institute of Technology, 1996.

[34] R. J. Lipton. New directions in testing. Unpublished manuscript, 1989.

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

[36] M. Mihail. Conductance and convergence of Markov chains – A combinatorial treatment of expanders. In *Proceedings 30th Annual Symp. on Foundations of Computer Science*, pages 526–531, 1989.

[37] M. Parnas and D. Ron. Testing the Diameter of Graphs. In *Random99*, to appear.

[38] A. L. Rosenberg. On the time required to recognize properties of graphs: A problem. *SIGACT News*, 5:15–16, 1973.

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

[40] R. L. Rivest and J. Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3:371–384, 1976.

[41] L. Trevisan. Recycling queries in PCPs and in linearity tests. In *Proc. of the 30th ACM Symp. on Theory of Computing*, 1998.

[42] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[43] A. C. C. Yao. Lower bounds to randomized algorithms for graph properties. In *Proceedings of the Twenty-Eighth Annual Symposium on Foundations of Computer Science*, pages 393–400, 1987.