Hierarchy Theorems for Property Testing

Oded Goldreich[1], Michael Krivelevich[2], Ilan Newman[3], and Eyal Rozenberg[4]

[1] Faculty of Math. and Computer Science, Weizmann Institute, Rehovot, Israel
[2] School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel
[3] Department of Computer Science, Haifa University, Haifa, Israel
[4] Department of Computer Science, Technion, Haifa, Israel

Abstract. Referring to the query complexity of property testing, we prove the existence of a rich hierarchy of corresponding complexity classes. That is, for any relevant function q, we prove the existence of properties that have testing complexity $\Theta(q)$. Such results are proven in three standard domains often considered in property testing: generic functions, adjacency predicates describing (dense) graphs, and incidence functions describing bounded-degree graphs. While in two cases the proofs are quite straightforward, the techniques employed in the case of the dense graph model seem significantly more involved. Specifically, problems that arise and are treated in the latter case include (1) the preservation of distances between graph under a blow-up operation, and (2) the construction of monotone graph properties that have local structure.

Keywords: Property Testing, Graph Properties, Monotone Graph Properties, Graph Blow-up, One-Sided vs Two-Sided Error, Adaptivity vs Non-adaptivity.

1 Introduction

In the last decade, the area of property testing has attracted much attention (see the surveys of [F, R], which are already somewhat out-of-date). Loosely speaking, property testing typically refers to sub-linear time 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 testers get oracle access to this function (and thus may be expected to work in time that is sub-linear in the length of the object).

Following most work in the area, we focus on the query complexity of property testing, where the query complexity is measured as a function of the size of the object as well as the desired proximity (parameter). Interestingly, many natural properties can be tested in complexity that only depends on the proximity parameter; examples include linearity testing [BLR], and testing various graph properties in two natural models (e.g., [GGR, AFNS] and [GR1, BSS], respectively). On the other hand, properties for which testing requires essentially maximal query complexity were proved to exist too; see [GGR] for artificial examples in two models and [BHR, BOT] for natural examples in other models. In between these two extremes, there exist natural properties for which the query complexity of testing is logarithmic (e.g., monotonicity [EKK+, GGL+]), a square root (e.g., bipartitness in the bounded-degree model [GR1, GR2]), and possibly other constant powers (see [FM, PRR]).

One natural problem that arises is whether there exist properties of arbitrary query complexity. We answer this question affirmative, proving the existence of a rich hierarchy of query complexity classes. Such hierarchy theorems are easiest to state and prove in the generic case (treated in Section 2): Loosely speaking, for every sub-linear function q, *there exists a property of functions over* $[n]$ *that is testable using* $q(n)$ *queries but is not testable using* $o(q(n))$ *queries.*

Similar hierarchy theorems are proved also for two standard models of testing graph properties: the adjacency representation model (of [GGR]) and the incidence representation model (of [GR1]). For the incidence representation model (a.k.a the bounded-degree graph model), we show (in Section 3) that, for every sub-linear function q, *there exists a property of bounded-degree* N*-vertex graphs that is testable using* $q(N)$ *queries but is not testable using* $o(q(N))$ *queries.* Furthermore, one such property corresponds to the set of N-vertex graphs that are 3-colorable and consist of connected components of size at most $q(N)$.

The bulk of this paper is devoted to hierarchy theorems for the adjacency representation model (a.k.a the dense graph model), where complexity is measured in terms of the number of vertices rather than the number of all vertex pairs. Our main results for the adjacency matrix model are:

1. For every sub-quadratic function q, *there exists a graph property* Π *that is testable in* q *queries, but is not testable in* $o(q)$ *queries.* Furthermore, for "nice" functions q, it is the case that Π is in $\mathcal{P}$ and the tester can be implemented in poly(q)-time. (See Section 4.)
2. For every sub-quadratic function q, there exists a *monotone* graph property Π that is testable in $O(q)$ queries, but is not testable in $o(q)$ queries. (See Section 5.)

The adjacency representation model is further studied in Sections 6 and 7.

Organization of this version. Due to space limitations, several proofs have been either omitted or trimmed. Full proofs can be found in our technical report [GKNR].

Conventions. For sake of simplicity, we state all results while referring to query complexity as a function of the input size; that is, we consider a fixed (constant) value of the proximity parameter, denoted ϵ. In such cases, we sometimes use the term ϵ-testing, which refers to testing when the proximity parameter is fixed to ϵ. All our lower bounds hold for any sufficiently small value of the proximity parameter, whereas the upper bounds hide a (polynomial) dependence on (the reciprocal of) this parameter. In general, bounds that have no dependence on the proximity parameter refer to some (sufficiently small but) fixed value of this parameter.

A remotely related prior work. In contrast to the foregoing conventions, we mention here a result that refers to graph properties that are testable in (query) complexity that only depends on the proximity parameter. This result, due to [AS], establishes a (very sparse) hierarchy of such properties. Specifically, [AS, Thm. 4] asserts that for every function q there exists a function Q and a graph property that is ϵ-testable in $Q(\epsilon)$ queries but is *not* ϵ-testable in $q(\epsilon)$ queries.[1]

2 Properties of Generic Functions

In the generic function model, the tester is given oracle access to a function over $[n]$, and distance between such functions is defined as the fraction of (the number of) arguments on which these functions differ. In addition to the input oracle, the tester is explicitly given two parameters: a size parameter, denoted n, and a proximity parameter, denoted ϵ.

Definition 1. *Let $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$, where Π_n contains functions defined over the domain $[n] \overset{\text{def}}{=} \{1, ..., n\}$. A* tester for a property *Π is a probabilistic oracle machine T that satisfies the following two conditions:*

1. *The tester accepts each $f \in \Pi$ with probability at least $2/3$; that is, for every $n \in \mathbb{N}$ and $f \in \Pi_n$* (and every $\epsilon > 0$), *it holds that* $\Pr[T^f(n, \epsilon)=1] \geq 2/3$.
2. *Given $\epsilon > 0$ and oracle access to any f that is ϵ-far from Π, the tester rejects with probability at least $2/3$; that is, for every $\epsilon > 0$ and $n \in \mathbb{N}$, if $f : [n] \to \{0, 1\}^*$ is ϵ-far from Π_n, then* $\Pr[T^f(n, \epsilon)=0] \geq 2/3$.

We say that the tester has one-sided error *if it accepts each $f \in \Pi$ with probability 1* (i.e., for every $f \in \Pi$ and every $\epsilon > 0$, it holds that $\Pr[T^f(n, \epsilon)=1] = 1$).

Definition 1 does not specify the query complexity of the tester, and indeed an oracle machine that queries the entire domain of the function qualifies as a tester (with zero error probability...). Needless to say, we are interested in testers that have significantly lower query complexity. Recall that [GGR] asserts that in some cases such testers do not exist; that is, there exist properties that require linear query complexity. Building on this result, we show:

Theorem 2. *For every $q : \mathbb{N} \to \mathbb{N}$ that is at most linear, there exists a property Π of Boolean functions that is testable* (with one-sided error) *in $q+O(1)$ queries, but is not testable in $o(q)$ queries* (even when allowing two-sided error).

We start with an arbitrary property Π' of Boolean functions for which testing is known to require a linear number of queries (even when allowing two-sided error). The existence of such properties was first proved in [GGR]. Given

[1] We note that while Q depends only on q, the dependence proved in [AS, Thm. 4] is quite weak (i.e., Q is lower bounded by a non-constant number of compositions of q), and thus the hierarchy obtained by setting $q_i = Q_{i-1}$ for $i = 1, 2, ...$ is very sparse.

$\Pi' = \bigcup_{m\in\mathbb{N}} \Pi'_m$, we define $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$ such that Π_n consists of "duplicated versions" of the functions in $\Pi'_{q(n)}$. Specifically, for every $f' \in \Pi'_{q(n)}$, we define $f(i) = f'(i \bmod q(n))$ and add f to Π_n, where $i \bmod m$ is (non-standardly) defined as the smallest positive integer that is congruent to i modulo m, The proof that Π satisfies the conditions of Theorem 2 appears in our technical report [GKNR].

Comment. Needless to say, Boolean functions over $[n]$ may be viewed as n-bit long binary strings. Thus, Theorem 2 means that, for every sub-linear q, there are properties of binary strings for which the query complexity of testing is $\Theta(q)$. Given this perspective, it is natural to comment that such properties exist also in $\mathcal{P}$. This comment is proved by starting with the hard-to-test property asserted in Theorem 7 of our technical report [GKNR] (or alternatively with the one in [LNS], which is in $\mathcal{L}$).

3 Graph Properties in the Bounded-Degree Model

The bounded-degree model refers to a fixed (constant) 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.[2] Distance between graphs is measured in terms of their aforementioned representation; that is, as the fraction of (the number of) different array entries (over dN). Graph properties are properties that are invariant under renaming of the vertices (i.e., they are actually properties of the underlying unlabeled graphs).

Recall that [BOT] proved that, in this model, testing 3-Colorability requires a linear number of queries (even when allowing two-sided error). Building on this result, we show:

Theorem 3. *In the bounded-degree graph model, for every $q : \mathbb{N} \to \mathbb{N}$ that is at most linear, there exists a graph property Π that is testable* (with one-sided error) *in $O(q)$ queries, but is not testable in $o(q)$ queries* (even when allowing two-sided error). *Furthermore, this property is the set of N-vertex graphs of maximum degree d that are 3-colorable and consist of connected components of size at most $q(N)$.*

We start with an arbitrary property Π' for which testing is known to require a linear number of queries (even when allowing two-sided error). We further assume that Π' is downward monotone (i.e., if $G' \in \Pi'$ then any subgraph of G' is in Π'). Indeed, by [BOT], 3-Colorability is such a property. Given $\Pi' = \bigcup_{n\in\mathbb{N}} \Pi'_n$, we define $\Pi = \bigcup_{N\in\mathbb{N}} \Pi_N$ such that each graph in Π_N consists of connected components that are each in Π' and have size at most $q(N)$; that is, each connected component in any $G \in \Pi_N$ is in Π'_n for some $n \leq q(N)$ (i.e., n denotes this component's size). The proof that Π satisfies the conditions of Theorem 3 appears in our technical report [GKNR].

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

Comment. The construction used in the proof of Theorem 3 is slightly different from the one used in the proof of Theorem 2: In the proof of Theorem 3 each object in Π_N corresponds to a sequence of (possibly different) objects in Π'_n, whereas in the proof of Theorem 2 each object in Π_N corresponds to multiples copies of a single object in Π'_n. While Theorem 2 can be proved using a construction that is analogous to one used in the proof of Theorem 3, the current proof of Theorem 2 provides a better starting point for the proof of the following Theorem 4.

4 Graph Properties in the Adjacency Matrix Model

In the adjacency matrix 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; that is, as the fraction of (the number of) different matrix entries (over N^2). In this model, we state complexities in terms of the number of vertices (i.e., N) rather than in terms of the size of the representation (i.e., N^2). Again, we focus on graph properties (i.e., properties of labeled graphs that are invariant under renaming of the vertices).

Recall that [GGR] proved that, in this model, there exist graph properties for which testing requires a quadratic (in the number of vertices) query complexity (even when allowing two-sided error). It was further shown that such properties are in $\mathcal{NP}$. Slightly modifying these properties, we show that they can be placed in $\mathcal{P}$; see Appendix A of our technical report [GKNR]. Building on this result, we show:

Theorem 4. *In the adjacency matrix model, for every $q : \mathbb{N} \to \mathbb{N}$ that is at most quadratic, there exists a graph property Π that is testable in q queries, but is not testable in $o(q)$ queries.*[3] *Furthermore, if $N \mapsto q(N)$ is computable in $\mathrm{poly}(\log N)$-time, then Π is in $\mathcal{P}$, and the tester is relatively efficient in the sense that its running time is polynomial in the total length of its queries.*

We stress that, unlike in the previous results, the positive part of Theorem 4 refers to a two-sided error tester. This is fair enough, since the negative side also refers to two-sided error testers. Still, one may seek a stronger separation in which the positive side is established via a one-sided error tester. Such a separation is presented in Theorem 6 (except that the positive side is established via a tester that is not relatively efficient).

Outline of the proof of Theorem 4. The basic idea of the proof is to implement the strategy used in the proof of Theorem 2. The problem, of course, is that we need to obtain graph properties (rather than properties of generic Boolean functions). Thus, the trivial "blow-up" (of Theorem 2) that took place on the truth-table (or function) level has to be replaced by a blow-up on the vertex level. Specifically,

[3] Both the upper and lower bounds refer to two-sided error testers.

starting from a graph property Π' that requires quadratic query complexity, we consider the graph property Π consisting of N-vertex graphs that are obtained by a $(N/\sqrt{q(N)})$-factor blow-up of $\sqrt{q(N)}$-vertex graphs in Π', where G is a t-factor blow-up of G' if the vertex set of G can be partitioned into (equal size) sets that correspond to the vertices of G' such that the edges between these sets represent the edges of G'; that is, if $\{i,j\}$ is an edge in G', then there is a complete bipartite between the i^{th} set and the j^{th} set, and otherwise there are no edges between this pair of sets.[4]

Note that the notion of "graph blow-up" does not offer an easy identification of the underlying partition; that is, given a graph G that is as a t-factor blow-up of some graph G', it is not necessary easy to determine a t-way partition of the vertex set of G such that the edges between these sets represent the edges of G'. Things may become even harder if G is merely close to a t-factor blow-up of some graph G'. We resolve these as well as other difficulties by augmenting the graphs of the starting property Π'.

The proof of Theorem 4 is organized accordingly: In Section 4.1, we construct Π based on Π' by first augmenting the graphs and then applying graph blow-up. In Section 4.2 we lower-bound the query complexity of Π based on the query complexity of Π', while coping with the non-trivial question of *how does the blow-up operation affect distances between graphs.* In Section 4.3 we upper-bound the query complexity of Π, while using the aforementioned augmentations in order to obtain a tight result (rather than an upper bound that is off by a polylogarithmic factor).

4.1 The Blow-Up Property Π

Our starting point is any graph property $\Pi' = \bigcup_{n\in\mathbb{N}} \Pi'_n$ for which testing requires quadratic query complexity. Furthermore, we assume that Π' is in $\mathcal{P}$. Such a graph property is presented in Theorem 7 of our technical report [GKNR] (which builds on [GGR]).

The notion of graphs that have "vastly different vertex neighborhoods" is central to our analysis. Specifically, for a real number $\alpha > 0$, we say that a graph $G = (V,E)$ is α-dispersed if the neighbor sets of any two vertices differ on at least $\alpha \cdot |V|$ elements (i.e., for every $u \neq v \in V$, the symmetric difference between the sets $\{w : \{u,w\} \in E\}$ and $\{w : \{v,w\} \in E\}$ has size at least $\alpha \cdot |V|$). We say that a set of graphs is dispersed if there exists a constant $\alpha > 0$ such that every graph in the set is α-dispersed.[5]

The augmentation. We first augment the graphs in Π' such that the vertices in the resulting graphs are dispersed, while the augmentation amount to adding a linear number of vertices. The fact that these resulting graphs are dispersed will be useful for establishing both the lower and upper bounds. The augmentation

[4] In particular, there are no edges inside any set.

[5] Our notion of dispersibility has nothing to do with the notion of dispersers, which in turn is a weakening of the notion of (randomness) extractors (see, e.g., [S]).

is performed in two steps. First, setting $n' = 2^{\lceil \log_2(2n+1) \rceil} \in [2n+1, 4n]$, we augment each graph $G' = ([n], E')$ by $n' - n$ isolated vertices, yielding an n'-vertex graph $H' = ([n'], E')$ in which every vertex has degree at most $n - 1$. Next, we augment each resulting graph H' by a clique of n' vertices and connect the vertices of H' and the clique vertices by a bipartite graph that corresponds to a Hadamard matrix; that is, the i^{th} vertex of H' is connected to the j^{th} vertex of the clique if and only if the inner product modulo 2 of $i - 1$ and $j - 1$ (in $(\log_2 n')$-bit long binary notation) equals 1. We denote the resulting set of (unlabeled) graphs by Π'' (and sometimes refer to Π'' as the set of all labeled graphs obtained from these unlabeled graphs).

We first note that Π'' is indeed dispersed (i.e., the resulting $2n'$-vertex graphs have vertex neighborhoods that differ on at least $n \geq n'/4$ vertices). Next note that testing Π'' requires a quadratic number of queries, because testing Π' can be reduced to testing Π'' (i.e., ϵ-testing membership in Π'_n reduces to ϵ'-testing membership in $\Pi''_{2n'}$, where $n' \leq 4n$ and $\epsilon' = \epsilon/64$). Finally, note that Π'' is also in $\mathcal{P}$, because it is easy to distinguish the original graph from the vertices added to it, since the clique vertices have degree at least $n' - 1$ whereas the vertices of G' have degree at most $(n-1) + (n'/2) < n' - 1$ (and isolated vertices of H' have neighbors only in the clique).[6]

Applying graph blow-up. Next, we apply an (adequate factor) graph blow-up to the augmented set of graphs Π''. Actually, for simplicity of notation we assume, without loss of generality, that $\Pi' = \bigcup_{n \in \mathbb{N}} \Pi'_n$ itself is dispersed, and apply graph blow-up to Π' itself (rather than to Π''). Given a desired complexity bound $q : \mathbb{N} \to \mathbb{N}$, we first set $n = \sqrt{q(N)}$, and next apply to each graph in Π'_n an N/n-factor blow-up, thus obtaining a set of N-vertex graphs denoted Π_N. (Indeed, we assume for simplicity that both $n = \sqrt{q(N)}$ and N/n are integers.) Recall that G is a t-factor blow-up of G' if the vertex set of G can be partitioned into t (equal size) sets, called clouds, such that the edges between these clouds represent the edges of G'; that is, if $\{i, j\}$ is an edge in G', then there is complete bipartite between the i^{th} cloud and the j^{th} cloud, and otherwise there are no edges between this pair of clouds. This yields a graph property $\Pi = \bigcup_{N \in \mathbb{N}} \Pi_N$.

Let us first note that Π is in $\mathcal{P}$. This fact follows from the hypothesis that Π' is dispersed: Specifically, given any graph N-vertex graph G, we can cluster its vertices according to their neighborhood, and check whether the number of clusters equals $n = \sqrt{q(N)}$. (Note that if $G \in \Pi_N$, then we obtain exactly n (equal sized) clusters, which correspond to the n clouds that are formed in the N/n-factor blow-up that yields G.) Next, we check that each cluster has size N/n and that the edges between these clusters correspond to the blow-up of some n-vertex G'. Finally, we check whether G' is in Π'_n (relying on the fact that $\Pi' \in \mathcal{P}$). Proving that the query complexity of testing Π indeed equals $\Theta(q)$ is undertaken in the next two sections.

[6] Once this is done, we can verify that the original graph is in Π (using $\Pi \in \mathcal{P}$), and that the additional edges correspond to a Hadamard matrix.

4.2 Lower-Bounding the Query Complexity of Testing Π

In this section we prove that the query complexity of testing Π is $\Omega(q)$. The basic idea is reducing testing Π' to testing Π; that is, given a graph G' that we need to test for membership in Π'_n, we test its N/n-factor blow-up for membership in Π_N, where N is chosen such that $n = \sqrt{q(N)}$. This approach relies on the assumption that the N/n-factor blow-up of any n-vertex graph that is far from Π'_n results in a graph that is far from Π_N. (Needless to say, the N/n-factor blow-up of any graph in Π'_n results in a graph that is in Π_N.)

As shown by Arie Matsliah (see Appendix B of our technical report [GKNR]), the aforementioned assumption does *not* hold in the strict sense of the word (i.e., it is not true that the blow-up of any graph that is ϵ-far from Π' results in a graph that is ϵ-far from Π). However, for our purposes it suffices to prove a relaxed version of the aforementioned assumption that only asserts that *for any $\epsilon' > 0$ there exists an $\epsilon > 0$ such that the blow-up of any graph that is ϵ'-far from Π' results in a graph that is ϵ-far from Π*. Below we prove this assertion for $\epsilon = \Omega(\epsilon')$ and rely on the fact that Π' is dispersed. In Appendix B of our technical report [GKNR], we present a more complicated proof that holds for arbitrary Π' (which need not be dispersed), but with $\epsilon = \Omega(\epsilon')^2$.

Claim 4.1. *There exists a universal constant $c > 0$ such that the following holds for every n, ϵ', α and* (unlabeled) *n-vertex graphs G'_1, G'_2. If G'_1 is α-dispersed and ϵ'-far from G'_2, then for any t the* (unlabeled) *t-factor blow-up of G'_1 is $c\alpha \cdot \epsilon'$-far from the* (unlabeled) *t-factor blow-up of G'_2.*

Using Claim 4.1 we infer that *if G' is ϵ'-far from Π' then its blow-up is $\Omega(\epsilon')$-far from Π*. This inference relies on the fact that Π' is dispersed (and on Claim 4.1 when applied to $G'_2 = G'$ and every $G'_1 \in \Pi'$).

Proof. Let G_1 (resp., G_2) denote the (unlabeled) t-factor blow-up of G'_1 (resp., G'_2), and consider a bijection π of the vertices of $G_1 = ([t \cdot n], E_1)$ to the vertices of $G_2 = ([t \cdot n], E_2)$ that minimizes the size of the set (of violations)

$$\{(u, v) \in [t \cdot n]^2 : \{u, v\} \in E_1 \text{ iff } \{\pi(u), \pi(v)\} \notin E_2\}. \tag{1}$$

(Note that Eq. (1) refers to ordered pairs, whereas the distance between graphs refers to unordered pairs.) Clearly, if π were to map to each cloud of G_2 only vertices that belong to a single cloud of G_1 (equiv., for every u, v that belong to the same cloud of G_1 it holds that $\pi(u), \pi(v)$ belong to the same cloud of G_2), then G_2 would be ϵ'-far from G_1 (since the fraction of violations under such a mapping equals the fraction of violations in the corresponding mapping of G'_1 to G'_2). The problem, however, is that it is not clear that π behaves in such a nice manner (and so violations under π do not directly translate to violations in mappings of G'_1 to G'_2). Still, we show that things cannot be extremely bad. Specifically, we call a cloud of G_2 good if at least $(t/2) + 1$ of its vertices are mapped to it (by π) from a single cloud of G_1.

Letting 2ϵ denote the fraction of violations in Eq. (1) (i.e., the size of this set divided by $(tn)^2$), we first show that at least $(1 - (6\epsilon/\alpha)) \cdot n$ of the clouds of G_2 are good. Assume, towards the contradiction, that G_2 contains more that $(6\epsilon/\alpha) \cdot n$ clouds that are not good. Considering any such a (non-good) cloud, we observe that it must contain at least $t/3$ disjoint pairs of vertices that originate in different clouds of G_1 (i.e., for each such pair (v, v') it holds that $\pi^{-1}(v)$ and $\pi^{-1}(v')$ belong to different clouds of G_1).[7] Recall that the edges in G_2 respect the cloud structure of G_2 (which in turn respects the edge relation of G'_2). But vertices that originate in different clouds of G_1 differ on at least $\alpha \cdot tn$ edges in G_1. Thus, every pair (v, v') (in this cloud of G_2) such that $\pi^{-1}(v)$ and $\pi^{-1}(v')$ belong to different clouds of G_1 contributes at least $\alpha \cdot tn$ violations to Eq. (1).[8] It follows that the set in Eq. (1) has size greater than

$$\frac{6\epsilon n}{\alpha} \cdot \frac{t}{3} \cdot \alpha tn \; = \; 2\epsilon \cdot (tn)^2$$

in contradiction to our hypothesis regarding π. Having established that at least $(1 - (6\epsilon/\alpha)) \cdot n$ of the clouds of G_2 are good and recalling that a good cloud of G_2 contains a strict majority of vertices that originates from a single cloud of G_1, we consider the following bijection π' of the vertices of G_1 to the vertices of G_2: For each good cloud g of G_2 that contains a strict majority of vertices from cloud i of G_1, we map all vertices of the i^{th} cloud of G_1 to cloud g of G_2, and map all other vertices of G_1 arbitrarily. The number of violations under π' is upper-bounded by four times the number of violations occuring under π between good clouds of G_2 (i.e., at most $4 \cdot 2\epsilon \cdot (tn)^2$) plus at most $(6\epsilon/\alpha) \cdot tn \cdot tn$ violations created with the remaining $(6\epsilon/\alpha) \cdot n$ clouds. This holds, in particular, for a bijection π' that maps to each remaining cloud of G_2 vertices originating in a single cloud of G_1. This π', which maps complete clouds of G_1 to clouds of G_2, yields a mapping of G'_1 to G'_2 that has at most $(8\epsilon + (6\epsilon/\alpha)) \cdot n^2$ violations. Recalling that G'_1 is ϵ'-far from G'_2, we conclude that $8\epsilon + (6\epsilon/\alpha) \geq 2\epsilon'$, and the claim follows (with $c = 1/7$). □

Recall that Claim 4.1 implies that if G' is ϵ'-far from Π', then its blow-up is $\Omega(\epsilon')$-far from Π. Using this fact, we conclude that ϵ'-testing of Π' reduces to $\Omega(\epsilon')$-testing of Π. Thus, a quadratic lower bound on the query complexity of ϵ'-testing Π'_n yields an $\Omega(n^2)$ lower bound on the query complexity of $\Omega(\epsilon')$-testing Π_N, where $n = \sqrt{q(N)}$. Thus, we obtain an $\Omega(q)$ lower bound on the query complexity of testing Π, for some constant value of the proximity parameter.

[7] This pairing is obtained by first clustering the vertices of the cloud of G_2 according to their origin in G_1. By the hypothesis, each cluster has size at most $t/2$. Next, observe that taking the union of some of these clusters yields a set containing between $t/3$ and $2t/3$ vertices. Finally, we pair vertices of this set with the remaining vertices. (A better bound of $\lfloor t/2 \rfloor$ can be obtained by using the fact that a t-vertex graph of minimum degree $t/2$ contains a Hamiltonian cycle.)

[8] For each such pair (v, v'), there exists at least $\alpha \cdot tn$ vertices u such that exactly one of the (unordered) pairs $\{\pi^{-1}(u), \pi^{-1}(v)\}$ and $\{\pi^{-1}(u), \pi^{-1}(v')\}$ is an edge in G_1. Recalling that for every u, the pair $\{u, v\}$ is an edge in G_2 if and only if $\{u, v'\}$ is an edge in G_2, it follows that for at least $\alpha \cdot tn$ vertices u either $(\pi^{-1}(u), \pi^{-1}(v))$ or $(\pi^{-1}(u), \pi^{-1}(v'))$ is a violation.

4.3 An Optimal Tester for Property Π

In this section we prove that the query complexity of testing Π is at most q (and that this can be met by a relatively efficient tester). We start by describing this (alleged) tester.

Algorithm 4.2. *On input N and proximity parameter ϵ, and when given oracle access to a graph $G = ([N], E)$, the algorithm proceeds as follows:*

1. *Setting $\epsilon' \stackrel{\text{def}}{=} \epsilon/3$ and computing $n \leftarrow \sqrt{q(N)}$.*
2. *Finding n representative vertices; that is, vertices that reside in different alleged clouds, which corresponds to the n vertices of the original graph. This is done by first selecting $s \stackrel{\text{def}}{=} O(\log n)$ random vertices, hereafter called the* signature vertices*, which will be used as a basis for clustering vertices* (according to their neighbors in the set of signature vertices)*. Next, we select $s' \stackrel{\text{def}}{=} O(\epsilon^{-2} \cdot n \log n)$ random vertices, probe all edges between these new vertices and the signature vertices, and cluster these s' vertices accordingly* (i.e., two vertices are placed in the same cluster if and only if they neighbor the same signature vertices)*. If the number of clusters is different from n, then we reject. Furthermore, if the number of vertices that reside in each cluster is not $(1 \pm \epsilon') \cdot s'/n$, then we also reject. Otherwise, we select* (arbitrarily) *a vertex from each cluster, and proceed to the next step.*
3. *Note that the signature vertices* (selected in Step 2) *induce a clustering of all the vertices of G. Referring to this clustering, we check that the edges between the clusters are consistent with the edges between the representatives. Specifically, we select uniformly $O(1/\epsilon)$ vertex pairs, cluster the vertices in each pair according to the signature vertices, and check that their edge relation agrees with that of their corresponding representatives. That is, for each pair (u, v), we first find the cluster to which each vertex belongs* (by making s adequate queries per each vertex)*, determine the corresponding representatives, denoted (r_u, r_v), and check* (by two queries) *whether $\{u, v\} \in E$ iff $\{r_u, r_v\} \in E$.* (Needless to say, if one of the newly selected vertices does not reside in any of the n existing clusters, then we reject.)
4. *Finally, using $\binom{n}{2} < q(N)/2$ queries, we determine the subgraph of G induced by the n representatives. We accept if and only if this induced subgraph is in Π'.*

Note that, for constant value of ϵ, the query complexity is dominated by Step 4, and is thus upper-bounded by $q(N)$. Furthermore, in this case, the above algorithm can be implemented in time $\text{poly}(n \cdot \log N) = \text{poly}(q(N) \cdot \log N)$. We comment that the Algorithm 4.2 is adaptive, and that a straightforward non-adaptive implementation of it has query complexity $O(n \log n)^2 = \widetilde{O}(q(N))$.

Remark 4.3. *In fact, a* (non-adaptive) *tester of query complexity $\widetilde{O}(q(N))$ can be obtained by a simpler algorithm that selects a random set of s' vertices and accepts if and only if the induced subgraph is ϵ'-close to being a (s'/n-factor) blow-up of some graph in Π'_n. Specifically, we can cluster these s' vertices by* *using them also in the role of the signature vertices. Furthermore, these vertices (or part of them) can also be designated for use in Step 3. We note that the analysis of this simpler algorithm does not rely on the hypothesis that Π' is dispersed.*

We now turn to analyzing the performance of Algorithm 4.2. We note that the proof that this algorithm accepts, with very high probability, any graph in Π_N relies on the hypothesis that Π' is dispersed.[9]

We first verify that any graph in Π_N is accepted with very high probability. Suppose that $G \in \Pi_N$ is a N/n-factor blow-up of $G' \in \Pi'_n$. Relying on the fact that Π' is dispersed we note that, for every pair of vertices in $G' \in \Pi'_n$, with constant probability a random vertex has a different edge relation to the members of this pair. Therefore, with very high (constant) probability, a random set of $s = O(\log n)$ vertices yields n different neighborhood patterns for the n vertices of G'. It follows that, with the same high probability, the s signature vertices selected in Step 2 induced n (equal sized) clusters on the vertices of G, where each cluster contains the cloud of N/n vertices (of G) that replaces a single vertex of G'. Thus, with very high (constant) probability, the sample of $s' = O(\epsilon^{-2} \cdot n \log n)$ additional vertices selected in Step 2 hits each of these clusters (equiv., clouds) and furthermore has $(1 \pm \epsilon') \cdot s'/n$ hits in each cluster. We conclude that, with very high (constant) probability, Algorithm 4.2 does not reject G in Step 2. Finally, assuming that Step 2 does not reject (and we did obtain representatives from each cloud of G), Algorithm 4.2 never rejects $G \in \Pi$ in Steps 3 and 4.

We now turn to the case that G is ϵ-far from Π_N, where we need to show that G is rejected with high constant probability (say, with probability 2/3). We will actually prove that if G is accepted with sufficiently high constant probability (say, with probability 1/3), then it is ϵ-close to Π_N. We call a set of s vertices good if (when used as the set of signature vertices) it induces a clustering of the vertices of G such that n of these clusters are each of size $(1 \pm 2\epsilon') \cdot N/n$. Note that good s-vertex sets must exist, because otherwise Algorithm 4.2 rejects in Step 2 with probability at least $1 - \exp(\Omega(\epsilon^2/n) \cdot s') > 2/3$. Fixing any good s-vertex set S, we call a sequence of n vertices $R = (r_1, ..., r_n)$ well-representing if (1) the subgraph of G induced by R is in Π'_n, and (2) at most ϵ' fraction of the vertex pairs of G have edge relation that is inconsistent with the corresponding vertices in R (i.e., at most ϵ' fraction of the vertex pairs in G violate the condition by which $\{u, v\} \in E$ if and only if $\{r_i, r_j\} \in E$, where u resides in the i^{th} cluster (w.r.t S) and v resides in the j^{th} cluster). Now, note that there must exist a good s-vertex set S that has a well-representing n-vertex sequence $R = (r_1, ..., r_n)$, because otherwise Algorithm 4.2 rejects with probability at least 2/3 (i.e., if a ρ fraction of the s-vertex sets are good (but have no corresponding n-sequence that is well-representing), then Step 2 rejects with probability at least $(1-\rho) \cdot 0.9$ and either Step 3 or Step 4 reject with probability $\rho \cdot \min((1 - (1-\epsilon')^{\Omega(1/\epsilon)}), 1))$.

[9] In contrast, the proof that Algorithm 4.2 rejects, with very high probability, any graph that is ϵ-far from Π_N does not rely on this hypothesis.

Fixing any good s-vertex set S and any corresponding $R = (r_1, ..., r_n)$ that is well-representing, we consider the clustering induced by S, denoted $(C_1, ..., C_n, X)$, where X denotes the set of (untypical) vertices that do not belong to the n first clusters. Recall that, for every $i \in [n]$, it holds that $r_i \in C_i$ and $|C_i| = (1 \pm 2\epsilon') \cdot N/n$. Furthermore, denoting by $i(v)$ the index of the cluster to which vertex $v \in [N] \setminus X$ belongs, it holds that the number of pairs $\{u, v\}$ (from $[N] \setminus X$) that violate the condition $\{u, v\} \in E$ iff $\{r_{i(u)}, r_{i(v)}\} \in E$ is at most $\epsilon' \cdot \binom{N}{2}$. Now, observe that by modifying at most $\epsilon' \cdot \binom{N}{2}$ edges in G we can eliminate all the aforementioned violations, which means that we obtain n sets with edge relations that fit some graph in Π'_n (indeed the graph obtained as the subgraph of G induced by R, which was not modified). Recall that these sets are each of size $(1 \pm 2\epsilon') \cdot N/n$, and so we may need to move $2\epsilon' N$ vertices in order to obtain sets of size N/n. This movement may create up to $2\epsilon' N \cdot (N-1)$ new violations, which can be eliminated by modifying at most $2\epsilon' \cdot \binom{N}{2}$ additional edges in G. Using $\epsilon = 3\epsilon'$, we conclude that G is ϵ-close to Π_N.

5 Revisiting the Adj. Matrix Model: Monotonicity

In continuation to Section 4, which provides a hierarchy theorem for generic graph properties (in the adjacency matrix model), we present in this section a hierarchy theorem for *monotone* graph properties (in the same model). We say that a graph property Π is monotone if adding edges to any graph that resides in Π yields a graph that also resides in Π. (That is, we actually refer to upward monotonicity, and an identical result for downward monotonicity follows by considering the complement graphs.)[10]

Theorem 5. *In the adjacency matrix model, for every $q : \mathbb{N} \to \mathbb{N}$ that is at most quadratic, there exists a* monotone *graph property Π that is testable in $O(q)$ queries, but is not testable in $o(q)$ queries.*

Note that Theorem 5 refers to two-sided error testing (just like Theorem 4). Theorems 4 and 5 are incomparable: the former provides graph properties that are in $\mathcal{P}$ (and the upper bound is established via relatively efficient testers), whereas the latter provides graph properties that are monotone.

Outline of the proof of Theorem 5. Starting with the proof of Theorem 4, one may want to apply a monotone closure to the graph property Π (presented in the proof of Theorem 4).[11] Under suitable tuning of parameters, this allows to retain the proof of the lower bound, but the problem is that the tester presented for the upper bound fails. The point is that this tester relies on the structure of graphs obtained via blow-up, whereas this structure is not maintained by

[10] We stress that these notions of monotonicity are different from the notion of monotonicity considered in [AS], where a graph property Π is called monotone if any subgraph of a graph in Π is also in Π.

[11] Indeed, this is the approach used in the proof of [GT, Thm. 1].

the monotone closure. One possible solution, which assumes that all graphs in Π have approximately the same number of edges, is to augment the monotone closure of Π with all graphs that have significantly more edges, where the corresponding threshold (on the number of edges) is denoted T. Intuitively, this way, we can afford accepting any graph that has more than T edges, and handle graphs with fewer edges by relying on the fact that in this case the blow-up structure is essentially maintained (because only few edges are added). Unfortunately, implementing this idea is not straightforward: On one hand, we should set the threshold high enough so that the lower bound proof still holds, whereas on the other hand such a setting may destroy the local structure of a constant fraction of the graph's vertices. The solution to this problem is to use an underlying property Π' that supports "error correction" (i.e., allows recovering the original structure even when a constant fraction of it is destroyed as above). (The actual proof of Theorem 5 is given in our technical report [GKNR].)

6 Revisiting the Adj. Matrix Model: One-Sided Error

In continuation to Section 4, which provides a hierarchy theorem for two-sided error testing of graph properties (in the adjacency matrix model), we present in this section a hierarchy theorem that refers to one-sided error testing. Actually, the lower bounds will hold also with respect to two-sided error, but the upper bounds will be established using a tester of one-sided error.

Theorem 6. *In the adjacency matrix model, for every $q : \mathbb{N} \to \mathbb{N}$ that is at most quadratic, there exists a graph property Π that is testable with one-sided error in $O(q)$ queries, but is not testable in $o(q)$ queries even when allowing two-sided error. Furthermore, Π is in $\mathcal{P}$.*

Theorems 4 and 6 are incomparable: in the former the upper bound is established via relatively efficient testers (of two-sided error), whereas in the latter the upper bound is established via one-sided error testers (which are not relatively efficient). (Unlike Theorem 5, both Theorems 4 and 6 do not provide monotone properties.)

Outline of the proof of Theorem 6. Starting with the proof of Theorem 4, we observe that the source of the two-sided error of the tester is in the need to approximate set sizes. This is unavoidable when considering graph properties that are blow-ups of some other graph properties, where blow-up is defined by replacing vertices of the original graph by *equal-size* clouds. The natural solution is to consider a *generalized* notion of blow-up in which each vertex is replaced by a (non-empty) cloud of arbitrary size. That is, G is a (generalized) blow-up of $G' = ([n], E')$ if the vertex set of G can be partitioned into n non-empty sets (of arbitrary sizes) that correspond to the n vertices of G' such that the edges between these sets represent the edges of G'; that is, if $\{i, j\}$ is an edge in G' (i.e., $\{i, j\} \in E'$), then there is a complete bipartite between the i^{th} set and the

$j^{\rm th}$ set, and otherwise (i.e., $\{i,j\} \notin E'$) there are no edges between this pair of sets.

The actual proof of Theorem 6 is given in our technical report [GKNR]. Among other things, this proof copes with the non-trivial question of how does the *generalized* (rather than the standard) blow-up operation affect distances between graphs.

7 Concluding Comments

Theorems 4, 5 and 6 (and their proofs) raise several natural open problems, listed next. We stress that all questions refer to the adjacency matrix graph model considered in Sections 4–6.

1. *Preservation of distance between graphs under blow-up*: Recall that the proof of Theorem 4 relies on the preservation of distances between graphs under the blow-up operation. The partial results (regarding this matter) obtained in this work suffice for the proof of Theorem 4, but the problem seems natural and of independent interest.

   Recall that Claim 4.1 asserts that in some cases the distance between two unlabeled graphs is preserved *up to a constant factor* by any blow-up (i.e., "linear preservation"), whereas Theorem 8 of our technical report [GKNR] asserts a quadratic preservation for any pair of graphs. Also recall that it is not true that the distance between any two unlabeled graphs is *perfectly preserved* by any blow-up (see beginning of Appendix B in our technical report [GKNR]).

   In earlier versions of this work we raised the natural question of whether the distance between any two unlabeled graphs is preserved up to a constant factor by any blow-up. This question has been recently resolved by Oleg Pikhurko, who showed that *the distance is indeed preserved up to a factor of three* [P, Sec. 4]. Note that Arie Matsliah's counterexample to perfect preservation (presented in Appendix B of our technical report [GKNR]) shows that the said constant factor cannot be smaller than 6/5. Indeed, determining the true constant factor remains an open problem.
2. *Combining the features of all three hierarchy theorems*: Theorems 4, 5 and 6 provide incomparable hierarchy theorems, each having an additional feature that the others lack. Specifically, Theorem 4 refers to properties in $\mathcal{P}$ (and testing, in the positive part, is relatively efficient), Theorem 5 refers to monotone properties, and Theorem 6 provides one-sided testing (in the positive part). Is it possible to have a single hierarchy theorem that enjoys all three additional feature? Intermediate goals include the following:

   (a) *Hierarchy of monotone graph properties in $\mathcal{P}$*: Recall that Theorem 4 is proved by using non-monotone graph properties (which are in $\mathcal{P}$), while Theorem 5 refers to monotone graph properties that are not likely to be in $\mathcal{P}$. Can one combine the good aspects of both results?

(b) *Hard-to-test monotone graph property in $\mathcal{P}$*: Indeed, before addressing Problem 2a, one should ask whether a result analogous to Theorem 7 of our technical report [GKNR] holds for a monotone graph property? Recall that [GT, Thm. 1] provides a monotone graph property in $\mathcal{NP}$ that is hard-to-test.

(c) *One-sided versus two-sided error testers*: Recall that the positive part of Theorem 6 refers to testing with one-sided error, but these testers are not relatively efficient. In contrast, the positive part of Theorem 4 provides relatively efficient testers, but these testers have two-sided error. Can one combine the good aspects of both results?

Acknowledgments

We are grateful to Ronitt Rubinfeld for asking about the existence of hierarchy theorems for the adjacency matrix model. Ronitt raised this question during a discussion that took place at the Dagstuhl 2008 workshop on sub-linear algorithms. We are also grateful to Arie Matsliah and Yoav Tzur for helpful discussions. In particular, we thank Arie Matsliah for providing us with a proof that the blow-up operation does not preserve distances in a perfect manner. Oded Goldreich was partially supported by the Israel Science Foundation (grant No. 1041/08). Michael Krivelevich was partially supported by a USA-Israel BSF Grant, by a grant from the Israel Science Foundation, and by Pazy Memorial Award.

References

[ABI] Alon, N., Babai, L., Itai, A.: A fast and Simple Randomized Algorithm for the Maximal Independent Set Problem. J. of Algorithms 7, 567–583 (1986)

[AFKS] Alon, N., Fischer, E., Krivelevich, M., Szegedy, M.: Efficient Testing of Large Graphs. Combinatorica 20, 451–476 (2000)

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

[AGHP] Alon, N., Goldreich, O., Hastad, J., Peralta, R.: Simple constructions of almost k-wise independent random variables. Journal of Random structures and Algorithms 3(3), 289–304 (1992)

[AS] Alon, N., Shapira, A.: Every Monotone Graph Property is Testable. SIAM Journal on Computing 38, 505–522 (2008)

[BSS] Benjamini, I., Schramm, O., Shapira, A.: Every Minor-Closed Property of Sparse Graphs is Testable. In: 40th STOC, pp. 393–402 (2008)

[BLR] Blum, M., Luby, M., Rubinfeld, R.: Self-Testing/Correcting with Applications to Numerical Problems. JCSS 47(3), 549–595 (1993)

[BHR] Ben-Sasson, E., Harsha, P., Raskhodnikova, S.: 3CNF Properties Are Hard to Test. SIAM Journal on Computing 35(1), 1–21 (2005)

[BOT] Bogdanov, A., Obata, K., Trevisan, L.: A lower bound for testing 3-colorability in bounded-degree graphs. In: 43rd FOCS, pp. 93–102 (2002)

[EKK+] Ergun, F., Kannan, S., Kumar, S.R., Rubinfeld, R., Viswanathan, M.: Spot-checkers. JCSS 60(3), 717–751 (2000)

[F] Fischer, E.: The art of uninformed decisions: A primer to property testing. Bulletin of the European Association for Theoretical Computer Science 75, 97–126 (2001)

[FM] Fischer, E., Matsliah, A.: Testing Graph Isomorphism. In: 17th SODA, pp. 299–308 (2006)

[GGL+] Goldreich, O., Goldwasser, S., Lehman, E., Ron, D., Samorodnitsky, A.: Testing Monotonicity. Combinatorica 20(3), 301–337 (2000)

[GGR] Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. Journal of the ACM, 653–750 (July 1998)

[GKNR] Goldreich, O., Krivelevich, M., Newman, I., Rozenberg, E.: Hierarchy Theorems for Property Testing. ECCC, TR08-097 (2008)

[GR1] Goldreich, O., Ron, D.: Property Testing in Bounded Degree Graphs. Algorithmica 32(2), 302–343 (2002)

[GR2] Goldreich, O., Ron, D.: A Sublinear Bipartitness Tester for Bounded Degree Graphs. Combinatorica 19(3), 335–373 (1999)

[GT] Goldreich, O., Trevisan, L.: Three theorems regarding testing graph properties. Random Structures and Algorithms 23(1), 23–57 (2003)

[LNS] Lachish, O., Newman, I., Shapira, A.: Space Complexity vs. Query Complexity. Computational Complexity 17, 70–93 (2008)

[NN] Naor, J., Naor, M.: Small-bias Probability Spaces: Efficient Constructions and Applications. SIAM J. on Computing 22, 838–856 (1993)

[PRR] Parnas, M., Ron, D., Rubinfeld, R.: Testing Membership in Parenthesis Laguages. Random Structures and Algorithms 22(1), 98–138 (2003)

[P] Pikhurko, O.: An Analytic Approach to Stability (2009), `http://arxiv.org/abs/0812.0214`

[R] Ron, D.: Property testing. In: Rajasekaran, S., Pardalos, P.M., Reif, J.H., Rolim, J.D.P. (eds.) Handbook on Randomization, vol. II, pp. 597–649 (2001)

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

[S] Shaltiel, R.: Recent Developments in Explicit Constructions of Extractors. In: Current Trends in Theoretical Computer Science: The Challenge of the New Century. Algorithms and Complexity, vol. 1, pp. 67–95. World scientific, Singapore (2004); Preliminary version in Bulletin of the EATCS, vol. 77, pp. 67–95 (2002)