# A Separation Theorem in Property Testing

Noga Alon [*] Asaf Shapira [†]

**Abstract**

Consider the following seemingly rhetorical question: Is it crucial for a property-tester to know the error parameter $\epsilon$ in advance? Previous papers dealing with various testing problems, suggest that the answer may be no, as in these papers there was no loss of generality in assuming that $\epsilon$ is given as part of the input, and is not known in advance. Our main result in this paper, however, is that it is possible to separate a natural model of property testing in which $\epsilon$ is given as part of the input from the model in which $\epsilon$ is known in advance (without making any computational hardness assumptions). To this end, we construct a graph property $\mathcal{P}$, which has the following two properties:

(i) There is no tester for $\mathcal{P}$ accepting $\epsilon$ as part of the input, whose number of queries depends only on $\epsilon$.

(ii) For any fixed $\epsilon$, there is a tester for $\mathcal{P}$ (that works only for that specific $\epsilon$), which makes a constant number of queries.

Interestingly, we manage to construct a separating property $\mathcal{P}$, which is combinatorially natural as it can be expressed in terms of forbidden subgraphs and also computationally natural as it can be shown to belong to *coNP*.

The main tools in this paper are efficiently constructible graphs of high girth and high chromatic number, a result about testing monotone graph properties, as well as basic ideas from the theory of recursive functions. Of independent interest is a *precise characterization* of the monotone graph properties that can be tested with $\epsilon$ being part of the input, which we obtain as one of the main steps of the paper.

## 1 Introduction

### 1.1 Definitions and Background

We start with definitions related to the area of property-testing. Let $\mathcal{P}$ be a property of graphs, namely, a family of graphs closed under isomorphism. All graph properties discussed in this paper are assumed to be decidable, that is, we disregard properties for which it is not possible to tell whether a given graph satisfies them, even if one has complete knowledge of the graph. In this paper

[*]Schools of Mathematics and Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. Email: nogaa@tau.ac.il. Research supported in part by a USA Israeli BSF grant, by a grant from the Israel Science Foundation, and by the Hermann Minkowski Minerva Center for Geometry at Tel Aviv University.

[†]School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. Email: asafico@tau.ac.il. This work forms part of the author's Ph.D. thesis. Research supported by a Charles Clore Foundation Fellowship and by an IBM Ph.D. Fellowship.

we focus on property-testing in the dense-graph model as defined in [12]. In this model a graph $G$ with $n$ vertices is said to be *$\epsilon$-far from satisfying $\mathcal{P}$* if one must add or delete at least $\epsilon n^2$ edges in order to turn $G$ into a graph satisfying $\mathcal{P}$. A *tester* for $\mathcal{P}$ is a randomized algorithm which, given the size of the input $n$, and the ability to query, whether a desired pair of vertices of the input are adjacent or not, distinguishes with high probability (say, 2/3), between the case of $G$ satisfying $\mathcal{P}$ and the case of $G$ being $\epsilon$-far from satisfying $\mathcal{P}$. The following notion of efficient testing is the main focus of property-testing in the dense graph model:

**Definition 1.1 (Testable)** *A graph property $\mathcal{P}$ is* testable*, if there is a tester for $\mathcal{P}$ whose query complexity $q(\epsilon, n)$ can be bounded by a function $Q(\epsilon)$, which is independent of the size of the input.*

We stress that the definition of a tester for a testable property allows the query complexity to depend on $n$. It just requires that it will be possible to *upper bound* $q(\epsilon, n)$ by some function $Q(\epsilon)$. Therefore, for example $q(\epsilon, n) = 1/\epsilon + (-1)^n$ is a legitimate query complexity as is can be bounded by $Q(\epsilon) = 1/\epsilon + 1$. As we will see later, in some cases the distinction between query complexity depending only on $\epsilon$ and query complexity bounded by a function of $\epsilon$ may have interesting and non-trivial applications.

One of the most interesting phenomena in the area of property-testing in the dense graph model is that many natural graph properties such as $k$-colorability and having a large cut are testable (see [12]). A general tester may err in both directions and is thus said to have *two-sided error.* A tester is said to have *one-sided error* if whenever $G$ satisfies $\mathcal{P}$, the algorithm declares that this is the case with probability 1. The general notion of property testing was first formulated by Rubinfeld and Sudan [22], who were motivated mainly by its connection to the study of program checking and by Blum, Luby and Rubinfeld [7], who were mainly interested in testing properties of functions, such as linearity. The study of the notion of testability for combinatorial structures, and mainly for labelled graphs, was introduced in the seminal paper of Goldreich, Goldwasser and Ron [12], who showed that several natural graph properties are testable. In the wake of [12], many other graph properties were shown to be testable, while others were shown to be non-testable. See [3] for a recent combinatorial characterization of the testable graph properties, and [10, 11, 21] for additional results and references on graph property-testing as well as on testing properties of other combinatorial structures.

One of the fundamental problems of complexity theory is in understanding the relations between various models of computation. In particular, one would like to know if two models are equivalent or if there are problems, which can be solved in one model but not in the other. Regretfully, in many cases, though it seems obvious that two models of computation are not equivalent, the current techniques are far from enabling one to formally prove that. In this paper we introduce two natural and realistic models of property-testing [1]. Surprisingly, in our case, though it seems at first that these models are equivalent, we manage to formally prove that they are in fact distinct. We also show that in some cases a tester can make a non-trivial usage of both the error parameter $\epsilon$ and the size of the input graph $n$, when making its decisions.

## 1.2 The Main Result: Uniform vs. Non-Uniform Property Testing

As we have mentioned in the previous subsection, in this paper we focus on the dense-graph model. The main goal in the dense graph model is to design a tester, whose query complexity can be bounded

[1]These models are natural and realistic in the sense that they capture all the previous results on testing graph properties. See Subsection 1.2.

by a function, which is independent of the size of the input graph, and thus establish that a certain property is testable. In defining a tester in the previous subsection we have allowed the tester to use the size of the input in order to make its decisions. We now define a slightly weaker notion of a tester, which is not allowed to use the size of the input graph in order to make its decisions.

**Definition 1.2 (Oblivious Tester)** *A tester (one-sided or two-sided) is said to be* oblivious *if it works as follows: given $\epsilon$ the tester computes an integer $Q = Q(\epsilon)$ and asks an oracle for a subgraph induced by a set of vertices $S$ of size $Q$, where the oracle chooses $S$ randomly and uniformly from the vertices of the input graph. If $Q$ is larger than the size of the input graph then the oracle returns the entire graph. The tester then accepts or rejects (possibly randomly) according to $\epsilon$ and the graph induced by $S$.*

The notion of an oblivious tester was defined in [5]. As was discussed in [5], there are no natural graph properties for which even oblivious testers are weaker than standard testers that are given access to $n$ (= size of the input). The intuition is that as the query complexity is independent of $n$, the algorithm does not need to know the value of $n$. Indeed, essentially all the testers that have been designed in the literature were in fact oblivious, or could trivially be turned into oblivious testers. For example, $k$-colorability (see [12]) can be tested by an oblivious tester that samples a set of vertices $S$ and accepts if and only if the graph spanned by $S$ is $k$-colorable. In fact, any hereditary property [2] can be tested in a similar way, see [5, 17]. As another example, the property of having a cut of size, say, at least $\frac{1}{8}n^2$ (see [12]), can be tested by an oblivious tester that samples a set of vertices $S$ and accepts if and only if the graph spanned by $S$ has a cut of size at least $(\frac{1}{8} - \frac{\epsilon}{2})|S|^2$. Note, that the property of having a large cut can be tested by an oblivious tester although the definition of the property involves the size of the graph. The notion of an oblivious tester was used in [5] in order to derive a characterization of the "natural" graph properties that are testable with one-sided error. Of course, some properties cannot be tested by an oblivious tester. In particular, properties whose definition involves the size of the input cannot be so tested. An example is the property $\tilde{\mathcal{P}}$ of being bipartite if the number of vertices is even, and being triangle-free if the number of vertices is odd. However, these properties are not natural. Informally, the notion of an oblivious tester means that the size of the input is not an important resource when studying property testing of "natural" graph properties in the dense graph model, such as hereditary properties, whose definition is independent of the input size. We stress that as opposed to the dense graph model, property-testing in the bounded degree model [13] and the general density model [20, 15], usually requires query complexity, which depends on the size of the input graph. Therefore, the notion of oblivious testing is not adequate for those models.

The main resource that we seek to study in this paper, is the value of the error parameter $\epsilon$. In defining a tester in the previous subsection, we did not mention whether the error parameter $\epsilon$ is given as part of the input, or whether the tester is designed to distinguish between graphs that satisfy $\mathcal{P}$ from those that are $\epsilon$-far from satisfying it, when $\epsilon$ is a known fixed constant. The current literature about property testing is not clear about this issue as in some papers $\epsilon$ is assumed to be a part of the input while in others it is not. We thus introduce the following two definitions:

**Definition 1.3 (Uniformly testable)** *A graph property $\mathcal{P}$ is* uniformly testable *if it can be tested by an oblivious tester as in Definition 1.2. Note that such a tester accepts $\epsilon$ as part of the input.*

[2]A graph property is hereditary if it is closed under removal of vertices.

**Definition 1.4 (Non-uniformly testable)** *A graph property $\mathcal{P}$ is* non-uniformly testable *if for every $\epsilon$ there is a tester $T_\epsilon$ and an integer $Q = Q(\epsilon)$ with the following properties: $T_\epsilon$ is an algorithm for distinguishing between graphs satisfying $\mathcal{P}$ from those that are $\epsilon$-far from satisfying it, that operates like an oblivious tester with the restriction that it asks the oracle for a subgraph induced on a set of vertices of size at most $Q$.*

Note, that in Definition 1.4 a tester $T_\epsilon$ does not receive $\epsilon$ as part of the input. Note also that we can think of a tester $T_\epsilon$ as a uniform tester, where the tester "knows" the quantity $Q(\epsilon)$ in advance and does not have to compute it. For this reason it is clear that if $\mathcal{P}$ is uniformly testable then it is also non-uniformly testable: For every $\epsilon$ define $T_\epsilon$ to perform like the uniform tester for $\mathcal{P}$, while setting $Q = Q(\epsilon)$. As in the definition of a tester in Subsection 1.1, we generally allow a property to be uniformly (resp. non-uniformly) tested with two-sided error. If a property is uniformly (resp. non-uniformly) testable in a way that graphs satisfying the property are always accepted then the property is said to be uniformly (resp. non-uniformly) testable with one-sided error.

We believe that the distinction between uniform and non-uniform testing was not previously introduced in the literature because all the testable graph (and non graph) properties that were previously studied were in fact uniformly testable. As we have mentioned above, any property that is uniformly testable is also non-uniformly testable. It may thus seem, at least at first glance, that uniformly and non-uniformly property-testing are identical notions. However, the problem with trying to simulate a non-uniform tester using a uniform one is that computing the query-complexity, $Q(\epsilon)$, may be non-recursive. Our main result in this paper is that when considering oblivious testers these two notions are in fact distinct. Moreover, these notions can be shown to be distinct while confining ourselves to graph properties, which are natural with respect to both their combinatorial structure and their computational difficulty.

**Theorem 1 (The Main Result)** *There is a graph property $\mathcal{P}$ with the following properties:*

1. *$\mathcal{P}$ can be non-uniformly tested with one-sided error.*

2. *$\mathcal{P}$ cannot be uniformly tested, even with two-sided error.*

*Moreover, satisfying $\mathcal{P}$ belongs to coNP and can be expressed in terms of forbidden subgraphs.*

For a family of graphs $\mathcal{F}$ we define the property of being $\mathcal{F}$-free as the property of not containing a copy of any graph $F \in \mathcal{F}$ as a (not necessarily induced) subgraph. The property $\mathcal{P}$, which we construct in order to prove Theorem 1, is simply the property of being $\mathcal{F}$-free for some carefully defined family of graphs $\mathcal{F}$.

The reader should note that the difference between being uniformly testable and non-uniformly testable, is not as sharp as, say, the difference between $P$ and $P/Poly$. The reason is that in $P/Poly$ the non-uniformity is with respect to the *inputs*, while in our case the non-uniformity is over the *error parameter*. In particular, a non-uniform tester $T_\epsilon$ should be able to handle *any* input graph. We note that it is possible to prove Theorem 1 by defining an undecidable graph property that can be non-uniformly tested with one-sided error, but obviously cannot be uniformly tested (because by setting $\epsilon = 1/n^2$ we precisely solve the undecidable problem). Theorem 1 however, gives a *natural* separation of these two models of property-testing by defining a decidable and combinatorially natural graph property, which satisfies the assertions of Theorem 1. We note that the main focus of property testing is in solving problems using the smallest possible amount of information about the input. Hence, undecidable properties are particulary unnatural in the context of property testing as such properties are not solvable/testable even if one has complete knowledge of the input.

### 1.3 Separations in Other Models of Property Testing

Is is natural to ask if it is possible to prove versions of Theorem 1 for other models of property testing. In particular, one can ask if such a separation can be proved for the general model of property testing, where the tester can use the size of the graph in order to determine its query complexity (which should still be bounded by a function of $\epsilon$). As it turns out such a separation is not possible.

**Proposition 1.5** *The following holds in the general model of property-testing (as defined in Subsection 1.1), where the tester* can *use the size of the input: a property $\mathcal{P}$ is testable when $\epsilon$ is known in advance, if and only if it is testable when $\epsilon$ is given as part of the input.*

Theorem 1 asserts that in the oblivious model, there are properties that can be tested by a tester if it knows $\epsilon$ in advance, which cannot be tested if $\epsilon$ is part of the input. In other words, it asserts that there are non-trivial computations the tester may perform with the error parameter $\epsilon$. Therefore, Proposition 1.5 can be interpreted as asserting that knowing the size of the input can help a tester in a non-trivial way. More precisely, it shows that in some cases it is possible for the tester to compute the query complexity with the aid of the size of the input, while by Theorem 1 it is impossible to do so without this information. The main point is that given the size of the input graph, a tester can "search" for the optimal query-complexity for graphs of that size. See Subsection 5.2 for the proof.

We note that Proposition 1.5 has some additional interesting implications. Its proof gives a non-trivial example, where the query complexity of a tester can be bounded by a function of $\epsilon$ only (as Definition 1.1 requires), while at the same time the query complexity depends on the size of the graph. It also gives a non-trivial example, where though the query complexity of a tester can be bounded by a function of $\epsilon$ only, the running time of the tester depends (exponentially!) on the size of the input. See Subsection 5.2 for the full details.

## 2 Overview of the Proof

We start with a short introduction of the main result of [4], which will be a key tool in this paper. Throughout the paper we will make an extensive use of the notion of graph homomorphism, which we turn to formally define.

**Definition 2.1 (Homomorphism)** *A* homomorphism *from a graph $F$ to a graph $K$, is a mapping $\varphi : V(F) \mapsto V(K)$ that maps edges to edges, namely $(v,u) \in E(F)$ implies $(\varphi(v), \varphi(u)) \in E(K)$.*

In the rest of the paper, $F \mapsto K$ will denote that there is a homomorphism from $F$ to $K$, and $F \not\mapsto K$ will denote that no such homomorphism exists. Just to practice the definition, note that if $F \mapsto K$ then $\chi(F) \leq \chi(K)$. In particular, this means that a graph $G$ has a homomorphism into a clique of size $k$ if and only if $G$ is $k$-colorable. A key ingredient in the main result of [4] as well as in this paper, is a certain graph theoretic functional, defined below.

**Definition 2.2 (The function $\Psi_{\mathcal{F}}$)** *For any (possibly infinite) family of graphs $\mathcal{F}$, and any integer $k$ let $\mathcal{F}_k$ be the following set of graphs: A graph $R$ belongs to $\mathcal{F}_k$ if it has at most $k$ vertices and there is at least one $F \in \mathcal{F}$ such that $F \mapsto R$. For any such family $\mathcal{F}$ and integer $k$, for which $\mathcal{F}_k \neq \emptyset$, let*

$$\Psi_{\mathcal{F}}(k) = \max_{R \in \mathcal{F}_k} \min_{\{F \in \mathcal{F} : F \mapsto R\}} |V(F)| \,. \tag{1}$$

*Furthermore, in case $\mathcal{F}_k = \emptyset$, define $\Psi_{\mathcal{F}}(k) = 0$.*

Practicing definitions again, note that if $\mathcal{F}$ is the family of odd cycles, then $\mathcal{F}_k$ is precisely the family of non-bipartite graphs of size at most $k$. Also, in this case $\Psi_{\mathcal{F}}(k) = k$ when $k$ is odd, and $\Psi_{\mathcal{F}}(k) = k-1$ when $k$ is even. The "right" way to think of $\Psi_{\mathcal{F}}$ is the following: Let $R$ be a graph of size at most $k$ and suppose we are guaranteed that there is a graph $F' \in \mathcal{F}$ such that $F' \mapsto R$ (thus $R \in \mathcal{F}_k$). Then by this information only and *without* having to know the structure of $R$ itself, the definition of $\Psi_{\mathcal{F}}$ implies that there is a graph $F \in \mathcal{F}$ of size at most $\Psi_{\mathcal{F}}(k)$, such that $F \mapsto R$.

As it turns out, $\Psi_{\mathcal{F}}(k)$, which seems to have little, if anything, to do with property testing, is in fact crucial to testing monotone graph properties [3]. Call a function *recursive* if there is an algorithm for computing it in finite time (see [19]). The first connection between $\Psi_{\mathcal{F}}(k)$ and testing monotone graph properties is part of the main result of [4], which can be formulated as follows.

**Theorem 2 ([4])** *For every (possibly infinite) family of graphs $\mathcal{F}$, the property of being $\mathcal{F}$-free is non-uniformly testable with one-sided error. Moreover, if $\Psi_{\mathcal{F}}$ is recursive then being $\mathcal{F}$-free is also uniformly testable with one-sided error.*

**Comment 2.3** *We remind the reader that we consider only decidable graph properties. Hence, in the above theorem we assume that being $\mathcal{F}$-free is a decidable property.*

**Comment 2.4** *The reader should note that Theorem 2 immediately applies also to any monotone property $\mathcal{P}$. The reason is that given $\mathcal{P}$ we can define $\mathcal{F} = \mathcal{F}_{\mathcal{P}}$ to be the set of graphs, which are minimal with respect to not satisfying the property $\mathcal{P}$. For example, if $\mathcal{P}$ is the property of being bipartite then $\mathcal{F}_{\mathcal{P}}$ is the (infinite) family of odd cycles. It is clear that satisfying $\mathcal{P}$ is equivalent to being $\mathcal{F}$-free. For convenience and ease of notation, in this paper we describe monotone properties via their family of forbidden subgraphs.*

The proof of Theorem 1 consists of two steps. In the first step we prove the somewhat surprising fact, that $\Psi_{\mathcal{F}}(k)$ being recursive is not only sufficient for inferring that being $\mathcal{F}$-free is uniformly testable (this is guaranteed by Theorem 2), but this condition is also necessary. This is formulated in the following Theorem.

**Theorem 3** *Suppose $\mathcal{F}$ is a family of graphs for which the function $\Psi_{\mathcal{F}}$ is not recursive. Then, the property of being $\mathcal{F}$-free cannot be uniformly tested with one-sided error.*

Note, that in Definition 1.3 the tester is defined as one that may have arbitrarily large query complexity, as long as it can be bounded by a (recursive) function of $\epsilon$ only. Hence, in the case that

[3]A graph property is monotone if it is closed under removal of both vertices and edges. Standard examples are $k$-colorability and triangle-freeness

$\Psi_{\mathcal{F}}$ is not recursive, Theorem 3 rules out the possibility of designing a uniform tester with arbitrarily large query complexity that can be bounded by a function of $\epsilon$ only.

The main idea behind the proof of Theorem 3 is that by "inspecting" the behavior of a property tester for the property of being $\mathcal{F}$-free one can compute the function $\Psi_{\mathcal{F}}$. The main combinatorial tool in the proof of Theorem 3 is a Theorem of Erdős [9] in extremal graph theory, which can be considered as a hypergraph version of the Zarankiewicz problem [16]. As an immediate corollary of Theorems 2 and 3 we obtain the following result, which *precisely* characterizes the families of graphs $\mathcal{F}$, for which the property of being $\mathcal{F}$-free can be tested uniformly (recall that by Theorem 2, for *any* family $\mathcal{F}$, being $\mathcal{F}$-free is non-uniformly testable). By Comment 2.4, this also gives a *precise* characterization of the monotone graph properties that are uniformly testable.

**Corollary 2.5** *For every family of graphs $\mathcal{F}$, the property of being $\mathcal{F}$-free is uniformly testable with one-sided error if and only if the function $\Psi_{\mathcal{F}}$ is recursive.*

An immediate consequence of Theorems 2 and 3 is that in order to separate uniform testing with one-sided error from non-uniform testing with one-sided error, and thus (almost) prove Theorem 1, it is enough to construct a family of graphs $\mathcal{F}$ with the following two properties: (i) There is an algorithm for deciding whether a graph $F$ belongs to $\mathcal{F}$ (recall that we confine ourselves to decidable graph properties). (ii) The function $\Psi_{\mathcal{F}}$ is non-recursive. The main combinatorial ingredient in the construction of $\mathcal{F}$ is the fundamental theorem of Erdős [8] in extremal graph theory, which guarantees the existence of graphs with arbitrarily large girth and chromatic number. As we want to prove Theorem 1 with a graph property, which is not only decidable, but even belongs to *coNP*, we need explicit constructions of such graphs. To this end, we use explicit constructions of expanders, which are given in [18]. For the construction we also apply some ideas from the theory of recursive functions. Finally, in order to obtain that being $\mathcal{F}$-free cannot be tested even with two-sided error, we use a result of the first author ([14] Appendix D) about testing hereditary graph properties.

**Organization:** The proof of Theorem 3 appears in Section 3, and the proof of Theorem 1 appears in Section 4. Section 5 contains concluding remarks. The proof of Proposition 1.5 appears in Subsection 5.2.

## 3 Computing $\Psi_{\mathcal{F}}$ via Testing $\mathcal{F}$-freeness

In this section we describe the proof of Theorem 3. Recall that $F \mapsto K$ denotes the fact that there is a homomorphism from $F$ to $K$ (see Definition 2.1). In what follows, an $s$-blowup of a graph $K$ is the graph obtained from $K$ by replacing every vertex $v_i \in V(K)$ with an independent set $I_i$, of size $s$, and replacing every edge $(v_i, v_j) \in E(K)$ with a complete bipartite graph, whose partition classes are $I_i$ and $I_j$. It is easy to see that a blowup of $K$ is far from being $K$-free (recall that $K$-free is the property of not containing a copy of $K$). It is also easy to see that if $F \mapsto K$, then a blowup of $K$ is far from being $F$-free (see Lemma 3.3 in [1]). However, in this case the distance of the blowup from being $F$-free is a function of the size of $F$. As it turns out, for the proof of Theorem 3, we need a stronger assertion where the distance is only a function of $k = |V(K)|$. This stronger assertion is guaranteed by Lemma 3.1 below, whose proof relies on the following theorem of Erdős [9], which is a hypergraph extension of Zarankiewicz's problem [16].

**Theorem 4 ([9])** *For every integer $f$ there is an integer $N = N(f)$ with the following property: Every $k$-uniform hypergraph on $n > N$ vertices that contains at least $n^{k-f^{1-k}}$ edges, contains a copy of $K_f^k$, which is the complete $k$-partite $k$-uniform hypergraph, where each partition class is of size $f$.*

**Lemma 3.1** *Let $F$ be a graph on $f$ vertices with at least one edge, let $K$ be a graph on $k$ vertices, and suppose $F \mapsto K$ (thus, $k \geq 2$). Then, for every sufficiently large $n \geq N(f)$, an $n/k$-blowup of $K$, is $\frac{1}{2k^2}$-far from being $F$-free.*

**Proof:** Denote by $B$ the $n$-vertex $n/k$-blowup of $K$. Our goal is to show that after removing any set of $n^2/2k^2$ edges from $B$, the resulting graph still contains a copy of $F$. Name the vertices of $K$ by $1, \dots, k$ and the independent sets that replaced them by $I_1, \dots, I_k$. Note, that for every choice of $v_1 \in I_1, \dots, v_k \in I_k$, these $k$ vertices span a copy of $K$ with $v_i \in I_i$ playing the role of vertex $i \in V(K)$. We thus get that there are precisely $(n/k)^k$ such copies of $K$ in $B$. ($B$ may very well contain more copies of $K$ but it is simpler to disregard them). Denote the set of these $(n/k)^k$ copies of $K$ by $\mathcal{K}$, and note that each edge in $B$ belongs to *precisely* $(n/k)^{k-2}$ copies of $K$ that belong to $\mathcal{K}$. We conclude that removing any set of $n^2/2k^2$ edges, destroys at most $\frac{1}{2}(n/k)^k$ of the copies of $K$ that belong to $\mathcal{K}$. Thus, after removing any set of $n^2/2k^2$ edges, the new graph, denoted $B'$, contains at least $\frac{1}{2}(n/k)^k$ copies of $K$ that belong to $\mathcal{K}$.

We now define a $k$-uniform $k$-partite hypergraph $H$, based on $B'$. We think of the $k$ partition classes of $H$ as the $k$ vertex sets of $B$ denoted above by $I_1, \dots, I_k$. For every $k$ vertices $v_1 \in I_1, \dots, v_k \in I_k$ that span a copy of $K$ that belongs to $\mathcal{K}$ in $B'$, we put an edge in $H$ containing $v_1, \dots, v_k$. As $B'$ contains at least $\frac{1}{2}(n/k)^k$ copies of $K$ from $\mathcal{K}$, the hypergraph $H$ contains at least $\frac{1}{2}(n/k)^k$ edges. By Theorem 4 for large enough $n$ (i.e. large enough so that $n \geq N(f)$ and so that $\frac{1}{2}(n/k)^k \geq n^{k-f^{1-k}}$), the $k$-uniform hypergraph $H$ contains a copy of $K_f^k$. For $1 \leq i \leq k$, let $S_i$ denote the $f$ vertices in $I_i$, which span this copy of $K_f^k$. By the definition of $H$, as well as the definition of the copies of $K$ that belong to $\mathcal{K}$, we may conclude the following: In $B'$, for every $1 \leq i < j \leq k$ for which $(i,j) \in E(K)$, every vertex $v \in S_i$ is connected to every vertex $u \in S_j$. As $F \mapsto K$ it is now obvious that the vertices $\bigcup_{i=1}^{k} S_i$ span a copy of $F$ in $B'$, which is what we wanted to show. ■

To prove Theorem 3 we also need the following simple observation.

**Claim 3.2** *Let $\mathcal{F}$ be a family of graphs and let $T$ be a one-sided error uniform tester for the property of being $\mathcal{F}$-free, whose query complexity is $Q(\epsilon)$. Suppose that given an error parameter $\epsilon_0$, after $T$ samples a set of vertices $S$ of size $Q(\epsilon_0)$, the graph induced by $S$ is $\mathcal{F}$-free. Then $T$ must accept the input.*

**Proof:** Suppose $T$ does not accept the graph induced by $S$, and denote by $G'$ the graph induced $S$. Suppose now that we were to execute $T$ with the same error parameter $\epsilon_0$ where the input graph is now $G'$. In that case the algorithm would just get back from the oracle the same graph $G'$ and would reject the input $G'$. This however contradicts the assumption that $T$ has one-sided error. ■

To simplify the proof of Theorem 3 we claim that we may assume that $\mathcal{F}$ contains no edgeless graph. Indeed, if $\mathcal{F}$ contains such a graph on $t$ vertices, then by definition $\Psi_{\mathcal{F}}(k) \leq t$ for any $k$ (this is because an edgeless graph has a homomorphism to a single vertex). Thus, it is easy to see (e.g. by applying the algorithm described in the proof below) that in this case $\Psi_{\mathcal{F}}(k)$ is recursive, and there is thus nothing to prove.

**Proof of Theorem 3:** We prove that if the property of being $\mathcal{F}$-free is uniformly testable with one-sided error and with arbitrary query complexity $Q(\epsilon)$, then $\Psi_{\mathcal{F}}(k)$ is recursive. Given a family of graphs $\mathcal{F}$ with no edgeless graph, consider the following algorithm for (nearly) computing $\Psi_{\mathcal{F}}(k)$, which simply implements its definition. The algorithm goes over all graphs $R$, of size at most $k$. For each such graph $R$, it searches for the smallest (in terms of number of vertices) $F \in \mathcal{F}$ for which $F \mapsto R$, if one such $F$ exists [4]. The algorithm then takes the maximum over all graphs $R$ for which it found at least one $F \in \mathcal{F}$ such that $F \mapsto R$. If for all graphs $R$ of size at most $k$, there is no $F \in \mathcal{F}$ for which $F \mapsto R$, the algorithm returns 0.

The only problem with implementing the above algorithm is that given $R$, it is not clear when should the algorithm stop looking for a graph $F$ for which $F \mapsto R$, if none exists. However, note that in order to make sure that the algorithm always terminates with the correct value of $\Psi_{\mathcal{F}}(k)$, it is enough for the algorithm to be able to compute an *upper bound* on the size of such a graph $F$. In other words, it is enough to be able to compute an integer $M$ such that if there is no $F \in \mathcal{F}$ of size at most $M$ for which $F \mapsto R$, then no such $F \in \mathcal{F}$ exists.

We claim that for any $k \geq 2$ we can take $M = Q(1/2k^2)$ as such an upper bound, where $Q(\epsilon)$ is the upper bound for the query complexity of the uniform tester for the property of being $\mathcal{F}$-free[5]. Note, that $M$ is thus computable, since we can simulate the alleged uniform tester with input $\epsilon = 1/2k^2$ and "see" what is the upper bound $Q = Q(\epsilon)$ that it is going to use. Here we use the fact that a uniform tester operates by first computing an upper bound for its query complexity $Q = Q(\epsilon)$. Thus, we can "run" the tester on, say, an edgeless graph.

We thus only have to show that it cannot be the case that for some $R$ on $k$ vertices, the smallest $F \in \mathcal{F}$ for which $F \mapsto R$ is larger than $Q(1/2k^2)$. Assume that one such $R$ exists, let $F_R$ denote the smallest $F \in \mathcal{F}$ for which $F \mapsto R$, and consider an $n/k$ blowup of $R$, denoted by $B$. As by assumption $F_R \mapsto R$, we get by Lemma 3.1, that for every sufficiently large $n$, this blowup is $\frac{1}{2k^2}$-far from being $F_R$-free (here we also use the fact that $F_R$ contains at least one edge). As $F_R \in \mathcal{F}$, the graph $B$ is also $\frac{1}{2k^2}$-far from being $\mathcal{F}$-free. On the other hand, note that for any graph $F'$ that is spanned by $B$ (i.e. $F'$ is a subgraph of $B$, which is not necessarily induced), there is a natural homomorphism $\varphi$, from $F'$ to $R$, which maps all the vertices of $F'$ that belong to the independent set that replaced vertex $v$, to $v$. Since by assumption $F_R$ is the smallest $F \in \mathcal{F}$ for which $F \mapsto R$, and $F_R$ has more than $Q(1/2k^2)$ vertices, we conclude that there is no $F' \in \mathcal{F}$ on at most $Q(1/2k^2)$ vertices which is spanned by $B$. Now, by Claim 3.2, a one-sided error tester must find a member of $\mathcal{F}$ in order to declare that $B$ is not $\mathcal{F}$-free. However, as the smallest member of $\mathcal{F}$ spanned by $B$ has more than $Q(1/2k^2)$ vertices, this cannot be done with query complexity $Q(1/2k^2)$. ■

Observe, that when computing the integer $M$ in the above proof we do not know the size of the smallest graph $F$ such that $F \mapsto R$. Hence, had we used a version of Lemma 3.1 where the distance from being $F$ free is also a function of the size of $F$, we could not have computed the integer $M$, and thus could not have inferred that $\Psi_{\mathcal{F}}$ is recursive. Note also that the proof works no matter how large the query complexity $Q(\epsilon)$ is (this only affects the running time of the algorithm for computing $\Psi_{\mathcal{F}}(k)$), as long as it is a function of $\epsilon$ only.

[4]As we assume that it is decidable to tell whether a graph belongs to $\mathcal{F}$, we can go over the graphs in $\mathcal{F}$ in order of increasing number of vertices, and for every graph try all possible homomorphisms from $F$ to $R$

[5]$\Psi_{\mathcal{F}}(1) = 0$ because $\mathcal{F}$ does not contain independent sets.

## 4 Separating Uniform Testing from Non-Uniform Testing

In this section we prove Theorem 1 by constructing a family of graphs $\mathcal{F}$ for which it is possible to test the property of being $\mathcal{F}$-free non-uniformly, however it is impossible to test this property uniformly. The key combinatorial part of the construction of $\mathcal{F}$ is Lemma 4.1 below. For the proof of this lemma, we need an algorithm that can efficiently produce graphs with arbitrary large chromatic number and girth, where the girth of a graph $G$ denotes the size of the smallest cycle spanned by $G$. We denote by $\chi(F)$ and $g(G)$ the chromatic number and girth of a graph $G$. One of the best-known results of Erdős ([8], see also [6]), widely considered to be the most striking early application of the Probabilistic Method, asserts that such graphs exist. For our purposes however, we need explicit construction of such graphs. It is known that the efficiently constructible $d$-regular expanders of [18], have this property. This is formulated in the following theorem.

**Theorem 5 ([18])** *There is an algorithm $Alg(k, g)$ that given a pair of positive integers $k$ and $g$, $Alg(k, g)$ deterministically constructs a graph $F_{k,g}$ satisfying $\chi(F_{k,g}) > k$ and $g(F_{k,g}) > g$. Moreover, the running time of $Alg(k, g)$ is polynomial in $|V(F_{k,g})|$.*

The reader can find some additional details about the above theorem in Subsection 5.1. Applying the above theorem we prove the following key lemma.

**Lemma 4.1** *There is an infinite family of graphs $F_1, F_2, \ldots$ with the following properties:*

1. *All the graphs $F_1, F_2, \ldots$ are connected and have no vertex of degree 1.*
2. *For any $1 \leq i < j$ we have $F_i \not\mapsto F_j$ and $F_j \not\mapsto F_i$.*
3. *There is an algorithm, which given $i$, constructs $F_i$.*

**Proof:** We define the graphs $F_1, F_2, \ldots$ inductively as follows; $F_1$ is defined to be $K_3$ (i.e., a triangle). For every $i \geq 2$ we pick $F_i$ to be the graph returned by calling the algorithm $Alg(k, g)$ of Theorem 5 with the parameters $k = \chi(F_{i-1})$ and $g = |V(F_{i-1})|$. To get item (1) of the lemma we can now remove repeatedly from $F_i$ any vertex of degree 1 because removing such a vertex does not change either the girth or the chromatic number of a graph (observe that all $F_i$ satisfy $\chi(F_i) \geq 3$). Also, we can assume without loss of generality that each graph $F_i$ is connected, because if it is not, then we can take as $F_i$ an appropriate connected component of $F_i$, which has girth and chromatic number at least as large as those of $F_i$. We thus get item (1) of the lemma. As we can use Theorem 5 in order to generate these graphs one after the other, we also get item (3).

We turn to prove item (2). First, note that if $\varphi : V(F) \mapsto V(K)$ is a homomorphism then any legal $c$-coloring of the vertices of $K$ induces a legal $c$-coloring of the vertices of $F$; We simply color $v \in V(F)$ with the color of $\varphi(v)$. Therefore, if $\chi(F) > \chi(K)$ then we have $F \not\mapsto K$. Consider any pair $F_i$ and $F_j$ with $i < j$. As $\chi(F_j) > \chi(F_i)$ we immediately have that $F_j \not\mapsto F_i$. As $g(F_j) > |V(F_i)|$, every subgraph of $F_j$ of size at most $|V(F_i)|$ does not span any cycle. In particular, any such subgraph is 2-colorable. Hence, as $\chi(F_i) > 2$, we also have $F_i \not\mapsto F_j$, completing the proof. ■

In order to define the family of graphs $\mathcal{F}$, which we need in order to prove Theorem 1, we introduce the following definition.

**Definition 4.2 (The language $L_{BH}$)** *Fix any binary encoding of Turing-Machines. Define $L_{BH}$ (short for* Bounded Halting*) to be the set of all pairs $i\#j$, such that the binary representation of $i$ is a legal encoding of a Turing-Machine, which halts on an empty string within at most $j$ steps.*

Clearly, $L_{BH}$ is a decidable language; we first check if the binary representation of $i$ is a legal encoding of a Turing-Machine. If it is not we reject. Otherwise, we simulate this machine for $j$ steps on an empty string and check if during these $j$ steps the machine halts.

In what follows $P_j$ denotes a path of length $j$, and $F + P_j$ denotes the graph obtained by connecting $P_j$ to an arbitrary vertex of $F$. We are now ready to define $\mathcal{F}$.

**Definition 4.3 (The family $\mathcal{F}$)** *Let $F_1, F_2, \ldots$ be the graphs from Lemma 4.1. Define*

$$\mathcal{F} = \bigcup_{i\#j \in L_{BH}} (F_i + P_j)$$

We now turn to prove that the family $\mathcal{F}$ has the required properties needed in order to satisfy the two assertions of Theorem 1. As in this paper we confine ourselves to decidable properties, we first show that being $\mathcal{F}$-free is a decidable property. In fact, we also need this in order to apply Theorem 2 (recall Comment 2.3). As shown in the next lemma, we can even show that being $\mathcal{F}$-free belongs to $coNP$.

**Lemma 4.4** *Being $\mathcal{F}$-free, where $\mathcal{F}$ is the family of graphs from Definition 4.3, is in $coNP$.*

**Proof:** We prove the equivalent statement that the property of having a subgraph isomorphic to one of the graphs of $\mathcal{F}$ is in $NP$. Given a graph $G$ of size $n$, the non-deterministic algorithm guesses a (not necessarily induced) subgraph of $G$, which we denote by $T'$, a number $1 \leq t \leq n$ and an injective mapping $\sigma : [1, \ldots, t] \mapsto [1, \ldots, n]$. We next describe how the algorithm checks, using $t$ and $\sigma$, whether $T' \in \mathcal{F}$.

The algorithm first verifies that $T'$ has the structure of the graphs in $\mathcal{F}$. To this end, it first counts the number of vertices of degree 1 in $T'$. If this number is not precisely 1, or if $T'$ is not connected the algorithm rejects the input (because by item (1) of Lemma 4.1 all the graphs in $\mathcal{F}$ are connected and have precisely one vertex of degree 1). Otherwise, let $j$ be the length of the walk starting from the single vertex of degree 1, until the first vertex of degree at least 3 (including this last vertex), and let $T$ be the graph obtained from $T'$ by removing the $j$ vertices of this path. The algorithm also rejects if $T$ is not of size $t$. The algorithm now turns to check if $T$ is isomorphic to one of the graphs $F_i$ of Lemma 4.1, and if this is the case, whether $i\#j \in L_{BH}$.

The algorithm uses Theorem 5 in order to produce the graphs $F_1, F_2, \ldots$ as they were defined in the proof of Lemma 4.1. Note, that by our definition of these graphs each $F_i$ must be strictly larger than $F_{i-1}$. If Theorem 5 produces a graph of size larger than $t$ without first producing one of size $t$ the algorithm rejects. Assume now that Theorem 5 produces a graph, say $F_i$, of size precisely $t$. The algorithm now checks whether for every edge $(i, j) \in E(F_i)$ the vertices $(\sigma(i), \sigma(j))$ also form an edge in $G$ (recall that $\sigma$ is an injective mapping from $[t]$ to $n$). If all these (at most $\binom{t}{2} \leq \binom{n}{2}$) tests succeed the algorithm moves to the last step, otherwise it rejects. Note, that at this step we know that $T$ is isomorphic to some graph $F_i$ from Lemma 4.1. To complete the verification that $T' \in \mathcal{F}$ the algorithm runs the algorithm (which is polynomial in $i$ and $j$, which are bounded by $n$) for checking if $i\#j$ belongs to $L_{BH}$ and accepts if and only if this algorithm accepts.

The above algorithm clearly rejects any $G$ that is $\mathcal{F}$-free, and for any $G$ that is not $\mathcal{F}$-free there is a choice of $T'$, $t$, and $\sigma$, for which it accepts $G$. Finally, as for any $i$ we have $|V(F_i)| > |V(F_{i-1})|$, we invoke Theorem 5 at most $n$ times. As by Theorem 5 the time needed to produce each of the graphs $F_i$ is polynomial in $|V(F_i)|$, we almost infer that the total running time of this algorithm is polynomial in $n$. The only annoying technicality is that it might be the case that we try to invoke Theorem 5 on inputs $k$ and $g$ for which the size of the graph it produces is super-polynomial in the size of the input graph $G$. To overcome this difficulty we can simply simulate the algorithm of Theorem 5 and reject if it runs longer than the time needed to produce a graph of size at most $n$, which is polynomial in $n$. ■

We turn to prove the main result of the section:

**Lemma 4.5** *The function $\Psi_{\mathcal{F}}$, where $\mathcal{F}$ is the family of graphs from Definition 4.3, is non-recursive.*

**Proof:** We show that if $\Psi_{\mathcal{F}}$ is recursive, then given a legal encoding of a Turing-Machine $M$, we can compute an integer $N$ with the following property: If $M$ halts on the empty string, then it does so after at most $N$ steps. We will thus get that we can decide whether $M$ halts on the empty string, because we can simulate $M$ on the empty string for $N$ steps and accept if and only if $M$ halts within these $N$ steps. This will obviously be a contradiction, as deciding if a Turing-Machine halts on an empty string is well-known to be undecidable (see [19]).

Given an integer $i$, which (correctly) encodes some Turing-Machine $M$, the algorithm first computes the graph $F_i$. To this end we rely on item (3) of Lemma 4.1. Let $k$ denote the number of vertices of $F_i$. We claim that we can set $N = \Psi_{\mathcal{F}}(k)$. First, observe that $N$ is thus computable as $\Psi_{\mathcal{F}}$ is by assumption recursive. If $M$ does not halt on the empty string, then we do not care about the value of $N$ as no matter for how many steps we simulate $M$, it will never halt, and we will return the correct answer. Assume thus that $M$ halts after $T$ steps. We only have to show that $T \leq N = \Psi_{\mathcal{F}}(k)$.

First, observe that for any graph $F$ and integer $j$ we trivially have $F \mapsto F + P_j$ and $F + P_j \mapsto F$. By item (2) of Lemma 4.1, we know that for any $i < i'$ we have $F_i \not\mapsto F_{i'}$ and $F_{i'} \not\mapsto F_i$. Therefore, for any $i < i'$ and for any $j, j'$ we also have $F_i + P_j \not\mapsto F_{i'} + P_{j'}$ and $F_{i'} + P_{j'} \not\mapsto F_i + P_j$. It thus follows that the only $F \in \mathcal{F}$ such that $F \mapsto F_i$, are the graphs of the form $F_i + P_j$ for some integer $j$. However, as we only put in $\mathcal{F}$ the graphs $F_i + P_j$ for which $i\#j \in L_{BH}$ we infer that the only $F \in \mathcal{F}$ such that $F \mapsto F_i$, are the graphs of type $F_i + P_j$ for $j \geq T$. In particular, the smallest $F \in \mathcal{F}$ such that $F \mapsto F_i$ has size at least $T$. As $\Psi_{\mathcal{F}}(k)$ takes the maximum over all the graphs of size at most $k$, and $F_i$ is one of these graphs, we get that $N = \Psi_{\mathcal{F}}(k) \geq T$. Hence, $N$ is indeed an upper bound on the running time of $M$ in the case that it halts on an empty string. ■

Call a graph property *hereditary* if it is closed under removal of vertices. The last tool we need is the following result of the first author ([14], Appendix D). In [14], the notion of uniformly testing a property was not used, but the statement as it appears below is equivalent to what is proved in [14].

**Theorem 6 (c.f. [14])** *A hereditary graph property is uniformly testable with two-sided error if and only if it is uniformly testable with one-sided error.*

**Proof of Theorem 1:** We claim that in order to prove the theorem we can set $\mathcal{P}$ to be the property of being $\mathcal{F}$-free with $\mathcal{F}$ being the family given in Definition 4.3. First, being $\mathcal{F}$-free is by definition expressed in terms of forbidden subgraphs and by Lemma 4.4 this property is in *coNP*. In particular, this property is decidable, therefore by Theorem 2 it can be tested non-uniformly with one-sided error. Now, by Lemma 4.5 the corresponding function $\Psi_{\mathcal{F}}$ is not recursive. Hence, by Theorem 3 this property cannot be tested uniformly with one-sided error. Finally, as this property is hereditary, Theorem 6 implies that this property cannot be tested uniformly, even with two-sided error. ■

## 5 Concluding Remarks and Open Problems

### 5.1 Some remarks on LPS Expanders

The result of Lubotzky, Philips and Sarnak [18] can be stated as follows (see [6] for background on expander graphs)

**Theorem 7 ([18])** *Suppose $p$ and $q$ are primes congruent to 1 modulo 4, where $p$ is a quadratic residue modulo $q$, and put $d = p + 1$ and $n = q(q^2 - 1)/2$. Then, there is a $d$-regular expander on $n$ vertices, denoted $G_{n,d}$, with second eigenvalue $\lambda \leq 2\sqrt{d-1}$. Moreover,*

- *The chromatic number of $G_{n,d}$ is at least $\sqrt{d}/2$.*
- *The girth of $G_{n,d}$ is at least $\frac{2}{3}\log n/\log d$.*
- *$G_{n,d}$ can be constructed in time polynomial in $|V(G_{n,d})|$.*

Therefore, given integers $k$ and $g$ we can use the known results about the distribution of primes in arithmetic progressions, as well as the above theorem with $n$ and $d$ satisfying $\sqrt{d}/2 > k$ and $\frac{2}{3}\log n/\log d > g$, in order to efficiently construct the graphs $F_{k,g}$ as in Theorem 5.

### 5.2 Proof of Proposition 1.5

Clearly, if a graph property can be tested when $\epsilon$ is given as part of the input, then for every fixed $\epsilon$ there is a tester for distinguishing between graphs satisfying $\mathcal{P}$ from those that are $\epsilon$-far from satisfying it. To show the other direction, we need a theorem of [14] (extending a result of [2]) stating that for every $\epsilon$ and $n$ if a graph property is testable with query complexity $q(n, \epsilon)$, then it can also be tested by a so called "canonical tester", which operates by randomly selecting a set of $2q(n, \epsilon)$ vertices $S$, and then accepting or rejecting according to the graph spanned by $S$, the value of $\epsilon$ and the size of the input $n$.

Suppose then that for any $\epsilon > 0$ there is a tester $T_\epsilon$ that given the size of an input can distinguish between graphs satisfying $\mathcal{P}$ from those that are $\epsilon$-far from satisfying it, such that the query complexity of $T_\epsilon$ is at most $Q(\epsilon)$. Note, that we do not assume that the query complexity is a function of $\epsilon$ only, but just that it is bounded by a function of $\epsilon$ as in Definition 1.1. We turn to show that in this case there is a tester for $\mathcal{P}$ that receives $\epsilon$ as part of the input. The tester works as follows: Given $n$ and $\epsilon$ the algorithm constructs the following families of $n$-vertex graphs: $A$, which consists of all the $n$-vertex graphs satisfying $\mathcal{P}$, and $B$, which consists of all the $n$-vertex graph, which are

$\epsilon$-far from satisfying $\mathcal{P}$. Starting from $q = 1$ the algorithm now goes over all the possible canonical-testers with query complexity $q$, and for each such tester, checks if it will accept the graphs of $A$ with probability 2/3, and reject the graphs of $B$ with probability 2/3. Recall that a canonical tester works by sampling a set of vertices and then accepting/rejecting according to the graph spanned by the sample. Therefore, when we say that the algorithm goes over all testers with query complexity $q$ we mean that it tries all the possible $2^{2^{\binom{q}{2}}}$ ways of partitioning the set of $q$-vertex graphs into those that will make the canonical tester accept and those that will make it reject. Also, when we say that the algorithm checks, whether a given canonical tester accepts a graph from $A$ with probability 2/3, we mean that the algorithm checks if 2/3 of the subsets of $q$ vertices of the graph span a graph, which makes the canonical tester accept. Now, the main point is that as we assume that $\mathcal{P}$ can be tested using a number of queries that is bounded by a function $Q(\epsilon)$, the algorithm will eventually find that for some $q \leq Q(\epsilon)$ there is a canonical tester $T'$ for $\epsilon$-testing $\mathcal{P}$ on $n$-vertex graphs. Once $q$ and $T'$ are found the algorithm executes $T'$ on the input graph. By definition, this algorithm is a tester for $\mathcal{P}$, whose query-complexity is at most $Q(\epsilon)$. $\blacksquare$

The tester used in the above proof has two interesting features, which we have alluded to at the end of Subsection 1.3. First, although the query complexity of the tester, which we construct in the above proof, is bounded by a function of $\epsilon$ only, it's running time is exponential in $n$, due to the need to go over all graphs of size $n$. Second, note that although the query complexity of the tester is bounded by a function of $\epsilon$ only, it is in fact a function of $\epsilon$ and $n$. The reason is that what the algorithm does, is look for the smallest query complexity $q$, which is sufficient for testing $\mathcal{P}$ on $n$-vertex graphs. As $\mathcal{P}$ is assumed to be testable with a constant number of queries, we are guaranteed that for every $n$ this quantity is bounded by some function of $\epsilon$, which is independent of $n$. However, it may be the case that for fixed $\epsilon$ the optimal query complexity is different for different values of $n$. Therefore, for fixed $\epsilon$ the query complexity may be different for different values of $n$.

## 5.3 Possible extensions

The main result of the paper, Theorem 1, establishes that if we confine ourselves to slightly weakened testers then there are non-trivial tasks (computing the query complexity), which cannot be done if $\epsilon$ is an unknown that is given as part of the input. Moreover, this phenomenon holds for properties that are natural in terms of their combinatorial structure (as they are monotone) and also in terms of their computational difficulty (as they are in *coNP*). This means that we can formally prove that in some cases knowing the error parameter $\epsilon$ in advance can help the tester in a non-trivial way. An interesting problem is whether one can find a separating property satisfying the assertions of Theorem 1, which belongs to $P$ or perhaps even to a lower complexity class.

**Acknowledgements:** First and foremost, we are grateful to an anonymous FOCS'05 reviewer for communicating Proposition 1.5 to us. We would also like to thank Eldar Fischer and Oded Goldreich for their helpful suggestions regarding the presentation of our results, and Vera Asodi for a careful reading of this paper and for her helpful comments.

## References

[1] N. Alon, Testing subgraphs in large graphs, Proc. $42^{nd}$ IEEE FOCS, IEEE (2001), 434-441. Also: Random Structures and Algorithms 21 (2002), 359-370.

[2] N. Alon, E. Fischer, M. Krivelevich and M. Szegedy, Efficient testing of large graphs, Proc. of $40^{th}$ FOCS, New York, NY, IEEE (1999), 656–666. Also: Combinatorica 20 (2000), 451-476.

[3] N. Alon, E. Fischer, I. Newman and A. Shapira, A combinatorial characterization of the testable graph properties: it's all about regularity, Proc. of STOC 2006, 251-260.

[4] N. Alon and A. Shapira, Every monotone graph property is testable, Proc. of STOC 2005, 128-137. Also, SIAM Journal on Computing, to appear.

[5] N. Alon and A. Shapira, A characterization of the (natural) graph properties testable with one-sided error, Proc. of FOCS 2005, 429-438. Also, SIAM Journal on Computing, to appear.

[6] N. Alon and J. H. Spencer, **The Probabilistic Method**, Second Edition, Wiley, New York, 2000.

[7] M. Blum, M. Luby and R. Rubinfeld, Self-testing/correcting with applications to numerical problems, JCSS 47 (1993), 549-595.

[8] P. Erdős, Graph theory and probability, Canad. J. Mathematics, (11) 1959, 34-38.

[9] P. Erdős, On extremal problems of graphs and generalized graphs, Israel J. Math. 2, 1964, 183-190.

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

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

[12] O. Goldreich, S. Goldwasser and D. Ron, Property testing and its connection to learning and approximation, JACM 45(4): 653-750 (1998).

[13] O. Goldreich and D. Ron, Property Testing in Bounded-Degree Graphs, Proc. of STOC (1997), 406-415.

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

[15] T. Kaufman, M. Krivelevich and D. Ron, Tight bounds for testing bipartiteness in general graphs, SIAM J. on Computing 33 (2004), 1441-1483.

[16] T. Kővári, V. T. Sós and P. Turán, On a problem of K. Zarankiewicz, Colloquium Math., 3, (1954), 50-57.

[17] L. Lovász and B. Szegedy, Graph limits and testing hereditary graph properties, manuscript, 2005.

[18] A. Lubotzky, R. Philips and P. Sarnak, Ramanujan graphs, Combinatorica, 8 (1988), 261-277.

[19] C. Papadimitriou, **Computational Complexity**, Addison Wesley, 1994.

[20] M. Parnas and D. Ron, Testing the diameter of graphs, Random structures and algorithms, 20 (2002), 165-183.

[21] D. Ron, Property testing, in: P. M. Pardalos, S. Rajasekaran, J. Reif and J. D. P. Rolim, editors, *Handbook of Randomized Computing*, Vol. II, Kluwer Academic Publishers, 2001, 597–649.

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