Electronic Colloquium on Computational Complexity, Report No. 13 (2011)

Sublinear Time Algorithms

Ronitt Rubinfeld [*] Asaf Shapira [†]

Abstract

Sublinear time algorithms represent a new paradigm in computing, where an algorithm must give some sort of an answer after inspecting only a very small portion of the input. We discuss the types of answers that one can hope to achieve in this setting.

1 Introduction

Traditionally, showing the existence of a linear time algorithm for a problem was considered to be the gold standard of achievement. Nevertheless, due to the recent tremendous increase in computational power that is inundating us with a multitude of data, we are now encountering a paradigm shift from traditional computational models. The scale of these data sets, coupled with the typical situation in which there is very little time to perform our computations, raises the issue of whether there is time to consider any more than a miniscule fraction of the data in our computations? For most natural problems, an algorithm which runs in sublinear time must necessarily use randomization and must give an answer which is in some sense imprecise. Nevertheless, there are many situations in which a fast approximate solution is more useful than a slower exact solution.

A first example of sublinear time computation that comes to mind is the classical result from the theory of sampling that one can, in time independent of the size of the data, determine a good estimate of the average value of a list of numbers of bounded magnitude. But what about more interesting algorithmic questions? For example, suppose one is given the ability to answer questions of the form "does person A know person B?". Then consider the graph in which nodes correspond to people and edges correspond to pairs of people that know each other – can one determine whether this graph has the "small world property", i.e., does it have small diameter? This is easily detectable after asking whether person A knows person B for all possible pairs of people. However, by the time all of these questions are fully answered the answer might be very different, and moreover it might be too late to make use of the information. Fortunately, it might be feasible to construct algorithms based on random sampling which do not need to ask about all pairs. The recently emerging theory of sublinear time algorithms addresses questions of precisely this nature for problems in various domains.

[*]CSAIL, MIT, Cambridge MA 02139 and the Blavatnik School of Computer Science at Tel Aviv University. Supported by NSF grants 0732334 and 0728645, Marie Curie Reintegration grant PIRG03-GA-2008-231077 and the Israel Science Foundation grant nos. 1147/09 and 1675/09.

[†]Shool of Mathematics and College of Computing, Georgia Institute of Technology, 686 Cherry St., Atlanta, GA 30332. Email: asafico@math.gatech.edu. Research supported in part by NSF Grant DMS-0901355.

ISSN 1433-8092

This paper will describe a number of problems that can be solved in sublinear time, using different types of approximations.

Outline of the paper. We devote Section 2 to a simple example of a testing problem which will motivate the definition we will later use in the paper. In Section 3 we formally define the key notions that will be used later on in the paper. In Section 4 we discuss property testing algorithms for problems that are algebraic in nature. In Section 5, we discuss property testing algorithms for graphs (sparse and dense), and in Section 6 we give examples of sublinear time approximation algorithms for optimization problems. Throughout the paper we will try to emphasize both the algorithmic aspects of designing sublinear algorithms, as well as the mathematical/structural aspects of this theory, which stem from the need to find so called *robust characterizations* of different properties. We wish to stress that there are many beautiful results that we are not able to mention in this survey. Fortunately, some of these are mentioned in other excellent surveys (for example, see [55, 64, 112, 113, 120]).

2 A simple example: monotonicity of a list

Let us begin with a simple example, which will motivate our subsequent definitions of the types of approximations that we will be interested in achieving. A list of integers $\vec{x} = x_1, \ldots, x_n$, is *monotone* (increasing) if $x_i \leq x_j$ for all $1 \leq i < j \leq n$. Given input $\vec{x}$, the task is to determine whether or not $\vec{x}$ is monotone.

It is clear that finding a sublinear time algorithm for the above task is impossible, since any algorithm that does not look at some x_j could be fooled by an input for which all the x_i's are in monotone order for $i \neq j$. Thus, we can only hope to solve an approximate version of this problem, but what is a meaningful notion of an approximation?

One natural approximate version is defined as follows: Say that $x_1, \ldots, x_n$ is *ϵ-close* to monotone if by changing at most ϵn of the values of the x_i's one can transform $x_1, \ldots, x_n$ into a monotone list. Then, a *property tester for monotonicity* is a randomized algorithm that on input $\vec{x}, \epsilon$, must output "pass" if $x_1, \ldots, x_n$ is monotone, and "fail" if $x_1, \ldots, x_n$ is not ϵ-close to monotone. The algorithm is allowed to err with probability at most $1/3$. However, once an algorithm with error probability at most $1/3$ is achieved, it is easy to see that for any β, a probability of error of at most β can be achieved by repeating the algorithm $O(\log \frac{1}{\beta})$ times and taking the majority answer. Note that if $x_1, \ldots, x_n$ is, say, $\epsilon/2$-close to monotone, the property testing algorithm is allowed to output "pass" or "fail". Indeed, in this case, since the list is close to monotone, it may not be too harmful to pass it. On the other hand, since it is not actually monotone, it is also not a big injustice to fail it.

How do we construct a property tester for monotonicity? To design such an algorithms we need to come up with a *robust characterization* of monotone lists, where, informally speaking, a robust characterization of a property is a characterization which applies to both objects that satisfy and to objects that are close to satisfying the property. On first try to prove that if a string is ϵ-far from being monotone then there must be $\delta(\epsilon)n^2$ pairs of indices $i < j$ for which $x_i > x_j$, where $\delta(\epsilon)$ is independent of n. If true, that would allow us to design a trivial algorithm which simply sample $1/\delta(\epsilon)$ pairs $i < j$ and checks if $x_i > x_j$. However, it is easy to construct examples showing that there are lists that are not even $1/2$-close to monotone, yet only $n^{3/2}$ of the pairs $i < j$ satisfy $x_i > x_j$. This means that at least $\sqrt{n}$ pairs must be sampled if one is to find a reason to output "fail". We will soon see that one can do significantly better by finding a more "efficient" characterization of monotone lists! In the following, we describe an $O(\log n)$ time algorithm from the work of Ergün et. al. [53] which tests if $\vec{x}$ has a long monotone increasing subsequence. Note that the problem is known to require $\Omega(\log n)$ queries [53, 56].

Let c be a constant that is set appropriately. For simplicity, let us assume that the elements in $\vec{x}$ are distinct. The last assumption is without loss of generality, since one can append the index of an item to the least significant bits of its value in order to break ties without changing the proximity of the list to monotone.

1. Let $\ell = c/\epsilon$. Choose indices $i_1, \ldots, i_\ell$ uniformly from $[n]$.
2. For each such chosen index i_j, assume the list is monotone and perform a binary search in $\vec{x}$ as if to determine whether the value x_{i_j} is present in $\vec{x}$ or not.
3. Output "fail" if the binary search fails to find any x_{i_j} in location i_j or finds a pair of out-of-order elements along the search path. Output "pass" if all the ℓ binary searches succeed.

The running time of the algorithm is $O((1/\epsilon)\log n)$. Moreover, if $\vec{x}$ is monotone, then the algorithm will always output "pass" as each of the binary searches will succeed (since in this case there is no probability of error, we refer to this as a "1-sided error tester"). To show that if $\vec{x}$ is not ϵ-close to monotone, then the algorithm will output "fail" with probability at least $2/3$, we show the contrapositive. Namely, assume that the input is such that the algorithm outputs "pass" with probability at least $1/3$. To see that that $\vec{x}$ has a long increasing subsequence, let $G \subseteq [n]$ denote the set of indices for which the binary search would succeed, i.e., $i \in G$ if and only if x_i can be found by a binary search on $\vec{x}$. The constant c can be chosen such that if $|G| < (1-\epsilon)n$, then the algorithm would pick some $i_j \notin G$ with probability at least $1/3$, causing it to output "fail". Thus, since the algorithm outputs "pass" with probability at least $1/3$, we know that $|G| \geq (1-\epsilon)n$. We now argue that the restriction of $\vec{x}$ to the indices in G is an increasing subsequence: Let $i, j \in G$ and $i < j$. Let k be the least common ancestor index where the binary searches for x_i and x_j diverge. Then, since $i < j$, it must be that the search for x_i goes left and the search for x_j goes right, and so $x_i < x_k < x_j$. Finally, if $\vec{x}$ has an increasing subsequence of size at least $(1-\epsilon)n$ then it is easy to see that $\vec{x}$ is ϵ-close to monotone. Thus we have the following theorem:

Theorem 1 ([53]) *There is an algorithm that, given a sequence $\vec{x} = x_1, \ldots, x_n$ and an $\epsilon > 0$, runs in $O((1/\epsilon)\log n)$ time and outputs* (1) "pass"*, if $\vec{x}$ is monotone and* (2) "fail"*, with probability* $2/3$*, if $\vec{x}$ does not have an increasing subsequence of length at least $(1-\epsilon)n$ (in particular, if $\vec{x}$ is ϵ-far from monotone).*

An equivalent way to view the algorithm presented here is as a way of testing whether a function mapping from $[n]$ into $\mathbb{R}$ is monotone. There has been much interest in the problem of testing monotonicity of functions over more general domains and ranges – see [65, 50, 57, 35, 27, 75, 105, 2, 1] for a few examples of further work. In [57] it is shown that testing the monotonicity of functions over poset domains is equivalent to several other testing problems. In [35] a very interesting connection is given between testing the monotonicity of functions and the construction of transitive closure graph spanners.[1]

3 What do we mean by an "approximation"?

Now that we have developed some intuition, we present our model and definitions in more detail: We are interested in computing some function f on input x without reading all of x. This is an impossible task in general, since a change to a single bit of x could alter the value of f. For most nontrivial functions f, if one cannot read all of the input x, then one can only hope to approximate the value of f on x. Standard notions of multiplicative and additive approximations exist for functions f which map to such ranges as the real numbers. In Section 6, we mention several sublinear time approximation algorithms which give such guarantees.

However, what do we mean by an approximation when f is the characteristic function of a property or a language? In recent years, a notion of approximation for Boolean functions has emerged, referred to as *property testing*, which has led to a wealth of sublinear time algorithms. Given an input, a *property tester* tries to distinguish whether the input has the property from the case where the input is not even close to having the property.

More specifically, a property $\mathcal{P}$ of certain objects is a subset $\mathcal{S}_\mathcal{P}$ of the universe of the objects we consider, and an object x is said to satisfy $\mathcal{P}$ if belongs $\mathcal{S}_\mathcal{P}$. For example, the property of a graph being triangle-free is the family of all triangle-free graphs. In this case, $f(x) = 1$ if x has the property and $f(x) = 0$ otherwise. We next formalize one notion of what it means to be close.

Definition 3.1 An input x, represented as a function $x : \mathcal{D} \to \mathcal{R}$, is *$\epsilon$-close* to satisfying property P if there is some y satisfying P such that x and y differ on at most $\epsilon|\mathcal{D}|$ places in their representation. Otherwise, x is said to be *ϵ-far* from satisfying P.

In the monotonicity example of the previous section, $\mathcal{D} = [n]$ and $x(i)$ denotes the i^{th} element of the list. This notion of "close" depends strongly on the representation of the object being tested. There may be multiple standard ways of representing various objects, for example, two common representations of graphs are via adjacency lists and adjacency matrices. The choice of the representation affects the notion of distance that makes sense for the particular application as well as the complexity of the testing task.

We now formalize what it means for an algorithm to test a property. As in the previous section, we assume in our model of computation that algorithms have query access to the input.

Definition 3.2 Let P be a property. On input x of size $n = |\mathcal{D}|$ and ϵ, a *property tester* for P must satisfy the following:

- If x satisfies property P, the tester must output "pass" with probability at least 2/3.
- If x is ϵ-far from satisfying P, the tester must output "fail" with probability at least 2/3.

[1]A k-transitive-closure-spanner (k-TC-spanner) of G is a directed graph $H = (V, E_H)$ that has (1) the same transitive-closure as G and (2) diameter at most k.

The probability of error may depend only on the coin tosses of the algorithm and not on any assumptions of the input distribution. The number of queries made by the property tester $q = q(\epsilon, n)$ is referred to as the query complexity of the property tester. We say that a property tester for P has 1-*sided error* if it outputs "pass" with probability 1 when x satisfies P. The property tester for monotonicity given in the previous section has 1-sided error. If the query complexity is independent of n, then we say that the property is *easily testable.*

Note that if x does not satisfy P but x is also not ϵ-far from satisfying P, then the output of the property tester can be either "pass" or "fail". We have already seen that it is this gap which allows property testers to be so efficient.

The probability that the property tester errs is arbitrarily set to $1/3$ and may alternatively be defined to be any constant less than $1/2$. As we have seen earlier for the case of testing monotonicity, it is then easy to see that for any β, a probability of error of at most β can be achieved by repeating the algorithm $O(\log \frac{1}{\beta})$ times and taking the majority answer.

Property testing was defined by Rubinfeld and Sudan [116] in the context of program testing. Goldreich, Goldwasser, Ron [66] refined and generalized the definition, which allowed the application of property testing to graphs and other combinatorial structures. Various more general definitions are given in several works, including [53, 69, 103], which mostly differ in terms of the generality of the distance function and other specifications as to when the tester should accept and reject.

4 Algebraic Problems

In this section, we consider property testing algorithms for problems that are algebraic in nature. We begin with the problem of testing whether a function is a homomorphism. We then show that the ideas used to construct property testers for homomorphisms extend to other properties with similar underlying structure. See the survey of [113] for several other recent examples of algebraic property testers.

4.1 Homomorphism testing

We begin with an example that was originally motivated by applications in program testing [37] and was later used in the construction of Probabilistically Checkable Proof systems [17]. For groups $\mathcal{D}$ and $\mathcal{R}$, suppose you are given oracle access to a function $f : \mathcal{D} \to \mathcal{R}$. That is, you may query the oracle on any input $x \in \mathcal{D}$ and it will reply with $f(x)$. Is f a homomorphism?

In order to determine the answer exactly, it is clear that you need to query f on the entire domain $\mathcal{D}$. However, consider the property testing version of the problem, for which on input ϵ, the property tester should output "pass" with probability at least $2/3$ if f is a homomorphism and "fail" with probability at least $2/3$ if f is ϵ-far from a homomorphism (that is, there is no homomorphism g such that f and g agree on at least $(1-\epsilon)|\mathcal{D}|$ inputs). In order to construct such a property tester, a natural idea would be to test that the function satisfies certain relationships that all homomorphisms satisfy. We next describe two such relationships and discuss their usefulness in constructing property testers.

Two Characterizations of Homomorphisms over $\mathcal{Z}_q$ Consider the case when f is over the domain and range $\mathcal{D} = \mathcal{R} = \mathcal{Z}_q$ for large integer q. The set of homomorphisms over $\mathcal{Z}_p$ can be characterized as the set of functions which satisfy $\forall x, f(x+1) - f(x) = f(1)$. This suggests that a property tester might test that $f(x+1) - f(x) = f(1)$ for most x. However, it is easy to see that there are functions f which are very far from any homomorphism, but would pass such a test with overwhelmingly high probability. For example, $g(x) = x \bmod \lceil\sqrt{q}\rceil$ satisfies $g(x+1) - g(x) = g(1)$ for $1 - \frac{1}{\sqrt{q}}$ fraction of the $x \in \mathcal{Z}_q$ but $g(x)$ is $(1 - \frac{1}{\sqrt{q}})$-far from a homomorphism.

The set of homomorphisms over $\mathcal{D}$ can alternatively be characterized as the set of functions which satisfy $\forall x, y,\ f(x) + f(y) = f(x+y)$. This suggests that one might test that $f(x) + f(y) = f(x+y)$ for most x, y. It might be worrisome to note that when $q = 3n$, the function $h(x)$ defined by $h(x) = 0$ if $x \equiv 0 \bmod 3$, $h(x) = 1$ if $x \equiv 1 \bmod 3$ and $h(x) = 3n - 1$ if $x \equiv -1 \bmod 3$ passes the above test for $7/9$ fraction of the choices of pairs $x, y \in \mathcal{D}$ and that $h(x)$ is $2/3$-far from a homomorphism [44]. However, $7/9$ is a type of threshold: one can show that for any $\delta < 2/9$, if $f(x) + f(y) = f(x+y)$ for at least $1 - \delta$ fraction of the choices of $x, y \in \mathcal{D}$, then there is some homomorphism g, such that $f(x) = g(x)$ on at least $1 - \delta/2$ fraction of the $x \in \mathcal{D}$ [31].

Once one has established such a theorem, then one can construct a property tester based on this characterization by sampling $O(1/\epsilon)$ pairs x, y and ensuring that each pair in the sample satisfies $f(x) + f(y) = f(x+y)$. This property tester clearly passes all homomorphisms. On the other hand, if f is ϵ-far from a homomorphism then the above statement guarantees that at least 2ϵ fraction of the choices of x, y pairs do not satisfy $f(x) + f(y) = f(x+y)$, and the property tester is likely to fail.

In both cases, homomorphisms are characterized by a collection of *local* constraints, where by local, we mean that few function values are related within each constraint. What is the difference between the first and the second characterization of a homomorphism that makes the former lead to a bad test and the latter to a much better test? In [116] (see also [115]), the notion of a *robust characterization* was introduced to allow one to quantify the usefuleness of a characterization in constructing a property test. Loosely, a robust characterization is one in which the "for all" quantifier can be replaced by a "for most" quantifier while still characterizing essentially the same functions. That is, for a given ϵ, δ, a characterization is *(ϵ, δ)-robust* if for any function f that satisfies at least $1 - \delta$ fraction of the constraints, f must be ϵ-close to some function g that satisfies all of the constraints and is thus a solution of the "for all" characterization. As we saw above, once we have an (ϵ, δ)-robust characterization for a property, it is a trivial matter to construct a property tester for the property.

Homomorphism testing, a history Let G, H be two finite groups. For an arbitrary map $f : G \to H$, define δ, the probability of group law failure, by

$$\delta = 1 - \Pr_{x,y}\left[f(x) + f(y) = f(x+y)\right].$$

Define ϵ to be the distance of f from the set of homomorphisms, i.e., ϵ is the minimum τ for which f is τ-close to a homomorphism. We will be interested in the relationship between ϵ and δ.

Blum, Luby and Rubinfeld [37], considered this question and showed that over cyclic groups, there is a constant δ_0, such that if $\delta \le \delta_0$, then one can upper bound ϵ in terms of a function of δ that is independent of $|G|$. This yields a homomorphism tester with query complexity that depends (polynomially) on $1/\epsilon$, but is independent of $|G|$, and therefore shows that the property of being a homomorphism is easily testable. The final version of [37] contains an improved argument due to Coppersmith [44], which applies to all Abelian groups, shows that $\delta_0 < 2/9$ suffices, and that ϵ is upper bounded by the smaller root of $x(1-x) = \delta$ (yielding a homomorphism tester with query complexity linear in $1/\epsilon$). Furthermore, the bound on δ_0 was shown to be tight for general groups [44]. In [31], it was shown that for general (non-Abelian) groups, for $\delta_0 < 2/9$, then f is ϵ-close to a homomorphism where $\epsilon = (3 - \sqrt{9-24\delta})/12 \leq \delta/2$ is the smaller root of $3x - 6x^2 = \delta$. The condition on δ, and the bound on ϵ as a function of δ, are shown to be tight, and the latter improves that of [37, 44]. Though $\delta_0 < 2/9$ is optimal over general Abelian groups, using Fourier techniques, Bellare et al. [29] have shown that for groups of the form $(\mathcal{Z}_2)^n$, $\delta_0 \leq 45/128$ suffices. (see also [82] for further results). Kopparty and Saraf have investigated the more difficult *tolerant* testing problem, where one wants to distinguish functions that are close to homomorphisms from those that are far from homomorphisms in a distribution free setting [89].

There has been interest in improving various parameters of homomorphism testing results, due to their use in the construction of Probabilistically Checkable Proof Systems (cf. [17]). In particular, both the constant δ_0 and the number of random bits required by the homomorphism test affect the efficiency of the proof system and in turn the hardness of approximation results that one can achieve using the proof system. Several works have shown methods of reducing the number of random bits required by the homomorphism tests. That is, in the natural implementation of the homomorphism test, $2\log_2|G|$ random bits per trial are used to pick x, y. The results of [118, 77, 41, 122] have shown that fewer random bits are sufficient for implementing the homomorphism tests. The recent work of [122] gives a homomorphism test for general (non-Abelian) groups that uses only $(1+o(1))\log_2|G|$ random bits. Given a Cayley graph that is an expander with normalized second eigenvalue γ, and for the analogous definitions of δ, ϵ, they show that for $\delta < (1-\gamma)/12$, ϵ is upper bounded by $4\delta/(1-\gamma)$.

A proof of a homomorphism test We describe the following proof, essentially due to Coppersmith [37, 44] of the robustness of the homomorphism characterization over Abelian groups. Though this is not the strongest known result, we include this proof to give a flavor of the types of arguments used to show robustness of algebraic properties.

Theorem 2 *Let G be an Abelian group. Let δ be such that*

$$1 - \delta = \Pr_{x,y}\left[f(x) + f(y) = f(x+y)\right].$$

Then if $\delta < 2/9$, f is 2δ-close to a homomorphism.

Proof: Define $\phi(x) = \text{maj}_{y\in G}(f(x+y) - f(y))$, that is, let $\phi(x)$ be the value that occurs with the highest probability when evaluating $f(x+y) - f(y)$ over random y (breaking ties arbitrarily).

The theorem follows immediately from the following two claims showing that ϕ is a homomorphism and that f and ϕ are 2δ-close.

Claim 4.1 $|\{y|f(y) = \phi(y)\}| \geq (1-2\delta)|G|$.

Proof (of Claim 4.1): Let $B = \{x \in G : \Pr_y[f(x) \neq f(x+y) - f(y)] > 1/2]\}$. If $x \notin B$, then $\phi(x) = f(x)$. Thus, it suffices to bound $\frac{|B|}{|G|}$. If $x \in B$, then $\Pr_y[f(x) + f(y) \neq f(x+y)] > 1/2$. Thus $\delta = \Pr_{x,y}[f(x) \neq f(x+y) - f(y)] \geq \frac{|B|}{|G|} \cdot \frac{1}{2}$ or equivalently $\frac{|B|}{|G|} \leq 2\delta$. ■

Claim 4.2 *If $\delta < 2/9$, then $\forall x, z,\ \phi(x) + \phi(z) = \phi(x+z)$.*

Proof (of Claim 4.2): Fix x, we first show that most pairs y_1, y_2 agree to vote for the same value of $\phi(x)$. Pick random $y_1, y_2 \in G$, and we have:

$$\Pr_{y_1,y_2}[f(x+y_1) - f(y_1) = f(x+y_2) - f(y_2)] = \Pr_{y_1,y_2}[f(x+y_1) + f(y_2) = f(x+y_2) + f(y_1)].$$

Both $x+y_1$ and y_2 are uniformly distributed elements of G. Thus, we have $\Pr_{y_1,y_2}[f(x+y_1)+f(y_2) \neq f(x+y_1+y_2)] = \delta < 2/9$. Similarly, we have $\Pr_{y_1,y_2}[f(x+y_2)+f(y_1) \neq f(x+y_1+y_2)] = \delta < 2/9$. If neither of the above events happens, then $f(x+y_1) - f(y_1) = f(x+y_2) - f(y_2)$. Via a union bound we have that

$$\Pr_{y_1,y_2}[f(x+y_1) - f(y_1) = f(x+y_2) - f(y_2)] > 1 - 2\delta \geq 5/9.$$

It is straightforward to show that for any distribution in which the collision probability is at least 5/9, the maximum probability element must have probability at least 2/3. Thus,

$$\forall x \in G, \quad \Pr_y[\phi(x) \neq f(x+y) - f(y)] < 1/3. \tag{1}$$

To show that for all $x, z \in G$, $\phi(x) + \phi(z) = \phi(x+z)$, fix x and z. Then apply Equation (1) to x, z and $x+z$ to get

$$\begin{aligned} \Pr_y[\phi(x) \neq f(x+(y-x)) - f(y-x)] &< 1/3 & (2)\\ \Pr_y[\phi(z) \neq f(z+y) - f(y)] &< 1/3 & (3)\\ \Pr_y[\phi(x+z) \neq f((x+z)+(y-x)) - f(y-x)] &< 1/3 & (4)\end{aligned}$$

Thus $\Pr_y[\phi(x) = f(y) - f(y-x)$ and $\phi(z) = f(z+y) - f(y)$ and $\phi(x+z) = f(z+y) - f(y-x)] > 0$, and so there exists a y for which

$$\phi(x) + \phi(z) = (f(y) - f(y-x)) + (f(z+y) - f(y)) = f(z+y) - f(y-x) = \phi(x+z).$$

The above equality holds for every $x, z \in G$, showing that ϕ is a homomorphism and completing the proof of Claim 4.2. ■ ■

A word about self-correcting In the proof, we note that ϕ is defined so that it is the "self-correction" of f. Observe that there is a simple randomized algorithm that computes $\phi(x)$ given oracle access to f: pick $O(\log 1/\beta)$ values y, compute $f(x+y) - f(y)$ and output the value that you see most often. If f is $\frac{1}{8}$-close to a homomorphism ϕ, then since both y and $x+y$ are uniformly distributed, we have that for at least 3/4 of the choices of y, both of the equalities $\phi(x+y) = f(x+y)$ and $\phi(y) = f(y)$ hold. In this latter case, $f(x+y) - f(y) = \phi(x)$. Thus it is easy to show that if f is $\frac{1}{8}$-close to a homomorphism ϕ, then the above algorithm will output $\phi(x)$ with probability at least $1 - \beta$.

4.2 Other algebraic functions

It is natural to wonder what other classes of functions have robust characterizations as in the case of homomorphisms? There are many other classes of functions that are defined via characterizations that are local. The field of functional equations is concerned with the prototypical problem of characterizing the set of functions that satisfy a given set of properties (or functional equations). For example, the class of functions of the form $f(x) = \tan Ax$ are characterized by the functional equation

$$\forall x, y,\ f(x+y) = \frac{f(x) + f(y)}{1 - f(x)f(y)}.$$

The above functional equation is referred to as an *addition theorem*, as it relates the values of the function on $f(x), f(y)$ and $f(x+y)$. Such addition theorems characterize trigonometric and hyperbolic functions. D'Alembert's equation

$$\forall x, y,\ f(x+y) + f(x-y) = 2f(x)f(y)$$

characterizes the functions $0, \cos Ax, \cosh Ax$. Multivariate polynomials of total degree d over $\mathcal{Z}_p$ for $p > md$ can be characterized by the equation

$$\forall \hat{x}, \hat{h} \in Z_p^m, \sum_{i=0}^{d+1} \alpha_i f(\hat{x} + i\hat{h}) = 0,$$

where $\alpha_i = (-1)^{i+1}\binom{d+1}{i}$. All of the above characterizations are known to be (ϵ, δ)-robust for ϵ and δ independent of the domain size (though for the case of polynomials, there is a polynomial dependence on the total degree d) thus showing that the corresponding properties are easily testable [115, 116]. A long series of works have given increasingly robust characterizations of functions that are low total degree polynomials (cf. [17, 106, 18, 109, 8, 83, 79]). In [116, 115], the general question of what characteristics of functional equations influence their robustness was considered.

We note that all of these results can be extended to apply over domains that are subsets of infinite cyclic groups. They can further be extended to the case of computation with finite precision, which requires that one address the stability of functional equations [54, 86].

Binary operations The above notions of characterizations of properties of functions can be applied to binary operations as well. Assume you are given an $n \times n$ table T of a binary operation $\circ$ on a domain $\mathcal{D}$ of size n – that is, $T(a, b) = a \circ b$ for all $a, b \in \mathcal{D}$. Is $\circ$ a group operation? In [53] it is shown that $O(n^{3/2} \operatorname{polylog} n)$ queries suffice for the property testing problem. Note that the running time depends on the domain size, but is sublinear in n^2, the size of the table describing $\circ$.

The characterization of $\circ$ that is used for property testing requires the following of $\circ$: (i) $\circ$ is close to some cancellative associative operation $\circ'$ (in a cancellative operation, each row and column of the table describing the operation is a permutation of the elements in the domain), (ii) $\circ'$ has an identity element, and (iii) each element in the domain has an inverse under $\circ'$. The main challenge is to find a characterization of associativity that is robust. The result is shown via a probabilistic proof, in a similar style to the above proof of the homomorphism test. However, it is not clear if this result is optimal or even if any dependence on n is necessary for this task.

Problem 1 *Given an $n \times n$ table describing a binary operation $\circ$ over a domain $\mathcal{D}$, can one distinguish the case when $\circ$ is a group operation from the case when $\circ$ is not ϵ-close to a group operation in time independent of n?*

Convolutions of distributions We now turn to a seemingly unrelated question about distributions that are close to their self-convolutions: Let $A = \{a_g | g \in G\}$ be a distribution on group G. The convolution of distributions A, B is

$$C = A * B,\ c_x = \sum_{y,z \in G;\ yz=x} a_y b_z.$$

Let A' be the *self-convolution* of A, $A * A$, i.e. $a'_x = \sum_{y,z \in G; yz=x} a_y a_z$. It is known that $A = A'$ exactly when A is the uniform distribution over a subgroup of G. Suppose we know that A is close to A', can we say anything about A in this case? Suppose $dist(A, A') = \frac{1}{2}\sum_{x \in G} |a_x - a'_x| \leq \epsilon$ for small enough ϵ. Then [31] show that A must be close to the uniform distribution over a subgroup of G. More precisely, in [31] it is shown that for a distribution A over a group G, if $dist(A, A') = \frac{1}{2}\sum_{x \in G} |a_x - a'_x| \leq \epsilon \leq 0.0273$, then there is a subgroup H of G such that $dist(A, U_H) \leq 5\epsilon$, where U_H is the uniform distribution over H [31]. On the other hand, in [31] there is an example of a distribution A such that $dist(A, A * A) \approx .1504$, but A is not close to uniform on any subgroup of the domain.

A weaker version of this result was used to prove a preliminary version of the homomorphism testing result in [37]. To give a hint of why one might consider the question on convolutions of distributions when investigating homomorphism testing, consider the distribution A_f achieved by picking x uniformly from G and outputting $f(x)$. It is easy to see that the error probability δ in the homomorphism test is at least $dist(A_f, A_f * A_f)$. The other, more useful, direction is less obvious. In [31] it is shown that this question on distributions is "equivalent" in difficulty to homomorphism testing:

Theorem 3 *Let G, H be finite groups. Assume that there is a parameter β_0 and function ϕ such that the following holds:*

> *For all distributions A over group G, if $dist(A * A, A) \leq \beta \leq \beta_0$ then A is $\phi(\beta)$-close to uniform over a subgroup of G.*

Then, for any $f : G \to H$ and $\delta < \beta_0$ such that $1 - \delta = Pr[f(x) * f(y) = f(x * y)]$, and $\phi(\delta) \leq 1/2$, we have that f is $\phi(\delta)$-close to a homomorphism.

4.3 Locally testable error-correcting codes

Recently there has been much activity in the study of locally testable error-correcting codes (LTCs), that is, codes for which a tester can make a number of queries that is independent of the size of the codeword (also referred to as "easily testable" in the property testing terminology that we presented earlier) and determine whether or not an input is close or far from a legal codeword. Such codes were first defined in [116], though predecessors of the definition date back to Babai, Fortnow, Levin, Szegedy [20]. Though a detailed description of the area is beyond the scope of this survey, we briefly mention a few of the recent results in this exciting direction.

There are two broad directions of research in locally testable error-correcting codes. The first direction is the study of how to design LTCs, with the emphasis on maximizing their rate. That is, if a code takes k-bit messages and encodes them as n-bit strings, how small can we make n as a function of k? A systematic investigation of this question was started by Goldreich and Sudan [71]. The current state of the art constructions, which combine Dinur's techniques [49] with those of Ben-Sasson and Sudan [32], achieve $n = k \operatorname{polylog} k$ (see also the work of Meir [99] for an alternate construction.) The following question remains open.

Problem 2 *Are there LTCs with linear rate (i.e., $n = O(k)$)?*

The second direction concerns understanding broad reasons for the ability to locally test codes. That is, what are the properties of codes that imply local testability? For example, are all linear codes locally testable? Examples of results which show that linear codes with large distance and few codewords are locally testable are in [81, 84, 90]. Other broad classes of results show that codes based on low-degree polynomials over finite fields are locally testable [37, 116, 8, 83, 79]. Some attempt to unify this collection of results using the concept of invariance of the code was made in [85].

5 Testing Combinatorial Structures

In this section we discuss testing properties of combinatorial structures. Though a wide variety of combinatorial structures have been studied, we will focus our attention on graphs, and describe two models that have received a lot of attention; the case of dense graphs, and the case of bounded-degree graphs. Intermediate models appropriate for testing properties of general graphs have also been investigated (cf. [9, 91, 30]). Property testers for combinatorial properties of matrices, strings, metric spaces and geometric objects have been devised.

5.1 Testing dense graphs

As we have previously mentioned, when considering a testing problem we first need to define how objects are represented and how can the testing algorithm gain access to this representation, as well as what does it mean for an object to be ϵ-far from satisfying a property. For dense graphs, the most common representation is via an *adjacency matrix*. That is, for an n vertex graph G, its adjacency matrix $A = A(G)$ is the $n \times n$ matrix whose (i, j)'th entry is 1 if (i, j) is an edge of G and 0 otherwise. In this subsection we consider testing graphs in the *adjacency matrix model*, in which a graph G is given as input to the testing algorithm represented via an adjacency matrix A. The tester can make an *edge query* – namely, to query what is the (i, j) entry of A, or equivalently if i and j have an edge connecting them in G. We define two graphs G, G' on n vertices to be ϵ-far from each other if one has to add/delete at least ϵn^2 edges in G in order to make it isomorphic to G'. We say that a graph G is *ϵ-far* from satisfying a property $\mathcal{P}$ if it is ϵ-far from any graph satisfying $\mathcal{P}$. This model of property testing is appropriate for studying dense graphs since the definition of what it means to be ϵ-far essentially assumes that the graph has at least ϵn^2 edges. Note that if the graph has $o(n^2)$ edges then an algorithm that makes a constant number of edge queries is unlikely to detect any edge. Let us say that a graph property $\mathcal{P}$ is *easily testable*, if for any $\epsilon > 0$ there is a testing algorithm for $\mathcal{P}$ whose number of queries depends only on ϵ and is independent of size of the input.

The above model of property testing was introduced in 1996 by Goldreich, Goldwasser and Ron [66]. Before discussing the main results of [66], let us start by describing some implicit results that actually preceded [66], which were not even stated as algorithms, let alone testing algorithms. Our starting point is a 1973 paper of Brown, Erdős and Sós [40] which considers the following extremal problem: what is the maximum number of edges that can be present in a 3-uniform hypergraph[2] on n vertices, if it does not contain 6 vertices spanning 3 edges? Let us denote this quantity by $f(n)$. This question was answered by Ruzsa and Szemerédi [117] who showed that $n^{2-o(1)} < f(n) = o(n^2)$. Both parts of this theorem turned out to be extremely influential results in extremal combinatorics. The proof of the upper bound relies on the the following result, which has ever since became known as the *triangle removal lemma.*

Theorem 4 (Triangle Removal Lemma [117]) *There is a function $q : (0, 1) \mapsto (0, 1)$ satisfying the following: If G is ϵ-far from being triangle-free, then G contains at least $q(\epsilon)n^3$ triangles.*

We claim that the above theorem immediately implies that we can test the property of being triangle-free with a constant number of queries. Indeed, given $\epsilon > 0$, we sample a set S of $9/q(\epsilon)$ triples of vertices of G, and accept the input if and only if none of the triples spans a triangle. We clearly accept every triangle free-graph with probability 1, hence the tester actually has 1-sided error. On the other hand, if G is ϵ-far from being triangle-free then Theorem 4 implies that G has at least $q(\epsilon)n^3$ triangles, so a randomly chosen set of $9/q(\epsilon)$ triples of vertices will contain a triangle with probability at least $1/2$.

The proof of Theorem 4 is based on a (by now) simple application of the regularity lemma of Szemerédi [121], a fundamental result in extremal combinatorics. Since the constants involved in this lemma are huge, the proof of the removal lemma via the regularity lemma only bounds $q(\epsilon)$ from above by a tower of exponents of height polynomial in $1/\epsilon$. Obtaining a better upper bound on $q(\epsilon)$ is a major open problem in extremal combinatorics. One of the reasons is that, as shown by Ruzsa and Szemeredi [117], Theorem 4 can be used to give a simple proof of Roth's Theorem [114], which states that every subset of $[n]$ of size $\Omega(n)$ contains a 3-term arithmetic progression. Therefore, improved bounds on $q(\epsilon)$ could give improved bounds on Roth's Theorem.

If a graph G is ϵ-far from being triangle-free, it clearly contains at least one triangle. Therefore, we know that somewhere in G, there is a small "proof" to the fact that G is not triangle-free (in the form of a triangle). The novelty of Theorem 4 is that it shows that such a proof can be found by sampling a constant number of vertices. Stating this result as a robust characterization (as we have done in previous sections) we can say that one should remove $\Omega(n^2)$ edges from G in order to make it triangle-free if and only if G has $\Omega(n^3)$ triangles.

Let us consider now the property of being k-colorable. Since k-colorability cannot be expressed using a finite number of forbidden subgraphs, it is not clear a-priori that if a graph is ϵ-far from being triangle-free then it necessarily contains even a single non k-colorable subgraph of small size! Another implicit result on testing graph properties was motivated by a question of Erdős, who conjectured that if a graph G is ϵ-far from being k-colorable then it must contain a non-k-colorable subgraph on at most $c(\epsilon)$ vertices. Bollobás, Erdős, Simonovits and Szemerédi [39] confirmed this conjecture in the special case $k = 2$, and Rödl and Duke [110] resolved the conjecture for arbitrary fixed k. An

[2]A k-uniform hypergraph $H = (V, E)$ has a set of vertices V and a set of edges E where every edge $e \in E$ has k distinct vertices of V. So a 2-uniform hypergraph is a (simple) graph.

interesting aspect of the proof of [110] is that they actually established that if G is ϵ-far from being k-colorable then a randomly chosen set of vertices S of size $c(\epsilon)$ spans a non k-colorable graph with high probability. We thus get, using the same argument we have used above in the case of testing triangle-freeness, that the result of [110] implies that k-colorability is testable with a constant number of queries (and 1-sided error). Like the proof of Theorem 4, the proof of [110] relies on the regularity lemma and thus it only supplies a huge upper bound for $c(\epsilon)$ given by a tower of exponents of height polynomial in $1/\epsilon$.

Let us now (finally) discuss the main result of Goldreich, Goldwasser and Ron [66]. A partition problem Φ of order t is defined using a sequence of t positive reals $\alpha_1, \dots, \alpha_t$, satisfying $\sum_i \alpha_i = 1$, as well as $\binom{t}{2} + t$ non-negative reals $\beta_{i,j}$. A graph satisfies Φ if one can partition $V(G)$ into t sets $V_1, \dots, V_t$ such that $|V_i| = \alpha_i n$ and for every $i \leq j$ the number of edges connecting the vertices of V_i to those of V_j is $\beta_{i,j}n^2$. Several basic graph properties can be expressed as partition problems. Some examples are k-colorability, having a clique of a certain size, having a cut with a certain number of edges, etc.

Theorem 5 ([66]) *The property defined by a partition-problem of order t is testable with $O(1/\epsilon^{2t+8})$ queries.*

For the special case of testing k-colorability, the algorithm of [66] works by sampling a set of $O(1/\epsilon^3)$ vertices and checking if it spans a k-colorable graph. Thus the result of [66] implies that if G is ϵ-far then it contains a non k-colorable subgraph of size $O(1/\epsilon^3)$. Recall that we have previously mentioned that both triangle-freeness and k-colorability were (implicitly) shown to be testable by applications of the regularity lemma [121], but with a huge dependence on ϵ. The result of [66] was thus a major breakthrough in two respects; first, it supplied a proof for the testability of k-colorability which avoided the regularity lemma. Second, it dramatically reduced the dependence of the number of queries from a tower of exponents of height polynomial in $1/\epsilon$, to a mere (fixed) polynomial in $1/\epsilon$.

It is natural to ask what is the size of the smallest non-k-colorable subgraph that must exist in any graph that is ϵ-far from being k-colorable. Komlos [88] has shown that when $k = 2$ the answer is $\Theta(\sqrt{1/\epsilon})$. For $k \geq 3$, Alon and Krivelevich [10] have improved the result of [66] by giving an $O(1/\epsilon^2)$ upper bound and a $\Omega(1/\epsilon)$ lower bound. The following is still open:

Problem 3 *What is the query complexity of testing 3-colorability (in dense graphs)?*

Given the surprising "computer science" proof of the testability of k-colorability, which avoided the regularity lemma, it is natural to ask if a similar much more efficient proof can be obtained for the case of testing triangle-freeness. Equivalently, we can ask if one can obtain better bounds for the triangle removal lemma (Theorem 4). Regretfully, [117] proved a $(1/\epsilon)^{\Omega(\log 1/\epsilon)}$ lower bound for $q(\epsilon)$ in Theorem 4. Given the huge gap between the best lower bound, which is just slightly super polynomial, and the best upper bound, which is a tower of exponents, the following is an intruiging open problem:

Problem 4 *Improve the lower/upper bounds for the triangle removal lemma.*

Note that the efficiency of a property tester is not directly related to the complexity of deciding the property exactly: Though k-colorability is easily testable, the property of k-colorability is NP-complete to determine – meaning, that though we know how to verify that a certain coloring is a valid k-coloring, we have no idea how to determine whether a graph has a k coloring in time polynomial in the size of the graph.

Though the proof of correctness of the property testers for k-colorability and triangle freeness that we have previously mentioned are involved, the algorithms themselves are very easy to describe: pick a constant sized random sample of the vertices, query all the edges among this random sample and then output "pass" or "fail", according to whether the graph spanned by the sample satisfies the property. Since the sample is of constant size, the determination of whether the sample satisfies the property could be made in constant time. We refer to such algorithms as "natural algorithms". In fact, all the constant time testers that were designed in the literature are essentially natural algorithms. Modulo a technicality about how the final output decision is made, Goldreich and Trevisan [72] essentially show that for any graph property that is easily testable, the natural algorithm gives a property tester. Thus, *all* easily testable graph properties provably have easy-to-describe algorithms.

The work of [66] sparked a flurry of results in the dense graph model. As we have mentioned before, the triangle removal lemma implies that triangle freeness can be tested with a constant, yet huge, number of queries. This result was extended in [5] who (implicitly) showed that for any H the property of being H-free is easily testable, though again the best known upper bounds are huge. Alon [3] showed that the property of being H-free is easily testable with poly$(1/\epsilon)$ queries and 1-sided error if and only if H is bipartite. This result was extended in [12] to hold for 2-sided error testers as well. The triangle removal lemma was extended in [5] to arbitrary fixed graphs H; that is, their extension states that if G is ϵ-far from being H-free then G has $f(\epsilon)n^h$ copies of H, where $h = |V(H)|$. As above, this result immediately implies that for any fixed H the property of being H-free is testable with a constant number of queries.

A very interesting line of work was initiated in the work of Alon, Fischer, Krivelevich and Szegedy [6], in which they use a "functional" variant of the Szemerédi Regularity Lemma to show that the property of a graph being induced H-free (that is, the graph does not contain any *induced* copy of H as a subgraph) is easily testable for any constant sized graph H. The property tester given has an outrageous dependence on ϵ. It was later shown in [14] that testing induced H-freeness indeed requires a super-polynomial number of queries for (almost) any H.

Very recently, the above lines of work culminated in a result of Alon and Shapira [15] who have shown that one can *completely* characterize the class of graph properties that are easily testable with 1-sided error in the dense graph model using a "natural algorithm" as defined above[3]. Before describing their result, we make two definitions. A graph property P is *hereditary* if it is closed under the removal of vertices (but not necessarily under the removal of edges). A graph property P is *semi-hereditary* if there is a hereditary graph property H such that (1) any graph satisfying P also satisfies H and (2) for any $\epsilon > 0$, there is an $M(\epsilon)$, such that any graph G of size at least $M(\epsilon)$ which is ϵ-far from satisfying P does not satisfy H. The result of [15] is then the following:

Theorem 6 ([15]) *A graph property P is easily testable with* 1*-sided error using a "natural algorithm" if and only if P is semi-hereditary.*

[3]See the discussion in [15] as to why it is natural to consider "natural algorithms" in this context.

Hereditary graph properties include all monotone graph properties (including k-colorability and H-freeness), as well as other interesting non-monotone graph properties such as being a perfect, chordal, or interval graph. The techniques used in [15] are somewhat involved, and were based on the new functional variant of the Szemerédi Regularity Lemma that was developed in [6]. Previously in the literature, the "Regularity Lemma" type arguments were used to develop testers for graph properties that were characterized by a finite set of forbidden subgraphs. Here the set of forbidden subgraphs may be infinite, and they are forbidden as *induced* subgraphs. Theorem 6 was later reproved by Lovász and Szegedy [96] using the machinery of graph limits. It was also extended to hereditary properties of hypergraph by Rödl and Schacht [111] and by Austin and Tao [19].

The bounds supplied by Theorem 6 are huge. It is thus natural to ask if better bounds could be obtained for specific hereditary graph properties. We remind the reader that the result of [14] supplies a (nearly complete) characterization of the hereditary graph properties expressible by single forbidden induced subgraphs, and can be tested using $\mathrm{poly}(1/\epsilon)$ queries. But for many hereditary graph properties that can not be so expressed, the question of whether they can be tested with $\mathrm{poly}(1/\epsilon)$ queries is still open. For example, this question is open for graph properties that include being Perfect, Chordal or Interval. More ambitiously, one could try and resolve the following problem:

Problem 5 *Characterize the hereditary graph properties that are testable with* $\mathrm{poly}(1/\epsilon)$ *queries.*

The hierarchy theorem in computation complexity is a basic result that says that there are problems that can be solved within certain time/space constrains but cannot be solved with significantly less time/space. It is natural to ask if a similar phenomenon holds in testing graphs (or other structures). Goldreich et. al. [67] have recently shown that for any function $q(n)$ there is a graph property that can be tested with $q(n)$ queries but cannot be tested with $o(q(n))$ queries. Since essentially any graph property can actually be tested with a constant number of queries which is independent of n it is natural to ask if for any function $q(\epsilon)$ there is a graph property that can be tested with $q(\epsilon)$ queries but cannot be tested with $o(q(\epsilon))$. Somewhat surprisingly even the following problem is open

Problem 6 *Is there an easily testable graph property, whose query complexity is* $2^{\Omega(1/\epsilon)}$.

The fact that this problem is open is quite remarkable given the fact that the only known upper bounds for testing many easily testable properties are those given by the general result in Theorem 6 which are given by towers of exponents of height polynomial in $1/\epsilon$. The best lower bounds for testing an easily testable graph property where given in [12], where it is shown that for every non-bipartite H the property of being H-free has query complexity $(1/\epsilon)^{\Omega(\log 1/\epsilon)}$.

The only result similar in spirit to Problem 6 appears in [13] where it is shown that for every function $q(\epsilon)$ there is an easily testable graph property $\mathcal{P}_q$ that cannot be tested with 1-sided error with less than $q(\epsilon)$ queries. We note that for every q the upper bound for testing $\mathcal{P}_q$ is much (much) larger than $q(\epsilon)$, therefore this theorem does not give a (tight) hierarchy theorem for 1-sided error testers. Perhaps proving such a hierarchy theorem could be a step towards resolving Problem 6.

There are many interesting properties that are not easily testable, but do have sublinear time property testers. For example, the graph isomorphism problem asks whether two graphs are identical under relabeling of the nodes. In [60], it is shown that the property testing problem requires $\Omega(n)$ queries and that there is a property tester for this problem which uses $O(n^{5/4}\,\mathrm{polylog}\,n)$ queries, which is sublinear in the input size n^2.

5.2 Testing bounded degree graphs

We now turn our focus to the problem of testing properties in bounded degree graphs. We will see that the testability of a problem is very sensitive to the model in which it is being tested. In this model we fix an integer d which is an upper bound on the degrees of vertices in G. Graphs are represented using adjacency lists, that is, each vertex has a list containing its neighbors in G (in an arbitrary order). The tester can access the representation of G by asking what is the i^{th} neighbor of a vertex v. We say that an n vertex graph G of bounded degree d is ϵ-far from another such graph if one needs to add/delete at least ϵdn edges in order to make G isomorphic to G'. We say that G is ϵ-far from satisfying a property $\mathcal{P}$ if G is ϵ-far from any graph of bounded degree d that satisfies $\mathcal{P}$.

The model of testing graphs of bounded degree was defined and first studied by Goldreich and Ron [69]. As in the case of testing dense graphs, implicit results related to testing bounded-degree graphs actually appeared before the model was explicitly defined. Recall the result of Rödl and Duke [110], which we have previously mentioned, that states that if a dense graph is ϵ-far from being triangle-free then it contains a non k-colorable subgraph on $c(\epsilon)$ vertices. It is natural to ask if the same is true in bounded degree graphs. It is easy to see that the answer is negative. Even when $k = 2$ a d-regular expander of logarithmic girth is far from being bipartite, since in any bipartition the larger of the two sets contains a linear number of edges, while any set of $O(\log n)$ vertices has no cycles, and is in particular bipartite. This implies that we cannot hope to test 2-colorability in bounded-degree graphs simply by "looking" for a small non 2-colorable subgraph. In particular this means that we cannot test 2-colorability using $o(\log n)$ queries and 1-sided error. As it turns out, if G is far from being bipartite then it must contain an odd cycle of length $O(\log n)$.

It is natural to ask how does the answer to the above question change for $k > 2$. Somewhat surprisingly it turns out that there are graphs that are far from being 3-colorable yet *all* sets of vertices of size $\Omega(n)$ span a 3-colorable graph. Actually, for any $k \geq 3$ there are graphs that are far from being k-colorable yet *all* sets of vertices of size $\Omega(n)$ span a 3-colorable graph. This result was proved in 1962 by Erdős [52] and it implies that when considering bounded degree graphs there is a much weaker relation between the local and the global properties of a graph. In particular, it means that any 1-sided error tester for 3-colorability has query complexity $\Omega(n)$.

Getting back to [69], the main result of Goldreich and Ron was that any testing algorithm for 2-colorability (even one with 2-sided error) must have query complexity $\Omega(\sqrt{n})$. This is a significant improvement over the $\Omega(\log n)$ bound we have mentioned before which holds only for 1-sided error testers. Goldreich and Ron [70] later proved that this $\Omega(\sqrt{n})$ lower bound is essentially tight by giving a testing algorithm with query complexity $\tilde{O}(\sqrt{n})$ which finds (with high probability) an odd cycle in any graph that is far from being bipartite. An intriguing aspect of the result of [70] was that as opposed to testing dense graphs, in which the algorithm itself is simple, the bipartiteness testing algorithm of [70] is non-trivial. Similar ideas where later used in testing other properties of bounded degree graphs [46, 80, 68, 101].

Let us mention that Bogdanov, Obata and Trevisan [38] have shown that the $\Omega(n)$ lower bound for testing 3-colorability with 1-sided error, which we have mentioned above, holds also for 2-sided error testers. Therefore, the trivial algorithm which simply asks about all edges of G is essentially the most efficient one.

The fact that even 2-colorability cannot be tested with a constant number of queries in bounded-degree graphs means that we cannot hope to obtain efficient testing algorithms for arbitrary hereditary properties as in the case of dense graphs. For a long time there was no general result on testing properties of sparse graphs similar in nature to Theorem 6. In addition to the above mentioned results on k-colorability, the only other results on testing bounded-degree graphs appears in [69], where it is shown that k-connectivity (for any fixed k), being Eulerian and being Cycle-free are testable in bounded-degree graphs with a constant number of queries. A question that is raised in [69] is whether planarity is testable in a bounded degree graph with a constant number of queries. In addition to being one of the most well studied graph properties, this problem was interesting for two other reasons: first, planarity is "trivially" testable in dense graphs since it is well known that a graph with more than $3n - 6$ edges is not planar. Therefore, it is natural to study the testability in the more appropriate model of bounded degree graphs. Second, as in the case of 2-colorability, it is easy to see that planarity cannot be tested in bounded degree graphs with 1-sided error and $o(\log n)$ queries. This observation follows from considering high girth expanders. These graph are far from being planar since they are far from having small separators, and planar graphs have small separators by the result of Lipton and Tarjan [95]. On the other hand, these graphs do not contain small non-planar graphs since they have large girth. It was thus natural to ask if planarity can be tested with 2-sided error and with a constant number of queries, since previously there was no hereditary property that could be tested with 2-sided error (much) more efficiently than it could be tested with 1-sided error. In a recent paper, Benjamini, Schramm and Shapira [33] resolved the above open problem by showing that planarity is testable with a constant number of queries. More generally, the main result of [33] is that any minor-closed[4] property is testable in bounded-degree graphs. The query complexity of the algorithm presented in [33] for testing planarity was $2^{2^{2^{\mathrm{poly}(1/\epsilon)}}}$. This was recently improved in [76] to $2^{\mathrm{poly}(1/\epsilon)}$. The following problem, however, is still open:

Problem 7 *Can planarity be tested with poly$(1/\epsilon)$ queries?*

Recall that we have argued above that planarity cannot be tested with $o(\log n)$ queries and with 1-sided error. Moreover, it can be shown that planarity cannot be tested with 1-sided error and $o(\sqrt{n})$ queries. This suggests the following problem:

Problem 8 *Can planarity be tested with $\tilde{O}(\sqrt{n})$ queries and 1-sided error?*

A number of interesting recent results have given specialized property testing algorithms which are significantly more efficient for any class of graphs with a fixed excluded minor [48, 33, 51].

5.3 Combinatorial approach to testing Boolean functions

As we have discussed in Section 4, many properties of Boolean functions have been shown to be testable. The techniques involved in almost all these results are algebraic in nature, applying methods ranging from Fourier analysis to algebraic properties of error correcting codes. Recently, methods that were "classically" applied to test properties of graphs were applied to testing properties of Boolean functions. Let us give a brief account of these developments.

[4]A graph property is minor closed if it is closed under the removal of edges and vertices, as well as under the contraction of edges.

As opposed to graphs, where we have a reasonably good understanding of the properties that can be tested with a constant number of queries, our understanding of the properties of Boolean functions that can be so tested is much more limited. Following a work of Kaufman and Sudan [85], Bhattacharyya, Chen, Sudan and Xie [34] conjectured that a certain family of properties of Boolean functions can be efficiently tested. The original formulation of their conjecture is somewhat involved, but as observed in [119], the conjecture of [34] is a special case of a slightly easier to formulate problem. Fix a set of ℓ linear equations in k unknowns $Mx = b$ over a finite field $\mathbb{F}$ (suppose $rank(M) = \ell$). We say that a subset $S \subseteq \mathbb{F}$ is (M, b)-free if S contains no solution to $Mx = b$. In other words, there is no vector $v \in S^k$ satisfying $Mv = b$. We also say that S is ϵ-far from being (M, b)-free if one must remove at least $\epsilon|\mathbb{F}|$ elements from S in order to get a set which is (M, b)-free. Finally, we say that a set of equations $Mx = b$ is *nice* if the following holds; if a subset $S \subseteq \mathbb{F}$ is ϵ-far from being (M, b)-free then S contains at least $\delta(\epsilon)n^{k-\ell}$ solutions to $Mx = b$, where $\delta(\epsilon) > 0$ for every $\epsilon > 0$ and depends only on ϵ.

Observe that if $Mx = b$ is nice, then one can quickly test if S is (M, b)-free. Indeed, given S we uniformly and independently sample $8/\delta(\epsilon)$ vectors satisfying $Mx = b$, and reject if and only if one of the vectors belongs to S^k. If S is (M, b)-free then we clearly accept with probability 1. On the other hand, if S is ϵ-far from being (M, b)-free then at least $\delta(\epsilon)n^{k-\ell}$ of the $n^{k-\ell}$ vectors satisfying $Mx = b$ belong to S^k. Therefore, sampling $8/\delta(\epsilon)$ of these vectors guarantees that we find such a vector with probability at least $3/4$.

The main result of [119] was that any set of linear equations is nice. This result implies that the properties of Boolean functions studied in [34] are indeed testable, thus confirming their conjecture. The main tool used in [119] in order to prove this result is the so called hypergraph removal lemma. This result is the (far reaching) hypergraph variant of the triangle removal lemma, which we stated in Theorem 4. This result was obtained recently by Gowers [73], Kohayakawa, Nagle, Rödl, Schacht and Skokan [87] and Tao [123]. As we have discussed in the previous subsections, these types of results were very useful for showing the testability of graph properties, but as it turns out, they seem to be useful also for studying properties of Boolean functions. Green [74] has shown that if $Mx = b$ consists of a single equation then it is nice. He also conjectured that every set of linear equations $Mx = b$ is nice. Green's results on the niceness of single equations were later reproved by Král', Serra and Vena [93] using a very short a elegant application of the graph removal lemma. Král', Serra and Vena [94] later extended their result from [93] and independently obtained a result equivalent to the one in [119].

Many open problems related to this new approach are mentioned in [119]. Since the statement of some of them requires some preparation, we refrain from quoting them here and refer the interested reader to [119].

We conclude with one open problem which is similar in some aspects to the investigation of the triangle removal lemma mentioned earlier. Suppose we consider the case when the field we work over is $\mathbb{F}_2^n$ and we consider a single linear equation $x + y = z$. Then the above result of Green [74] (as well as the later proofs) imply that if $S \subseteq \mathbb{F}_2^n$ is such that one should remove at least $\epsilon 2^n$ of the elements of S in order to make S free of solutions to $x + y = z$, then S contains at least $\delta(\epsilon)2^{2n}$ such solutions. However, as in the case of the triangle removal lemma, the bound one obtains for $\delta(\epsilon)$ is a function which approaches 0 extremely fast. This motivated Green [74] to raise the following natural problem as to the possibility of obtaining polynomial bounds.

Problem 9 *Suppose one has to remove at least $\epsilon 2^n$ of the elements of subset $S \subseteq \mathbb{F}_2^n$ in order to destroy all solutions to $x + y = z$, with $x, y, z \in S$. Is it true that S contains at least $\epsilon^C 2^{2n}$ such solutions, where C is an absolute constant?*

This problem has been investigated recently by Bhattacharyya and Xie [36] who obtained some lower bounds for C. The problem, however, remains open.

6 Sublinear time approximation algorithms

Alongside the property testing results, many sublinear time approximation schemes have been developed for classical optimization problems. Some examples of such optimization problems include problems on graphs, metric spaces [78, 42], clustering [100], bin packing [21], and weak approximations for string compressibility [107, 108] and edit distance [23]. In the following, we will give a few intriguing examples of sublinear algorithms which approximate the solutions to graph problems.

6.1 Cut and partition problems on dense graphs

Recall that testing algorithms try to distinguish between objects satisfying a property from those that are far from satisfying it. It is natural to ask if the ideas used in the design/analysis of testing algorithms can then be used in order to design algorithms that approximate the distance of an object from one that satisfies the property or approximate any other related quantity. The first example of a result of this nature already appeared in [66] where it was shown that one can approximate the size of the largest cut in a dense graph up to an error ϵn^2 in time $O(\frac{\log^2(1/(\epsilon))}{\epsilon^7})$. A similar result was also obtained by Frieze and Kannan [59]. It was also shown in [66] that in time $2^{poly(1/\epsilon)} + n \cdot poly(1/\epsilon)$ one can actually find a partition of an input graph which approximates the maximum cut up to an error ϵn^2. Note that this running time is sublinear in the size of the graph, which is $O(n^2)$. Actually, such an algorithm can be designed for all of the partition problems discussed before the statement of Theorem 5. We note that the main idea behind these approximation algorithms is to use the analysis of the respective property testing algorithm. Perhaps the most general (and surprising) result of this type was obtained by Fischer and Newman [62] who have shown that in dense graphs, if a graph property is easily testable, then one can also estimate how far (in terms of the number of edge modifications required) is an input graph from satisfying the property, up to an additive error of δn^2, in time that depends only on δ.

Recently the algorithm of [66] for approximating graph partition problem was extended by Fischer, Matsliach and Shapira [61] to the setting of k-uniform hypergraphs. Besides being a natural generalization of the result of [66], the hypergraph partition algorithm can be used in order to construct a regular partition of an input graph (in the sense of Szemerédi's regularity lemma). This algorithm improved over several previous algorithms both in terms of the running time and in terms of the "quality" of the partition it returns.

6.2 Connectivity and Minimum Spanning tree on sparse graphs

A second set of examples of problems that have been conducive to sublinear time algorithms are the problems of estimating the number of connected components and the minimum spanning tree (MST) weight of a graph. Here we assume that the graph is sparse and hence is represented in the adjacency list format. As we will see, the problems of estimating the number of connected components and of estimating the MST weight are related since one may use algorithms for estimating the number of connected components in order to estimate the minimum spanning tree weight.

Estimating the number of connected components Building on an idea of Goldreich and Ron [69] in their tester for graph connectivity property, Chazelle, Rubinfeld and Trevisan [43] show that it is possible to additively estimate the number of connected components in a bounded degree graph to within ϵn in time independent of the number of nodes:

Theorem 7 *[43] Let c be the number of components in a graph with n vertices and maximum degree d. Then there is an algorithm which runs in time $O(d\epsilon^{-2}\log\frac{d}{\epsilon})$ and with probability at least $3/4$ outputs $\hat{c}$ such that $|c-\hat{c}| \le \epsilon n$.*

(Note that similar results which depend on the average degree rather than the maximum degree have been shown in [43] as well.) The main idea in estimating the sizes of the components is as follows: For vertex u, let d_u denote its degree and m_u denote the number of edges in its component. Let c denote the number of connected components in graph G with vertex set V. Then, observe that for every set of vertices $I \subseteq V$ that forms a connected component, we have $\sum_{u\in I}\frac{d_u}{2m_u} = 1$ and $\sum_{u\in V}\frac{d_u}{2m_u} = c$.[5] The value of c can then be quickly approximated via this sum by performing two types of approximations: The first is that instead of computing the sum over all u of $\frac{d_u}{2m_u}$, we need only compute the average value of a sample of random vertices u of $\frac{d_u}{2m_u}$, and this can be used to give a good approximation of the original sum using standard sampling arguments. The second is that for specific values of u, we can estimate $\frac{d_u}{2m_u}$ rather than compute it exactly. Here we use the fact that since we are aiming for an additive approximation, one does not need to do a good job on estimating m_u for large components, say of size larger than d/ϵ^2, and in such a case, even 0 gives a good approximation to $d_u/2m_u$. Furthermore, for all smaller components, one can exactly compute m_u very quickly, in $O(d/\epsilon^2)$ time.

Employing other tricks to reduce the running time, a nearly optimal algorithm is given for achieving an additive estimate to within ϵn of the number of connected components of a graph. In [43], it is shown that $\Omega(d\epsilon^{-2})$ edge lookups are required, assuming $C/\sqrt{n} < \epsilon < 1/2$, for some large enough constant C. Though nearly tight, the following question is open:

Problem 10 *Is it possible to additively estimate the number of connected components to within ϵ in time $O(d\epsilon^{-2})$?*

Estimating the weight of the minimum spanning tree The above estimation of the number of connected components in a graph is useful for the problem of estimating the weight of a minimum spanning tree. Assume we are given a graph G graph with n vertices, maximum degree d, and that each edge is assigned an integral weight in the range $[1..w]$ (the assumption that weights are integral can be removed with a slight loss in efficiency of the algorithm). Chazelle et. al. [43] show

[5]For isolated vertices, we use the convention that $d_u/m_u = 2$.

that the weight of the minimum spanning tree can be estimated to within ϵn additive error in time independent of n, depending only on d, w and ϵ.

The idea of the algorithm is to show a relationship between the MST of a graph G and the number of connected components of various subgraphs of G: For each $0 \le \ell \le w$, let $G^{(\ell)}$ denote the subgraph of G consisting of all the edges of weight at most ℓ. Define $c^{(\ell)}$ to be the number of connected components in $G^{(\ell)}$ (with $c^{(0)}$ defined to be n). By our assumption on the weights, $c^{(w)} = 1$. Let $M(G)$ be the weight of the minimum spanning tree of G. Then the following lemma relates the weight of the minimum spanning tree to the number of connected components in the subgraphs $G^{(\ell)}$:

Claim 6.1 ([43]) *For integer $w \ge 2$,*

$$M(G) = n - w + \sum_{i=1}^{w-1} c^{(i)} .$$

Thus, to estimate the weight of the MST of G, it suffices to estimate the values $c^{(i)}$.

Theorem 8 ([43]) *Let $w/n < 1/2$. Let v be the weight of the MST of G. Algorithm* approx-mst-weight *runs in time $O(dw\epsilon^{-2} \log \frac{dw}{\epsilon})$ and outputs a value $\hat{v}$ that, with probability at least 3/4, differs from v by at most ϵv.*

In [43], a lower bound of $\Omega(dw\epsilon^{-2})$ is shown (assuming that $w > 1$ and $C\sqrt{w/n} < \epsilon < 1/2$). Although this is nearly tight, it leaves us with the following question:

Problem 11 *Is it possible to approximate the MST of a graph in time $O(dw\epsilon^{-2})$?*

Czumaj et. al. [45] have shown that, when supported with access to the input via certain geometric data structures such as orthogonal range queries and cone approximate nearest neighbor queries, the MST for points in bounded dimensional Euclidean space can be estimated in time $\tilde{O}(\sqrt{n}\text{poly}(1/\epsilon)))$. For points on arbitrary metric spaces, Czumaj and Sohler [47] have shown that one can compute a $(1+\epsilon)$-approximation in $O(n\text{poly}(1/\epsilon))$ time. More recently, some of these ideas have been used to develop streaming algorithms for estimating the weight of the minimum spanning tree in [58].

6.3 Local distributed algorithms vs. sublinear time algorithms for sparse problems

Very recently, much progress has been made in finding sublinear time approximation algorithms for a number of combinatorial optimization problems that can be phrased as packing or covering problems with sparsity conditions. Such problems include vertex cover, dominating set, and maximum matching on sparse graphs, and set cover of sets with certain sparsity conditions. One key insight that led to this flurry of results was given by Parnas and Ron [104], who observed a relationship between local distributed algorithms and sublinear time algorithms. This relationship allows one to take existing very efficient distributed algorithms for degree bounded distributed networks and to use them to construct sublinear time approximation algorithms.

As an example, consider the vertex cover problem on graphs of degree at most d, where d is a constant. Suppose one has an oracle that for any vertex v, tells the vertex cover approximation algorithm whether or not v is in the vertex cover. Then, using standard sampling arguments, one can show that a vertex cover approximation algorithm which samples $O(1/\epsilon^2)$ vertices can get an ϵn additive approximation to the size of the vertex cover. But how does one get access to such an oracle? Parnas and Ron note that if there is a *local* distributed algorithm, that is, a distributed algorithm for vertex cover that runs in a number of rounds which is independent of the number of vertices, then one can use it to construct such an oracle. (Note that in the distributed algorithm, it is assumed that the vertex cover is being computed on the interconnection graph of the underlying distributed network of processors.) The reason is that for constant k, the k-neighborhood of any given vertex v, consisting of those vertices within constant distance k of v, can only be of constant size. Thus, one can simulate the runs of any processor that is near enough to the processor assigned to vertex v so that it might actually affect the computation of v's processor within the time bound. Parnas and Ron achieve an algorithm for approximating vertex cover using the local distributed c-approximation algorithm of Kuhn, Moscibroda and Wattenhoffer [92]. The approximation algorithms achieved in this way are not fully multiplicative nor fully additive approximations, rather they achieve an approximation factor with the following behavior:

Definition 6.2 *We say that $\hat{y}$ is an (α, β)-*approximation *to y if $y \le \hat{y} \le \alpha \cdot y + \beta$.*

Parnas and Ron get a $(c, \epsilon \cdot n)$-approximation algorithm with query complexity independent of n, namely, $d^{O(\log(d/\epsilon^3))}$, where d is a bound on the degree of the graph. By devising an elegant new local distributed algorithm, which in turn uses ideas from Luby's parallel algorithm for the independent set problem [97], Marko and Ron [98] later improved the upper bound to $d^{O(\log(d/\epsilon))}$, where d is a bound on the degree of the graph, giving a $(2, \epsilon \cdot n)$-approximation algorithm for vertex cover.

Since Kuhn, Moscibroda and Wattenhoffer [92] give a local distributed algorithm for any packing or covering problem with certain sparsity requirements, the algorithmic paradigm of Parnas and Ron can be used for a large class of problems.

It is natural to wonder what is the best approximation guarantee α that an $(\alpha, \epsilon n)$-approximation algorithm can achieve in time that is only a function of the maximum degree d and ϵ? Trevisan complements the $(2, \epsilon \cdot n)$-approximation algorithm for vertex cover by showing that there is no such algorithm if α is a constant less than 2 [104]. Alon shows that $\alpha = \Omega(\log d)$ for minimum dominating set, and $\alpha = \Omega(d/\log d)$ for maximum independent set [4]. Alon also gives an $(O(d \log\log d/\log d), \epsilon \cdot n)$-approximation algorithm for maximum independent set.

Very recent work, using the same framework, but constructing a new class of local distributed algorithms for use in approximating these problems, has greatly improved the dependence of the running time on d and ϵ so that it is polynomial. Nguyen and Onak [102] construct an algorithmic technique which simulates the standard sequential greedy algorithm in a distributed and local way. Recall the greedy algorithm for vertex cover which first selects an arbitrary maximal matching and then returns the set of matched vertices as the vertex cover [63]. The maximal matching is in turn selected by considering the edges in an arbitrary order, and adding edges that are not adjacent to edges already in the matching. A naive distributed implementation of this greedy algorithm would be as follows:

1. Randomly assign a distinct rank $r(u, v)$ to each edge (u, v), corresponding to its rank in the

arbitrary order used by the sequential algorithm.

2. Assign a processor to each edge (u, v) of the graph.
3. Each processor for (u, v) considers all edges adjacent to u and v and recursively determines whether any of these edges that have lower rank than $r(u, v)$ have been placed in the matching.
4. If at least one of the adjacent edges is in the matching, then (u, v) is not in the matching, otherwise, (u, v) is placed in the matching.

The correctness of the algorithm follows immediately from the correctness of the sequential greedy algorithm. While termination follows from the fact that the recursion is always on lower ranked edges, showing query complexity bounds which are independent of the number of nodes in the graph is more difficult. In fact, if r were to be chosen adversarially rather than randomly, the query complexity of the algorithm could be very bad. The difficulty in turning greedy algorithms into local or sublinear time sampling algorithms is in that the dependency chains of the recursion can be of linear length. Nguyen and Onak show that if one randomly chooses the ranking function r, the dependency chains of each call are likely to be short. One can bound the expected query complexity of a single oracle call by noting that the probability any recursion path of length k is explored is bounded by the probability that the ranks of the edges are decreasing along the path, which is at most $1/(k+1)!$. The number of neighbors at distance k is at most d^k and thus the expected number of explored vertices is at most $\sum_{k=0}^{\infty} d^k/(k+1)! \leq e^d/d$, so that the expected query complexity is at most $O(e^d)$. Formalizing an argument of this sort over several oracle calls is nontrivial because of dependencies between the queries of the algorithm, however, [102] are able to give such a formalization.

Building on this paradigm, Nguyen and Onak [102] showed that several classical greedy algorithms can be transformed into constant-time approximation algorithms. They give a $(1, \epsilon \cdot n)$-approximation algorithm for maximum matching with running time that only depends on d and ϵ, by locally constructing a sequence of improved matchings. Their algorithm has running time of $2^{O(d)}/\epsilon^2$, which is better than [104, 98] in terms of ϵ but worse in terms of d. Other examples of problems their techniques apply to are vertex cover, set cover, dominating set, and maximum weight matching.

Though the above algorithms are independent of n, as we saw above, the dependence of [104, 98, 102] on other parameters such as d, ϵ, is less attractive and in many cases superpolynomial. However, Yoshida, Yamamoto, and Ito [124] showed that a specific implementation of the method of Nguyen and Onak yields dramatically better dependence of the running times in terms of d and ϵ. In the algorithm for the vertex cover problem, whenever one considers adjacent edges e' with $r(e') < r(e)$ for a given edge e, it suffices to consider them in ascending number of their $r(e')$, and to stop making recursive calls when a neighbor is found to be in the maximal matching being constructed. The running time of the $(2, \epsilon n)$-approximation algorithm for vertex cover can then be bounded by $O(d^4/\epsilon^2)$. For maximum matchings, the running time of the $(1, \epsilon n)$-approximation algorithm becomes $d^{O(1/\epsilon^2)}$. The following question remains open:

Problem 12 *Is there a* $(1, \epsilon n)$*-approximation algorithm for maximum matching with query complexity* $\mathrm{poly}(d/\epsilon)$*?*

For certain classes of graphs, one can sometimes get even better results in terms of both the approximation factors and the running time. As we have mentioned earlier, recent results have given better algorithms, as well as better property testers, for any class of graphs with a fixed excluded minor [48, 33, 51]. Though such graphs may have arbitrary maximum degree, their average degree is bounded by a constant. Very recently, Hassidim, Kelner, Nguyen, and Onak [76] showed that for any class of graphs with a fixed excluded minor there is a $(1, \epsilon n)$-approximation algorithm for vertex cover, dominating set, and maximum independent set, which runs in $2^{\mathrm{poly}(1/\epsilon)}$ time. Their algorithm locally computes a partition of the graph into constant-size connected components by removing a small number of edges. It is known, by the Graph Separator Theorem [95, 11] that such a partition exists after one removes any high degree nodes (of which there are few in a graph with an excluded minor). The removed edges can be covered with a small number of vertices, and thus do not alter the solution size significantly. For the partitioned graph, an optimal solution can be found by computing an optimal solution for each (constant size) connected component independently. The solution size in the modified graph can then be approximated by sampling. It would be interesting to solve the following problem:

Problem 13 *Given an input graph from a family of graphs with an excluded minor, is there a $(1, \epsilon n)$-approximation algorithm for vertex cover with query complexity* $\mathrm{poly}(1/\epsilon)$*?*

7 Some final comments

We have seen several contexts in which one can test properties in sublinear time. We emphasize that we have only touched on a few topics of interest within sublinear time algorithms. There are many recent results giving sublinear time algorithms and property testers for a wide variety of problems.

The study of sublinear time algorithms has led to a new understanding of many problems that had already been well-studied. For example, some of these algorithms have even resulted in better linear time approximation algorithms than what was previously known, such as for the problem of max cut [66] and very recently for the problem of estimating the edit distance [16]. A second example is the use of homomorphism testers as an ingredient in the construction of Probabilistically Checkable Proof Systems, which provide a way to write down a proof so that another person can verify it by viewing only a constant number of locations (cf. [17]). A third example is the impact of sublinear time algorithms on the development of algorithms for streaming data (cf. [58]). Much still remains to be understood about the scope of sublinear time algorithms, and we expect that this understanding will lead to further insights.

Acknowledgements We thank Ran Canetti, Krzysztof Onak, Madhu Sudan and Ning Xie for their extensive and helpful comments on this manuscript.

References

[1] N. Ailon and B. Chazelle. Information theory in property testing and monotonicity testing in higher dimensions. *Inf. Comput.*, 204(11): 1704-1717, 2006.

[2] N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Estimating the distance to a monotone function. Random Struct. Algorithms 31(3). pp. 371-383 (2007).

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

[4] N. Alon, On constant time approximation of parameters of bounded degree graphs, manuscript, 2010.

[5] N. Alon, R. A. Duke, H. Lefmann, V. Rdl and R. Yuster, The algorithmic aspects of the Regularity Lemma, J. of Algorithms 16 (1994), 80-109.

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

[7] N. Alon, E. Fischer, I. Newman and A. Shapira, A combinatorial characterization of the testable graph properties: it's all about regularity, SIAM Journal on Computing, to appear.

[8] N. Alon, T. Kaufman, M. Krivilevich, S. Litsyn, and D. Ron. Testing low-degree polynomials over gf(2). In *Proceedings of RANDOM '03*, pages 188–199, 2003.

[9] N. Alon, T. Kaufman, M. Krivelevich, D. Ron. Testing Triangle-Freeness in General Graphs. *SIAM J. Discrete Math.* 22(2), pp. 786-819 (2008).

[10] N. Alon and M. Krivelevich, Testing k-colorability, SIAM J. Discrete Math. 15 (2002), 211-227.

[11] N. Alon, P. Seymour and R. Thomas, "A separator theorem for graphs with an excluded minor and its applications", in STOC 1990, pp. 293-299.

[12] N. Alon and A. Shapira, Testing Subgraphs in Directed Graphs, Journal of Computer and System Sciences, 69 (2004), 354-382.

[13] N. Alon and A. Shapira, Every monotone graph property is testable, SIAM Journal on Computing, 38 (2008), 505-522.

[14] N. Alon and A. Shapira, A characterization of easily testable induced subgraphs, Combinatorics, Probability and Computing, 15 (2006), 791-805

[15] N. Alon and A. Shapira, A characterization of the (natural) graph properties testable with one-sided error, SIAM Journal on Computing, 37 (2008), 1703-1727.

[16] A. Andoni, R. Krauthgamer and K. Onak. The query complexity of edit distance. Manuscript.

[17] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[18] S. Arora and M. Sudan. Improved low degree testing and its applications. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 485–495, 1997.

[19] T. Austin and T. Tao, On the testability and repair of hereditary hypergraph properties, manuscript, 2008.

[20] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy Checking Computations in Polylogarithmic Time. STOC 1991, pp. 21-31.

[21] T. Batu, P. Berenbrink, C. Sohler, A sublinear-time algorithm for bin packing. Theoretical Computer Science. 2009.

[22] T. Batu, S. Dasgupta, R. Kumar, and R. Rubinfeld. The complexity of approximating the entropy. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 678 – 687, 2002.

[23] T. Batu, F. Ergun, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, and Rahul Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 316–324, 2003.

[24] T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. In *Proc. 42nd IEEE Conference on Foundations of Computer Science*, pages 442–451, 2001.

[25] T. Batu, L. Fortnow, R. Rubinfeld, W. Smith, and P. White. Testing that distributions are close. In *Proceedings of the Forty-First Annual Symposium on Foundations of Computer Science*, pages 259–269, 2000.

[26] T. Batu, R. Kumar, and R. Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 381–390, 2004.

[27] T. Batu, R. Rubinfeld, and P White. Fast approximate PCPs for multidimensional bin-packing problems. *Inf. Comput.* 196(1). pp. 42-56, 2005.

[28] F. A. Behrend, On sets of integers which contain no three terms in arithmetic progression, Proc. National Academy of Sciences USA 32 (1946), 331-332.

[29] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing over characteristic two. *IEEE Transactions on Information Theory*, 42(6):1781–1795, 1996.

[30] I. Ben-Eliezer, T. Kaufman, M. Krivelevich, D. Ron. Comparing the strength of query types in property testing: the case of testing k-colorability. *SODA 2008* pp. 1213-1222.

[31] M. Ben-Or, D. Coppersmith, M. Luby, and R. Rubinfeld, Non-abelian homomorphism testing, and distributions close to their self-convolutions. Random Struct. Algorithms 32(1): 49-70 (2008)

[32] E. Ben-Sasson and M. Sudan. Short PCPs with Polylog Query Complexity. SIAM J. Comput. 38(2), pp. 551-607, 2008.

[33] I. Benjamini, O. Schramm and A. Shapira, Every minor-closed property of sparse graphs is testable, Proc. of STOC 2008, 393-402.

[34] A. Bhattacharyya, V. Chen, M. Sudan and N. Xie, Testing linear-invariant non-linear properties, manuscript, 2008.

[35] A. Bhattacharyya, E. Grigorescu, K. Jung, S. Raskhodnikova, and D. P. Woodruff. Transitive-closure spanners. SODA 2009. pp. 932-941.

[36] A. Bhattacharyya and N. Xie, Lower bounds for testing triangle-freeness in Boolean functions. SODA 2010, pp.87-98.

[37] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *JCSS*, 47:549–595, 1993.

[38] A. Bogdanov, Kenji Obata, and L. Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science*, pages 93–102, 2002.

[39] B. Bollobás, P. Erdős, M. Simonovits and E. Szemerédi, Extremal graphs without large forbidden subgraphs, Annals of Discrete Mathematics 3 (1978), 2941.

[40] W. G. Brown, P. Erdős and V.T. Sós, On the existence of triangulated spheres in 3-graphs and related problems, Periodica Mathematica Hungaria, 3 (1973), 221-228.

[41] E. Ben Sasson, M. Sudan, S. Vadhan, A. Wigderson, "Randomness-efficient Low degree tests and short PCP's via Epsilon-biased sets", in *Proceedings of the 35th STOC*, pp. 612–621, 2003.

[42] B. Chazelle, D. Liu, A. Magen, "Sublinear Geometric Algorithms", *SIAM J. Comput.* 35(3): 627-646 (2005)

[43] B. Chazelle and R. Rubinfeld and L. Trevisan, Approximating the minimum spanning tree weight in sublinear time, SIAM J. Comput. 34(6):1370-1379(2005).

[44] D. Coppersmith. Manuscript, 1989.

[45] A. Czumaj, F. Ergun, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler, Sublinear-time approximation of euclidean minimum spanning tree, SIAM Journal on Computing, 35(1): 91 - 109, 2005.

[46] A. Czumaj and C. Sohler, Testing expansion in bounded-degree graphs, Proc. of FOCS 2007.

[47] A. Czumaj and C. Sohler. Estimating the Weight of Metric Minimum Spanning Trees in Sublinear Time. SIAM Journal on Computing, 39(3): 904 - 922, August 2009.

[48] A. Czumaj, A. Shapira and C. Sohler, Testing hereditary properties of non-expanding bounded-degree graphs, SIAM J. on Computing, to appear.

[49] I. Dinur. The PCP theorem by gap amplification. J. ACM 54(3), pp. 12, 2007.

[50] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron and A. Samorodnitsky, Improved Testing Algorithms for Monotonocity, Proceedings of RANDOM 1999, volume 1671 of *Lecture Notes in Computer Science*, pp. 97–108. Springer, 1999.

[51] G. Elek, Parameter testing in bounded degree graphs of subexponential growth. To appear in Random Structures and Algorithms.

[52] P. Erdős, On circuits and subgraphs of chromatic graphs, Mathematika, 9 (1962), 170-175.

[53] F. Ergün, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *JCSS*, 60(3):717–751, 2000.

[54] F. Ergun, R. Kumar, and R. Rubinfeld. Checking approximate computations of polynomials and functional equations. *SIAM J. Comput*, 31(2):550–576, 2001.

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

[56] E. Fischer. On the strength of comparisons in property testing. *Electronic Colloqium on Computational Complexity*, 8(20), 2001.

[57] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky, Monotonicity Testing Over General Poset Domains, STOC 2002, pages 474-483.

[58] G. Frahling, P. Indyk, and C. Sohler. Sampling in dynamic data streams and applications. In *Symposium on Computational Geometry*, pp. 142-149, 2005.

[59] A. Frieze and R. Kannan. Quick Approximation to Matrices and Applications. Combinatorica, Vol. 19 No. 2, pp. 175-220.

[60] E. Fischer and A. Matsliah. Testing graph isomorphism. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.

[61] E. Fischer, A. Matsliah and A. Shapira. Approximate hypergrph partitioning and applications. Proc. of FOCS 2007.

[62] E. Fischer and I. Newman, Testing versus estimation of graph properties, SIAM J. Comput, 37 (2007) 482-501.

[63] M. Garey and D. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, 1979.

[64] O. Goldreich. Combinatorial property testing - a survey. In *Randomization Methods in Algorithm Design*, pages 45–60, 1998.

[65] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron and A. Samordinsky, Testing Monotonicity, *Combinatorica*, 20(3), pp.301–337, 2000.

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

[67] O. Goldreich, M. Krivelevich, I. Newman and E. Rozenberg, Hierarchy theorems for property testing, Proc. of RANDOM 2009.

[68] O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. *Electronic Colloqium on Computational Complexity*, 7(20), 2000.

[69] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32:302–343, 2002.

[70] O. Goldreich and D. Ron, A sublinear bipartiteness tester for bounded degree graphs, Combinatorica 19 (1999), 335-373.

[71] O. Goldreich and M. Sudan Locally testable codes and PCPs of almost-linear length. J. ACM 53(4). pp. 558-655 (2006).

[72] O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. In *Proc. 42nd IEEE Conference on Foundations of Computer Science*, pages 460–469, 2001.

[73] T. Gowers, Hypergraph regularity and the multidimensional Szemerédi theorem, Ann. of Math. Volume 166, Number 3 (2007), 897-946.

[74] B. Green, A Szemerédi-type regularity lemma in abelian groups, GAFA 15 (2005), 340-376.

[75] S. Halevy and E. Kushilevitz. Testing monotonicity over graph products. In proceedings of *ICALP*, vol. 3142 of LNCS pp. 721-732, Springer, 2004.

[76] A. Hassidim, J. A. Kelner, H. N. Nguyen and K. Onak, Local graph partitions for approximation and testing, FOCS 2009.

[77] J. Hastad, A. Wigderson, "Simple Analysis of Graph Tests for Linearity and PCP", *Random Structures and Algorithms*, 22(2): 139–160, 2003.

[78] P. Indyk, Sublinear Time Algorithms for Metric Space Problems, STOC, 1999.

[79] C.S. Jutla, A.C. Patthak, A. Rudra, and D. Zuckerman. Testing low-degree polynomials over prime fields. In *Proceedings of the Forty-Fifth Annual Symposium on Foundations of Computer Science*, pages 423–432, 2004.

[80] S. Kale and C. Seshadhri Testing expansion in bounded-degree graphs Proc. of ICALP 2008.

[81] T. Kaufman and S. Litsyn Almost Orthogonal Linear Codes are Locally Testable. *FOCS 2005* pp. 317-326.

[82] T. Kaufman, S. Litsyn, N. Xie. Breaking the ϵ-soundness bound of the linearity test over GF(2). SIAM Journal on Computing, 39(5), pp. 1988-2003, 2010.

[83] T. Kaufman and D. Ron. Testing polynomials over general fields. In *Proceedings of the Forty-Fifth Annual Symposium on Foundations of Computer Science*, pages 413–422, 2004.

[84] T. Kaufman and M. Sudan. Sparse Random Linear Codes are Locally Decodable and Testable. FOCS 2007. pp. 590-600.

[85] T. Kaufman and M. Sudan. Algebraic property testing: the role of invariance. STOC 2008. pp. 403-412.

[86] M. Kiwi, F. Magniez, and M. Santha. Approximate testing with error relative to input size. *JCSS*, 66(2):371–392, 2003.

[87] Y. Kohayakawa, B. Nagle, V. Rödl, M. Schacht and J. Skokan, The hypergraph regularity method and its applications, Proceedings of the National Academy of Sciences USA, 102(23): 8109-8113.

[88] L. Komlós, Covering odd cycles, Combinatorica 17 (1997), 393-400.

[89] S. Kopparty and S. Saraf. Tolerant Linearity Testing and Locally Testable Codes. APPROX-RANDOM 2009. pp. 601-614.

[90] S. Kopparty and S. Saraf. Local List-Decoding and Testing of Sparse Random Linear Codes from High-Error. STOC 2010.

[91] T. Kaufman, M. Krivelevich, D. Ron. Tight Bounds for Testing Bipartiteness in General Graphs. *SIAM J. Comput.* 33(6), pp. 1441-1483 (2004).

[92] F. Kuhn, T. Moscibroda, R. Wattenhofer, The price of being near-sighted, In proceedings of SODA 2006, pp. 980-909.

[93] D. Král', O. Serra and L. Vena, A combinatorial proof of the removal lemma for groups, arXiv:0804.4847v1.

[94] D. Král', O. Serra and L. Vena, A removal lemma for linear systems over finite fields, Jornadas de Matematica Discreta y algortimica 2008.

[95] R. J. Lipton and R. E. Tarjan, A separator theorem for planar graphs, SIAM J. Appl. Math., 36 (1979), 177-189.

[96] L. Lovász and B. Szegedy, Graph limits and testing hereditary graph properties, manuscript, 2006.

[97] M. Luby, A Simple Parallel Algorithm for the Maximal Independent Set Problem. SIAM J. Comput. 15(4): 1036-1053 (1986).

[98] S. Marko, D. Ron, Approximating the distance to properties in bounded-degree and general sparse graphs. ACM Transactions on Algorithms 5(2), 2009.

[99] O. Meir, Combinatorial Construction of Locally Testable Codes. *SIAM J. Comput.* 39(2), pp. 491-544, 2009.

[100] N. Mishra and D. Oblinger and L. Pitt, Sublinear Time Approximate Clustering, soda 2001, pp.439–447.

[101] A. Nachmias and A. Shapira, Testing the expansion of a graph, Information and Computation, to appear.

[102] H. N. Nguyen and K. Onak, Constant-Time Approximation Algorithms via Local Improvements. *Proceedings of FOCS 2008*, pp. 327-336.

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

[104] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3): 183-196 (2007)

[105] M. Parnas, D. Ron and R. Rubinfeld. Tolerant property testing and distance approximation. J. Comput. Syst. Sci. 72(6). pp. 1012-1042 (2006).

[106] A. Polischuk and D. Spielman. Nearly linear-size holographic proofs. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 194–203, 1994.

[107] S. Raskhodnikova, D. Ron, R. Rubinfeld, and A. Smith. Sublinear algorithms for string compressibility and the distribution support size. Proceedings of APPROX-RANDOM 2007, pp. 609-623.

[108] S. Raskhodnikova, D. Ron, A. Shpilka, and A. Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. In *Proc. 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 559–569, 2007.

[109] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 475–484, 1997.

[110] V. Rödl and R. Duke, On graphs with small subgraphs of large chromatic number, Graphs and Combinatorics 1 (1985), 9196.

[111] V. Rödl and M. Schacht, Property testing in hypergraphs and the removal lemma, Proc. of STOC 2007.

[112] D. Ron. Property testing. In *Handbook on Randomization, Volume II*, pages 597–649, 2001.

[113] D. Ron. Property Testing: A Learning Theory Perspective. *Foundations and Trends in Machine Learning* 1(3). pp. 307-402, 2008.

[114] K.F. Roth, On certain sets of integers, J. London Math. Soc. 28 (1953), 104-109.

[115] R. Rubinfeld. On the robustness of functional equations. *SIAM J. Comput*, 28(6):1972–1997, 1999.

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

[117] I. Ruzsa and E. Szemerédi, Triple systems with no six points carrying three triangles, in Combinatorics (Keszthely, 1976), Coll. Math. Soc. J. Bolyai 18, Volume II, 939-945.

[118] A. Samorodnitsky, L. Trevisan, "A PCP characterization of NP with optimal amortized query complexity", in *Proceedings of 32nd STOC*, pp. 191–199, 2000.

[119] A. Shapira, Green's conjecture and testing linear-invariant properties. STOC 2009. pp. 159-166.

[120] M. Sudan, Invariance in property testing, ECCC report TR10-051, 2010.

[121] E. Szemerédi, Regular partitions of graphs, In: *Proc. Colloque Inter. CNRS* (J. C. Bermond, J. C. Fournier, M. Las Vergnas and D. Sotteau, eds.), 1978, 399–401.

[122] A. Shpilka, A. Wigderson, “Derandomizing Homomorphism Testing in General Groups”, in *Proceedings of 36th STOC*, pp. 427–435, 2004.

[123] T. Tao, A variant of the hypergraph removal lemma, J. Combin. Theory, Ser. A 113 (2006), 1257-1280.

[124] Y. Yoshida, M. Yamamoto, H. Ito, An improved constant-time approximation algorithm for maximum matchings, *In proceedings of STOC 2009*, pp. 225-234.