Sublinear time Algorithms

Ronitt Rubinfeld[*]

June 5, 2006

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 sorts of answers that one might be able to achieve in this new setting.

1 Introduction

The goal of algorithmic research is to design efficient algorithms, where efficiency is typically measured as a function of the length of the input. For instance, the elementary school algorithm for multiplying two n digit integers takes roughly n^2 steps, while more sophisticated algorithms have been devised which run in less than $n \log^2 n$ steps. It is still not known whether a linear time algorithm is achievable for integer multiplication. Obviously any algorithm for this task, as for any other nontrivial task, would need to take at least linear time in n, since this is what it would take to read the entire input and write the output. Thus, showing the existence of a linear time algorithm for a problem was traditionally 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? Analogous to the reasoning that we used for multiplication, 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

[*]ronitt@theory.csail.mit.edu Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139. Supported by NSF grant CCF0514771.

estimate of the average value of a list of numbers of bounded magnitude. But what about more interesting algorithmic questions? For example, given access to all transcripts of trades in the stock exchange, can we determine whether there is a trend change? This is easily detectable after a careful scan of the entire transcript, but by the time the scan is performed, it might be too late to make use of the information. However, it might be feasible to contruct much faster algorithms based on random sampling. The recently emerging theory of sublinear time algorithms addresses questions of precisely this nature for problems in various domains.

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 begin by giving a motivating example of a sublinear time algorithm in Section 2. In Section 3 we formalize the definitions of approximations that sublinear time algorithms are able to achieve. We then describe various examples of sublinear time algorithms.

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.

In order to construct an algorithm that runs in sublinear time for determining whether $\vec{x}$ is monotone, we first need to make our model of computation precise. For example, if our algorithm must scan $x_1, \ldots, x_{i-1}$ in order to reach x_i, then there is no hope for the existence of a sublinear time algorithm. However, it is often natural to assume that our algorithms have *query* (also called *oracle*) access to the input. That is, they can access x_i in one step for any $1 \leq i \leq n$.

Even with this model of computation, 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? On first thought, one might try picking random indices i, j and performing tests of the form "is $x_i \leq x_j$?" or "is $x_i \leq x_{i+1}$?". However, these tests do not work very well. It is easy to construct examples showing that there are lists that are not even $(1 - \frac{1}{n^{1/2}})$-close to monotone, yet pass such tests with probablity at least $1 - \frac{1}{n^{1/4}}$. This means that at least $n^{1/4}$ such tests must be performed if one is to find a reason to output "fail". Though this does not rule out the possibility of a a sublinear time property tester, we will see that one can do much better. In the following, we describe an $O(\log n)$ time algorithm from the work of Ergün et. al. [17] which tests if $\vec{x}$ has a long monotone increasing subsequence. Note that the problem is known to require $\Omega(\log n)$ queries [17, 20].

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.

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 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. 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 $\vec{x}$ that sees no pair of out-of-order elements along the search path. 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 $x_i < x_k$ and $x_k < x_j$, which implies $x_i < x_j$. 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 2.1 ([17]) *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).*

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. When f is the characteristic function of a property, i.e., $f(x) = 1$ if x has the property and $f(x) = 0$ otherwise, the following notion of approximation has emerged: 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. We first formalize 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 $x(i)$ returns the i^{th} element of the list.

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 $2/3$.

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. 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. 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 first defined by Rubinfeld and Sudan [37] in the context of program testing. Goldreich, Goldwasser, Ron [23] refined and generalized the definition. Various more general definitions are given in several works, including [17, 25, 31], which mostly differ in terms of the generality of the distance function and natural generalizations 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 homorphism. We then show that the ideas used to construct property testers for homomorphisms extend to other properties with similar underlying structure.

4.1 Homomorphism testing

We begin with an example that was originally motivated by applications in program testing [14] and was later used in the construction of Probabilistically Checkable Proof systems [6]. 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 [16]. However, here the situation is much different: 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 $x \in \mathcal{D}$ [13].

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 [37] (see also [36]), 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. We are interested in the relationship between ϵ and δ, as well as the range of δ for which the property is (ϵ, δ)-robust, since the value of δ directly influences the running time of the property tester.

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

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

Define ϵ such that 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 [14], considered this question and showed that over cyclic groups, there is a constant δ_0, such that if $\delta \leq \delta_0$, then the 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 [14] contains an improved argument due to Coppersmith [16], 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 [16]. In [13], 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 [14, 16]. Though $\delta_0 < 2/9$ is optimal over general Abelian groups, using Fourier techniques, Bellare et. al. [12] have shown that for groups of the form $(\mathbf{Z}/2)^n$, $\delta_0 \leq 45/128$ suffices.

A proof of a homomorphism test We describe the following proof, essentially due to Coppersmith [14, 16] 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 4.1 *Let G be a finite Abelian group and $f : G \to G$. 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 $\phi(x) = \mathrm{maj}_{y \in G}(f(x + y) - f(y))$, 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 4.2 $|\{y|f(y) = \phi(y)\}| \geq (1 - 2\delta)|G|$.

Proof:[of Claim 4.2] 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 4.3 *If $\delta < 2/9$, then $\forall x, z$, $\phi(x) + \phi(z) = \phi(x + z)$.*

Proof:[of Claim 4.3] 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:

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

$x+y_1$ and y_2 are both 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

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

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 4.3.

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 $c \log 1/\beta$ 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, $\phi(x+y) = f(x+y)$ *and* $\phi(y) = f(y)$, in which case $f(x+y) - f(y) = \phi(x)$. Thus it is easy to show that there is a constant c such 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

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

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 [36, 37]. A long series of works have given increasingly robust characterizations of functions that are low total degree polynomials (cf. [6, 32, 7, 34, 3, 29, 27]).

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 [18, 30].

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 [13] show that A must be close to the uniform distribution over a subgroup of G. More precisely, in [13] 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 [13]. On the other hand, in [13] 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 [14]. 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 [13] it is shown that this question on distributions is "equivalent" in difficulty to homomorphism testing:

Theorem 4.4 *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.

5 Combinatorial objects

In 1996, Goldreich, Goldwasser and Ron [23] focused attention on the problem of testing various properties of graphs and other combinatorial objects. Their work introduced what is now referred to as the *dense graph model* of property testing. In this model, a graph on n nodes is represented via an $n \times n$ adjacency matrix, where the (i, j)th entry of the matrix contains a 1 if the edge (i, j) is present in the graph and a 0 otherwise. Two graphs G and H are ϵ-close if at most ϵn^2 edges need to be modified (inserted or deleted) to turn G into H. In [23], several graph properties were shown to be easily testable. In fact, as we shall soon see, the question of which graph properties are easily testable has led to a series of intriguing results.

One property that [23] consider is that of k-colorability – is it possible to assign one of k colors to each of the nodes so that no pair of nodes that have an edge between them are assigned the same color? 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. Somewhat surprisingly, k-colorability is easily testable, so we can distinguish k-colorable graphs from those that are ϵ-far from k-colorable in constant time. Thus we see that the efficiency of a property tester is not directly related to the complexity of deciding the property exactly.

Though the proof of correctness of the property tester for k-colorability is involved, the algorithm used to conduct the property test is easy to describe: It simply picks a constant sized random sample of the vertices, queries all the edges among this random sample and then outputs "pass" or "fail", according to whether the sample is k-colorable. Since the sample is of constant size, the determination of whether the sample is k-colorable can be made in constant time.

Such algorithms that (1) pick a constant sized random sample of the vertices, (2) query all the edges among this random sample, and then (3) output "pass" or "fail" based on whether the subgraph has the property or not, are referred to as "natural algorithms". Modulo a technicality about how the final output decision is made, Goldreich and Trevisan [26] 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 [23] sparked a flurry of results in the dense graph model. A very interesting line of work was initiated in the work of Alon, Fischer, Krivelevich and Szegedy [2], in which they use the Szemerédi Regularity Lemma to show that the property of a graph being H-free (that is, the graph does not contain any copy of H as a subgraph) is easily testable for any constant sized graph H.

Very recently, the above line of work culminated in the following amazing result: Alon and Shapira [5] have shown that one can *completely* characterize the classes of graph properties that are easily testable with 1-sided error in the dense graph model. 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 Alon and Shapira is then the following:

Theorem 5.1 *A graph property P is easily testable with one-sided error if and only if P is semi-hereditary.*

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 by Alon and Shapira are quite involved, and are based on developing new variants of the Szemerédi Regularity Lemma. 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.

Several interesting questions regarding easily testable graph properties remain. For example, because of the use of the Szemerédi Regularity Lemma, the upper bounds given by the previously mentioned results have a dependence on $1/\epsilon$ that is enormous. It would be interesting to characterize which problems have property testers whose dependence on $1/\epsilon$ is polynomial (cf. [1]).

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 [21], it is shown that the property testing problem requires $\Omega(n)$ queries and that there is a property tester for this problem which uses $O(n^{5/4} \operatorname{polylog} n)$ queries, which is sublinear in the input size n^2.

The area of property testing has been very active, with a number of property testers devised for other models of graphs as well as other combinatorial objects. The testability of a problem is very sensitive to the model in which it is being tested. In contrast to the dense graph model, where k-colorability is easily testable, it is known that there are no sublinear time property testers for the k-colorability problem in models suitable for testing sparse graphs [15]. Property testers have also been studied in models of general graphs, and threshold-like behaviors have been found for the complexity of the testing problems in terms of the average degree of the graph [28, 4].

Property testers for combinatorial properties of matrices, strings, metric spaces and geometric objects have been devised. We refer the reader to the excellent surveys of Goldreich [22], Ron [35] and Fischer [19].

6 Testing properties of distributions

In a wide variety of settings, data is most naturally viewed as coming from a probability distribution. In order to effectively make use of such data, one must understand various properties of the underlying probability distribution. Some of these properties are “local” in nature, for example focusing on whether or not a specific domain element appears with large probability. Other properties have a rather “global” feel in the sense that they are a property of the distribution as a whole and not of a small subset of the domain elements. Unlike the case for local properties, it makes sense to characterize a distribution in terms of some meaningful distance measure to the closest distribution that has the global property. This yields a somewhat different model than the property testing model in terms of the assumption on how the data is presented for here we do not assume that an explicit description of the distribution is given.

In the following, we assume that there is an underlying distribution from which the testing algorithm receives independent identically distributed (iid) samples. The complexity of the algorithm is measured in terms of the number of samples required in order to produce a meaningful answer (the sample complexity). As mentioned in the introduction, it is a 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. However, more recently, properties such as closeness between two distributions, closeness to an explicitly given distribution, independence, and high entropy, have been studied in this model [24, 10, 9]. For many properties, well-known statistical techniques, such as the χ^2-test or the straightforward use of Chernoff bounds, have sample complexities that are at least linear in the size of the support of the underlying probability distribution. In contrast, there are algorithms whose sample complexity is sublinear in the size of the support for various properties of distributions.

We mention one illustrative example: Given samples of a distribution X on $[n]$, for example all the previous winners of the lottery, how can one tell whether X is close to uniform? We will measure closeness in terms of the L_2 norm, i.e., letting $X(i)$ denote the probability that X assigns to i,

$$\|X\|_2 = \sum_{i \in [n]} (X(i) - 1/n)^2.$$

Goldreich and Ron [24] note that since

$$\sum_{i \in [n]} (X(i) - 1/n)^2 = \sum_{i \in [n]} X(i)^2 - 1/n,$$

it is enough to estimate the collision probability. They then show that this can be done by considering only $O(\sqrt{n})$ samples and counting the number of pairs that are the same. By bounding the variance of their estimator, they obtain the following:

Theorem 6.1 ([24]) *There is an algorithm that, given a distribution X on $[n]$ via a generation oracle, approximates $\|X\|_2$ to within a multiplicative factor of $(1 \pm \epsilon)$ using $O(\sqrt{n}/\epsilon^2)$ samples, with constant probability.*

Such techniques are very useful for achieving sublinear time algorithms for testing whether distributions satisfy several other global properties. For example, for the properties of closeness of two arbitrary distributions [10], independence of a joint distribution [9], high entropy [8], and monotonicity of the probability density function (when the distribution is over a totally ordered domain) [11], the testing problem can be reduced to the problem of testing the near-uniformity of the distribution on various subdomains.

7 Some final comments

We have seen several contexts in which one can test properties in sublinear time. The study of sublinear time algorithms has led to a new understanding of many problems that had already been well-studied. Though we have mentioned only property testing problems in this survey, other, more traditional, types of approximations are achievable in sublinear time. Such algorithms have been used to design very fast approximation algorithms for graph problems and for string compressibility problems (cf. [23, 10, 33]). Some of these algorithms have even resulted in better linear time approximation algorithms than what was previously known.

Probabilistically Checkable Proof Systems (PCPs) can be thought of as a way to write down a proof so that another person can verify it by viewing only a constant number of locations (cf. [6]). PCPs can thus be viewed as a type of robust characterization and their verification is a sublinear algorithm. More interestingly, property testers for homomorphisms and low degree polynomials are used as key ingredients in the construction of Probabilistically Checkable Proof Systems.

As we have seen, the study of sublinear algorithms gives a new perspective that has yielded insights to other areas of theoretical computer science. 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, Tali Kaufman and Madhu Sudan for their extensive and helpful comments on this manuscript.

References

[1] N. Alon. Testing subgraphs in large graphs In *Proceedings of the Forty-Second Annual Symposium on Foundations of Computer Science* , pages 434–441, 2001.

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

[3] 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.

[4] N. Alon, T. Kaufman, M. Krivilevich, and D. Ron. Testing triangle-freeness in general graphs. In *Proceedings of the* 17^{th} *Symposium on Discrete Algorithms*, pages 279–288, 2006.

[5] N. Alon and A. Shapira. A characterization of the (natural) graph properties testable with one-sided error. In *Proceedings of the Forty-Sixth Annual Symposium on Foundations of Computer Science* , pages 429–438, 2005.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] M. Ben-Or, D. Coppersmith, M. Luby, and R. Rubinfeld. Non-abelian homomorphism testing, and distributions close to their self-convolutions. In *Proceedings of APPROX-RANDOM*, pages 273–285, 2004.

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

[15] A. Bogdanov, K. 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.

[16] D. Coppersmith. Manuscript, 1989.

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

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

[19] 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.

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

[21] E. Fischer and A. Matsliah. Testing graph isomorphism. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 299-308, 2006.

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

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

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

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

[26] 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.

[27] 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.

[28] T. Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. In *Proceedings of RANDOM-APPROX*, pages 341–353, 2003.

[29] 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.

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

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

[32] 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.

[33] S. Raskhodnikova, D. Ron, R. Rubinfeld, A. Shpilka, and A. Smith. Sublinear algorithms for string compressibility and the distribution support size. *Electronic Colloqium on Computational Complexity*, 5(125), 2005.

[34] 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.

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

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

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