Testing Juntas: A Brief Survey

Eric Blais

School of Computer Science, Carnegie Mellon University,
Pittsburgh PA 15213, USA
eblais@cs.cmu.edu

Abstract. A function on n variables is called a *k-junta* if it depends on at most k of its variables. In this survey, we review three recent algorithms for testing k-juntas with few queries.

1 Introduction

A function $f : \{0,1\}^n \to \{0,1\}$ is said to be a k-junta if it depends on at most k variables. Juntas provide a clean model for studying learning in the presence of many irrelevant features [7,10] and have consequently been of particular interest to the computational learning theory community [7,8,9,10,22,23,25].

As is typical in the machine learning setting, all learning results on k-juntas assume that the unknown function is a k-junta. In practice, however, it is often not known *a priori* whether a function being learned is a k-junta or not. It is therefore desirable to have an efficient algorithm for *testing* whether a function is a k-junta or "far" from being a k-junta before attempting to run any k-junta learning algorithm.

We consider the problem of testing k-juntas in the standard property testing framework originally defined by Rubinfeld and Sudan [27]. In this framework, we say that a function f is *ϵ-far* from being a k-junta if for every k-junta g, the functions f and g disagree on at least an ϵ fraction of all inputs.

An *ϵ-tester for k-juntas* is an algorithm $\mathcal{A}$ that queries an unknown function f on q inputs of its choosing, and then (1) accepts f with probability at least $2/3$ when f is a k-junta, and (2) rejects f with probability at least $2/3$ when f is ϵ-far from being a k-junta. When the algorithm $\mathcal{A}$ chooses all its queries in advance (i.e., before observing the values of the function on any of its previous queries), it is *non-adaptive*; otherwise it is *adaptive*. The main parameter of interest for our purposes is the number q of queries required by testers for k-juntas. In particular, the question we study is the following:

What is the minimum number of queries required to ϵ-test k-juntas?

A simple way to test k-juntas is to *learn* a target hypothesis k-junta using membership queries, and to then use a separate set of randomly-chosen queries to test this hypothesis [18,22]. Such an approach yields a valid tester but requires $O(k \log n/\epsilon)$ queries. In the rest of this survey, we will examine three algorithms that improve dramatically on this bound by requiring a number of queries that is *independent* of n.

O. Goldreich (Ed.): Property Testing, LNCS 6390, pp. 32–40, 2010.

2 Boolean Functions: Preliminaries

2.1 Basic Definitions

Throughout this survey, $f : \{0,1\}^n \to \{0,1\}$ represents a (generic) boolean function. The *complement* of f is the function $\bar{f} : \{0,1\}^n \to \{0,1\}$ defined by $\bar{f}(x) = 1 - f(x)$.

Given $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ from $\{0,1\}^n$, addition and multiplication are defined componentwise: $x + y = (x_1 + y_1, \ldots, x_n + y_n)$ and $x \cdot y = (x_1 y_1, \ldots, x_n y_n)$. We also define a *hybridization* operation: for a set $S \subseteq [n]$, the element $z = x_{\bar{S}} y_S \in \{0,1\}^n$ is formed by setting $z_i = x_i$ for every $i \in \bar{S} = [n] \setminus S$ and setting $z_i = y_i$ for every $i \in S$.

2.2 Notable Boolean Functions

The function that maps all inputs to 0 is the *constant zero* (or just *zero*) function; its complement is the *constant one* function that maps all inputs to 1. When there is an index $i \in [n]$ such that f is defined by $f(x) = x_i$, then we say that f is a *dictator* function. A function is an *anti-dictator* function if its complement is a dictator function.

For a set $S = \{i_1, i_2, \ldots, i_k\} \subseteq [n]$, the *linear* function corresponding to S is the function χ_S defined by $\chi_S(x) = x_{i_1} + x_{i_2} + \cdots + x_{i_k}$. By convention, we define $\chi_\emptyset$ to be the constant zero function. An alternative characterization of linear functions is provided by the following proposition.

Proposition 1. *A function $f : \{0,1\}^n \to \{0,1\}$ is linear if and only if for every $x, y \in \{0,1\}^n$, $f(x) + f(y) = f(x + y)$.*

For a set $S = \{i_1, i_2, \ldots, i_k\} \subseteq [n]$, the (monotone) *monomial* function corresponding to S is the function ξ_S defined by $\xi_S(x) = x_{i_1} x_{i_2} \cdots x_{i_k}$. (I.e., $\xi_S(x) = 1$ iff $x_{i_1} = \cdots = x_{i_k} = 1$.) As with linear functions, monomials have a useful alternative characterization.

Proposition 2. *A non-constant function $f : \{0,1\}^n \to \{0,1\}$ is a monomial if and only if for every $x, y \in \{0,1\}^n$, $f(x) \cdot f(y) = f(x \cdot y)$.*

2.3 Influence

For an index $i \in [n]$, the *influence* of the ith variable in the function f is

$$\mathrm{Inf}_f(i) = \Pr_x[f(x) \neq f(x^{(i)})],$$

where the probability is over the uniform distribution on $\{0,1\}^n$ and the input $x^{(i)} \in \{0,1\}^n$ is obtained by flipping the value of the ith variable of x.

The *influence* of the set $S \subseteq [n]$ in the function f is

$$\mathrm{Inf}_f(S) = 2 \Pr_{x,y}[f(x) \neq f(x_{\bar{S}} y_S)].$$

A *k-junta* is a function f for which there are at most k indices $i \in [n]$ such that $\mathrm{Inf}_f(i) > 0$. Alternatively, f is a k-junta if there exists a set $S \subseteq [n]$ of size $|S| \leq k$ such that $\mathrm{Inf}_f(\bar{S}) = 0$.

3 Testing 1-Juntas

We begin with the simplest case: testing 1-juntas. The family of 1-junta functions is small. It contains only the constant functions, the dictator functions, and the anti-dictator functions. Furthermore, dictator functions have the useful distinction of being the only non-constant functions that are both linear functions and monomials. This distinction lies at the heart of the 1-junta tester that we will examine in this section.

3.1 The Algorithm

As suggested above, our main building block for testing 1-juntas is an algorithm that accepts functions that are both linear and monomials. The characterizations of linear functions and of monomials from Propositions 1 and 2 suggest the following simple algorithm for this task:

> LINEAR MONOMIAL TEST(f, ϵ)
>
> 1. For $O(1/\epsilon)$ randomly selected pairs $x, y \in \{0,1\}^n$,
>   1.1. Verify that $f(x) + f(y) = f(x+y)$.
>   1.2. Verify that $f(x) \cdot f(y) = f(x \cdot y)$.
> 2. **Accept** iff all verifications pass.

Clearly, the LINEAR MONOMIAL TEST always accepts the zero function and dictator functions. To accept all 1-juntas, it suffices to test f and its complement $\bar{f}$ for the property of being a linear monomial:

> 1-JUNTA TEST(f, ϵ)
>
> 1. Call LINEAR MONOMIAL TEST(f, ϵ).
> 2. Call LINEAR MONOMIAL TEST($\bar{f}$, ϵ).
> 3. **Accept** iff one of the above tests accepts.

The 1-JUNTA TEST algorithm always accepts 1-juntas. To establish that it is a valid tester for 1-juntas, we need to show that it rejects functions that are ϵ-far from 1-juntas with high probability. We do so in two steps.

First, we show that the 1-JUNTA TEST rejects functions that are far from linear with high probability. This statement follows from the *robustness* of the linearity characterization in Proposition 1: when a function f is ϵ-far from linear and x, y are generated uniformly at random, then $f(x) + f(y) \neq f(x+y)$ with probability at least ϵ [3,11].

Lemma 3 (Blum et al. [11], Bellare et al. [3]). *Let f be ϵ-far from linear. Then the* 1-JUNTA TEST *rejects with probability at least* $2/3$.

Second, we show that functions that are ϵ-close to a linear function χ_S for some set S of size $|S| \geq 2$ are rejected by the monomial test in Line 1.2 of the *Linear Monomial Test* with high probability. This is indeed the case, as an elementary counting argument shows.

Lemma 4 (Bellare et al. [4]). *Fix* $0 < \epsilon < \frac{1}{8}$ *and let* f *be* ϵ*-close to* χ_S *for some set* $S \subseteq [n]$ *of size* $|S| \geq 2$*. Then the* 1-Junta Test *rejects with probability at least* $2/3$*.*

Together, Lemmas 3 and 4 show that the 1-Junta Test rejects functions that are ϵ-far from 1-juntas with probability at least $2/3$. This completes the proof of correctness of the algorithm. We can also easily verify that the tester makes only $O(1/\epsilon)$ queries to the input function; this is optimal.

3.2 History

The problem of testing dictator functions was first studied by Bellare, Goldreich, and Sudan [4] in the context of testing the Long Code for constructing probabilistically-checkable proof (PCP) systems. As pointed out in [26], testing the Long Code is equivalent to testing dictator functions, and their test for dictator functions is roughly equivalent to the 1-Junta Test algorithm above.[1] The analysis of the dictator test was further generalized and extended by Parnas, Ron, and Samorodnitsky [26].

Due to the key role of dictator functions in PCP systems, many other variants of the dictatorship testing problem have been studied – see [13] in this volume and the references therein for more information on this topic.

4 Testing *k*-Juntas

We now turn our attention to the general problem of testing k-juntas for any value of $k \geq 1$. In contrast to the case of 1-juntas, when $k \geq 2$ the class of k-juntas does not have a simple characterization that directly suggests a testing algorithm. Nonetheless, as we will see in this section it is still possible to test k-juntas with a small number of queries.

4.1 The Algorithm

The algorithm for testing k-juntas relies on two basic components: the Independence Test, and the idea of *randomly partitioning* the coordinates.

The Independence Test is a simple algorithm for verifying whether a given function f is independent of a set $S \subseteq [n]$ of coordinates:

> Independence Test$(f,\ S)$
>
> 1. Generate $x, y \in \{0,1\}^n$ uniformly at random.
> 2. **Accept** iff $f(x) = f(x_{\bar{S}} y_S)$.

[1] There is one difference: when testing dictator functions, constant functions must be rejected. In our case we want to accept them; this simplifies the algorithm slightly.

By our definition of influence, the probability that the INDEPENDENCE TEST rejects is exactly $\frac{1}{2}\mathrm{Inf}_f(S)$. In particular, when f is independent of the variables in S, then $\mathrm{Inf}_f(S) = 0$ and the test always accepts.

A naïve way to use the INDEPENDENCE TEST for testing k-juntas is to run the test (sufficiently many times) on each singleton set $S = \{1\}, \{2\}, \ldots, \{n\}$ and to accept iff at most k of the sets are rejected. This proposed algorithm is indeed a valid tester for k-juntas, but it requires $\Omega(n)$ queries. A simple trick, however, can dramatically reduce the number of queries required: take a sufficiently fine partition of the coordinates $[n]$ and run the INDEPENDENCE TEST on each part.

> k-JUNTA TEST(f, ϵ)
>
> 1. Randomly partition the coordinates into $O(k^2)$ buckets.
> 2. Run INDEPENDENCE TEST $\tilde{O}(k^2/\epsilon)$ times.
> 3. **Accept** iff at most k buckets fail the independence test.

Clearly, the k-JUNTA TEST always accepts k-juntas: if there are only k indices $i \in [n]$ for which $\mathrm{Inf}_f(i) > 0$, then at most k parts in the random partition will have influence $\mathrm{Inf}_f(S) > 0$. Conversely, when f is ϵ-far from being a k-junta, Fischer et al. [17] showed that with high probability over the choice of the random partition, at least $k+1$ parts have large influence.

Lemma 5 (Fischer et al. [17]). *Let $f : \{0,1\}^n \to \{0,1\}$ be ϵ-far from being a k-junta and $s = \Theta(k^2)$. Then with high probability a random partition $S_1 \dot\cup S_2 \dot\cup \cdots \dot\cup S_s$ of $[n]$ will have at least $k+1$ parts with influence $\mathrm{Inf}_f(S_j) > \epsilon/k^2$.*

The proof of Lemma 5 uses Fourier analysis. The rest of the proof of correctness of the k-JUNTA TEST follows almost immediately. The k-JUNTA TEST uses $\tilde{O}(k^4/\epsilon)$ queries. This bound is significant in that it is independent of n; as we discuss below, however, variants on this algorithm can test k-juntas with fewer queries.

4.2 History

Fischer, Kindler, Ron, Safra, and Samorodnitsky [17] first studied the problem of testing juntas and introduced the algorithm presented in this section. They also designed multiple other testing algorithms that improve on the query complexity of the k-JUNTA TEST. In particular, by using the INDEPENDENCE TEST on carefully chosen *sets* of parts in a random partition, they showed that $\tilde{O}(k^2/\epsilon)$ queries are sufficient to test k-juntas.

Fischer et al. [17] also introduced the first non-trivial lower bound on the query complexity of junta testing problem: they showed that for $k = o(\sqrt{n})$, non-adaptive testing algorithms for testing k-juntas must make at least $\tilde{\Omega}(\sqrt{k})$ queries. This lower bound implies a lower bound of $\Omega(\log k)$ queries for all adaptive k-junta testers as well. The lower bound was improved shortly afterwards by Chockler and Gutfreund [14], who showed that $\Omega(k)$ queries are required to test k-juntas (adaptively or non-adaptively).

The gap between the $\Omega(k)$ and $\tilde{O}(k^2/\epsilon)$ bounds on the query complexity of the junta testing problem remained unchanged until recently, when a new algorithm was introduced to test k-juntas with $\tilde{O}(k^{1.5}/\epsilon)$ queries [5]. This was followed by the introduction of another algorithm for testing k-juntas with $O(k \log k + k/\epsilon)$ queries [6]; we examine this algorithm in the next section.

5 Testing k-Juntas Nearly Optimally

The algorithm we saw in the last section relied on the Independence Test. To improve the query complexity, the algorithm we present in this section relies on a slightly stronger building block.

5.1 The Algorithm

The starting point for the algorithm is an observation due to Blum, Hellerstein, and Littlestone [9]: if we have two inputs $x, y \in \{0,1\}^n$ such that $f(x) \neq f(y)$, then the set of coordinates $i \in [n]$ for which $x_i \neq y_i$ contains a coordinate that is relevant in f. Furthermore, by performing a binary search over the hybrid inputs formed from x and y, we can identify a relevant coordinate with $O(\log n)$ queries.

Even more interestingly, if we have a partition $\mathcal{I}$ of $[n]$ and we have a pair of inputs x, y such that $f(x) \neq f(y)$, we can use the same binary search idea to identify a part that contains a relevant coordinate with only $O(\log |\mathcal{I}|)$ queries. We use this idea to create an algorithm that attempts to find a part with a relevant coordinate as follows:

Find Relevant Part(f, $\mathcal{I}$, S)

1. Generate $x, y \in \{0,1\}^n$ uniformly at random.
2. If $f(x) \neq f(x_{\bar{S}} y_S)$ then
   2.1. Use a binary search to identify a part $I \in \mathcal{I}$ that contains a relevant variable;
   2.2. **Return** I.
3. Otherwise, **Return** $\emptyset$.

Note that by the test in Line 2, if the algorithm finds a part with a relevant variable, that relevant variable is guaranteed to be in S. Also, the probability that Find Relevant Part succeeds in identifying a relevant part is the probability that $f(x) \neq f(x_{\bar{S}} y_S)$, which as we have seen previously is exactly $\frac{1}{2}\mathrm{Inf}_f(S)$.

The algorithm we now consider for testing k-juntas uses the Find Relevant Part in the obvious way: after taking a random partition of the coordinates, the algorithm calls this routine a large number of times and rejects the input if it identifies $k+1$ distinct parts that contain relevant coordinates.

NEARLY OPTIMAL k-JUNTA TEST(f, ϵ)

1. Randomly partition $[n]$ into a partition $\mathcal{I}$ with poly(k/ϵ) parts and initialize $J \leftarrow \emptyset$.
2. For each of $O(k/\epsilon)$ rounds,
   2.1. $J \leftarrow J \cup$ FIND RELEVANT PART$(f, \mathcal{I}, \bar{J})$.
   2.2. If J contains $> k$ parts, quit and **Reject**.
3. **Accept**.

As with the algorithms in the previous sections, it is easy to check that this algorithm always accepts k-juntas. Once again, the non-trivial part of the proof of correctness involves showing that functions ϵ-far from k-juntas are rejected with high probability. The key to proving that statement is the following lemma:

Lemma 6 ([6]). *Let $f : \{0,1\}^n \to \{0,1\}$ be ϵ-far from being a k-junta, and let $\mathcal{I}$ be a sufficiently fine partition of $[n]$. Then with high probability every set J formed by taking the union of at most k parts of $\mathcal{I}$ satisfies $\mathrm{Inf}_f(\overline{J}) \geq \epsilon/2$.*

The proof of Lemma 6 can be completed with Fourier analysis. Alternatively, and more generally, it can also be completed using the *Efron-Stein* decomposition of functions [16]. This is the approach taken in [6], and it enables the analysis of the algorithm to hold even in the more general setting where the algorithm is testing functions with any finite product domain and any finite ranges for the property of being k-juntas.[2]

6 Open Problems and Future Directions

There are many possible directions for future research on testing k-juntas. We highlight three particularly intriguing open problems.

6.1 Classical vs. Quantum Property Testing

The field of property testing can be extended to allow the tester to use the quantum oracle model of Beals et al. [2]. The resulting model is called *quantum property testing* and was first studied by Buhrman, Fortnow, Newman, and Röhrig [12]. They showed that there are properties that can be tested with significantly fewer queries in the quantum model than in the classical model and that for some other properties, the extra power of the quantum oracle does not improve the query complexity of the associated testing problem.

The first open problem asks if quantum oracles help when testing juntas: *Is there a gap between the quantum and classical query complexities for testing k-juntas?*

Atıcı and Servedio [1] studied the problem of testing juntas in the quantum model. They showed that in this model, $O(k/\epsilon)$ queries are sufficient and $\Omega(\sqrt{k})$

[2] We note that the result in [6] was not the first one to generalize the analysis of a junta testing algorithm to non-boolean functions; Diakonikolas et al. [15] did so as well with a more technically intricate argument.

queries are necessary to ϵ-test k-juntas. At the time that this algorithm was introduced, it provided a quadratic improvement over the query complexity of the best classical k-junta tester. Of course, the algorithm presented in Section 5 reduces the gap to be only logarithmic in k, and in fact our strongest lower bounds in the classical model are not strong enough to guarantee the existence of a gap in the query complexities.

6.2 Adaptive vs. Non-Adaptive Testing

Gonen and Ron [21], and Goldreich and Ron [19] (see also [20] in this volume) recently began a systematic study of the benefits of adaptivity for testing properties in the dense-graph model. They showed that for some properties, there is a gap between the query complexity of the best adaptive and non-adaptive testing algorithms, while for other properties no such gap exists.

The current gap between query complexity of the best adaptive and non-adaptive algorithms for testing k-juntas — $O(k \log k + k/\epsilon)$ and $\tilde{O}(k^{3/2}/\epsilon)$, respectively — leaves the following basic problem open: *Does adaptivity help when testing k-juntas?*

6.3 Improved Testers for Other Properties

Following the work of Fischer et al. [17], junta testers have been used as a basic building block to design testers for many other properties of boolean functions, including function isomorphism [17], halfspaces [24], and many concise representation properties (e.g., being computable by a small decision tree or by a small circuit, having low Fourier degree) [15] (see also [28] in this volume).

All of the above testing algorithms use one of the k-junta testers presented in Section 4. The last open problem that we wish to mention is the following: *Can the* Nearly Optimal k-Junta Test *be used (or extended) to obtain improved testing algorithms for function isomorphism, halfspaces, or concise representation properties?*

References

1. Atıcı, A., Servedio, R.A.: Quantum algorithms for learning and testing juntas. Quantum Information Processing 6(5), 323–348 (2007)
2. Beals, R., Buhrman, H., Cleve, R., Mosca, M., de Wolf, R.: Quantum lower bounds by polynomials. J. of the ACM 48(4), 778–797 (2001)
3. Bellare, M., Coppersmith, D., Håstad, J., Kiwi, M., Sudan, M.: Linearity testing in characteristic two. IEEE Transactions on Information Theory 42(6), 1781–1795 (1996)
4. Bellare, M., Goldreich, O., Sudan, M.: Free bits, PCPs and non-approximability – towards tight results. SIAM J. Comput. 27(3), 804–915 (1998)
5. Blais, E.: Improved bounds for testing juntas. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) APPROX and RANDOM 2008. LNCS, vol. 5171, pp. 317–330. Springer, Heidelberg (2008)
6. Blais, E.: Testing juntas nearly optimally. In: Proc. 41st Symposium on Theory of Computing, pp. 151–158 (2009)

7. Blum, A.: Relevant examples and relevant features: thoughts from computational learning theory. In: AAAI Fall Symposium on 'Relevance' (1994)
8. Blum, A.: Learning a function of r relevant variables. In: Proc. 16th Conference on Computational Learning Theory, pp. 731–733 (2003)
9. Blum, A., Hellerstein, L., Littlestone, N.: Learning in the presence of finitely or infinitely many irrelevant attributes. J. Comp. Syst. Sci. 50(1), 32–40 (1995)
10. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence 97(2), 245–271 (1997)
11. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. J. Comput. Syst. Sci. 47(3), 549–595 (1993)
12. Buhrman, H., Fortnow, L., Newman, I., Röhrig, H.: Quantum property testing. In: Proc. 14th Symp. on Discrete Algorithms, pp. 480–488 (2003)
13. Chen, V.: Query-Efficient dictatorship testing with perfect completeness. In: Goldreich, O. (ed.) Property Testing. LNCS, vol. 6390, pp. 276–279. Springer, Heidelberg (2010)
14. Chockler, H., Gutfreund, D.: A lower bound for testing juntas. Information Processing Letters 90(6), 301–305 (2004)
15. Diakonikolas, I., Lee, H.K., Matulef, K., Onak, K., Rubinfeld, R., Servedio, R.A., Wan, A.: Testing for concise representations. In: Proc. 48th Symposium on Foundations of Computer Science, pp. 549–558 (2007)
16. Efron, B., Stein, C.: The jackknife estimate of variance. Ann. of Stat. 9(3), 586–596 (1981)
17. Fischer, E., Kindler, G., Ron, D., Safra, S., Samorodnitsky, A.: Testing juntas. J. Comput. Syst. Sci. 68(4), 753–787 (2004)
18. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. J. of the ACM 45(4), 653–750 (1998)
19. Goldreich, O., Ron, D.: Algorithmic aspects of property testing in the dense graphs model. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX–RANDOM 2009. LNCS, vol. 5687, pp. 520–533. Springer, Heidelberg (2009)
20. Goldreich, O., Ron, D.: Algorithmic aspects of property testing in the dense graphs model. In: Goldreich, O. (ed.) Property Testing. LNCS, vol. 6390, pp. 295–305. Springer, Heidelberg (2010)
21. Gonen, M., Ron, D.: On the benefits of adaptivity in property testing of dense graphs. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) RANDOM 2007 and APPROX 2007. LNCS, vol. 4627, pp. 525–539. Springer, Heidelberg (2007)
22. Guijarro, D., Tarui, J., Tsukiji, T.: Finding relevant variables in PAC model with membership queries. In: Watanabe, O., Yokomori, T. (eds.) ALT 1999. LNCS (LNAI), vol. 1720, pp. 313–322. Springer, Heidelberg (1999)
23. Lipton, R.J., Markakis, E., Mehta, A., Vishnoi, N.K.: On the Fourier spectrum of symmetric boolean functions with applications to learning symmetric juntas. In: Proc. 20th Conference on Computational Complexity, pp. 112–119 (2005)
24. Matulef, K., O'Donnell, R., Rubinfeld, R., Servedio, R.A.: Testing halfspaces. In: Proc. 19th Symp. on Discrete Algorithms, pp. 256–264 (2009)
25. Mossel, E., O'Donnell, R., Servedio, R.A.: Learning functions of k relevant variables. J. Comput. Syst. Sci. 69(3), 421–434 (2004)
26. Parnas, M., Ron, D., Samorodnitsky, A.: Testing basic boolean formulae. SIAM J. Discret. Math. 16(1), 20–46 (2003)
27. Rubinfeld, R., Sudan, M.: Self-testing polynomial functions efficiently and over rational domains. In: Proc. 3rd Symp. on Discrete Algorithms, pp. 23–32 (1992)
28. Servedio, R.: Testing by implicit learning: a brief survey. In: Goldreich, O. (ed.) Property Testing. LNCS, vol. 6390, pp. 197–210. Springer, Heidelberg (2010)

Sublinear-time Algorithms*

Artur Czumaj** and Christian Sohler***

[1] Department of Computer Science and Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick
A.Czumaj@warwick.ac.uk
[2] Department of Computer Science, TU Dortmund,
christian.sohler@tu-dortmund.de

Abstract. In this paper we survey recent advances in the area of sublinear-time algorithms.

Keywords: Sublinear time algorithms, sublinear approximation algorithms.

1 Introduction

The area of *sublinear-time algorithms* is a new rapidly emerging area of computer science. It has its roots in the study of massive data sets that occur more and more frequently in various applications. Financial transactions with billions of input data and Internet traffic analyses (Internet traffic logs, clickstreams, web data) are examples of modern data sets that show unprecedented scale. Managing and analyzing such data sets forces us to reconsider the traditional notions of efficient algorithms: processing such massive data sets in more than linear time is by far too expensive and often even linear time algorithms may be too slow. Hence, there is the desire to develop algorithms whose running times are not only polynomial, but in fact are *sublinear* in n.

Constructing a sublinear time algorithm may seem to be an impossible task since it allows one to read only a small fraction of the input. However, in recent years, we have seen development of sublinear time algorithms for optimization problems arising in such diverse areas as graph theory, geometry, algebraic computations, and computer graphics. Initially, the main research focus has been on designing efficient algorithms in the framework of *property testing* (for excellent surveys, see [28,32,33,43,53]), which is an alternative notion of approximation for decision problems. But more recently, we have seen some major progress in sublinear-time algorithms in the classical model of randomized and approximation algorithms. In this paper, we survey some of the recent advances in this area. Our main focus is on sublinear-time algorithms for combinatorial problems, especially for graph problems and optimization problems in metric spaces.

* This survey is a slightly updated version of a survey that appeared in *Bulletin of the EATCS*, 89: 23–47, June 2006.

** Research supported by EPSRC award EP/G064679/1 and by the Centre for Discrete Mathematics and its Applications (DIMAP), EPSRC award EP/D063191/1.

*** Supported by DFG grant SO 514/3-1.

O. Goldreich (Ed.): Property Testing, LNCS 6390, pp. 41–64, 2010.