Approximating the Influence of Monotone Boolean Functions in $O(\sqrt{n})$ Query Complexity*

Dana Ron† Ronitt Rubinfeld ‡ Muli Safra§ Alex Samorodnitsky¶ Omri Weinstein ‖

Abstract

The *Total Influence* (*Average Sensitivity*) of a discrete function is one of its fundamental measures. We study the problem of approximating the total influence of a monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$, which we denote by $I[f]$. We present a randomized algorithm that approximates the influence of such functions to within a multiplicative factor of $(1 \pm \epsilon)$ by performing $O\left(\frac{\sqrt{n}}{I[f]}\mathrm{poly}(1/\epsilon)\right)$ queries. We also prove a lower bound of $\Omega\left(\frac{\sqrt{n}}{\log n \cdot I[f]}\right)$ on the query complexity of any constant-factor approximation algorithm for this problem (which holds for $I[f] = \Omega(1)$), hence showing that our algorithm is almost optimal in terms of its dependence on n. For general functions we give a lower bound of $\Omega\left(\frac{n}{I[f]}\right)$, which matches the complexity of a simple sampling algorithm.

Keywords: Influence of a Boolean function, Sublinear query approximation algorithms, Symmetric Chains.

1 Introduction

The influence of a function, first introduced by Ben-Or and Linial [2] in the context of "collective coin-flipping", captures the notion of the sensitivity of a multivariate function. More precisely, for a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the *individual influence* of coordinate i on f is defined as $I_i[f] \stackrel{\text{def}}{=} \Pr_{x\in\{0,1\}^n}[f(x) \neq f(x^{(\oplus i)})]$, where x is selected uniformly[1] in $\{0,1\}^n$ and $x^{(\oplus i)}$ denotes x with the i^{th} bit flipped. The *total influence* of a Boolean function f (which we simply refer to as *the influence* of f) is $I[f] = \sum_i I_i[f]$.

The study of the influence of a function and its individual influences (distribution) has been the focus of many papers ([2, 27, 8, 20, 40, 9, 41, 18, 7, 19, 34, 14] to mention a few – for a survey see [21]).

*Part of the work described in this paper appeared in the proceedings of RANDOM 2011 (where we presented a different algorithm and analysis).

†School of Electrical Engineering at Tel Aviv University, danar@eng.tau.ac.il. This work was supported by the Israel Science Foundation (grant number 246/08).

‡CSAIL at MIT, and the Blavatnik School of Computer Science at Tel Aviv University, ronitt@csail.mit.edu. This work was 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.

§Blavatnik School of Computer Science at Tel Aviv University, safra@post.tau.ac.il.

¶The institute for Computer Science, Hebrew University, salex@cs.huji.ac.il. This work was supported by BSF grant number 2006377.

‖Computer Science Department, Princeton University, oweinste@cs.princeton.edu.

[1]The influence can be defined with respect to other probability spaces (as well as for non-Boolean functions), but we focus on the above definition.

The influence of functions has played a central role in several areas of computer science. In particular, this is true for distributed computing (e.g., [2, 27]), hardness of approximation (e.g., [15, 28]), learning theory (e.g., [24, 10, 35, 36, 12])[2] and property testing (e.g., [16, 4, 5, 32, 37]). The notion of influence also arises naturally in the context of probability theory (e.g., [38, 39, 3]), game theory (e.g., [30]), reliability theory (e.g., [29]), as well as theoretical economics and political science (e.g., [1, 25, 26]).

Given that the influence is such a basic measure of functions and it plays an important role in many areas, we believe it is of interest to study the algorithmic question of approximating the influence of a function as efficiently as possible, that is, by querying the function on as few inputs as possible. Specifically, the need for an efficient approximation for a function's influence might arise in the design of sublinear algorithms, and in particular property testing algorithms.

As we show, one cannot improve on a standard sampling argument for the problem of estimating the influence of a general Boolean function, which requires $\Omega(\frac{n}{I[f]})$ queries to the function, for any constant multiplicative estimation factor.[3] This fact justifies the study of subclasses of Boolean functions, among which the family of monotone functions is a very natural and central one. A function over the n-dimensional Boolean hypercube is *monotone* if $f(x) \leq f(y)$ for every two points x, y such that $x \prec y$, where '$\prec$' is the standard partial order over the Boolean hypercube. Namely, for $x = x_1, \ldots, x_n, y = y_1, \ldots, y_n \in \{0,1\}^n$, we use the notation $x \prec y$ to mean that $x_i \leq y_i$ for every $1 \leq i \leq n$, and $x_i < y_i$ for some $1 \leq i \leq n$. We show that the special structure of monotone functions implies a useful behavior of their influence, making the computational problem of approximating the influence of such functions significantly easier.

1.1 Our results and techniques

We present a randomized algorithm that approximates the influence of a monotone Boolean function to within any multiplicative factor using sublinear query complexity, so long as the influence is not too small. More precisely, we prove the following theorem:

Theorem 1.1 *For every $\delta, \epsilon > 0$ and for every monotone function $f : \{0,1\}^n \to \{0,1\}$ such that $I[f] \geq \exp(-c_1\epsilon^2 n + c_2 \log(n/\epsilon))$ for certain fixed constants c_1 and c_2, there is a randomized algorithm, whose output $\hat{I}$ satisfies*

$$(1-\epsilon) \cdot I[f] \leq \hat{I} \leq (1+\epsilon) \cdot I[f] \ .$$

with probability at least $1-\delta$. Furthermore, with probability at least $1-\delta$, the number of queries performed by the algorithm is $O\left(\frac{\log(1/\delta)}{\epsilon^2} \cdot \frac{\sqrt{n}}{I[f]}\right)$.

Observe that if ϵ is a constant, then the required lower bound on $I[f]$ is exponentially small in n which is a rather weak assumption, and as long as $\epsilon \geq c'\sqrt{\log n/n}$ for some constant c', the lower bound is (inverse) polynomial in n. On the other hand, if $\epsilon < c'\sqrt{\log n/n}$ then we can easily get a $(1 \pm \epsilon)$-approximation by simply sampling $O\left(\frac{\sqrt{n}}{I[f]} \cdot \mathrm{poly}(1/\epsilon)\right) = O\left(\frac{n}{I[f]} \cdot \mathrm{poly}(1/\epsilon)\right)$ edges (as explained shortly).

We also prove a nearly matching lower bound of $\Omega\left(\frac{\sqrt{n}}{\log n \cdot I[f]}\right)$ on the query complexity of any constant-factor approximation algorithm for this problem (which holds for $I[f] = \Omega(1)$):

[2] Here we referenced several works in which the influence appears explicitly. The influence of variables plays an implicit role in many learning algorithms, and in particular those that build on Fourier analysis, beginning with [31].

[3] If one wants an *additive* error of ϵ, then $\Omega((n/\epsilon)^2)$ queries are necessary (when the influence is large) [33].

Theorem 1.2 *For every I^* such that $2 \leq I^* \leq \sqrt{n}/\log n$, there exists a family of monotone functions F_{I^*} and a function g for which the following holds. For every $f \in F_{I^*}$ we have that $I[f] \geq I^*$, while $I[g] = 1 - o(1)$, but any algorithm that distinguishes with probability at least $2/3$ between a uniformly selected function in F_{I^*} and g must perform $\Omega\left(\frac{\sqrt{n}}{I^* \cdot \log n}\right)$ queries.*

The high level idea for the algorithm. As noted above, the influence of a function can be approximated by sampling random edges (i.e., pairs $(x, x^{(\oplus i)})$ that differ on a single coordinate) from the Boolean hypercube. A random edge has probability exactly $\frac{I[f]}{n}$ to be influential (i.e, satisfy $f(x) \neq f(x^{(\oplus i)})$), so a standard sampling argument implies that it suffices to ask $O\left(\frac{n}{I[f]}\mathrm{poly}(1/\epsilon)\right)$ queries in order to approximate this probability to within $(1 \pm \epsilon)$. We also note that in the case of monotone functions, it is well known that the total influence equals twice the sum of the Fourier coefficients that correspond to singleton sets $\{i\}$, $i \in \{1, \ldots, n\}$. Therefore, it is possible to approximate the influence of a function by approximating this sum, which equals $\frac{1}{2^n} \cdot \sum_{i=1}^{n} \left(\sum_{x \in \{0,1\}^n : x_i = 1} f(x) - \sum_{x \in \{0,1\}^n : x_i = 0} f(x)\right)$. However, the direct sampling approach for such an approximation again requires $\Omega(n/I[f])$ samples, since an algorithm performing $o(n/I[f])$ (random) queries will fail to observe any influential edge with high probability, that is, the above sum would be 0.

In order to achieve better query complexity, we would like to increase the "success probability" of a single test, that is, the probability of hitting an influential edge in a single trial. The algorithm we present captures this intuition by selecting pairs of points x, y that are endpoints of *chains* in the Boolean hypercube. A chain is a sequence of points $z^1, \ldots, z^t$ such that $z^j \prec z^{j+1}$ for each $1 \leq j \leq t-1$. We consider chains in which z^j and z^{j+1} differ on a single coordinate and hence have an edge between them. A simple but important observation is that if a function f is monotone, then every chain contains at most one influential edge with respect to f, and if a chain contains such an edge, then f assigns its endpoints different values.

A well know result of Dilworth [13] show that $\{0,1\}^n$ can be partitioned into $N = \binom{n}{\lfloor n/2 \rfloor} = \Theta(2^n/\sqrt{n})$ disjoint chains, and furthermore, this can be done constructively (e.g., [23]). Since this is a partition of the vertices of the Boolean hypercube, it covers all vertices, but it does not cover all edges, and in particular may not cover all (or even any) influential edges, which is what we are interested in. However, by considering all partitions that are induced by taking one single partition and applying all possible permutations over the n coordinates, we do obtain a covering of all edges. Furthermore, almost all edges are covered roughly the same number of times. This fact, together with the fact that the number of edges in each chain is $\sqrt{n}$ (on the average), is what allows us to save a factor of $\sqrt{n}$ in the query complexity.

1.2 A Note on Testing Monotonicity

A natural question is whether an algorithm in the spirit of the one described in this paper can be used for testing monotonicity of Boolean functions over the Boolean hypercube. An algorithm for testing monotonicity is given query access to a function $f : \{0,1\}^n \to \{0,1\}$, and is required to distinguish (with high constant success probability) between the case that f is a monotone function and the case that f must be modified on more than an ϵ-fraction of the domain so that it become monotone (where ϵ is a given distance parameter). The best known algorithm for testing monotonicity of Boolean functions over the Boolean hypercube performs $O(n/\epsilon)$ queries [22]. This algorithm uniformly selects $O(n/\epsilon)$ pairs of points $(x, x^{(\oplus i)})$ where $x \in \{0,1\}^n$ and $i \in \{1, \ldots, n\}$, and rejects if an only if it encounters a violation of monotonicity. Indeed, there are functions that are far from being monotone, for which it is necessary to select $\Omega(n)$ such

pairs in order to observe a violation [22]. It is an open problem whether this complexity can be reduced by considering pairs of points $x, y \in \{0,1\}^n$ which are the endpoints of a chain *of length greater than* 1 of the hypercube, and checking whether f violates monotonicity on each selected pair. It is known that every non-adaptive one-sided error algorithm for testing monotonicity of Boolean functions over the Boolean hypercube (which must find a violating pair of this form), needs to perform $\Omega(\sqrt{n})$ queries [17], and it is possible that $O(\sqrt{n})$ queries suffice. We note that if the range is of size $\Omega(\sqrt{n})$ (and the domain is still the n-dimensional Boolean hypercube), then $\Omega(n)$ queries are necessary for testing monotonicity [6].

2 Preliminaries

In the introduction we defined the influence of a function as the sum of its individual influences $I_i[f]$. Equivalently, the influence of a function f is the expected number of sensitive coordinates for a random input $x \in \{0,1\}^n$ (that is, those coordinates i for which $f(x) \neq f(x^{(\oplus i)})$). In this context, it will be convenient to view f as a 2-coloring of the Boolean hypercube. Under this setting, any "bi-chromatic" edge, i.e, an edge $(x, x^{(\oplus i)})$ such that $f(x) \neq f(x^{(\oplus i)})$, will be called an *influential edge*. The number of influential edges of a Boolean function f is[4] $2^{n-1} \cdot I[f]$.

Notations. We use the notation $f(n) = \tilde{O}(g(n))$ if $f(n) = O(g(n)\text{polylog}(g(n)))$. Similarly, $f(n) = \tilde{\Omega}(g(n))$ if $f(n) = \Omega(g(n)/\text{polylog}(g(n)))$. For a point $x \in \{0,1\}^n$, let $w(x) = \sum_{i=1}^n x_i$ denote the Hamming weight of x. We think of the Boolean hypercube as consisting of $n+1$ *levels*, where in each level all points have the same Hamming weight.

3 The Algorithm

As noted in the introduction, our algorithm selects pairs of points x, y that are endpoints of certain chains in the Boolean hypercube (so that in particular $x \prec y$). The algorithm queries f on these endpoints and uses the number of pairs that are assigned different values by f to obtain an estimate for $I[f]$. The main issue that needs to be addressed is what is the distribution over such pairs, and why does it give a good estimate.

From this point on we assume that n is odd, so that the number of levels in the n-dimensional Boolean hypercube is even. This can be done without loss of generality, since if n is even, then we can consider the function $f' : \{0,1\}^{n+1} \to \{0,1\}$ defined by $f'(x_1, \ldots, x_n, x_{n+1}) = f(x_1, \ldots, x_n)$, so that $I[f'] = I[f]$.

Our starting point is that $\{0,1\}^n$ can be partitioned into $N = \binom{n}{\lfloor n/2 \rfloor} = \Theta(2^n/\sqrt{n})$ disjoint symmetric chains [13] (where chains are as defined in the introduction). In *symmetric* we mean that for each chain, one endpoint belongs to level $(n-1)/2+t$ and the other to level $(n-1)/2-t$, for some $0 \leq t \leq (n+1)/2$. In particular, there is a relatively simple constructive partition due to Greene and Kleitman [23]. Furthermore, the construction of Greene and Kleitman is such that given a point x in $\{0,1\}^n$ we can construct the chain x belongs to in time polynomial in n (more precisely, in time $O(n \cdot |C(x)|)$ where $|C(x)|$ is the size of the chain that x belongs to). While this is immaterial to the bound we obtain on the query complexity of our algorithm, it implies that its running time is polynomial in n (rather than exponential in n).

As noted in the introduction, for a monotone function f, a chain can contain at most one influential edge with respect to f, and if it contains such an edge, then its endpoints have different values (according to

[4] To verify this, observe that when partitioning the Boolean hypercube into two sets with respect to a coordinate i, we end up with 2^{n-1} vertices in each set. The individual influence of variable i, $I_i[f]$, is the fraction of the "bi-chromatic" edges among all edges crossing the cut. Since $I[f] = \sum_{i=1}^n I_i[f]$ we get that the total number of influential edges is $2^{n-1} \cdot I[f]$.

f). Since the chains in the aforementioned partition are vertex-disjoint, they are clearly also edge-disjoint. However, while the partition into chains covers all vertices, it does not cover all edges, and in particular does not cover all influential edges. In order to cover all (influential) edges, we take a union over many partitions into chains, as described next.

For a permutation $\sigma \in S_n$ and a point $x \in \{0,1\}^n$, let $\sigma(x) \stackrel{\text{def}}{=} x_{\sigma(1)} \ldots x_{\sigma(n)}$. Fixing a partition $\mathcal{P}$ of $\{0,1\}^n$ into N chains, each permutation $\sigma \in S_n$ defines a partition $\mathcal{P}_\sigma$, where for each chain $C = z^1, \ldots, z^t$ in $\mathcal{P}$, there is a chain $\sigma(C) = \sigma(z^1), \ldots, \sigma(z^t)$ in $\mathcal{P}_\sigma$. For an edge $e = (x, y)$ and a permutation σ, we shall use the (slightly abused) notation $e \in \mathcal{P}_\sigma$ to mean that there exists a chain $C = z^1, \ldots, z^t$ in $\mathcal{P}_\sigma$ such that $x = z^j$ and $y = z^{j+1}$ for some $1 \leq j \leq t-1$ (i.e., $\mathcal{P}_\sigma$ *covers* the edge e).

If we consider the union of all chains in $\mathcal{P}_\sigma$, taken over all permutations σ, then we cover all edges. We next show that most edges are covered roughly the same number of times. We shall use the following notation: For an edge $e = (x, y)$ in the Boolean hypercube where $x \prec y$, if $w(y) > n/2$ then $w(e) = w(y)$, and otherwise $w(e) = n - w(x)$ (recall that $w(x)$ denotes the Hamming weight of x). Observe that by this definition, $w(e) \geq n/2$ for every edge e.

Lemma 3.1 *For every edge e and for a uniformly selected permutation $\sigma \in S_n$, $\Pr_\sigma[e \in \mathcal{P}_\sigma] = 1/w(e)$.*

Proof: In order to prove the lemma we shall show that the number of permutations σ such that $e \in \mathcal{P}_\sigma$ is $n!/w(e)$. Let $e = (x, y)$, and assume that $w(y) > n/2$, so that $w(e) = w(y)$. (The case that $w(x) < n/2$, so that $w(e) = n - w(x)$ is treated in a similar manner.) Let $x^1, \ldots, x^{w(y)}$ denote the neighbors of y whose Hamming weight is the same as that of x, where $x = x^j$ for some $1 \leq j \leq w(y)$. Observe that for every σ, the point y belongs to exactly one chain in $\mathcal{P}_\sigma$, and exactly one among $x^1, \ldots, x^{w(y)}$ belongs to the same chain. For each $1 \leq j \leq w(y)$, let $R(j)$ consist of all permutations σ such that x^j belongs to the same chain as y in $\mathcal{P}_\sigma$. It remains to show that all sets $R(j)$ have the same size.

To this end, Let x^i and x^j be (any) two neighbors of y obtained by flipping coordinates $c(i)$ and $c(j)$ of y from 1 to 0 correspondingly. We claim that $|R(i)| = |R(j)|$. Indeed, let τ be a permutation preserving y (i.e, $\tau(y_1) \ldots \tau(y_n) = y$) but moving x^i to x^j (for example, an involution between $c(i)$ and $c(j)$). Then $(y, x_i) \in P_\sigma$ if and only if $(y, x_j) \in P_{\tau\sigma}$. This establishes a one to one mapping between $R(i)$ and $R(j)$. ■

We establish one more lemma before presenting the algorithm. We shall say that a chain C is *influential* if C contains an influential edge.

Lemma 3.2 *Suppose that $I[f] \geq 16n \cdot e^{-\epsilon^2 n/96}/\epsilon$ and we uniformly select a random permutation $\sigma \in S_n$ and a random chain C in $\mathcal{P}_\sigma$. Then*

$$(1 - \epsilon/2)\frac{I[f] \cdot 2^n}{n \cdot N} \leq \Pr_{\sigma, C \in \mathcal{P}_\sigma}[C \textit{ is influential }] \leq \frac{I[f] \cdot 2^n}{n \cdot N} \ .$$

Proof: Let $\mathcal{E}(f)$ denote the set of influential edges of f (so that $|\mathcal{E}(f)| = 2^{n-1} \cdot I[f]$). Since any chain can contain at most one edge $e \in \mathcal{E}(f)$, and applying Lemma 3.1,

$$\Pr_{\sigma, C \in \mathcal{P}_\sigma}[C \text{ is influential }] = \sum_{e \in \mathcal{E}(f)} \Pr_\sigma[e \in \mathcal{P}_\sigma] \cdot \frac{1}{N} = \frac{1}{N} \sum_{e \in \mathcal{E}(f)} \frac{1}{w(e)} \ . \tag{1}$$

On one hand, since $w(e) > n/2$ for every edge e, we have that

$$\sum_{e \in \mathcal{E}(f)} \frac{1}{w(e)} < |\mathcal{E}(f)| \cdot \frac{2}{n} = \frac{2^n \cdot I[f]}{n} \ . \tag{2}$$

On the other hand, if we let $\mathcal{E}_\epsilon(f) = \{e \in \mathcal{E}(f) \ : \ w(e) \leq (1+\epsilon/4)(n/2)\}$, then

$$\sum_{e \in \mathcal{E}(f)} \frac{1}{w(e)} \geq \sum_{e \in \mathcal{E}_\epsilon(f)} \frac{1}{w(e)} \geq |\mathcal{E}_\epsilon(f)| \cdot \frac{1}{1+\epsilon/4} \cdot \frac{2}{n} \geq |\mathcal{E}_\epsilon(f)| \cdot (1-\epsilon/4) \cdot \frac{2}{n} \ . \tag{3}$$

Since the total number of edges e for which $w(e) > (1+\epsilon/4)(n/2)$ is upper bounded by $n \cdot 2^n \cdot 2e^{-\epsilon^2 n/96}$, we have that

$$\sum_{e \in \mathcal{E}_\epsilon(f)} \frac{1}{w(e)} \geq \left(|\mathcal{E}(f)| - n \cdot 2^{n+1} \cdot e^{-\epsilon^2 n/96}\right) \cdot (1-\epsilon/4) \cdot \frac{2}{n} \ . \tag{4}$$

By the premise of the lemma, $I[f] \geq 16ne^{-\epsilon^2 n/96}/\epsilon$, so that $n \cdot 2^{n+1} \cdot e^{-\epsilon^2 n/96} \leq (\epsilon/4)|\mathcal{E}(f)|$, and we get that

$$\sum_{e \in \mathcal{E}(f)} \frac{1}{w(e)} \geq (1-\epsilon/2) \cdot \frac{2^n \cdot I[f]}{n} \ . \tag{5}$$

The Lemma follows by combining Equations (1), (2), and (5). ■

Given Lemma 3.2 and the fact that $\frac{2^n}{n \cdot N} = \Theta(1/\sqrt{n})$, in order to obtain an estimate of $I[f]$ that is accurate up to a factor of $(1 \pm \epsilon)$ with high success probability (for $I[f]$ that is not too small), it suffices to take a sample of $\Theta(\sqrt{n}/I[f])$ chains according to the distribution induced by selecting $\sigma \in S_n$ uniformly and then selecting a chain $C \in \mathcal{P}_\sigma$ uniformly. Albeit, the sample size depends on $I[f]$ which is (obviously) not known, and so our algorithm takes a slightly different approach, described in Figure 1 below.

Algorithm 1: **Approximating the Influence (given ϵ, δ and oracle access to f)**

1. *Let $t = \frac{24 \ln\left(\frac{2}{\delta}\right)}{\epsilon^2}$, and initialize $\alpha \leftarrow 0$, $m \leftarrow 0$, and $\hat{I} \leftarrow 0$.*
2. *Repeat the following until $\alpha = t$:*
   (a) *Select $\sigma \in S_n$ uniformly at random and select a chain $C \in \mathcal{P}_\sigma$ uniformly at random, where $\mathcal{P}$ is the partition defined by the construction in [23]. (This step can be implemented by selecting a point z such that $w(z) = \lceil n/2 \rceil$ uniformly at random, selecting a permutation σ uniformly at random, and constructing the chain that $\sigma(z)$ belongs to.)*
   (b) *Let $b(C)$ and $t(C)$ be the bottom and top endpoints of C, respectively.*
   (c) *If $f(b(C)) \neq f(t(C))$, then $\alpha \leftarrow \alpha + 1$.*
   (d) $m \leftarrow m + 1$
3. *$\hat{I} \leftarrow \frac{n \cdot N}{2^n} \cdot \frac{t}{m}$, and return $\hat{I}$.*

Figure 1: The algorithm for approximating the influence of a monotone function f.

The following theorem asserts the correctness and desired query complexity of Algorithm 1, thereby proving Theorem 1.1.

Theorem 3.3 *For every $\delta, \epsilon > 0$ and for every monotone function $f : \{0,1\}^n \to \{0,1\}$ such that $I[f] \geq 16ne^{-\epsilon^2 n/96}/\epsilon$, with probability at least $1-\delta$, the output, $\hat{I}$, of Algorithm 1 satisfies:*

$$(1-\epsilon) \cdot I[f] \leq \hat{I} \leq (1+\epsilon) \cdot I[f] \ .$$

Furthermore, with probability at least $1-\delta$*, the number of queries performed by the algorithm is* $O\left(\frac{\log(1/\delta)}{\epsilon^2}\cdot\frac{\sqrt{n}}{I[f]}\right)$.

We note that the bound on the number of queries performed by the algorithm implies that the expected query complexity of the algorithm is $O\left(\frac{\log(1/\delta)}{\epsilon^2}\cdot\frac{\sqrt{n}}{I[f]}\right)$. Furthermore, the probability that the algorithm performs a number of queries that is more than k times the expected value decreases exponentially with k.

Proof: Let $p(f)$ denote the probability that a chain C (selected uniformly in $\mathcal{P}_\sigma$ for a uniformly selected $\sigma \in S_n$) is influential. By Lemma 3.2 (and the lower bound on $I[f]$), we have that $(1-\epsilon/2)\frac{I[f]\cdot 2^n}{n\cdot N} \leq p(f) \leq \frac{I[f]\cdot 2^n}{n\cdot N}$. Recall that m is a random variable denoting the number of iterations performed by the algorithm until it stops (once $\alpha = t$). Let $\tilde{m} = \frac{t}{p(f)}$, $\tilde{m}_1 = \frac{\tilde{m}}{(1+\epsilon/2)}$, and $\tilde{m}_2 = \frac{\tilde{m}}{(1-\epsilon/2)}$. We say that an iteration of the algorithm is *successful* if an influential chain is selected in the iteration. Let $\hat{p}(f) = \frac{t}{m}$ denote the fraction of successful iterations.

Suppose that $\tilde{m}_1 \leq m \leq \tilde{m}_2$. In such a case,

$$(1-\epsilon/2)\cdot p(f) \;\leq\; \hat{p}(f) \;\leq\; (1+\epsilon/2)p(f)\;, \tag{6}$$

since $\hat{p}(f) = \frac{t}{m} = \frac{p(f)\cdot\tilde{m}}{m}$. By the definition of the algorithm, $\hat{I} = \frac{n\cdot N}{2^n}\cdot\frac{t}{m} = \frac{n\cdot N}{2^n}\cdot\hat{p}(f)$, and so we get that $(1-\epsilon)\cdot I[f] \leq \hat{I} \leq (1+\epsilon/2)\cdot I[f]$ (assuming $\tilde{m}_1 \leq m \leq \tilde{m}_2$).

It remains to prove that $\tilde{m}_1 \leq m \leq \tilde{m}_2$ with probability at least $1-\delta$. Let X_i denote the indicator random variable whose value is 1 if and only if the i^{th} iteration of the algorithm was successful, and let $X = \sum_{i=1}^{\tilde{m}_1} X_i$. By the definition of X_i, we have that $\mathrm{E}[X_i] = p(f)$, and so (by the definition of $\tilde{m}_1$ and $\tilde{m}$) we have that $\mathrm{E}[X] = \tilde{m}_1\cdot p(f) = \frac{t}{1+\epsilon/2}$. Hence, by applying the multiplicative Chernoff bound [11],

$$\Pr[m < \tilde{m}_1] \;\leq\; \Pr[X \geq t] \;=\; \Pr[X \geq (1+\epsilon/2)\mathrm{E}[X]] \;\leq\; \exp(-\epsilon^2 t/24)\;. \tag{7}$$

By an analogous argument we get that $\Pr[m > \tilde{m}_2] \leq \exp(-\epsilon^2 t/16)$, and so choosing $t = \frac{24\ln(\frac{2}{\delta})}{\epsilon^2}$ ensures that both events happen with probability at most $\delta/2$, and so $\Pr[\tilde{m}_1 \leq m \leq \tilde{m}_2] \geq 1-\delta$, as desired.

Since in particular $m \leq \tilde{m}_2$ with probability at least $1-\delta$, and the query complexity of the algorithm is $O(m)$, we have that, with probability at least $1-\delta$, the query complexity is upper bounded by

$$O(\tilde{m}_2) \;=\; O\left(\frac{t}{p(f)}\right) \;=\; O\left(\frac{t\cdot n\cdot N}{2^n\cdot I[f]}\right) \;=\; O\left(\frac{\log(1/\delta)}{\epsilon^2}\cdot\frac{\sqrt{n}}{I[f]}\right) \tag{8}$$

as required. ■

4 A Lower Bound

In this section we prove a lower bound of $\Omega\left(\frac{\sqrt{n}}{I[f]\cdot\log n}\right)$ on the query complexity of approximating the influence of monotone functions. Following it we explain how a related construction gives a lower bound of $\Omega\left(\frac{n}{I[f]}\right)$ on approximating the influence of *general* functions. The idea for the first lower bound is the following. We show that any algorithm that performs $o\left(\frac{\sqrt{n}}{I[f]\cdot\log n}\right)$ queries cannot distinguish with constant success probability between the following: (1) A certain threshold function (over a relatively small number

of variables), and (2) A function selected uniformly at random from a certain family of functions that have significantly higher influence than the threshold function. The functions in this family can be viewed as "hiding their influence behind the threshold function". More precise details follow.

We first introduce one more notation. For any integer $1 \leq k \leq n$ and $0 \leq t \leq k$, let $\tau_k^t : \{0,1\}^n \to \{0,1\}$ be the *t-threshold function* over $x_1, \ldots, x_k$. That is, $\tau_k^t(x) = 1$ if and only if $\sum_{i=1}^k x_i \geq t$. Observe that (since for every $1 \leq i \leq k$ we have that $I_i[\tau_k^t] = 2^{-k} \cdot 2 \cdot \binom{k-1}{t-1}$ while for $i > k$ we have that $I_i[\tau_k^t] = 0$), $I[\tau_k^t] = k \cdot 2^{-(k-1)} \cdot \binom{k-1}{t-1}$.

The above observation implies that for every sufficiently large k ($k \geq 2 \log n$ suffices), there exists a setting of $t < k/2$, which we denote by $t(k,1)$, such that $I[\tau_k^{t(k,1)}] = 1 - o(1)$ (where the $o(1)$ is with respect to k). This setting satisfies $\binom{k-1}{t(k,1)-1} = \Theta(2^k/k)$ (so that $t(k,1) = k/2 - \Theta(\sqrt{k \log k})$).
We are now ready to prove Theorem 1.2. For clarity, we restate it in terms of our new notation.

Theorem 4.1 *For every I^* such that $2 \leq I^* \leq \sqrt{n}/\log n$, there exists a family of monotone functions F_{I^*} such that $I[f] \geq I^*$ for every $f \in F_{I^*}$, but any algorithm that distinguishes with probability at least $2/3$ between a uniformly selected function in F_{I^*} and $\tau_k^{t(k,1)}$ for $k = 2 \log n$, must perform $\Omega\left(\frac{\sqrt{n}}{I^* \cdot \log n}\right)$ queries.*

In particular, considering $I^* = c$ for any constant $c \geq 2$, we get that every algorithm for approximating the influence to within a multiplicative factor of $\sqrt{c}$ must perform $\tilde{\Omega}(\sqrt{n})$ queries. If we increase the lower bound on the influence, then the lower bound on the complexity of the algorithm decreases, but the approximation factor (for which the lower bound holds), increases. We note that the functions for which the lower bound construction hold are not balanced, but we can easily make them very close to balanced without any substantial change in the argument (by "ORing" $\tau_k^{t(k,1)}$ as well as every function in F_{I^*} with x_1). We also note that for $I^* = \Omega(\sqrt{\log n})$ we can slightly improve the lower bound on approximating the influence to $\Omega\left(\frac{\sqrt{n}}{I^* \cdot \sqrt{\log(\sqrt{n}/I^*)}}\right)$ (for a slightly smaller approximation factor). We address this issue following the proof.

Proof: For $k = 2 \log n$ and for any $0 \leq t \leq k$, let $L_k^t \stackrel{\text{def}}{=} \{x \in \{0,1\}^k : \sum_{i=1}^k x_i = t\}$. We shall also use the shorthand $\tilde{t}$ for $t(k,1)$. Fixing a choice of I^*, each function in F_{I^*} is defined by a subset R of $L_k^{\tilde{t}}$ where $|R| = \beta(I^*) \cdot 2^k$ for $\beta(I^*)$ that is set subsequently. We denote the corresponding function by f_R and define it as follows: For every $x \in \{0,1\}^n$, if $x_1, \ldots, x_k \notin R$, then $f_R(x) = \tau_k^{\tilde{t}}(x)$, and if $x_1, \ldots, x_k \in R$, then $f_R(x) = \mathrm{maj}'_{n-k}(x)$, where $\mathrm{maj}'_{n-k}(x) = 1$ if and only if $\sum_{i=k+1}^n x_i > (n-k)/2$. By this definition, each $f_R \in F_{I^*}$ is a monotone function, and

$$I[f_R] \geq \beta(I^*) \cdot I[\mathrm{maj}'_{n-k}] \,. \tag{9}$$

If we take $\beta(I^*)$ to be $\beta(I^*) = I^*/I[\mathrm{maj}'_{n-k}] = cI^*/\sqrt{n-k}$ (for c that is roughly $\sqrt{\pi/2}$), then in F_{I^*} every function has influence at least I^*. Since $\beta(I^*)$ is upper bounded by $|L_k^{\tilde{t}}|/2^k$, which, (by the definition of $\tilde{t} = t(k,1)$), is of the order of $1/k = \Theta(1/\log n)$ this construction is applicable so long as $I^* \leq c\sqrt{n}/\log n$ for sufficiently small c.

Consider an algorithm that needs to distinguish between $\tau_k^{\tilde{t}}$ and a uniformly selected $f_R \in F_{I^*}$. Clearly, as long as the algorithm doesn't perform a query on x such that $x_1, \ldots, x_k \in R$, the value returned by f_R is the same as that of $\tau_k^{\tilde{t}}$. But since R is selected uniformly in $L_k^{\tilde{t}}$, as long as the algorithm performs less than $\frac{|L_k^{\tilde{t}}|}{c' \cdot \beta(I^*) \cdot 2^k}$ queries (where c' is some sufficiently large constant), with high constant probability (over the choice of R), it will not "hit" a point in R. Since $\frac{|L_k^{\tilde{t}}|}{c' \cdot \beta(I^*) \cdot 2^k} = \Theta\left(\frac{\sqrt{n}}{\log n \cdot I^*}\right)$, the theorem follows. ■

In order to get the aforementioned slightly higher lower bound for $I^* = \Omega(\sqrt{\log n})$, we modify the settings in the proof of Theorem 4.1 in the following manner. We set $k = \log(\sqrt{n}/I^*)$ and $t = k/2$ (so that the "low influence" function is simply a majority function over k variables, $\tau_k^{k/2}$). For the "high influence" function, we let R consist of a single point $\tilde{x}$ in $L_k^{k/2}$, where for each $R = \{\tilde{x}\}$ we have a different function in F_{I^*} (as defined in the proof of Theorem 4.1). It follows that for each such R, $I[f_R] = (1 - o(1))\sqrt{k} + \frac{1}{2^k}\sqrt{n-k} \geq I^*$, while $I[\tau_k^{k/2}] \approx \sqrt{k} = O(\sqrt{\log n})$. By the same argument as in the proof of Theorem 4.1, if the algorithm performs less than $\frac{c'|L_k^{k/2}|}{|R|} = \frac{c''2^k}{\sqrt{k}} = \frac{c''\sqrt{n}}{I^*\sqrt{\log(\frac{\sqrt{n}}{I^*})}}$ queries (for small enough c', which determines the constant c''), with high probability it will not "hit" $\tilde{x}$, and thus will not be able to distinguish between a randomly selected function $f \in f_R$ (where the randomness is over the choice of $\tilde{x} \in L_k^{k/2}$) and $\tau_k^{k/2}$.

A lower bound of $\Omega(n/I[f])$ for general functions. We note that for general (not necessarily monotone) functions, we can obtain a stronger lower bound of $\Omega(n/I[f])$ on estimating the influence, which implies that it is not possible in general to improve on the simple edge-sampling approach (in terms of the dependence on n and $I[f]$). The idea behind the following construction is similar to the one used in the previous section, only now the lack of the monotonicity constraint allows us to "hide" the influential part of the function "behind" an *arbitrary* small set R of assignments to the first k variables. We then compensate for the small probability of hitting R with a function having very high influence (Parity) over the remaining $n - k$ variables. More precisely, we prove the following:

Theorem 4.2 *For every I^* in the range $2 \leq I^* \leq n$, there exists a family of functions F_{I^*} such that $I[f] \geq I^*$ for every $f \in F_{I^*}$, but any algorithm that distinguishes with probability at least $2/3$ between a uniformly selected function in F_{I^*} and the dictatorship function $D_1(x) = x_1$, must perform $\Omega(n/I^*)$ queries.*

Proof: First, note that the theorem is trivial for $I^* = \Omega(n)$, since any algorithm must perform at least one query to accomplish the above task, so we may henceforth assume that $I^* = o(n)$. Once again, we consider the first k variables, where here $k = \log n$.

Fixing I^*, each function in F_{I^*} is defined by a subset R of $\{0,1\}^k$ such that $|R| = 2I^*$. We denote the corresponding function by f_R and define it as follows: For every $x \in \{0,1\}^n$, if $x_1, \ldots, x_k \notin R$ then $f_R(x) = x_1$, and if $x_1, \ldots, x_k \in R$, we let $f_R(x) = \bigoplus_{i=k+1}^n x_i$. By this definition, and since the influence of $\bigoplus_{i=k+1}^n x_i$ is $n - k$ (every bit is influential w.p 1), we have that for every $f_R \in F_{I^*}$, it holds that

$$I[f_R] = (1 - 2I^*/2^k) + (2I^*/2^k) \cdot (n-k) = I^* + 1 + I^*(1 - 2(\log n + 1)/n) \geq I^* \,, \qquad (10)$$

where we use the fact that $k = \log(n)$.

Consider an algorithm that needs to distinguish between D_1 and a uniformly selected $f_R \in F_{I^*}$ (that is, f_R for a uniformly random subset $R \subset [k]$). Clearly, as long as the algorithm doesn't perform a query on x such that $x_1, \ldots, x_k \in R$, the value returned by f_R is the same as that of D_1. But since R is selected uniformly in $\{0,1\}^k$, as long as the algorithm performs less than $\frac{2^k}{c' \cdot |R|} = \frac{n}{c' \cdot 2I^*}$ queries (where c' is some sufficiently large constant), with high constant probability (over the choice of R), it will not "hit" a point in R, and hence will not be able to distinguish between the aforementioned classes. ■

Acknowledgements. We would like to thank several anonymous reviewers for their helpful comments.

References

[1] K. Arrow. A difficulty in the theory of social welfare. *Journal of political economics*, 58:328–346, 1950.

[2] M. Ben-Or and N. Linial. Collective coin flipping, robust voting schemes and minima of Banzhaf values. In *Proceedings of FOCS*, pages 408–416, 1985.

[3] I. Benjamini, G. Kalai, and O. Schramm. Noise sensitivity of boolean functions and applications to percolation. *Inst. Hautes Etudes Sci. Publ. Math.*, 90:5–43, 1999.

[4] E. Blais. Improved bounds for testing juntas. In *Proceedings of RANDOM*, pages 317–330, 2008.

[5] E. Blais. Testing juntas nearly optimally. In *Proceedings of STOC*, pages 151–158, 2009.

[6] E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. In *26th annual Conference on Computational Complexity (CCC)*, 2011.

[7] R. Boppana. The average sensitivity of bounded depth circuits. *Information Processing Letters*, 63:257–261, 1997.

[8] J. Bourgain, J. Kahn, G. Kalai, Y. Katznelson, and N. Linial. The influence of variables in product spaces. *Israel Journal of Math*, 77:55–64, 1992.

[9] J. Bourgain and G. Kalai. Influences of variables and threshold intervals under group symmetries. *Geometric Functional Analaysis*, 7:438–461, 1997.

[10] N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.

[11] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of the Mathematical Statistics*, 23:493–507, 1952.

[12] I. Diakonikolas, P. Harsha, A. Klivans, R. Meka, P. Raghavendra, R. Servedio, and L.-Y. Tan. Bounding the average sensitivity and noise sensitivity of polynomial threshold functions. In *Proceedings of STOC*, pages 533–543, 2010.

[13] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950.

[14] I. Dinur, E. Friedgut, G. Kindler, and R. O'Donnell. On the Fourier tails of bounded functions over the discrete cube. *Israel Journal of Mathematics*, 160(1):389–412, 2007.

[15] I. Dinur and S. Safra. The importance of being biased. In *Proceedings of STOC*, pages 33–42, 2002.

[16] E. Fischer, G. Kindler, D. Ron, S. Safra, and S. Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004.

[17] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of STOC*, pages 474–483, 2002.

[18] E. Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):27–36, 1998.

[19] E. Friedgut. Influences in product spaces: KKL, BKKKL revisited. *Combinatorics, Probability and Computing*, 13:17–29, 2004.

[20] E. Friedgut and G. Kalai. Every monotone graph property has a sharp threshold. In *Proc. Amer. Math. Soc. 124*, pages 2993–3002, 1996.

[21] G.Kalai and S.Safra. Threshold phenomena and influence. *Computational Complexity and Statistical Physics*, pages 25–60, 2006.

[22] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

[23] C. Greene and D. J. Kleitman. Strong version of Sperner's theorem. *Journal of Combinatorial Theory Ser A*, 20(1):80–88, 1976.

[24] T. Hancock and Y. Mansour. Learning monotone k-μ DNF formulas on product distributions. In *Proceedings of COLT*, pages 179–183, 1991.

[25] G. Kalai. A Fourier-theoretic perspective for the Condorcet Paradox and Arrow's Theorem. *Advances in Applied Mathematics*, 29:412–426, 2002.

[26] G. Kalai. Social indeterminacy. *Econometrica*, 72:1565–1581, 2004.

[27] G. Kalai, J. Kahn, and N. Linial. The influence of variables on Boolean functions. In *Proceedings of FOCS*, pages 68–80, 1988.

[28] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of STOC*, pages 767–775, 2002.

[29] M. Krivelevich, B. Sudakov, and V. H. Vu. A sharp threshold for network reliability. *Combinatorics, Probability and Computing*, 11:465–474, 2002.

[30] E. Lehrer. An axiomatization of the Banzhaf value. *International Journal of Game Theory*, 17:89–99, 1988.

[31] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

[32] K. Matulef, R. O'Donnell, R. Rubinfed, and R. A. Servedio. Testing $\{-1, +1\}$ halfspaces. In *Proceedings of RANDOM*, pages 646–657, 2009.

[33] K. Matulef, R. Servedio, and K. Wimmer. Personal communication, 2009.

[34] R. O'Donnell, M. Saks, O. Schramm, and R. Servedio. Every decision tree has an influential variable. In *Proceedings of FOCS*, pages 31–39, 2005.

[35] R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. *SIAM Journal on Computing*, 37(3):827–844, 2007.

[36] R. O'Donnell and R. Servedio. The Chow parameters problem. In *Proceedings of STOC*, pages 517–526, 2008.

[37] D. Ron and G. Tsur. Testing computability by width-2 OBDDs. In *Proceedings of RANDOM*, pages 686–699, 2009.

[38] L. Russo. On the critical percolation probabilities. *Z. Wahrsch. Verw. Geb.*, 56:229–237, 1981.

[39] M. Talagrand. On Russo's approximate $0-1$ law. *Annals of Probability*, 22:1576–1587, 1994.

[40] M. Talagrand. How much are increasing sets correlated. *Combinatorica*, 16:243–258, 1996.

[41] M. Talagrand. On boundaries and influences. *Combinatorica*, 17:275–285, 1997.