58th Annual IEEE Symposium on Foundations of Computer Science

Boolean Unateness Testing with $\widetilde{O}(n^{3/4})$ Adaptive Queries

Xi Chen
Department of Computer Science
Columbia University
New York, USA
Email: xichen@cs.columbia.edu

Erik Waingarten
Department of Computer Science
Columbia University
New York, USA
Email: eaw@cs.columbia.edu

Jinyu Xie
Department of Computer Science
Columbia University
New York, USA
Email: jinyu@cs.columbia.edu

***Abstract*—We give an adaptive algorithm that tests whether an unknown Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is unate (i.e. every variable of f is either non-decreasing or non-increasing) or ϵ-far from unate with one-sided error and $\widetilde{O}(n^{3/4}/\epsilon^2)$ many queries. This improves on the best adaptive $O(n/\epsilon)$-query algorithm from Baleshzar, Chakrabarty, Pallavoor, Raskhodnikova and Seshadhri [1] when $1/\epsilon \ll n^{1/4}$. Combined with the $\widetilde{\Omega}(n)$-query lower bound for non-adaptive algorithms with one-sided error of [2], [3], we conclude that adaptivity helps for the testing of unateness with one-sided error. A crucial component of our algorithm is a new subroutine for finding bi-chromatic edges in the Boolean hypercube called adaptive edge search.**

***Keywords*-property testing; unateness;**

I. Introduction

A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is *monotone* if every variable of f is non-decreasing, and is *unate* if every variable of f is either non-decreasing or non-increasing (or equivalently, there exists a string $r \in \{0,1\}^n$ such that $g(x) = f(x \oplus r)$ is monotone, where $\oplus$ denotes the bit-wise XOR). Both problems of testing *monotonicity* and *unateness* were first introduced in [4]. The goal is to design an algorithm that decides whether an unknown $f : \{0,1\}^n \to \{0,1\}$ has the property being tested or is far from having the property (see Section II for the formal definition) with as few queries as possible. After a sequence of developments from the past few years [5], [6], [7], [8], [9], [10], [2], the query complexity of non-adaptive algorithms for monotonicity has been pinned down at $\widetilde{\Theta}(n^{1/2})$; for adaptive monotonicity testing, there remains a gap between $\widetilde{O}(n^{1/2})$ and $\widetilde{\Omega}(n^{1/3})$. The query complexity of testing unateness, however, is less well-understood.

The seminal work of [4] gave an $O(n^{3/2}/\epsilon)$-query algorithm for testing unateness of Boolean functions. It proceeds by sampling $O(n^{3/2}/\epsilon)$ *edges*[1] of $\{0,1\}^n$ uniformly at random and rejects only when it sees a so-called *edge violation* — a pair of edges (x, y) and (x', y') in the same direction such that one is monotone while the other is anti-monotone. By definition the existence of an edge violation ensures that the unknown function is not unate and thus, the algorithm has one-sided error (i.e., it always accepts a unate function); we refer to such algorithms as *one-sided*. This algorithm is also non-adaptive (i.e., all queries can be made at once). For the correctness, [4] showed that after sampling $O(n^{3/2}/\epsilon)$ edges an edge violation is found with high probability when f is ϵ-far from unate.

Recently [11] obtained the first improvement to the upper bound of [4] with an $O(n \log n/\epsilon)$-query adaptive, one-sided algorithm. Later, [12] generalized the algorithm to work for real-valued functions over the n-dimensional hypergrid, i.e., $f\colon [m]^n \to \mathbb{R}$. The current best upper bounds for testing the unateness of Boolean functions are $O((n/\epsilon)\log(n/\epsilon))$ for non-adaptive algorithms [13], [1], and $O(n/\epsilon)$ for adaptive algorithms [1] (with a logarithmic advantage). Both algorithms work for real-valued functions and are shown to be optimal for real-valued functions in [1].

On the lower bound side, [12] was the first to give a lower bound on testing unateness by showing that a non-adaptive algorithm with one-sided error must make $\Omega(\sqrt{n}/\epsilon)$ queries. Then [2] showed that unateness testing of Boolean functions requires $\widetilde{\Omega}(n^{2/3})$ queries for adaptive algorithms with two-sided error, which implies a polynomial gap between testing monotonicity and unateness for Boolean functions.[2] For non-adaptive algorithms with one-sided error, [2] and [3] showed that $\widetilde{\Omega}(n)$ queries are necessary (for some constant $\epsilon > 0$), which implies the algorithm of [13], [1] is optimal among non-adaptive, one-sided algorithms for Boolean functions.

Our Contribution. Generally, the power of adaptivity in property testing of Boolean functions is not yet well understood. Taking the examples of monotonicity and unateness, the current best algorithms are both non-adaptive[3] (ignoring polylogarithmic factors); polynomial gaps remain between known bounds for the complexity of adaptive algorithms.

[1]A pair of points (x, y) in $\{0,1\}^n$ is an edge in the Boolean hypercube if $x_i \neq y_i$ at exactly one coordinate $i \in [n]$. We will refer to i as the direction of (x, y). An edge (x, y) along direction i is *bi-chromatic* (in f) if $f(x) \neq f(y)$; it is *monotone* if it is bi-chromatic and has $f(x) = x_i$; it is *anti-monotone* if it is bi-chromatic but not monotone.

[2]The conference version of the paper included a weaker lower bound of $\widetilde{\Omega}(\sqrt{n})$ for testing unateness. Recently the authors improved it to $\widetilde{\Omega}(n^{2/3})$ and have updated its full version available as `arXiv:1702.06997`.

[3]For real-valued functions, [1] showed that adaptivity helps by a logarithmic factor.

0272-5428/17 $31.00 © 2017 IEEE
DOI 10.1109/FOCS.2017.85

	Adaptive	Non-adaptive
Upper bounds	$\widetilde{O}(n^{3/4})$ (this work)	$O(n)$ [1]
Lower bounds	$\widetilde{\Omega}(n^{2/3})$ [2]	$\widetilde{\Omega}(n)$ (one-sided) [2], [3]

Figure 1. Current knowledge on upper and lower bounds for testing unateness. We consider the regime where $\epsilon = \Theta(1)$.

The main result of this work is an adaptive and one-sided algorithm with $\widetilde{O}(n^{3/4}/\epsilon^2)$ queries for the unateness testing of Boolean functions.

Theorem 1. *There is an $\widetilde{O}(n^{3/4}/\epsilon^2)$-query [4], adaptive algorithm with the following property: Given an $\epsilon > 0$ and query access to an unknown $f\colon \{0,1\}^n \to \{0,1\}$, it always returns "unate" if f is unate and returns "non-unate" with probability at least $2/3$ if f is ϵ-far from unate.*

Compared to the $\widetilde{\Omega}(n)$ lower bound for non-adaptive and one-sided algorithms [2], Theorem 1 implies that adaptivity helps by a polynomial factor for one-sided algorithms. Additionally, compared to the lower bound of $\Omega(n/\epsilon)$ for unateness testing of real-valued functions over $\{0,1\}^n$ [1], our result shows that Boolean functions are polynomially easier to test than real-valued functions. The current known upper and lower bounds for testing unateness of Boolean functions with $\epsilon = \Theta(1)$ are summarized in the Figure 1.

Our new algorithm is heavily inspired by the work of [7] where they obtained a directed analogue of an isoperimetric inequality of Talagrand [14] and employed it to reveal strong connections between the structure of anti-monotone edges of a Boolean function and its distance to monotonicity. In particular, their new inequality implies that when f is far from monotone, there must exist a highly regular bipartite graph of certain size that consists of only anti-monotone edges of f. The analysis of our algorithm relies on this implication. (See more discussion later in Section I-A.)

A recent work of [15] introduced the notion of "*rounds*" of adaptivity to quantify the degree of adaptivity used by a property testing algorithm. We notice that our algorithm can be implemented using only two rounds of adaptivity.

A. Binary search versus adaptive edge search

We give some high-level ideas behind our main algorithm. First, it outputs "non-unate" only when an edge violation is found and thus, it is one-sided and our analysis focuses on showing that, given a function that is ϵ-far from unate, the algorithm finds an edge violation with high probability.

Recall that an edge violation occurs when a pair of bi-chromatic edges *collide*, i.e., they are in the same direction i but one is monotone and the other is anti-monotone. Thus, an algorithm may proceed by designing a subroutine for finding bi-chromatic edges and invoking it multiple times in hopes of finding a collision. A subroutine for finding bi-chromatic edges that has been widely used in Boolean function property testing (e.g., [16], [9], [11]) is *binary search* (see an illustration in Figure 2):

1) Find $x, y \in \{0,1\}^n$ with $f(x) \neq f(y)$, and let $S = \{i \in [n] : x_i \neq y_i\}$.
2) Pick a subset $S' \subset S$ of size $|S|/2$, let $z = x^{(S')}$,[5] and query $f(z)$.
3) If $f(z) = f(x)$, let $x \leftarrow z$; if $f(z) = f(y)$, let $y \leftarrow z$. Repeat until (x, y) is an edge.

Clearly, the above procedure, when initiated with x, y: $f(x) \neq f(y)$, will always return a bi-chromatic edge along some direction $i \in S$ with $O(\log n)$ many queries. One can choose to *randomize* the subroutine by drawing x and y uniformly at random at the beginning and then drawing the subset S' uniformly at random from S in each round. Given an f, the binary search subroutine naturally induces a distribution over bi-chromatic edges of f. A high-level question is: Can one analyze this distribution for functions f that are ϵ-far from unate? Can this strategy yield better algorithms for finding an edge violation?

While we do not analyze the binary search subroutine as described above in this paper, we introduce a new kind of edge search strategy, which we call *adaptive edge search* and denote by AE-SEARCH. It is a crucial component of our algorithm and allows for a relatively straightforward analysis. It takes two inputs, a point $x \in \{0,1\}^n$ and a nonempty set $S \subseteq [n]$,[6] and aims to find a bi-chromatic edge $(x, x^{(i)})$ for some $i \in S$ (using $O(\log n)$ queries only). The subroutine AE-SEARCH proceeds as follows:

1) Sample $L = O(\log n)$ subsets $T_1, \ldots, T_L \subset S$ of size $|S|/2$ uniformly, and query each $f(x^{(T_\ell)})$.
2) Consider all T_ℓ's with $f(x^{(T_\ell)}) \neq f(x)$. If the intersection of such T_ℓ's consists of exactly one index $i \in S$, query $f(x^{(i)})$ and output i if $f(x^{(i)}) \neq f(x)$ (meaning that a bi-chromatic edge $(x, x^{(i)})$ along direction i has been found); otherwise return "fail."

[4]See (4) for the hidden polylogarithmic factor; we have made no effort to optimize the polynomial dependence on $\log n$ and $\log(1/\epsilon)$.

[5]We use $x^{(S')} \in \{0,1\}^n$ to denote the point obtained from x by flipping its coordinates in S'; we also write $x^{(i)}$ for $x^{(\{i\})}$.

[6]It is not important for the moment but later we will always choose the size of S to be smaller than $\sqrt{n}$.

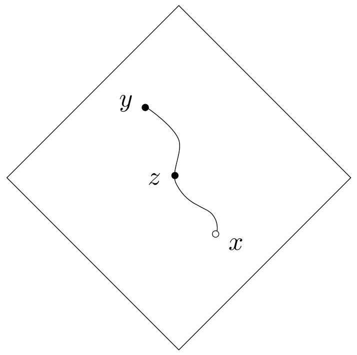

Figure 2. Pictorial representation of one step of the binary search strategy for finding an bi-chromatic edge. The hypercube $\{0,1\}^n$ is represented as the diamond. Points x and y are given with $f(x) = 0$ and $f(y) = 1$, and a particular path represents flipping variables in S one at a time. Finally, $z = x^{(S')}$ corresponds to picking some z between x and y; in this case, $f(z) = 1$, so y would be updated to z.

See Figure 3 for a pictorial representation of AE-SEARCH. While AE-SEARCH does not always returns a bi-chromatic edge (unlike the binary search), its behavior is much easier to analyze. Informally, when $(x, x^{(i)})$ is a bi-chromatic edge and $i \in S$ (otherwise AE-SEARCH can never return i), we show that AE-SEARCH(x, S) returns i with high probability if (1) *most* $T \subset S$ of size $|S|/2$ with $i \notin T$ have $f(x^{(T)}) = f(x)$, and (2) *most* $T \subset S$ of size $|S|/2$ with $i \in T$ have $f(x^{(T)}) = f(x^{(i)}) \neq f(x)$. (See Figure 3.)

With the adaptive edge search in hand, the proof of Theorem 1 proceeds in two steps. For the first step, we show that when f is far from unate, there must be "many" bi-chromatic edges $(x, x^{(i)})$ such that running AE-SEARCH on x paired with a random set $S \subset [n]$ *containing* i would lead to the discovery of $(x, x^{(i)})$ with high probability. There are a lot of technical details hidden in the word "many": (i) subsets $S \subset [n]$ of different sizes contribute differently (intuitively, the larger S is, it is more likely for S to contain i when S is drawn from $[n]$ uniformly at random); (ii) we need to balance the contribution from monotone and anti-monotone edges in the same direction by taking their minimum. Intuitively, it will not help us find an edge violation if AE-SEARCH works well over many bi-chromatic edges along a direction i, but all these edges turn out to be monotone. Following the high-level discussion above, we formally introduce the notion of SCORE_i^+ and SCORE_i^- for a Boolean function in Section IV (to measure the performance of AE-SEARCH) and show that

$$\sum_{i\in[n]} \min\left\{\text{SCORE}_i^+, \text{SCORE}_i^-\right\} = \widetilde{\Omega}(\epsilon^2), \tag{1}$$

when f is ϵ-far from unate. The proof of (1) is omitted here and can be found in the full version of this paper. It heavily relies on the directed isoperimetric inequality of [7] and its combinatorial implications for functions far from monotone.

In the second step of the proof, we present an algorithm that keeps calling AE-SEARCH (strategically) and show that it finds an edge violation with high probability, given (1). At a high level, it starts by sampling an $S \subset [n]$ of certain size and a sequence of M points $\{x_i\}$ from $\{0,1\}^n$. Then it runs AE-SEARCH(x_i, S) for each i and keeps the directions of monotone edges found in set A. Next it samples M subsets $T_i \subseteq S$ of certain size and M points $\{y_i\}$ and use them to run AE-SEARCH(y_i, T_i) for each i. Similarly, it keeps the directions of anti-monotone edges found in B. Finally, the algorithm outputs "non-unate" if $A \cap B \neq \emptyset$, i.e., an edge violation is found; otherwise, it outputs "unate".

The tricky part is the choices of sizes of sets S and T_i as well as the parameter M. For technical reasons our algorithm is split into two cases, depending on how the $\widetilde{\Omega}(\epsilon^2)$ in (1) is achieved, e.g., what scale of $\min\{\text{SCORE}_i^-, \text{SCORE}_i^+\}$ contributes the most in the sum. The parameters are chosen differently in the two cases (case 2 needs one more parameter K) and their proofs use slightly different techniques.

Organization. We introduce the AE-SEARCH subroutine in Section III. Next we introduce the notion of scores and state (1) in Lemma 2 in Section IV. We present the algorithm and its analysis in Section V, assuming Lemma 2. The proof of Lemma 2 can be found in the full version of the paper.

II. PRELIMINARIES

We reserve bold font letters such as $\mathbf{T}$ and $\mathbf{x}$ for random variables. Given $n \geq 1$, we write $[n]$ to denote $\{1, \ldots, n\}$. Given a point x in the Boolean hypercube $\{0,1\}^n$ and $S \subset [n]$, we let $x^{(S)}$ denote the string obtained from x by flipping each entry x_i with $i \in S$. When $S = \{i\}$ is a singleton, we write $x^{(i)}$ instead of $x^{(\{i\})}$ for convenience. Given x and y in $\{0,1\}^n$, $x \oplus y \in \{0,1\}^n$ denotes their bit-wise XOR.

We define the *distance* between two Boolean functions f and $g\colon \{0,1\}^n \to \{0,1\}$ using the uniform distribution:

$$\text{dist}(f, g) := \Pr_{\mathbf{x}\sim\{0,1\}^n}\big[f(\mathbf{x}) \neq g(\mathbf{x})\big].$$

The distance of a function f to unateness is then defined as the minimum value of $\text{dist}(f, g)$ over all unate functions g. We say f is *ϵ-far from unate* if its distance to unateness is at least ϵ, or equivalently, $\text{dist}(f, g) \geq \epsilon$ for all unate g.

We say that an algorithm tests the unateness of Boolean functions if, given ϵ and query access to a Boolean function f, (1) it ouputs "unate" with probability at least $2/3$ when f is unate; and (2) it outputs "non-unate" with probability at least $2/3$ when f is ϵ-far from unate. We say the algorithm is one-sided if it always outputs "unate" when f is unate.

Recall that an *edge violation* of unateness for f consists of two bi-chromatic edges along the same direction, one being monotone and the other being anti-monotone. All algorithms discussed in this paper output "non-unate" only when an edge violation is found among the queries they made. We commonly refer to edge violations simply as violations.

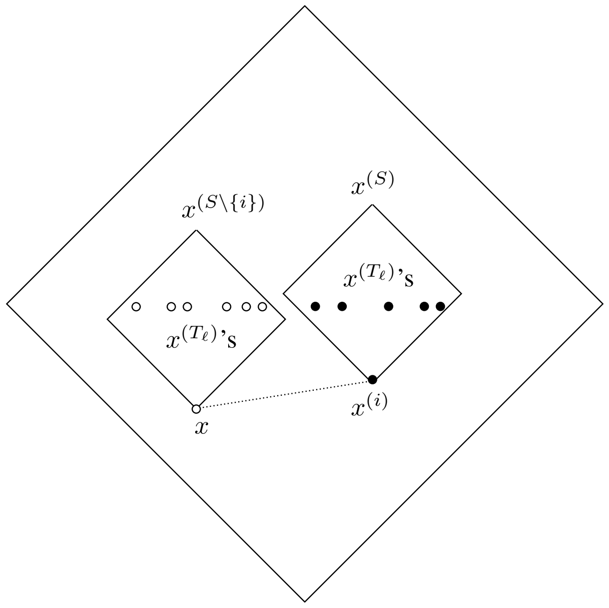

Figure 3. Pictorial representation of the adaptive edge search subroutine, AE-SEARCH(x,S), for finding a bi-chromatic edge. We consider the case when $(x,x^{(i)})$ is a bi-chromatic edge in the direction i with $f(x)=0$ and $i\in S$. The two sub-cubes in the picture above correspond to points that agree with x outside of $S\setminus\{i\}$, and points that agree with $x^{(i)}$ outside of $S\setminus\{i\}$, respectively. The points $x^{(T_\ell)}$ sampled in AE-SEARCH(x,S) lie in one of the sub-cubes according to whether $i\in T_\ell$ or not. Under certain conditions one can show that with high probability, all sets T_ℓ's satisfying $f(x^{(T_\ell)})=1$ lie in the right sub-cube and furthermore, their intersection is exactly $\{i\}$. In this case, AE-SEARCH(x,S) returns i.

The *total influence* I_f of a Boolean function f is the number of bi-chromatic edges of f divided by 2^n. We combine a lemma from [7] and a unateness testing algorithm of [1] to find an edge violation in a function of *high* total influence using $\widetilde{O}(\sqrt{n})$ queries only. The proof is omitted, and can be found in the full version of the paper.

Lemma 1. *There is an $\widetilde{O}(\sqrt{n})$-query and non-adaptive algorithm that, given any function $f\colon\{0,1\}^n\to\{0,1\}$ with $I_f>6\sqrt{n}$, finds an edge violation of f to unateness with probability at least $2/3$.*

Given Lemma 1, it suffices for us to give an $\widetilde{O}(n^{3/4}/\epsilon^2)$-query algorithm that can find a violation in any function that is ϵ-far from unate and satisfies $I_f\le 6\sqrt{n}$.

III. Adaptive edge search

In this section, we introduce a subroutine called *adaptive edge search* (i.e., AE-SEARCH) which will be heavily used in our main algorithm for testing unateness. We present the subroutine in Figure 4. It has query access to a Boolean function $f\colon\{0,1\}^n\to\{0,1\}$ and takes two inputs: $x\in\{0,1\}^n$ is a point in the hypercube and $S\subseteq[n]$ is a nonempty set of even size.

Subroutine AE-SEARCH(x,S)
Input: Query access to $f\colon\{0,1\}^n\to\{0,1\}$, a point $x\in\{0,1\}^n$, and a nonempty set $S\subseteq[n]$ of even size.
Output: Either an $i\in S$ with $f(x^{(i)})\neq f(x)$, or "fail."

1) Query $f(x)$ and set $b\leftarrow f(x)$.
2) If $|S|=2$, pick one coordinate $\boldsymbol{i}\in S$ uniformly at random. Query $f(x^{(\boldsymbol{i})})$ and return $\boldsymbol{i}$ if $f(x^{(\boldsymbol{i})})\neq b$; otherwise return "fail."
3) Sample $L=\lceil 4\log n\rceil$ subsets $\mathbf{T}_1,\ldots,\mathbf{T}_L\subset S$ of size $|S|/2$ uniformly at random. Query $f(x^{(\mathbf{T}_\ell)})$ and set the output to be $\mathbf{b}_\ell$ for each ℓ. Let $\mathbf{C}\subset S$ where
$$\mathbf{C}=\bigcap_{\ell\in[L]:\ \mathbf{b}_\ell\neq b}\mathbf{T}_\ell$$
($\mathbf{C}=\emptyset$ by default if $\mathbf{b}_\ell=b$ for all ℓ). If $\mathbf{C}=\{\boldsymbol{i}\}$ for some index $\boldsymbol{i}$, query $f(x^{(\boldsymbol{i})})$ and return $\boldsymbol{i}$ if $f(x^{(\boldsymbol{i})})\neq b$; otherwise return "fail."

Figure 4. Description of the adaptive edge search subroutine.

The goal of AE-SEARCH(x,S) is to find an index $i\in S$ such that $(x,x^{(i)})$ is a bi-chromatic edge in f. It returns an index $i\in S$ if it finds one (note that AE-SEARCH always checks and makes sure that $(x,x^{(i)})$ is bi-chromatic before it outputs i), or returns "fail" if it fails to find one (which does not necessarily mean that none of the edges $(x,x^{(i)})$, $i\in S$, are bi-chromatic). While a naive search would consider each $i\in S$ and query each $f(x^{(i)})$, as well as $f(x)$, incurring a cost of $|S|+1$ queries that can be expensive when S is large, AE-SEARCH(x,S) only uses $L+2=O(\log n)$ queries, as we set $L=\lceil 4\log n\rceil$ in Figure 4. We record the following simple observation that follows easily from the description of AE-SEARCH.

Fact 1. AE-SEARCH(x,S) *makes $O(\log n)$ queries and returns either an index i or "fail." Whenever it returns an i, we have $i\in S$ and $(x,x^{(i)})$ is a bi-chromatic edge in f.*

We analyze the performance of AE-SEARCH in detail in the full version (as part of the proof of Lemma 2 that we omit). Informally, we show that under the assumption that f is far from unate, AE-SEARCH(x,S) succeeds in finding a bi-chromatic edge $(x,x^{(i)})$ for some $i\in S$ for "many" input pairs (x,S) with high probability. This is summarized using the notion of scores (see the next section) in Lemma 2.

IV. Scores

In this section, we use AE-SEARCH to introduce the notion of *scores* for monotone and anti-monotone edges of a Boolean function f. We start with some notation.

Consider a fixed Boolean function $f\colon\{0,1\}^n\to\{0,1\}$. For each $i\in[n]$, let E_i^+ denote the set of monotone edges in direction i and E_i^- denote the set of anti-monotone edges in direction i. Let

$$\Lambda = \left\lfloor \log_2 \left(\frac{\sqrt{n}}{\log n} \right) \right\rfloor = \Theta(\log n)$$

be a parameter which will be used in the rest of the paper. Given $i \in [n]$ and $j \in [\Lambda]$, we let

$$\mathcal{P}_{i,j} = \left\{ S \subset [n] \setminus \{i\} : |S| = 2^j - 1 \right\}.$$

We need the following definitions:

Definition 1 (Good pairs). *Let $(x, x^{(i)})$ be a monotone edge in E_i^+ for some $i \in [n]$ and let S be a set in $\mathcal{P}_{i,j}$ for some $j \in [\Lambda]$. We say (x, S) is a* good pair for E_i^+ *if* AE-SEARCH$(x, S \cup \{i\})$ *returns i with probability at least* $1/2$ *(i.e., running the adaptive edge search subroutine over x and $S \cup \{i\}$ would help us discover the monotone edge $(x, x^{(i)})$ in E_i^+ with probability at least* $1/2$*).*

By definition (x, S) can be a good pair for E_i^+ only if the edge $(x, x^{(i)})$ is monotone. On the other hand, if $(x, x^{(i)})$ is monotone then (x, S) is always a good pair for all $S \in \mathcal{P}_{i,1}$. This simply follows from the fact that, since $|S \cup \{i\}| = 2$, AE-SEARCH$(x, S \cup \{i\})$ will pick i with probability $1/2$ on line 2 and find the monotone edge $(x, x^{(i)})$.

Next we use good pairs to define *strong* points.

Definition 2 (Strong points). *A point $x \in \{0,1\}^n$ with $(x, x^{(i)}) \in E_i^+$ is said to be j-*strong *(or a j-*strong point*) for E_i^+, for some $j \in [\Lambda]$, if (x, S) is a good pair for E_i^+ for at least $3/4$ of $S \in \mathcal{P}_{i,j}$.*

Consider an x that is j-strong for E_i^+. If we sample a set $\mathbf{S}$ from $\mathcal{P}_{i,j}$ uniformly and run AE-SEARCH$(x, \mathbf{S} \cup \{i\})$, we will discover $(x, x^{(i)}) \in E_i^+$ with probability $(3/4)(1/2) = 3/8$. Note that if $(x, x^{(i)})$ is monotone, then x is always 1-strong. We also extend both definitions of *good pairs* and *strong points* to E_i^-, so we may consider a good pair (x, S) for E_i^- as well as a point x which is j-strong for E_i^-, when $(x, x^{(i)}) \in E_i^-$ is an anti-monotone edge.

For each $i \in [n]$ and $j \in [\Lambda]$, let $\textsc{Score}_{i,j}^+$ be the fraction of points that are j-strong for E_i^+:

$$\textsc{Score}_{i,j}^+ = \frac{\text{number of } j\text{-strong points for } E_i^+}{2^n} \in [0, 1].$$

Intuitively, the higher $\textsc{Score}_{i,j}^+$ is, it becomes easier to find a monotone edge in direction i using AE-SEARCH with 2^j-sized sets ($|S \cup \{i\}| = 2^j$) without using too many queries.

Finally we define $\textsc{Score}_i^+$ for each $i \in [n]$ as (recall that we have $2^j \le \sqrt{n}/\log n$ by the choice of Λ)

$$\textsc{Score}_i^+ = \max_{j \in [\Lambda]} \left\{ \textsc{Score}_{i,j}^+ \cdot \frac{2^j}{\sqrt{n}} \right\} \in [0, 1]. \quad (2)$$

Note that $\textsc{Score}_{i,j}^+$'s are adjusted in (2) by weights $2^j/\sqrt{n}$ before taking the maximum. Roughly speaking, this is done to reflect the fact that with the same $\textsc{Score}_{i,j}^+$, the larger j is, the easier it becomes to find an edge in E_i^+ using sets of size 2^j in AE-SEARCH. Consider a point $x \in \{0,1\}^n$ that is j-strong for E_i^+. As noted earlier, if an algorithm draws $\mathbf{S} \sim \mathcal{P}_{i,j}$ uniformly and runs AE-SEARCH$(x, \mathbf{S} \cup \{i\})$, it will discover $(x, x^{(i)})$ with probability at least $3/8$. However, the situation we will encounter later is that the algorithm only knows j but not i. So from the algorithm's perspective, the point x has a bi-chromatic edge $(x, x^{(i)})$, for some i, but it does now know which i it is. A natural attempt is then to run AE-SEARCH$(x, \mathbf{S}')$ with an $\mathbf{S}'$ of size 2^j sampled from $[n]$ uniformly at random, with the hope that 1) $\mathbf{S}'$ contains i and 2) $\mathbf{S}' \setminus \{i\} \in \mathcal{P}_{i,j}$ forms a good pair with x. As j increases, it becomes easier for $\mathbf{S}' \sim [n]$ to contain the unknown index i. This is the reason why the weight grows as j grows.

We also extend the scores to $\textsc{Score}_{i,j}^-, \textsc{Score}_i^-$ for E_i^-.

A. Plan for the proof of Theorem 1

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function that is ϵ-far from unate. Our goal is to present an $\widetilde{O}(n^{3/4}/\epsilon^2)$-query algorithm which finds an edge violation of f with probability at least $2/3$. By Lemma 1, we may assume without loss of generality that f in addition satisfies $I_f \le 6\sqrt{n}$.

We rely on the following technical lemma for the scores of f. Its proof can be found in the full version of the paper.

Lemma 2. *If $f : \{0,1\}^n \to \{0,1\}$ is ϵ-far from unate and satisfies $I_f \le 6\sqrt{n}$, then we have*

$$\sum_{i=1}^{n} \min\left\{ \textsc{Score}_i^+, \textsc{Score}_i^- \right\} \ge \Omega\left(\frac{\epsilon^2}{\log^8 n} \right). \quad (3)$$

We present our $\widetilde{O}(n^{3/4}/\epsilon^2)$-query (adaptive) algorithm in the next section and show that, given any f that satisfies (3), it finds an edge violation of f with probability at least $2/3$.

V. Main algorithm and its analysis

We describe our main algorithm and show that, given any function $f : \{0,1\}^n \to \{0,1\}$ that satisfies (3), it uses

$$O\left(\frac{n^{3/4}}{\epsilon^2} \cdot \log^{16} n \cdot \log^2(n/\epsilon) \right) = \widetilde{O}(n^{3/4}/\epsilon^2) \quad (4)$$

queries and finds a violation with probability at least $2/3$.

A. Preparation: Bucketing scores

We start with some preparation for the algorithm. First we use standard bucketing techniques to make (3) easier to use (while only losing a polylogarithmic factor in the sum).

Recall that

$$\textsc{Score}_i^+ = \max_{j \in [\Lambda]} \left\{ \textsc{Score}_{i,j}^+ \cdot \frac{2^j}{\sqrt{n}} \right\} \quad \text{and}$$

$$\textsc{Score}_i^- = \max_{j \in [\Lambda]} \left\{ \textsc{Score}_{i,j}^- \cdot \frac{2^j}{\sqrt{n}} \right\}.$$

We will say that the ith direction is of *type*-(t, r), for some $t, r \in [\Lambda]$, if we have

$$\textsc{Score}_i^+ = \textsc{Score}_{i,t}^+ \cdot \frac{2^t}{\sqrt{n}}, \text{ and}$$

$$\textsc{Score}_i^- = \textsc{Score}_{i,r}^- \cdot \frac{2^r}{\sqrt{n}}.$$

Since $\Lambda = O(\log n)$, there are only $O(\log^2 n)$ types. By (3) we know that there is a pair (t, r) such that

$$\sum_{i:\text{type-}(t,\,r)} \min\left\{\textsc{Score}_i^+, \textsc{Score}_i^-\right\} = \Omega\left(\frac{\epsilon^2}{\log^{10} n}\right). \quad (5)$$

In the rest of the section, we fix such a type (t, r). (Looking ahead, we may assume that our algorithm knows (t, r) as it can afford to try all $O(\log^2 n)$ possible pairs of (t, r).)

Let $I^* \subseteq [n]$ be the set of all type-(t, r) directions. We next divide I^* into $\lceil 2\log(n/\epsilon)\rceil$ buckets according to

$$\min\left\{\textsc{Score}_i^+, \textsc{Score}_i^-\right\}.$$

An $i \in I^*$ lies in the k-th bucket if it satisfies

$$\frac{1}{2^k} \le \min\left\{\textsc{Score}_i^+, \textsc{Score}_i^-\right\} \le \frac{1}{2^{k-1}}.$$

Note that some $i \in I^*$ may not lie in any bucket when

$$\min\left\{\textsc{Score}_i^+, \textsc{Score}_i^-\right\} \le \epsilon^2/n^2;$$

however, all such $i \in I^*$ in total contribute at most $O(\epsilon^2/n)$ to the LHS of (5), which is negligible compared to its RHS. Since k has $\lceil 2\log(n/\epsilon)\rceil = O(\log(n/\epsilon))$ possibilities, there exists an h such that

$$\sum_{i\in I^*:\text{ bucket } h} \min\left\{\textsc{Score}_i^+, \textsc{Score}_i^-\right\}$$

is at least

$$\Omega\left(\frac{\epsilon^2}{\log^{10} n \cdot \log(n/\epsilon)}\right). \quad (6)$$

We fix such an h in the rest of the section (due to the same reason we may assume that the algorithm knows h), and let $I \subseteq I^*$ be the indices of I^* in bucket h.

To simplify the notation, we let $H = 2^h$ and

$$\widetilde{\epsilon}^2 = \frac{c\epsilon^2}{\log^{10} n \cdot \log(n/\epsilon)},$$

where we use $\widetilde{\epsilon}$ to hide the polylogarithmic factor in ϵ and n, and c is some constant which ensures

$$\sum_{i\in I} \min\left\{\textsc{Score}_i^+, \textsc{Score}_i^-\right\} \ge \widetilde{\epsilon}^2.$$

Given that $H = 2^h$, we have

$$1/H \le \min\left\{\textsc{Score}_i^+, \textsc{Score}_i^-\right\} \le 2/H$$

for each $i \in I$, and $|I| \cdot (2/H) \ge \widetilde{\epsilon}^2$ from (6). This implies

$$H \le 2|I|/\widetilde{\epsilon}^2 = O(n/\widetilde{\epsilon}^2)$$

since $|I| \le n$. Moreover, using

$$1/H \le \textsc{Score}_i^+ = \textsc{Score}_{i,t}^+ \cdot (2^t/\sqrt{n}) \le 2^t/\sqrt{n},$$

we have $H2^t \ge \sqrt{n}$ and similarly, $H2^r \ge \sqrt{n}$.

We summarize the discussion with the following lemma.

Lemma 3. *Suppose that f satisfies* (3)*. Then there exist $t, r \in [\Lambda]$, $H = O(n/\widetilde{\epsilon}^2)$ as a power of 2 with $H2^t, H2^r \ge \sqrt{n}$, and a nonempty $I \subseteq [n]$ of size $|I| \ge H\widetilde{\epsilon}^2/2$ such that every $i \in I$ satisfies*

$$\min\left\{\textsc{Score}_i^+, \textsc{Score}_i^-\right\}$$
$$= \min\left\{\textsc{Score}_{i,t}^+ \cdot \frac{2^t}{\sqrt{n}}, \textsc{Score}_{i,r}^- \cdot \frac{2^r}{\sqrt{n}}\right\} \in \left[\frac{1}{H}, \frac{2}{H}\right].$$

B. Preparation: Informative sets

We introduce more notation and then state Lemma 4 that will be heavily used in the analysis of the main algorithm. We defer the proof of Lemma 4 to Subsection V-F. Below t, r and H are considered as fixed parameters, and I is a set of indices that satisfies the condition of Lemma 3. We further assume that $t \ge r$; all our discussion below holds when $t < r$ by switching the roles of t and r (and E_i^+ and E_i^-). We start with some useful notation related to good pairs.

Recall (x, S) is a good pair for E_i^+ (or E_i^-) if $(x, x^{(i)})$ is a monotone edge (or anti-monotone edge, respectively) and AE-$\textsc{Search}(x, S \cup \{i\})$ returns i with probability at least $1/2$. Given an $S \in \mathcal{P}_{i,j}$, let

$$\textsc{Good-Set}_i^+(S) = \left\{x : (x, S) \text{ is a good pair for } E_i^+\right\},$$

$$\textsc{Good-Set}_i^-(S) = \left\{x : (x, S) \text{ is a good pair for } E_i^-\right\}.$$

We also use

$$\textsc{Good-Frac}_i^+(S) = \frac{|\textsc{Good-Set}_i^+(S)|}{2^n} \quad \text{and}$$

$$\textsc{Good-Frac}_i^-(S) = \frac{|\textsc{Good-Set}_i^-(S)|}{2^n}$$

to denote the fractions.

Recall $x \in \{0, 1\}^n$ is j-strong for E_i^+ (or E_i^-) if (x, S) is a good pair for E_i^+ (or E_i^-) for at least 3/4 of sets $S \in \mathcal{P}_{i,j}$. Given an $i \in I$, we let $\textsc{Strong}_i^+$ denote the set of t-strong points for E_i^+, and let $\textsc{Strong}_i^-$ denote the set of r-strong points for E_i^-. By Lemma 3, we have

$$\textsc{Score}_{i,t}^+ = \frac{|\textsc{Strong}_i^+|}{2^n} \ge \frac{\sqrt{n}}{H \cdot 2^t} \quad \text{and} \quad (7)$$

$$\textsc{Score}_{i,r}^- = \frac{|\textsc{Strong}_i^-|}{2^n} \ge \frac{\sqrt{n}}{H \cdot 2^r}.$$

We define the following two parameters α and β, which will be very important for the algorithm:

$$\alpha = \frac{|I| \cdot 2^t}{n} \quad \text{and} \quad \beta = \frac{|I| \cdot 2^r}{n}.$$

They measure the expectations of $|I \cap \mathbf{S}|$ and $|I \cap \mathbf{T}|$, when $\mathbf{S}$ is a random subset of $[n]$ of size 2^t and $\mathbf{T}$ is a subset of $[n]$ of size 2^r, both drawn uniformly at random.

Finally we introduce the notion of *informative sets*.

Definition 3 (Informative Sets)**.** *We say a set $S \in \mathcal{P}_{i,t}$ for some $i \in I$ is* informative *for the ith coordinate if both of the following two conditions hold:*

1) $\textsc{Good-Frac}_i^+(S) \geq 0.1 \cdot \widetilde{\epsilon}^2/(\alpha\sqrt{n})$*; and*
2) $\textsc{Good-Frac}_i^-(T) \geq 0.1 \cdot \widetilde{\epsilon}^2/(\beta\sqrt{n})$ *for at least 0.1-fraction of $(2^r - 1)$-sized subsets T of S. We refer to $T \cup \{i\}$ as an i-*revealing *set when T has* $\textsc{Good-Frac}_i^-(T) \geq 0.1 \cdot \widetilde{\epsilon}^2/(\beta\sqrt{n})$.

*Additionally, we say the set $S \cup \{i\}$ is i-*informative *if S is informative for the ith coordinate.*

To gain some intuition, if the algorithm is given $S \in \mathcal{P}_{i,t}$ for some $i \in I$ that is informative for the ith coordinate, it can use S and i to find a violation along i as follows:

1) Sample $O(\alpha\sqrt{n}/\widetilde{\epsilon}^2)$ points $\mathbf{x} \in \{0,1\}^n$ uniformly and run $\textsc{AE-Search}(\mathbf{x}, S \cup \{i\})$.
2) Sample a subset $\mathbf{T} \subseteq S$ of size $2^r - 1$, sample $O(\beta\sqrt{n}/\widetilde{\epsilon}^2)$ points $\mathbf{y} \in \{0,1\}^n$ uniformly at random, and then run $\textsc{AE-Search}(\mathbf{y}, \mathbf{T} \cup \{i\})$.

We get a violation if we find a monotone edge in direction i in step 1 and an anti-monotone edge in direction i in step 2. By Definition 3, this occurs with probability $\Omega(1)$. Now of course the algorithm does not have knowledge of S and i, so we need to incorporate other ideas; however, the intuition is that informative sets can help reveal edge violations of f efficiently using the $\textsc{AE-Search}$ subroutine.

The key will be to show that there are many informative sets for each $i \in I$, which we do in the following lemma using standard averaging arguments, but delay its proof to Section V-F.

Lemma 4. *For each $i \in I$, at least $1/8$ of sets $S \in \mathcal{P}_{i,t}$ are informative for the ith coordinate.*

C. Cases of the main algorithm

We are now ready to describe the main algorithm (which is one-sided and returns "non-unate" *only* when it finds an edge violation of unateness). As mentioned earlier we focus on the case when f satisfies (3) and show that for any such f, the algorithm finds an edge violation with probability at least $2/3$. We assume that the algorithm knows all the parameters r, t and H from Lemma 3 (algorithmically, we just try all possibilities for these parameters, which will incur a factor of $O(\log^2 n \cdot \log(n/\epsilon))$ in the final query complexity). Let $I \subseteq [n]$ be the set promised in Lemma 3 (note that algorithm has no knowledge about I). We also assume that $t \geq r$; if not one can switch the roles of monotone and anti-monotone edges by running the algorithm on $g(x) = f(x \oplus 1^n)$.

Subroutine `Alg-Case-1`, handling the case $\alpha \geq \log^2 n$
Input: Query access to $f\colon \{0,1\}^n \to \{0,1\}$
Output: Either "unate," or two edges that form a violation.

```
Repeat the following $O(1)$ times:
  1) Sample uniformly an $\mathbf{S}$ of size $2^t$ from $[n]$.
  2) Repeat $M$ times:
     – Sample an $\mathbf{x} \in \{0,1\}^n$ uniformly at random
       and run AE-Search$(\mathbf{x}, \mathbf{S})$.
  3) Let $\mathbf{A}$ be the set of $i \in [n]$ such that a monotone
     edge in direction $i$ is found.
  4) Repeat $M$ times:
     – Sample uniformly a subset $\mathbf{T} \subseteq \mathbf{S}$ of size $2^r$
       and $\mathbf{y} \in \{0,1\}^n$, and run AE-Search$(\mathbf{y}, \mathbf{T})$.
  5) Let $\mathbf{B}$ be the set of $i \in [n]$ such that an
     anti-monotone edge in direction $i$ is found.
  6) If $\mathbf{A} \cap \mathbf{B} \neq \emptyset$, output an edge violation found.
If no edge violation is found on line 6, output "unate."
```

Figure 5. Description of `Alg-Case-1` for Case 1 of the algorithm.

The algorithm is divided into two cases: $\alpha \geq \log^2 n$ and $\alpha < \log^2 n$. In each case, we present an algorithm, analyze its query complexity, and show that it finds an edge violation with high probability for this case. Although the algorithm does not know the exact value of α, it can simply run both, which incurs another factor of 2 in the query complexity.

D. Case 1: $\alpha \geq \log^2 n$

In this case, we expect a random set $\mathbf{S}$ of size 2^t to have intersection with (the unknown) I of size at least $\log^2 n$. The algorithm, `Alg-Case-1`, is presented in Figure 5 with the following parameter:

$$M = \left\lceil \frac{\sqrt{\alpha n}}{\widetilde{\epsilon}^2} \cdot \log^3 n \right\rceil.$$

Fact 2 (Query complexity)**.** *The number of queries used by* `Alg-Case-1` *is* (*using* $\alpha \leq \sqrt{n}$)

$$O(1) \cdot 2M \cdot O(\log n) = O\left(\frac{\sqrt{\alpha n} \cdot \log^4 n}{\widetilde{\epsilon}^2}\right)$$
$$= O\left(\frac{n^{3/4} \cdot \log^{14} n \cdot \log(n/\epsilon)}{\epsilon^2}\right).$$

Correctness: Below we show that `Alg-Case-1` finds a violation with high probability. We split the proof into two lemmas. The first one, Lemma 5, shows that each time we sample $\mathbf{S}$ at the beginning of an iteration, a certain condition for $\mathbf{S}$ holds with constant probability. The second, Lemma 6, shows that when $\mathbf{S}$ satisfies this condition, the algorithm can find a violation with high probability in that iteration.

Lemma 5. *Let $\mathbf{S}$ be a subset of size 2^t sampled from $[n]$ uniformly at random, and let $\mathbf{I_S} \subseteq I \cap \mathbf{S}$ be the set of $i \in I \cap \mathbf{S}$ such that $\mathbf{S}$ is i-informative. Then we have*

$$\alpha/10 \le |\mathbf{I_S}| \le 4\alpha$$

with probability $\Omega(1)$.

Proof: Recall that α is the expected size of $I \cap \mathbf{S}$. As a result of $\alpha \ge \log^2 n$, the fraction of $S \subset [n]$ of size 2^t with $|S \cap I| > 4\alpha$ is at most $\exp(-\Omega(\log^2 n))$. Let

$$\mathcal{S} = \big\{ S \subset [n] : |S| = 2^t \text{ and } |S \cap I| \le 4\alpha \big\}.$$

We define a bipartite graph H^*: vertices on the two sides correspond to I and $\mathcal{S}$; (i, S) is an edge if S is i-informative. By Lemma 4, the degree of each $i \in I$ is at least

$$\frac{1}{8}|\mathcal{P}_{i,t}| - \exp\left(-\Omega(\log^2 n)\right)\binom{n}{2^t} \ge \frac{1}{9}|\mathcal{P}_{i,t}| = \frac{1}{9}\binom{n-1}{2^t-1}$$

Let γ denote the fraction of $S \in \mathcal{S}$ (among $\mathcal{S}$) with degree at least $\alpha/10$ in H^*. On the one hand, the number of edges in H^* is at least (counting from the I-side; using $|\mathcal{S}| \le \binom{n}{2^t}$)

$$|I| \cdot \frac{1}{9} \cdot \binom{n-1}{2^t-1} \ge \frac{1}{9} \cdot |I| \cdot \frac{2^t}{n} \cdot |\mathcal{S}| = \frac{1}{9} \cdot \alpha |\mathcal{S}|.$$

On the other hand, the number of edges is at most (counting from the $\mathcal{S}$-side)

$$\gamma|\mathcal{S}| \cdot 4\alpha + (1-\gamma)|\mathcal{S}| \cdot (\alpha/10) = \alpha|\mathcal{S}| \cdot \left(\frac{39\gamma}{40} + \frac{1}{10}\right).$$

As a result, $\gamma = \Omega(1)$. Since $\mathcal{S}$ consists of $(1-o(1))$-fraction of all sets $S \subset [n]$ of size 2^t, the set $\mathbf{S}$ sampled in Step 1 of `Alg-Case-1` lies in $\mathcal{S}$ and has degree between $\alpha/10$ and 4α with probability at least $\Omega(1)$. ∎

Lemma 6. *Suppose `Alg-Case-1` samples a set S, and let $I_S \subseteq I \cap S$ be the set of i such that S is i-informative. If $\alpha/10 \le |I_S| \le 4\alpha$, then `Alg-Case-1` finds an edge violation with probability $1 - o(1)$ in that iteration.*

The proof of the lemma is divided into simple claim. We consider a fixed $S \subset [n]$ of size 2^t such that $\alpha/10 \le |I_S| \le 4\alpha$. We let `Alg-Case-1` run up to Step 3, and let

$$\lambda = |\mathbf{A} \cap I_S|.$$

Claim 1. *After M iterations of Step 2 in `Alg-Case-1`, $\lambda \ge \sqrt{\alpha}$ with probability $1 - o(1)$.*

Proof: Divide the M iterations into $\sqrt{\alpha}$ batches, each of

$$M/\sqrt{\alpha} = \Omega\left(\frac{\sqrt{n}}{\tilde{\epsilon}^2} \cdot \log^3 n\right) \tag{8}$$

rounds of Step 2. For batch ℓ, we let $\mathbf{X}_\ell$ denote the indicator random variable for the event that at the beginning of the ℓth batch, $|\mathbf{A} \cap I_S| < \sqrt{\alpha}$, and the ℓth batch fails to discover a monotone edge along a new direction i in $I_S \setminus \mathbf{A}$. We prove that all $\mathbf{X}_\ell$ are 0 with probability $1 - o(1)$. The lemma then follows, since when all $\mathbf{X}_\ell$ are 0, we have either (1) one of the $\mathbf{X}_\ell$ is 0 because $|\mathbf{A} \cap I_S| \ge \sqrt{\alpha}$ at the beginning of the ℓth batch, in which case we are done, or (2) every $\mathbf{X}_\ell$ is 0 because a new direction is discovered in the ℓth batch, from which we can also conclude that $|\mathbf{A} \cap I_S| \ge \sqrt{\alpha}$ at the end.

Suppose that at the start of the ℓth batch, $|\mathbf{A} \cap I_S| < \sqrt{\alpha}$. Then consider the auxiliary bipartite graph H^*: vertices on the left-hand side consist of all points $x \in \{0,1\}^n$; vertices on the right-hand side consist of indices of I_S; an edge (x, i) is present if x and $S \setminus \{i\}$ forms a good pair for E_i^+.

Notice that every vertex on the right-hand side has degree at least $0.1 \cdot \tilde{\epsilon}^2/(\alpha\sqrt{n}) \cdot 2^n$ and every vertex on the left-hand side has degree at most 2 (because (x, i) is an edge only if AE-SEARCH(x, S) returns $(x, x^{(i)})$ with probability at least $1/2$). Thus, the fraction of points on the left-hand side which are connected to at least one vertex on right-hand side that is currently not in $\mathbf{A}$ is at least

$$\left(|I_S| - \sqrt{\alpha}\right) \cdot \frac{0.1 \cdot \tilde{\epsilon}^2}{\alpha\sqrt{n}} \cdot \frac{1}{2} \ge \frac{|I_S|}{2} \cdot \frac{0.1 \cdot \tilde{\epsilon}^2}{\alpha\sqrt{n}} \cdot \frac{1}{2} = \Omega\left(\frac{\tilde{\epsilon}^2}{\sqrt{n}}\right).$$

By (8), we discover a new index in I_S during the ith batch with probability at least $1 - \exp(-\Omega(\log^3 n))$. We can then apply a union bound over the $\sqrt{\alpha} \le n^{1/4}$ batches. ∎

Assume that at the end of Step 3, we have obtained set A with $\lambda = |A \cap I_S| \ge \sqrt{\alpha}$. We move on to prove that in Step 4, we will discover an anti-monotone edge which, together with a monotone edge from Step 2, forms an edge violation. We divide the proof into two cases. The first two claims correspond to the case when $\beta \ge \alpha \log^2 n/\lambda$, and the next two claims correspond to the case when $\beta < \alpha \log^2 n/\lambda$.

Claim 2. *Suppose $\beta \ge \alpha \log^2 n/\lambda$. Let A be a fixed set after running up to Step 4 of `Alg-Case-1` satisfying $|A \cap I_S| = \lambda \ge \sqrt{\alpha}$. Then with probability at least $\Omega(1)$ over the draw of a 2^r-sized random subset $\mathbf{T}$ of S, the number of indices $i \in A \cap I_S$ such that $\mathbf{T}$ is i-revealing is at least $\beta/(100\sqrt{\alpha})$.*

Proof: First we let $\mathcal{T}$ denote the following set:

$$\mathcal{T} = \big\{ T \subseteq S : |T| = 2^r \text{ and } |T \cap A \cap I_S| \le 4\lambda \cdot \beta/\alpha \big\}.$$

The expectation of $|\mathbf{T} \cap A \cap I_S|$ when $\mathbf{T}$ is a random subset of S of size 2^r is at most $\lambda \cdot \beta/\alpha$. Since $\lambda \cdot \beta/\alpha \ge \log^2 n$ (by assumption), $\mathcal{T}$ consists of all but an $\exp(-\Omega(\log^2 n))$-fraction of subsets of S of size 2^r.

Next, consider a bipartite graph H^*: vertices on the LHS correspond to $i \in A \cap I_S$; vertices on the RHS correspond to $T \in \mathcal{T}$; (i, T) is an edge if T is i-revealing. Since $i \in A \cap I_S$, S is i-informative and thus, the degree of each i is at least

$$0.1\binom{2^t-1}{2^r-1} - \exp\left(-\Omega(\log^2 n)\right)\binom{2^t}{2^r} \ge \frac{1}{20}\binom{2^t-1}{2^r-1}.$$

We show below that many sets $T \in \mathcal{T}$ have degree at least $\lambda\beta/(100\alpha)$. To this end, we let γ be the fraction of $T \in \mathcal{T}$ on the RHS which have degree at least $\lambda\beta/(100\alpha)$ (among all 2^r-subsets of S). Then

$$\gamma\binom{2^t}{2^r}\frac{4\lambda\beta}{\alpha} + (1-\gamma)\binom{2^t}{2^r}\frac{\lambda\beta}{100\alpha} \geq \lambda\frac{1}{20}\binom{2^t-1}{2^r-1}.$$

As $2^t/2^r = \alpha/\beta$, canceling the factors we obtain $\gamma = \Omega(1)$. This shows that at least $\Omega(1)$-fraction of the 2^r-subsets of S have degree at least $\lambda\beta/(100\alpha) \geq \beta/(100\sqrt{\alpha})$. ■

Claim 3. *Suppose that $\beta \geq \alpha\log^2 n/\lambda$ and $|A \cap I_S| = \lambda \geq \sqrt{\alpha}$. After M iterations of Step 4 in* `Alg-Case-1`*, we have $A \cap \mathbf{B} \neq \emptyset$ with probability at least $1 - o(1)$.*

Proof: Note that with probability $\Omega(1)$, we have

$$|\mathbf{T} \cap A \cap I_S| \geq \beta/(100\sqrt{\alpha}).$$

Similar to the proof of Claim 1, we let H^* denote a bipartite graph: vertices on the LHS correspond to points $x \in \{0,1\}^n$; vertices on the RHS correspond to indices $i \in \mathbf{T} \cap A \cap I_S$; (x, i) are connected if $(x, T \setminus \{i\})$ forms a good pair for E_i^-. Note that each i on the RHS has degree at least

$$0.1 \cdot \widetilde{\epsilon}^2/(\beta\sqrt{n}) \cdot 2^n;$$

each x on the LHS has degree at most 2. Hence the fraction of points on the left-hand size which are connected to points on the right-hand side is at least

$$\frac{\beta}{100\sqrt{\alpha}} \cdot \frac{0.1 \cdot \widetilde{\epsilon}^2}{\beta\sqrt{n}} \cdot \frac{1}{2} = \Omega\left(\frac{\widetilde{\epsilon}^2}{\sqrt{\alpha n}}\right).$$

By our choice of M, `Alg-Case-1` finds an edge violation with probability at least $1 - o(1)$. ■

This finishes the case of $\beta \geq \alpha\log^2 n/\lambda$. Next we work on the case when $\beta < \alpha\log^2 n/\lambda$.

Claim 4. *Suppose $\beta < \alpha\log^2 n/\lambda$. Let A be a fixed set after running up to Step 4 of* `Alg-Case-1` *satisfying $|A \cap I_S| = \lambda \geq \sqrt{\alpha}$. Then with probability at least $\Omega(\beta/(\sqrt{\alpha}\log^2 n)$ over the draw of a 2^r-sized random subset $\mathbf{T}$ of S, there is at least one index $i \in A \cap I_S$ such that $\mathbf{T}$ is i-revealing.*

Proof: First we let $\mathcal{T}$ denote the following set:

$$\mathcal{T} = \left\{T \subset S : |T| = 2^r \text{ and } |T \cap A \cap I_S| \leq 4\log^2 n\right\}.$$

As $|A \cap I_S| = \lambda$, the expectation of $|A \cap I_S \cap \mathbf{T}|$ when $\mathbf{T}$ is a random 2^r-subset of S is at most $\lambda\beta/\alpha < \log^2 n$. Thus, $\mathcal{T}$ consists of all but an $\exp(-\Omega(\log^2 n))$-fraction of subsets of S of size 2^r.

We consider a bipartite graph H^*: vertices on its LHS are indices $i \in A \cap I_S$; vertices on its RHS are sets $T \in \mathcal{T}$; (i, T) is an edge if T is i-revealing. Note that since $i \in A \cap I_S$, S is i-informative and thus, the degree of each i is at least

$$0.1\binom{2^t-1}{2^r-1} - \exp\left(-\Omega(\log^2 n)\right)\binom{2^t}{2^r} \geq \frac{1}{20}\binom{2^t-1}{2^r-1}.$$

On the other hand, the degree of each vertex on the RHS is at most $4\log^2 n$. Hence the fraction of vertices on the RHS (among all 2^r-subsets of S) that are not isolated is at least

$$\begin{aligned}|A \cap I_S| \cdot \frac{1}{20}\binom{2^t-1}{2^r-1} \cdot \frac{1}{4\log^2 n} \cdot \frac{1}{\binom{2^t}{2^r}} &\geq \frac{\lambda\beta}{20\alpha \cdot 4\log^2 n} \\ &\geq \Omega\left(\frac{\beta}{\sqrt{\alpha}\log^2 n}\right)\end{aligned}$$

where the second inequality used $\lambda \geq \sqrt{\alpha}$. ■

Claim 5. *Suppose that $\beta < \alpha\log^2 n/\lambda$ and $|A \cap I_S| = \lambda \geq \sqrt{\alpha}$. After M iterations of Step 4 in* `Alg-Case-1`*, we have $A \cap \mathbf{B} \neq \emptyset$ with probability at least $1 - o(1)$.*

Proof: It follows from Claim 4 that, with probability at least $\Omega(\beta/(\sqrt{\alpha}\log^2 n))$, there exists an $i \in A \cap I_S$ such that $i \in \mathbf{T}$ and $\mathbf{T}$ is i-revealing. When such a T is sampled, since T is i-revealing, there exist at least $0.1 \cdot \widetilde{\epsilon}^2/(\beta\sqrt{n}) \cdot 2^n$ many y's for which AE-SEARCH(y, T) returns an anti-monotone edge in direction i with probability at least $1/2$. Thus, with probability at least

$$\Omega\left(\frac{\beta}{\sqrt{\alpha}\log^2 n} \cdot \frac{\widetilde{\epsilon}^2}{\beta\sqrt{n}}\right) = \Omega\left(\frac{\widetilde{\epsilon}^2}{\sqrt{\alpha n} \cdot \log^2 n}\right)$$

over the draw of $\mathbf{T}$, $\mathbf{y}$, and the randomness of AE-SEARCH, we find a violation to unateness. This finishes the proof by our choice of the parameter M. ■

E. Case 2: $\alpha < \log^2 n$

In this case, we expect a random subset $\mathbf{S}$ of size 2^t and a random subset $\mathbf{T}$ of size 2^r (recall that $r \leq t$) to have a relatively small intersection with (the unknown) I. We can actually achieve an $\widetilde{O}(\sqrt{n}/\epsilon^2)$ query complexity in this case. The algorithm, `Alg-Case-2`, is presented in Figure 6 with the following parameters:

$$K = \left\lceil\frac{\log^3 n}{\alpha}\right\rceil \quad \text{and} \quad M = \left\lceil\frac{\alpha\sqrt{n} \cdot \log n}{\widetilde{\epsilon}^2}\right\rceil.$$

Both K and M are $\Omega(\log n)$ using $\alpha < \log^2 n$, $\alpha = |I|2^t/n$, $|I| \geq H\widetilde{\epsilon}^2/2$ and $H2^t \geq \sqrt{n}$ from Lemma 3.

Fact 3 (Query Complexity). *The number of queries used by* `Alg-Case-2` *is*

$$K \cdot (M + M) \cdot O(\log n) = O\left(\frac{\sqrt{n} \cdot \log^5 n}{\widetilde{\epsilon}^2}\right).$$

Subroutine `Alg-Case-2`, handling the case $\alpha < \log^2 n$
Input: Query access to $f\colon \{0,1\}^n \to \{0,1\}$
Output: Either "unate," or two edges that form a violation.

```
Repeat the following $K$ times:
    1) Sample uniformly an $\mathbf{S}$ of size $2^t$ from $[n]$.
    2) Repeat $M$ times:
        – Sample an $\mathbf{x} \in \{0,1\}^n$ uniformly at random
          and run AE-SEARCH$(\mathbf{x}, \mathbf{S})$.
    3) Let $\mathbf{A}$ be the set of $i \in [n]$ such that a monotone
       edge in direction $i$ is found.
    4) Repeat $M$ times:
        – Sample uniformly a $\mathbf{T} \subseteq \mathbf{S}$ of size $2^r$ and a
          $\mathbf{y} \in \{0,1\}^n$, and run AE-SEARCH$(\mathbf{y}, \mathbf{T})$.
    5) Let $\mathbf{B}$ be the set of $i \in [n]$ such that an
       anti-monotone edge in direction $i$ is found.
    6) If $\mathbf{A} \cap \mathbf{B} \neq \emptyset$, return an edge violation found.
If no violation is found on line 6, return "unate."
```

Figure 6. Description of `Alg-Case-2` for Case 2 of the algorithm.

Correctness: Below we show that `Alg-Case-2` finds an edge violation with high probability. We further divide the proof into two lemmas. The first lemma obtains a sufficient condition for finding an edge violation for f, and the second shows that the condition is satisfied with high probability.

Lemma 7. *Suppose* `Alg-Case-2` *starts with a set S that is i-informative for some $i \in I$. Then during this iteration, it finds an edge violation for f along the ith direction with probability at least* $1 - o(1)$.

Proof: Let $S' = S \setminus \{i\} \in \mathcal{P}_{i,t}$. Since S' is informative for the ith coordinate, we have

$$\text{GOOD-FRAC}_i^+(S') \geq 0.1 \cdot \frac{\widetilde{\epsilon}^2}{\alpha\sqrt{n}} \quad \text{and} \tag{9}$$
$$\text{GOOD-FRAC}_i^-(T') \geq 0.1 \cdot \frac{\widetilde{\epsilon}^2}{\beta\sqrt{n}}$$

for at least 0.1-fraction of (2^r-1)-sized $T' \subset S'$. We show below that $i \in \mathbf{A} \cap \mathbf{B}$ at the end of the loop with probability at least $1 - o(1)$.

First by the definition of good pairs, every time an x sampled in Step 2 lies in $\text{GOOD-SET}_i^+(S')$, AE-SEARCH(x, S) outputs the monotone edge $(x, x^{(i)})$ with probability at least $1/2$. Using our choice of M, we have $i \in \mathbf{A}$ at the end of Step 3 in this loop with probability at least $1 - o(1)$.

Next, the number of (2^r-1)-sized subsets T' of S' that satisfy (9) is at least

$$0.1 \cdot \binom{2^t-1}{2^r-1}.$$

As a result, $T' \cup \{i\}$ from such T' consist of at least an

$$\Omega\left(\binom{2^t-1}{2^r-1}\Big/\binom{2^t}{2^r}\right) = \Omega\left(\frac{2^r}{2^t}\right) = \Omega\left(\frac{\beta}{\alpha}\right)$$

fraction of 2^r-subsets of S. When such a $T' \cup \{i\}$ is sampled in Step 4, the fraction of y that can help us discover an anti-monotone edge in direction i using AE-SEARCH$(y, T' \cup \{i\})$ is at least $\Omega(\widetilde{\epsilon}^2/(\beta\sqrt{n}))$. Thus we observe an anti-monotone edge in direction i with probability at least $\Omega(\widetilde{\epsilon}^2/\alpha\sqrt{n})$ over the draw of each pair of $\mathbf{T}$ and $\mathbf{y}$ in Step 4. So by our choice of M, we observe such a violation with probability at least $1 - o(1)$. This finishes the proof of the lemma. ∎

Lemma 8. *The probability of a random 2^t-sized subset* $\mathbf{S}$ *being i-informative for some $i \in I$ is at least* $\Omega(\alpha/\log^2 n)$.

Proof: We lowerbound the number of subsets $S \subset [n]$ of size 2^t that are i-informative for some $i \in I$.

Using $\alpha < \log^2 n$, the fraction of 2^t-subsets S with

$$|S \cap I| \geq 4\log^2 n$$

is at most $\exp(-\Omega(\log^2 n))$. Next we let

$$\mathcal{S} = \left\{S \subset [n] : |S| = 2^t \text{ and } |S \cap I| \leq 4\log^2 n\right\}$$

and consider the following auxiliary bipartite graph H^*: vertices on the LHS are $i \in I$; vertices on the RHS are $S \in \mathcal{S}$; a pair (i, S) is an edge if S contains i and is i-informative. Thus, it suffices to show that many $S \in \mathcal{S}$ on the RHS of H^* are not isolated.

Using Lemma 4, for each $i \in I$, at least $1/8$ of $S' \in \mathcal{P}_{i,t}$ are informative for the ith direction. If $S' \in \mathcal{P}_{i,t}$ is one such set then $(i, S' \cup \{i\})$ is an edge when $S' \cup \{i\} \in \mathcal{S}$. So the degree of i is at least

$$\frac{1}{8} \cdot |\mathcal{P}_{i,t}| - \exp\left(-\Omega(\log^2 n)\right)\binom{n}{2^t} = \Omega\big(|\mathcal{P}_{i,t}|\big)$$
$$= \Omega\left(\binom{n-1}{2^t-1}\right).$$

On the other hand, each $S \in \mathcal{S}$ has degree at most $4\log^2 n$, since $|S \cap I| \leq 4\log^2 n$ for every $S \in \mathcal{S}$. Thus, the number of vertices on the RHS that are not isolated is at least

$$|I| \cdot \Omega\left(\binom{n-1}{2^t-1}\right) \cdot \frac{1}{4\log^2 n} \geq \Omega\left(\frac{|I|}{\log^2 n} \cdot \binom{n-1}{2^t-1}\right).$$

As a result, the probability of a random 2^t-sized set $\mathbf{S}$ being i-informative for some $i \in I$ is at least

$$\Omega\left(\frac{|I|}{\log^2 n} \cdot \frac{\binom{n-1}{2^t-1}}{\binom{n}{2^t}}\right) = \Omega\left(\frac{\alpha}{\log^2 n}\right).$$

This finishes the proof of the lemma. ∎

By our choice of K, a set S that is i-informative for some $i \in I$ is sampled in the K loops with probability $1 - o(1)$. By Lemma 7 a violation is found with probability $1 - o(1)$.

F. Proof of Lemma 4

Proof: Let γ denote the fraction of $S \in \mathcal{P}_{i,t}$ that are not informative for the ith coordinate. Then by definition, at least one of the two conditions must hold:

1) At least $(\gamma/2)$-fraction of $S \in \mathcal{P}_{i,t}$ have
$$\textsc{Good-Frac}_i^+(S) < 0.1 \cdot \frac{\widetilde{\epsilon}^2}{\alpha\sqrt{n}}; \qquad (10)$$
2) At least $(\gamma/2)$-fraction of $S \in \mathcal{P}_{i,t}$ have at least 0.9-fraction of $(2^r - 1)$-sized subsets $T \subseteq S$ with
$$\textsc{Good-Frac}_i^-(T) < 0.1 \cdot \frac{\widetilde{\epsilon}^2}{\beta\sqrt{n}}.$$

Below we show that $\gamma \le 5/8$ in the first case, and $\gamma \le 7/8$ in the second case.

We start with the first case, where at least $\gamma/2$ fraction of $S \in \mathcal{P}_{i,t}$ have (10). Consider the following two methods of sampling a pair $(\mathbf{x}, \mathbf{S})$ which is *not good* for E_i^+:

- We first sample $\mathbf{x}$ from $\textsc{Strong}_i^+$ and then sample $\mathbf{S}$ from $\mathcal{P}_{i,t}$, both uniformly at random.
- We first sample $\mathbf{S}$ from $\mathcal{P}_{i,t}$ and then sample $\mathbf{x}$ from $\textsc{Strong}_i^+$, both uniformly at random.

The probabilities of sampling a pair $(\mathbf{x}, \mathbf{S})$ that is not good for E_i^+ under the two methods are the same since both are equal to the fraction of (x, S) that are not good among
$$\textsc{Strong}_i^+ \times \mathcal{P}_{i,t}.$$

Using the first way of sampling, the probability that $(\mathbf{x}, \mathbf{S})$ is not good is at most $1/4$, since each $x \in \textsc{Strong}_i^+$ has at least $(3/4)$-fraction of $S \in \mathcal{P}_{i,t}$ such that (x, S) is a good pair. Using the second method, on the other hand, we have
$$\Pr\big[(\mathbf{x}, \mathbf{S}) \text{ is not good}\big] \ge \frac{\gamma}{2}\left(1 - \frac{2^n\left(0.1\frac{\widetilde{\epsilon}^2}{\alpha\sqrt{n}}\right)}{|\textsc{Strong}_i^+|}\right) \ge \frac{\gamma}{2}\cdot 0.8$$
where we used $H \le 2|I|/\widetilde{\epsilon}^2$ and thus,
$$\frac{|\textsc{Strong}_i^+|}{2^n} \ge \frac{\sqrt{n}}{H2^t} \ge \frac{\sqrt{n}\widetilde{\epsilon}^2}{2|I|2^t} = \frac{\widetilde{\epsilon}^2}{2\alpha\sqrt{n}}.$$

Combining both inequalities, we obtain that $\gamma \le 5/8$.

Now we handle the second case using a similar argument, by sampling a pair $(\mathbf{x}, \mathbf{T})$ that is not good for E_i^- using the following two methods:

- We first sample $\mathbf{x}$ from $\textsc{Strong}_i^-$, $\mathbf{S}$ from $\mathcal{P}_{i,t}$, and then sample $\mathbf{T} \subseteq \mathbf{S}$ of size $2^r - 1$, which is essentially sampling $\mathbf{T}$ uniformly from $\mathcal{P}_{i,r}$.
- We first sample $\mathbf{S} \in \mathcal{P}_{i,t}$ uniformly at random, and then sample a subset $\mathbf{T} \subseteq \mathbf{S}$ of size $2^r - 1$ uniformly at random, and finally we sample $\mathbf{x}$ from $\textsc{Strong}_i^-$.

Similarly to the first case, the probability of sampling a pair $(\mathbf{x}, \mathbf{T})$ that is not good is at most $1/4$ using the first method. Using the second method, we have
$$\Pr\big[(\mathbf{x}, \mathbf{T}) \text{ is not good}\big] \ge 0.9 \cdot \frac{\gamma}{2} \cdot \left(1 - \frac{2^n\left(0.1\frac{\widetilde{\epsilon}^2}{\beta\sqrt{n}}\right)}{|\textsc{Strong}_i^-|}\right)$$
$$\ge \frac{\gamma}{2} \cdot 0.9 \cdot 0.8.$$

Combining the two inequalities, we obtain that $\gamma \le 7/8$. ■

Acknowledgment

We thank Rocco Servedio and Li-Yang Tan for countless discussions and suggestions. This work is supported in part by NSF CCF-1149257, CCF-1423100 and the NSF Graduate Research Fellowship under Grant No. DGE-16-44869.

References

[1] R. Baleshzar, D. Chakrabarty, R. K. S. Pallavoor, S. Raskhodnikova, and C. Seshadhri, "Optimal unateness testers for real-values functions: Adaptivity helps," in *Proceedings of the 44th International Colloquium on Automata, Languages and Programming*, 2017.

[2] X. Chen, E. Waingarten, and J. Xie, "Beyond Talagrand functions: New lower bounds for testing monotonicity and unateness," in *Proceedings of the 49th ACM Symposium on the Theory of Computing*, 2017.

[3] R. Baleshzar, D. Chakrabarty, R. K. S. Pallavoor, S. Raskhodnikova, and C. Seshadhri, "A lower bound for nonadaptive, one-sided error testing of unateness of Boolean functions over the hypercube," *preprint arXiv:1706.00053*, 2017.

[4] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samordinsky, "Testing monotonicity," *Combinatorica*, vol. 20, no. 3, pp. 301–337, 2000.

[5] D. Chakrabarty and C. Seshadhri, "An optimal lower bound for monotonicity testing over hypergrids," *Theory of Computing*, vol. 10, no. 17, pp. 453–464, 2014.

[6] X. Chen, R. A. Servedio, and L.-Y. Tan, "New algorithms and lower bounds for monotonicity testing," in *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science*, 2014.

[7] S. Khot, D. Minzer, and M. Safra, "On monotonicity testing and Boolean isoperimetric type theorems," in *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, 2015.

[8] X. Chen, A. De, R. A. Servedio, and L.-Y. Tan, "Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries," in *Proceedings of the 47th ACM Symposium on the Theory of Computing*, 2015.

[9] A. Belovs and E. Blais, "A polynomial lower bound for testing monotonicity," in *Proceedings of the 48th ACM Symposium on the Theory of Computing*, 2016.

[10] D. Chakrabarty and S. Comandur, "An $o(n)$ monotonicity tester for Boolean functions over the hypercube," *SIAM Journal on Computing*, vol. 45, no. 2, pp. 461–472, 2016.

[11] S. Khot and I. Shinkar, "An $\widetilde{O}(n)$ queries adaptive tester for unateness," in *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, 2016.

[12] R. Baleshzar, M. Murzabulatov, R. K. S. Pallavoor, and S. Raskhodnikova, "Testing unateness of real-valued functions," *preprint arXiv:1608.07652*, 2016.

[13] D. Chakrabarty and C. Seshadhri, "A $\widetilde{O}(n)$ non-adaptive tester for unateness," *preprint arXiv:1608.06980*, 2016.

[14] M. Talagrand, "Isoperimetry, logarithmic Sobolev inequalities on the discrete cube, and Margulis' graph connectivity theorem," *Geometric and Functional Analysis*, vol. 3, no. 3, pp. 295–314, 1993.

[15] C. L. Canonne and T. Gur, "An adaptivity hierarchy theorem for property testing," in *Proceedings of the 32nd Computational Complexity Conference*, 2017.

[16] E. Blais, "Testing juntas nearly optimally," in *Proceedings of the 41st ACM Symposium on the Theory of Computing*, 2009.