Testing Unateness Nearly Optimally

Xi Chen
xichen@cs.columbia.edu
Columbia University
United States

Erik Waingarten
eaw@cs.columbia.edu
Columbia University
United States

ABSTRACT

We present an $\tilde{O}(n^{2/3}/\varepsilon^2)$-query algorithm that tests whether an unknown Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is unate (i.e., every variable is either non-decreasing or non-increasing) or ε-far from unate. The upper bound is nearly optimal given the $\tilde{\Omega}(n^{2/3})$ lower bound of Chen, Waingarten and Xie (2017). The algorithm builds on a novel use of the binary search procedure and its analysis over long random paths.

CCS CONCEPTS

• **Theory of computation → Streaming, sublinear and near linear time algorithms**.

KEYWORDS

Property testing, Boolean functions, unateness

ACM Reference Format:
Xi Chen and Erik Waingarten. 2019. Testing Unateness Nearly Optimally. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), June 23–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3313276.3316351

1 INTRODUCTION

A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is *monotone* if every variable is non-decreasing, and *unate* if every variable is either non-decreasing or non-increasing (equivalently, f is unate iff there exists a string $a \in \{0,1\}^n$ such that $g(x) := f(x \oplus a)$ is monotone). Both problems of *testing* monotonicity and unateness were introduced in [16], where a tester is a randomized algorithm that, given query access to an unknown Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, outputs "accept" with probability at least 2/3 when f is monotone (or unate) and outputs "reject" with probability at least 2/3 when f is ε-far from monotone (or unate).[1] The work of [16] analyzed the non-adaptive[2], one-sided error[3] edge tester[4] which led to the upper bounds of $O(n/\varepsilon)$ and $O(n^{3/2}/\varepsilon)$ for testing monotonicity and unateness, respectively. These remained the best upper bounds for over a decade.

Recently there have been some exciting developments in understanding the query complexity of both problems. Progress made on the upper bound side is due, in part, to new *directed* isoperimetric inequalities on the hypercube. In particular, [6] and [17] showed that various isoperimetric inequalities on the hypercube have directed analogues, where the edge boundary is now measured by considering anti-monotone bichromatic edges[5]. These inequalities were then used in the analysis of non-adaptive algorithms for testing monotonicity [6, 10, 17]. For example, to obtain the $\tilde{O}(\sqrt{n}/\varepsilon^2)$ upper bound, [17] used their new inequality to prove the existence of a large and almost regular bipartite graph that consists of anti-monotone bichromatic edges in any function that is ε-far from monotone. These upper bounds are complemented with lower bounds for testing monotonicity [4, 8, 10, 11, 15]. For non-adaptive algorithms, the query complexity has been pinned down to $\tilde{\Theta}(\sqrt{n})$ for constant ε; for general adaptive algorithms, a gap remains between $\tilde{O}(\sqrt{n}/\varepsilon^2)$ of [17] and the best lower bound of $\tilde{\Omega}(n^{1/3})$ [11].

Given the similarity in their definitions, it is natural to expect that the same directed isoperimetric inequalities can be used to test unateness: if f is far from unate, then by definition $f(x \oplus a)$ is far from monotone for any $a \in \{0,1\}^n$, on which one can then apply these inequalities to obtain rich graph structures. This is indeed the approach [12] followed to obtain an $\tilde{O}(n^{3/4})$-query adaptive algorithm for unateness by leveraging the directed isoperimetric inequality of [17]. It improved the upper bound of [16] as well as recent linear upper bounds for testing unateness [2, 3, 6, 18] (which turned out to be optimal [1, 11] for non-adaptive and one-sided error algorithms). Shortly before the work of [12], an adaptive lower bound of $\tilde{\Omega}(n^{2/3})$ was obtained in [11] for testing unateness.

Our main contribution is an $\tilde{O}(n^{2/3}/\varepsilon^2)$-query, adaptive algorithm for testing unateness. This essentially settles the problem since it matches the $\tilde{\Omega}(n^{2/3})$ adaptive lower bound of [11] up to a poly-logarithmic factor (when ε is a constant).

Theorem 1 (Main). *There is an $\tilde{O}(n^{2/3}/\varepsilon^2)$-query, adaptive algorithm with the following property: Given $\varepsilon > 0$ and query access to*

[1]Given a property $\mathcal{P}$ of Boolean functions, we say f is ε-far from $\mathcal{P}$ if for every $g \in \mathcal{P}$, $\mathbf{Pr}_{\boldsymbol{x}\sim\{0,1\}^n}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] \geq \varepsilon$ where $\boldsymbol{x} \sim \{0,1\}^n$ is sampled uniformly at random.

[2]An algorithm is non-adaptive if queries made cannot depend on answers to previous queries and thus, all queries can be made in a single batch. In contrast a general adaptive algorithm proceeds round by round: the point it queries in each round can depend on answers to previous queries.

[3]We say a tester makes one-sided error if it always accepts a function that satisfies the property.

[4]An edge tester keeps drawing edges $(\boldsymbol{x}, \boldsymbol{y})$ from the hypercube uniformly at random and querying $f(\boldsymbol{x})$ and $f(\boldsymbol{y})$.

[5]An edge $(x, x^{(i)})$ (where $x^{(i)}$ denotes the point obtained from x by flipping the ith bit) in $\{0,1\}^n$ is bichromatic if $f(x) \neq f(x^{(i)})$, is monotone (bichromatic) if $x_i = f(x)$, and is anti-monotone (bichromatic) if $x_i \neq f(x)$.

an unknown Boolean function $f: \{0,1\}^n \to \{0,1\}$, it always accepts when f is unate and rejects with probability at least $2/3$ when f is ε-far from unate.

In addition to the bipartite graph structure implied by the isoperimetric inequality of [17], the algorithm relies on novel applications of the standard binary search procedure on long random paths.

Given a path between two points x and y in the hypercube with $f(x) \neq f(y)$, the binary search (see Figure 1) returns a bichromatic edge along the path with $\log \ell$ queries where ℓ is the length of the path. The idea of using binary search in Boolean function property testing is not new. In every application we are aware of in this area (e.g., in testing conjunctions [13, 14], testing juntas [5, 9], unateness [18] and monotonicity [7]), one runs binary search to find bichromatic edges (or pairs, as in testing juntas) that can be directly used to form a violation (or at least part of it) to the property being tested. This is indeed how we use binary search in one of the cases of the algorithm (Case 2) to search for an *edge violation* (i.e., a pair of bichromatic edges along the same variable, one is monotone and the other is anti-monotone). However, in the most challenging case (Case 1) of the algorithm, binary search plays a completely different role. Instead of searching for an edge violation, binary search is used to *preprocess* a large set $S_0 \subseteq [n]$ of variables to obtain a subset $S \subseteq S_0$. This set S is used to search for bichromatic edges more efficiently using a procedure called AE-Search from [12]. Analyzing the performance of S for AE-Search is technically the most demanding part of the paper, where new ideas are needed for understanding the behavior of binary search running along long random paths in the hypercube.

1.1 Technical Overview

In this section we present a high-level overview of the algorithm, focusing on why and how we use binary search in Case 1 of the algorithm. For simplicity we assume ε is a constant.

First our algorithm rejects a function only when an edge violation to unateness is found. Since an edge violation is a certificate of non-unateness, the algorithm always accepts a function when it is unate and thus, it makes one-sided error. As a result, it suffices to show that the algorithm finds an edge violation with high probability when the unknown function f is far from unate.

For simplicity, we explain Case 1 of the algorithm using the following setting:[6] All edge violations of f are along a hidden set $\mathcal{I} \subset [n]$ of $\Omega(n)$ variables. For each variable $i \in \mathcal{I}$, there are $\Theta(2^n/n)$ monotone edges and $\Theta(2^n/n)$ anti-monotone edges. Let P_i^+ denote the set of points incident on monotone edges along i and P_i^- denote the set of points incident on anti-monotone edges along i. The sets P_i^+'s for $i \in \mathcal{I}$ are disjoint, so monotone edges along variables in $\mathcal{I}$ form a matching of size $\Theta(2^n)$; similarly, the sets P_i^-'s are disjoint and anti-monotone edges along $\mathcal{I}$ also form a matching of size $\Theta(2^n)$. Along each $i \notin \mathcal{I}$, there are $\Theta(2^n/\sqrt{n})$ bichromatic edges along i which are all either monotone or anti-monotone, but not both.

This particular case will highlight some of the novel ideas in the algorithm and the analysis, so we focus on this case for the technical overview.

[6]The following conditions on the function f are satisfied by the hard functions in [11] used for proving the $\tilde{\Omega}(n^{2/3})$ lower bound.

An appealing approach for finding an edge violation is to keep running binary search on points $\boldsymbol{x}, \boldsymbol{y}$ that are drawn independently and unifomly at random. Since a function that is far from unate must be ε-far from constant as well, $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$ with a constant probability and when this happens, binary search returns a bichromatic edge. Now in order to analyze the chance of observing an edge violation by repeating this process, two challenges arise. First, the output distribution given by the variable of the bichromatic edge found by binary search can depend on f in subtle ways, and becomes difficult to analyze formally (partly because of its adaptivity). Second, since the influence of variables outside $\mathcal{I}$ is $\Omega(1/\sqrt{n})$, a random path between $\boldsymbol{x}$ and $\boldsymbol{y}$ of $\Omega(n)$ edges may often cross $\Omega(\sqrt{n})$ bichromatic edges along variables outside of $\mathcal{I}$ and $O(1)$ bichromatic edges along variables in $\mathcal{I}$. In this case, binary search will likely return a bichromatic edge along a variable outside $\mathcal{I}$, which is useless for finding an edge violation.

A less adaptive (and thus much simpler to analyze) variant of binary search called AE-Search was introduced [12] to overcome these two difficulties. The subroutine AE-Search (f, x, S) queries f and takes two additional inputs: $x \in \{0,1\}^n$ and a set $S \subseteq [n]$ of variables, uses $O(\log n)$ queries and satisfies the following property:

Property of AE-Search: If $(x, x^{(i)})$ is a bichromatic edge with $i \in S$ and both x and $x^{(i)}$ are $(S \setminus \{i\})$-*persistent* (which for x informally means that $f(x) = f(x^{(\mathbf{T})})$ with high probability when $\mathbf{T}$ is a uniformly random subset of $S \setminus \{i\}$ of half of its size), then AE-Search (f, x, S) finds the edge $(x, x^{(i)})$ with probability at least $2/3$. (1)

In some sense, AE-Search (f, x, S) efficiently checks whether there exists an $i \in S$ such that $(x, x^{(i)})$ is bichromatic, whereas the trivial algorithm for this task takes $O(|S|)$ queries.[7]

In this simplified setting, the algorithm of [12] starts by drawing a size-$\sqrt{n}$ set $\mathbf{S} \subseteq [n]$ uniformly at random and runs AE-Search $(f, \boldsymbol{x}, \mathbf{S})$ on independent samples $\boldsymbol{x}$ for $n^{3/4}$ times, hoping to find an edge violation. To see why this works we first note that $|\mathbf{S} \cap \mathcal{I}| = \Omega(\sqrt{n})$ with high probability. Moreover, the following property holds for $\mathbf{S}$:

Property of the Random Set S: With $\Omega(1)$ probability over the randomness of $\mathbf{S}$, most $i \in \mathbf{S} \cap \mathcal{I}$ satisfy that most points in P_i^+ and P_i^- are $(\mathbf{S} \setminus \{i\})$-persistent. (2)

We sketch its proof since it highlights the technical challenge we will face later.

First we view the sampling of $\mathbf{S}$ as $\mathbf{S}' \cup \{\boldsymbol{i}\}$, where $\mathbf{S}'$ is a random set of size $\sqrt{n} - 1$ and $\boldsymbol{i}$ is a random variable in $[n]$. Since the influence of each variable in $\mathbf{S}'$ is at most $O(1/\sqrt{n})$, for many points $x \in \{0,1\}^n$ most random paths of length $O(\sqrt{n})$ along variables in $\mathbf{S}'$ starting at x will not cross any bichromatic edges. In other words, most random sets $\mathbf{S}'$ of size $\sqrt{n} - 1$ satisfy that most of points in $\{0,1\}^n$ are $\mathbf{S}'$-persistent with high constant probability. Given that $\cup_i P_i^+$ and $\cup_i P_i^-$ are both $\Omega(1)$-fraction of $\{0,1\}^n$, most points in $\cup_i P_i^+$ and $\cup_i P_i^-$ must be $\mathbf{S}'$-persistent as well. On the other hand, given that $\boldsymbol{i}$ is *independent* from $\mathbf{S}'$ and that $\mathcal{I}$ is $\Omega(n)$,

[7]See Definition 2.2 and its relation to the performance of AE-Search in Lemma 2.3 for a formal description

with probability $\Omega(1)$ many points in P_i^+ and P_i^- are S'-persistent. The property of S follows by an argument of expectation.

With properties of both S and AE-Search in hand in (1) and (2), as well as the fact that $|\mathrm{S} \cap \mathcal{I}| = \Omega(\sqrt{n})$ with high probability, we expect to find a bichromatic edge along a variable in $\mathrm{S} \cap \mathcal{I}$ after $\sqrt{n}$ executions of AE-Search (since the union of P_i^+ and P_i^- for $i \in \mathrm{S} \cap \mathcal{I}$ consists of $\Omega(1/\sqrt{n})$-fraction of $\{0,1\}^n$). Moreover, the variable is (roughly) uniformly over $\mathrm{S} \cap \mathcal{I}$ and (roughly) equally likely to be monotone or anti-monotone. It follows from the birthday paradox that repeating AE-Search for $O(n^{1/4}) \cdot \sqrt{n}$ rounds is enough to find an edge violation.

The natural question is whether we can make S larger (e.g., of size $n^{2/3}$) without breaking property (2). This would lead to an $\tilde{O}(n^{2/3})$-query algorithm (for the simplified setting). However, it is no longer true that many random paths of length $\Omega(n^{2/3})$ do not cross bichromatic edges because the influence of variables along variables in $\mathrm{S} \setminus \mathcal{I}$ is $\Omega(1/\sqrt{n})$. Therefore, large S may not satisfy property (2) and as a result, AE-Search may never output bichromatic edges along variables in $\mathrm{S} \cap \mathcal{I}$. This limit to sets of size at most $O(\sqrt{n})$ was a similar bottleneck in [17], and the connection between $|\mathrm{S}|$ and the total influence of f was later explored in [7]. Indeed, if (2) held for S of size larger than $\sqrt{n}$, then one could improve on the $O(\sqrt{n})$-query algorithm of [17] for testing monotonicity. Consequently, if one believes that monotonicity testing requires $\Omega(\sqrt{n})$ *adaptive* queries, it is natural to conjecture that the algorithm in [12] is optimal for testing unateness.

The key insight in this work is to *preprocess* the set S before using AE-Search. For our simplified setting, we first sample $\mathrm{S}_0 \subset [n]$ of size $n^{2/3}$ (much larger than what the analysis in [7, 12, 17] would allow) uniformly at random. Then, we set $\mathrm{S} = \mathrm{S}_0$, and repeat the following steps for $n^{2/3} \cdot \mathrm{polylog}(n)$ many iterations:

> `Preprocess`: Sample $\boldsymbol{x} \in \{0,1\}^n$ uniformly at random. Check if $\boldsymbol{x}$ is S-persistent by drawing $\mathrm{polylog}(n)$ many subsets $\mathbf{T} \subseteq \mathrm{S}$ of half of its size uniformly at random. If a $\mathbf{T}$ with $f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{(\mathbf{T})})$ is found, run binary search on a random path from $f(\boldsymbol{x})$ to $f(\boldsymbol{x}^{(\mathbf{T})})$ to find a bichromatic edge along variable $\boldsymbol{i}$ and remove $\boldsymbol{i}$ from S.

At a high level, the analysis of the algorithm would proceed as follows. At the end of `Preprocess`, for every $i \in \mathrm{S}$, most points in $\{0,1\}^n$ are $(\mathrm{S} \setminus \{i\})$-persistent. Otherwise, `Preprocess` would remove more variables from S since points which are not $(\mathrm{S} \setminus \{i\})$-persistent cannot be very S-persistent. At the same time, most variables in $\mathrm{S}_0 \cap \mathcal{I}$ at the beginning survive in S at the end (given that variables in $\mathcal{I}$ have very low influence). It may seem that we can now conclude property (2) holds for S, and that a violation is found after $O(n^{2/3})$ rounds of AE-Search$(f, \boldsymbol{x}, \mathrm{S})$ when $\boldsymbol{x}$ is uniform.

However, the tricky (and somewhat subtle) problem is that, even though most points in $\{0,1\}^n$ are $(\mathrm{S} \cap \{i\})$-persistent for every $i \in \mathrm{S} \cap \mathcal{I}$, it is not necessarily the case that points *inside* P_i^+ and P_i^- are $(\mathrm{S} \cap \{i\})$-persistent, since $P_i^+ \cup P_i^-$ is only a $O(1/n)$-fraction of the hypercube. Compared with the argument from [12] above for $\sqrt{n}$-sized uniformly random sets, after preprocessing S_0 (which was a uniform random set) with multiple rounds of binary search, the set S left can be very far from random. More specifically, the set S obtained from S_0 will heavily depend on the function f and, in principle, a clever adversary could design a function so that `Preprocess` running on S_0 deliberately outputs a set S that where points in P_i^+ and P_i^- are not $(\mathrm{S} \setminus \{i\})$-persistent.

The main technical challenge is to show that this is not possible when variables in $\mathcal{I}$ have low influence,[8] and the desired property for S remains valid. To this end, we show that for any variable i with low influence, the following two distributions supported on preprocessed sets S have small total variation distance. The first distribution samples $\mathrm{S}_0' \subset [n]$ of size $(n^{2/3} - 1)$ and outputs the set $\mathrm{S}' \cup \{i\}$ obtained from preprocessing S_0'. The second distribution $\mathrm{S}_0' \subset [n]$ of size $(n^{2/3} - 1)$ and outputs the set S obtained from preprocessing $\mathrm{S}_0' \cup \{i\}$. Intuitively this means that a low-influence variable i has little impact on the result S of `Preprocess` and thus, `Preprocess` is oblivious to i and cannot deliberately exclude P_i^+ and P_i^- from the set of S-persistent points.

To analyze the total variation distance between the results of running `Preprocess` on S_0' and $\mathrm{S}_0' \cup \{i\}$, we need to understand how a low-influence variable i can affect the result of a binary search on a *long random path* (given that `Preprocess` is just a sequence of calls to binary search). The random paths have length $|\mathrm{S}_0| = n^{2/3}$ at the beginning of `Preprocess`, and are repeated for $\tilde{O}(n^{2/3})$ rounds. Giving more details, we show that a variable i with influence $\mathbf{Inf}_f[i]$ can affect the result of a binary search on a random path of length ℓ with probability at most $\log \ell \cdot \mathbf{Inf}_f[i]$, instead of the trivial upper bound of $\ell \cdot \mathbf{Inf}_f[i]$, which is the probability that a variable i affects the evaluation of f on vertices of a random path of length ℓ. This is proved in Claim 3.3 (although the formal statement is slightly different since we need to introduce a placeholder when running binary search on the set without i so that the two paths have the same length; see Section 2.1).

In order to go beyond the assumptions on the function given in this overview, the algorithm needs to deal with more general cases: (1) Monotone (or anti-monotone) edges of $\mathcal{I}$ may not form a matching, but rather, a large and almost-regular bipartite graph whose existence follows from the directed isoperimetric inequality of [17]. (2) Although [17] implies the existence of such graphs with bichromatic edges from $\mathcal{I}$, there may be more bichromatic edges along $\mathcal{I}$ outside of these two graphs, which would raise the influence of these variables to the point where `Preprocess` is no longer oblivious of these variables. Intuitively, this implies that bichromatic edges which give rise to edge violations are abundant, so finding them becomes easier. This is handled in Case 2, where we give an algorithm (also based on binary search) which finds many bichromatic edges along these high influence variables, and combine it with the techniques from [12] to find an edge violation. (3) The set $\mathcal{I}$ can be much smaller than n, in which case, the techniques from [12] actually achieves better query complexity. We formalize this in Case 3 of the algorithm.

1.2 Organization

We review preliminaries, recall the binary search procedure and review the definition of persistency and the AE-Search procedure in Section 2. We present the preprocessing procedure in Section

[8]In the simplified setting, each variable $i \in \mathcal{I}$ has influence only $O(1/n)$; In the real situation, we need to handle the case even when each variable has influence as high as $1/n^{2/3}$.

3 and prove that a low-influence variable has small impact on its output. We use the directed isoperimetric lemma of [17] to establish a so-called Scores Lemma in Section 4, which roughly speaking helps us understand how good the set S is after preprocessing (in terms of using it to run AE-Search to find a bichromatic edge along a certain variable). We separate our main algorithm into three cases in Section 5, depending on different combinations of parameters. As we are limited in space, we provide a sketch of the required lemma statements without proofs for Case 1 in Section 6. We omit the necessary proofs, as well as the analysis for Cases 2 and 3 of the algorithm to the full version of this paper.

2 PRELIMINARIES

We will use bold-faced letters such as $\mathbf{T}$ and $\boldsymbol{x}$ to denote random variables. For $n \geq 1$, we write $[n] = \{1, \ldots, n\}$. In addition, we write $g = \tilde{O}(f)$ to mean $g = O(f \cdot \text{polylog}(f))$ and $g = \tilde{\Omega}(f)$ to mean $g = \Omega(f/\text{polylog}(f))$.

For $x \in \{0,1\}^n$, and a set $S \subset [n]$, we write $x^{(S)} \in \{0,1\}^n$ as the point given by letting $x^{(S)}_k = x_k$ for all $k \notin S$, and $x^{(S)}_k = 1 - x_k$ for all $k \in S$ (i.e., $x^{(S)}$ is obtained from x by *flipping* variables in S). When $S = \{i\}$ is a singleton set, we abbreviate $x^{(i)} = x^{(\{i\})}$ and say that $x^{(i)}$ is obtained from x by flipping the ith variable. Throughout the paper, we use $n+1$ as the name of a *placeholder* variable (i.e., a dummy variable). If $x \in \{0,1\}^n$ and $S \subseteq [n+1]$, then $x^{(S)} := x^{(S \setminus \{n+1\})}$, and in particular, $x^{(n+1)} = x$. We will refer to this as *flipping* variable $n+1$ (see Section 2.1) although no change is made on x. For a subset $S \subseteq [n+1]$ and a variable $i \in [n]$, we let $\mathsf{Sub}(S, i) \subseteq [n+1]$ be the subset obtained by *substituting* $n+1$ with i and i with $n+1$. In other words,

$$\mathsf{Sub}(S,i) = \begin{cases} S & \text{if } i, n+1 \in S \text{ or } i, n+1 \notin S \\ (S \cup \{n+1\}) \setminus \{i\} & \text{if } i \in S \text{ and } n+1 \notin S \\ (S \cup \{i\}) \setminus \{n+1\} & \text{if } n+1 \in S \text{ and } i \notin S. \end{cases}$$

We will at times endow $S \subseteq [n+1]$ with an *ordering* $\pi \colon [|S|] \to S$ which is a bijection indicating that $\pi(i)$ is the ith element of S under π. When $T \subset S$, the ordering $\tau \colon [|T|] \to T$ obtained from π is the unique bijection such that for all $i, j \in T$, $\tau^{-1}(i) < \tau^{-1}(j)$ if and only if $\pi^{-1}(i) < \pi^{-1}(j)$. Moreover, when $S \subseteq [n+1]$ and π is an ordering of S, the ordering π' of $\mathsf{Sub}(S,i)$ obtained from π is obtained by substituting $n+1$ with i and i with $n+1$ in the ordering, i.e., $\pi'(k) = \pi(k)$ when $\pi(k) \notin \{i, n+1\}$, $\pi'(k) = n+1$ if $\pi(k) = i$ and $\pi'(k) = i$ if $\pi(k) = n+1$.

Given a Boolean function $f \colon \{0,1\}^n \to \{0,1\}$, and a variable $i \in [n]$, we say that $(x, x^{(i)})$ is a *bichromatic edge* of f along variable i if $f(x) \neq f(x^{(i)})$; it is monotone (bichromatic) if $x_i = f(x)$ and anti-monotone (bichromatic) if $x_i \neq f(x)$. The *influence* of variable i in f is defined as

$$\mathbf{Inf}_f[i] = \Pr_{\boldsymbol{x} \sim \{0,1\}^n} \left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{(i)}) \right],$$

which is twice the number of bichromatic edges of f along i divided by 2^n. The *total influence* of f, $\mathbf{I}_f = \sum_{i \in [n]} \mathbf{Inf}_f[i]$, is twice the number of bichromatic edges of f divided by 2^n. Given distributions μ_1 and μ_2 on some sample space Ω, the *total variation distance* between μ_1 and μ_2 is given by

$$d_{\mathrm{TV}}(\mu_1, \mu_2) = \max_{S \subseteq \Omega} \left| \mu_1(S) - \mu_2(S) \right|.$$

```
Subroutine BinarySearch $(f, x, S, \pi)$
Input: Query access to $f \colon \{0,1\}^n \to \{0,1\}$, a point $x \in \{0,1\}^n$, a
nonempty set $S \subseteq [n+1]$ and an ordering $\pi$ of $S$.
Output: Either $i \in S$ and a point $y \in \{0,1\}^n$ where $(y, y^{(i)})$ is a
bichromatic edge, or nil.
(1) Query $f(x)$ and $f(x^{(S)})$ and return nil if $f(x) = f(x^{(S)})$.
(2) Let $m = |S|$ and $x = x_0, x_1, \ldots, x_m = x^{(S)}$ be the sequence
    of points obtained from $x$ by flipping variables in the order
    of $\pi(1), \ldots, \pi(m)$: $x_i = x_{i-1}^{(\pi(i))}$. Let $\ell = 0$ and $r = m$.
(3) While $r - \ell > 1$ do
(4)     Let $t = \lceil (\ell + r)/2 \rceil$ and query $f(x_t)$. If $f(x_\ell) \neq f(x_t)$
        set $r = t$; otherwise set $\ell = t$.
(5) Return $\pi(r)$ and $y = x_\ell$.
```

Figure 1: Description of the binary search subroutine for finding a bichromatic edge.

2.1 Binary Search With A Placeholder

We use the subroutine $\mathsf{BinarySearch}(f, x, S, \pi)$ described in Figure 1, where $f \colon \{0,1\}^n \to \{0,1\}$ is a Boolean function, $x \in \{0,1\}^n$, S is a nonempty subset of $[n+1]$, and π is an ordering of S.

When $S \subseteq [n]$, $\mathsf{BinarySearch}(f, x, S, \pi)$ performs as the standard binary search algorithm: $x = x_0, x_1, \ldots, x_{|S|} = x^{(S)}$ is a path from x to $x^{(S)}$ in which x_t is obtained from x_{t-1} by flipping variable $\pi(t) \in S \subseteq [n]$, and when $f(x) \neq f(x^{(S)})$, the binary search is done along this path to find an edge that is bichromatic. Now in general S may also contain $n+1$, which we use as the name of a placeholder variable. Similarly, when $f(x) \neq f(x^{(S)})$, the binary search is done along the path $x = x_0, x_1, \ldots, x_{|S|} = x^{(S)}$ (recall that $x^{(S)}$ is defined as $x^{(S \setminus \{n+1\})}$ when S contains $n+1$) where x_t is obtained from x_{t-1} by flipping variable $\pi(t)$ (in particular, when $\pi(t) = n+1$, $x_t = x_{t-1}$).

Note that even though $n+1$ is a placeholder variable, given $S \subseteq [n+1]$ with $n+1 \in S$ and an ordering π of S, queries made by $\mathsf{BinarySearch}(f, x, S, \pi)$ and $\mathsf{BinarySearch}(f, x, S \setminus \{n+1\}, \pi')$ (where π' is the ordering of $S \setminus \{n+1\}$ obtained from π) are different, so their results may also be different. We summarize properties of $\mathsf{BinarySearch}$ in the following lemma.

Lemma 2.1. *$\mathsf{BinarySearch}(f, x, S, \pi)$ uses $O(\log n)$ queries and satisfies the following property. If $f(x) = f(x^{(S)})$, it returns* nil*; if $f(x) \neq f(x^{(S)})$, it returns a variable $i \in S \setminus \{n+1\}$ and a point $y \in \{0,1\}^n$ along the path from x to $x^{(S)}$ with ordering π such that $(y, y^{(i)})$ is bichromatic.*

2.2 Persistency With Respect To A Set Of Variables

We need the following notion of persistency for points and edges with respect to a set of variables.

Definition 2.2. *Given a Boolean function $f \colon \{0,1\}^n \to \{0,1\}$, a set $S \subseteq [n+1]$ of variables and a point $x \in \{0,1\}^n$, we say that x is*

S-persistent *if the following two conditions hold:*

$$\Pr_{\substack{\mathbf{T}\subseteq S\\ |\mathbf{T}|=\lfloor |S|/2\rfloor}}\Big[f(x)=f(x^{(\mathbf{T})})\Big]\geq 1-\frac{1}{\log^2 n}\quad \textit{and}$$

$$\Pr_{\substack{\mathbf{T}\subseteq S\\ |\mathbf{T}|=\lfloor |S|/2\rfloor+1}}\Big[f(x)=f(x^{(\mathbf{T})})\Big]\geq 1-\frac{1}{\log^2 n}.$$

where $\mathbf{T}$ *is a subset of* S *of certain size drawn uniformly at random. Note that when* $S=\emptyset$*, every point in* $\{0,1\}^n$ *is trivially* S*-persistent.*

Let e *be an edge in* $\{0,1\}^n$*. We say that* e *is* S-persistent *if both points of* e *are* S*-persistent.*

The notion of persistency above is useful because it can be used to formulate a clean sufficient condition for AE-Search (f,x,S) to find a bichromatic edge $(x,x^{(i)})$ for some $i\in S$ with high probability. This is captured in Lemma 2.3 (see Lemma 6.5 in [12]) below.

Lemma 2.3. *Given a point* $x\in\{0,1\}^n$ *and a set* $S\subseteq[n+1]$*, AE-Search* (f,x,S) *makes* $O(\log n)$ *queries to* f*, and returns either an* $i\in S$ *such that* $(x,x^{(i)})$ *is a bichromatic edge, or "fail."*

Let $(x,x^{(i)})$ *be a bichromatic edge of* f *along* $i\in[n]$*. If* $i\in S$ *and* $(x,x^{(i)})$ *is* $(S\setminus\{i\})$*-persistent, then both AE-Search* (f,x,S) *and AE-Search* $(f,x^{(i)},S)$ *output* i *with probability at least* $2/3$*.*

Lemma 2.3 has the following immediate corollary.

Corollary 2.4. *Given a set* $S\subseteq[n+1]$ *and a point* $x\in\{0,1\}^n$*, there exists at most one variable* $i\in S$ *such that* $(x,x^{(i)})$ *is both bichromatic and* $(S\setminus\{i\})$*-persistent.*

Proof. If the condition holds for both $i\neq j\in S$, then from Lemma 2.3 AE-Search (f,x,S) would return both i and j with probability at least $2/3$, a contradiction. □

3 PREPROCESSING VARIABLES

Our goal in this section is to present a preprocessing procedure called `Preprocess`. Given query access to a Boolean function $f\colon\{0,1\}^n\to\{0,1\}$, a nonempty set $S_0\subseteq[n+1]$ (again, $n+1$ serves here as a placeholder variable), an ordering π of S_0 and a parameter $\xi\in(0,1)$, `Preprocess` (f,S_0,π,ξ) makes $(|S_0|/\xi)\cdot\text{polylog}(n)$ queries and returns a subset $\mathbf{S}$ of S_0. At a high level, `Preprocess` keeps running `BinarySearch` to remove variables from S_0 until the set $\mathbf{S}\subseteq S_0$ left satisfies that at least $(1-\xi)$-fraction of points in $\{0,1\}^n$ are $\mathbf{S}$-persistent (recall Definition 2.2).

In addition to proving the above property for `Preprocess` in Lemma 3.1, we show in Lemma 3.2 the following: When $i\in S_0\subseteq[n]$ has low influence, then the result of running `Preprocess` on S_0 is *close* (see Lemma 3.1 for the formal statement) to that of running it on Sub (S_0,i) (in which we substitute i with the placeholder variable $n+1$).

3.1 The Preprocessing Procedure

The procedure `Preprocess` (f,S_0,π,ξ) is described in Figure 3. It uses a subroutine `CheckPersistence` (f,S,π,ξ) described in Figure 2. Roughly speaking, `CheckPersistence` checks if at least $(1-\xi)$-fraction of points in $\{0,1\}^n$ are S-persistent for the current set S. This is done by sampling points $\boldsymbol{x}$ and subsets $\mathbf{T}$ of S of the right sizes uniformly at random, and checking if $f(\boldsymbol{x})=f(\boldsymbol{x}^{(\mathbf{T})})$, for $\log^4 n/\xi$ many rounds. If `CheckPersistence` finds $\boldsymbol{x}$ and $\mathbf{T}$ such that $f(\boldsymbol{x})\neq f(\boldsymbol{x}^{(\mathbf{T})})$, it runs binary search on them to find a bichromatic edge along some variable $i\in S$ and outputs i; otherwise it returns `nil`.

The main property we prove for `CheckPersistence` is that when the fraction of points that are not S-persistent is at least ξ, it returns a variable $i\in S$ with high probability.

The procedure `Preprocess` (f,S_0,π,ξ) sets $S=S_0$ and $\tau=\pi$ at the beginning and keeps calling `CheckPersistence`(f,S,τ,ξ) and removing the variable `CheckPersistence`(f,S,τ,ξ) returns from both S and the ordering τ, until `CheckPersistence` returns `nil` or S becomes empty in which case `Preprocess` terminates and returns S. As a result, `Preprocess` makes at most $|S_0|$ calls to `CheckPersistence`. It is unlikely that ξ-fraction of points are not S-persistent but somehow `CheckPersistence` (f,S,τ,ξ) returns `nil`. This implies that at least $(1-\xi)$-fraction of $\{0,1\}^n$ are $\mathbf{S}$-persistent for $\mathbf{S}=$ `Preprocess` (f,S_0,π,ξ) at the end with high probability.

We summarize our discussion above in the following lemma.

```
Subroutine CheckPersistence (f, S, π, ξ)
Input: Query access to $f\colon\{0,1\}^n\to\{0,1\}$, a nonempty set
$S\subseteq[n+1]$, an ordering $\pi$ of $S$ and a parameter $\xi\in(0,1)$.
Output: Either nil or a variable $i\in S$.
  (1) Repeat the following steps $\log^4 n/\xi$ many times:
    (a) Sample a point $\boldsymbol{x}$ from $\{0,1\}^n$ uniformly at random.
    (b) Flip a fair coin and perform one of the following tasks:
        • Sample $\mathbf{T}\subseteq S$ with size $\lfloor|S|/2\rfloor$ uniformly. Run
          BinarySearch$(f,\boldsymbol{x},\mathbf{T},\boldsymbol{\pi}')$ where $\boldsymbol{\pi}'$ is the ordering of
          $\mathbf{T}$ defined by $\pi$ restricted on $\mathbf{T}$. If
          BinarySearch$(f,\boldsymbol{x},\mathbf{T},\boldsymbol{\pi}')$ returns a variable $i$ and a
          point $y$, output $i$.
        • Sample $\mathbf{T}\subseteq S$ with size $\lfloor|S|/2\rfloor+1$ uniformly. Run
          BinarySearch$(f,\boldsymbol{x},\mathbf{T},\boldsymbol{\pi}')$ where $\boldsymbol{\pi}'$ is the ordering of
          $\mathbf{T}$ defined by $\pi$ restricted on $\mathbf{T}$. If
          BinarySearch$(f,\boldsymbol{x},\mathbf{T},\boldsymbol{\pi}')$ returns a variable $i$ and a
          point $y$, output $i$.
  (2) If BinarySearch always returned nil, output nil.
```

Figure 2: Description of the subroutine `CheckPersistence`**.**

```
Procedure Preprocess (f, S_0, π, ξ)
Input: Query access to $f\colon\{0,1\}^n\to\{0,1\}$, a nonempty set
$S_0\subseteq[n+1]$, an ordering $\pi$ of $S_0$ and a parameter $\xi\in(0,1)$.
Output: A subset $S\subseteq S_0$.
  (1) Initially, let $S=S_0$ and $\tau=\pi$.
  (2) While $S$ is nonempty do
  (3)     Run CheckPersistence $(f,S,\tau,\xi)$.
  (4)     If it returns nil, return $S$; otherwise (it returns an
          $i\in S$), remove $i$ from $S$ and $\tau$.
  (5) Return $S$ (which must be the empty set to reach this line).
```

Figure 3: Description of the procedure `Preprocess` **for preprocessing a set of variables.**

Lemma 3.1. *Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, a nonempty $S_0 \subseteq [n+1]$, an ordering π of S_0 and a parameter $\xi \in (0,1)$,* Preprocess (f, S_0, π, ξ) *makes at most $O(|S_0|\log^5 n/\xi)$ queries to f and with probability at least $1 - \exp\left(-\Omega(\log^2 n)\right)$, it outputs a subset* $\mathsf{S} \subseteq S_0$ *such that at least $(1-\xi)$-fraction of points in $\{0,1\}^n$ are* S*-persistent.*

3.2 Low Influence Variables Have Low Impact On Preprocess

In the rest of the section, we show that when $S_0 \subseteq [n]$, a variable $i \in S_0$ with low influence $\textbf{Inf}_f[i]$ has low impact on the result of $\mathsf{S} =$ Preprocess(f, S_0, π, ξ). More formally, we show that one can substitute i by the placeholder $n+1$ and the result of running Preprocess on Sub(S_0, i) is almost the same (after substituting $n+1$ back to i in the result of Preprocess).

This is made more precise in the following lemma:

Lemma 3.2. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. Let $i \in S_0 \subseteq [n]$, π be an ordering of S and $\xi \in (0,1)$. Let $S_0' =$* Sub(S_0, i) *be the subset of $[n+1]$ and let π' be the ordering of S_0' obtained from π by substituting i with $n+1$. Then we have*

$$d_{\mathrm{TV}}(\texttt{Preprocess}(f, S_0, \pi, \xi), \quad \texttt{Sub}(\texttt{Preprocess}(f, S_0', \pi', \xi), i)) \le O\left(\frac{|S_0|\log^5 n}{\xi}\right)\cdot \textbf{Inf}_f[i].$$

Because Preprocess keeps calling CheckPersistence which keeps calling BinarySearch, we start the proof of Lemma 3.2 with the following claim concerning the binary search procedure.

Claim 3.3. *Let $i \in S \subseteq [n]$ and π be an ordering of S. Let $S' =$* Sub(S, i)*, and π' be the ordering of S' obtained from π by substituting i with $n+1$. We let $\boldsymbol{u}$ and $\boldsymbol{v}$ be the random variables where*

- *$\boldsymbol{u}$ is the output of* BinarySearch $(f, \boldsymbol{x}, S, \pi)$ *when $\boldsymbol{x}$ is drawn from $\{0,1\}^n$ uniformly, and*
- *$\boldsymbol{v}$ is the output of* BinarySearch $(f, \boldsymbol{z}, S', \pi')$ *when $\boldsymbol{z}$ is drawn from $\{0,1\}^n$ uniformly.*

Then, we have $d_{\mathrm{TV}}(\boldsymbol{u}, \boldsymbol{v}) \le O(\log n)\cdot \textbf{Inf}_f[i]$.

Proof. Our plan is to show that for every point $x \in \{0,1\}^n$ with a certain property, we have

$$\left\{\texttt{BinarySearch}(f, x, S, \pi), \texttt{BinarySearch}(f, x^{(i)}, S, \pi)\right\} \tag{1}$$

as a multiset is the same as

$$\left\{\texttt{BinarySearch}(f, x, S', \pi'), \texttt{BinarySearch}(f, x^{(i)}, S', \pi')\right\}. \tag{2}$$

It turns out that the property holds for most points in $\{0,1\}^n$. The lemma then follows.

To describe the property we let $m = |S| = |S'|$ and let $k = \pi^{-1}(i)$ (with $\pi'(k) = n+1$). We let $J \subseteq [0:m]$ denote the set of indices taken by variables ℓ and r (see Figure 1 for settings of ℓ and r) in an execution of BinarySearch along a path of length m that outputs the kth edge at the end. For example, ignoring the rounding issue, J always contains $0, m$ and $m/2$: these are indices of the first three points that binary search examines. It contains $3m/4$ if $k > m/2$, or $m/4$ if $k \le m/2$, so on and so forth. The set J also always contains $k-1$ and k: these are indices of the last two points that binary search examines before returning the kth edge.

Now we describe the property. Given $x \in \{0,1\}^n$ we let $x = x_0, \ldots, x_m = x^{(S)}$ with $x_t = x_{t-1}^{(\pi(t))}$ for all $t \in [m]$. We let $C(x)$ be the indicator of the condition that:

$$f(x_j) = f(x_j^{(i)}), \quad \text{for all } j \in J. \tag{3}$$

We show that $\boldsymbol{x} \sim \{0,1\}^n$ satisfies $C(\boldsymbol{x})$ with high probability. Because $\boldsymbol{x}$ is drawn uniformly from $\{0,1\}^n$, $\boldsymbol{x}_j$ defined above is also distributed uniformly for each $j \in J$ and thus, the probability that a specific $j \in J$ violates the condition above is at most $\textbf{Inf}_f[i]$. It then follows from a union bound over $j \in J$ that the fraction of points that violate the condition $C(x)$ is at most $\textbf{Inf}_f[i]\cdot O(\log n)$.

It suffices to prove that when $x \in \{0,1\}^n$ satisfies $C(x)$, the two multisets in (1) and (2) are the same. To this end we write down the two paths in the multiset (1) that start with x and $x^{(i)}$ as

$$x_0, x_1, \ldots, x_m \quad \text{and} \quad y_0, y_1, \ldots, y_m$$

in which $x_t = x_{t-1}^{(\pi(t)}$ and $y_t = x_t^{(i)}$. Similarly we write down the two paths for (2) as

$$z_0, z_1, \ldots, z_m \quad \text{and} \quad w_0, w_1, \ldots, w_m,$$

in which we have $z_t = x_t$ for all $t < k$ and $z_t = y_t$ for all $t \ge k$; $w_t = y_t$ for all $t < k$ and $w_t = x_t$ for all $t \ge k$. It follows from the property (3) of x that

$$f(x_j) = f(y_j) = f(z_j) = f(w_j), \quad \text{for all } j \in J. \tag{4}$$

Since $0, m \in J$ we have that $f(x_0) = f(x_m)$ implies the same holds for y, z and w in which case (1) and (2) are trivially the same since they all return nil. So we assume below that $f(x_0) \ne f(x_m)$ and thus, all four binary searches return a variable and a bichromatic edge.

Next, since $k-1, k \in J$ we have

$$\begin{aligned} f(x_{k-1}) &= f(x_k) = f(y_{k-1}) = f(y_k) \\ &= f(z_{k-1}) = f(z_k) = f(w_{k-1}) = f(w_k). \end{aligned}$$

As a result, the kth edge is not bichromatic in all four paths and thus, during each run of binary search, k is removed from the interval $[\ell : r]$ (see Figure 1) after a certain number of rounds. Moreover, it follows from the definition of J and (4) that in all four runs of binary search, the values of ℓ and r are the same at the moment when k is removed from consideration (i.e., at the first time when either ℓ or r is updated so that $k \notin [\ell : r]$. We consider two cases for the values of ℓ and r.

(1) $\ell > k$: In this case, BinarySearch (f, x, S, π) continues to search on the path $x_\ell, \ldots, x_r$ and BinarySearch $(f, x^{(i)}, S', \pi')$ continues to search on the path $w_\ell, \ldots, w_r$ which is the same as $x_\ell, \ldots, x_r$ given that $\ell > k$. As a result, their outputs are the same. Similarly we have that BinarySearch $(f, x^{(i)}, S, \pi)$ is the same as BinarySearch (f, x, S', π') in this case.

(2) $r < k$: In this case, BinarySearch (f, x, S, π) continues to search on the path $x_\ell, \ldots, x_r$ and BinarySearch (f, x, S', π') continues to search on the path $z_\ell, \ldots, z_r$ which is the same as $x_\ell, \ldots, x_r$ given that $r < k$. As a result, their outputs are the same. Similarly we have

that $\texttt{BinarySearch}\,(f, x^{(i)}, S, \pi)$ is the same as $\texttt{BinarySearch}\,(f, x^{(i)}, S', \pi')$ in this case.

As a result, the two multisets are the same when x satisfies the condition $C(x)$. □

Claim 3.3 gives the following corollary using a union bound:

Corollary 3.4. *Let $i \in S \subseteq [n]$ and π be an ordering of S. Let $S' = \texttt{Sub}\,(S, i)$ and π' be the ordering of S' obtained from π by substituting i with $n+1$. Then we have*

$$\begin{aligned} d_{\mathrm{TV}}(&\texttt{CheckPersistence}(f, S, \pi, \xi), \\ &\qquad\qquad \texttt{CheckPersistence}(f, S', \pi', \xi)) \\ &\le O\left(\frac{\log^5 n}{\xi}\right) \cdot \mathbf{Inf}_f[i]. \end{aligned}$$

Proof. We use the following coupling to run $\texttt{CheckPersistence}\,(f, S, \pi, \xi)$ and $\texttt{CheckPersistence}\,(f, S', \pi', \xi)$ in parallel.

For each round of CheckPersistence we first flip a fair coin and draw a subset $\mathbf{T}$ of S of the size indicated by the coin uniformly. Then we couple the binary search on $\boldsymbol{x} \sim \{0,1\}^n$ and $\mathbf{T}$ and the binary search on $\boldsymbol{z} \sim \{0,1\}^n$ and $\texttt{Sub}\,(\mathbf{T}, i)$ using the best coupling between them.

It then follows from Claim 3.3 and a union bound over the $\log^4 n/\xi$ rounds that the probability of this coupling of $\texttt{CheckPersistence}\,(f, S, \pi, \xi)$ and $\texttt{CheckPersistence}\,(f, S', \pi', \xi)$ returning different results is at most

$$(\log^4 n/\xi) \cdot \mathbf{Inf}_f[i] \cdot O(\log n).$$

This finishes the proof of the corollary. □

Now we prove Lemma 3.2.

Proof of Lemma 3.2. Let $m = |S| = |S'|$. For each $j \in [m]$, let X_j denote the output of the jth call to CheckPersistence in $\texttt{Preprocess}\,(f, S_0, \pi, \xi)$ with X_j set to nil by default if the procedure terminates before the jth call. Similarly we use Y_j to denote the output of the jth call in $\texttt{Preprocess}\,(f, S_0', \pi', \xi)$. Let $\mathrm{X} = (\mathrm{X}_1, \ldots, \mathrm{X}_m)$ and $\mathrm{Y} = (\mathrm{Y}_1, \ldots, \mathrm{Y}_m)$. Then $\mathrm{X} = \mathrm{Y}$ implies that $\mathrm{S} = \texttt{Preprocess}\,(f, S_0, \pi, \xi)$ is the same as $\mathrm{S}' = \texttt{Preprocess}\,(f, S_0', \pi', \xi)$. As a result, it suffices to show that

$$d_{\mathrm{TV}}(\mathrm{X}, \mathrm{Y}) \le \frac{m \log^5 n}{\xi} \cdot \mathbf{Inf}_f[i].$$

To this end, we first note that by Corollary 3.4 the total variation distance between X_1 and Y_1 is at most $\beta := O(\log^5 n/\xi) \cdot \mathbf{Inf}_f[i]$. On the other hand, note that if the outputs from the first $\ell - 1$ calls in $\texttt{Preprocess}\,(f, S_0, \pi, \xi)$ and $\texttt{Preprocess}\,(f, S_0', \pi', \xi)$ are the same, say $a_1, \ldots, a_{\ell-1}$, then before the ℓth call, the set S in the former still contains i and the S' in the latter can be obtained by substituting its i with $n+1$. It follows from Corollary 3.4 that, for any $\ell > 1$ and any $a_1, \ldots, a_{\ell-1}$, the total variation distance between the distribution of X_ℓ conditioning on $\mathrm{X}_1 = a_1, \ldots, \mathrm{X}_{\ell-1} = a_{\ell-1}$ and the distribution of Y_ℓ conditioning on $\mathrm{Y}_1 = a_1, \ldots, \mathrm{Y}_{\ell-1} = a_{\ell-1}$ is also at most β. We prove that these properties together imply that $d_{\mathrm{TV}}(\mathrm{X}, \mathrm{Y}) \le m\beta$, from which the lemma follows.[9]

For this purpose we use the following coupling of X and Y. First we use the best coupling for the distribution of X_1 and the distribution of Y_1 to draw $(\boldsymbol{a}_1, \boldsymbol{b}_1)$. Then we draw $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ from the the best coupling for the distribution of X_2 conditioning on $\mathrm{X}_1 = \boldsymbol{a}_1$ and the distribution of Y_2 conditioning on $\mathrm{Y}_1 = \boldsymbol{b}_1$. We then repeat until $(\boldsymbol{a}_m, \boldsymbol{b}_m)$ is drawn. It follows from the description that the marginal distribution of $\boldsymbol{a} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m)$ is the same as X and the marginal distribution of $\boldsymbol{b} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ is the same as Y. Moreover, we have

$$\begin{aligned} d_{\mathrm{TV}}(\mathrm{X}, \mathrm{Y}) &\le \mathbf{Pr}\left[\boldsymbol{a} \ne \boldsymbol{b}\right] \\ &= \mathbf{Pr}\left[\boldsymbol{a}_1 \ne \boldsymbol{b}_1\right] + \mathbf{Pr}\left[\boldsymbol{a}_1 = \boldsymbol{b}_1 \wedge \boldsymbol{a}_2 \ne \boldsymbol{b}_2\right] + \cdots \\ &\qquad + \mathbf{Pr}\left[\boldsymbol{a}_j = \boldsymbol{b}_j \text{ for } j < m \wedge \boldsymbol{a}_m \ne \boldsymbol{b}_m\right], \end{aligned}$$

which is at most $m\beta$ by the description of the coupling and properties of X and Y. □

4 THE SCORES LEMMA

By definition when f is ε-far from unate, $f(x \oplus a)$ is ε-far from monotone for every $a \in \{0,1\}^n$. This means that we can utilize the directed isoperimetric inequality of [17] to show the existence of relatively large and almost-regular bipartite graphs that consist of bichromatic edges.The goal of this section is to show that, using these bipartite graphs, there exist certain probability distributions over subsets of variables such that a set S drawn from any of these distributions can be used to search for bichromatic edges via AE-Search efficiently.

To this end, we start by introducing three distributions $\mathcal{H}_{\xi,m}, \mathcal{D}_{\xi,m}$ and $\mathcal{P}_{i,m}$ in Section 4.1. We then use them to define a *score* for each variable $i \in [n]$ which aims to quantify the chance of finding a bichromatic edge along i using AE-Search and a set S drawn from some of those distributions. Finally we give the Scores Lemma in Section 4.2, which shows that the sum of scores over $i \in [n]$ is large when f is ε-far from unate and has total influence $O(\sqrt{n})$.

4.1 Distributions $\mathcal{D}_{\xi,m}, \mathcal{H}_{\xi,m}$ And $\mathcal{P}_{i,m}$ And The Definition Of Scores

We start by defining two distributions $\mathcal{D}_{\xi,m}$ and $\mathcal{H}_{\xi,m}$.

Definition 4.1. *Given $\xi \in (0,1)$ and $m : 1 \le m \le n$, we let $\mathcal{D}_{\xi,m}$ denote the following distribution supported on subsets of $[n]$: $\mathrm{S} \sim \mathcal{D}_{\xi,m}$ is drawn by first sampling a subset S_0 of $[n]$ of size m and an ordering $\boldsymbol{\pi}$ of S_0 uniformly at random. We then call $\texttt{Preprocess}\,(f, \mathrm{S}_0, \boldsymbol{\pi}, \xi)$ to obtain* S.

Similarly, let $\mathcal{H}_{\xi,m}$ denote the following distribution supported on subsets of $[n+1]$: $\mathrm{S} \sim \mathcal{H}_{\xi,m}$ is drawn by first sampling a subset S_0 of $[n+1]$ of size m with $n+1 \in \mathrm{S}_0$ and an ordering $\boldsymbol{\pi}$ of S_0 uniformly at random. We then call $\texttt{Preprocess}\,(f, \mathrm{S}_0, \boldsymbol{\pi}, \xi)$ to obtain S. *Notice that as $n+1$ is just a placeholder, we always have $n+1 \in \mathrm{S} \sim \mathcal{H}_{\xi,m}$.*

As it will become clear later, our unateness tester will sample subsets according to the distribution $\mathcal{D}_{\xi,m}$ and use them to find an

[9]We suspect that this is probably known in the literature but were not able to find a reference.

edge violation to unateness when f is far from unate. While this section is mainly concerned about $\mathcal{H}_{\xi,m}$, it will only be used in the analysis to help us understand how good those samples from $\mathcal{D}_{\xi,m}$ are in terms of revealing an edge violation to unateness.

Let $\Lambda = \lceil 2\log(n/\varepsilon)\rceil$ in the rest of the paper. Given $i \in [n]$ and $m : 1 \le m \le n-1$ we use $\mathcal{P}_{i,m}$ to denote the uniform distribution over all size-m subsets of $[n] \setminus \{i\}$.

Next we use $\mathcal{H}_{\xi,m}$ and $\mathcal{P}_{i,m}$ to define strong edges.

Definition 4.2 (Strong edges). *Let e be a bichromatic edge of f along variable $i \in [n]$. We say e is ℓ-strong, for some integer $\ell \in [\Lambda]$, if the following two conditions hold:*

1. *For every $m \le n^{2/3}$ as a power of 2 and every $\xi = 1/2^k$ with $\ell \le k \le \Lambda$, the edge e is $\mathbf{S}$-persistent (recall Definition 2.2) with probability at least $1-(1/\log n)$ when $\mathbf{S} \sim \mathcal{H}_{\xi,m}$.*
2. *The edge e is $\mathbf{S}$-persistent with probability at least $1-(1/\log n)$ when $\mathbf{S} \sim \mathcal{P}_{i,\lceil\sqrt{n}/2^\ell\rceil}$.*

For each $i \in [n]$ and $\ell \in [\Lambda]$, we define

$$\textsc{Score}^{+}_{i,\ell}(f) = \frac{1}{2^n} \cdot \text{number of } \ell\text{-strong monotone edges along variable } i.$$

We analogously define $\textsc{Score}^{-}_{i,\ell}(f)$ for anti-monotone edges along variable i. Finally we define

$$\textsc{Score}^{+}_{i}(f) = \max_{\ell\in[\Lambda]} \left\{ \textsc{Score}^{+}_{i,\ell}(f) \cdot \frac{1}{2^\ell} \right\}, \tag{5}$$

and we analogously define $\textsc{Score}^{-}_{i}(f)$.

4.2 The Scores Lemma

We state the Scores Lemma:

Lemma 4.3. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be ε-far from unate with $\mathbf{I}_f < 6\sqrt{n}$. Then there are $s,t \in [\Lambda]$, $h \in [3\Lambda]$ and a set $\mathcal{I} \subseteq [n]$ such that $|\mathcal{I}|$ is a power of 2, $|\mathcal{I}|/2^h = \Omega(\varepsilon^2/\Lambda^{11})$ and every $i \in \mathcal{I}$ has*

$$\min\left\{ \textsc{Score}^{+}_{i,s} \cdot \frac{1}{2^s},\ \textsc{Score}^{-}_{i,t} \cdot \frac{1}{2^t} \right\} \ge \frac{1}{2^h}. \tag{6}$$

We note that our Scores Lemma above looks very similar to Lemma 4.3 from [12]. Thus the proof follows a similar trajectory. The main difference is that we are varying the distributions from which the set $\mathbf{S}$ of variables is drawn. Compared to [12] we not only consider the quality of $\mathbf{S}$ drawn from the $\mathcal{P}$ distribution in the definition of strong edges but also those drawn from $\mathcal{H}_{\xi,m}$ with a number of possible combinations of ξ and m in the indicated range. This makes the proof of the lemma slightly more involved than that of Lemma 4.3 in [12].

5 THE MAIN ALGORITHM

We now describe the main algorithm for testing unateness. The algorithm rejects a function f only when an edge violation has been found. As a result, for its correctness it suffices to show that when the input function f is ε-far from unate, the algorithm finds an edge violation with probability at least 2/3. For convenience we will suppress $\text{polylog}(n/\varepsilon)$ factors using $\tilde{O}(\cdot)$ in the rest of analysis.

The main algorithm has four cases. Case 0 is when the input function f satisfies $\mathbf{I}_f > 6\sqrt{n}$. In this case an $\tilde{O}(\sqrt{n})$-query algorithm is known [2] (also see Lemma 2.1 of [12]).

From now on, we assume that f is not only ε-far from unate but also satisfies $\mathbf{I}_f \le 6\sqrt{n}$. Then there are parameters $s,t \in [\Lambda]$ and $h \in [3\Lambda]$ and a set $\mathcal{I} \subseteq [n]$ with which Lemma 4.3 holds for f. We may assume that the algorithm knows s,t,h and $|\mathcal{I}| = 2^\ell$ (by trying all possibilities, which just incurs an addition factor of $O(\Lambda^4)$ in the query complexity). We may further assume without loss of generality that $s \ge t$ since the case of $s < t$ is symmetric.

We consider the following three cases of f:

Case 1: $|\mathcal{I}|/2^t \ge n^{2/3}$ and and at least half of $i \in \mathcal{I}$ satisfy

$$\mathbf{Inf}_f[i] \le \left(\frac{\varepsilon^2}{\Lambda^{13}}\right) \cdot \frac{n^{1/3}}{|\mathcal{I}|}\ ; \tag{7}$$

Case 2: $|\mathcal{I}|/2^t \ge n^{2/3}$ and and at least half of $i \in \mathcal{I}$ violate (7); and
Case 3: $|\mathcal{I}|/2^t \le n^{2/3}$.

Lemma 5.1. *Let $s \ge t \in [\Lambda]$, $h \in [3\Lambda]$, and $\ell \in [\lfloor\log n\rfloor]$ with $2^\ell/2^t \ge n^{2/3}$. There is a $\tilde{O}(n^{2/3}/\varepsilon^2)$-query algorithm with the following property. Given any Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ that satisfies (i) Lemma 4.3 holds for f with s,t,h and a set $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| = 2^\ell$; and (ii) at least half of $i \in \mathcal{I}$ satisfy (7), the algorithm finds an edge violation to unateness with probability at least 2/3.*

Lemma 5.2. *Let $s \ge t \in [\Lambda]$, $h \in [3\Lambda]$, and $\ell \in [\lfloor\log n\rfloor]$ with $2^\ell/2^t \ge n^{2/3}$. There is an algorithm that makes $\tilde{O}(n^{2/3}/\varepsilon^2)$ queries and satisfies the following property. Given any Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ that satisfies (i) Lemma 4.3 holds for f with s,t,h and a set $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| = 2^\ell$ and (ii) at least half of $i \in \mathcal{I}$ violate (7), the algorithm finds an edge violation to unateness with probability at least 2/3.*

Lemma 5.3. *Let $s \ge t \in [\Lambda]$, $h \in [3\Lambda]$, and $\ell \in [\lfloor\log n\rfloor]$ with $2^\ell/2^t \le n^{2/3}$. There is an algorithm that makes $\tilde{O}(n^{2/3} + \sqrt{n}/\varepsilon^2)$ queries and satisfies the following property. Given any Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ that satisfies Lemma 4.3 with s,t,h and a set $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| = 2^\ell$, the algorithm finds an edge violation of f to unateness with probability at least 2/3.*

Theorem 1 follows by combining all these lemmas.

6 THE ALGORITHM FOR CASE 1

Let $s \ge t \in [\Lambda]$, $h \in [3\Lambda]$ and $\ell \in [\lfloor\log n\rfloor]$. In Case 1 the input function $f\colon \{0,1\}^n \to \{0,1\}$ satisfies Lemma 4.3 with parameters s,t,h and $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| = 2^\ell$, with $|\mathcal{I}|/2^t \ge n^{2/3}$. At least half of the variables $i \in \mathcal{I}$ have low influence as given in (7). Let i be such a variable. Then by (7),

$$\left(\frac{\varepsilon^2}{\Lambda^{13}}\right) \cdot \frac{n^{1/3}}{|\mathcal{I}|} > \mathbf{Inf}_f[i] \ge 2 \cdot \textsc{Score}^{+}_{i,s} \ge \frac{2^s}{2^h}.$$

Letting $\xi = 1/2^s$ throughout this section, it follows from Lemma 4.3 that

$$\xi = \Omega\left(\frac{\Lambda^{13}}{\varepsilon^2} \cdot \frac{|\mathcal{I}|}{2^h n^{1/3}}\right) = \Omega\left(\frac{\Lambda^2}{n^{1/3}}\right). \tag{8}$$

6.1 Informative Sets

We start with the notion of *informative sets*. Note that we will have different notions of informative sets in different cases of the algorithm. We use the same name because they serve similar purposes.

Given $i \in [n]$ and a set $S \subseteq [n+1]$ we use $\text{PE}_i^+(S)$ to denote the set of s-strong monotone edges along variable i that are S-persistent. We define $\text{PE}_i^-(S)$ similarly for antimonotone edges.

Definition 6.1 (Informative Sets). *A set $S \subseteq [n+1]$ is i-*informative *for monotone edges if*

$$\frac{|\text{PE}_i^+(S)|}{2^n} \geq \frac{\textsc{Score}_{i,s}^+}{4} \geq \frac{2^{s-h}}{4} \tag{9}$$

*and that $S \subseteq [n+1]$ is i-*informative *for anti-monotone edges if*

$$\frac{|\text{PE}_i^-(S)|}{2^n} \geq \frac{\textsc{Score}_{i,t}^-}{4} \geq \frac{2^{t-h}}{4}. \tag{10}$$

*We simply say that $S \subseteq [n+1]$ is i-*informative *if S satisfies both (9) and (10).*

Lemma 6.2. *For each $i \in \mathcal{I}$ and each positive integer $m \leq n^{2/3}$ that is a power of 2, $\mathbf{S} \sim \mathcal{H}_{\xi,m}$ is i-informative with probability at least $1 - o(1)$.*

Proof. We first show that $\mathbf{S} \sim \mathcal{H}_{\xi,m}$ satisfies (9) with probability at least $1 - o(1)$. The same argument works to show $\mathbf{S} \sim \mathcal{H}_{\xi,m}$ satisfies (10). The lemma then follows from a union bound. To this end, let α be the probability of $\mathbf{S} \sim \mathcal{H}_{\xi,m}$ being i-informative for monotone edges. We examine

$$\Pr_{\boldsymbol{e},\mathbf{S}}\Big[\boldsymbol{e} \text{ is } \mathbf{S}\text{-persistent}\Big],$$

where $\boldsymbol{e}$ is an s-strong monotone edge along variable i drawn uniformly at random and $\mathbf{S} \sim \mathcal{H}_{\xi,m}$. It follows from the definition of strong edges that the probability is at least $1 - 1/\log n$. On the other hand, we can also upperbound the probability using α (and the definition of i-informative sets) as $(1-\alpha)/4 + \alpha$. Solving the inequality we get $\alpha \geq 1 - o(1)$. □

Next we introduce two new families of distributions that will help us connect $\mathcal{H}_{\xi,m}$ with $\mathcal{D}_{\xi,m}$.

Definition 6.3. *Given $\xi \in (0,1)$, m: $1 \leq m \leq n$ and $i \in [n]$, we let $\mathcal{D}_{\xi,m,i}$ denote the following distribution supported on subsets of $[n]$: $\mathbf{S} \sim \mathcal{H}_{\xi,m,i}$ is drawn by first sampling a subset $\mathbf{S}_0$ of $[n]$ of size m with $i \in \mathbf{S}_0$ and an ordering $\boldsymbol{\pi}$ of $\mathbf{S}_0$ uniformly at random. We then call* `Preprocess` $(f, \mathbf{S}_0, \boldsymbol{\pi}, \xi)$ *and set $\mathbf{S}$ to be its output.*

Similarly, $\mathcal{H}_{\xi,m,i}$ denotes the following distribution supported on subsets of $[n+1]$: $\mathbf{S} \sim \mathcal{H}_{\xi,m,i}$ is drawn by first sampling a subset $\mathbf{S}_0$ of $[n+1] \setminus \{i\}$ of size m with $n+1 \in \mathbf{S}_0$ and an ordering $\boldsymbol{\pi}$ of $\mathbf{S}_0$ uniformly at random. We then call `Preprocess` $(f, \mathbf{S}_0, \boldsymbol{\pi}, \xi)$ *and set $\mathbf{S}$ to be its output.*

Using the fact that the total variation distance between the $\mathbf{S}_0$ used in $\mathcal{H}_{\xi,m}$ (at the beginning of the process) and the $\mathbf{S}_0$ used in $\mathcal{H}_{\xi,m,i}$ is at most m/n, we have

$$d_{\text{TV}}\Big(\mathcal{H}_{\xi,m}, \mathcal{H}_{\xi,m,i}\Big) \leq m/n$$

and the following corollary from Lemma 6.2.

Corollary 6.4. *For every $i \in \mathcal{I}$ and every positive integer $m \leq n^{2/3}$ as a power of 2, we have that $\mathbf{S} \sim \mathcal{H}_{\xi,m,i}$ is i-informative with probability at least $1 - o(1)$.*

The next two lemmas allow us to draw random subsets and still obtain i-informative sets. They enable us to use techniques from [12] for particular cases of our algorithm.

Lemma 6.5. *For every $i \in \mathcal{I}$ we have $\mathbf{T} \sim \mathcal{P}_{i,\lceil\sqrt{n}/2^t\rceil}$ is i-informative for anti-monotone edges with probability at least $1 - o(1)$.*

Proof. Similarly to the proof of Lemma 6.2, we write α to denote the probability of $\mathbf{T} \sim \mathcal{P}_{i,\lceil\sqrt{n}/2^t\rceil}$ being i-informative for anti-monotone edges. We examine

$$\Pr_{\boldsymbol{e},\mathbf{T}}\Big[\boldsymbol{e} \text{ is } \mathbf{T}\text{-persistent}\Big],$$

where $\boldsymbol{e}$ is a t-strong anti-monotone edge along variable i drawn uniformly and $\mathbf{T} \sim \mathcal{P}_{i,\lceil\sqrt{n}/2^t\rceil}$. It follows from the definition of strong edges that this probability is at least $1 - 1/\log n$. On the other hand, we can also upperbound the probability using α (and the definition of i-informative sets for anti-monotone edges) as $(1-\alpha)/4 + \alpha$. Solving the inequality we get $\alpha \geq 1 - o(1)$. □

Similarly, we may conclude the analogous lemma for monotone edges, whose proof follows similarly to Lemma 6.5.

Lemma 6.6. *For every $i \in \mathcal{I}$ we have that $\mathbf{S} \sim \mathcal{P}_{i,\lceil\sqrt{n}/2^s\rceil}$ is i-informative for monotone edges with probability at least $1 - o(1)$.*

6.2 Catching Variables: Relating $\mathcal{D}_{\xi,m}$ And $\mathcal{H}_{\xi,m}$

Now we focus on the variables in $\mathcal{I}$ that satisfy (7). To this end, we let $\mathcal{I}^*$ be a subset of $\mathcal{I}$ of size $\lceil|\mathcal{I}|/2\rceil$ such that all variables in $\mathcal{I}^*$ satisfy (7). Given that the algorithm knows the size of $\mathcal{I}$, it also knows the size of $\mathcal{I}^*$ (though not variables within). Next we use m to denote the largest power of 2 that is at most $\xi|\mathcal{I}|/n^{1/3}$. In other words, m is the unique power of 2 satisfying

$$\frac{\xi|\mathcal{I}|}{2n^{1/3}} < m \leq \frac{\xi|\mathcal{I}|}{n^{1/3}}. \tag{11}$$

Given that $|\mathcal{I}| \geq |\mathcal{I}|/2^t \geq n^{2/3}$ and (8), we have $m \gg 1$ and $m = \Theta(\xi|\mathcal{I}|/n^{1/3})$.

We now turn to analyzing the distribution $\mathcal{D}_{\xi,m}$ with the m defined above.

Definition 6.7 (Catching Variables). *Let $i \in \mathcal{I}^*$. We say that a set $S \subseteq [n]$* catches *the variable i if $i \in S$ and* $\textsf{Sub}\,(S, i) = (S \cup \{n+1\}) \setminus \{i\}$ *is i-informative (see Definition 6.1). We let*

$$\textsc{Caught}(S) = \Big\{i \in \mathcal{I}^* : S \text{ catches } i\Big\}.$$

Intuitively, if we sample $\mathbf{S} \sim \mathcal{D}_{\xi,m}$ and $i \in \textsc{Caught}(\mathbf{S})$, then we have an upper bound for how many samples $\boldsymbol{x}$ we need for $\textsc{AE-Search}(f, \boldsymbol{x}, \mathbf{S} \cup \{n+1\})$ to reveal a bichromatic edge along i.

Claim 6.8. *For every $i \in \mathcal{I}^*$, we have*

$$\Pr_{\mathbf{S}\sim\mathcal{D}_{\xi,m,i}}\Big[\mathbf{S} \text{ catches } i\Big] \geq \Pr_{\mathbf{T}\sim\mathcal{H}_{\xi,m,i}}\Big[\mathbf{T} \text{ is } i\text{-informative }\Big] - o(1).$$

Proof. We show the total variation distance between $\mathbf{S} \sim \mathcal{D}_{\xi,m,i}$ and $\textsf{Sub}\,(\mathbf{T}, i)$ over $\mathbf{T} \sim \mathcal{H}_{\xi,m,i}$ is

$$O\left(\frac{m\log^8 n}{\xi} \cdot \mathbf{Inf}_f[i]\right) = O\left(\frac{|\mathcal{I}|\log^8 n}{n^{1/3}} \cdot \mathbf{Inf}_f[i]\right) = o(1), \tag{12}$$

```
Procedure AlgorithmCase1.1 ($f$)
Input: Query access to a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$
Output: Either "unate," or two edges constituting an edge violation of $f$ to unateness.
(1) Repeat the following $O(1)$ times:
(2)     Draw $\mathbf{S} \sim \mathcal{D}_{\xi,m}$: First draw a size-$m$ subset $\mathbf{S}_0$ of $[n]$ and an ordering $\boldsymbol{\pi}$ of $\mathbf{S}_0$
        uniformly at random and then call Preprocess $(f, \mathbf{S}_0, \boldsymbol{\pi}, \xi)$.
(3)     Repeat $q$ times, where $q = O\left(n^{2/3}\Lambda^{13}\big/\varepsilon^2\right)$:
(4)         Draw an $\boldsymbol{x} \in \{0,1\}^n$ uniformly and run AE-Search $(f, \boldsymbol{x}, \mathbf{S} \cup \{n+1\})$
(5)     Let $\mathbf{A}$ be the set of $i \in [n]$ such that an anti-monotone edge along $i$ is found
(6)     Repeat $q$ times:
(7)         Draw an $\boldsymbol{y} \in \{0,1\}^n$ uniformly and run AE-Search $(f, \boldsymbol{y}, \mathbf{S} \cup \{n+1\})$
(8)     Let $\mathbf{B}$ be the set of $i \in [n]$ such that a monotone edge along variable $i$ is found
(9)     If $\mathbf{A} \cap \mathbf{B} \neq \emptyset$, output an edge violation of $f$ to unateness.
(10) Output "unate."
```

Figure 4: Algorithm for Case 1.1

given (7) and $i \in \mathcal{I}^*$. The lemma follows from the observations that $\texttt{Sub}\,(\mathbf{T}, i)$ contains i and when $\mathbf{T}$ is i-informative, $\texttt{Sub}\,(\mathbf{T}, i)$ catches i.

To upperbound the total variation distance between $\mathbf{S} \sim \mathcal{D}_{\xi,m,i}$ and $\texttt{Sub}\,(\mathbf{T}, i)$ over $\mathbf{T} \sim \mathcal{H}_{\xi,m,i}$, we use the following coupling. First we draw a subset $\mathbf{S}_0$ of $[n]$ with $i \in \mathbf{S}_0$ and an ordering $\boldsymbol{\pi}$ of $\mathbf{S}_0$ uniformly at random. Then we set $\mathbf{S}_0' = \texttt{Sub}\,(\mathbf{S}_0, i)$ and $\boldsymbol{\pi}'$ to be the ordering of $\mathbf{S}_0'$ obtained from $\boldsymbol{\pi}$ by replacing i with $n+1$. Finally we draw the output from the best coupling for $\texttt{Preprocess}\,(f, \mathbf{S}_0, \boldsymbol{\pi}, \xi)$ and $\texttt{Sub}\,(\texttt{Preprocess}\,(f, \mathbf{S}_0', \boldsymbol{\pi}', \xi), i)$. The upper bound in (12) follows directly from Lemma 3.2. □

6.3 Algorithm For Case 1.1

There are two sub-cases in Case 1. Specifically, for the remainder of Section 6.3, we assume that

$$m \geq \frac{n^{1/3}\log^2 n}{2^t}, \tag{13}$$

and handle the other case in Case 1.2. We let

$$r := \frac{m|\mathcal{I}^*|}{n} = \Omega(\log^2 n), \tag{14}$$

the expected size of the intersection of a random size-m subset of $[n]$ with $\mathcal{I}^*$, where we used (13) and $|\mathcal{I}^*|/2^t = \Omega(n^{2/3})$. Note that both m and r are known to the algorithm. We give Lemma 5.1 assuming (13) using $\texttt{AlgorithmCase1.1}$ in Figure 4, with the following query complexity.

Claim 6.9. *The query complexity of* $\texttt{AlgorithmCase1.1}$ *is* $\tilde{O}(n^{2/3}/\varepsilon^2)$.

The algorithm for Case 1.1 starts by sampling a set $\mathbf{S} \sim \mathcal{D}_{\xi,m}$. It then keeps drawing points $\boldsymbol{x}$ uniformly at random to run AE-Search $(f, \boldsymbol{x}, \mathbf{S} \cup \{n+1\})$ to find bichromatic edges, with the hope to find an edge violation along one of the variables in $\mathcal{I}^*$. We break lines 3–8 into the search of anti-monotone edges and the search of monotone edges separately only for the analysis later; algorithm wise there is really no need to do so. (The reason we use $\mathbf{S} \cup \{n+1\}$ instead of $\mathbf{S}$ in the algorithm lies in the proof of Lemma 6.10; roughly speaking, we need it to establish a connection between $\mathcal{D}_{\xi,m}$ and $\mathcal{H}_{\xi,m}$ so that we can carry the analysis on $\mathcal{H}_{\xi,m}$ that has been done so far over to $\mathcal{D}_{\xi,m}$.) On the one hand, recall from Lemma 2.3 that if $i \in S \subsetneq [n]$ and a bichromatic edge e along variable i is $\texttt{Sub}\,(S, i) = (S \cup \{n+1\}) \setminus \{i\}$-persistent, then running AE-Search on $S \cup \{n+1\}$ and any of the two points of e would reveal e with high probability. On the other hand, if a set (e.g., $\texttt{Sub}\,(S, i)$) is i-informative then it is persistent on a large fraction of edges along variable i.

Our first goal is to prove Lemma 6.10, which states that $\mathbf{S} \sim \mathcal{D}_{\xi,m}$ catches many variables.

Lemma 6.10. *We have* $|\textsc{Caught}(\mathbf{S})| \geq r/6$ *with probability* $\Omega(1)$.

Given Lemma 6.10, a constant fraction of the intersection of $\mathbf{S}$ and $\mathcal{I}^*$ will be caught, and therefore, AE-Search $(f, \boldsymbol{x}, \mathbf{S} \cup \{n+1\})$ will output a bichromatic edge along a variable from these caught coordinates for sufficiently many points $\boldsymbol{x}$. We fix S to be a set that catches at least $r/6$ many variables in $\mathcal{I}^*$, and claim that during this loop, an edge violation is found with probability $1 - o(1)$.

Given S, we write $\mathcal{J} \subseteq \mathcal{I}^*$ to denote the set of variables caught by S with $|\mathcal{J}| \geq r/6$. Then by definition we have that $\mathcal{J} \subseteq S$ and $\texttt{Sub}\,(S, j)$ is j-informative for every $j \in \mathcal{J}$.

We start by showing that $\mathbf{A} \cap \mathcal{J}$ is large with high probability.

Lemma 6.11. *We have* $|\mathbf{A} \cap \mathcal{J}| \geq \Omega\left(m2^t/n^{1/3}\right)$ *with probability at least* $1 - o(1)$.

Fix an A such that $C = A \cap \mathcal{J}$ satisfies the lower bound of Lemma 6.11. We finally show that $C \cap \mathbf{B}$ with high probability. This finishes Case 1.

Lemma 6.12. *We have that* $C \cap \mathbf{B}$ *is not empty with probability at least* $1 - o(1)$.

6.4 Algorithm For Case 1.2

The second subcase of Case 1 occurs when

$$m < \frac{n^{1/3}\log^2 n}{2^t}, \tag{15}$$

and the crucial difference is that unlike Case 1.1, we cannot conclude with (14). More specifically, two potential issues which were not present in Section 6.3 may arise: 1) sampling a set $\mathbf{S} \sim \mathcal{D}_{\xi,m}$ may result in $\textsc{Caught}(\mathbf{S}) = \emptyset$, and 2) even if $|\textsc{Caught}(\mathbf{S})|$ is large, too few points $\boldsymbol{x}$ may result in AE-Search $(f, \boldsymbol{x}, \mathbf{S})$ returning an anti-monotone edge (for instance, when $2^t = 1$ and $2^h = n$). Thus, we address these two problems with $\texttt{AlgorithmCase1.2}$, which is described in Figure 5.

At a high level, $\texttt{AlgorithmCase1.2}$ proceeds by sampling a set $\mathbf{T} \subseteq [n]$ of size

$$p := \lceil \sqrt{n}/2^t \rceil + 1 = \Theta(\sqrt{n}/2^t)$$

uniformly at random, and directly uses the set $\mathbf{T}$ to search for anti-monotone edges by repeating AE-Search $(f, \boldsymbol{x}, \mathbf{T})$ for $\tilde{O}(n^{2/3}/\varepsilon^2)$

```
Procedure AlgorithmCase1.2 ($f$)
Input: Query access to a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$
Output: Either "unate," or two edges constituting an edge violation of $f$ to unateness.
(1)  Repeat the following $O(1)$ times:
(2)      Draw a set $\mathbf{T} \subset [n]$ of size $p$ uniformly at random.
(3)      Repeat $O\left(n^{2/3}\Lambda^{11}\log^2 n\big/\varepsilon^2\right)$ times:
(4)          Draw an $\boldsymbol{x} \in \{0,1\}^n$ uniformly and run AE-Search $(f, \boldsymbol{x}, \mathbf{T})$
(5)      Let $\mathbf{A}$ be the set of $i \in [n]$ such that an anti-monotone edge along $i$ is found.
(6)      Repeat $O\left(n^{1/3}\log^3 n\big/(m2^t)\right)$ times:
(7)          Draw $\mathbf{S}$ by first drawing a subset $\mathbf{S}_0$ of $\mathbf{T}$ of size $m$ and an ordering
             $\boldsymbol{\pi}$ of $\mathbf{S}_0$ both uniformly at random and then call Preprocess $(f, \mathbf{S}_0, \boldsymbol{\pi}, \xi)$.
(8)          Repeat $O\left((2^h/2^s)\cdot\log n\right)$ times:
(9)              Draw $\boldsymbol{y} \in \{0,1\}^n$ uniformly and run AE-Search $(f, \boldsymbol{y}, \mathbf{S} \cup \{n+1\})$.
(10)     Let $\mathbf{B}$ be the set of $i \in [n]$ such that a monotone edge along variable $i$ is found.
(11)     If $\mathbf{A} \cap \mathbf{B} \neq \emptyset$, output an edge violation of $f$ to unateness.
(12) Output "unate."
```

Figure 5: Algorithm for Case 1.2

many iterations. We show at the end the algorithm will obtain anti-monotone edges along at least $\Omega(n^{1/6})$ variables in $\mathbf{T} \cap \mathcal{I}$ (as an anti-monotone edge is found every $\tilde{O}(\sqrt{n}/\varepsilon^2)$ iterations of AE-Search $(f, \boldsymbol{x}, \mathbf{S})$). The algorithm will then sample $\mathbf{S}_0 \subset \mathbf{T}$ of size m (notice that $m \ll p$ by (15)), pass it through Preprocess to obtain $\mathbf{S} \subseteq \mathbf{S}_0$, and then repeat AE-Search $(f, \boldsymbol{y}, \mathbf{S}_0)$ in hopes of observing an edge violation. In order to do so, $\mathbf{S}$ must contain a variable where the algorithm has already observed an anti-monotone edge. Since the number of variables with anti-monotone edges observed may be $O(n^{1/6})$ and $m \cdot n^{1/6}$ could be much smaller than p, the algorithm needs to sample the subset $\mathbf{S}_0$ from $\mathbf{T}$ multiple times.

We state the query complexity of the algorithm for Case 1.2, where we note that the upper bound will follow from (15), (8), (11), the fact that $2^t \geq 1$ and $2^h/\mathcal{I} \geq \tilde{\Omega}(\varepsilon^2)$.

Lemma 6.13. *The query complexity of* AlgorithmCase1.2 *is (using (8))*

$$\tilde{O}\left(\frac{n^{2/3}}{\varepsilon^2}\right) + \tilde{O}\left(\frac{n^{1/3}}{m2^t}\right)\left(\tilde{O}\left(\frac{m}{\xi}\right) + \tilde{O}\left(\frac{2^h}{2^s}\right)\right) = \tilde{O}(n^{2/3}/\varepsilon^2).$$

In order to analyze AlgorithmCase1.2 we consider one of the main iterations and let T denote the set drawn in line 2. We define the following subset $C \subseteq T \cap \mathcal{I}^*$ to capture how good T is: A variable $i \in T \cap \mathcal{I}^*$ belongs to C if it satisfies both of the following conditions:

(i) The set $T \setminus \{i\}$ is i-informative for anti-monotone edges; and
(ii) If we draw $\mathbf{S}$ by first drawing a subset $\mathbf{S}_0$ of T of size m conditioning on $i \in \mathbf{S}_0$ and an ordering of $\boldsymbol{\pi}$ of $\mathbf{S}_0$ uniformly at random and then calling Preprocess $(f, \mathbf{S}_0, \boldsymbol{\pi}, \xi)$ to get $\mathbf{S}$, then the probability that $\mathbf{S}$ catches i is at least $1/2$.

We prove that when $\mathbf{T}$ is a random size-p set, the set $\mathbf{C}$ is large with constant probability.

Lemma 6.14. *With probability at least $\Omega(1)$ over the draw of $\mathbf{T} \subset [n]$ in line 2, we have*

$$|\mathbf{C}| = \Omega\left(\frac{p|\mathcal{I}^*|}{n}\right).$$

Having established Lemma 6.14, we consider a fixed iteration of line 2 where the set C obtained from T satisfies the size lower bound from Lemma 6.14. Note that since line 2 is executed $O(1)$ times, this will happen with large constant probability. After fixing T and C, we will show that in this iteration, AlgorithmCase1.2 finds an edge violation to unateness with high probability.

Lemma 6.15. *With probability $1 - o(1)$ over the randomness in lines 3–5, $|\mathbf{A} \cap C| \geq \Omega(n^{1/6})$.*

By Lemma 6.15 we consider the case when $|\mathbf{A} \cap C| = \Omega(n^{1/6})$. Fix a particular A and a subset of $\mathcal{J} = A \cap C$ such that $|\mathcal{J}| = \Theta(n^{1/6})$.

Lemma 6.16. *With probability at least $1 - o(1)$, at least one of the $\mathbf{S}$ sampled in line 7 catches at least one variable in $\mathcal{J}$.*

Finally we show that if S catches a $j \in \mathcal{J}$, then line 9 finds a monotone edge along j.

Lemma 6.17. *If S catches $j \in \mathcal{J}$ in line 7, then line 9 finds a monotone edge along variable j with probability at least $1 - o(1)$.*

ACKNOWLEDGMENTS

We would like to thank Jinyu Xie for useful discussions. This work is supported in part by the NSF Graduate Research Fellowship under Grant No. DGE-16-44869 and NSF awards CCF-1703925 and IIS-1838154.

REFERENCES

[1] Roksana Baleshzar, Deeparnab Chakrabarty, Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and C. Seshadhri. 2017. A lower bound for nonadaptive, one-sided error testing of unateness of Boolean functions over the hypercube. *arXiv preprint arXiv:1706.00053* (2017).
[2] Roksana Baleshzar, Deeparnab Chakrabarty, Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and C. Seshadhri. 2017. Optimal Unateness Testers for Real-Values Functions: Adaptivity Helps. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP '2017)*.
[3] Roksana Baleshzar, Meiram Murzabulatov, Ramesh Krishnan S. Pallavoor, and Sofya Raskhodnikova. 2016. Testing unateness of real-valued functions. *arXiv preprint arXiv:1608.07652* (2016).
[4] Aleksandrs Belovs and Eric Blais. 2016. A polynomial lower bound for testing monotonicity. In *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC '2016)*. 1021–1032.
[5] Eric Blais. 2009. Testing juntas nearly optimally. In *Proceedings of the 41st ACM Symposium on the Theory of Computing (STOC '2009)*. 151–158.
[6] Deeparnab Chakrabarty and Seshadhri Comandur. 2016. An o(n) monotonicity tester for boolean functions over the hypercube. *SIAM J. Comput.* 45, 2 (2016), 461–472.
[7] Deeparnab Chakrabarty and C. Seshadhri. 2018. Adaptive Boolean monotonicity testing in total influence time. (2018).
[8] Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. 2015. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*. 519–528.
[9] Xi Chen, Zhengyang Liu, Rocco A. Servedio, Ying Sheng, and Jinyu Xie. 2018. Distribution-free junta testing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*.
[10] Xi Chen, Rocco A. Servedio, and Li-Yang Tan. 2014. New algorithms and lower bounds for monotonicity testing. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2014)*. 285–295.

[11] Xi Chen, Erik Waingarten, and Jinyu Xie. 2017. Beyond Talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proceedings of the 50th ACM Symposium on the Theory of Computing (STOC '2018)*.

[12] Xi Chen, Erik Waingarten, and Jinyu Xie. 2017. Boolean unateness testing with $\widetilde{\Omega}(n^{3/4})$ adaptive queries. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2017)*.

[13] X. Chen and J. Xie. 2016. Tight Bounds for the Distribution-Free Testing of Monotone Conjunctions. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[14] E. Dolev and D. Ron. 2011. Distribution-Free Testing for Monomials with a Sublinear Number of Queries. *Theory of Computing* 7, 1 (2011), 155–176.

[15] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. 2002. Monotonicity testing over general poset domains. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '2002)*. 474–483.

[16] Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samordinsky. 2000. Testing Monotonicity. *Combinatorica* 20, 3 (2000), 301–337.

[17] Subhash Khot, Dor Minzer, and Muli Safra. 2015. On monotonicity testing and boolean isoperimetric type theorems. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2015)*. IEEE Computer Society, 52–58.

[18] Subhash Khot and Igor Shinkar. 2016. An $\widetilde{O}(n)$ queries adaptive tester for unateness. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*. 37:1–37:7.