Lower Bounds for Testing Computability by Small Width OBDDs

Joshua Brody* and Kevin Matulef* and Chenggang Wu*

IIIS, Tsinghua University
[joshua.e.brody | matulef | wuchenggang0316] @ gmail.com

Abstract. We consider the problem of testing whether a function $f : \{0,1\}^n \to \{0,1\}$ is computable by a read-once, width-2 *ordered binary decision diagram* (OBDD), also known as a *branching program*. This problem has two variants: one where the variables must occur in a fixed, known order, and one where the variables are allowed to occur in an arbitrary order. We show that for both variants, any nonadaptive testing algorithm must make $\Omega(n)$ queries, and thus any adaptive testing algorithm must make $\Omega(\log n)$ queries.

We also consider the more general problem of testing computability by width-w OBDDs where the variables occur in a fixed order. We show that for any constant $w \geq 4$, $\Omega(n)$ queries are required, resolving a conjecture of Goldreich [15].

We prove all of our lower bounds using a new technique of Blais, Brody, and Matulef [6], giving simple reductions from known hard problems in communication complexity to the testing problems at hand. Our result for width-2 OBDDs provides the first example of the power of this technique for proving strong nonadaptive bounds.

1 Introduction

In this work we consider the problem of testing whether a function $f : \{0,1\}^n \to \{0,1\}$ is computable by a very limited type of computational device, a read-once *Ordered Binary Decision Diagram* (OBDD), also known as a *Branching Program*. We formalize this question in the language of *property testing*. The goal of a property tester is to distinguish objects which have a property from those that are "far" from having the property, with limited access to the object. Here, the object is a boolean function $f : \{0,1\}^n \to \{0,1\}$, and we would like a randomized algorithm that accepts f with high probability if it is computable by a read-once OBDD, and rejects f with high probability if it disagrees with any OBDD on an ϵ fraction of inputs. Property testing algorithms are *adaptive* if queries are chosen based on answers to previous queries; otherwise, testers are *non-adaptive*. The complexity of the algorithm is the number of times it queries f, which should hopefully be a small function of n and ϵ.

* Supported in part by the National Basic Research Program of China Grant 2007CB807900, 2007CB807901, and the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174.

Property testing, and in particular testing of boolean functions, has a long history. Over the last two decades researchers have studied algorithms for testing many different properties of functions, such as the property of being linear [7], being monotone [16, 13], being a dictator, a monomial [22] or a k-junta [12, 4, 5], or being expressible in various different "concise" forms such as an s-sparse polynomial, a size-s decision tree, etc. [9] (see, e.g., the survey of [24]).

The class of OBDDs has been studied in many areas of theoretical computer science, particularly in computational learning theory (e.g. [11, 23, 3, 14, 8]), where there is a well-known connection to testing. In particular, Goldreich et. al. [17] observed that any proper learning algorithm for a class of functions $\mathcal{C}$ can be used to test whether f is in $\mathcal{C}$ versus far from $\mathcal{C}$. Thus, the complexity of a proper learning algorithm serves as an upper bound on the number of queries required to test. However, this bound is often weak, since for many interesting classes, the query complexity of testing is much smaller than the complexity of learning (for instance, the class of linear functions is testable with $O(1/\epsilon)$ queries [7], independent of n, even though any learning algorithm must see at least $\Omega(n)$ values of f). It is natural to ask whether this is also the case for OBDDs.

1.1 Results for Width-2 and Width-3 OBDDs

The problem of testing whether a function f is computable by an OBDD was first studied by Ron and Tsur in [25]. They point out that although previous testing results have looked at the problem of testing whether f has a simple *form*, it seems reasonable instead to fix a simple *model of computation*, and test whether f is computable within the model. They focused on the model of *width-2* read-once OBDDs, because this class has a simple structure, yet still generalizes some previously well-studied classes, such as linear functions and monomials.

Ron and Tsur identified two variants of this problem. In the first, the tester wishes to determine whether f is computable by an OBDD where the variables must appear in a fixed, known order (say, $x_1 \dots x_n$). In the second, studied in [26], the tester wishes to determine whether f is computable by an OBDD where the variables can appear in any arbitrary order. Note that the complexity of the former problem is not a priori related to the complexity of the latter, since a function can be far from an OBDD with variables in order $x_1 \dots x_n$, but still be equal to an OBDD when the variables are rearranged.[1]

For the first variant of the width-2 OBDD testing problem, where the variables must occur in a fixed order, Ron and Tsur gave an adaptive upper bound of $\tilde{O}(\log n) \cdot \mathrm{poly}(1/\epsilon)$ queries [25] (here the $\tilde{\mathrm{O}}$ notation hides factors of $\log \log n$). This is an exponential improvement over the $\Omega(n)$ queries required for learning the same class.[2] Their upper bound raises the obvious question of whether the

[1] In general, if $\mathcal{C}'$ is a subset of $\mathcal{C}$, the complexity of testing membership in $\mathcal{C}'$ may be quite different than the complexity of testing membership in $\mathcal{C}$. However, they are often the same when the classes are conceptually related.

[2] The learning bound is easy to show, since the class of width-2 read-once OBDDs contains, for instance, all linear functions- a set of 2^n functions which are all far from each other.

$\tilde{O}(\log n)$ dependence on n is necessary, and indeed, whether any dependence on n is necessary at all. Our first result provides an answer to this question.

Theorem 1. *Any nonadaptive testing algorithm requires $\Omega(n)$ queries (and thus any adaptive testing algorithm requires $\Omega(\log n)$ queries[3]) to test*

i. width-2 OBDDs with variables in a fixed order.
ii. width-3 OBDDs with variables in a fixed order.

For width-2 OBDDs, Theorem 1 is essentially tight in terms of n. As mentioned, for adaptive algorithms Ron and Tsur gave an upper bound of $\tilde{O}(\log n)$, and for nonadaptive algorithms, a simple Occam learning algorithm yields an upper bound of $O(n)$.[4]

A few words are in order regarding the history of Theorem 1. In [25], Ron and Tsur gave a lower bound of $\Omega(\log n)$ for testing fixed-order width-2 OBDDs with one-sided error. However, this proof was recently found to contain a flaw, which we discuss in Appendix B. In subsequent work [28], the same authors gave a different proof which improved the bound to $\Omega(n)$ for nonadaptive testers, even with two-sided error, thus achieving the same bound as in Theorem 1. The "NO" instances (functions that are far from width-2 OBDDs) we use in our proof are partially inspired by [28]. However, our proof technique uses a different, more modular approach, as we shall discuss below.

For the problem of testing width-2 OBDDs when the variable order is unknown, the lower bound of Ron and Tsur does not apply. Tackling this case was posed as an open question in [28]. Our technique is able to handle this case as well, as shown in our second main result.

Theorem 2. *Any two-sided error, nonadaptive algorithm for testing computability by width-2 OBDDs with variables in arbitrary order requires $\Omega(n)$ queries (and thus, any adaptive algorithm requires $\Omega(\log n)$ queries).*

Remark: An earlier version of this paper claimed that the lower bound in Theorem 2 was essentially optimal, since it matched an upper bound given by Ron and Tsur in [26]. However, after reading a draft of this paper, Ron and Tsur discovered a flaw in their upper bound in [26], and showed that in the arbitrary-order case, Theorem 2 can be strengthened to $\Omega(n)$, even for adaptive algorithms [27]. Their improvement is nearly optimal, since in this case one can test using a simple Occam learning algorithm which makes $O(n \log n)$ queries.

[3] It is well known, and simple to show, that any adaptive testing algorithm which makes q queries can be transformed into a nonadaptive algorithm which makes 2^q queries. Hence a nonadaptive bound of $\Omega(g(n))$ implies an adaptive bound of $\Omega(\log g(n))$. As our result shows, this exponential gap is sometimes necessary.

[4] An Occam learning algorithm for a class $\mathcal{C}$ is simply a nonadaptive algorithm that draws a set of random examples, and searches for a member of $\mathcal{C}$ consistent with f evaluated on those examples. It is well known that such an algorithm will produce an ϵ-accurate hypothesis with constant probability after seeing $O(\log |\mathcal{C}|/\epsilon)$ examples.

1.2 Results for Width-w OBDDs, for constant $w \geq 4$

We also consider the problem of testing computability by width-w read-once OBDDs, for constant $w \geq 4$, where the variables must appear in a fixed order. This problem was previously studied by Goldreich in the $w = 4$ case [15]. He showed that unlike the width-2 case, where testing can be done with exponentially fewer queries than learning, testing width-4 OBDDs requires $\Omega(\sqrt{n})$ queries. He conjectured that the true bound is $\Omega(n)$. In this case, the complexity of testing would be essentially the same as the complexity of learning, since for any constant w, a simple Occam learning algorithm implies an $O(n)$ upper bound. Our final result confirms Goldreich's conjecture.

Theorem 3. *For any constant $w \geq 4$, any adaptive algorithm for testing computability by width-w OBDDs with variables in fixed order requires $\Omega(n)$ queries.*

1.3 Techniques

All of our lower bounds are proven using a new technique developed by the first two authors, along with Blais [6]. The result in [6] shows to reduce communication problems to testing problems, and thus leverage known lower bounds in communication complexity to prove lower bounds in testing.

Traditionally, property testing lower bounds are proven using Yao's Minimax Lemma. One starts by designing two families of objects- one family of YES instances (objects which have the property), and another of NO instances (objects which are far). In the next step, one defines a distribution over each family, and then one must show that no deterministic algorithm can distinguish with high probability whether a function is drawn from the YES or NO family. This step often involves nontrivial technical analysis. The philosophy put forth in [6] is that one can often eliminate the work required in this step, by creating YES and NO instances which correspond to the YES and NO instances of a communication problem. If strong lower bounds for the communication problem are already known, no new technical analysis is required.

Roughly speaking, the technique given in [6] translates *one-way* communication lower bounds to *nonadaptive* testing bounds, and *two-way* communication lower bounds to *adaptive* testing bounds. This was observed in [6], though no examples were given there of nonadaptive bounds that did not also apply to the adaptive case. For testing width-2 OBDDs in the fixed-order case, the nonadaptive complexity is much higher than the adaptive complexity. Thus, to prove Theorem 1, we must reduce from a communication problem that is hard when one-way communication is allowed, but easier when two-way communication is allowed. Our solution is to use an asymmetric communication problem for which this is known to be the case. Namely, we use the AUGMENTED-INDEX problem, a variant of the well-known INDEX problem. These problems have long been useful for proving streaming lower bounds [19, 21, 10], and now find a use in property testing as well. (We note that our proof of Theorem 2 has a similar flavor, and features a reduction from the INDEX problem. However, as we remarked earlier,

Ron and Tsur have subsequently improved this to an adaptive lower bound of $\Omega(n)$, via a reduction from the symmetric SET-DISJOINTNESS problem.)

2 Preliminaries

For two boolean functions $f, g : \{0,1\}^n \to \{0,1\}$, the *distance* between f and g, denoted $d(f,g)$ is equal to $\Pr_x[f(x) \neq g(x)]$.

We will work in the standard property testing model. Let $\mathcal{P}$ denote a property (a subset) of boolean functions. Then a *tester* for $\mathcal{P}$ is a randomized algorithm which when given query access to a function $f : \{0,1\}^n \to \{0,1\}$ and a distance parameter ϵ, outputs "accept" with probability at least $2/3$ if $f \in \mathcal{P}$, and "reject" with probability at least $2/3$ if $d(f,g) > \epsilon$ for all $g \in \mathcal{P}$. A *one-sided* tester is a tester which outputs "accept" with probability 1 if $f \in \mathcal{P}$. A *nonadaptive* tester is a tester which chooses its entire query set at the start of the algorithm.

We use $Q(\mathcal{P})$ to denote the minimum query complexity of a (possibly adaptive) tester for $\mathcal{P}$, and $Q^{\text{na}}(\mathcal{P})$ to denote the minimum query complexity of a nonadaptive tester for $\mathcal{P}$.

A two-party *communication problem* is defined by a function $C : A \times B \to \{0,1\}$. Alice has an input $a \in A$ and Bob has an input $b \in B$, and they would like to compute the value of $C(a,b)$. We work in the *public randomness* model, where Alice and Bob generate messages based on random bits they both see.

We use $R(C)$ to denote the minimum number of bits they must communicate for them both to compute $C(a,b)$ with probability at least $2/3$ on any input pair (a,b). We use $R^{\rightarrow}(C)$ to denote the *one-way* complexity- that is, the number of bits Alice must communicate for Bob to compute $C(a,b)$ with high probability (without sending any message to Alice).

For more details on communication complexity, consult the standard work of Kushilevitz and Nisan [20].

2.1 Reducing Communication Problems to Testing Problems

In this section, we give a sketch of the lower bound approach in [6]. We refer the reader to that paper for a more formal treatment.

Lemma 1. *Let C be a communication problem and $\mathcal{P}$ a property. Given functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^{n+1} \to \{0,1\}$, define the function $h = h(f,g) : \{0,1\}^n \to \{0,1\}$ as*

$$h(x) := g(f(x), x)$$

Suppose there is a way for Alice and Bob to create functions f_a and g_b based on their inputs such that (i) $h(f_a, g_b) \in \mathcal{P}$ if $C(a,b) = 1$, and (ii) $h(f_a, g_b)$ is $\Omega(1)$-far from $\mathcal{P}$ if $C(a,b) = 0$. Then,

1. *$R^{\rightarrow}(C) \leq Q^{\text{na}}(\mathcal{P})$, and*
2. *$R(C) \leq 2Q(\mathcal{P})$.*

Remark: In most reductions, h is defined as $h(x) := f_a(x) \oplus g_b(x)$ for some g_b independent of $f(x)$; however, this need not always be the case. In Section 4, we crucially rely on the asymmetry of the general construction.

Proof. (sketch) For the latter case, suppose there is an adaptive testing algorithm A for $\mathcal{P}$. Alice and Bob can cooperatively run this algorithm on h, using shared randomness to decide which values of h to query. Every time A queries $h(x)$ on some x, Alice sends $f(x)$ to Bob, who then computes $h(x)$ and returns it to Alice. In this way, both players maintain the list of query results $\{h(x)\}$ and can therefore generate successive values to query. If A is indeed a testing algorithm for $\mathcal{P}$, then at the end of the protocol they will know the value of $C(a, b)$ with high probability; the number of bits exchanged is twice the number of queries made by A.

The former case is similar. The only difference is that since queries are generated nonadaptively, Alice can send Bob $\{f(x)\}$ in a single message. Thus, the communication cost equals the query complexity of the testing algorithm.

2.2 Communication Complexity Problems

To prove our lower bounds, we give reductions from some standard communication problems. First, the SET-DISJOINTNESS problem:

Set-Disjointness. Alice and Bob are given n-bit inputs a and b and must compute

$$\text{DISJ}(a, b) := \bigvee_{i=1}^{n} a_i \wedge b_i$$

It is well-known that the two-way communication complexity, $R(\text{DISJ})$, is $\Omega(n)$, even with the promise that $a_i \wedge b_i = 1$ for at most one i [18].

For our width-2 OBDD bounds, where the nonadaptive complexity is larger than the adaptive complexity, it is necessary for us to reduce from communication problems where the one-way complexity (where Alice is allowed to send to Bob, but not vice-versa) is larger than the two-way complexity. We use the following two problems:

Index. Alice has an n-bit input a, and Bob has a $\log n$-bit index $i \in [n]$ (throughout, we will use the shorthand $[n]$ to denote the set $\{1, \ldots, n\}$). Bob's goal is to compute

$$\text{INDEX}(a, i) = a_i$$

It is well-known that the one-way complexity, $R^{\rightarrow}(\text{INDEX})$, is $\Omega(n)$ [1].

Augmented-Index. AUGMENTED-INDEX is nearly identical to INDEX, but Bob is also allowed to see bits $a_1 \ldots a_{i-1}$ without incurring any communication cost. Even with this additional "free" information, the one-way communication complexity, $R^{\rightarrow}(\text{AUGMENTED-INDEX})$, remains $\Omega(n)$ [2].

2.3 Branching Program Basics

A *binary decision diagram* (BDD), or *branching program*, is a layered directed acyclic graph with a distinguished source vertex and two sink vertices, labeled 0 and 1. Each internal vertex is labeled with an input variable and has outgoing edges to vertices in the next layer of the BDD. These edges are labeled with possible outcomes for the variable. In an *ordered binary decision diagram* (OBDD), all vertices in a layer are labeled with the same input variable. The *width* of an OBDD is the maximum number of vertices in any layer. An OBDD is *read-once* if vertices in different layers are labeled by different variables. In this work, we are only concerned with read-once OBDDs, and so we will often drop the "read-once" modifier.

We will use some basic definitions and claims developed by Ron and Tsur. The following can be found in [25]. The proofs of the facts are left as exercises.

Definition 1. *A function $f : \{0,1\}^n \to \{0,1\}$ is a* linear *function of $x_1 \dots x_n$ if it can be written in the form $f(x) = b_0 + \sum_{i=1}^{n} b_i x_i$ where $b_0 \dots b_n \in \{0,1\}$.*[5]

Fact 4 *A function $f : \{0,1\}^n \to \{0,1\}$ is computable by a width-2 OBDD with variables in* fixed *order $x_1 \dots x_n$ if and only if it can be written as*

$$f(x) = f_n(x_n, f_{n-1}(x_{n-1}, \dots f_2(x_2, f_1(x_1))))$$

where f_1 is a boolean function on one bit and $f_2, \dots, f_n$ are boolean functions on two bits. f is computable by a width-2 OBDD with variables in arbitrary *order if and only if it can be written in the same form after some permutation $\pi \in S_n$ is applied to the variables.*

We will slightly abuse notation, and use f_i both to refer to a function on 2 variables (the i'th variable and f_{i-1}), as well as a function on the first i variables.

Definition 2. *Consider a function $f : \{0,1\}^n \to \{0,1\}$ expressed in the form given in Fact 4. We say that a level i is* relevant *if the function f_i depends on the value of x_i. Relevant levels can be either* linear *or* blocking. *A relevant level is a* linear *level if f_i is a linear function of x_i and f_{i-1}. Otherwise, it is* blocking.

Fact 5 *If i is a blocking level, then f_i is either the* AND *or* OR *of $a \in \{x_i, \overline{x_i}\}$ and $b \in \{f_{i-1}, \overline{f_{i-1}}\}$. Thus, if i is blocking level, there exists a setting $t \in \{0,1\}$ for x_i such that f_i is constant, regardless of the value of f_{i-1}.*

Definition 3. *Let $f : \{0,1\}^n \to \{0,1\}$. We define the* influence *of variable i in f as $\mathrm{Inf}_f(i) := \Pr_x[f(x) \neq f(x^{\oplus i})]$, where $x^{\oplus i}$ denotes x with the i'th bit flipped.*

Lemma 2. *Let M be a width-2 OBDD with variables in order $x_1 \dots x_n$. Let $j < i$ and suppose level i is a blocking level. Then $\mathrm{Inf}_{f_i}(j) \leq \frac{1}{2}\mathrm{Inf}_{f_{i-1}}(j)$.*

Fact 6 *Let $f, g : \{0,1\}^n \to \{0,1\}$ and suppose there is a variable x_i such that $|\mathrm{Inf}_f(i) - \mathrm{Inf}_g(i)| = \tau$. Then $d(f,g) \geq \tau/2$.*

[5] In this definition and throughout the paper, addition is taken to be over $GF(2)$.

3 A Lower Bound for Testing Fixed-Order Width-2 and Width-3 OBDDs

In this section we prove Theorem 1. The proof will be via a reduction from AUGMENTED-INDEX. The main idea is that Alice will use her input to form a linear function, and Bob will use his index (plus extra knowledge) to form a linear function plus an AND of two consecutive variables. They then take the XOR of their two functions. If $a_i = 1$, then in the resulting function the AND will appear before the linear part, so it will be computable by a width-2 OBDD. However if $a_i = 0$, then in the resulting function the AND will appear after a variable in the linear part, so it will be far from any width-2 or width-3 OBDD (where the variables must appear in a fixed order). To show that our "no" instances are far, we make use of the following lemma.

Lemma 3. *Let $h : \{0,1\}^n \to \{0,1\}$ be a function of the form $h(x) = x_i + (x_{i+1} \wedge x_{i+2}) + \sum_{k \in S} x_k$ for some $i \in [n-2]$ and $S \subseteq \{i+3, \ldots, n\}$. Then,*

i. h is $1/4$-far from any width-2 OBDD, with variables in fixed order $x_1 \cdots x_n$.
ii. h is $1/8$-far from any width-3 OBDD, with variables in fixed order $x_1 \cdots x_n$.

In the case of width-2 OBDDs, Lemma 3 essentially appears as Claim 6 in [25]. We leave the proof of Lemma 3 to Appendix A. We are now ready to prove Theorem 1.

Theorem 1 (Restated). *Any nonadaptive testing algorithm requires $\Omega(n)$ queries (and thus any adaptive algorithm requires $\Omega(\log n)$ queries) to test*

i. width-2 OBDDs with variables in a fixed order.
ii. width-3 OBDDs with variables in a fixed order.

Proof. Let $n' = \lfloor (n-3)/4 \rfloor$. We will show a reduction from AUGMENTED-INDEX on n' variables. Since the one-way communication complexity of AUGMENTED-INDEX is $\Omega(n')$, and n' is linear in n, this will imply an $\Omega(n)$ testing bound.

First, Alice uses her input $a \in \{0,1\}^{n'}$ to form the function f, and Bob uses his input $i \in [n']$ (plus knowledge of $a_1 \ldots a_{i-1}$) to form the function g as follows

$$f(x) := \sum_{k=1}^{n'} x_{4k+3a_k} \qquad \text{and} \qquad g(x) := (\sum_{k=1}^{i-1} x_{4k+3a_k}) + (x_{4i+1} \wedge x_{4i+2})$$

Bob can then solve AUGMENTED-INDEX by running a testing algorithm on the joint function $h(x) = f(x) + g(x)$ and having Alice send him the value of $f(x)$ whenever he needs to query h. It is easy to see that if $a_i = 1$, then $h(x) = (x_{4i+1} \wedge x_{4i+2}) + x_{4i+3} + \sum_{k=i+1}^{n'} x_{4k+3a_k}$, so h is a width-2 OBDD. On the other hand, if $a_i = 0$ then $h(x) = x_{4i} + (x_{4i+1} \wedge x_{4i+2}) + (\sum_{k=i+1}^{n'} x_{4k+3a_k})$, so by Lemma 3, h is far from any width-2 or width-3 OBDD.

4 A Lower Bound for Testing Arbitrary-Order Width-2 OBDDs

In this section we prove Theorem 2. The proof will be via a reduction from INDEX. The main idea is that Alice will use her inputs to form a width-2 OBDD on pairs of variables corresponding to the indices where $a_k = 1$. Then Bob will use his index i to append two more variables to the end of the OBDD. If $a_i = 0$, then no variable will be used more than once, so the resulting function will remain a width-2 OBDD. If $a_i = 1$, then two variables will be used twice, which will cause the resulting function to be far from any (read-once) width-2 OBDD.

Theorem 2 (Restated). *Any two-sided error, nonadaptive algorithm for testing computability by width-2 OBDDs with variables in arbitrary order requires $\Omega(n)$ queries (and thus, any adaptive algorithm requires $\Omega(\log n)$ queries).*

Proof. Let $n' = \lfloor (n-1)/2 \rfloor$. We will show a reduction from INDEX on n' variables. Since the one-way communication complexity of INDEX is $\Omega(n')$, and n' is linear in n, this will imply an $\Omega(n)$ testing bound.

First, Alice uses her input $a \in \{0,1\}^{n'}$ to form the function

$$f(x) := x_1 + \sum_{k=1}^{n'} a_k(x_{2k} + x_{2k+1})$$

Then Bob uses his index $i \in [n']$ to form the combined function

$$h(x) := (f(x) \wedge x_{2i}) + x_{2i+1}$$

We claim that Bob can solve INDEX by running a testing algorithm on h. To see this, first note that f is just a linear function on some subset of variables. If $a_i = 0$, then the variables x_{2i} and x_{2i+1} do not appear in f, so the resulting h is clearly a read-once width-2 OBDD.

If $a_i = 1$, then the variables x_{2i} and x_{2i+1} do appear in f. In this case, we can write $f(x) = f'(x) + x_{2i} + x_{2i+1}$, where $f'(x)$ is a linear function on some non-empty subset of variables not involving x_{2i} or x_{2i+1} (note that the variable x_1 is included in f just to guarantee that f' is always a linear function on at least one variable). We can thus express the resulting h as $h(x) = ((f'(x) + x_{2i} + x_{2i+1}) \wedge x_{2i}) + x_{2i+1}$, which simplifies to the following

$$h(x) = \begin{cases} x_{2i+1} & \text{if } x_{2i} = 0 \\ f'(x) + 1 & \text{if } x_{2i} = 1 \end{cases}$$

We claim that h is 1/8-far from a read-once, width-2 OBDD. To see this, first note that any variable x_k relevant to h has $\mathrm{Inf}_h(k) = 1/2$. This is easily checked, since (i) x_{2i+1} is influential if and only if $x_{2i} = 0$, (ii) the variables relevant to f' are influential if and only if $x_{2i} = 1$, and (iii) x_{2i} is influential if and only if $x_{2i+1} \neq f'(x) + 1$, which happens exactly half the time (as x_{2i+1} and $f'(x)$ are linear functions on disjoint subsets of variables).

Assume for contradiction that h is 1/8-close to some width-2 OBDD ℓ. Without loss of generality, we can assume that ℓ's irrelevant variables come at the beginning. This means that the last level of ℓ must be a blocking level. Otherwise, if it is a linear level, the variable at that level will have influence 1 in ℓ, but only influence 1/2 or 0 in h, so by Fact 6, h and ℓ will be 1/4-far.

Since the last level of ℓ is blocking, there must exist a setting $t \in \{0,1\}$ for the variable at the last level which makes the function ℓ constant. However, for any fixed setting of any single variable in h, the resulting function is at least 1/4-far from constant. This is easily checked, since (i) fixing the value of x_{2i} induces either x_{2i+1} or $f'(x)+1$, both of which are 1/2-far from constant, (ii) fixing the value of x_{2i+1} still allows h to be balanced when $x_{2i}=1$, so in this case h is 1/4-far from constant, and (iii) fixing the value of a variable which appears in f' still allows h to be balanced when $x_{2i}=0$, so in this case h is also 1/4-far from constant. Since h and ℓ disagree on at least 1/4 of the settings where ℓ's last variable is set to t, this implies they must be 1/8-far.

5 A Lower Bound for Testing Fixed-Order Width-w OBDDs, for constant $w \geq 4$.

In this section we prove Theorem 3, via a reduction from SET-DISJOINTNESS. To perform the reduction, Alice and Bob will use the elements in their sets to produce a set of AND clauses. They then run a testing algorithm on the XOR of these clauses. Crucially, they will construct the clauses in such a way that when their sets intersect, they produce at least $k := \lfloor \log_2 w \rfloor$ clauses with *interleaving* variables. This large number of interleaving variables forces the hard instances to be far from computable by (fixed-order) width-w OBDDs.

In order to show that our hard instances are far, we make use of the following technical lemma, which we prove in Appendix A. The lemma is a generalization of the $w=4$ case proved in Theorem 4.2 of Goldreich [15].

Lemma 4. *Let $k := \lfloor \log_2 w \rfloor$ and $n' := \lfloor n/2k \rfloor - 1$. For boolean variables $x_1, \ldots, x_{2k}$, define predicates $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ as*

$$\sigma_1(x_1,\ldots,x_{2k}) := 0, \quad \sigma_2(x_1,\ldots,x_{2k}) := (x_1 \wedge x_{k+1}) + \ldots + (x_{k-1} \wedge x_{2k-1})$$

$$\sigma_3(x_1,\ldots,x_{2k}) := x_k \wedge x_{2k}, \quad \sigma_4(x_1,\ldots,x_{2k}) := (x_1 \wedge x_{k+1}) + \ldots + (x_k \wedge x_{2k})$$

and for $v_1, \ldots, v_{n'} \in \{1,2,3,4\}$, define a boolean function $h = h_{v_1,\ldots,v_{n'}}$ as

$$h(x) := x_1 + \sum_{i=1}^{n'} \sigma_{v_i}(x_{2ki+1}, \ldots, x_{2ki+2k})$$

i. If $v_i \in \{1,2,3\}$ for all i, then h is computable by a width-w OBDD.
ii. If $v_j = 4$ for a unique j, then h is $\frac{1}{2w^2}$-far from any width-w OBDD.

Theorem 3 (Restated). *For any constant $w \geq 4$, any adaptive algorithm for testing computability by width-w OBDDs with variables in fixed order requires $\Omega(n)$ queries.*

Proof. Let $k := \lfloor \log w \rfloor$ and $n' := \lfloor n/2k \rfloor - 1$. We will reduce from SET-DISJOINTNESS (with the promise that Alice and Bob's sets intersect in at most one place) on n' variables. Since n' is $\Theta(n)$, this will imply an $\Omega(n)$ lower bound.

Let $S, T \subseteq [n']$ denote Alice's and Bob's inputs respectively. Then Alice constructs the function f and Bob constructs the function g as follows

$$f(x) := \sum_{i \in S} (x_{2ki+1} \wedge x_{2ki+k+1}) + \ldots + (x_{2ki+k-1} \wedge x_{2ki+2k-1})$$

$$g(x) := \sum_{i \in T} (x_{2ki+k} \wedge x_{2ki+2k})$$

We will show that they can solve SET-DISJOINTNESS by running a testing algorithm for width-w OBDDs on $h(x) := x_1 + f(x) + g(x)$. Note that each set of $2k$ consecutive variables $(x_{2ki+1}, x_{2ki+2}, \ldots, x_{2ki+2k})$ depends on a unique coordinate from $[n']$, and that h contains $\sigma_4(x_{2ki+1}, \ldots, x_{2ki+2k})$ if and only if $i \in S \cap T$. The theorem thus follows from Lemma 4.

Acknowledgments

We would like to thank Dana Ron and Gilad Tsur for helpful discussions, and for sharing their manuscript [28] and their improvement to our Theorem 2.

References

1. Farid Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theoretical Computer Science*, 175(2):139–159, 1996.
2. Z. Bar-Yossef, T. S. Jayram, R. Krauthgamer, and R. Kumar. The sketching complexity of pattern matching. In *8th International Workshop on Randomization and Computation (RANDOM)*, 2004.
3. Francesco Bergadano, Nader H Bshouty, Christino Tamon, and Stefano Varricchio. On learning branching programs and small depth circuits. In *Computational Learning Theory: Proc. Third European Conference. Lecture Notes in Articial Intelligence*, pages 150–161. Springer-Verlag, 1997.
4. Eric Blais. Improved bounds for testing juntas. In *Proc. 12th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 317–330, 2008.
5. Eric Blais. Testing juntas nearly optimally. In *Proc. 41st Annual ACM Symposium on the Theory of Computing*, pages 151–158, 2009.
6. Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. http://web.mit.edu/matulef/www/papers/PTviaCC.pdf, 2011.
7. Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47:549–595, 1993. Earlier version in STOC'90.
8. Nader H. Bshouty, Christino Tamon, and David K. Wilson. On learning width two branching programs. *Inf. Process. Lett.*, 65:217–222, February 1998.

9. Ilias Diakonikolas, Homin Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco Servedio, and Andrew Wan. Testing for concise representations. In *Proc. 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 549–558, 2007.
10. Khanh Do Ba, Piotr Indyk, Eric Price, and David P. Woodruff. Lower bounds for sparse recovery. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, 2010.
11. F Ergün, R Kumar, and R Rubenfeld. On learning boundedwidth branching programs. In *in Proc. 8th International Conference on Learning Theory*, pages 361–368, 1995.
12. Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *J. Comput. Syst. Sci.*, 68:753–787, 2004.
13. Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proc. 34th Annual ACM Symposium on the Theory of Computing*, pages 474–483, 2002.
14. Ricard Gavald'a and David Guijarro. Learning ordered binary decision diagrams. In *In 6th International Workshop on Algorithmic Learning Theory, Lecture Notes in Artificial Intelligence No. 997, Jantke, Shinohara, Zeugmann (Eds.* Springer-Verlag, 1995.
15. Oded Goldreich. On testing computability by small width OBDDs. In *Proc. 14th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 574–587, 2010.
16. Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.
17. Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.
18. Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, pages 211–219, 2007.
19. Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, 2010.
20. Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, Cambridge, 1997.
21. Frédéric Magniez, Claire Mathieu, and Ashwin Nayak. Recognizing well-parenthesized expressions in the streaming model. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, 2010.
22. Michal Parnas, Dana Ron, and Alex Samorodnitsky. Testing basic boolean formulae. *SIAM J. Disc. Math.*, 16(1):20–46, 2002.
23. Vijay Raghavan and Dawn Wilkins. Learning branching programs with queries. In *Proc. 6th Annual Workshop on Computational Learning Theory*, 1993.
24. Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5(2):73–205, 2009.
25. Dana Ron and Gilad Tsur. Testing computability by width two obdds. In *13th International Workshop on Randomization and Computation (RANDOM)*, 2009.
26. Dana Ron and Gilad Tsur. Testing computability by width-2 obdds where the variable order is unknown. In *7th International Conference on Algorithms and Complexity*, 2010.
27. Dana Ron and Gilad Tsur. Personal communication. 2011.
28. Dana Ron and Gilad Tsur. Testing computability by width-two obdds. (Journal version). Manuscript, 2011.

A Distance Lemmas

In this section we prove various lemmas to show that our hard instances are $\Omega(1)$-far from small-width OBDDs. We will often leverage the following version of Yao's XOR Lemma applied to OBDDs, due to Goldreich [15].

Lemma 5 ([15], Lemma A.6). *Fix $0 \leq p_1, p_2 < 1$ and a width w. If f_1 is p_1-far from computable by width-w OBDDs and f_2 is p_2-far from computable by width-w OBDDs, then the function $f := f_1 \oplus f_2$ is at least $(p_1 + p_2 - 2p_1p_2)$-far from computable by width-w OBDDs.*

First, we prove Lemma 3. We note that the first part of the lemma was proved in [25], using a different method.

Lemma 3 (Restated). *Let $h : \{0,1\}^n \to \{0,1\}$ be a function of the form $h(x) = x_i + (x_{i+1} \wedge x_{i+2}) + \sum_{k \in S} x_k$ for some $i \in [n-2]$ and $S \subseteq \{i+3, \ldots, n\}$. Then,*

i. h is $1/4$-far from any width-2 OBDD, with variables in fixed order $x_1 \cdots x_n$.
ii. h is $1/8$-far from any width-3 OBDD, with variables in fixed order $x_1 \cdots x_n$.

Proof. Decompose h into boolean functions f, g, where $f : \{0,1\}^3 \to \{0,1\}$ is defined as $f(x) = x_1 + (x_2 \wedge x_3)$, and $g : \{0,1\}^{n-3} \to \{0,1\}$ is defined as $g(x) := \sum_{k \in S} x_k$.

First, we claim that f is $1/4$ far from computable by width-2 OBDDs and $1/8$-far from computable by width-3 OBDDs. To see this, consider polynomials of the form $\phi_a(x_3) := a_1 + a_2 x_3$ for some $a = (a_1, a_2) \in \{0,1\}^2$. Note that any distinct $a, \hat{a}$ yield distinct polynomials $\phi_a, \phi_{\hat{a}}$, hence there must be some setting of x_3 where $\phi_a(x_3) \neq \phi_{\hat{a}}(x_3)$. Therefore, any time an OBDD maps $a, \hat{a}$ to the same state at layer 3, there must be a mistake for some setting of x_3 and for one of $\{a, \hat{a}\}$.

There are four values for a. Hence, we must err twice in a width-2 OBDD and once in a width-3 OBDD. With eight possible values for the input $x \in \{0,1\}^3$, it follows that f is $1/4$-far from width-2 OBDDs and $1/8$-far from width-3 OBDDs. The rest of the proof follows from Lemma 5.

Next, we prove Lemma 4. This is a generalization of a similar proof for the width $w = 4$ case, given in Theorem 4.2 in [15].

Lemma 4 (Restated). *Let $k := \lfloor \log w \rfloor$ and $n' := \lfloor n/2k \rfloor - 1$. For boolean variables $x_1, \ldots, x_{2k}$, define predicates $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ as*

$$\sigma_1(x_1, x_2, \ldots, x_{2k}) := 0\,, \quad \sigma_2(x_1, x_2, \ldots, x_{2k}) := (x_1 \wedge x_{k+1}) + \ldots + (x_{k-1} \wedge x_{2k-1})$$

$$\sigma_3(x_1, \ldots, x_{2k}) := x_k \wedge x_{2k}\,, \quad \sigma_4(x_1, \ldots, x_{2k}) := (x_1 \wedge x_{k+1}) + \ldots + (x_k \wedge x_{2k})$$

Finally, for $v_1, \ldots, v_{n'} \in \{1,2,3,4\}$, define a boolean function $h = h_{v_1,\ldots,v_{n'}}$ as

$$h(x) := x_1 + \sum_{i=1}^{n'} \sigma_{v_i}(x_{2ki+1}, \ldots, x_{2ki+2k})$$

i. If $v_i \in \{1,2,3\}$ for all i, then h is computable by a width-w OBDD.
ii. If $v_j = 4$ for a unique j, then h is $\frac{1}{2w^2}$-far from any width-w OBDD.

Proof. For convenience, we call $\sigma_{v_i}(x_{2ki+1}, \ldots, x_{2ki+2k})$ the ith block of h.

For the first case, when $v_i \in \{1,2,3\}$ for all i, it is easy to construct a width-w OBDD computing h. We can use the $w \geq 2^k$ vertices in each layer to maintain k bits of information—one bit maintains the value of the function computed up to the ith block, and the other bits maintain either $x_{2ki+1} \ldots x_{2ki+k-1}$, or x_{2ki+k}, depending on which AND clause is active in σ_{v_i}.

For the second case, let j be the unique index such that $v_j = 4$. Then we can write h as a sum $h := h_1 + h_2$ where $h_1(x) := x_1 + \sigma_4(x_{2kj+1}, \ldots, x_{2kj+2k})$ and $h_2(x) := \sum_{i:i\neq j} \sigma_{v_i}(x_{2ki+1}, \ldots, x_{2ki+2k})$. Note that h_2 is computable by a width-w OBDD. However, by Lemma 5 (with $p_2 = 0$), it suffices to show that h_1 is $\frac{1}{2w^2}$-far from any width-w OBDD. This is accomplished via Lemma 6 below. When $k := \lfloor \log w \rfloor$, Lemma 6 implies that h_1 is at least $\frac{2^{k+1}-w}{2^{2k+1}} \geq \frac{1}{2w^2}$-far from any width-$w$ OBDD.

Lemma 6. *Fix k, w such that $2^{k+1} > w$, and define a boolean function $f : \{0,1\}^{2k+1} \to \{0,1\}$ such that*

$$f(x) := x_0 + \sum_{j=1}^{k} x_j \wedge x_{k+j} \ .$$

Then f is $(2^{k+1} - w)/2^{2k+1}$-far from computable by width-w OBDDs.

Proof. For any $a \in \{0,1\}^{k+1}$, define the multivariate polynomial $\phi_a(z) := a_0 + \sum_{j=1}^{k} a_j z_j$. Any two distinct $a \neq \hat{a}$ give distinct polynomials $\phi_a, \phi_{\hat{a}}$, hence there must be some z such that $\phi_a(z) \neq \phi_{\hat{a}}(z)$.

Fix any width-w OBDD ℓ, and consider how ℓ maps strings a to level $k+1$. It's easy to see that for any two $a \neq \hat{a}$ that get mapped to the same state at level $k+1$, there exists z such that ℓ makes a mistake. Since ℓ has at most w states at each level, it must make at least $2^{k+1} - w$ mistakes (as this is the number of "collisions" at level $k+1$). f has 2^{2k+1} total inputs, so f is at least $(2^{k+1} - w)/2^{2k+1}$-far from computable by width-w OBDDs.

B Explanation of [25]'s One-Sided Lower Bound

In Section 3 of [25], the authors give a one-sided lower bound of $\Omega(\log n)$ queries for testing computability by width-2 OBDDs in the fixed-order case. However, this bound was found to be flawed. In this section we explain the error. We include this exposition for the benefit of the reader, since we believe it illustrates some of the subtleties involved with this problem.

We start with some basic terminology and facts from [25].

Definition 4 ([25], Definition 6). *A set S is an i-*Prefix Equivalence Class *for a function $f : \{0,1\}^n \to \{0,1\}$ when S is a* maximal *subset of $\{0,1\}^i$ such that for all $x, y \in S$ and $z \in \{0,1\}^{n-i}$, it holds that $f(xz) = f(yz)$.*

Fact 7 ([25], Corollary 1) *A function $f : \{0,1\}^n \to \{0,1\}$ is computable by a width-k OBDD if and only if $\forall i \in [n]$, f has at most k distinct i-prefix equivalence classes.*

The lower bound proceeds by constructing a family of $O(n)$ functions, indexed by $1 < j < (n-1)$, which are all 1/4-far from width-2 OBDDs. These functions are defined as $\gamma_j := x_1 + \cdots + (x_j \wedge x_{j+1}) + \cdots + x_n$. It is easy to verify that for all prefixes except for the j'th, γ_j has 2 equivalence classes, but for the j'th prefix, γ_j has more than 2 equivalence classes.

The authors then claim that any one-sided algorithm for testing width-2 OBDDs implicitly corresponds to an exact learning algorithm for the family $\{\gamma_i\}$. They make this claim because any one-sided testing algorithm can only reject a function if the set of answers it receives to its queries are inconsistent with any width-2 OBDD. The property of being a width-2 OBDD is equivalent to having at most 2 i-prefix equivalence classes for all i, and each γ_j only has more than 2 equivalence classes for a single j. Thus, it seems reasonable to assume that a one-sided algorithm can only reject a particular γ_j if the set of answers it receives to its queries reveals *which* index j is the one with more than 2 equivalence classes.

Unfortunately, this assumption turns out to be false. In fact, it is possible to construct a partial function h which is consistent with two different γ_j's, even though it is inconsistent with any width-2 OBDD.

To construct such a function h, consider the functions $f(x) := x_1 + (x_2 \wedge x_3) + x_4$, and $g(x) := x_1 + x_2 + (x_3 \wedge x_4)$. Note that when $n = 4$, $f = \gamma_2$ and $g = \gamma_3$, and thus they are both far from fixed-order width-2 OBDDs. Next, we define the partial function h only on the 12 values of x where $f(x) = g(x)$.

One can easily verify that h only has 2 i-prefix equivalence classes for all i. This follows from the fact that h is consistent with both f and g, and there is no level i where both f and g have more than 2 equivalence classes.

On the other hand, one can also verify that h is inconsistent with any width-2 OBDD. Brute-force analysis shows that any attempt to extend h on the 4 undefined values of x will cause h to become incompatible with a width-2 OBDD. In particular, the only way to extend h so that it has 2 equivalence classes at level 3 is to make $h = f$, in which case h will be far from a width-2 OBDD.