An Adaptivity Hierarchy Theorem for Property Testing

Clément L. Canonne[1] and Tom Gur[2]

1 **Columbia University, New York, NY, USA**
`ccanonne@cs.columbia.edu`
2 **UC Berkeley, Berkeley, CA, USA**
`tom.gur@berkeley.edu`

Abstract

Adaptivity is known to play a crucial role in property testing. In particular, there exist properties for which there is an exponential gap between the power of *adaptive* testing algorithms, wherein each query may be determined by the answers received to prior queries, and their *non-adaptive* counterparts, in which all queries are independent of answers obtained from previous queries.

In this work, we investigate the role of adaptivity in property testing at a finer level. We first quantify the degree of adaptivity of a testing algorithm by considering the number of "rounds of adaptivity" it uses. More accurately, we say that a tester is k-(round) adaptive if it makes queries in $k+1$ rounds, where the queries in the i'th round may depend on the answers obtained in the previous $i-1$ rounds. Then, we ask the following question:

Does the power of testing algorithms smoothly grow with the number of rounds of adaptivity?

We provide a positive answer to the foregoing question by proving an adaptivity hierarchy theorem for property testing. Specifically, our main result shows that for every $n \in \mathbb{N}$ and $0 \leq k \leq n^{0.99}$ there exists a property $\mathcal{P}_{n,k}$ of functions for which (1) there exists a k-adaptive tester for $\mathcal{P}_{n,k}$ with query complexity $\tilde{O}(k)$, yet (2) any $(k-1)$-adaptive tester for $\mathcal{P}_{n,k}$ must make $\Omega(n)$ queries. In addition, we show that such a qualitative adaptivity hierarchy can be witnessed for testing natural properties of graphs.

1998 ACM Subject Classification F.1.3 [Computation by Abstract Devices] Complexity Measures and Classes

Keywords and phrases Property Testing, Coding Theory, Hierarchy Theorems

Digital Object Identifier 10.4230/LIPIcs.CCC.2017.27

1 Introduction

The study of property testing, initiated by Rubinfeld and Sudan [36] and Goldreich, Goldwasser and Ron [21], has attracted significant attention in the last two decades (see, e.g., recent books [18, 20, 5] and surveys [32, 33, 15]). Loosely speaking, property testers are highly efficient randomized algorithms (typically running in sublinear time) that solve approximate decision problems, while only inspecting a tiny fraction of their inputs. More accurately, an ε-tester $\mathcal{T}$ for property $\mathcal{P}$ is a randomized algorithm that, given query access to an input x, decides whether $x \in \mathcal{P}$ or x is ε-far (say, in Hamming distance) from $\mathcal{P}$. The query complexity of $\mathcal{T}$ is then the number of queries it makes to x.

In general, a testing algorithm may select its queries adaptively such that the i'th query is determined by the answers to the previous $i-1$ queries, in which case it is said to be an

32nd Computational Complexity Conference (CCC 2017).
Editor: Ryan O'Donnell; Article No. 27; pp. 27:1–27:25
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
COMPUTATIONAL COMPLEXITY CONFERENCE

adaptive tester. However, in many natural cases, testers may actually determine their queries solely based on their randomness (and input length), without any dependency on answers to previous queries; a tester that satisfies this condition is called a *non-adaptive tester*. A natural question, which commonly arises in query-based models, is whether the ability to make adaptive queries can significantly affect the query complexity.

Adaptive queries can be easily emulated at the cost of a large blowup in query complexity (exponential in the number of queries). More accurately, any q-query adaptive tester for a property of objects represented by functions $f\colon D \to R$ can be emulated by an $|R|^q$-query non-adaptive tester (see e.g., [20, Section 1.5]). While for certain types of properties and models – e.g., linear properties [4] and properties in the dense graph model [25] – one has better emulations which come with little or no overhead, such efficient emulations cannot exist for all properties. As was shown by Raskhodnikova and Smith [31], in the bounded-degree graph model [23] there is a large chasm between the adaptive and non-adaptive query complexities of testing many natural graph properties. In particular, any property over bounded-degree graphs with n vertices, which is not determined by the *vertex degree distribution*,[1] requires $\Omega(\sqrt{n})$ queries to test non-adaptively, whereas many such properties (e.g., triangle-freeness and connectivity) have ε-testers with query complexity $\mathrm{poly}(1/\varepsilon)$.

In this work, we investigate the role of adaptivity in property testing at a finer level. Rather than considering the extreme cases of fully adaptive testers versus completely non-adaptive testers, we consider testers with various levels of restricted adaptivity and ask the following question:

> *Can the power of testers gradually grow with the "amount" of adaptivity they are allowed to use?*

Besides the sheer theoretical interest of understanding the role of adaptivity in property testing, a motivation for this question comes from the *constraints* that come with adaptive algorithms, which may counterbalance the apparent gain in efficiency. Indeed, non-adaptive algorithms (or at least those which only use a small number of adaptive "stages") may be preferred in practice to their adaptive counterparts, in spite of the larger number of queries they make. The reason for this preference is the significant gains obtained by being able to make many queries *in parallel*: when each query is an experiment which, while relatively cheap by itself, may take several hours, assessing the trade-off between rounds of adaptivity and total number of queries becomes crucial. An archetypal example where such considerations prevail is the (different) setting of group testing (see e.g. [17, Section 1.2]).

To answer the foregoing question, we shall first need to give a precise definition for the "amount" of adaptivity that a tester uses. To this end, it is natural to consider the number of "rounds of adaptivity" used by a tester.[2] More precisely, we say that a tester is *k-round-adaptive* if it generates and makes queries in $k+1$ rounds, where in the i'th round the tester queries a set of locations Q_i that may depend on the answers to queries in $Q_0, \ldots, Q_{i-1}$, obtained in previous rounds. We will quantify the "amount" of adaptivity that a tester uses by the number of rounds of adaptivity that it uses. Equipped with the notion of round adaptivity, we can proceed to present our results.

[1] Loosely speaking, a property $\mathcal{P}$ of bounded-degree graphs is not determined by the vertex degree distribution if there exist two graphs, $G_1 \in \mathcal{P}$ and G_2 that is "far" from $\mathcal{P}$, such that the vertices of G_1 and G_2 have the same degrees.

[2] We also consider an alternative notion of *tail adaptivity*, which roughly speaking refers to testers that first make a large number of non-adaptive queries and subsequently make a bounded number of adaptive queries. See Section 3 for details regarding how these two notions relate.

1.1 Our Results

Our main result provides a positive answer to the foregoing question by showing an adaptivity hierarchy theorem for property testing; that is, we show a family of properties $\{\mathcal{P}_k\}_k$ such that for every k, the property $\mathcal{P}_k$ is "easy" for k-adaptive testers and "hard" for $(k-1)$-adaptive testers.

▶ **Theorem 1** (Informally stated (see Theorem 5)). *For every $n \in \mathbb{N}$ and $0 \leq k \leq n^{0.33}$ there is a property $\mathcal{P}_{n,k}$ of strings over $\mathbb{F}_n$ (of length that is nearly linear in n) such that:*
1. *there exists a k-round-adaptive tester for $\mathcal{P}_{n,k}$ with query complexity $\tilde{O}(k)$, yet*
2. *any $(k-1)$-round-adaptive tester for $\mathcal{P}_{n,k}$ must make $\tilde{\Omega}(n/k^2)$ queries.*

The above theorem relies on an arguably contrived family of properties, which was specifically tailored towards maximizing the separations; hence, one may wonder whether such strong separations also hold for more natural properties. As we show below, this is indeed the case: namely, we establish another adaptivity hierarchy theorem that, albeit weaker than Theorem 1, applies to the well-studied natural problem of testing k-cycle freeness in the bounded-degree graph model (see Section 5.1 for definitions).

▶ **Theorem 2.** *Let $k \in \mathbb{N}$ be a constant. Then,*
1. *there exists a k-round-adaptive tester with query complexity $O(1/\varepsilon)$ for $(2k+1)$-cycle freeness in the bounded-degree graph model; yet*
2. *any $(k-1)$-round-adaptive tester for $(2k+1)$-cycle freeness in the bounded-degree graph model must make $\Omega(\sqrt{n})$ queries, where n is the number of vertices in the graph.*

Lifting Query Complexity Bounds to Property Testing. Notably, the proof of Theorem 1 relies on a technique that allows us to "lift" *query complexity* bounds to property testing, *via* the use of error-correcting codes that admit a strong form of local testability as well as a relaxed form of local decodability. We believe that this framework, which we detail in Section 4.4, is of interest in its own right, and will find further applications in property testing.

We conclude this section by posing two open problems that naturally arise from our work.

▶ **Open Problem 1** (One property to rule them all). *Does there exist an adaptivity hierarchy with respect to a* single *property? That is, for any m and all sufficiently large n, is there a property $\mathcal{P}$ of elements of size n, and $q_1 > \ldots > q_m$ (m "levels" of hierarchy) such that for every $k \in [m]$ there exists a k-adaptive tester for $\mathcal{P}$ with query complexity q_k, yet every $(k-1)$-adaptive tester must make $\omega(q_k)$ queries to test $\mathcal{P}$?*

▶ **Open Problem 2** (Au naturel is just as good). *Does there exist a family of* natural *properties which exhibits an adaptivity hierarchy with separations as strong as in Theorem 1?*

1.2 Previous Work

As previously mentioned, the role of adaptivity in property testing has been the focus of several works before. It is well known that for any property of Boolean functions, there exists at most an exponential gap between adaptive and non-adaptive testers: any (adaptive) q-query testing algorithm for a property $\mathcal{P}$ of n-variate Boolean functions can be simulated by a non-adaptive tester with query complexity $2^q - 1$. Further, such gaps are known to exist for some natural properties, such as read-once width-2 OBDDs [35, 12] and signed majorities [28, 34] (importantly, there also exist cases where adaptivity is known *not* to help [11, 4]). Another prominent example of a class of Boolean functions where adaptivity is known to help is that of k-juntas [8, 7, 38, 16], which can be tested adaptively with $\tilde{O}(k)$ queries, yet for which the non-adaptive query complexity is $\tilde{\Theta}\left(k^{3/2}\right)$.

Of course, the Boolean function setting is not the only one: in the dense graph model, it is known that while adaptivity can help [24], it will be at most by a quadratic factor [2, 25]: that is, every graph property testable (adaptively) with q queries has an $O\left(q^2\right)$-query non-adaptive tester. This is no longer the case in the bounded-degree model, however; where Raskhodnikova and Smith showed that there exist many properties which can be tested adaptively with a constant number of queries, but for which any non-adaptive tester must have query complexity $\Omega(\sqrt{n})$ [31].[3]

However, all these results, even when they establish cases where adaptivity does help, leave open the question of *how much* adaptivity is needed for this to happen. In particular, for the case of properties of Boolean functions, many known adaptive testers which outperforms their non-adaptive counterpart do so, at some level, by conducting a binary search of some sort (see, e.g., [8, 35, 34]) and thus comes inherently with a logarithmic numbers of "adaptive rounds."

Our proof of Theorem 1 relies on a connection between the property testing and linear decision tree models. Although many of the ingredients we use are new, the connection itself is not and was first observed in [39] (see also [6] for a slightly different connection between property testing and parity decision trees).

Adaptivity in other settings. We remark that the notion of round complexity in communication complexity and interactive proof systems is somewhat analogous to that of round adaptivity, since in those models each round of communication or interaction allows the parties to adapt their strategies. Moreover, a round complexity hierarchy is known for communication complexity [29] and interactive proofs of proximity [26]. Finally, we also mention that the role of the number of adaptive measurements used by sparse recovery algorithms was shown to be very significant [27].

Organization

In Section 2 we provide the preliminaries required for the technical sections. In Section 3 we provide a precise definition for testers with bounded adaptivity. In Section 4 we prove our main result, which is a strong adaptivity hierarchy theorem for a property of functions. In Section 5 we prove an adaptivity hierarchy theorem with respect to a natural property of graphs. Finally, in Section 6 we discuss adaptivity round reductions, as well as a connection to communication complexity, and the relation between round and tail adaptivity.

2 Preliminaries

We begin with standard notations:

- We denote the *relative Hamming distance*, over alphabet Σ, between two vectors $x \in \Sigma^n$ and $y \in \Sigma^n$ by $\operatorname{dist}(x,y) \stackrel{\text{def}}{=} |\{x_i \neq y_i \,:\, i \in [n]\}|/n$. If $\operatorname{dist}(x,y) \leq \varepsilon$, we say that x is ε-close to y, and otherwise we say that x is ε-far from y. Similarly, we denote the

[3] We remark that in the bounded-degree model, algorithms typically rely on random walks or breadth-first searches. In this case the depth of the walks or searches, which may be smaller than the query complexity, tends to determine the adaptivity.

relative distance of x from a non-empty set $S \subseteq \Sigma^n$ by $\mathrm{dist}(x, S) \stackrel{\text{def}}{=} \min_{y \in S} \mathrm{dist}(x, y)$. If $\mathrm{dist}(x, S) \leq \varepsilon$, we say that x is ε-close to S, and otherwise we say that x is ε-far from S.
- We denote by $A^x(y)$ the output of algorithm A given direct access to input y and oracle access to string x. Given two interactive machines A and B, we denote by $(A^x, B(y))(z)$ the output of A when interacting with B, where A (respectively, B) is given oracle access to x (respectively, direct access to y) and both parties have direct access to z. Throughout this work, probabilistic expressions that involve a randomized algorithm A are taken over the inner randomness of A (e.g., when we write $\Pr[A^x(y) = z]$, the probability is taken over the coin tosses of A).
- We use the notations $\tilde{O}(f), \tilde{\Omega}(f)$ to hide polylogarithmic dependencies on the argument, i.e. for expressions of the form $O(f \log^c f)$ and $\Omega(f \log^c f)$ (for some absolute constant c). Finally, all our logarithms are in base 2.

Integrality. For simplicity of notation, we hereafter use the convention that all (relevant) integer parameters that are stated as real numbers are implicitly rounded to the closest integer.

Uniformity. To facilitate notation, throughout this work we define all algorithms *non-uniformly*; that is, we fix an integer $n \in \mathbb{N}$ and restrict the algorithms to inputs of length n. Despite fixing n, we view it as a generic parameter and allow ourselves to write asymptotic expressions such as $O(n)$. We remark that while our results are proved in terms of non-uniform algorithms, they can be extended to the uniform setting in a straightforward manner.

3 The Definition of Testers with Bounded Adaptivity

In this section, we provide a formal abstraction that captures the notion of *bounded adaptivity* within the framework of property testing. We define two notions of bounded adaptivity: (1) *round-adaptivity*, which refers to algorithms that are allowed to make a bounded number of "batches" of queries, where the queries in each batch may depend on the answers to previous batches; (2) *tail-adaptivity*, which refers to algorithms that first make a large number of non-adaptive queries and subsequently make a bounded number of adaptive queries.

We remark that while tail-adaptivity can be easily emulated via round-adaptivity, the converse does *not* hold. Indeed, in Section 6.3 we show that round-adaptive testers can be much more powerful than tail-adaptive testers. Nonetheless, our lower bounds hold for the stronger round-adaptivity notion, whereas out upper bounds hold for the more restrictive tail-adaptivity.

▶ **Definition 3** (Round-Adaptive Testing Algorithms)**.** Let $[n]$ be a domain of cardinality n, and let $k, q \leq n$. A randomized algorithm is said to be a (k, q)-*round-adaptive* tester for a property $\mathcal{P} \subseteq 2^{[n]}$, if, on proximity parameter $\varepsilon \in (0, 1]$ and granted query access to a function $f\colon [n] \to \{0, 1\}$, the following holds.

1. Query Generation: The algorithm proceeds in $k + 1$ rounds, such that at round $\ell \geq 0$, it produces a set of queries $Q_\ell \stackrel{\text{def}}{=} \{x^{(\ell),1}, \ldots, x^{(\ell),|Q_\ell|}\} \subseteq [n]$ (possibly empty), based on its own internal randomness and the answers to the previous sets of queries $Q_0, \ldots, Q_{\ell-1}$, and receives $f(Q_\ell) = \{f(x^{(\ell),1}), \ldots, f(x^{(\ell),|Q_\ell|})\}$;
2. Completeness: If $f \in \mathcal{P}$, then the algorithm outputs accept with probability at least $2/3$;
3. Soundness: If $\mathrm{dist}(f, \mathcal{P}) > \varepsilon$, then the algorithm outputs reject with probability at least $2/3$.

The *query complexity* q of the tester is the total number of queries made to f, i.e., $q = \sum_{\ell=0}^{k} |Q_\ell|$. If the algorithm returns accept with probability one whenever $f \in \mathcal{P}$, it is said to have *one-sided* error (otherwise, it has *two-sided* error). We will sometimes refer to a tester with respect to proximity parameter ε as an ε-tester.

▶ Remark (On amplification). We note that, as usual in property testing, the probability of success can be amplified by repetition to any $1-\delta$, at the price of an $O(\log(1/\delta))$ factor in the query complexity. Crucially, this can be done with no increase in the number of adaptive rounds: while repetition would naïvely multiply both q *and* k by this factor, one can avoid the latter by running the $O(\log(1/\delta))$ independent copies of the algorithm in parallel, instead of sequentially.

▶ **Definition 4** (Tail-Adaptive Testing Algorithms)**.** Let $[n]$ be a domain of cardinality n, and let $k, q \leq n$. A randomized algorithm is said to be a (k,q)-*tail-adaptive* tester for a property $\mathcal{P} \subseteq 2^{[n]}$, if, on proximity parameter $\varepsilon \in (0,1]$, error parameter $\delta \in (0,1]$, and granted query access to a function $f\colon [n] \to \{0,1\}$, the following holds.

1. Query Generation: The algorithm proceeds in $k+1$ rounds, such that in the first round, it produces a set of queries $Q \stackrel{\text{def}}{=} \{x^{(0),1}, \ldots, x^{(0),|Q|}\} \subseteq [n]$ (possibly empty), based on its own internal randomness; and receives $f(Q) = \{f(x^{(0),1}), \ldots, f(x^{(0),|Q|})\}$; then it makes, over the next k rounds, k adaptive queries to f, denoted $x^{(1)}, \ldots, x^{(k)}$;
2. Completeness: If $f \in \mathcal{P}$, then the algorithm outputs accept with probability at least $1-\delta$;
3. Soundness: If $\operatorname{dist}(f, \mathcal{P}) > \varepsilon$, then the algorithm outputs reject with probability at least $1-\delta$.

The *query complexity* q of the tester is the total number of queries made to f, i.e., $q = |Q| + k$. If the algorithm returns accept with probability one whenever $f \in \mathcal{P}$, it is said to be *one-sided* (otherwise, it is *two-sided*).

▶ Remark (On (lack of) amplification). Unlike the round-adaptive algorithms, tail-adaptive testing algorithms do not enjoy a simple success amplification procedure which would leave unchanged the adaptivity parameter, only affecting the query complexity. This is the reason why the success probability δ is explicitly mentioned in Definition 4.

4 A Strong Adaptivity Hierarchy

In this section we prove the adaptivity hierarchy theorem, which shows that, loosely speaking, up to a nearly linear threshold, each additional round of adaptivity can significantly augment the power of testing algorithms.

▶ **Theorem 5** (Adaptivity Hierarchy Theorem)**.** *Fix any $\alpha \in (0,1)$. There exists a constant $\beta \in (0,1)$ such that, for every $n \in \mathbb{N}$, the following holds. For every integer $0 \leq k \leq n^\beta$, there exists a property $\mathcal{P}_k \subseteq \mathbb{F}_n^{n^{1+\alpha}}$ such that, for any constant $\varepsilon \in (0,1]$,*

1. *there exists a $(k, \tilde{O}(k))$-round-adaptive (one-sided) tester for $\mathcal{P}_k$; yet*
2. *any $(k-1, q)$-round-adaptive (two-sided) tester for $\mathcal{P}_k$ must satisfy $q = \tilde{\Omega}(n/k^2)$.*

We remark that, in fact, the algorithm shown in the first item of Theorem 5 also gives an upper bound for the more restricted model of *tail adaptivity*. Specifically, for every k there also exists an $(O(k), \tilde{O}(k))$-tail-adaptive (one-sided) tester for $\mathcal{P}_k$. Since a $(k-1,q)$-round-adaptive lower bound implies a $(k-1,q)$-tail-adaptive lower bound (see discussion in Section 3), this implies an adaptivity hierarchy (albeit slightly weaker than in Theorem 5) with respect to tail-adaptive testers.

To prove Theorem 5, we use a technique that allows us to "lift" bounds on decision tree complexity to the setting of property testing, relying on error-correcting codes with local properties. In more detail, we shall begin by proving an analogous version of Theorem 5 for the (linear) decision tree model, which is an elementary computation model that is significantly easier to analyze than property testing (in particular, it deals with *exact* decision problems, rather than *approximate* decision problems). Then, we shall consider an encoded version of the decision tree problem, where the encoding is via a code that admits strong local testability and a relaxed form of local decodability. Capitalizing on the foregoing properties of the code, we show transference lemmas that allow us to "lift" our bounds on the decision tree problem to bounds on the complexity of the encoded problem in the property testing model. (We believe that this "lifting" technique may find further uses in property testing, by allowing one to show results in the simpler decision tree problem before carrying them over to property testing.)

We begin by describing the decision tree problem with respect to which we prove the hierarchy theorem. Hereafter we assume, without loss of generality,[4] that n is a prime number, and consider $\mathbb{F}_n$, the field of order n. We will consider the following sequence of "k-iterated address" functions $(f_k)_{k\geq 0}$ from $\mathbb{F}_n^n$ to $\{0,1\}$, which will in turn lead to the definition of the properties $(\mathcal{P}_k)_{k\geq 0}$ that we use to show the hierarchy theorem. Loosely speaking, f_k receives a vector x of n pointers (indices in $[n]$) and indicates whether when jumping from pointer to pointer k times, starting from an arbitrarily predetermined pointer, we reach a location in which x takes an even value.

To formally define the foregoing functions, first consider $g\colon \mathbb{F}_n^n \times \mathbb{F}_n \to \mathbb{F}_n$ given by $g(x,a) = x_{a+1}$; that is, g returns the coordinate of $x \in \mathbb{F}_n^n$ "pointed to" by $a \in \{0,\dots,n-1\}$. Based on this, we define the iterated versions of g, $g_0,\dots,g_n,\dots : \mathbb{F}_n^n \to \mathbb{F}_n$, as

$$\begin{aligned} g_0(x) &= g(x,0)\,, \\ g_k(x) &= g(x, g_{k-1}(x))\,. && (k\geq 1) \end{aligned}$$

Finally, we define the *k-iterated address* function $f_k\colon \mathbb{F}_n^n \to \{0,1\}$ by

$$f_k(x) = \mathbb{1}_{\{g_k(x)\text{ even}\}} = \begin{cases} 1 & \text{if } g_k(x) \text{ is even,} \\ 0 & \text{otherwise.} \end{cases}$$

(For instance, $f_0(x) = 1$ if and only if x_1 is even; and $f_1(x) = 1$ if and only if the coordinate of x pointed to by x_1, that is x_{x_1+1}, is even.) We proceed to describe the outline of the proof of Theorem 5.

4.1 High-Level Overview

Broadly speaking, our roadmap for proving Theorem 5 consists of two main steps:

1. We first consider the adaptivity hierarchy question in the setting of randomized *decision tree* (DT) complexity (see Section 4.2). We can view a randomized DT for computing a function f as a probabilistic algorithm that is given query access to an input x and is required to output $f(x)$ with high probability. Adapting the definition of round adaptivity (Definition 3) in the natural way to decision trees, we will prove the randomized DT analogue of our adaptivity hierarchy theorem, using the foregoing family of address

[4] If n is not prime, we choose a prime p such that $n \leq p \leq 2p$, and use standard padding techniques.

functions $(f_k)_{k\geq 0}$. Namely, we prove that for any $k \geq 0$ with $k = o(n)$, it holds that (i) f_k can be computed by an algorithm making $k+1$ queries, in k adaptive rounds; but (ii) any algorithm using only $k-1$ rounds of adaptivity must make $\tilde{\Omega}(n/k^2)$ queries.
2. We then show a bidirectional connection between *adaptivity-bounded* randomized DT and property testers, which extends the connection observed by Tell [39]. This allows us to “lift” the DT adaptivity hierarchy theorem to property testing. Specifically, we provide two blackbox reductions between the DT problem of computing function f and property testing for a related property $\mathcal{P}_f$, which preserve both the number of adaptive rounds and (roughly) the number of queries. We remark these reductions strongly rely on high-rate codes that exhibit both strong local testability and relaxed local decodability.

The caveat with the above is that to “lift” DT lower bounds to testing algorithms via our methodology, we actually need to show lower bounds on a stronger model of DT (this stems from the reductions of the second item, in which we will encode the input via linear codes, requiring the DT algorithm to compute coordinates of this encoding).

Hence, we will actually work in the *linear decision tree* (LDT) model, wherein the algorithm is allowed to query any linear combination (over $\mathbb{F}_n$) of the coordinates, instead of only querying individual coordinates. (We note that in the case of $\mathbb{F}_2$, this corresponds to the *parity decision tree* model.) That is, we will proceed as follows:

1. (L)DT hierarchy: show that for any $k \geq 0$, the function f_k (i) can be computed by an efficient $(k, O(k))$-round-adaptive (deterministic) DT algorithm, but (ii) does not admit any $(k-1, o(n))$-round-adaptive (randomized, two-sided) LDT algorithm;
2. Transference lemmas: Show that for any function $f\colon \mathbb{F}_n^n \to \mathbb{F}_n$, there exists a property $\mathcal{C}_f \subseteq \mathbb{F}_n^{m(n)}$ such that, for any $k \geq 0$,
   a. a (k, q)-round-adaptive testing algorithm for $\mathcal{C}_f$ implies a (k, q)-round-adaptive LDT algorithm for f (Lemma 14).
   b. a (k, q)-round-adaptive DT algorithm for f implies a $(k, \tilde{O}(q))$-round-adaptive testing algorithm for $\mathcal{C}_f$ (Lemma 15).

Combining the items above will directly imply our hierarchy theorem for property testing (Theorem 5):

Proof. Proof of Theorem 5 The upper bound 1 follows immediately from Claim 7 and Lemma 15, while combining Lemma 8 and Lemma 14 establishes the lower bound 2. ◀

Organization for the rest of the section. In Section 4.2, we define the decision tree models and complexities that we shall need. Then, in Section 4.3, we prove the adaptivity hierarchy theorem for randomized (linear) decision trees. Finally, in Section 4.4 we prove the transference lemmas that allow us to lift the foregoing hierarchy theorem to the property testing framework.

4.2 Decision Tree Zoo

We shall need to extend the definitions of several different types of decision tree algorithms (see [13] for an extensive survey of decision tree complexity) to the setting of bounded adaptivity.

Recall that a deterministic decision tree is a model of computation for computing a function $f\colon [n]^n \to [n]$. The decision tree is a rooted ordered $|[n]|$-ary tree. Each internal vertex of the tree is labeled with a value $i \in \{1, \ldots, n\}$ and the leaves of the tree are labeled with the elements in $[n]$. Given an input $x \in [n]^n$, the decision tree is recursively evaluated by choosing to recurse on the i'th subtree in the j'th level if and only if $x_j = i$. Once a leaf is reached, we output the label of that leaf and halt.

Equivalently, we can view deterministic decision trees as algorithms that get oracle access to an input $x \in [n]^n$, then adaptively make queries to x, to the end of computing $f(x)$. (Note that the j'th query corresponds to the j'th layer of the corresponding decision tree, and that the different vertices in the j'th layer represent the choices of the next queries, with respect to the answers obtained for previous queries). We define the *deterministic decision tree complexity* of a function f to be the minimal number of queries a deterministic decision tree algorithm needs to make to compute f in the worst case.[5]

Taking the algorithmic perspective, we define k-round-adaptive deterministic decision tree algorithms as algorithms that generate their queries in k rounds, where queries in each round may depend on queries from previous rounds. The extension of the foregoing definition to *randomized* decision tree algorithms is done in the natural way, by allowing the algorithm to toss random coins and succeed with high probability (say, 2/3) in computing $f(x)$. Finally, we shall also extend the definition to linear decision trees, which are decision trees algorithms wherein each query is a linear combination of the elements of the domain. We remark that linear decision trees can be thought of as generalizing both *parity decision trees* and *algebraic query complexity algorithms* [1].

More accurately, the aforementioned notions are defined below. We provide the definition of the most general model and derive the more restricted models as special cases.

▶ **Definition 6** (Round-Adaptive Decision Tree Algorithms)**.** Let $\mathbb{F}$ be a finite field of cardinality n, and let $k, q \le n$. A (randomized) algorithm D is said to be a (k, q)*-round-adaptive* (linear) decision tree algorithm for computing a function $f\colon \mathbb{F}^n \to \mathbb{F}$ if, granted query access to a string $x \in \mathbb{F}^n$, the following holds.

1. Query Generation: The algorithm proceeds in $k+1$ rounds, such that at round $\ell \ge 0$, it produces a set of (linear) queries $Q_\ell \stackrel{\text{def}}{=} \{L_{\ell,1}, \ldots, L_{\ell,|Q_\ell|}\}$, where $L_{\ell,j} \in \mathbb{F}^n$ specifies a linear combination, based on its internal randomness and the answers to the previous sets of queries $Q_0, \ldots, Q_{\ell-1}$, and receives the answers $\langle L_{\ell,1}, x\rangle, \ldots, \langle L_{\ell,|Q_\ell|}, x\rangle$.
2. Computation: The algorithm computes $f(x)$ with high probability using the answers it received in all k rounds; that is, $\Pr[D^x = f(x)] \ge 2/3$.

The *query complexity* q of the tester is the total number of (linear) queries made to f, i.e., $q = \sum_{\ell=0}^{k} |Q_\ell|$. The *randomized* (k, q)*-round-adaptive linear decision tree complexity* of a function f, denoted $R_k^{\oplus}(f)$, is the minimal query complexity for a (k, q)-round-adaptive randomized linear decision tree algorithm that computes f.

If for all $\ell \in [k+1]$ and $j \in [|Q_\ell|]$ the linear combination $L_{\ell,j}$ only includes a single element (i.e., $L_{\ell,j}$ only has a single non-zero entry), we say that D is a *randomized* (k, q)*-round-adaptive decision tree algorithm complexity*, and denote its corresponding complexity by $R_k(f)$. If, in addition, the algorithm does not toss any random coins and succeeds with probability 1, we say that D is a *deterministic* (k, q)*-round-adaptive decision tree algorithm complexity*, and denote its corresponding complexity by $D_k(f)$.

[5] We remark that this definition corresponds to the depth the of decision tree, and not to the number of vertices or edges in the tree.

4.3 Decision Tree Hierarchy: Some Things Only Adaptivity Can Address

We first establish the upper bound part of our adaptivity hierarchy theorem for DT, which follows immediately from the construction.

▶ **Claim 7.** *For every $k \geq 0$, there exists a $(k, k+1)$-round-adaptive (deterministic) DT algorithm which computes f_k; that is, $D_k(f_k) \leq k+1$.*

Proof. The algorithm is straightforward: on input $x \in \mathbb{F}_n^n$, it sequentially queries $x_1 = g_0(x)$, $x_{g_0(x)+1} = g_1(x)$, …, $x_{g_{k-1}(x)+1} = g_k(x)$; and returns 1 if $g_k(x)$ is even, and 0 otherwise. By definition of f_k, this always correctly computes the function, is deterministic, and clearly satisfies the definition of a $(k, k+1)$-round-adaptive DT algorithm. ◀

We proceed to show the lower bound part of our adaptivity hierarchy theorem for DT, which is proven via a reduction from communication complexity.

▶ **Lemma 8.** *There exists an absolute constant $c > 0$ such that the following holds. For every $0 \leq k \leq c\left(\frac{n}{\log n}\right)^{1/3}$, there is no $(k, o(n/(k^2 \log n)))$-round-adaptive (randomized) LDT algorithm which computes f_{k+1}; that is, $R_k^{\oplus}(f_{k+1}) = \Omega\big(n/(k^2 \log n)\big)$.*

Proof. We will reduce to the computation of f_{k+1} (in k rounds of adaptivity) a related k-round two-party randomized communication complexity problem, the "pointer-following" problem introduced by Papadimitriou and Sipser [30], and conclude by invoking the lower bound of Nisan and Wigderson [29] on this problem.

This communication complexity problem between two computationally unbounded players, Alice and Bob, is defined as follows. Let V_A and V_B be two disjoint sets of cardinality $n/2$, and let $v_0 \in V_A$ be a fixed element known to both players. The input is a pair of functions (χ_A, χ_B), where $\chi_A \colon V_A \to V_B$ and $\chi_B \colon V_B \to V_A$. Alice and Bob are given χ_A and χ_B respectively, as well as a common random string, and their goal is to compute $\pi_k(\chi_A, \chi_B) \stackrel{\text{def}}{=} \chi^{(k)}(v_0)$ with high probability, where $\chi^{(\ell)}$ is the ℓ-iterate of the function χ:

$$\begin{aligned}\chi \colon V_A \cup V_B &\to V_A \cup V_B \\ v &\mapsto \begin{cases} \chi_A(v) & v \in V_A \\ \chi_B(v) & v \in V_B. \end{cases}\end{aligned}$$

(In other terms, one can see the communication problem as Alice and Bob sharing the edges of a bipartite directed graph where each node has out-degree exactly one, and the goal is to find at which vertex the path of length k starting at a prespecified vertex v_0, on Alice's side, ends.)

We will rely on the following lower bound on the k-round, randomized (public-coin) version of this problem.

▶ **Theorem 9** ([29], rephrased)**.** *Any k-round randomized communication protocol for the "pointer-following" problem, in which Bob sends the first message, must have total communication complexity $\Omega\big(\frac{n}{k^2} - k \log n\big)$, even to only compute a single bit of $\pi_k(\chi_A, \chi_B)$ with probability at least $2/3$.*

Note that as long as $k \ll \left(\frac{n}{\log n}\right)^{1/3}$, this lower bound is $\Omega\big(\frac{n}{k^2}\big)$. We remark that the fact that the lower bound still holds even when only a single bit of the answer is to be computed will be crucial for us, as our goal is to reduce the communication complexity problem of "pointer-following" to computing the Boolean function f_{k+1} in the randomized decision tree model.

Let $\mathcal{A}$ be any (k, q)-round-adaptive (randomized) LDT algorithm computing f_{k+1}. Writing $V_A = \{v_0, \dots, v_{\frac{n}{2}-1}\}$ and $V_B = \{u_0, \dots, u_{\frac{n}{2}-1}\}$, fix a bijection between $V \stackrel{\text{def}}{=} V_A \cup V_B$ (of size n) and $\mathbb{F}_n$ mapping v_0 to 1, so that we identify V with $\mathbb{F}_n$. On input (χ_A, χ_B), Alice and Bob implicitly define the element $x \in \mathbb{F}_n^n$ by $x_1 = \chi_A(v_0)$, $x_2 = \chi_A(v_1)$, $\dots$, $x_{\frac{n}{2}} = \chi_A(v_{\frac{n}{2}-1})$ and $x_{\frac{n}{2}+1} = \chi_B(u_0)$, $x_{\frac{n}{2}+2} = \chi_A(u_1)$, $\dots$, $x_n = \chi_A(u_{\frac{n}{2}-1})$. From this, we get that $\pi_{k+2}(\chi_A, \chi_B) = g_{k+1}(x)$, recalling that $g_k(x) = g(x, g_{k-1}(x))$ is recursively defined for $k \geq 1$, and $g_0(x) = x_1$. Hence deciding whether $\pi_{k+2}(\chi_A, \chi_B)$ is even is exactly equivalent to computing $f_{k+1}(x)$.

Alice and Bob can then simulate the execution of $\mathcal{A}$ as follows. Without loss of generality, assume it is Alice's turn to speak. To answer a query of the form $\phi_a(x) = \sum_{i=1}^n a_i x_i$, she computes $\sum_{i\in V_A} a_i x_i$ and sends it to Bob; on his side, Bob computes $\sum_{i\in V_B} a_i x_i$, and receiving Alice's message can then recover the value $\phi_a(x)$ and feed it to the algorithm. (In the next round, when sending his side of the (new) queries to Alice, Bob will also send this value $\phi_a(x)$, to make sure that both sides know the answers to all queries so far.) Since all queries of a given adaptive round of $\mathcal{A}$ can be prepared and sent in parallel (costing $O(\log n)$ bits of communication per query), this simulation can be performed in $k + 1$ rounds (as many as $\mathcal{A}$ takes) with communication complexity $O(q \log)$. At the end, whichever of Alice and Bob received the latest message holds the answer (to "is $\pi_{k+1}(\chi_A, \chi_B)$ an even node?"), which by assumption on $\mathcal{A}$ is correct with probability at least 2/3. Alice and Bob then use an extra round of communication to broadcast the answer to the other party, bringing the total number of rounds to $k + 2$.

But by Theorem 9, computing this bit of $\pi_{k+2}(\chi_A, \chi_B)$ with only $k + 2$ rounds of communication (Bob speaking first) requires $\Omega(\frac{n}{k^2})$ bits of communication, and so we must have $q = \Omega\left(\frac{n}{k^2 \log n}\right)$. ◀

4.4 Adaptivity Bounded Testers and Decision Trees: There and Back Again

In this section we show how to reduce problems in the adaptivity bounded property testing model to problems in the adaptivity bounded (linear) decision tree model, and vice versa. We begin in Section 4.4.1, by presenting the required preliminaries regarding error-correction codes. Then, in Section 4.4.2, we prove the "transference lemmas" between these models.

4.4.1 Preliminaries: Locally Testable and Decodable Codes

Let $k, n \in \mathbb{N}$. A code over alphabet Σ with distance d is a function $C \colon \Sigma^k \to \Sigma^n$ that maps messages to codewords such that the distance between any two codewords is at least $d = d(n)$. If $d = \Omega(n)$, C is said to have linear distance. If $\Sigma = \{0, 1\}$, we say that C is a binary code. If C is a linear map, we say that it is a linear code. The relative distance of C, denoted by $\delta(C)$, is d/n, and its rate is k/n. When it is clear from the context, we shall sometime abuse notation and refer to the code C as the set of all codewords $\{C(x)\}_{x\in\Sigma^k}$. Following the discussion in the introduction, we define locally testable codes and locally decodable codes as follows.

▶ **Definition 10** (Locally Testable Codes). A code $C \colon \Sigma^k \to \Sigma^n$ is a locally testable code (LTC) if there exists a probabilistic algorithm (tester) T that makes $O(1)$ queries to a purported codeword $w \in \Sigma^n$ and satisfies:

1. Completeness: For any codeword w of C it holds that $\Pr_T[T^w = 1] \geq 2/3$.
2. Strong Soundness: For all $w \in \Sigma^n$,

$$\Pr_T[T^w = 0] \geq \mathrm{poly}\left(\mathrm{dist}(w, C)\right).$$

▶ **Definition 11** (Locally Decodable Codes). A code $C\colon \Sigma^k \to \Sigma^n$ is a locally decodable code (LDC) if there exists a constant $\delta_{\mathsf{radius}} \in (0, \delta(C)/2)$ and a probabilistic algorithm (decoder) D that, given oracle access to $w \in \Sigma^n$ and direct access to index $i \in [k]$, satisfies the following condition: For any $i \in [k]$ and $w \in \Sigma^n$ that is δ_{radius}-close to a codeword $C(x)$ it holds that $\Pr[D^w(i) = x_i] \geq 2/3$. The query complexity of a LDC is the number of queries made by its decoder.

We shall also need the notion of relaxed-LDCs (introduced in [3]). Similarly to LDCs, these codes have decoders that make few queries to an input in attempt to decode a given location in the message. However, unlike LDCs, the relaxed decoders are allowed to output a special symbol that indicates that the decoder detected a corruption in the codeword and is unable to decode this location. Note that the decoder must still avoid errors (with high probability).[6]

▶ **Definition 12** (Relaxed-LDC). A code $C\colon \Sigma^k \to \Sigma^n$ is a relaxed-LDC if there exists a constant $\delta_{\mathsf{radius}} \in (0, \delta(C)/2)$ such that the following holds.

1. (Perfect) Completeness: For any $i \in [k]$ and $x \in \Sigma^k$ it holds that $D^{C(x)}(i) = x_i$.
2. Relaxed Soundness: For any $i \in [k]$ and any $w \in \Sigma^n$ that is δ_{radius}-close to a (unique) codeword $C(x)$, it holds that

$$\Pr[D^w(i) \in \{x_i, \perp\}] \geq 2/3.$$

There are a couple of efficient constructions of codes that are both relaxed-LDCs and LTCs (see [3, 22]). We shall need the construction in [22], which has the best parameters for our setting.[7]

▶ **Theorem 13** (e.g., [22, Theorem 1.1]). *For every $k \in \mathbb{N}$, $\alpha > 0$, and finite field $\mathbb{F}$ there exists an $\mathbb{F}$-linear code $C\colon \mathbb{F}^k \to \mathbb{F}^{k^{1+\alpha}}$ with linear distance, which is both a* relaxed-LDC *and a (one-sided error)* LTC *with query complexity* $\mathrm{poly}(1/\varepsilon)$*; furthermore, both testing and (relaxed) decoding procedures are non-adaptive.*

4.4.2 Transference Lemmas

Fix any $\alpha > 0$. Let $C\colon \mathbb{F}_n^n \to \mathbb{F}_n^m$ be a code with constant relative distance $\delta(C) > 0$, with the following properties:

- *linearity*: for all $i \in [m]$, there exists an element $a^{(i)} \in \mathbb{F}_n^n$ such that $C(x)_i = \langle a^{(i)}, x\rangle$ for all $x \in \mathbb{F}_n^n$;

[6] The full definition of relaxed-LDCs, as defined in [3] includes an additional condition on the success rate of the decoder. Namely, for every $w \in \{0,1\}^n$ that is δ_{radius}-close to a codeword $C(x)$, and for at least a ρ fraction of the indices $i \in [k]$, with probability at least $2/3$ the decoder D outputs the i'th bit of x. That is, there exists a set $I_w \subseteq [k]$ of size at least ρk such that for every $i \in I_w$ it holds that $\Pr\left[D^w(i) = x_i\right] \geq 2/3$. We omit this condition since it is irrelevant to our application, and remark that every relaxed-LDC that satisfies the first two conditions can also be modified to satisfy the third conditions (see [3, Lemmas 4.9 and 4.10]).

[7] Specifically, the codes in [22] are meaningful for every value of the proximity parameter, whereas the codes in [3] require $\varepsilon > 1/\mathrm{polylog}(k)$.

- *rate*: $m \leq n^{1+\alpha}$;
- *testability*: C is a *strong-LTC* with one-sided error and *non-adaptive* tester;
- *decodability*: C is a *relaxed-LDC*.

We will rely on Theorem 13 for the existence of such codes. Before delving into the details, we briefly explain the reason for each of the points above. The linearity will be crucial to reduce to and from the LDT model: indeed, any coordinate of a codeword corresponds to a fixed linear combination of the coordinates of the message, which corresponds to a single LDT query on that particular linear combination. The rate bound is required since our lower bounds are in terms of the dimension n and upper bounds in terms of the block-lengh m. Ideally, we would like $m = O(n)$, to have a direct correspondence between the LDT and the property testing query complexities; however, this nearly-linear rate is the best known achievable for constant-query LTCs and relaxed-LDCs [22]. The LTC property will be useful to us in the reduction from property testing to DT query complexity (where we will need to first check that our input is close to a codeword, in view of decoding the closest message during the reduction), where the *strong* testability (i.e., rejection with probability proportional to the distance from a valid codeword) will allow us do deal with arbitrarily small values of the proximity parameter. Similarly, we will rely on the (relaxed) LDC property in that same reduction, in order to obtain individual coordinates of the message, given query access to an input close to a codeword.

We proceed to show the framework for reducing property testing to decision tree complexity and vice-versa. For a fixed function $f\colon \mathbb{F}_n^n \to \{0,1\}$, consider the subset $f^{-1}(1) \subseteq \mathbb{F}_n^n$; and define the sets of codewords $\mathcal{C} \stackrel{\text{def}}{=} C(\mathbb{F}_n^n) \subseteq \mathbb{F}_n^m$, $\mathcal{C}_f \stackrel{\text{def}}{=} C(f^{-1}(1)) = \{\, C(x) : x \in \mathbb{F}_n^n,\ f(x) = 1 \,\} \subseteq \mathcal{C}$.

Consider now testing the property $\mathcal{C}_f$: we will reduce the LDT computation of f to the testing of $\mathcal{C}_f$. Specifically, we prove the following.

▶ **Lemma 14** (LDT ⇝ PT Reduction Lemma). *Fix any $f\colon \mathbb{F}_n^n \to \{0,1\}$. If there exists an (k,q)-round-adaptive tester for $\mathcal{C}_f$, then there is an (k,q)-round-adaptive LDT algorithm for f.*

Proof. Suppose there exists a (k,q)-round-adaptive tester $\mathcal{T}$ for $\mathcal{C}_f$. On input $x \in \mathbb{F}_n^n$, we emulate the invocation of $\mathcal{T}$, with respect to proximity parameter $\varepsilon = \delta(C)$, on the encoded input $y \stackrel{\text{def}}{=} C(x) \in \mathbb{F}_n^m$ and output 1 if and only if $\mathcal{T}$ returns accept. To see why this is correct, observe that by definition, if $f(x) = 1$ then $y \in \mathcal{C}_f$. However, if $f(x) = 0$, then for any $y' \in \mathcal{C}_f$ such that $y' = C(x)$ we must have $\operatorname{dist}(y, y') > \varepsilon$, by the distance of our code.

It remains to show that this simulation can be achieved efficiently, as claimed. To do so, we will rely on the fact that C is a linear code: whenever $\mathcal{T}$ queries y_i, we can compute the element $a^{(i)} \in \mathbb{F}_n^n$ (which only depends on C, and not on x), and perform the LDT query $\langle a^{(i)}, x \rangle$. The simulation clearly preserves the number of adaptive rounds as well, concluding the proof. ◀

In our next lemma, we give a partial converse relating property testing and decision tree complexity, with some logarithmic overhead in the resulting query complexity.

▶ **Lemma 15** (PT ⇝ DT Reduction Lemma). *Fix any $f\colon \mathbb{F}_n^n \to \{0,1\}$. If there exists an (k,q)-round-adaptive (randomized) DT algorithm for f, then there is a $(k, O(q\log q) + \operatorname{poly}(1/\varepsilon))$-round-adaptive tester for $\mathcal{C}_f$. (Moreover, if the DT algorithm is always correct, then this tester is one-sided.)*

Proof. Fix $k \geq 0$, and suppose there exists such a (k,q)-round-adaptive DT algorithm $\mathcal{A}$ for f. On input $y \in \mathbb{F}_n^m$ and proximity parameter $\varepsilon \in (0,1]$, we would like to decode y to a message $x \in \mathbb{F}_n^n$ and invoke the algorithm on x to determine if $f(x) = 1$; more precisely, we wish to invoke the DT algorithm while simulating each query to x by locally decoding y using $O(1)$ queries. The issue, however, is that the success of the local decodable is only guaranteed for inputs that are sufficiently close to a valid codeword, and we have no such guarantee on y *a priori*. However, recalling that C is a strong-LTC, we can handle this as follows. Letting $\delta_{\mathsf{radius}} > 0$ be the decodability radius of the relaxed-LDC C, we set $\delta^* \stackrel{\text{def}}{=} \min(\delta_{\mathsf{radius}}, \varepsilon)$.

1. Run independently $O(\text{poly}(1/\delta^*))$ times the local tester for the strong-LTC C on y, and output reject if any of these rejected. Since every invocation of the local tester makes $O(1)$ queries to y, this has query complexity $O(\text{poly}(1/\delta^*)) = O(\text{poly}(1/\varepsilon))$; and if $\text{dist}(y, \mathcal{C}) > \delta^*$ then this step outputs reject with probability at least $9/10$.
2. Invoke $\mathcal{A}$ on the message $x \stackrel{\text{def}}{=} \text{argmin}\{\, \text{dist}(C(x), y) \,:\, x \in \mathbb{F}_n^n \,\}$, answering each query x_i by calling the local decoder for the relaxed-LDC C. This is done so that the decoder is correct with probability at least $1/(10q)$, by standard repetition (taking the plurality value); with the subtlety that we output reject immediately whenever the decoder returns $\perp$. Since each query can be simulated by $O(\log(q))$ queries (repeating the $O(1)$ queries of the decoder $O(\log q)$ times), this step has query complexity $O(q \log q)$; and at the end, we output accept if, and only if, $\mathcal{A}$ returns the value 1 for $f(x)$.

Importantly, Step 1 can be run in parallel to Step 2, and in particular can be executed during the first "batch" of queries $\mathcal{A}$ makes. This guarantees that the whole simulation above uses the same number of adaptive rounds as $\mathcal{A}$, as claimed. It remains to argue correctness.

Completeness. Assume $y \in \mathcal{C}_f$. In particular, y is a codeword of C, and the (one-sided) local tester returns accept with probability one in 1. Then, since by definition there is a unique $x \in \mathbb{F}_n^n$ such that $C(x) = y$, the local decoder of Step 2 will correctly output the correct answer for each query with probability 1, and therefore $\mathcal{A}$ will correctly output $f(x)$ with probability $2/3$ – so that the tester returns accept with probability at least $2/3$ overall. (Moreover, if the DT algorithm $\mathcal{A}$ always correctly compute f, then the tester returns accept with probability one.)

Soundness. Assume $\text{dist}(y, \mathcal{C}_f) > \varepsilon$. If $\text{dist}(y, \mathcal{C}) > \delta^*$, then the local tester returns reject with probability at least $9/10$ in Step 1. Therefore, we can continue assuming that $\text{dist}(y, \mathcal{C}) \leq \delta^*$, which satisfies the precondition of the relaxed-LDC decoder in Step 2. By a union bound over all q queries, with probability at least $9/10$ we have that the decodings performed in Step 2 are all correct; in which case we answer the queries of the algorithm according to $x \stackrel{\text{def}}{=} \text{argmin}\{\, \text{dist}(C(x), y) \,:\, x \in \mathbb{F}_n^n \,\}$ (or possibly answered by $\perp$, in which case the tester immediately outputs reject and we are done). Since $\text{dist}(y, C(x)) \leq \delta^* \leq \varepsilon$, we must have $C(x) \notin \mathcal{C}_f$, which implies that $\mathcal{A}$ correctly returns $f(x) = 0$ with probability at least $2/3$, in which case the tester outputs reject. Overall, this happens with probability at least $9/10 \cdot 9/10 \cdot 2/3 = 27/50$.

Thus, in both cases the tester is correct with probability at least $27/50$; repeating a constant number of times (as explained in the remark of page 27:6) and taking the majority vote allows us to amplify the probability of success to $2/3$. ◀

5 An Adaptivity Hierarchy with respect to a Natural Property

In this section we show a *natural* property of graphs for which, broadly speaking, more adaptivity implies more power. More specifically, we prove the following adaptivity hierarchy theorem with respect to the property of k-cycle freeness in the bounded-degree graph model (see definitions in Section 5.1).

▶ **Theorem 16.** *Let $k \in \mathbb{N}$ be a constant. Then,*

1. *there exists a $(k, O(1/\varepsilon))$-round-adaptive (one-sided) tester for $(2k+1)$-cycle freeness in the bounded-degree graph model; yet*
2. *any $(k-1, q)$-round-adaptive (two-sided) tester for $(2k+1)$-cycle freeness in the bounded-degree graph model must satisfy $q = \Omega(\sqrt{n})$.*

We stress that although Theorem 5 establishes an adaptivity hierarchy with stronger separations, the merit of Theorem 16 is in showing that an adaptivity hierarchy also holds for a *natural* well-studied property. We further observe that the choice of the bounded-degree graph model is not insignificant: one cannot hope to establish such a striking gap in other settings such as the dense graph model or in the Boolean function testing setting. Indeed, as discussed in Section 1.2 it is well-known that in these two models, any adaptive tester can be made (fully) non-adaptive at the price of only a quadratic and exponential blowup in the query complexity, respectively(see [2, 25] for the former; the latter is folklore). We remark that in Section 6.1 we discuss emulating testers with k rounds of adaptivity by testers with $k' < k$ rounds.

5.1 Cycle Freeness in the Bounded Degree Graph Model

In the subsection we provide the necessary definitions and establish a basic upper bound on the complexity of k-adaptive testing of cycle freeness in the bounded degree graph model. We begin with a definition of the model.

Let $G = (V, E)$ be a graph with constant degree bound $d < |V|$, represented by its adjacency list; that is, represented by a function $g : V \times d \to V$ such that $g(v, i) = u \in V$ if u is the ith neighbor of v and $g(v, i) = 0$ if v has fewer than i neighbors. A bounded degree graph property $\mathcal{P}$ is a subset of graphs (represented by their adjacency list) that is closed under isomorphism; that is, for every permutation π it holds that $G \in \mathcal{P}$ if and only if $G \in \pi(G)$. The distance of graph G from property $\mathcal{P}$ is the minimal fraction of entries in g one has to change to reach an element of $\mathcal{P}$.

We extend the definition of functional round-adaptive testing algorithms to the bounded degree graph model in the natural way.

▶ **Definition 17** (Round-Adaptive Testing in the Bounded Degree Graph Model)**.** Let $G = (V, E)$ be a graph with constant degree bound $d < |V|$, represented by its adjacency list $g \colon V \times d \to V$, and let $k, q \leq n$. A randomized algorithm is said to be a (k, q)-*round-adaptive* tester for a (bounded degree) graph property $\mathcal{P}$, if, on proximity parameter $\varepsilon \in (0, 1]$ and granted query access to g, the following holds.

1. Query Generation: The algorithm proceeds in $k + 1$ rounds, such that at round $\ell \geq 0$, it produces a set of queries $Q_\ell \stackrel{\text{def}}{=} \{x^{(\ell),1}, \ldots, x^{(\ell),|Q_\ell|}\} \subseteq [n]$ (possibly empty), based on its own internal randomness and the answers to the previous sets of queries $Q_0, \ldots, Q_{\ell-1}$, and receives $f(Q_\ell) = \{g(x^{(\ell),1}), \ldots, g(x^{(\ell),|Q_\ell|})\}$;
2. Completeness: If $G \in \mathcal{P}$, then the algorithm outputs accept with probability at least $2/3$;
3. Soundness: If $\mathrm{dist}(G, \mathcal{P}) > \varepsilon$, then the algorithm outputs reject with probability at least $2/3$.

The *query complexity* q of the tester is the total number of queries made to f, i.e., $q = \sum_{\ell=0}^{k} |Q_\ell|$. If the algorithm returns accept with probability one whenever $f \in \mathcal{P}$, it is said to have *one-sided* error (otherwise, it has *two-sided* error). As before, we will sometimes refer to a tester with respect to proximity parameter ε as an ε-tester.

Next, we define the (bounded degree) graph property of k-cycle freeness.

▶ **Definition 18** (Cycle Freeness). Let $k \in \mathbb{N}$. A graph $G = (V, E)$ is said to be k-cycle free if it does not contain any cycle of length less or equal to k; that is, if for every $t \leq k$ and $v_1, \dots, v_t \in V$ either $(v_t, v_1) \notin E$ or there exists $i \in [t-1]$ such that $(v_i, v_{i+1}) \notin E$.

Finally, we make the following observation, which roughly speaking implies that when surpassing a certain threshold of round adaptivity, testing cycle freeness in the bounded degree graph model becomes "easy."[8]

▶ **Observation 19.** *For every $k \in \mathbb{N}$ there exists a (k, q)-round-adaptive testing algorithm for $(2k+1)$-cycle freeness and $(2k+2)$-cycle freeness in the bounded-degree graph model with query complexity $q = O(d^{k+1}/\varepsilon)$.*

Proof. The algorithm explores the graph in the most natural way: starting from $O(1/\varepsilon)$ "source vertices" selected uniformly at random, it adaptively explore their neighborhoods by querying at each round the neighbors of the previously reached vertices, in a breadth-first-search fashion. If any $(2k+1)$-cycle (resp. $(2k+2)$-cycle) is detected, the algorithm rejects, and accepts otherwise. (Clearly, this tester is one-sided.) It is easy to see that if any of the source vertices belongs to a $(2k+1)$- or $(2k+2)$-cycle, then this bounded-depth BFS will detect it; thus, we only need to argue that if the graph is ε-far from cycle freeness, with constant probability, one of the source vertices will participate in such a cycle. But this is the case, as any such graph must have at least εn vertices participating in a cycle (indeed, otherwise one could "correct" the graph by removing fewer than εdn vertices, contradicting the distance).

Finally, for each source vertex, after k rounds of adaptivity the number of nodes visited is at most $O(d^{k+1})$, hence the claimed query complexity. ◀

5.2 Lower Bounds for Round-Adaptive Testers

In this subsection, we prove the following lemma, which roughly speaking shows that testing $(2k+3)$-cycle freeness is hard for k-round-adaptive testing algorithms.

▶ **Lemma 20.** *Let $k \in \mathbb{N}$ be constant. Then, any (k, q)-round-adaptive testing algorithm for $(2k+3)$-cycle freeness in the bounded-degree graph model must satisfy $q = \Omega(\sqrt{n})$.*

In stark contrast, recall that Observation 19 shows that testing $(2k+2)$-cycle freeness is easy for k-round-adaptive testing algorithms. Indeed, the proof of Theorem 16 follows by combining Observation 19 and Lemma 20 together.

Proof. Proof of Lemma 20 We will show a distribution of $(2k+3)$-cycle free graphs, denoted $\mathcal{Y}$, and a distribution of graphs that are "far" from being $(2k+3)$-cycle free, denoted $\mathcal{N}$, and prove that no (k, q)-round-adaptive testing algorithm can distinguish, with high probability, between $\mathcal{Y}$ and $\mathcal{N}$. Loosely speaking, $\mathcal{Y}$ consists of all graphs whose vertices are covered via

[8] This is a specific case of a more general algorithm for testing subgraph freeness; see e.g. [20, Section 9.2.1].

disjoint $(2k+4)$-cycles, and $\mathcal{N}$ consists of all graphs whose vertices are covered via disjoint $(2k+3)$-cycles.

More accurately, denote by $\mathcal{P}_{t,n,d}$ the subset of n-node graphs with maximum degree at most d that are t-cycle-free. Let $\Sigma_{t,s}$ be the 2-regular graph on st vertices made of s disjoint t-cycles, namely $(v_1,\dots,v_t)$, $(v_{t+1},\dots,v_{2t})$, $(v_{(s-1)t+1},\dots,v_{st})$. Denote also by Is_r the independent set on r vertices. For two graphs G,G' on respectively m and m' vertices and with e and e' edges, we write $G\sqcup G'$ for the graph on $m+m'$ vertices and with $e+e'$ edges obtained by concatenating disjoint copies of G,G'.

For $k=O(1)$, we let $\ell \stackrel{\text{def}}{=} \left\lfloor \frac{n}{(2k+4)} \right\rfloor$, $\ell' \stackrel{\text{def}}{=} \left\lfloor \frac{n}{(2k+3)} \right\rfloor$, and define the two distributions over n-node graphs $\mathcal{Y}$ and $\mathcal{N}$ as follows.

- $\mathcal{Y}$ is the uniform distribution over all isomorphic copies of $G_k^{\textsf{yes}} \stackrel{\text{def}}{=} \Sigma_{(2k+4),\ell} \sqcup \mathrm{Is}_{n-(2k+4)\ell}$;
- $\mathcal{N}$ is the uniform distribution over all isomorphic copies of $G_k^{\textsf{no}} \stackrel{\text{def}}{=} \Sigma_{(2k+3),\ell'} \sqcup \mathrm{Is}_{n-(2k+3)\ell'}$.

The next claim establishes that indeed $\mathcal{Y}$ consists of yes-instances, whereas $\mathcal{N}$ consists of no-instances.

▶ **Claim 21.** *$\mathcal{Y}$ is supported on $\mathcal{P}_{(2k+3),n,d}$, while every graph in the support of $\mathcal{N}$ is $\Omega(1)$-far from $\mathcal{P}_{(2k+3),n,d}$.*

Proof. The first part is obvious, as the only cycles in $G_k^{\textsf{yes}}$ are $(2k+4)$-cycles. As for the second, it immediately follows from observing that $G_k^{\textsf{no}}$ contains ℓ' disjoint $(2k+3)$-cycles, and thus at least ℓ' edges have to be removed to make it $(2k+3)$-cycle free. Thus, $\mathrm{dist}\big(G_k^{\textsf{no}}, \mathcal{P}_{(2k+3),n,d}\big) \geq \frac{\ell'}{dn/2} = \Omega\big(\frac{1}{dk}\big) = \Omega_d(1)$. ◀

Let $\mathcal{T}$ be a deterministic testing algorithm with k rounds of adaptivity and query complexity $q' = o(\sqrt{n})$. The following lemma concludes the proof of Lemma 20 by showing that $\mathcal{T}$ cannot distinguish, with high probability, between graphs in $\mathcal{Y}$ and graphs in $\mathcal{N}$. Denote $\mathcal{T}$'s (disjoint) query sets, per round, by $Q_0,\dots,Q_k \subseteq V$, where a query is a vertex v. Denote the corresponding sets of answers by $A_0,\dots,A_k$, where the answer to a query v consists of the labels of all neighbors of v (i.e., either two or zero vertices). Since $k=O(1)$, without loss of generality, we can assume (by padding) that all query sets have the same size $q \stackrel{\text{def}}{=} |Q_i| = \frac{q'}{k+1} = \Theta(q')$ for every $i\in\{0,\dots,k\}$. Moreover, we can also assume that no vertex is queried twice, i.e. that all Q_i's are disjoint.

▶ **Lemma 22.** $\big|\Pr_{G\sim\mathcal{Y}}\big[\mathcal{T}^G \textit{ accepts}\big] - \Pr_{G\sim\mathcal{N}}\big[\mathcal{T}^G \textit{ accepts}\big]\big| \leq \frac{1}{10}$.

Proof. For $j\in\{0,\dots,k\}$, define by Y_j and N_j the distribution of $(A_0,\dots,A_j)$ when $G\sim\mathcal{Y}$ and when $G\sim\mathcal{N}$, respectively. We shall prove that $\mathrm{d}_{\mathrm{TV}}(Y_k,N_k)\leq\frac{1}{10}$, which by the data processing inequality will imply the claim of Lemma 22.

The high-level idea is that in each round, the tester can either query "fresh" vertices, of which it has no prior information, or query the boundaries (i.e., the direct neighbors) of previously queried vertices. Then, loosely speaking we can argue that, on the one hand, if the total number of queries is $o(\sqrt{n})$, then both for graphs in $\mathcal{Y}$ and $\mathcal{N}$ all queries of "fresh" vertices (obtained during all rounds) with high probability would only fall into previously unattained disjoint cycles, in which case the answer would be a uniform sequence of "fresh" labels. On the other hand, the local view obtained by querying the boundary, using at most k rounds of adaptive queries, of each vertex previously obtained via a "fresh" query (which by the above lies in a cycle wherein the tester has no information of the labels of the other vertices participating in this cycle) is isomorphic to the tail graph over fresh labels, both for instances taken from $\mathcal{Y}$ and $\mathcal{N}$ (that is, we do not have enough adaptive queries to observe a full cycle). The foregoing intuition is formalized below.

For $i \in \{0, \dots, k\}$, define

$$S_i^{\mathrm{f}} \stackrel{\mathrm{def}}{=} Q_i \setminus \cup_{j=0}^{i-1} A_j$$
$$S_i^{\mathrm{b}} \stackrel{\mathrm{def}}{=} Q_i \cap \cup_{j=0}^{i-1} A_j$$

to be, respectively, the set of "entirely fresh" nodes queried at round i (that is, nodes that are not neighbors of any previously queried node), and the set of "boundary nodes" (which are the not-yet-queried nodes neighbors of a previously queried node).

First, we bound the probability that any of the q' queries made "hits" the set of disconnected nodes:

▶ **Claim 23.** *Let $E_1(G)$ denote the event that $\mathcal{T}$ queries an isolated vertex of G, that is $E_1(G) \stackrel{\mathrm{def}}{=} \{\exists i, v \text{ s.t. } v \in Q_i,\ \deg(v) = 0\}$. Then $\Pr_{G\sim\mathcal{Y}}[E_1(G)], \Pr_{G\sim\mathcal{N}}[E_1(G)] = o(1)$.*

Proof. This follows by induction: at step i, conditioned on no isolated node having been queried yet, the algorithm has degree information about $\left|\cup_{j=0}^{i-1} Q_j \bigcup \cup_{j=0}^{i-1} A_j\right| \leq \sum_{j=0}^{i-1} |Q_j| + \sum_{j=0}^{i-1} |A_j| \leq 3q \cdot i$ nodes, so there remain at least $n - 3kq$ nodes on which the algorithm has no degree information at all. Among these, there are $n - (2k+4)\ell \leq (2k+4)$ (or $n - (2k+3)\ell' \leq (2k+3)$, in the no-case) isolated nodes. By symmetry, this means that in the new batch of q queries, the algorithm will query one of these isolated nodes with probability at most $1 - \left(1 - \frac{(2k+4)}{n-3kq-(2k+4)}\right)^q = 1 - \left(1 - \frac{O(1)}{n}\right)^q = O\left(\frac{q}{n}\right) = o(1)$. Therefore, overall there will be an isolated node queried with probability at most $k \cdot o(1) = o(1)$. ◀

Next, we argue that at each step, with overwhelming probability all the "fresh nodes" queried fall in distinct cycles, which have not been attained yet.

▶ **Claim 24.** *Let $E_2(G)$ denote the event that at some round i, one of the queries in S_i^{f} belongs to the same cycle (either a $(2k+4)$- or a $(2k+3)$-cycle, depending on whether the graph is drawn from $\mathcal{Y}$ or $\mathcal{N}$) as one of the previous queries $\bigcup_{j=0}^{i-1} Q_j$. Then $\Pr_{G\sim\mathcal{Y}}[E_2(G)], \Pr_{G\sim\mathcal{N}}[E_2(G)] = o(1)$.*

Proof. We will show that $\Pr_{G\sim\mathcal{Y}}[E_2(G)] = o(1)$; the no-case is similar. For $i \in \{1, \dots, k\}$, let $E_2^{(i)}(G)$ denote the event that at some round i, one of the queries in S_i^{f} belongs to the same cycle as a previous query, so that $E_2(G) = \bigcup_{i=1}^{k} E_2^{(i)}(G)$.

Note that since $|\bigcup_{j=0}^{i-1} Q_j| = iq$, we have $|\bigcup_{j=0}^{i-1} A_j| \leq 2iq$ (and the number of distinct cycles reached is at most $|\bigcup_{j=0}^{i-1} Q_j|$). Therefore, at round i each of the at most q distinct queries in S_i^{f} falls independently in a previously visited cycle with probability upper bounded by

$$\frac{iq \cdot (2k+4)}{n - 3iq} \leq \frac{kq \cdot (2k+4)}{n - 3kq} \leq \frac{2k^2 q}{n}$$

recalling that $q = o(n)$ and $k = O(1)$. A union bound over all at most q queries of S_i^{f}, and then over the k rounds then shows that $\Pr_{G\sim\mathcal{Y}}[E_2(G)] \leq \frac{2k^3q^2}{n} = o(1)$ (since $q = o(\sqrt{n})$). ◀

To conclude the proof, note that by the above, with probability $1 - o(1)$ neither E_1 nor E_2 occurs; that is, none of the isolated vertices was queried, and all the "fresh" queries (during all rounds) fell in previously unattained distinct cycles. In this case, at each round of adaptivity the algorithm can at most discover two new nodes out of every cycle it reached before (by including the one or two end nodes of the current "discovered portion" into S_i^{b}). Therefore, on any cycle ever reached, the (k, q)-round-adaptive testing algorithm can observe at most $2k+2$ nodes (which then form a consecutive path). We show that this implies that the algorithm cannot distinguish between a no-instance and a yes-instance, as loosely speaking, in both cases its local view is of a tail graph over uniformly distributed fresh labels, and so it is unable to determine whether it belongs to a cycle of length $2k+3$ or $2k+4$.

To make the argument more precise, we will actually show a stronger statement; namely, we show that, conditioning on neither E_1 nor E_2 occuring, a simulator with no access to the graph can answer the queries of the testing algorithm in a way that is indistinguishable from the tuple of answers obtained from querying a graph distributed according to either $\mathcal{Y}$ or $\mathcal{N}$. This simulator operates as follows: at round i,

1. Order (arbitrarily) all the nodes of Q_i: $v_1, \dots, v_q$, and initialize the set of available-to-sample nodes $U \leftarrow V \setminus \left(Q_i \cup \bigcup_{j=0}^{i-1} Q_j \cup \bigcup_{j=0}^{i-1} A_j\right)$.
2. Do sequentially the following, for $s = 1 \dots q$:
   - if $v_s \in S_i^{\mathrm{f}}$ (fresh node: no previous neighbors known), pick uniformly at random two distinct nodes u, u' in U_s and return them as answers (i.e., declare them as neighbors of v_s);
   - otherwise, $v_s \in S_i^{\mathrm{b}}$ (boundary node: exactly one already known neighbor, call it u): pick uniformly at random one other node u' in U_s, and return (u, u') as answers;
   - update U by removing u, u': $U \leftarrow U \setminus \{u, u'\}$

It is straightforward to verify that, since we conditioned on $\overline{E_1}$ and $\overline{E_2}$, this simulates exactly the same distribution over nodes (over the choice of G); since this is the same both for $\mathcal{Y}$ and $\mathcal{N}$, we get that $\mathrm{d_{TV}}\big((Y_k \mid \overline{E_1 \cup E_2}), (N_k \mid \overline{E_1 \cup E_2})\big) = 0$, which combined with Claim 23 and Claim 24 finishes the proof. ◀

This concludes the proof of Lemma 20. ◀

6 Some Miscellaneous Remarks

In this section we discuss adaptivity round reductions, as well as a connection to communication complexity, and the relation between round and tail adaptivity. Specifically, in Section 6.1 we show how to simulate k rounds of adaptivity via $k-1$ rounds (at the cost of an increase in query complexity). In Section 6.2 we extend the communication complexity methodology for proving property testing lower bounds [9] to k-round adaptive testers, then sketch an alternative proof of item (2) of Theorem 5 using it; and show how it can also be leveraged to prove a hierarchy of lower bounds on the power of k-adaptive testers for a fundamental class of Boolean functions. Finally, in Section 6.3 we show a separation between the power of round-adaptive and tail-adaptive testers.

6.1 On Simulating k Rounds With Fewer

As mentioned in the beginning of Section 5, in the Boolean setting any adaptive property testing algorithm can be simulated non-adaptively with only an exponential blowup in the query complexity. Phrased differently, this implies that any property of Boolean functions which admits a (k,q)-round-adaptive tester also has a $(0, 2^q - 1)$-round-adaptive tester.

This begs the following more general question: let $\mathcal{P} = \bigcup_n \mathcal{P}_n$ be a property of Boolean functions, such that there exists a (k,q)-round-adaptive tester for $\mathcal{P}$. For $\ell < k$, what upper bound can we obtain on the query complexity q' of the best (ℓ, q')-round-adaptive tester for $\mathcal{P}$?

Denoting by q_ℓ this query complexity, the above discussion immediately implies:

▶ **Fact 25.** *For any $0 \leq \ell \leq k$, one has $q_k \leq q_\ell \leq 2^{q_k} - 1$.*

In what follows, we provide a example of a more fine-grained version of this fact, in the case when $\ell = k-1$ (that is, one wishes to reduce the number of rounds of adaptivity by one).

▶ **Proposition 26.** *For any $0 < k$, one has $q_k \leq q_{k-1} \leq q_k(1+2^{\frac{q_k}{k}})$.*

Proof. Let $\mathcal{T}_k$ be a (k,q)-round-adaptive tester for $\mathcal{P}$, which can be viewed as a distribution over deterministic algorithms. Thus, it is sufficient to explain how to simulate any deterministic algorithm with k rounds of adaptivity by one with ℓ rounds. Fix such a (k,q)-round deterministic algorithm: this can be seen equivalently as a depth-$(k+1)$ binary tree, where each internal node v is labeled by the set of queries Q_v made at that stage, and the leaves are either accept or reject. By assumption, we have that on each path $(v_0, v_1, \ldots, v_k, v^*)$ from the root to a leaf, $\sum_{j=0}^{k} \left|Q_{v_j}\right| \leq q$; moreover, one can assume without loss of generality that this is an equality.

The idea is then to contract, on any path, two consecutive nodes as follows: instead of querying Q_{v_j}, receiving the answers, and then querying the (adaptively chosen) set $Q_{v_{j+1}}$, one can idea query simultaneously Q_{v_j} and the union of all possible sets $Q_{v_{j+1}}$: since the latter depends only on the previous queries, and the only unknown answers are those to the queries in Q_{v_j}, there are at most $2^{\left|Q_{v_j}\right|}$ possibilities for $Q_{v_{j+1}}$. As clearly no matter what $Q_{v_{j+1}}$ would be, its size is at most q, the set $Q'_i = Q_{v_j} \cup \bigcup_{Q:\text{ possible } Q_{v_{j+1}}} Q$ queried has size at most $\left|Q_{v_j}\right| + q2^{\left|Q_{v_j}\right|}$. Thus, by contradicting the two rounds i and $i+1$, one incurs an additional number of queries upper bounded by $q2^{\left|Q_{v_j}\right|} - \left|Q_{v_{j+1}}\right| \leq q2^{\left|Q_{v_j}\right|}$

By an averaging argument, since on every such path we have $\sum_{j=0}^{k} \left|Q_{v_j}\right| = q$, there must exist an index j^* such that $\left|Q_{v_{j^*}}\right| \leq \frac{q}{k+1}$. Since we would like to "contract" rounds j^* and j^*+1 into a single round, we additionally want to ensure $j^* < k$. But similarly, as $\sum_{j=0}^{k-1} \left|Q_{v_j}\right| \leq q$ there exists i^* such that $|Q_{v_{i^*}}| \leq \frac{q}{k}$. We then get an index $i^* < k$ (which depends on the path taken down the tree) to which we can apply the above transformation. That is, whenever the deterministic algorithm is executed it will reach an index $i^* < k$ where it should make $|Q_{v_{i^*}}| \leq \frac{q}{k}$ queries. At that point, it makes instead these queries, along with all queries this should have triggered at the next round, and thus is able to skip round i^*+1 at the price of an additional (at most) $q2^{\frac{q}{k}}$ queries. ◀

▶ Remark. Note that in the above proof, while one can assume without loss of generality that the algorithm always makes exactly q queries, one *cannot* however assume that for any two such paths $(v_0, v_1, \ldots, v_k, v^*)$ and $(u_0, u_1, \ldots, u_k, u^*)$, $\left|Q_{v_j}\right| = \left|Q_{u_j}\right|$ for all $0 \leq j \leq k$. That is, the number of queries made in round j may not be the same depending on the path followed down by the algorithm, but instead depend adaptively on the previous queries made.

The above remark shows the difficulty in extending the proof of Proposition 26 further than a single round. If one is willing to assume that the number of queries at each round is non-adaptive, it becomes possible to obtain a more general statement for $0 \leq \ell < k$; however, it is unclear how to proceed without this extra assumption, leading to the following question:

▶ **Open Problem 3.** *Can one obtain a general round-reduction upper bound for $0 \leq \ell < k$ of the form $q_\ell \leq \phi(q_k, \ell, k)$, improving on Fact 25 for $\ell > 0$?*

6.2 On the Connection with Communication Complexity

As exemplified in the proof of Lemma 8, there exists a striking parallel between the notion of k-round-adaptive testing algorithms, and that of k-round protocols in communication complexity. In this section, we make this parallel rigorous, and give a blackbox reduction between the two that one can leverage to establish lower bounds on k-round-adaptive testing.

In more detail, we build on the communication complexity methodology for proving property testing lower bounds due to [9] (more precisely, to the general formulation of this methodology as laid out in [19]). Although the results stated there hold for non-adaptive lower bounds (in the case of one-way communication or simultaneous message passing) or fully adaptive lower bounds in property testing (in the case of two-way communication), it is easy to obtain their counterpart for k-round-adaptive, given in Theorem 27 below. But first, we need to recall some notations.

In what follows, for a property $\mathcal{P}$, integer k, and parameters $\varepsilon, \delta \in [0,1]$, we write $Q_\delta^{(k)}(\varepsilon, \mathcal{P})$ for the minimum query complexity of any k-round-adaptive tester for $\mathcal{P}$ with error probability δ and distance parameter ε. Given a communication complexity predicate F, we let $\mathrm{CC}_\delta^{(k)}(F)$, $\overrightarrow{\mathrm{CC}}_\delta(F)$, and $\overleftarrow{\mathrm{CC}}_\delta(F)$ denote respectively the minimum communication complexity of a public-coin protocol for F with error δ in (i) k-rounds, (ii) one-way from Alice to Bob, and (iii) one-way from Bob to Alice, respectively (note that the case $\delta = 0$ then corresponds to protocols with perfect completeness).

▶ **Theorem 27.** *Let $\Psi = (P, S)$ be a promise problem such that $P, S \subseteq \{0,1\}^{2n}$, $\mathcal{P} \subseteq \{0,1\}^\ell$ be a property, and $\varepsilon, \delta > 0$. Suppose the mapping $F\colon \{0,1\}^{2n} \to \{0,1\}^\ell$ satisfies the following two conditions:*

1. *for every $(x, y) \in P \cap S$, it holds that $F(x, y) \in \mathcal{P}$;*
2. *for every $(x, y) \in P \setminus S$, it holds that $F(x, y)$ is ε-far from $\mathcal{P}$.*

Then $Q_\delta^{(k)}(\varepsilon, \mathcal{P}) \geq \frac{1}{B+1} \mathrm{CC}_{2\delta}^{(k+2)}(\Psi)$, where $B \stackrel{\text{def}}{=} \max_{i\in[\ell]} \max(\overrightarrow{\mathrm{CC}}_{\frac{\delta}{n}}(F_i), \overleftarrow{\mathrm{CC}}_{\frac{\delta}{n}}(F_i))$ (and $F_i(x, y)$ is the i'th bit of $F(x, y)$). Moreover, if $B' \stackrel{\text{def}}{=} \max_{i\in[\ell]} \max(\overrightarrow{\mathrm{CC}}_0(F_i), \overleftarrow{\mathrm{CC}}_0(F_i))$, then $Q_\delta^{(k)}(\varepsilon, \mathcal{P}) \geq \frac{1}{B'+1} \mathrm{CC}_\delta^{(k+2)}(\Psi)$.

Proof. The proof will be identical to that of [19, Theorem 3.1], where we only need to check that Alice and Bob can each simulate the execution of the property testing algorithm (using their public random coins), answering the queries made to $F(x, y)$ while preserving the number of rounds. Running the testing algorithm, Alice first sends the bits allowing Bob to compute the answers to the first q_0 queries, using her input x and the one-way protocols for the relevant F_i's. Bob then answers with the q_0 bits corresponding to the answers he computed, as well as the bits allowing Alice to compute the answers to the next q_1 queries made by the tester, using now his input y and the one-way protocols for the relevant F_i's. They do so for $k + 1$ rounds of communication in total, until the last player to receive a message gets from the other player both the answers to the queries in Q_{k-1} as well as the bits needed to compute (given their own input) the answers to the last q_k queries. At that point, it only remains to use a last round of communication (the $(k + 2)$'nd) to communicate to the other player the answers to these last q_k queries, so that both Alice and Bob can finish running their copy of the testing algorithm and know the answer.

Note that the number of bits communicated at round $1 \leq i \leq k + 2$ is by definition of B (resp. B') at most $B \cdot q_{i-1} + q_{i-2}$ (resp. $B' \cdot q_{i-1} + q_{i-2}$), so that at most $(B + 1)q$ (resp. $(B' + 1)q$) bits are communicated in total. This concludes the proof. ◀

To illustrate the above methodology, we show how it can be leveraged to prove a hierarchy of lower bounds on the power of k-adaptive testers for testing a very fundamental class of Boolean functions, that of m-linear functions.[9]

[9] We observe that establishing the upper bound counterpart to this result would provide an answer to

▶ **Proposition 28.** *Let* $\mathsf{PAR}_s^n \subseteq 2^{2^n}$ *denote the class of parities of size* s *(over* n *variables), and fix* $m \stackrel{\text{def}}{=} \frac{\sqrt{n}}{2}$*. Then, for any* $0 \leq k \leq \log^* m - 2$*, any* (k,q)*-round-adaptive tester for* PAR_{2m}^n *must satisfy* $q = \Omega\Big(m \log^{(k+2)} m\Big)$*.*

Proof. We will rely on a result of Sağlam and Tardos [37], which implies the following (tight) lower bound on the communication complexity of sparse set-disjointness (DISJ_m^n, where both inputs $x, y \in \{0,1\}^n$ are promised to have Hamming weight m):

▶ **Theorem** (Corollary of [37, Theorem 4])**.** *For any* $1 \leq k \leq \log^* m$*, any* k*-round probabilistic protocol for* $\mathsf{DISJ}_m^{4m^2}$ *with error probability at most* $1/3$ *must have communication* $\Omega\Big(m \log^{(k)} m\Big)$*.*

It then suffices to provide a reduction from $\mathsf{DISJ}_m^{4m^2}$ to testing $\mathsf{PAR}_{2m}^{4m^2}$. We follow the known reduction, as can be found in [9, 14]. Namely, on input $x \in \{0,1\}^n$ (resp. $y \in \{0,1\}^n$), Alice (resp. Bob) forms the parity function χ_x (resp. χ_y). As $|x \oplus y| = |x| + |y| - 2\,|x \cap y| = 2m - 2\,|x \cap y|$, the function $\chi_{x\oplus y}$ is a $2(m - |x \cap y|)$-parity. Moreover, as for any $z \in \{0,1\}^n$ we have $\chi_{x\oplus y}(z) = \chi_x(z) \oplus \chi_y(z)$, each query can be answered (with zero error) by one bit of communication in either direction.

Put in the language of our reduction theorem, $\Psi = (P, S)$ with $P = \{u \in \{0,1\}^n : |u| = m\}^2$ and $S = \{\, (x, y) \in P : \ |x \cap y| \neq 0 \,\}$; while $\ell = 2^n$, $\mathcal{P} = \mathsf{PAR}_{2m}^n \subseteq 2^\ell$; and $F\colon \{0,1\}^{2n} \to \{0,1\}^\ell$ maps (x, y) to the truth table of $\chi_{x\oplus y}$. Since any two distinct parities are at distance $\frac{1}{2}$, we can take any $\varepsilon \leq \frac{1}{2}$. We then have $B' = 1$, and by the theorem above we know that $\mathrm{CC}_{1/3}^{(k+2)}(\Psi) = \Omega\Big(m \log^{(k+2)} m\Big)$ for any $0 \leq k \leq \log^* m - 2$. Invoking Theorem 27 concludes the proof. ◀

We also outline below how Theorem 27 enables us to establish directly the lower bound part of Theorem 5, without relying on the transference theorem for LDTs.

Alternate proof of item (2) of Theorem 5. We start from the same communication complexity problem, "pointer-following," as in Section 4.3. Recall that an input to this problem is a pair of mappings (χ_A, χ_B) with $\chi_A\colon V_A \to V_B$, $\chi_B\colon V_B \to V_A$ (where V_A, V_B are two disjoint sets $\{v_0, \ldots, v_{n/2-1}\}$ and $\{v_{n/2}, \ldots, v_{n-1}\}$ of nodes of cardinality $n/2$). The function to compute is the indicator of the event where the vertex $v_i = \chi^{(k+2)}(v_0)$, reached by following the path of length $k+2$ starting at v_0, has an even index i.

We define $\Psi = (P, S)$ by setting $P \stackrel{\text{def}}{=} \{\, (\chi_A, \chi_B) : \ \chi_A\colon V_A \to V_B, \chi_B\colon V_B \to V_A \,\}$ and $S \stackrel{\text{def}}{=} \big\{\, (\chi_A, \chi_B) \in P : \ \chi^{(k+2)}(v_0) \text{ is an even-index node} \,\big\}$. The property is, as in Theorem 5, the subset of codewords $\mathcal{C}_{f_{k+1}}$ corresponding to the function $f_{k+1}\colon \mathbb{F}_n^n \to \{0,1\}$ of Section 4 (for a good code $C\colon \mathbb{F}_n^n \to \mathbb{F}_n^m$ as in Section 4.4.2).

Identifying $V \stackrel{\text{def}}{=} V_A \cup V_B = \{v_0, \ldots, v_{n-1}\}$ with $\mathbb{F}_n$ in the natural way, we define the mapping $F\colon P \to \mathbb{F}_n^n$ by

$$F(\chi_A, \chi_B) = (\chi_A(v_0), \chi_A(v_1) \ldots, \chi_A(n/2-1), \chi_B(n/2), \ldots, \chi_B(n-1)).$$

From there, it is easy to check that by construction, (i) $(\chi_A, \chi_B) \in P \cap S$ implies $F(\chi_A, \chi_B) \in \mathcal{C}_{f_{k+1}}$, while (ii) $(\chi_A, \chi_B) \in P \setminus S$ implies that $F(\chi_A, \chi_B)$ is ε_0-far from $\mathcal{C}_{f_{k+1}}$, for some constant $\varepsilon_0 > 0$ depending on the code C. Observing that $B' \stackrel{\text{def}}{=} \max_{i\in[\ell]} \max(\overrightarrow{\mathrm{CC}}_0(F_i), \overleftarrow{\mathrm{CC}}_0(F_i)) =$

Open Problem 1, although one rather weak quantitatively. It also, as a special case, would separate adaptive and non-adaptive testing of m-linearity for $m = o(n)$, a longstanding open question [10, 6].

$O(\log n)$ (as each F_i is specified by $O(\log n)$ bits), we can invoke Theorem 27 along with the communication complexity lower bound of Theorem 9 to obtain that $Q^{(k)}_{1/3}(\varepsilon_0, \mathcal{C}_{f_{k+1}}) \geq \frac{1}{B'+1} \operatorname{CC}^{(k+2)}_{1/3}(\Psi) = \Omega\Big(\frac{n}{k^2 \log n}\Big)$. ◀

6.3 On the Relative Power of Round-Adaptive and Tail-Adaptive Testers

In this section, we show that the two notions of round- and tail-adaptive testers we introduced are not equivalent. As mentioned in Section 3, while round-adaptive testers are at least as powerful as tail-adaptive ones, there exist properties for which the separation is strict:

▶ **Theorem 29.** *Fix any $\alpha \in (0,1)$. There exists a constant $\beta \in (0,1)$ such that, for every $n \in \mathbb{N}$, the following holds. For every integer $0 \leq k \leq n^\beta$, there exists a property $\mathcal{P}_k \subseteq \mathbb{F}_n^{n^{1+\alpha}}$ such that, for any constant $\varepsilon \in (0,1]$,*

1. *there exists a $(k, \tilde{O}(k))$-round-adaptive (one-sided) tester for $\mathcal{P}_k$; yet*
2. *any (k,q)-tail-adaptive (two-sided) tester for $\mathcal{P}_k$ must satisfy $q = \Omega(n)$.*

Proof Sketch. The argument is very similar to that of Theorem 5, and follows the same overall structure. Namely, we slightly modify the k-iterated function f_k of Section 4 (which *was* computable by a $(k, k+1)$-tail-adaptive algorithm) to rule out tail-adaptive algorithms but not round-adaptive ones: that is, we define the function $f'_k \colon \mathbb{F}_n^n \to \mathbb{F}_n$ by

$$f'_k(x) = \begin{cases} 1 & \text{if } x_{x,g_{k-1}(x)} = x_{x,g_{k-1}(x)+1 \bmod n} \\ 0 & \text{otherwise.} \end{cases}$$

(Perhaps more clearly, f'_k is computed by iterating the pointer function k times, and then checking if the value x_i at the final coordinate $i \in [n]$ reached, and the value x_{i+1} at the adjacent coordinate $i+1$, are equal.) It is not hard to see that the counterparts of Claim 7 and Lemma 8 still hold for f'_k: first, the function is still easy to compute by $(k, k+2)$-round-adaptive algorithms. However, because the very last round requires 2 queries and not one (to query x_i and x_{i+1}, once the value of $i = g_{k-1}(x)$ has been obtained), tail-round-adaptive algorithms are no longer able to leverage this, and analogously to Lemma 8 we can conclude that there is no $(k, o(n/(k^2 \log n)))$-round-adaptive (randomized) LDT algorithm which computes f'_k. It then only remains to lift this DT separation to property testing: we can do this as before (noting, in the case of lifting the lower bound, that the reduction of Lemma 14 preserves the number of queries per round, and thus the "tailness" of the algorithm). ◀

Acknowledgments. We are grateful to Oded Goldreich for suggesting cycle freeness as a candidate natural property for proving an adaptivity hierarchy theorem, as well as for enlightening conversations that significantly contributed to this work; and wish to thank Rocco Servedio for helpful comments on an earlier version of this paper.

References

1. Scott Aaronson and Avi Wigderson. Algebrization: a new barrier in complexity theory. In *Proceedings of STOC*, pages 731–740, 2008. doi:10.1145/1374376.1374481.
2. Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000. doi:10.1007/s004930070001.
3. Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, 36(4):889–974, 2006. doi:10.1137/S0097539705446810.

4 Eli Ben-Sasson, Prahladh Harsha, and Sofya Raskhodnikova. Some 3CNF properties are hard to test. *SIAM J. Comput.*, 35(1):1–21, 2005. doi:10.1137/S0097539704445445.

5 Arnab Bhattacharyya and Yuichi Yoshida. *Property Testing*. Forthcoming, 2017. URL: https://propertytestingbook.wordpress.com/.

6 Abhishek Bhrushundi, Sourav Chakraborty, and Raghav Kulkarni. Property testing bounds for linear and quadratic functions via parity decision trees. In *CSR*, volume 8476 of *Lecture Notes in Computer Science*, pages 97–110. Springer, 2014.

7 Eric Blais. Improved bounds for testing juntas. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 317–330. Springer, 2008.

8 Eric Blais. Testing juntas nearly optimally. In *Proceedings of STOC*, pages 151–158. ACM, 2009.

9 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012. doi:10.1007/s00037-012-0040-x.

10 Eric Blais and Daniel M. Kane. Tight bounds for testing k-linearity. In *Proceedings of APPROX-RANDOM*, volume 7408 of *Lecture Notes in Computer Science*, pages 435–446. Springer, 2012.

11 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

12 Joshua Brody, Kevin Matulef, and Chenggang Wu. Lower bounds for testing computability by small width OBDDs. In *TAMC*, volume 6648 of *Lecture Notes in Computer Science*, pages 320–331. Springer, 2011.

13 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.

14 Harry Buhrman, David García-Soriano, Arie Matsliah, and Ronald de Wolf. The non-adaptive query complexity of testing k-parities. *Chicago J. Theor. Comput. Sci.*, 2013, 2013.

15 Clément L. Canonne. A Survey on Distribution Testing: your data is Big. But is it Blue? *Electronic Colloquium on Computational Complexity (ECCC)*, 22:63, April 2015.

16 Xi Chen, Rocco A. Servedio, Li-Yang Tan, Erik Waingarten, and Jinyu Xie. Settling the query complexity of non-adaptive junta testing. In *Computational Complexity Conference (CCC)*, 2017. To appear.

17 Dingzhu Du and Frank K. Hwang. *Combinatorial Group Testing and Its Applications*. Applied Mathematics. World Scientific, 2000. URL: https://books.google.com/books?id=KW5-CyUUOggC, doi:10.1142/4252.

18 Oded Goldreich, editor. *Property Testing – Current Research and Surveys [outgrow of a workshop at the Institute for Computer Science (ITCS) at Tsinghua University, January 2010]*, volume 6390 of *Lecture Notes in Computer Science*. Springer, 2010. doi:10.1007/978-3-642-16367-8.

19 Oded Goldreich. On the communication complexity methodology for proving lower bounds on the query complexity of property testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:73, 2013.

20 Oded Goldreich. *Introduction to Property Testing*. Forthcoming, 2017. URL: http://www.wisdom.weizmann.ac.il/~oded/pt-intro.html.

21 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, July 1998.

22 Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong locally testable codes with relaxed local decoders. In *30th Conference on Computational Complexity (CCC 2015)*, volume 33 of *LIPIcs*, pages 1–41. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CCC.2015.1.

23 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7:20, 2000.

24 Oded Goldreich and Dana Ron. Algorithmic aspects of property testing in the dense graphs model. *SIAM J. Comput.*, 40(2):376–445, 2011. `doi:10.1137/090749621`.

25 Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Struct. Algorithms*, 23(1):23–57, 2003. `doi:10.1002/rsa.10078`.

26 Tom Gur and Ron D. Rothblum. A hierarchy theorem for interactive proofs of proximity. 2017. The 8th Innovations in Theoretical Computer Science (ITCS 2017) conference (to appear).

27 Piotr Indyk, Eric Price, and David P. Woodruff. On the power of adaptivity in sparse recovery. In *Proceedings of FOCS*, pages 285–294, 2011. `doi:10.1109/FOCS.2011.83`.

28 Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. Testing ± 1-weight halfspace. In *Proceedings of APPROX-RANDOM*, volume 5687 of *Lecture Notes in Computer Science*, pages 646–657. Springer, 2009.

29 Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM Journal on Computing*, 22(1):211–219, February 1993. `doi:10.1137/0222016`.

30 Christos H. Papadimitriou and Michael Sipser. Communication complexity. In *Proceedings of STOC*, Proceedings of STOC, pages 196–200, New York, NY, USA, 1982. ACM. `doi:10.1145/800070.802192`.

31 Sofya Raskhodnikova and Adam D. Smith. A note on adaptivity in testing properties of bounded degree graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 13(089), 2006. URL: `http://eccc.hpi-web.de/eccc-reports/2006/TR06-089/index.html`.

32 Dana Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008. `doi:10.1561/2200000004`.

33 Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5(2):73–205, 2009. `doi:10.1561/0400000029`.

34 Dana Ron and Rocco A. Servedio. Exponentially improved algorithms and lower bounds for testing signed majorities. In *Proceedings of SODA*, pages 1319–1336. SIAM, 2013.

35 Dana Ron and Gilad Tsur. Testing computability by width-two OBDDs. *Theor. Comput. Sci.*, 420:64–79, 2012.

36 Ronitt Rubinfeld and Madhu Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

37 Mert Sağlam and Gábor Tardos. On the communication complexity of sparse set disjointness and exists-equal problems. In *Proceedings of FOCS*, pages 678–687. IEEE Computer Society, 2013.

38 Rocco A. Servedio, Li-Yang Tan, and John Wright. Adaptivity helps for testing juntas. In *30th Conference on Computational Complexity (CCC 2015)*, volume 33 of *LIPIcs*, pages 264–279. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. `doi:10.4230/LIPIcs.CCC.2015.264`.

39 Roei Tell. Deconstructions of reductions from communication complexity to property testing using generalized parity decision trees. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:115, 2014.