Local Lipschitz Filters for Bounded-Range Functions with Applications to Arbitrary Real-Valued Functions

Jane Lange* Ephraim Linder† Sofya Raskhodnikova‡ Arsen Vasilyan§

Abstract

We study local filters for the Lipschitz property of real-valued functions $f : V \to [0, r]$, where the Lipschitz property is defined with respect to an arbitrary undirected graph $G = (V, E)$. We give nearly optimal local Lipschitz filters both with respect to ℓ_1-distance and ℓ_0-distance. Previous work only considered unbounded-range functions over $[n]^d$. Jha and Raskhodnikova (SICOMP '13) gave an algorithm for such functions with lookup complexity exponential in d, which Awasthi et al. (ACM Trans. Comput. Theory) showed was necessary in this setting. We demonstrate that important applications of local Lipschitz filters can be accomplished with filters for functions whose range is bounded in $[0, r]$. For functions $f : [n]^d \to [0, r]$, we achieve running time $(d^r \log n)^{O(\log r)}$ for the ℓ_1-respecting filter and $d^{O(r)} \operatorname{polylog} n$ for the ℓ_0-respecting filter, thus circumventing the lower bound. Our local filters provide a novel Lipschitz extension that can be implemented locally. Furthermore, we show that our algorithms are nearly optimal in terms of the dependence on r for the domain $\{0, 1\}^d$, an important special case of the domain $[n]^d$. In addition, our lower bound resolves an open question of Awasthi et al., removing one of the conditions necessary for their lower bound for general range. We prove our lower bound via a reduction from distribution-free Lipschitz testing and a new technique for proving hardness for *adaptive* algorithms.

Finally, we provide two applications of our local filters to real-valued functions, with no restrictions on the range. In the first application, we use them in conjunction with the Laplace mechanism for differential privacy and noisy binary search to provide mechanisms for privately releasing outputs of black-box functions, even in the presence of malicious clients. In particular, our differentially private mechanism for arbitrary real-valued functions runs in time $2^{\operatorname{polylog} \min(r, nd)}$ and, for honest clients, has accuracy comparable to the Laplace mechanism for Lipschitz functions, up to a factor of $O(\log \min(r, nd))$. In the second application, we use our local filters to obtain the first nontrivial tolerant tester for the Lipschitz property. Our tester works for functions of the form $f : \{0, 1\}^d \to \mathbb{R}$, makes $2^{\widetilde{O}(\sqrt{d})}$ queries, and has tolerance ratio 2.01. Our applications demonstrate that local filters for bounded-range functions can be applied to construct efficient algorithms for arbitrary real-valued functions.

*MIT, jlange@mit.edu. Supported in part by NSF Graduate Research Fellowship under Grant No. 2141064, NSF Awards CCF-2006664, DMS-2022448, CCF-2310818 and Microsoft Research. Part of this work was conducted while the author was visiting the Simons Institute for the Theory of Computing.

†ejlinder@bu.edu. Boston University, Department of Computer Science.

‡sofya@bu.edu. Boston University, Department of Computer Science.

§UC Berkeley. arsen@berkeley.edu. Supported in part by NSF awards CCF-2006664, DMS-2022448, CCF-1565235, CCF-1955217, CCF-2310818, Big George Fellowship and Fintech@CSAIL. Part of this work was conducted while the author was visiting the Simons Institute for the Theory of Computing.

1 Introduction

We study local Lipschitz filters for real-valued functions. Local Lipschitz filters were first investigated by Jha and Raskhodnikova [JR13] who motivated their research by an application in private data analysis. Intuitively, a local filter for some property of functions (in our case, the Lipschitz property) is a randomized algorithm that gets oracle access to a function f and locally reconstructs the desired property in the following sense: it provides query access to a related function g that is guaranteed to have the property (in our case, guaranteed to be Lipschitz). The implicit output function g may depend on the internal randomness of the algorithm, but not on the order of queries. When the input function f has the desired property, then $g = f$. If, in addition, the distance between f and g is relatively small compared to the distance from f to the nearest function with the desired property, the filter is called *distance-respecting*. The goal in the design of local filters is to minimize the running time and the number of *lookups*[1], i.e., oracle calls to the input function f.

The computational task performed by local filters is called *local reconstruction*. It was introduced by Saks and Seshadhri [SS10] and is one of the fundamental tasks studied in the area of local computation [RTVX11, ARVX12] and sublinear-time algorithms. It has been studied for properties of functions, including monotonicity [SS10, BGJ$^+$12, AJMR15, LRV22, LV23] and the Lipschitz property [JR13, AJMR15], as well as for properties of graphs [CGR13].

Local filters are useful in applications where some algorithm $\mathcal{A}$ computing on a large dataset requires that its input satisfy a certain property. For example, in the application to privacy, which we will discuss in detail later, correctness of algorithm $\mathcal{A}$ is contingent upon the input function f being Lipschitz. In such applications, rather than directly relying on the oracle for f, algorithm $\mathcal{A}$ can access its input via a local filter that guarantees that the output will satisfy the desired property, modifying f on the fly if necessary. Local filters can also be used in distributed settings, where multiple processes access different parts of the input, as well as in other applications described in previous work [SS10, BGJ$^+$12, JR13, AJMR15]. Local reconstruction is also naturally related to other computational tasks and, for example, has been recently used to improve learning algorithms for monotone functions [LRV22, LV23].

1.1 Our Contributions

We demonstrate that important applications of local Lipschitz filters can be accomplished with computational objects that are much weaker than local Lipschitz filters for general functions: it suffices to construct local Lipschitz filters for bounded-range functions. This holds even for applications that deal with arbitrary real-valued functions, with no a priori bound on the range. We achieve efficient local Lipschitz filters for bounded-range functions, circumventing the existing lower bounds that are exponential in the dimension and enabling applications to real-valued functions, with no restriction on the range.

1.1.1 Local Lipschitz Filters

Motivated by the applications, we consider functions over $[n]^d$, where $[n]$ is a shorthand for $\{1, \ldots, n\}$. A function $f : [n]^d \to \mathbb{R}$ is called *c-Lipschitz* if increasing or decreasing any coordinate by one can only change the function value by c. The parameter c is called the *Lipschitz constant* of f. A 1-Lipschitz function is simply referred to as Lipschitz[2]. Intuitively, changing the argument to the Lipschitz function by a small amount does not significantly change the value of the function.

In previous work, only unbounded-range functions were considered in the context of Lipschitz reconstruction. Jha and Raskhodnikova [JR13] obtained a deterministic local filter that runs in time $O((\log n+1)^d)$. This direction of research was halted by a strong lower bound obtained by Awasthi et al. [AJMR16]. They showed that every local Lipschitz filter, even with significant additive error, needs exponential in the dimension d number of lookups.

We demonstrate that important applications of local Lipschitz filters can be accomplished with filters for functions whose range is bounded in $[0, r]$. By focusing on this class of functions, we circumvent the lower bound from [AJMR16] and achieve running time polynomial in d for constant r. Moreover, our filters satisfy additional accuracy guarantees compared to the filter in [JR13], which is only required to (1) give access to a Lipschitz function g; (2) ensure that $g = f$ if the input function f is Lipschitz. Our filters achieve an additional feature of being *distance-respecting*, i.e., they ensure that g is close to f. We provide this feature w.r.t. both ℓ_1 and ℓ_0-distance. The ℓ_1-distance between functions f and g is defined by $\|f - g\|_1 = \mathbb{E}_x[|f(x) - g(x)|]$, and the

[1] Oracle calls *by* the filter are called *lookups* to distinguish them from the queries made *to* the filter.

[2] In this work, we focus on reconstruction to Lipschitz functions. All our results extend to the class of c-Lipschitz functions via scaling all function values by a factor of c.

ℓ_0-distance is defined by $\|f-g\|_0 = \Pr_x[f(x) \neq g(x)]$, where the expectation and the probability are taken over a uniformly distributed point in the domain. The distance of f to Lipschitzness is defined as the minimum over all Lipschitz functions g of the distance from f to g, and can be considered with respect to both norms. Our filters are *distance-respecting* in the following sense: the distance between the input function f and the output function g is at most twice the distance of f to the Lipschitz property (w.h.p.).

Our algorithms work for functions over general graphs. To facilitate comparison with prior work and our lower bound, we state their guarantees only for the $[n]^d$ domain. (This domain can be represented by the d-dimensional hypergrid graph $\mathcal{H}_n^d$.) Our first local Lipschitz filter is distance-respecting with respect to the ℓ_1-distance.

Theorem 1.1. (Informal version of Theorem 2.1 (ℓ_1-filter)) *There is an algorithm $\mathcal{A}$ that, given lookup access to a function $f : [n]^d \to [0, r]$, a query $x \in [n]^d$, and a random seed $\rho \in \{0, 1\}^\star$, has the following properties:*

- ***Efficiency:*** *$\mathcal{A}$ has lookup and time complexity $(d^r \cdot \operatorname{polylog} n)^{O(\log r)}$ per query.*
- ***Consistency:*** *With probability at least $1 - n^{-d}$ over the choice of ρ, algorithm $\mathcal{A}$ provides query access to a 1.01-Lipschitz function g_ρ with $\|g_\rho - f\|_1$ at most twice the ℓ_1-distance from f to Lipschitzness.*

  If f is Lipschitz, then the filter outputs $f(x)$ for all queries x and random seeds.

Our second local Lipschitz filter is distance-respecting w.r.t. ℓ_0. Unlike the filter in Theorem 2.1, the ℓ_0-respecting filter provides access to a 1-Lipschitz function.

Theorem 1.2. (Informal version of Theorem 3.1 (ℓ_0-filter)) *There is an algorithm $\mathcal{A}$ that, given lookup access to a function $f : [n]^d \to [0, r]$, a query $x \in [n]^d$, and a random seed $\rho \in \{0, 1\}^\star$, has the following properties:*

- ***Efficiency:*** *$\mathcal{A}$ has lookup and time complexity $d^{O(r)} \operatorname{polylog}(n)$.*
- ***Consistency:*** *With probability at least $1 - n^{-d}$ over the choice of ρ, algorithm $\mathcal{A}$ provides query access to a 1-Lipschitz function g_ρ with $\|g_\rho - f\|_0$ at most twice the ℓ_0-distance from f to Lipschitzness.*

  If f is Lipschitz, then the filter outputs $f(x)$ for all queries x and random seeds.

An important special case of the hypergrid domain is the *hypercube*, denoted $\mathcal{H}^d$. (It corresponds to the case $n = 2$, but its vertex set is usually represented by $\{0, 1\}^d$ instead of $[2]^d$.) Prior to our work, no local Lipschitz filter for the hypercube domain could avoid lookups on the entire domain (in the worst case). Our algorithms do that for the case when $r \leq d/\log^2 d$. Moreover, we show that our filters are nearly optimal in terms of their dependence on r for this domain. Our next theorem shows that the running time of $d^{\widetilde{\Omega}(r)}$ is unavoidable in Theorems 2.1 and 3.1; thus, our filters are nearly optimal. Moreover, even local Lipschitz filters that are not distance-respecting (as in [JR13]) must still run in time $d^{\widetilde{\Omega}(r)}$.

Theorem 1.3. (Informal version of Theorem 4.1 (Filter lower bound)) *Let $\mathcal{A}(x, \rho)$ be an algorithm that, given lookup access to a function $f : \{0, 1\}^d \to [0, r]$, a query $x \in \{0, 1\}^d$, and a random seed $\rho \in \{0, 1\}^*$, has the following property:*

- ***Weak Consistency:*** *With probability at least $\frac{3}{4}$ over the choice of ρ, the algorithm $\mathcal{A}(\cdot, \rho)$ provides query access to a 1-Lipschitz function g_ρ such that whenever f is 1-Lipschitz $g_\rho = f$.*

Then, for all integers $r \geq 4$ and $d \geq \Omega(r)$, there exists a function $f : \{0, 1\}^d \to [0, r]$ for which the lookup complexity of $\mathcal{A}$ is $(\frac{d}{r})^{\Omega(r)}$.

The lower bound for local Lipschitz filters in [AJMR15] applies only to filters that are guaranteed to output a Lipschitz function (or a Lipschitz function with small error in values) for all random seeds. They can still have a constant error probability, but the only mode of failure allowed is returning a function that is far from f when f already satisfied the property. Awasthi et al. [AJMR15] suggest as a future research direction to consider local filters whose output does not satisfy the desired property $\mathcal{P}$ with small probability and mention that their techniques do not work for this case. We overcome this difficulty by providing different techniques that work for filters whose output fails to satisfy the Lipschitz property with constant error probability. In particular, Theorem 4.1 applied with $r = \Theta(d)$ yields the lookup lower bound of $2^{\Omega(d)}$, as in [AJMR15], but without their restriction on the mode of failure, thus answering their open question.

1.1.2 Applications We showcase two applications of our local Lipchitz filters: to private data analysis and to tolerant testing. Both of them deal with real-valued functions with no a priori bound on the range.

Application to black-box privacy. Our first application is for providing differentially private mechanisms for releasing outputs of black-box functions, even in the presence of malicious clients. Differential privacy, introduced by [DMNS06], is an accepted standard of privacy protection for releasing information about sensitive datasets. A sensitive dataset can be modeled as a point x in $\{0,1\}^d$, representing whether each of the d possible types of individuals is present in the data[3]. More generally, it is modeled as a point x in $[n]^d$ that represents a histogram counting the number of individuals of each type.

Definition 1.1. (Differential Privacy) *Two datasets $x, x' \in [n]^d$ are* neighbors *if vertices x, x' are neighbors in the hypergrid $\mathcal{H}_n^d$. For privacy parameters $\varepsilon > 0$ and $\delta \in (0,1)$, a randomized mechanism $\mathcal{M} : [n]^d \to \mathbb{R}$ is* (ε, δ)-differentially private *if, for all neighboring $x, x' \in [n]^d$ and all measurable sets $Y \subset \mathbb{R}$,*

$$\Pr[\mathcal{M}(x) \in Y] \leq e^{\varepsilon} \Pr[\mathcal{M}(x') \in Y] + \delta.$$

When $\delta = 0$, then we call $\mathcal{M}$ purely differentially private*; otherwise, it is* approximately differentially private.

A statistic (or any information about the sensitive dataset) is modeled as a function $f(x)$. One of the most commonly used building blocks in the design of differentially private algorithms is the Laplace mechanism[4]. The Laplace mechanism computing on a sensitive dataset x can approximate the value $f(x)$ for a desired c-Lipschitz function f by adding Laplace noise proportional to c to the true value $f(x)$. The noisy value is safe to release while satisfying differential privacy.

Multiple systems that allow analysts to make queries to a sensitive dataset while satisfying differential privacy have been implemented, including PINQ [McS10], Airavat [RSK+10], Fuzz [HPN11], and PSI [GHK+16]. They all allow releasing approximations to (some) real-valued functions of the dataset. In these implementations, the client sends a program to the server, requesting to evaluate it on the dataset, and receives the output of the program with noise added to it. The program f can be composed from a limited set of trusted built-in functions, such as sum and count. In addition, f can use a limited set of (untrusted) data transformations, such as combining several types of individuals into one type, whose Lipschitzness can be enforced using programming languages tools.

The limitation of the existing systems is that the functionality of the program is restricted by the set of trusted built-in functions available and the expressivity of the programming languages tools. Ideally, future systems would allow analysts to query arbitrary functions, specified as a black box. One reason for the black-box specification is to allow the clients to construct arbitrarily complicated programs. Another reason is to allow researchers analyzing sensitive datasets to obfuscate their programs in order to hide what analyses they are running from the data curator and their competitors.

The difficulty with allowing general queries is that when f (supplied by a distrusted client) is given as a general purpose program, it is hard to compute its least Lipschitz constant, or even an upper bound on it. The data curator can ask the client to supply the Lipschitz constant for the query function f. However, as noted in [JR13], even deciding if f has Lipschitz constant at most c is NP-hard for functions over the finite domains we study (if f is specified by a circuit). Applying the Laplace mechanism with c smaller than a Lipschitz constant (if the client supplied incorrect c) would result in a privacy breach, while applying it with a generic upper bound on the least Lipschitz constant of f would result in overwhelming noise. One reason that a client might supply incorrect c is simply because analyzing Lipschitz constants is difficult even for specialists (see [CSVW22] for significant examples of underestimation of Lipschitz constants in the implementations of the simplest functions, such as sums, in differentially private libraries). Another reason is that client could lie in order to gain access to sensitive information[5].

[3]A point $x \in \{0,1\}^d$ can also represent a dataset containing data of d individuals, with one bit per individual: e.g., $x_i = 1$ could indicate that the individual i has a criminal record or some illness.

[4]Gaussian mechanism is another popular differentially private algorithm that calibrates noise to the Lipschitz constant of f; for simplicity, we focus on the Laplace mechanism as a canonical example.

[5]We give a specific example when f has domain $\{0,1\}^d$ and a small range. Suppose $x_i = 1$ if the individual i has some illness that would disqualify them from getting a good rate on insurance and 0 otherwise. Say, a client would like to determine the secret bit of individual i in the dataset. The client can submit a function f such that $f(x) = 0$ if $x_i = 0$ and $f(x) = 10/\varepsilon$ if $x_i = 1$, obfuscated or with really complicated code. The range of the function is $[0, 10/\varepsilon]$. If the data curator (incorrectly) believes that the function is 1-Lipschitz and uses the Laplace mechanism (formally specified in Lemma 5.1) with noise parameter $1/\varepsilon$ then the client will be able to figure out the bit x_i with high probability, violating the privacy of i in the strongest sense.

To the best of our knowledge, no existing (ε, δ)-DP mechanisms for the black-box privacy problem simultaneously achieve a runtime that is polynomial in d and $\log \frac{1}{\delta}$ while providing an accuracy guarantee that is comparable to the Laplace mechanism. One solution to the black-box privacy problem with the untrusted client can be obtained using the propose-test-release method of Dwork and Lei [DL09] and is described in [CD14]. However, the runtime of this mechanism is $d^{O(\frac{1}{\varepsilon}\log\frac{1}{\delta})}$ which, when $\delta = \frac{1}{\text{poly } d}$ is $d^{\Omega(\log d)}$, and it remains the same even when the input function has bounded range[6] (note that the propose-test-release method does not yield a local Lipschitz filter.) A second solution to the black-box privacy problem was recently proposed in [KL23]. They present a mechanism called "TAHOE" which runs in time $d^{O(\frac{1}{\varepsilon}\log\frac{1}{\delta})}$ and outputs an answer with significantly more noise than the Laplace mechanism for 1-Lipschitz functions. The advantage of TAHOE is that the algorithm need only query the function on subsets of the input. Another solution to the black-box privacy problem was proposed by Jha and Raskhodnikova [JR13] who designed the following *filter mechanism*: *A client who does not have direct access to x can ask the data curator for information about the dataset by specifying a Lipschitz*[7] *function f. The data curator can run a local filter to obtain a value $g(x)$, where g is guaranteed to be Lipschitz. Then the curator can use the Laplace mechanism and release the obtained noisy value.* If the client is truthful (i.e., the function is Lipschitz), then, assuming that the local filter gives access to $g = f$ in the case that f is already Lipschitz, the accuracy guarantee of the filter mechanism is inherited from the Laplace mechanism. However, if the client is lying about f being Lipschitz, the filter ensures that privacy is still preserved. Observe that the running time and accuracy of the filter mechanism directly depends on the running time and accuracy of the local Lipschitz filter. The lower bound by Awasthi et al. [AJMR16] on the complexity of local Lipschitz filters implies $2^{\Omega(d)}$ running time for filter mechanisms, even for releasing functions of the form $f : \{0,1\}^d \to \mathbb{R}$.

We provide a mechanism for privately releasing outputs of black-box functions, even in the presence of malicious clients, in time that is quasi-polynomial in the dimension d while providing accuracy comparable to the Laplace mechanism. For bounded-range functions, the running time of our mechanism is polynomial in d and $\log \frac{1}{\delta}$. We bypass the lower bound in [AJMR16] by using the filter mechanism for bounded-range functions repeatedly to simulate a noisy binary search. Our mechanism needs only query access to the input function, that is, it can be specified as a black box (e.g., as a complicated or obfuscated program).

Theorem 1.4. (Informal version of Theorem 5.2 (Binary search filter mechanism)) *For all $\varepsilon > 0$ and $\delta \in (0,1)$, there exists an (ε, δ)-differentially private mechanism $\mathcal{M}$ that gets lookup access to a function $f : [n]^d \to [0, r]$ and has the following properties. Let $\kappa = \log \min(r, nd)$.*

- ***Efficiency:*** *The lookup and time complexity of $\mathcal{M}$ are $d^{O(\frac{1}{\varepsilon}\kappa \log \kappa)}\,\text{polylog}\,\frac{n}{\delta}$.*
- ***Accuracy:*** *If f is Lipschitz, then for all $x \in [n]^d$, we have $\mathcal{M}(x) \sim f(x) + \text{Laplace}(\frac{\kappa}{\varepsilon})$ with probability at least 0.99.*

In Section 5, we state and prove more detailed guarantees for black-box privacy mechanisms. In particular, our guarantees are stronger for the case when the client can provide an upper bound r on the range diameter that is significantly smaller than nd. There are many bounded-range functions that are hard to implement with a fixed set of trusted functions (and thus they are not implemented in current systems). One primitive often used in differentially private algorithms is determining whether the secret dataset x is far from a specified set S (where "far" means that many records in x would have to change in order to obtain S). The set S could capture datasets with the desired property or satisfying a certain hypothesis. For instance, S is the set of datasets with no outliers in Brown et al. [BGS$^+$21]. To solve this, the client could submit a function $f : [n]^d \to [0, 10/\varepsilon]$ that outputs $\max(\text{distance}(x, S), 10/\varepsilon)$, where ε is the privacy parameter. The range of f is $[0, 10/\varepsilon]$. Here n and d can be huge, whereas ε could be, say 1/10 (a typical value of ε used in industry today is even larger than that). Since f is 1-Lipschitz, it can be released with sufficient accuracy to determine whether x is far from S with high probability.

---

[6]Since the mechanism stated in [CD14, Ch. 7.3, algorithm 13] computes the distance to the nearest "unstable" point, the runtime is actually n^d. However, the following small modification suffices to obtain the runtime of $d^{O(\frac{1}{\varepsilon}\log\frac{1}{\delta})}$. When releasing a noisy "distance to the nearest unstable instance" $\hat{d}$, one can add noise from a Laplace distribution truncated to $\pm\frac{1}{\varepsilon}\log\frac{1}{\delta}$ instead of a regular Laplace distribution. As a result, the mechanism need only consider points at distance at most $\frac{2}{\varepsilon}\log\frac{1}{\delta}$.

[7]If the function is c-Lipschitz, it can be rescaled by dividing by c.

Both of our filters can be used in the filter mechanism, and we show the type of resulting guarantees for bounded-range functions in Section 5. The work of [JR13] provides a Lipschitz filter as well, intended to be used in the filter mechanism. The filter mechanism instantiated with their filter satisfies the stronger guarantee of *pure* differential privacy, while performing $\Theta((\log n + 1)^d)$ lookups per query. In contrast, the filter mechanism instantiated with either of our filters uses only poly(d) lookups per query with constant-range functions, satisfies *approximate* differential privacy, and has a stronger accuracy guarantee because of the distance-respecting nature of the filters. Whereas the accuracy guarantee of [JR13] only holds when the client is honest about the function f being Lipschitz, our distance-respecting filters provide an additional accuracy guarantee for "clumsy clients" that submit a function that is close to Lipschitz—on average over possible datasets, the error of the mechanism is proportional to f's distance to the class of Lipschitz functions.

Finally, we use the filter mechanism for bounded-range functions to construct a mechanism for arbitrary-range functions and prove Theorem 5.2. Since every Lipschitz function with domain $[n]^d$ can have image diameter at most nd, we can require that the client translate the range of their function to the interval $[0, nd]$. Observe that the range restriction $f(x) \in [0, nd]$ can be easily enforced locally[8], i.e., without evaluating f at points other than x. In order to privately release $f(x)$ at some $x \in [n]^d$ in time exp(polylog(nd)), we simulate a noisy binary search for the value of $f(x)$. The simulation answers queries of the form "Is $f(x) > v$?", by clipping the range of f to the interval $[v - r, v + r]$, where $r = \Theta(\frac{1}{\varepsilon}\log nd)$, and running an instance of the filter mechanism on the clipped function to obtain a noisy answer $a(x)$. The noisy answer $a(x)$ can be interpreted as $f(x) > v$ if $a(x) > v + \frac{1}{\varepsilon}\log r$; $f(x) \approx v$ if $a(x) \in [v - \frac{1}{\varepsilon}\log r, v + \frac{1}{\varepsilon}\log r]$; and $f(x) < v$, otherwise. The resulting noisy implementation of the binary search provides accurate answers when f is Lipschitz and results in a mechanism that is always differentially private, no matter how the client behaves.

Application to tolerant testing. The second application we present is to tolerant testing of the Lipschitz property of real-valued functions on the hypercube domains. Tolerant testing, introduced in [PRR06] with the goal of understanding the properties of noisy inputs, is one of the fundamental computational tasks studied in the area of sublinear algorithms. Tolerant testing has been investigated for various properties of functions, including monotonicity, being a junta, and unateness [FF05, ACCL07, FR10, BCE$^+$19, LW19, CGG$^+$19, PRW22, BKR23].

In the standard property testing terminology, a *property* $\mathcal{P}$ is a set of functions. Given a parameter $\varepsilon \in (0,1)$, a function f is ε-far from $\mathcal{P}$ if at least an ε fraction of function values have to change to make $f \in \mathcal{P}$; otherwise, f is ε-close to $\mathcal{P}$. Given parameters $\varepsilon_0, \varepsilon \in (0,1)$ with $\varepsilon_0 < \varepsilon$ and query access to an input function f, an $(\varepsilon_0, \varepsilon)$-*tolerant tester* for $\mathcal{P}$ accepts with probability at least 2/3 if f is ε_0-close to $\mathcal{P}$ and rejects with probability at least 2/3 if f is ε-far from $\mathcal{P}$. For the special case when $\varepsilon_0 = 0$, the corresponding computational task is referred to as (standard) *testing*.

Testing of the Lipschitz property was introduced in [JR13] and subsequently studied in [CS13, DJRT13, BRY14, AJMR16, CDJS17, DRTV18, KRV23]. Lipschitz testing of functions $f : \{0,1\}^d \to \mathbb{R}$ can be performed with $O(\frac{d}{\varepsilon})$ queries [JR13, CS13]. In contrast, prior to our work, no nontrivial tolerant tester was known for this property. As shown in [FF05], tolerant testing can have drastically higher query complexity than standard testing: some properties have constant-query testers, but no sublinear-time tolerant testers. Moreover, $\exp(d^{1/4})$ queries are required for tolerantly testing the Lipschitz property with nonadaptive algorithms (i.e., algorithms that specify all queries in advance, before receiving any answers). This result follows from the monotonicity-to-Lipschitzness reduction of Chakrabarty et al. [CDJS17] and the lower bounds for tolerantly testing monotonicity [PRW22, CDL$^+$24]). Though our algorithm is adaptive, the existence of this lower bound provides evidence that the problem may be inherently hard.

As an application of our local filters,we construct the first nontrivial tolerant Lipschitz tester (see Theorem 6.1) for functions $f : \{0,1\}^d \to \mathbb{R}$.

THEOREM 1.5. (RESTATEMENT OF THEOREM 6.1 (TOLERANT TESTER)) *For all $\varepsilon \in (0, \frac{1}{3})$ and all sufficiently large $d \in \mathbb{N}$, there exists an $(\varepsilon, 2.01\varepsilon)$-tolerant tester for the Lipschitz property of functions on the hypercube $\mathcal{H}^d$. The tester has query and time complexity $\frac{1}{\varepsilon^2} d^{O(\sqrt{d \log(d/\varepsilon)})}$.*

[8] Other natural assumptions (which can be viewed as promises on the function that filter gets and which one can potentially try to use to circumvent strong lower bounds for local Lipschitz filters) cannot be enforced as easily. Some examples are monotonicity (which is not easy to enforce and has been studied in the context of local filters [SS10, BGJ$^+$12, AJMR15]) and C'-Lipschitzness (i.e., assuming the function is C'-Lipschitz and trying to enforce that it is c-Lipschitz for $c < C'$).

We stress that our tolerant tester can handle functions with any range. Given that Lipschitz functions on $\{0,1\}^d$ can have range $[0,d]$, and our ℓ_0-filter has time complexity $d^{O(r)}$ for functions $f : \{0,1\}^d \to [0,r]$, one might expect our tester to run in time $\exp(d)$ when no a priori upper bound on f is available. We leverage additional structural properties of Lipschitz functions to reduce this to $\exp(\sqrt{d})$.

1.2 Our Techniques

Algorithms. Our ℓ_1-respecting filter for functions with range $[0,r]$ is essentially a local simulation of a new distributed algorithm that iteratively "transfers mass" from large function values to small function values, with each round reducing a bound on the Lipschitz constant by a factor of 2/3. Transferring mass from element x to element y is defined as changing the values of $f(x)$ and $f(y)$ to make these values closer to each other by the same amount (which we refer to as the amount of mass transferred), in order to decrease the distance of f to Lipschitz. We use the notion of the **violation graph** for a function f, which connects the pairs of elements (x,y) for which the Lipschitzness property is violated, i.e., the difference $f(x) - f(y)$ exceeds $\text{dist}(x,y)$. Previous work [AJMR16] used the idea of transferring mass for integer-valued functions on the hypergrid and transferred one unit through edges in the violation graph along a single dimension in each iteration. Our work shows that if we take a **maximal** matching M in the violation graph and transfer mass equal to 2/3 of the maximum *violation score* $|f(x) - f(y)| - \text{dist}_G(x,y)$ along every edge in this matching, this results in a reduction of the maximum violation score by a factor of 2/3. Together with the fact that maximum violation score in the original function is at most r, this implies that $O(\log r)$ rounds suffice to turn f into a Lipschitz function. The local implementation is built on the recent algorithm of [Gha22] for giving local access to a maximal independent set.

Our local Lipschitz filters leverage powerful advances in local computation algorithms (LCAs). Both of them are built on an LCA for obtaining a maximal matching based on Ghaffari's LCA [Gha22] for maximal independent set, and they rely on the locality of the independent set algorithm to give lookup-efficient access to the corrected values. We run the maximal matching LCA on the *violation graph* of a function f with each edge labeled by the *violation score* $|f(x) - f(y)| - \text{dist}_G(x,y)$, as developed in property testing [DGL$^+$99, FLN$^+$02, JR13, AJMR16]. To get efficient local filters for functions with range $[0,r]$, we take advantage of the fact that, for such functions, the maximum degree of the violation graph is at most D_0^r (where D_0 denotes the maximum degree of G) and the fact that the matching LCA has lookup complexity that is polynomial in the degree.

Our ℓ_1-respecting filter runs in multiple stages. In each stage, it calls the maximal matching LCA on the current violation graph, which captures pairs of points with relatively large violation score. For each matched pair, the filter decreases the larger value and increases the smaller value by an amount proportional to the current bound on the violation score. This shrinking operation reduces the Lipschitz constant by a multiplicative factor, and does not increase the ℓ_1-distance to the class of Lipschitz functions. Our ℓ_0-respecting filter uses a different approach that only requires one stage. It relies on a well known technique for computing a Lipschitz extension of any real-valued function with a metric space domain. Leveraging an LCA for maximal matching allows us to simulate this extension procedure locally.

We remark that Lange, Rubinfeld and Vasilyan [LRV22] used an LCA for maximal matching to correct monotonicity of Boolean functions. Their corrector fixes violated pairs by swapping their labels; however, this technique fails to correct Lipschitzness. Additionally, unlike the corrector of [LRV22], which may change a monotone function on a constant fraction of the domain, our filters guarantee that Lipschitz functions are never modified.

Lower bounds. The first idea in the proof of our lower bound for local filters is to reduce from the problem of distribution-free property testing. Our hardness result for this problem uses novel ideas for proving lower bounds for adaptive algorithms, typically a challenging task, for which the community has developed relatively few techniques. Specifically, we show that our construction allows an adaptive algorithm to be simulated by a nonadaptive algorithm with extra information and the same query complexity. One of our main technical contributions is a query lower bound for distribution-free Lipschitz testing of functions $f : \{0,1\}^d \to [0,r]$ that is exponential in r and $\log(d/r)$ for any even r satisfying $4 \le r \le 2^{-16}d$. The lower bound we achieve demonstrates that our filters have nearly optimal query complexity.

Distribution-free testing — property testing with respect to an arbitrary distribution D on the domain using both samples from D and queries to the input — was first considered in [HK07]. Lipschitz testing has been investigated with respect to uniform distributions [CS13, DJRT13, BRY14, AJMR16, CDJS17, DRTV18, KRV23] and product distributions [DJRT13, CDJS17], but not with respect to arbitrary distributions. Our lower bound demonstrates a stark contrast in the difficulty of Lipschitz testing with respect to arbitrary distributions compared to product distributions. In particular, the Lipschitz tester of [CDJS17] for functions $f:\{0,1\}^d \to \mathbb{R}$ has query complexity linear in d for all product distributions, whereas our lower bound for distribution-free Lipschitz testing (formally stated in Theorem 4.2) implies that a query complexity of $2^{\Omega(d)}$ is unavoidable for arbitrary distributions. To prove our lower bound for distribution-free testing, we start by constructing two distributions, on positive and negative instances of this problem, respectively. The instances consist of a pair (f, U), where $f:\{0,1\}^d \to [0,r]$ is a function on the hypercube and U is a uniform distribution over an exponentially large set of points called *anchor points*. The anchor points come in pairs (x, y) such that x and y are at distance r for the positive distribution and distance $r-1$ for the negative. The function values are set to $f(x)=0$ and $f(y)=r$. For the points not in the support of U, the values are chosen to ensure that the Lipschitz condition is locally satisfied around the anchor points, and then the remaining values are set to $r/2$. We note that [HK05] also uses a construction involving pairs of anchor points to prove query and sample complexity lower bounds for distribution-free monotonicity testing; however, our approach introduces a novel "simulation" technique for proving lower bounds on the query complexity of adaptive algorithms.

The crux of the proof of Theorem 4.2 is demonstrating that every deterministic (potentially *adaptive*) tester $\mathcal{T}$ with insufficient sample and query complexity distinguishes the two distributions only with small probability. (By the standard Yao's principle this is sufficient.) An algorithm is called *nonadaptive* if it prepares all its queries before making them. A general (adaptive) algorithm, in contrast, can decide on queries based on answers to previous queries. One of the challenges in proving that the two distributions are hard to distinguish for $\mathcal{T}$ is dealing with adaptivity. We overcome this challenge by showing that $\mathcal{T}$ can be simulated by a *nonadaptive algorithm* $\mathcal{T}_{na}$ *that is provided with extra information.* Specifically, it gets one point from every pair of grouped anchor points. One of the key ideas in the analysis is that our hard distributions, and the sampling done by the tester, can be simulated by first obtaining the information provided to $\mathcal{T}_{na}$ using steps which are identical for the two hard distributions, and only then selecting the remaining anchor points to obtain the full description of the function f and the distribution U. It allows us to show that, conditioned on avoiding a small probability bad event, $\mathcal{T}$ cannot distinguish the distributions.

Applications. Our main technical contribution to the two application areas we consider is realizing that they can benefit from local filters for bounded-range functions, even when the functions in the applications have unbounded range. For the privacy application, we obtain our differentially private mechanism for general real-valued functions provided by using our local filters to simulate a noisy binary search. For the application to tolerant testing, we use McDiarmid's inequality and the observation that our ℓ_0-respecting Lipschitz filter works even with partial functions.

1.3 Preliminaries on Lipschitz Functions First, we define two important special families of graphs. We consider the hypercube $\mathcal{H}^d$ with vertices $\{0,1\}^d$ and the hypergrid $\mathcal{H}^d_n$ with vertices $[n]^d$. For both of them, two vertices are adjacent if they differ by one in one coordinate and agree everywhere else. Now, we give preliminaries on Lipschitz functions. When we discuss the range (or image) of functions, we often refer to its diameter. The *diameter* of a closed and bounded $S \subset \mathbb{R}$ is $\max_{y\in S}(y) - \min_{y\in S}(y)$. Let $G=(V,E)$ be an undirected graph and let $f: V \to \mathbb{R}$.

DEFINITION 1.2. (c-LIPSCHITZ FUNCTIONS) *Fix a constant $c>0$ and a graph $G=(V,E)$. A function $f: V\to\mathbb{R}$ is c-Lipschitz w.r.t. G if $|f(x)-f(y)| \le c\cdot \operatorname{dist}_G(x,y)$ for all $x, y \in V$. A 1-Lipschitz function is simply referred to as Lipschitz. Let $\mathcal{L}ip(G)$ be the set of Lipschitz functions w.r.t. G.*

DEFINITION 1.3. (DISTANCE TO LIPSCHITZNESS) *For all graphs $G=(V,E)$, functions $f: V\to\mathbb{R}$, distributions D over V, and $b\in\{0,1\}$, define the ℓ_b-distance to Lipschitz w.r.t. a distribution D as $\ell_{b,D}(f,\mathcal{L}ip(G)) = \min_{g\in\mathcal{L}ip(G)} \|f-g\|_{b,D}$, where*

$$\|f-g\|_{0,D} = \Pr_{x\sim D}[f(x)\neq g(x)];$$
$$\|f-g\|_{1,D} = \mathbb{E}_{x\sim D}[|f(x)-g(x)|].$$

When D is the uniform distribution, we omit it from the notation. The definition of $\|f-g\|_0$ applies when f and g are partial functions.

Next, we define the violation score of a pair of points, and the violation graph of a function.

Definition 1.4. (Violated pair, violation score) *For $x, y \in V$, let $\text{dist}_G(x,y)$ denote the shortest path distance from x to y in G. A pair (x,y) of vertices is* violated *with respect to f if $|f(x)-f(y)| > \text{dist}_G(x,y)$. The* violation score *of a pair (x,y) with respect to f, denoted $VS_f(x,y)$, is*

$$VS_f(x,y) = |f(x) - f(y)| - \text{dist}_G(x,y)$$

if (x,y) is violated and 0 otherwise. We extend these definitions to partialfunctions $g : V \to \mathbb{R} \cup \{?\}$, where ? denotes an undefined value, by stipulating that if x or y is in $g^{-1}(?)$ then (x,y) is not violated.

Definition 1.5. (Violation Graph) *The τ-violation graph with respect to f is a directed graph, denoted $B_{\tau,f}$, with vertex set V and edge set $\{(x,y) : VS_f(x,y) > \tau$ and $f(x) < f(y)\}$.*

1.4 Preliminaries on Local Computation Algorithms First, we define a *local computation algorithm (LCA)* for a graph problem.

Definition 1.6. (Graph LCA) *Fix $\delta \in (0,1)$. A graph LCA $\mathcal{A}(x,\rho)$ is a randomized algorithm that gets adjacency list access*[9] *to an input graph $G = (V,E)$, a query $x \in V$, and a random seed $\rho \in \{0,1\}^*$. For each ρ, the set of outputs $\{\mathcal{A}(x,\rho) : x \in V\}$ is consistent with some object defined with respect to G, such as a maximal matching in G. The fraction of possible random strings for which $\mathcal{A}$ fails (i.e., defines an object that does not satisfy the constraints) of the problem, is at most δ.*

We use an LCA for obtaining a maximal matching based on Ghaffari's LCA [Gha22] for maximal independent set. The description of how to obtain an LCA for a maximal matching based on Ghaffari's result [Gha22] is standard and appears, for example, in [LRV22].

Theorem 1.6. ([Gha22]) *Fix $N, D_0 \in \mathbb{N}$, and $\delta_0 \in (0,1)$. There exists a graph LCA* GhaMatch *for the maximal matching problem for graphs with N vertices and maximum degree D_0. Specifically, on input x, it outputs y if (x,y) or (y,x) is in the matching, and outputs $\perp$ if x has no match.* GhaMatch *uses a random seed of length* $\text{poly}(D_0 \cdot \log(N/\delta_0))$*, runs in time* $\text{poly}(D_0 \cdot \log(N/\delta_0))$ *per query, and has failure probability at most δ_0.*

We specify an LCA for accessing the violation graph. To simplify notation, we assume that any algorithm used as a subroutine gets access to the inputs of the algorithm which calls it; only the inputs that change in recursive calls are explicitly passed as parameters.

Algorithm 1 LCA: Viol$(f(\cdot), \tau, x)$

Input: Adjacency lists access to $G = (V,E)$, lookup access to $f : V \to [0,r]$, range diameter $r \in \mathbb{R}$, threshold $\tau \leq r$, vertex $x \in V$

Output: Neighbor list of x in $B_{\tau,f}$

1: **return** $\{y : \text{dist}_G(x,y) < |f(x) - f(y)| - \tau\}$ ▷ Compute by performing a BFS from x

Local filters were introduced by Saks and Seshadhri [SS10] and first studied for Lipschitz functions by Jha and Raskhodnikova [JR13].

Definition 1.7. (Local Lipschitz filter) *For all $c > 0$ and $\delta \in (0,1)$, a* local (c,δ)-Lipschitz filter[10] over a graph $G = (V,E)$ *is an algorithm $\mathcal{A}(x,\rho)$ that gets a query $x \in V$ and a random seed $\rho \in \{0,1\}^\star$, as well as lookup access to a function $f : V \to \mathbb{R}$ and adjacency lists access to G. With probability at least $1-\delta$ (over the random seed), the filter $\mathcal{A}$ provides query access to a c-Lipschitz function $g_\rho : V \to \mathbb{R}$ such that whenever f is c-Lipschitz $g_\rho = f$. In addition, for all $\lambda > 0$, the filter is ℓ_p-respecting with blowup λ if $\|f - g_\rho\|_p \leq \lambda \cdot \ell_p(f, \mathcal{L}ip(G))$ whenever g_ρ is c-Lipschitz.*

[9] An adjacency list lookup takes a vertex x and returns the set of vertices adjacent to x.
[10] While the graph is hardcoded in this definition, our filters work when given adjacency list access to any graph.

2 ℓ_1-respecting Local Lipschitz Filter

The d-dimensional hypergrid of side length n is the undirected graph, denoted $\mathcal{H}_n^d$, with the vertex set $[n]^d$ and the edge set $\{(x, y) : |x - y| = 1\}$.

THEOREM 2.1. *For all $\gamma > 0$ and $\delta \in (0, 1)$, there is an ℓ_1-respecting local $(1 + \gamma, \delta)$-Lipschitz filter with blowup 2 over the d-dimensional hypergrid $\mathcal{H}_n^d$. Given lookup access to a function $f : [n]^d \to [0, r]$, and a random seed ρ of length $d^{O(r)} \cdot \text{polylog}(n \log(r/\gamma)/\delta))$, the filter has lookup and time complexity $(d^r \cdot \text{polylog}(n/\delta))^{O(\log(r/\gamma))}$ for each query $x \in [n]^d$. If f is Lipschitz, then the filter outputs $f(x)$ for all queries x and random seeds.*

We first give a global Lipschitz filter (Algorithm 2) and then show how to simulate it locally (in Algorithm 3) by using the result of [Gha22] stated in Theorem 1.6.

Algorithm 2 GLOBALFILTER$_1$

Input: Graph $G = (V, E)$, function $f : V \to [0, r]$, range diameter $r \in \mathbb{R}$, and approximation parameter $\gamma > 0$
Output: $(1 + \gamma)$-Lipschitz function $g : V \to [0, r]$

```
1: Let g_1 ← f
2: for t ← 2 to log_{3/2}(r/γ) + 1 do        ▷ Start at t = 2 for GLOBALFILTER_1-LOCALFILTER_1 analogy.
3:     Set threshold τ ← r · (2/3)^{t−1} and move-amount Δ ← (r/3) · (2/3)^{t−2}
4:     Construct B_{τ,g_{t−1}} (Definition 1.5) and compute a maximal matching M_t of B_{τ,g_{t−1}}
5:     Set g_t ← g_{t−1}
6:     for (x, y) ∈ M_t do                                        ▷ Recall: f(x) < f(y)
7:         Set g_t(x) ← g_t(x) + Δ
8:         Set g_t(y) ← g_t(y) − Δ
9: return g_t, where t = log_{3/2}(r/γ) + 1
```

2.1 Analysis of the Global Filter The guarantees of GLOBALFILTER (Algorithm 2) are summarized in the following lemma.

LEMMA 2.1. *For all input graphs $G = (V, E)$, functions $f : V \to [0, r]$, and $\gamma > 0$, if g is the output of* GLOBALFILTER$_1$*, then g is a $(1 + \gamma)$-Lipschitz function and $\|g - f\|_1 \leq 2\ell_1(f, \mathcal{L}ip(G))$.*

We prove Lemma 2.1 via a sequence of claims. Claim 2.1 makes an important observation about the violation scores on adjacent edges in the violation graph. Claim 2.2 argues that the violation scores decrease after each iteration of the loop. Claim 2.3 converts the guarantee for each iteration to the guarantee on the Lipschitz constant for the output function. Finally, Claim 2.4 bounds the ℓ_1-distance between the input and output functions.

CLAIM 2.1. *If (x, y) and (y, z) are edges in the violation graph $B_{0,f}$ then $VS_f(x, z) \geq VS_f(x, y) + VS_f(y, z)$.*

Proof. Since (x, y) and (y, z) are edges in the violation graph $B_{0,f}$, then $f(x) < f(y) < f(z)$. Therefore,

$$\begin{aligned} VS_f(x, z) &= f(z) - f(x) - \text{dist}_G(x, z) \\ &\geq f(z) - f(y) + f(y) - f(x) - \text{dist}_G(x, y) - \text{dist}_G(y, z) \\ &= VS_f(x, y) + VS_f(y, z), \end{aligned}$$

by the definition of the violation score and the triangle inequality. ∎

Next, we abstract out and analyze the change to the function values made in each iteration of the loop in Algorithm 2.

CLAIM 2.2. (MAXIMUM VIOLATION SCORE REDUCTION) *Let $G = (V, E)$ be a graph and let $g : V \to [0, r]$ be a function such that $VS_g(x, y) \leq v$ for all $x, y \in V$. Let M be a maximal matching in $B_{2v/3,g}$ such that $g(x) < g(y)$ for all $(x, y) \in M$. Obtain h as follows: set $h = g$ and then, for every edge $(x, y) \in M$, set $h(x) \leftarrow h(x) + v/3$ and $h(y) \leftarrow h(y) - v/3$. Then $VS_h(x, y) \leq 2v/3$ for all $x, y \in V$.*

Proof. Suppose $x, y \in V$, assume w.l.o.g. that $g(x) \leq g(y)$, and consider the following two cases. Recall that edges in the violation graph (x, y) are directed from the smaller value x to the larger value y.

Case 1: (x, y) is an edge in $B_{2v/3,g}$. By Claim 2.1, we cannot have vertices a, b, c such that (a, b) and (b, c) are both in $B_{2v/3,g}$, since otherwise $VS_g(a, c)$ would be at least $VS_g(a, b) + VS_g(b, c) > 2v/3 + 2v/3 > v$, contradicting the upper bound of v on violation scores stated in Claim 2.2. Thus, in $B_{2v/3,g}$, each edge incident on x is outgoing and each edge incident on y is incoming. Consequently, $h(x) \geq g(x)$ and $h(y) \leq g(y)$.

Moreover, since (x, y) is in $B_{2v/3,g}$ and M is a maximal matching, at least one of x, y is matched in M. W.l.o.g. assume that M contains an edge (x, z). Then $h(x) = g(x) + v/3$. Since $h(y) \leq g(y)$, we have $VS_h(x, y) \leq VS_g(x, y) - v/3 \leq 2v/3$.

Case 2: (x, y) is not an edge in $B_{2v/3,g}$. Then $VS_g(x, y) \leq 2v/3$. Consider how the values of x and y change when we go from g to h. Observe that $|h(x) - g(x)|$ is 0 or $v/3$ for all $x \in V$. If both values for x and for y stay the same, or move in the same direction (both increase or both decrease), then $VS_h(x, y) = VS_g(x, y) \leq 2v/3$. If they move towards each other, then $VS_h(x, y) \leq 2v/3$, whether $h(x) \leq h(y)$ or not.

Now consider the case when they move away from each other, that is, $h(x) \leq g(x)$ and $h(y) \geq g(y)$, and at least one of the inequalities is strict. First, suppose both inequalities are strict. Then there are vertices z_x, z_y such that $(z_x, x), (y, z_y) \in M$. By Claim 2.1, pair (x, y) is not violated in g (since otherwise $VS_g(z_x, z_y)$ would be at least $VS_g(z_x, x) + VS_g(x, y) + VS_g(y, z_y) > 4v/3$, contradicting the assumption on violation scores in the claim). Since the values of the endpoints move by $v/3$ each, the new violation score $VS_h(x, y) \leq 2v/3$.

Finally, consider the case when only one of the inequalities is strict. W.l.o.g. suppose $h(y) > g(y)$. Then there is $z \in V$ such that $(y, z) \in M$. By Claim 2.1, the violation score $VS_g(x, y) \leq v/3$, since otherwise $VS_g(x, z)$ would be at least $VS_g(x, y) + VS_g(y, z) > v/3 + 2v/3 = v$, contradicting the assumption on violation scores in the claim. Thus, $VS_h(x, y) \leq VS_g(x, y) + v/3 \leq 2v/3$, as required. ∎

Next, we use Claim 2.2 to bound the Lipschitz constant of the function output by the global filter.

CLAIM 2.3. *For all $t \geq 1$, the function g_t computed in Algorithm 2 is $(1 + r(\frac{2}{3})^{t-1})$-Lipschitz. In particular, if $t^* = \log_{3/2}(r/\gamma) + 1$ then g_{t^*} is $(1 + \gamma)$-Lipschitz.*

Proof. Fix a graph $G = (V, E)$ and a function $f : V \to [0, r]$. Then $VS_f(x, y) < r$ for all $x, y \in V$.

For all $t \geq 1$, let $v_t = r(\frac{2}{3})^{t-1}$. Notice that, in Line 3 of Algorithm 2, we set $\tau = \frac{2}{3}v_{t-1}$ and $\Delta = \frac{r}{3}(\frac{2}{3})^{t-2} = \frac{1}{3}v_{t-1}$. To prove the claim, it suffices to show that for all $t \geq 1$ and $x, y \in V$,

$$VS_{g_t}(x, y) \leq v_t, \tag{2.1}$$

since then $|g_t(x) - g_t(y)| \leq \text{dist}_G(x, y)(1 + v_t)$. We prove Equation (2.1) by induction on t.

In the base case of $t = 1$, we have $VS_{g_1}(x, y) < r = v_1$ for all $x, y \in V$. Assume Equation (2.1) holds for some $t \geq 1$. Then, instantiating Claim 2.2 with $g = g_t, h = g_{t+1}$, and $v = v_t$ yields $VS_{g_{t+1}}(x, y) \leq 2v_t/3 = v_{t+1}$ for all $x, y \in V$.

In conclusion, since $v_{t^*} = \gamma$, the function g_{t^*} is $(1 + \gamma)$-Lipschitz. ∎

Finally, we argue that the ℓ_1-distance between the input and the output functions of Algorithm 2 is small.

CLAIM 2.4. *Fix a graph $G = (V, E)$ and a function $f : V \to [0, r]$. Suppose h is the closest (in ℓ_1-distance) Lipschitz function to f. Then for all $t \geq 1$, the functions g_t computed by Algorithm 2 satisfy $\|g_{t+1} - h\|_1 \leq \|g_t - h\|_1$ and $\|g_t - f\|_1 \leq 2\ell_1(f, \mathcal{L}ip(G))$.*

Proof. Fix $t \geq 1$. Since g_t and g_{t+1} only differ on the endpoints of the edges in the matching M_{t+1}, we restrict our attention to those points. For each edge $(x, y) \in M_{t+1}$, we will show

$$|g_{t+1}(x) - h(x)| + |g_{t+1}(y) - h(y)| \leq |g_t(x) - h(x)| + |g_t(y) - h(y)|. \tag{2.2}$$

Let $\tau = r(\frac{2}{3})^t$ and $\Delta = \frac{r}{3}(\frac{2}{3})^{t-1} = \frac{\tau}{2}$. Suppose $(x, y) \in M_{t+1}$. Recall that this implies that $VS_{g_t}(x, y) > \tau = 2\Delta$ and $g_t(x) < g_t(y)$. By construction, $g_{t+1}(x) = g_t(x) + \Delta$ and $g_{t+1}(y) = g_t(y) - \Delta$. Thus, the violation score of (x, y) decreased by 2Δ, so (x, y) is still violated by g_{t+1}, i.e.,

$$g_{t+1}(x) + \text{dist}_G(x, y) < g_{t+1}(y). \tag{2.3}$$

Define $\Phi(z) = |g_{t+1}(z) - h(z)| - |g_t(z) - h(z)|$ for all $z \in V$. (Intuitively, it captures how much further from $h(z)$ the value on z moved when we changed g_t to g_{t+1}.) Then Equation (2.2) is equivalent to $\Phi(x) + \Phi(y) \le 0$. If both $\Phi(x) \le 0$ and $\Phi(y) \le 0$, then Equation (2.2) holds. Otherwise, $\Phi(x) > 0$ or $\Phi(y) > 0$. Suppose w.l.o.g. $\Phi(x) > 0$. Since $g_{t+1}(x) = g_t(x) + \Delta$, we know that $\Phi(x) \le \Delta$. To demonstrate that Equation (2.2) holds, it remains to show that $\Phi(y) \le -\Delta$.

Since $\Phi(x) > 0$, the value $h(x)$ is closer to $g_t(x)$ than to $g_{t+1}(x)$. Since $g_{t+1}(x) = g_t(x) + \Delta$, it implies that $h(x)$ must be below the midpoint between $g_t(x)$ and $g_{t+1}(x)$, which is $g_{t+1}(x) = \Delta/2$. That is,

$$h(x) < g_{t+1}(x) - \Delta/2. \tag{2.4}$$

We use that h is Lipschitz, then apply Equations (2.4) and (2.3) to obtain

$$h(y) < h(x) + \mathrm{dist}_G(x, y) < g_{t+1}(x) - \Delta/2 + \mathrm{dist}_G(x, y) < g_{t+1}(y) - \Delta/2.$$

Since $h(y) < g_{t+1}(y) - \Delta/2$ and $g_{t+1}(y) = g_t(y) - \Delta$, we get that $g_t(y)$ and $g_{t+1}(y)$ are both greater than $h(y)$. Thus, $|g_{t+1}(y) - h(y)| = |g_t(y) - h(y)| - \Delta$ and hence, $\Phi(y) = -\Delta$, so Equation (2.2) holds.

We proved that Equation (2.2) holds for every edge in M_{t+1}. Moreover, for all vertices z outside of M_{t+1}, we have $g_{t+1}(z) = g_t(z)$ and, consequently, $\Phi(z) = 0$. Summing over all vertices, we get that $\sum_{x \in V} \Phi(x) \le 0$. Thus, $\|g_{t+1} - h\|_1 \le \|g_t - h\|_1$. By the triangle inequality, $\|g_t - f\|_1 \le \|g_t - h\|_1 + \|h - f\|_1 \le \|g_1 - h\|_1 + \|f - h\|_1 = 2\ell_1(f, \mathcal{L}ip(G))$. ◻

Lemma 2.1 follows from Claims 2.3 and 2.4.

2.2 Analysis of the Local Filter In this section, we present a local implementation of Algorithm 2 and complete the proof of Theorem 2.1. We claim that for each $t \in [\log(r/\gamma) + 1]$, Algorithm 3 simulates round t of Algorithm 2 and, for graphs on N vertices with maximum degree D, has lookup complexity $(D^r \cdot \mathrm{polylog}(N/\delta))^{O(\log(r/\gamma))}$.

Algorithm 3 LCA: LocalFilter$_1(x, t, \rho_1 \circ ... \circ \rho_t)$

Input: Adjacency lists access to graph $G = (V, E)$, lookup access to $f : V \to [0, r]$, range diameter $r \in \mathbb{R}$, vertex $x \in V$, iteration number t, approximation parameter $\gamma > 0$, and random seed $\boldsymbol{\rho} = \rho_1 \circ ... \circ \rho_t$
Subroutines: GhaMatch (see Theorem 1.6) and Viol (see Algorithm 1)
Output: Query access to $(r \cdot (\frac{2}{3})^{t-1})$-Lipschitz function $g_{\boldsymbol{\rho}} : V \to [0, r]$

```
if t = 1 or r · (2/3)^{t−1} < γ then
    return f(x)
Set threshold τ ← r · (2/3)^{t−1} and move amount Δ ← (r/3) · (2/3)^{t−2}
Set f_t(x) ← LocalFilter_1(x, t − 1, ρ_1 ∘ ... ∘ ρ_{t−1})
Set y ← GhaMatch(Viol(LocalFilter_1(·, t − 1, ρ_1 ∘ ... ∘ ρ_{t−1}), τ, ·), x, ρ_t)
if y ≠ ⊥ then
    f_{t−1}(y) ← LocalFilter_1(y, t − 1, ρ_1 ∘ ... ∘ ρ_{t−1})
    f_t(x) ← f_t(x) + sign(f_{t−1}(y) − f_t(x)) · Δ
return f_t(x)
```

Definition 2.1. (Good seed) *Let $G = (V, E)$ be a graph and fix $t \ge 1$. Consider a function $f : V \to [0, r]$. A string $\boldsymbol{\rho} = \rho_1 \circ ... \circ \rho_t$ is* a good seed *for G and f if, for all $i \in [t]$, the matching computed by* GhaMatch *in* LocalFilter$_1(\cdot, i, \rho_1 \circ ... \circ \rho_i)$ *is maximal.*

Claim 2.5. *Fix a graph $G = (V, E)$, a function $f : V \to [0, r]$, and $\gamma > 0$. Let $t^* = \log_{3/2}(r/\gamma) + 1$ and fix a good seed $\boldsymbol{\rho} = \rho_1 \circ \cdots \circ \rho_{t^*}$. Let $g(x)$ denote* LocalFilter$_1(x, t^*, \boldsymbol{\rho})$ *for all $x \in V$. Then g is a $(1 + \gamma)$-Lipschitz function with range $[0, r]$ and $\|f - g\|_1 \le 2\ell_1(f, \mathcal{L}ip(G))$.*

Proof. For all $t \in [t^*]$, let g_t be the function computed by GlobalFilter$_1$ on G, r, f, and γ after iteration t using the matching computed by the call to GhaMatch in LocalFilter$_1(x, t, \rho_1 \circ ... \circ \rho_t)$. Recall that the matching computed by each call to GhaMatch in LocalFilter$_1(x, t, \rho_1 \circ ... \circ \rho_t)$ is maximal and therefore can be used as the matching in the iteration t of the loop in GlobalFilter$_1$.

By an inductive argument, LocalFilter$_1(x, t, \rho_1 \circ \ldots \circ \rho_t) = g_t(x)$ for all $x \in V$ and $t \in [t^*]$. The base case is LocalFilter$_1(x, 1, \rho_1) = f(x) = g_1$, and every subsequent g_t computed by GlobalFilter$_1$ is the same as LocalFilter$_1(\cdot, t, \rho_1 \circ \ldots \circ \rho_t)$. Hence, LocalFilter$_1(x, t^*, \boldsymbol{\rho})$ provides query access to g_{t^*}. By Lemma 2.1, g_{t^*} is $(1+\gamma)$-Lipschitz and satisfies $\|g_{t^*} - f\| \leq 2\ell_1(f, \mathcal{L}ip(G))$. ∎

Lemma 2.2. *Fix $\gamma > 0$ and $\delta \in (0, 1)$. Let $G = (V, E)$ be a graph with $|V| = N$ and maximum degree D. Let $f : V \to [0, r]$ and $t^* = \log_{3/2}(r/\gamma) + 1$. Then, for a random seed $\boldsymbol{\rho} = \rho_1 \circ ... \circ \rho_{t^*}$, which is a concatenation of t^* strings of length $D^{O(r)} \operatorname{polylog}(Nt^*/\delta)$ each, the algorithm* LocalFilter$_1(\cdot, t^*, \boldsymbol{\rho})$ *is an ℓ_1-respecting local $(1 + \gamma, \delta)$-Lipschitz filter with blowup 2 and lookup and time complexity $(D^r \cdot \operatorname{polylog}(N/\delta))^{O(\log(r/\gamma))}$.*

Proof. Since the range of f is at most r, two vertices in a violated pair can be at distance at most $r - 1$. Hence, the maximum degree of the violation graph $B_{\tau,f}$ is at most D^r. By Theorem 1.6 instantiated with $D_0 = D^r$ and $\delta_0 = \frac{t^*}{\delta}$, the failure probability of each call to GhaMatch is at most $\frac{\delta}{t^*}$. Since there are at most t^* calls to GhaMatch, the probability that any call fails is at most δ. It follows that a random string $\boldsymbol{\rho}$ of length specified in the lemma is a good seed (see Definition 2.1) with probability at least $1 - \delta$. This allows us to apply Claim 2.5, and conclude that LocalFilter$_1(x, t^*, \boldsymbol{\rho})$ provides query access to a $(1 + \gamma)$-Lipschitz function and fails with probability at most δ over the choice of $\boldsymbol{\rho}$.

Let $Q(t)$ be the lookup complexity of LocalFilter$_1(x, t, \rho_1 \circ ... \circ \rho_t)$. Then $Q(1) = 1$ and, since the max degree of $B_{\tau,f}$ is D^r, each lookup made by GhaMatch to the violation graph oracle in the $(t-1)$-st iteration requires at most $D^r Q(t-1)$ lookups to compute. Since GhaMatch makes $D^{O(r)} \operatorname{polylog}(N/\delta)$ such lookups, $Q(t) \leq D^{O(r)} \operatorname{polylog}(n/\delta) Q(t-1)$. Thus, the final lookup complexity is $Q(t^*) \leq (D^r \cdot \operatorname{polylog}(N/\delta))^{O(\log(r/\gamma))}$. By inspection of the pseudocode, we see that the running time is polynomial in the number of lookups. ∎

Proof of Theorem 2.1. The theorem follows as a special case of Lemma 2.2 with G equal to the hypergrid $\mathcal{H}_n^d$. The hypergrid has n^d vertices and maximum degree $2d$. This gives lookup and time complexity $(d^r \cdot \operatorname{polylog}(n/\delta))^{O(\log(r/\gamma))}$. If f is Lipschitz, then all violation graphs are empty; therefore, any local matching algorithm returns an empty matching (or can otherwise be amended to do so by checking whether the returned edge is in the graph and returning $\perp$ if it is not). Thus, when f is Lipschitz, the returned value is always $f(x)$. ∎

3 ℓ_0-respecting Local Lipschitz Filter

In this section, we present a local Lipschitz filter that respects ℓ_0-distance rather than ℓ_1-distance. Unlike the ℓ_1-respecting filter, the ℓ_0-respecting filter outputs a function that is 1-Lipschitz.

Theorem 3.1. *For all $\delta \in (0, 1)$, there exists an ℓ_0-respecting local $(1, \delta)$-Lipschitz filter with blowup 2 over the d-dimensional hypergrid $\mathcal{H}_n^d$. Given lookup access to a function $f : [n]^d \to [0, r]$, and a random seed ρ of length $d^{O(r)} \cdot \operatorname{polylog}(n/\delta))$, the filter has lookup and time complexity $d^{O(r)} \cdot \operatorname{polylog}(n/\delta)$ for each query $x \in [n]^d$. If f is Lipschitz, then the filter outputs $f(x)$ for all queries x and random seeds. If for all $y \in [n]^d$ we have $|f(x) - f(y)| \leq |x - y|$ then the filter outputs $f(x)$.*

We give a global view (Algorithm 4) and prove its correctness before presenting a local implementation (Algorithm 5). We use the convention that $\max_{y \in S}(\cdot)$ is defined to be zero when S is the empty set.

Algorithm 4 GlobalFilter$_0$

Input: Graph $G = (V, E)$, function $f : V \to [0, r]$
Output: Lipschitz function $g : V \to [0, r]$

1: Construct $B_{0,f}$ (see Definition 1.5) and compute a vertex cover C of $B_{0,f}$
2: Set $g_C \leftarrow f$
3: **for** every vertex $u \in C$ **do**
4: Set $g_C(u) \leftarrow \max(0, \max_{v \in V \setminus C}(g_C(v) - \operatorname{dist}_G(u, v)))$
5: **return** g_C

3.1 Analysis of the Global Filter Algorithm 4 reassigns the labels on a vertex cover C of the violation graph $B_{0,f}$. Observe that the partial function f on the domain $V \setminus C$ is Lipschitz w.r.t. G, because its violation graph has no edges. We claim that this algorithm extends this partial function to a Lipschitz function defined on all of G. It is well known that for a function $f : X \to \mathbb{R}$ with a metric space domain, if f is Lipschitz on some subset $Y \subset X$, then f can be made Lipschitz while only modifying points in $X \setminus Y$. See, for example, [JR13] and [BL00]. We include a proof for completeness.

Claim 3.1. (Lipschitz extension) *Let $G = (V, E)$ be a graph, and $f : V \to [0, r]$ a function. Then, for all vertex covers C of $B_{0,f}$, the function g_C returned by Algorithm 4 is Lipschitz.*

Proof. Let $f : V \to [0, r] \cup \{?\}$ be a partial Lipschitz function and let A_f be the set of points on which f is defined. Fix a vertex $x \notin A_f$ and obtain the function g as follows: Set $g(y) = f(y)$ for all $y \in A_f$. Set $g(x) = \max_{v \in A_f}(f(v) - \mathrm{dist}_G(x, v)))$. Note that in the case where A_f is empty, the function f is nowhere defined, and hence setting $g(x) = 0$ will always result in a Lipschitz function. Thus, assume w.l.o.g. that A_f is not empty.

We will first argue that g is Lipschitz. Let $v^* = \arg\max_{v \in A_f}(f(v) - \mathrm{dist}_G(x, v))$, i.e., a vertex such that $g(x) = f(v^*) - \mathrm{dist}_G(x, v^*)$. Then, for all $v \in A_f$,

$$g(x) - g(v) = f(v^*) - \mathrm{dist}_G(x, v^*) - f(v) \le \mathrm{dist}_G(v, v^*) - \mathrm{dist}_G(x, v^*) \le \mathrm{dist}_G(x, v).$$

Similarly, $g(v) - g(x) \le f(v) + \mathrm{dist}_G(x, v) - f(v) = \mathrm{dist}_G(x, v)$, so g is Lipschitz. Notice that if g is a Lipschitz function, then $\max(0, g)$ is also a Lipschitz function (truncating negative values can only decrease the distance between $g(x)$ and $g(y)$ for all pairs x, y in the domain). Thus, setting $g(x) = \max(0, \max_{v \in A_f}(f(v) - \mathrm{dist}_G(x, v)))$ will also yield a Lipschitz function.

Next, we argue that the order of assignment does not affect the extension. Let A_g be set of points on which g is defined and note that $A_g = A_f \cup \{x\}$. We will show that for all $z \notin A_g$ we have

$$\max(0, \max_{v \in A_f}(f(v) - \mathrm{dist}_G(z, v))) = \max(0, \max_{v \in A_g}(g(v) - \mathrm{dist}_G(z, v))).$$

Let $f(z) = \max(0, \max_{v \in A_f}(f(v) - \mathrm{dist}_G(z, v)))$ and $g(z) = \max(0, \max_{v \in A_g}(g(v) - \mathrm{dist}_G(z, v)))$. Then, since $A_g = A_f \cup \{x\}$, we obtain $g(z) = \max(f(z), g(x) - \mathrm{dist}_G(x, z))$. If $g(x) = 0$ then $f(z) = g(z)$ since by definition $f(z) \ge 0$. On the other hand, if $g(x) > 0$ then $g(x) = f(v^*) - \mathrm{dist}_G(x, v^*)$ and hence,

$$\begin{aligned} f(z) &\ge f(v^*) - \mathrm{dist}_G(v^*, z) \ge (f(v^*) - \mathrm{dist}_G(x, v^*)) - \mathrm{dist}_G(x, z) \\ &= g(x) - \mathrm{dist}_G(x, z), \end{aligned}$$

which implies $f(z) = g(z)$. To complete the proof of Claim 3.1, Let $f : V \to [0, r]$ be a function with violation graph $B_{0,f}$. Notice that if C is a vertex cover of $B_{0,f}$ then $f : V \setminus C \to [0, r]$ is Lipschitz, and thus, setting $f(x) =?$ for all $x \in C$ and applying the extension procedure inductively, we see that g_C is a Lipschitz function. ∎

The following claim relates the distance to Lipschitzness to the vertex cover of the underlying graph. This relationship is standard for the Lipschitz and related properties, such as monotonicity over general partially ordered sets [FLN$^+$02, Corrolary 2]. See [CS13, Theorem 5] for the statement and proof for the special case of hypergrid domains. The arguments in these papers extend immediately to the setting of general domains.

Claim 3.2. (Distance to Lipschitz) *For all graphs $G = (V, E)$ on n vertices and functions $f : V \to \mathbb{R}$, the size of the minimum vertex cover of the violation graph $B_{0,f}$ is exactly $n \cdot \ell_0(f, \mathcal{L}ip(G))$.*

Proof of Claim 3.2. Let C be any minimum vertex cover of $B_{0,f}$. We first argue that $n \cdot \ell_0(f, \mathcal{L}ip(G)) \le |C|$. If g is a partial function that is equal to ? on vertices in C and equals f elsewhere, then g is Lipschitz outside of C. Then, by Claim 3.1 with C as the cover in Algorithm 4, there exist values $y_1, \dots, y_{|C|}$ such that setting $g(x_i) = y_i$ for each $x_i \in C$ yields a Lipschitz function. It follows that $n \cdot \ell_0(f, \mathcal{L}ip(G)) \le |C|$.

Next, using the function g defined in the previous paragraph, suppose the set $P = \{x : g(x) \ne f(x)\}$ is not a vertex cover for $B_{0,f}$. Then, there exists some edge (x, y) in $B_{0,f}$ such that $f(x) = g(x)$ and $f(y) = g(y)$ (i.e. $x, y \notin P$). But by definition of $B_{0,f}$, this implies that $|g(x) - g(y)| > \mathrm{dist}_G(x, y)$ which contradicts the fact that g is Lipschitz. It follows that $n \cdot \ell_0(f, \mathcal{L}ip(G)) = |C|$. ∎

3.2 Analysis of the Local Filter Using Algorithm 1 and GhaMatch from Theorem 1.6, we construct Algorithm 5, an LCA which provides query access to a Lipschitz function close to the input function. It is analyzed in Lemma 3.1.

Algorithm 5 LCA: LocalFilter$_0(x, \rho)$

Input: Adjacency lists access to graph $G = (V, E)$, lookup access to $f : V \to [0, r]$, range diameter $r \in \mathbb{R}$, vertex $x \in V$, random seed ρ
Subroutines: GhaMatch (see Theorem 1.6) and Viol (see Algorithm 1).
Output: Query access to Lipschitz function $g : V \to [0, r]$.

```
1: if GhaMatch(Viol(f, 0, ·), x, ρ) = ⊥ then
2:     return f(x)
3: else
4:     S ← {y : dist_G(x, y) ≤ r and GhaMatch(Viol(f, 0, ·), y, ρ) = ⊥}
5:     return max(0, max_{y∈S}(f(y) − dist_G(x, y)))
```

Lemma 3.1. (LocalFilter$_0$) *Fix $\delta \in (0, 1)$. Let $G = (V, E)$ be a graph with N vertices and maximum degree D. Then, for a random seed ρ of length $D^{O(r)} \cdot \mathrm{polylog}(N/\delta)$, the algorithm* LocalFilter$_0(x, \rho)$ *(Algorithm 5) is an ℓ_0-respecting local $(1, \delta)$-Lipschitz filter with blowup 2 and lookup and time complexity $D^{O(r)} \cdot \mathrm{polylog}(N/\delta)$.*

Proof. Since the range of f is bounded by r, a pair of violated vertices x, y must have $\mathrm{dist}_G(x, y) < r$, and thus, the maximum degree of $B_{0,f}$ is at most D^r. By Theorem 1.6 instantiated with $D_0 = D^r$ and $\delta_0 = \delta$, the algorithm GhaMatch has lookup and time complexity $D^{O(r)} \cdot \mathrm{polylog}(N/\delta)$ per query, and fails to provide query access to a maximal matching with probability at most δ over the choice of ρ. Since LocalFilter$_0$ makes at most D^r queries to GhaMatch and only fails when GhaMatch fails, LocalFilter$_0$ has lookup and time complexity $D^{O(r)} \cdot \mathrm{polylog}(N/\delta)$ and failure probability at most δ. Let ρ be a seed for which GhaMatch does not fail, and let C be the set of vertices that are matched by GhaMatch when given adjacency lists access to $B_{0,f}$. Since the matching is maximal, C is a 2-approximate vertex cover of $B_{0,f}$. Hence, we can run GlobalFilter$_0$ (Algorithm 4) and use C as the vertex cover. Since LocalFilter$_0$ and GlobalFilter$_0$ apply the same procedure to every vertex in V, and since C has at most twice as many vertices as a minimum vertex cover, Claims 3.1 and 3.2 imply that LocalFilter$_0$ provides query access to some Lipschitz function g satisfying $\|f - g\|_0 = |C| \le 2\ell_0(f, \mathcal{L}ip(G))$. ∎

Proof of Theorem 3.1. This is an application of Lemma 3.1 to the d-dimensional hypergrid $\mathcal{H}_n^d$. The hypergrid has n^d vertices and a maximum degree of $2d$, therefore, the lookup and time complexity are $d^{O(r)} \cdot \mathrm{polylog}(n^d/\delta) = d^{O(r)} \cdot \mathrm{polylog}(n/\delta)$. Similarly, the length of the random seed is also $d^{O(r)} \cdot \mathrm{polylog}(n/\delta)$. If f is Lipschitz, then all violation graphs are empty; therefore, any local matching algorithm returns an empty matching (or can otherwise be amended to do so by checking whether the returned edge is in the graph and returning $\perp$ if it is not). Thus, when f is Lipschitz, the returned value is always $f(x)$.

If for all $y \in [n]^d$ we have $|f(x) - f(y)| \le |x - y|$ then no edges in the violation graph are incident on$x$. Therefore, every local matching algorithm returns a matching that does not contain x, or can otherwise be amended to do so by checking whether the returned edge is in the graph and returning $\perp$ if it is not. Hence, the returned value is always $f(x)$. ∎

4 Lower Bounds

In this section, we prove our lower bound for local filters, stated in Section 1. We start with a more detailed statement of the lower bound.

Theorem 4.1. (Local Lispchitz filter lower bound) *For all local $(1, \frac{1}{4})$-Lipschitz filters $\mathcal{A}$ over the hypercube $\mathcal{H}^d$, for all even $r \ge 4$ and integer $d \ge 2^{16}r$, there exists a function $f : \{0, 1\}^d \to [0, r]$ for which the lookup complexity of $\mathcal{A}$ is $(\frac{d}{r})^{\Omega(r)}$.*

Note that the same bound on lookup complexity applies to local $(1 + \frac{1}{2r}, \frac{1}{4})$-Lipschitz filters over $\mathcal{H}^d$. This can be seen by observing the Lipschitz constant in the hard distributions constructed in the proof of the theorem (in Definition 4.2). Our proof is via a reduction from distribution-free testing. In Section 4.1, we state our lower bound on distribution-free testing of Lipschitz functions and use it to derive the lower bound on local Lipschitz filters stated in Theorem 4.1. In Section 4.2, we prove our lower bound for distribution-free testing.

4.1 Testing Definitions and the Lower Bound for Local Lipschitz Filters We start by defining distribution-free testing of the Lipschitz property.

Definition 4.1. (Distribution-free Lipschitz testing) *Fix* $\varepsilon \in (0, 1/2]$ *and* $r \in \mathbb{R}$. *A* distribution-free Lipschitz ε-tester $\mathcal{T}$ *is an algorithm that gets query access to the input function* $f : \{0,1\}^d \to [0, r]$ *and sample access to the input distribution* D *over* $\{0,1\}^d$. *If* f *is Lipschitz, then* $\mathcal{T}(f, D)$ *accepts with probability at least* $2/3$*, and if* $\ell_{0,D}(f, \mathcal{L}ip(\mathcal{H}^d)) \geq \varepsilon$*, then it rejects with probability at least* $2/3$.

We give a sample and query lower bound for this task.

Theorem 4.2. (Distribution-free testing lower bound) *Let* $\mathcal{T}$ *be a distribution-free Lipschitz* $\frac{1}{2}$*-tester. Then, for all sufficiently large* $d \in \mathbb{N}$ *and even integers* $4 \leq r \leq 2^{-16}d$*, there exists a function* $f : \{0,1\}^d \to [0, r]$ *and a distribution* D*, such that* $\mathcal{T}(f, D)$ *either has sample complexity* $2^{\Omega(d)}$*, or query complexity* $(\frac{d}{r})^{\Omega(r)}$.

Before proving Theorem 4.2, we use it to prove the lower bound on the lookup complexity of local Lipschitz filters, stated in Theorem 4.1. Recall that it says that every local $(1, \frac{1}{4})$-Lipschitz filter over the hypercube $\mathcal{H}^d$ w.r.t. ℓ_0-distance has worst-case lookup complexity $(\frac{d}{r})^{\Omega(r)}$.

Proof of Theorem 4.1. Let $\mathcal{A}$ be a local $(1, \frac{1}{4})$-Lipschitz filter for functions $f : \{0,1\}^d \to [0, r]$ over the hypercube $\mathcal{H}^d$. Then, given an instance (f, D) of the distribution-free Lipschitz testing problem with proximity parameter ε, we can run the following algorithm, denoted $\mathcal{T}(f, D)$:

1. Sample a set S of $3/\varepsilon$ points from D.
2. If $\mathcal{A}(x, \rho) \neq f(x)$ for some $x \in S$ then **reject**; otherwise, **accept**.

If f is Lipschitz then, with probability at least $3/4 > 2/3$, we have $\mathcal{A}(x, \rho) = f(x)$ for all $x \in S$ and, consequently, $\mathcal{T}(f, D)$ accepts. Now suppose f is ε-far from Lipschitz with respect to D. Then $\mathcal{A}$ fails with probability at most $1/4$. With the remaining probability, it provides query access to some Lipschitz function g_ρ. This function disagrees with f on a point sampled from D with probability at least ε. In this case, $\mathcal{T}$ incorrectly accepts with probability at most $(1 - \varepsilon)^{3/\varepsilon} \leq e^{-3}$. By a union bound, $\mathcal{T}$ fails or accepts with probability at most $\frac{1}{4} + e^{-3} \leq \frac{1}{3}$. Therefore, $\mathcal{T}$ satisfies Definition 4.1. By Theorem 4.2, $\mathcal{T}$ needs at least $(\frac{d}{r})^{\Omega(r)}$ queries, so $\mathcal{A}$ must make at least $(\frac{d}{r})^{\Omega(r)}$ lookups. ∎

4.2 Distribution-Free Testing Lower Bound We prove Theorem 4.2 by constructing two distributions, D_0 and D_1, on pairs (f, D) and then applying Yao's Minimax Principle [Yao77]. We show that D_0 has most of its probability mass on positive instances and D_1 has most of its mass on negative instances of distribution-free Lipschitz testing. The crux of the proof of Theorem 4.2 is demonstrating that every deterministic (potentially adaptive) tester with insufficient sample and query complexity distinguishes D_0 and D_1 only with small probability.

We start by defining our hard distributions. In both distributions, D is uniform over a large set of points, called *anchor points*, partitioned into sets A and A', both of size $2^{d/64}$. We treat A (and A'), both as a set and as an ordered sequence indexed by $i \in [2^{d/64}]$. Points in A and A' with the same index are paired up; specifically, the pairs are $(A[i], A'[i])$ for all $i \in [2^{d/64}]$. For every point in $x \in \{0,1\}^d$ and radius $t > 0$, let $\textsc{Ball}_t(x)$ denote the **open** ball centered at x, that is, the set $\{y \in \{0,1\}^d : |x - y| < t\}$. For each point $x \in A$, the function value of every point $y \in \textsc{Ball}_{\frac{r}{2}}(x)$ is equal to the distance from x to y. For each point $x \in A'$, the function value of every point $y \in \textsc{Ball}_{\frac{r}{2}}(x)$ is equal to r minus the distance from x to y where r is the desired image diameter of the functions. The points in A' are chosen so that every pair $(A[i], A'[i])$ satisfies the Lipschitz condition with equality in D_0 and violates the Lipschitz condition in D_1.

Definition 4.2. (Hard Distributions) *Fix sufficiently large* $d \in \mathbb{N}$ *and* $4 \leq r \leq 2^{-16}d$. *For all* $b \in \{0,1\}$*, let* D_b *be the distribution given by the following sampling procedure:*

1. *Sample a list* A *of* $2^{d/64}$ *elements in* $\{0,1\}^d$ *independently and uniformly at random.*

2. *Sample a list A' of the same length as A as follows. For each $i \in [2^{d/64}]$, pick the element $A'[i]$ uniformly and independently from $\{y \in \{0,1\}^d : |A[i] - y| = r - b\}$. The elements of $A \cup A'$ are called* anchor points. *Additionally, for each $i \in [2^{d/64}]$, we call $A[i]$ and $A'[i]$* corresponding anchor points.

3. *Define $f : \{0,1\}^d \to [0, r]$ by*

$$f(x) = \begin{cases} |x - A[i]| & \textit{if } x \in \textsc{Ball}_{\frac{r}{2}}(A[i]) \textit{ for some } i \in [2^{d/64}]; \\ r - |x - A'[i]| & \textit{else if } x \in \textsc{Ball}_{\frac{r}{2}}(A'[i]) \textit{ for some } i \in [2^{d/64}]; \\ r/2 & \textit{otherwise.} \end{cases}$$

4. *Output (f, U), where U is the uniform distribution over $A \cup A'$.*

Next, we define a bad event B that occurs with small probability and analyze the distance to Lipschitzness of functions arising in the support of distributions D_b, conditioned on $\overline{B}$. For a distribution D and an event E, let $D|_E$ denote the conditional distribution of a sample from D given E.

Lemma 4.1. (Distance to Lipschitzness) *Let B_0 be the event that $|A[i] - A[j]| \leq d/4$ for some distinct $i, j \in [2^{d/64}]$. Then*

1. $\Pr_{D_b}[B_0] \leq 2^{-d/32}$ *for all $b \in \{0,1\}$.*

2. *If $(f,U) \sim D_0|_{\overline{B_0}}$ then f is Lipschitz, and if $(f,U) \sim D_1|_{\overline{B_0}}$ then $\ell_{0,U}(f, \mathcal{L}ip(\mathcal{H}^d)) \geq \frac{1}{2}$.*

Proof. To prove Item 1, choose $x, y \in \{0,1\}^d$ by setting each coordinate to one independently with probability $p = \frac{1}{2}$. Let $\mu = \mathbb{E}_{x,y}[d(x,y)] = 2dp(1-p) = \frac{d}{2}$. By Chernoff bound, $\Pr_{x,y}[d(x,y) \leq \frac{\mu}{2}] \leq e^{-\frac{\mu}{8}} \leq 2^{-d/16}$. There are at most $2^{d/32}$ pairs of points in A. By a union bound over all such pairs, $\Pr[B_0] \leq 2^{-d/16} \cdot 2^{d/32} = 2^{-d/32}$. To prove Item 2, recall that $r \leq 2^{-16}d$. Suppose that B_0 did not occur. If $b = 0$ then f is Lipschitz because balls $\textsc{Ball}_{\frac{r}{2}}(x)$ are disjoint for all anchor points x. When $b = 1$, every pair $(A[i], A'[i])$ violates the Lipschitz condition, so f is 1/2-far from Lipschitz w.r.t. U. ∎

4.2.1 Indistinguishability of the Hard Distributions by a Deterministic Algorithm Fix a deterministic distribution-free Lipschitz $\frac{1}{2}$-tester $\mathcal{T}$ that gets access to input (f, U), takes $s = \frac{1}{8} \cdot 2^{d/128}$ samples from U and makes q queries to f. Since the samples from U are independent (and, in particular, do not depend on query answers), we assume w.l.o.g. that $\mathcal{T}$ receives all samples from U prior to making its queries. One of the challenges in proving that the distributions D_0 and D_1 are hard to distinguish for $\mathcal{T}$ is dealing with adaptivity. We overcome this challenge by showing that $\mathcal{T}$ can be simulated by a *nonadaptive algorithm $\mathcal{T}_{na}$ that is provided with extra information.* In addition to its samples: $\mathcal{T}_{na}$ gets at least one point from each pair $(A[i], A'[i])$, as well as function values on these points. Next, we define the extended sample given to $\mathcal{T}_{na}$ and the associated event B_1 that indicates that the sample is bad. We analyze the probability of B_1 immediately after the definition.

Definition 4.3. (Sample set, extended sample, bad sample event B_1) *Fix $b \in \{0,1\}$ and sample $(f, U) \sim D_b$. Let S denote the sample set of $\frac{1}{8} \cdot 2^{d/128}$ points obtained i.i.d. from U by the tester $\mathcal{T}$. The* extended sample *S^+ is the set $S \cup \{A[i] : i \in [2^{d/64}] \wedge A[i] \notin S \wedge A'[i] \notin S\}$. Let S^- denote the set $(A \cup A') \setminus S^+$. A set S^+ is* good *if all distinct $x, y \in S^+$ satisfy $|x - y| > d/5$ and* bad *otherwise. Define B_1 as the event that S^+ is bad.*

Lemma 4.2. (B_1 bound) *Fix $b \in \{0,1\}$. Then $\Pr_{D_b,S}[B_1] \leq \frac{1}{30}$.*

Proof. Recall the bad event B_0 from Lemma 4.1. By the law of total probability,

$$\Pr_{D_b,S}[B_1] = \Pr_{D_b,S}[B_1|B_0] \cdot \Pr_{D_b}[B_0] + \Pr_{D_b,S}[B_1|\overline{B_0}] \cdot \Pr_{D_b}[\overline{B_0}] \leq \Pr_{D_b}[B_0] + \Pr_{D_b,S}[B_1|\overline{B_0}]. \tag{4.5}$$

To bound $\Pr_{D_b,S}[B_1|\overline{B_0}]$, observe that if B_0 did not occur, then all pairs (x, y) of anchor points, except for the corresponding pairs, satisfy $|x - y| > \frac{d}{4} - 2r > \frac{d}{5}$ because $r < d \cdot 2^{-16}$. In particular, it means that all anchor points are distinct. Now, condition on $\overline{B_0}$. Then event B_1 can occur only if both $A[i]$ and $A'[i]$ for some $i \in [2^{d/64}]$

appear in the extended sample S^+. By Definition 4.3, this is equivalent to the event that both $A[i]$ and $A'[i]$ for some $i \in [2^{d/64}]$ appear in S. Each pair of samples in S is a pair of corresponding anchor points with probability at most $2^{-d/64}$. By a union bound over the at most $\frac{1}{64} \cdot 2^{d/64}$ pairs of samples taken for S, the probability that $A[i], A'[i] \in S$ for some i is at most $\frac{1}{64} \cdot 2^{d/64} \cdot 2^{-d/64} = \frac{1}{64}$. Hence, $\Pr_{D_b,S}[B_1|\overline{B_0}] \le \frac{1}{64}$. The lemma follows from Equation (4.5) and Lemma 4.1. ∎

4.2.2 The Simulator One of the key ideas in the analysis is that our hard distributions, and the sampling done by the tester, can be simulated by first obtaining the set S^+ using steps which are identical for $b = 0$ and $b = 1$, and only then selecting points in S^- to obtain the full description of the function f and the distribution U. Next, we state the simulation procedure. Note that the first 3 steps of the procedure do not use bit b, that is, are the same for simulating D_0 and D_1.

Definition 4.4. (Simulator) *Fix $b \in \{0, 1\}$. Let $\hat{D}_b$ be the distribution given by the following procedure:*

1. *Sample a list S^+ of $2^{d/64}$ elements in $\{0, 1\}^d$ independently and uniformly at random.*
2. *For each $i \in [2^{d/64}]$, do the following: if $i \le \frac{1}{8} \cdot 2^{d/128}$, then assign $S^+[i]$ to either $A[i]$ or $A'[i]$ uniformly and independently at random; if $i > \frac{1}{8} \cdot 2^{d/128}$ then assign $S^+[i]$ to $A[i]$.*
3. *Proceed as in Step 3 of the procedure in Definition 4.2 to set $f(x)$ for all $x \in S^+$.*
4. *For each $i \in [2^{d/64}]$, pick the element $S^-[i]$ uniformly and independently from $\{y \in \{0, 1\}^d : |S^+[i] - y| = r - b\}$. Assign it to $A'[i]$ if $S^+[i]$ was assigned to $A[i]$ and vice versa.*
5. *Proceed as in Step 3 of the procedure in Definition 4.2 to set $f(x)$ for all $x \notin S^+$ and output (f, U, S^+), where U is the uniform distribution over $A \cup A'$.*

Observation 4.1 states that, conditioned on $\overline{B_1}$, the simulator produces identical distributions on the extended sample S^+ and function values $f(x)$ on points $x \in S^+$, regardless of whether it is run with $b = 0$ or $b = 1$. Moreover, conditioned on $\overline{B_1}$, it faithfully simulates sampling (f, U) from D_b and S from U, and then extending S to S^+ according to the procedure described in Definition 4.3.

Observation 4.1. (Simulator facts) *Let $(f_b, U_b, S_b^+) \sim \hat{D}_b|_{\overline{B_1}}$ for each $b \in \{0, 1\}$. Let $f(S^+)$ denote function f restricted to the set S^+. Then the distribution of $(S_0^+, f(S_0^+))$ is identical to the distribution of $(S_1^+, f(S_1^+))$.*

Now fix $b \in \{0, 1\}$. Sample $(f, U) \sim D_b|_{\overline{B_1}}$ and $S \sim U$. Then the distribution of (f, U, S^+) is identical to the distribution of (f_b, U_b, S_b^+).

Proof. Since the first 3 steps in Definition 4.4 do not depend on b, the distribution of $(S_0^+, f(S_0^+))$ is the same as the distribution of $(S_1^+, f(S_1^+))$. Now, fix $b \in \{0, 1\}$. Notice that in the procedure for sampling from D_b (Definition 4.2), for all $i \in [2^{d/64}]$ the anchor point $A[i]$ is a uniformly random point in $\{0, 1\}^d$, and the anchor point $A'[i]$ is sampled uniformly from $\{y : |y - A[i]| = r - b\}$. By the symmetry of the hypercube, the marginal distribution of $A'[i]$ is uniform over $\{0, 1\}^d$. Hence, sampling $A'[i]$ uniformly at random from $\{0, 1\}^d$ and then $A[i]$ uniformly at random from $\{y : |y - A'[i]| = r - b\}$ yields the same distribution. Thus, the distribution of anchor points is the same under $\hat{D}_b$ as under D_b for each $b \in \{0, 1\}$. Now, conditioned on $\overline{B_1}$, the set S^+ contains exactly one anchor point from each corresponding pair. By the preceding remarks, we can assume the anchor point in S^+ was sampled uniformly at random from $\{0, 1\}^d$, and the corresponding anchor point was sampled uniformly from the set of points at distance $r - b$. Thus, conditioned on $\overline{B_0}$, the distribution over S^+ and S^- is the same as the distribution over S_b^+ and S_b^-. Since (U, f) and (U_b, f_b) are (respectively), uniquely determined by (S^+, S^-) and (S_b^+, S_b^-), the distribution of (U, f, S^+) is the same as the distribution of (U_b, f_b, S_b^+). ∎

4.2.3 The Bad Query Event and the Proof that Adaptivity Doesn't Help Observation 4.1 assures us that distributions of $\{(x, f(x)) : x \in S^+\}$ are the same (conditioned on $\overline{B_1}$) for both hard distributions. The function values $f(y)$ for $y \in \bigcup_{x \in S^+} \text{Ball}_{\frac{r}{2}}(x)$ are determined by $\{(x, f(x)) : x \in S^+\}$. Moreover, the function values $f(y)$ are set to $r/2$ for all $y \notin \bigcup_{x \in (S^+ \cup S^-)} \text{Ball}_{\frac{r}{2}}(x)$. So, intuitively, the tester can distinguish the distributions only if it queries a point in $\text{Ball}_{\frac{r}{2}}(x)$ for some $x \in S^-$. Our last bad event, introduced next, captures this possibility.

Definition 4.5. (Revealing point, bad query event $B_{\mathcal{T}}$) *Fix $b \in \{0,1\}$ and sample $(f,U) \sim D_b$. A point x is* revealing *if $x \in \text{Ball}_{\frac{r}{2}}(y)$ for some $y \in S^-$. Let $B_{\mathcal{T}}$ be the event that $\mathcal{T}$ queries a revealing point.*

To bound the probability of $B_{\mathcal{T}}$, we first introduce the nonadaptive tester $\mathcal{T}_{na}$ that simulates $\mathcal{T}$ to decide on all of its queries. Tester $\mathcal{T}_{na}$ gets query access to a function f sampled from D_b, a sample $S \sim U$ and $\{(x, f(x)) : x \in S^+\}$. Subsequently, in Claim 4.1, we argue that if $\mathcal{T}$ queries a revealing point then $\mathcal{T}_{na}$ queries such a point as well. This implies that $\Pr_{D_b,S}[B_{\mathcal{T}}] \le \Pr_{D_b,S}[B_{\mathcal{T}_{na}}]$, where $B_{\mathcal{T}_{na}}$ is defined analogously to $B_{\mathcal{T}}$, but for the tester $\mathcal{T}_{na}$. Finally, in Lemma 4.3, we upper bound the probability of $B_{\mathcal{T}_{na}}$ by first arguing that, conditioned on $\overline{B_1}$, the probability that $\mathcal{T}_{na}$ queries a revealing point is small. Combining this fact with the bound on $\overline{B_1}$ yields an upper bound on $B_{\mathcal{T}_{na}}$ and, consequently, $B_{\mathcal{T}}$.

Definition 4.6. ($\mathcal{T}_{na}$) *Let $\mathcal{T}_{na}$ be a nonadaptive deterministic algorithm that gets query access to f sampled from D_b, sets $S \sim U$ and $\{(x, f(x)) : x \in S^+\}$, and selects its queries by simulating $\mathcal{T}$ as follows:*

1. *Provide S as the sample and answer each query $x \in \{0,1\}^d$ with $g(x)$ defined by*

$$g(x) = \begin{cases} |x-y| & \text{if } x \in \text{Ball}_{\frac{r}{2}}(y) \text{ for some } y \in S^+ \text{ satisfying } f(y) = 0; \\ r - |x-y| & \text{else if } x \in \text{Ball}_{\frac{r}{2}}(y) \text{ for some } y \in S^+ \text{ satisfying } f(y) = r; \\ r/2 & \text{otherwise.} \end{cases}$$

2. *Let $x_1, \ldots, x_q$ be the queries made by $\mathcal{T}$ in the simulation. Query f on $x_1, \ldots, x_q$.*

Claim 4.1. (Adaptivity does not help) *If $\mathcal{T}$ queries a revealing point then $\mathcal{T}_{na}$ queries a revealing point.*

Proof. Let $x_1, \ldots, x_q$ be the queries made by $\mathcal{T}$ and $y_1, \ldots, y_q$ be the queries made by $\mathcal{T}_{na}$. Since $\mathcal{T}$ is deterministic and the sample set S is the same in both $\mathcal{T}_{na}$ and in $\mathcal{T}$, we have $x_1 = y_1$. Assume $\mathcal{T}$ queries a revealing point. Let $m \in [q]$ be the smallest index such that x_m is revealing. By definition of g and revealing point, $g(x_i) = f(x_i)$ for all $i \in [m-1]$. Consequently, $y_m = x_m$ and $\mathcal{T}_{na}$ queries a revealing point. ∎

Lemma 4.3. ($B_{\mathcal{T}}$ bound) *Fix $b \in \{0,1\}$. There exists a constant $\alpha > 0$ such that, for all sufficiently large d, if $\mathcal{T}$ makes $q = 2^{\alpha r \log(d/r)}$ queries then $\Pr_{D_b,S}[B_{\mathcal{T}}] < \frac{2}{30}$.*

Proof. By Claim 4.1, $\Pr_{D_b,S}[B_{\mathcal{T}}] \le \Pr_{D_b,S}[B_{\mathcal{T}_{na}}]$. Applying the law of total probability we obtain the inequality $\Pr_{D_b,S}[B_{\mathcal{T}_{na}}] \le \Pr_{D_b,S}[B_{\mathcal{T}_{na}} | \overline{B_1}] + \Pr_{D_b,S}[B_1]$. Next, we compute $\Pr_{D_b,S}[B_{\mathcal{T}_{na}} | \overline{B_1}]$, which is equal to $\Pr_{\hat{D}_b}[B_{\mathcal{T}_{na}} | \overline{B_1}]$, since by Observation 4.1, the simulator faithfully simulates sampling $(f,U) \sim D_b$ and then obtaining $S \sim U$, conditioned on $\overline{B_1}$. By the principle of deferred decisions, we can stop the simulator after Step 3, then consider queries from $\mathcal{T}_{na}$, and only then run the rest of the simulator. Since $\mathcal{T}_{na}$ is a nonadaptive q-query algorithm, it is determined by the collection $(x_1, \ldots, x_q)$ of query points that it chooses as a function of its input (sets S and $\{(x, f(x)) : x \in S^+\}$). We will argue that the probability (over the randomness of the simulator) that the set $\{x_1, \ldots, x_q\}$ contains a revealing point (i.e., a point on which f and g disagree) is small. Consider some query x made by $\mathcal{T}_{na}$. By Definition 4.5, a point x is revealing if $x \in \text{Ball}_{\frac{r}{2}}(S^-[i])$ for some $i \in [2^{d/64}]$. Recall that each $S^-[i]$ satisfies $|S^+[i] - S^-[i]| = r - b$, and thus each revealing point is in $\text{Ball}_{3r/2}(S^+[i])$ for some $i \in [2^{d/64}]$. All such balls around anchor points in S^+ are disjoint, because $r \le 2^{-16} \cdot d$ and we are conditioning on $\overline{B_1}$ (the event that all pairs of points in S^+ are at distance greater than $d/5$).

Suppose $x \in \text{Ball}_{3r/2}(S^+[i])$ for some $i \in [2^{d/64}]$. (If not, x cannot be a revealing point.) For x to be revealing, it must be in $\text{Ball}_{\frac{r}{2}}(S^-[i])$ or equivalently, $S^-[i]$ must be in $\text{Ball}_{\frac{r}{2}}(x)$. The simulator chooses $S^-[i]$ uniformly and independently from $\{y \in \{0,1\}^d : |S^+[i] - y| = r - b\}$. The number of points at distance $r - b \ge r - 1$ from $S^+[i]$ is at least $\binom{d}{r-b} > (\frac{d}{r})^{r-1} \ge (\frac{d}{r})^{3r/4}$, where the last inequality holds because $r \ge 4$. Out of these choices, only those that are in $\text{Ball}_{\frac{r}{2}}(x)$ will make x a revealing point. The number of points in $\text{Ball}_{\frac{r}{2}}(x)$ is $\sum_{i=0}^{r/2} \binom{d}{i} \le r\binom{d}{r/2} \le r(\frac{2de}{r})^{r/2}$. Then

$$\Pr_{\hat{D}_b}[x \text{ is revealing} \mid \overline{B_1}] \le r\Big(\frac{2de}{r}\Big)^{r/2}\Big(\frac{r}{d}\Big)^{3r/4} \le r(2e)^{r/2}\Big(\frac{r}{d}\Big)^{r/4} = 2^{\log(r) + \frac{r}{2}\log(2e) - \frac{r}{4}\log(d/r)}$$
$$\le 2^{2r - \frac{r}{4}\log(d/r)} \le 2^{-\frac{r}{8}\log(d/r)} < \frac{1}{30} \cdot 2^{-\alpha r \log(d/r)},$$

where the first inequality in the second line holds because $r \geq 4$, the next inequality holds since $r \leq 2^{-16}d$ and, for the last inequality, we set $\alpha = \frac{1}{32}$ and use both bounds on r. By a union bound over the $q = 2^{-\alpha r \log(d/r)}$ queries, $\Pr_{D_b,S}[B_{\mathcal{T}_{na}}|\overline{B_1}] = \Pr_{\hat{D}_b}[B_{\mathcal{T}_{na}}|\overline{B_1}] < \frac{1}{30}$. Using the bound from Lemma 4.2 on the probability of $\overline{B_1}$, we obtain

$$\Pr_{D_b,S}[B_{\mathcal{T}_{na}}] \leq \Pr_{D_b,S}[B_{\mathcal{T}_{na}}|\overline{B_1}] + \Pr_{D_b,S}[\overline{B_1}] \ \leq \frac{2}{30},$$

completing the proof of Lemma 4.2. □

4.2.4 Proof of Distribution-Free Testing Lower Bound Before proving Theorem 4.2, we argue that conditioned on $\overline{B_1 \cup B_{\mathcal{T}}}$, the distribution of samples and query answers seen by $\mathcal{T}$ is the same whether $(f, U) \sim D_0$ or $(f, U) \sim D_1$.

Definition 4.7. (D-view) *For all distributions D over instances of distribution-free testing, and all t-sample, q-query deterministic algorithms, let D-view be the distribution over samples $s_1, \ldots, s_t$ and query answers $a_1, \ldots, a_q$ seen by the algorithm on input (f, U) when $(f, U) \sim D$.*

Lemma 4.4. (Equal conditional distributions) *D_0-view$|_{\overline{B_1 \cup B_{\mathcal{T}}}} = D_1$-view$|_{\overline{B_1 \cup B_{\mathcal{T}}}}$.*

Proof. Conditioned on $\overline{B_1}$ and $\overline{B_{\mathcal{T}}}$, every query answer $f(x)$ given to $\mathcal{T}$ is determined by the function g in Definition 4.6. In particular, every query answer is a deterministic function of the points in S^+, the restricted function $f(S^+)$, and (possibly) previous query answers. By Observation 4.1, the distribution of $(S^+, f(S^+))$ is the same under both $D_0|_{\overline{B_1}}$ and $D_1|_{\overline{B_1}}$. Hence, the distribution of query answers $f(x_1), ..., f(x_q)$ is identical. The lemma follows. □

Next, we recall some standard definitions and facts that are useful for proving query lower bounds.

Definition 4.8. (Notation for statistical distance) *For two distributions D_1 and D_2 and a constant δ, let $D_1 \approx_\delta D_2$ denote that the statistical distance between D_1 and D_2 is at most δ.*

Fact 4.1. (Claim 4 [RS06]) *Let E be an event that happens with probability at least $1-\delta$ under the distribution D and let B denote the conditional distribution $D|_E$. Then $B \approx_{\delta'} D$ where $\delta' = \frac{1}{1-\delta} - 1$.*

We use the version of Yao's principle with two distributions from [RS06].

Fact 4.2. (Claim 5 [RS06]) *To prove a lower bound q on the worst-case query complexity of a randomized property testing algorithm, it is enough to give two distributions on inputs: $\mathcal{P}$ on positive instances, and $\mathcal{N}$ on negative instances, such that $\mathcal{P}$-view $\approx_\delta \mathcal{N}$-view for some $\delta < \frac{1}{3}$.*

We now complete the proof of Theorem 4.2, the main theorem on distribution-free testing.

Proof of Theorem 4.2. We apply Fact 4.2 (Yao's principle) with $\mathcal{P} = D_0|_{\overline{B_0}}$ and $\mathcal{N} = D_1|_{\overline{B_0}}$. By Lemma 4.1, $D_0|_{\overline{B_0}}$ is over positive instances and $D_1|_{\overline{B_0}}$ is over negative instances of distribution-free Lipschitz $\frac{1}{2}$-testing. Let $\delta_0 = \Pr[B_0]$ and $\delta_1 = \Pr[B_1 \cup B_{\mathcal{T}}]$. Set $\delta_0' = \frac{1}{1-\delta_0} - 1$ and $\delta_1' = \frac{1}{1-\delta_1} - 1$. By Fact 4.1, we have the following chain of equivalences:

$$D_0\text{-view}|_{\overline{B_0}} \approx_{\delta_0'} D_0\text{-view} \approx_{\delta_1'} D_0\text{-view}|_{\overline{B_1 \cup B_{\mathcal{T}}}} = D_1\text{-view}|_{\overline{B_1 \cup B_{\mathcal{T}}}} \approx_{\delta_1'} D_1\text{-view} \approx_{\delta_0'} D_1\text{-view}|_{\overline{B}},$$

where the equality follows from Lemma 4.4. By Lemmas 4.1, 4.2 and 4.3 (that upper bound the probabilities of bad events), for sufficiently large d, we have $\delta_0' \leq \frac{1}{27}$, and $\delta_1' \leq \frac{1}{9}$. Hence, $2(\delta_0' + \delta_1') < \frac{1}{3}$. Theorem 4.2 now follows from Yao's principle (as stated in Fact 4.2). □

5 Application to Differential Privacy

In this section, we show how to use a local Lipschitz filter for bounded-range functions to construct a mechanism (Theorem 5.1) for privately releasing outputs of bounded-range functions even when the client is malicious (i.e., lies about the range or Lipschitz constant of the function). Then, we show how the mechanism can be extended to privately release outputs of unbounded-range functions (Theorem 5.2).

5.1 Preliminaries on Differentially Private Mechanisms We start by defining the Laplace mechanism, used in the proofs of Theorems 5.1 and 5.2. It is based on the Laplace distribution, denoted Laplace(λ), that has probability density function $f(x) = \frac{1}{2\lambda}e^{-|x|/\lambda}$. We use abbreviation (ε, δ)-DP for "(ε, δ)-differentially private" (see Definition 1.1).

LEMMA 5.1. (LAPLACE MECHANISM [DMNS06]) *Fix $\varepsilon > 0$ and $c > 1$. Let $f : [n]^d \to \mathbb{R}$ be a c-Lipschitz function. Then the mechanism that gets a query $x \in [n]^d$ as input, samples $N \sim$ Laplace$(\frac{c}{\varepsilon})$, and outputs $L(x) = f(x) + N$, is $(\varepsilon, 0)$-DP. Furthermore, for all $\alpha \in (0,1)$, the mechanism satisfies $|L(x) - f(x)| \le \frac{c}{\varepsilon} + \ln\frac{1}{\alpha}$ with probability at least $1-\alpha$.*

In addition to the Laplace mechanism (Lemma 5.1), the proof of Theorem 5.2 uses the following well known facts about differentially private algorithms. These can be found in [CD14].

FACT 5.1. (COMPOSITION) *Fix $\varepsilon_1, \varepsilon_2 > 0$ and $\delta_1, \delta_2 \in (0,1)$. Suppose $\mathcal{M}_1$ and $\mathcal{M}_2$ are (respectively) $(\varepsilon_1, \delta_1)$-DP and $(\varepsilon_2, \delta_2)$-DP. Then, the mechanism that, on input x, outputs $(\mathcal{M}_1(x), \mathcal{M}_2(x))$ is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP.*

FACT 5.2. (POST-PROCESSING) *Fix $\varepsilon > 0$ and $\delta \in (0,1)$. Suppose $\mathcal{M} : D \to R$ is an (ε, δ)-DP mechanism. If $\mathcal{A}$ is an algorithm with input space R then the algorithm given by $\mathcal{A} \circ \mathcal{M}$ is (ε, δ)-DP.*

FACT 5.3. *If $X \sim$ Laplace(λ) then $\Pr[|X| \ge t\lambda] \le e^{-t}$ for all $t > 0$.*

In particular, if $\lambda = \frac{\log r}{\varepsilon}$ and $t = \log(200 \log r)$ then $\Pr[|X| \ge \frac{\log(r)\log(200\log r)}{\varepsilon}] \le \frac{1}{200 \log r}$.

5.2 Mechanism for Bounded-Range Functions The filter mechanism can be instantiated with either one of our filters (from Theorem 2.1 or from Theorem 3.1), providing slightly different accuracy guarantees. In Theorem 5.1, we state the guarantees for the mechanism based on the l_1-respecting filter. Next, we establish the terminology used in the theorem. Recall that $\textsc{Ball}_R(x)$ denotes the set $\{y : |x-y| < R\}$. We say a vertex $x \in \{0,1\}^d$ is *dangerous* w.r.t. f if there exists a vertex $y \in \{0,1\}^d$ such that $|f(x) - f(y)| > \text{dist}_G(x,y)$. A client that submits a Lipschitz function is called *honest*; a client that submits a non-Lipschitz function that is close to Lipschitz is called *clumsy* (the distance measure could be ℓ_1 or ℓ_0, depending on the filter used). Finally, we assume that sampling from the Laplace distribution requires unit time.

THEOREM 5.1. (FILTER MECHANISM) *For all $\varepsilon > 0$ and $\delta \in (0,1)$, there exists an (ε, δ)-differentially private mechanism $\mathcal{M}$ that, given a query $x \in [n]^d$, lookup access to a function $f : [n]^d \to [0, r]$, and range diameter $r \in \mathbb{R}$, outputs a value $h(x) \in \mathbb{R}$, and has the following properties.*

- ***Efficiency:*** *The lookup and time complexity of $\mathcal{M}$ are $(d^r \cdot \text{polylog}(n/\delta))^{O(\log r)}$.*
- ***Accuracy for an honest client:*** *If f is Lipschitz then for all $x \in [n]^d$ we have $h(x) \sim f(x) +$ Laplace$(\frac{2}{\varepsilon})$.*
- ***Accuracy for a clumsy client:*** *For all $x \in [n]^d$ such that $\textsc{Ball}_{r\log_{3/2}(r)}(x)$ does not contain any dangerous vertices, the "accuracy for an honest client" guarantee holds. Moreover, with probability at least $1-2\delta$, the mechanism satisfies $\mathbb{E}_{z\sim[n]^d}[|h(z) - f(z)|] \le 2\ell_1(f, \mathcal{L}ip(\mathcal{H}_n^d)) + O(\frac{1}{\varepsilon})$.*

We stress that the differential privacy guarantee in Theorem 5.1 holds whether or not the client is honest.

Proof of Theorem 5.1. Fix $\varepsilon > 0$ and $\delta \in (0,1)$. Let $\mathcal{A}$ denote $\textsc{LocalFilter}_1$ (Algorithm 3) run with iteration parameter $t = \log_{3/2}(r/2) + 1$. Recall that by Theorem 2.1 instantiated with $\gamma = 1$ and failure probability δ, the algorithm $\mathcal{A}$ is an ℓ_1-respecting local $(2, \delta)$-Lipschitz filter with blowup 2 over the d-dimensional hypergrid $\mathcal{H}_n^d$. The "efficiency" and "accuracy for honest client" guarantees hold for any local Lipschitz filter of the type stated in Theorem 2.1. However, the first guarantee of "accuracy for a clumsy client" requires properties specific to the construction of $\textsc{LocalFitler}_1$.

Let $\mathcal{M}$ be the following mechanism: *Sample a random seed ρ of length specified in Theorem 2.1, run $\mathcal{A}(x, \rho)$ to obtain $g_\rho(x)$, sample $N \sim$ Laplace$(\frac{2}{\varepsilon})$, and output $g_\rho(x) + N$.*

First, we prove that $\mathcal{M}$ is (ε, δ)-differentially private. If the function g_ρ is 2-Lipschitz, then, by Lemma 5.1 instantiated with $c = 2$ and privacy parameter ε, the mechanism $\mathcal{M}$ is $(\varepsilon, 0)$-DP. Conditioned on the event that g_ρ is 2-Lipschitz, we obtain that for all measurable sets $Y \subset \mathbb{R}$,

$$\Pr[\mathcal{M}(x,\rho) \in Y \mid g_\rho \text{ is 2-Lipschitz}] \le e^\varepsilon \Pr[\mathcal{M}(x',\rho) \in Y \mid g_\rho \text{ is 2-Lipschitz}].$$

By Theorem 2.1, $\mathcal{A}$ fails to output a 2-Lipschitz function with probability at most δ. By the law of total probability,

$$\begin{aligned}\Pr[\mathcal{M}(x,\rho) \in Y] &\le e^\varepsilon \Pr[\mathcal{M}(x',\rho) \in Y \mid g_\rho \text{ is 2-Lipschitz}] \Pr[g_\rho \text{ is 2-Lipschitz}] + \delta \\ &\le e^\varepsilon \Pr[\mathcal{M}(x',\rho) \in Y] + \delta.\end{aligned}$$

The efficiency guarantee follows directly from Theorem 2.1. The accuracy guarantee for an honest client holds since Theorem 2.1 guarantees that if f is Lipschitz, then $\mathcal{A}(x,\rho) = f(x)$ for all x and ρ. The average accuracy guarantee for the clumsy client follows from the ℓ_1-respecting, 2-blowup guarantee of Theorem 2.1 and the fact that $\mathbb{E}[|\text{Laplace}(\frac{2}{\varepsilon})|] \le O(\frac{1}{\varepsilon})$. To demonstrate that the stronger accuracy guarantee holds when no dangerous vertices are in $\textsc{Ball}_{r\log_{3/2}(r)}(x)$, we make the following observation: if no vertex $y \in \textsc{Ball}_{r\log_{3/2}(r)}(x)$ is dangerous, then $\mathcal{A}(x,\rho) = f(x)$. We prove this claim as follows. Recall that $t = \log_{3/2}(r/2) + 1$ and, for each $i \in [t]$, let $\mathcal{A}(x,i,\rho)$ denote the output after iteration i of Algorithm 3. By construction, $\mathcal{A}(x,1,\rho) = f(x)$. Suppose no vertex $y \in \textsc{Ball}_{r(t-1)}(x)$ is dangerous. Then, since the range of f is $[0,r]$, every dangerous vertex v can, in a single iteration, only create new dangerous vertices in $\textsc{Ball}_r(v)$. Thus, in $t-1$ iterations, no dangerous vertices can be introduced in $\textsc{Ball}_r(x)$. Since $\mathcal{A}(x,1,\rho) = f(x)$, and in every subsequent iteration $1 < i \le t$, no dangerous vertices are in $\textsc{Ball}_{r(i-1)}(x)$, we obtain $\mathcal{A}(x,t,\rho) = \mathcal{A}(x,1,\rho) = f(x)$ for each $i \in [t]$. Thus, if no dangerous vertex is in $\textsc{Ball}_{r\log_{3/2}(r)}(x)$ then $\mathcal{A}(x,\rho) = f(x)$, and hence, the "accuracy for an honest client" guarantee holds. ∎

5.3 Mechanism for Unbounded Range Functions In this section, we use the mechanism for bounded-range functions to construct a mechanism for arbitrary-range functions.

Theorem 5.2. (Binary search filter mechanism) *For all $\varepsilon > 0$ and $\delta \in (0, \frac{1}{200})$, there exists an (ε,δ)-differentially private mechanism $\mathcal{M}$ that, given a query $x \in [n]^d$, lookup access to a function $f : [n]^d \to [0,\infty)$ and an optional range parameter $r \in \mathbb{R}$, outputs value $h(x) \in \mathbb{R}$ and has the following properties.*

Let $\kappa = \log\min(r, nd)$, where the optional parameter r is set to ∞ by default.

- ***Efficiency:*** *The lookup and time complexity of $\mathcal{M}$ are $d^{O(\frac{1}{\varepsilon}\kappa\log\kappa)}\,\text{polylog}(\frac{n}{\delta})$.*
- ***Accuracy for an honest client:*** *If f is Lipschitz then, for all $x \in [n]^d$, we have $h(x) \sim f(x) + \text{Laplace}(\frac{\kappa}{\varepsilon})$ with probability at least 0.99.*
- ***Accuracy for a clumsy client:*** *There exists a constant $c > 0$ such that for all $x \in [n]^d$, if $f(x) \le nd$ and $|f(x) - f(y)| \le |x-y|$ for all $y \in \textsc{Ball}_{\frac{c}{\varepsilon}\kappa\log\kappa}(x)$, then the "accuracy for an honest client" guarantee holds.*

As in Theorem 5.1, we emphasize that the differential privacy guarantee holds whether or not the client is honest. Note that the accuracy guarantee for an honest client is subsumed by the guaranty for a clumsy client, but we state the former guarantee separately for clarity.

Proof of Theorem 5.2. Our private mechanism is presented in Algorithm 6. It uses the following "clipping" operation to truncate the range of a function.

Definition 5.1. (Clipped function) *For any $f : V \to \mathbb{R}$ and interval $[\ell, u] \subset \mathbb{R}$, the* clipped function *$f[\ell,u]$ is defined by*

$$f[\ell,u](x) = \begin{cases} f(x) & f(x) \in [\ell,u]; \\ \ell & f(x) < \ell; \\ u & f(x) > u. \end{cases}$$

Algorithm 6 Binary search filter mechanism $\mathcal{M}(x, f, \varepsilon)$

Input: Dataset $x \in [n]^d$, lookup access to $f : [n]^d \to [0, \infty)$, range diameter $r \in \mathbb{R}$, adjacency lists access to the hypercube $\mathcal{H}^d$, $\varepsilon > 0$, and $\delta \in (0, 1)$
Subroutines: Local $(1, \frac{\delta}{\log r})$-Lipschitz filter $\mathcal{A}$ obtained in Theorem 3.1
Output: Noisy value $h(x)$ satisfying the guarantees of Theorem 5.2

1: set $r \leftarrow \min(r, nd)$ and $f \leftarrow f[0, r]$ ▷ If the client is honest then $f[0, r] = f$.
2: set $t \leftarrow r/2$ and $\alpha \leftarrow \frac{1}{\varepsilon}\log(r)\log(200\log r)$
3: **for** $i = 2$ **to** $\lceil \log r \rceil$ **do**
4: let $h(x) \leftarrow \mathcal{A}(x, f[t - 2\alpha, t + 2\alpha]) + \text{Laplace}(\frac{\log r}{\varepsilon})$
5: **if** $h(x) \in [t - \alpha, t + \alpha]$ **then**
6: **return** $h(x)$
7: **else** $t \leftarrow t + \text{sign}(h(x) - t) \cdot \lceil r/2^i \rceil$
8: **return** $h(x)$

Next, we complete the analysis of the binary search filter mechanism. We first argue that $\mathcal{M}$ (Algorithm 6) is (ε, δ)-DP. In every iteration of the **for**-loop, $\mathcal{M}$ uses Laplace mechanism on a function that is 1-Lipschitz with probability at least $1 - \frac{\delta}{\log r}$. By Lemma 5.1 and an argument similar to the proof of Theorem 5.1, each iteration is$(\frac{\varepsilon}{\log r}, \frac{\delta}{\log r})$-DP. It follows by Facts 5.1 and 5.2 that Algorithm 6 is (ε, δ)-DP.

Next, we prove the accuracy guarantee for the clumsy client. Observe that it subsumes the accuracy guarantee for the honest client. Suppose $f : [n]^d \to [0, r]$ and that x satisfies $|f(x) - f(y)| \le |x - y|$ for all $y \in \textsc{Ball}_\alpha(x)$ (the α in line 2 of Algorithm 6). Then, for all intervals $\mathcal{I}$ of diameter α, the point x satisfies $|f[\mathcal{I}](x) - f[\mathcal{I}](y)| \le |x - y|$ for all $y \in [n]^d$. By Theorem 3.1, $\mathcal{A}(x, f[\mathcal{I}]) = f[\mathcal{I}](x)$, which is equal to $f(x)$ whenever $f(x) \in \mathcal{I}$.

Condition on the event that $\mathcal{A}$ does not fail and that the Laplace noise added is strictly less than α in every iteration of $\mathcal{M}$. Then, if $h(x) \in [t - \alpha, t + \alpha]$, we must have $f[t - 2\alpha, t + 2\alpha](x) \in (t - 2\alpha, t + 2\alpha)$, and therefore $f[t - 2\alpha, t + 2\alpha](x) = f(x)$. Next, suppose $h(x) \notin [t - \alpha, t + \alpha]$. If $f(x) < t$ then $h(x) < t + \alpha$ and thus $h(x) < t - \alpha$. Similarly, if $f(x) > t$ then $h(x) > t - \alpha$ and thus $h(x) > t + \alpha$. It follows that in every iteration $\mathcal{M}$ either continues the binary search in the correct direction, or halts and outputs $h(x)$ such that $|h(x) - f(x)| \le \alpha$. By the union bound and the guarantee obtained in Theorem 3.1, the algorithm $\mathcal{A}$ fails in some iteration of $\mathcal{M}$ with probability at most δ. Moreover, by Fact 5.3 and the union bound, the Laplace noise added is at least α in some iteration of $\mathcal{M}$ with probability at most $\frac{1}{200}$. It follows that for sufficiently small δ, the mechanism $\mathcal{M}$ outputs $h(x)$ such that $h(x) \sim f(x) + \text{Laplace}(\frac{\log \min(r, nd)}{\varepsilon})$ with probability at least $\frac{99}{100}$. □

6 Application to Tolerant Testing

In this section, we give an efficient algorithm for tolerant Lipschitz testing of real-valued functions over the d-dimensional hypercube $\mathcal{H}^d$ and prove Theorem 6.1, which we restate here for convenience.

Theorem 6.1. *For all $\varepsilon \in (0, \frac{1}{3})$ and all sufficiently large $d \in \mathbb{N}$, there exists an $(\varepsilon, 2.01\varepsilon)$-tolerant tester for the Lipschitz property of functions on the hypercube $\mathcal{H}^d$. The tester has query and time complexity $\frac{1}{\varepsilon^2} d^{O(\sqrt{d \log(d/\varepsilon)})}$.*

Our tester utilizes the fact that the image of a function which is close to Lipschitz exhibits a strong concentration about its mean on most of the points in the domain. Hence, if a function is close to Lipschitz, it can be truncated to a small interval around its mean without modifying too many points. This truncation guarantees that the the local filter in Theorem 3.1 runs in time subexponential in d. A key idea in the truncation procedure is that if a function f is ε-close to Lipschitz then either not very many values are truncated, or the truncated function is close to Lipschitz.

To prove Theorem 6.1, we design an algorithm (Algorithm 7) that, for functions $f : \{0, 1\}^d \to \mathbb{R}$, accepts if f is ε-close to Lipschitz, rejects if f is 2.01ε-far from Lipschitz and fails with probability at most $\frac{45}{100}$. The success probability can then be amplified to at least $\frac{2}{3}$ by repeating the algorithm $\Theta(1)$ times and taking the majority answer. Before presenting Algorithm 7, we introduce some additional notation. For all functions f and intervals $\mathcal{I} \subset \mathbb{R}$, the partial function $f_\mathcal{I}$ is defined by $f_\mathcal{I}(x) = f(x)$ whenever $f(x) \in \mathcal{I}$ and $f_\mathcal{I}(x) =?$ otherwise. Additionally, for all events E, let $\mathbf{1}_E$ denote the indicator for the event E. Algorithm 7 runs the ℓ_0-filter given by

Algorithm 5 with lookup access to a partial function $f_{\mathcal{I}}$. Our analysis of Algorithm 5 presented in Theorem 3.1 is for total functions. In Observation 6.1, we extend it to partial functions.

OBSERVATION 6.1. *Let $h : [n]^d \to \mathbb{R} \cup \{?\}$ be a partial function with ℓ_0-distance to the nearest Lipschitz partial function equal to ε_h. Let $\mathcal{A}$ denote Algorithm 5. Then, for all $\delta \in (0, 1)$, the algorithm $\mathcal{A}^h$ provides query access to a Lipschitz partial function g such that $g(x) =?$ if and only if $h(x) =?$, and $\|g - h\|_0 \le 2\varepsilon_h$. The runtime and failure probability guarantees are as in Theorem 3.1.*

Proof. Consider $x \in h^{-1}(?)$. By Definition 1.4, $VS_h(x, y) = 0$ for all $y \in [n]^d$, and hence the vertex x is not incident to any edge of the violation graph of h. By construction, $\mathcal{A}^h(x) = h(x) =?$. Next, consider the induced subgraph G of $\mathcal{H}^d$ with vertex set $V = \{x : h(x) \ne ?\}$. By Claim 3.2, the size of the minimum vertex cover of the violation graph of h is at most $|V|\varepsilon_h$. It follows that $\mathcal{A}^h$ provides query access to a Lipschitz partial function g such that $\|g - h\|_0 \le \varepsilon_h$. The runtime and failure probability are the same as for total functions by definition of the algorithm. ∎

Algorithm 7 Tolerant Lipschitz tester $\mathcal{T}(f, \varepsilon)$

Input: Query access to $f : \{0,1\}^d \to \mathbb{R}$, adjacency lists access to $\mathcal{H}^d$, and $\varepsilon \in (0, \frac{1}{3})$
Subroutines: Local $(1, \frac{1}{100})$-Lipschitz filter $\mathcal{A}$ given by Algorithm 5
Output: accept or reject

1: sample a point $p \sim \{0,1\}^d$ uniformly at random and a random seed ρ of length specified in Theorem 3.1
2: set $t \leftarrow 2\sqrt{d \log(d/\varepsilon)}$ and $\mathcal{I} \leftarrow [f(p) - t, f(p) + t]$
3: sample a set S of $(\frac{1500}{\varepsilon})^2$ points uniformly and independently from $\{0,1\}^d$
4: **for all** $x_i \in S$ **do**
5:     **if** $f_{\mathcal{I}}(x_i) =?$ **then** set $y_i \leftarrow ?$
6:     **else** set $y_i \leftarrow \mathcal{A}(x_i, \rho)$, where $\mathcal{A}$ is run with lookup access to $f_{\mathcal{I}}$ and adjacency lists access to $\mathcal{H}^d$
7: **if** $\frac{1}{|S|} \sum_{x_i \in S} \mathbf{1}_{f(x_i) \ne y_i} < 2.005\varepsilon$ **then accept**
8: **else reject**

We use McDiarmid's inequality [McD89], stated here for the special case of the $\{0,1\}^d$ domain.

FACT 6.1. (MCDIARMID'S INEQUALITY[McD89]) *Fix $d \ge 2$ and let $g : \{0,1\}^d \to \mathbb{R}$ be a Lipschitz function w.r.t. $\mathcal{H}^d$. Let $\mu_g = \mathbb{E}_{x \sim \{0,1\}^d}[g(x)]$. Then, for all $\gamma \in (0, 1)$,*

$$\Pr_{x \sim \{0,1\}^d}[|g(x) - \mu_g| \ge \sqrt{d \log(d/\gamma)}] \le \frac{\gamma}{d}.$$

Next, we introduced a definition which, for each function f, attributes some part of its ℓ_0-distance to Lipschitz to a particular interval $\mathcal{I}$ in the range of f.

DEFINITION 6.1. *Let $f : \{0,1\}^d \to \mathbb{R}$ and C be a minimum vertex cover of the violation graph of f. (If there are multiple vertex covers, use any rule to pick a canonical one.) For an interval $\mathcal{I} \subset \mathbb{R}$ define $\varepsilon[\mathcal{I}]$ as $|\{x \in C : f(x) \in \mathcal{I}\}|/2^d$.*

For a function f, let ε_f denote the ℓ_0-distance from f to Lipschitz. Then $\varepsilon_f = \varepsilon[\mathcal{I}] + \varepsilon[\overline{\mathcal{I}}]$ for all intervals $\mathcal{I}$. Moreover, since $f_{\mathcal{I}}$ is a partial function defined only on points x such that $f(x) \in \mathcal{I}$, the distance from $f_{\mathcal{I}}$ to the nearest Lipschitz partial function is at most $\varepsilon[\mathcal{I}]$. Using Definition 6.1, we argue that if $\varepsilon_f \le \varepsilon$, then with high probability the interval $\mathcal{I}$ chosen in Algorithm 7 satisfies $\|f_{\mathcal{I}} - f\|_0 \le \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d}$. Since $\varepsilon[\mathcal{I}] + \varepsilon[\overline{\mathcal{I}}] \le \varepsilon$, Lemma 6.1 implies that if $\|f_{\mathcal{I}} - f\|_0$ is very close to ε, then the distance of $f_{\mathcal{I}}$ to Lipschitz, which is at most $\varepsilon[\mathcal{I}]$, must be small. Leveraging this fact, we can approximate the ℓ_0-distance from f to Lipschitz using the distance from f to $f_{\mathcal{I}}$ and the distance from $f_{\mathcal{I}}$ to Lipschitz.

LEMMA 6.1. *Fix $\varepsilon \in (0, \frac{1}{3})$ and $d \ge 4$. Let $f : \{0,1\}^d \to \mathbb{R}$ be ε-close to Lipschitz over $\mathcal{H}^d$. Choose $p \in \{0,1\}^d$ uniformly at random. Set $t \leftarrow 2\sqrt{d \log(d/\varepsilon)}$ and $\mathcal{I} \leftarrow [f(p) - t, f(p) + t]$. Then $\|f - f_{\mathcal{I}}\|_0 > \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d}$ with probability at most $\frac{5}{12}$.*

Proof. Let C be a minimum vertex cover of the violation graph $B_{0,f}$ (see Definition 1.5). Let g be a Lipschitz function obtained by extending f from $\{0,1\}^d \setminus C$ to $\{0,1\}^d$ (such an extension exists by Claim 3.1). By Fact 6.1, $\Pr_{x\sim\{0,1\}^d}[|g(x)-\mu_g| \geq \frac{t}{2}] \leq \frac{\varepsilon}{d}$. Notice that if $|f(p)-\mu_g| \leq \frac{t}{2}$ then $[\mu_g - \frac{t}{2}, \mu_g + \frac{t}{2}] \subset \mathcal{I}$. Conditioned on this occurring,

$$\Pr_x[f(x) \notin \mathcal{I}] \leq \varepsilon[\overline{\mathcal{I}}] + \Pr_x[g(x) \notin \mathcal{I}] \leq \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d}.$$

Since $\Pr_{x\sim\{0,1\}^d}[x \in C] \leq \varepsilon$, we have $|f(p)-\mu_g| \leq \frac{t}{2}$ with probability at most $\varepsilon + \frac{\varepsilon}{d} \leq \frac{5}{12}$. ◻

Next, we argue that, after boosting the success probability via standard amplification techniques, we obtain a $(\varepsilon, 2.01\varepsilon)$-tolerant Lipschitz tester.

Proof of Theorem 6.1. Fix $\varepsilon \in (0, \frac{1}{3})$ and let $f : \{0,1\}^d \to \mathbb{R}$ and let $\mathcal{T}$ denote Algorithm 7. Define the following events: Let E_1 be the event that local filter $\mathcal{A}$ fails. Set $\omega = \Pr_x[f(x) \neq \mathcal{A}(x,\rho)]$ and $\hat{\omega} = \frac{1}{|S|}\sum_{x_i \in S} \mathbf{1}_{f(x_i)\neq y_i}$, and let E_2 be the event that $|\omega - \hat{\omega}| \geq \frac{\varepsilon}{300}$.

Suppose f is ε-close to Lipschitz, and let E_3 be the event that the interval $\mathcal{I}$ chosen in $\mathcal{T}$ satisfies $\|f - f_{\mathcal{I}}\| > \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d}$. Condition on the event that none of E_1, E_2, and E_3 occur. Then $f_{\mathcal{I}}$ is at distance at most $\varepsilon[\mathcal{I}]$ from some Lipschitz partial function and, by Observation 6.1, $\mathcal{A}$ provides query access to a Lipschitz partial function g such that $\|f - g\|_0 \leq 2\varepsilon[\mathcal{I}]$. Using the fact that $\varepsilon[\mathcal{I}] + \varepsilon[\overline{\mathcal{I}}] \leq \varepsilon$ we obtain

$$\hat{\omega} \leq 2\varepsilon[\mathcal{I}] + \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d} + \frac{\varepsilon}{300} < 2.005\varepsilon$$

for sufficiently large d. Hence $\mathcal{T}$ accepts.

Now consider the case that f is 2.01ε-far from Lipschitz and suppose neither of the events E_1 and E_2 occur. Since $\mathcal{A}$ provides query access to a Lipschitz function, and the nearest Lipschitz function is at distance at least 2.01ε, we must have $\hat{\omega} \geq 2.01\varepsilon - \frac{\varepsilon}{300} > 2.005\varepsilon$. Hence $\mathcal{T}$ rejects.

Next, we show that the events E_1, E_2 and E_3 all occur with small probability. By Observation 6.1, $\Pr[E_1] \leq \frac{1}{100}$. To bound $\Pr[E_2]$, notice that $\mathbb{E}[\hat{w}] = w$ and $\mathrm{Var}[\hat{w}] \leq \frac{1}{4|S|}$. By Chebyshev's inequality and our choice of $|S|$ we have, $\Pr_S[|w - \hat{w}| \geq \frac{\varepsilon}{300}] \leq \frac{300^2}{4|S|\varepsilon^2} \leq \frac{1}{100}$. Moreover, if f is ε-close to Lipschitz, then by Lemma 6.1, $\Pr[E_3] \leq \frac{5}{12}$. Thus, the failure probability of $\mathcal{T}$ can be bounded above by $\Pr[E_1 \cup E_2 \cup E_3] \leq \frac{2}{100} + \frac{5}{12} \leq \frac{45}{100}$. The success probability can the be boosted to $\frac{2}{3}$ by running the algorithm $O(1)$ times and taking the majority answer. Finally, we bound the query and time complexity of tester by bounding the query and time complexity of $\mathcal{T}$. The algorithm $\mathcal{T}$ runs the local filter $\mathcal{A}$ with lookup access to $f_{\mathcal{I}}$, a function with range $\mathcal{I}$ of diameter $O(\sqrt{d\log(d/\varepsilon)})$, and sets $\mathcal{A}$'s failure probability to $\delta = \frac{1}{100}$. Consequently, Observation 6.1 implies $\mathcal{T}$ has query and time complexity bound of $\frac{1}{\varepsilon^2} d^{O(\sqrt{d\log(d/\varepsilon)})}$. ◻

Acknowledgement We thank Adam Smith for comments on the initial version of this paper.

References

[ACCL07] Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31(3):371–383, 2007.

[AJMR15] Pranjal Awasthi, Madhav Jha, Marco Molinaro, and Sofya Raskhodnikova. Limitations of local filters of Lipschitz and monotone functions. *ACM Trans. Comput. Theory*, 7(1), Jan 2015.

[AJMR16] Pranjal Awasthi, Madhav Jha, Marco Molinaro, and Sofya Raskhodnikova. Testing Lipschitz functions on hypergrid domains. *Algorithmica*, 74(3):1055–1081, 2016.

[ARVX12] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In Yuval Rabani, editor, *SODA*, pages 1132–1139. SIAM, 2012.

[BCE+19] Eric Blais, Clément L. Canonne, Talya Eden, Amit Levi, and Dana Ron. Tolerant junta testing and the connection to submodular optimization and function isomorphism. *ACM Trans. Comput. Theory*, 11(4):24:1–24:33, 2019.

[BGJ+12] Arnab Bhattacharyya, Elena Grigorescu, Madhav Jha, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Lower bounds for local monotonicity reconstruction from transitive-closure spanners. *SIAM J. Discrete Math.*, 26(2):618–646, 2012.

[BGS+21] Gavin Brown, Marco Gaboardi, Adam D. Smith, Jonathan R. Ullman, and Lydia Zakynthinou. Covariance-aware private mean estimation without private covariance estimation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 7950–7964, 2021.

[BKR23] Hadley Black, Iden Kalemaj, and Sofya Raskhodnikova. Isoperimetric inequalities for real-valued functions with applications to monotonicity testing. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 25:1–25:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[BL00] Yoav Benyamini and Joram Lindenstrauss. *Geometric nonlinear functional analysis. Vol. 1*, volume 48 of *Colloquium Publications*. Amer. Math. Soc., 2000.

[BRY14] Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 164–173, 2014.

[CD14] Aaron Roth Cynthia Dwork. *The Algorithmic Foundations of Differential Privacy.* NOW, 2014.

[CDJS17] Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. Property testing on product distributions: Optimal testers for bounded derivative properties. *ACM Transactions on Algorithms (TALG)*, 13(2):20:1–20:30, 2017.

[CDL+24] Xi Chen, Anindya De, Yuhao Li, Shivam Nadimpalli, and Rocco A. Servedio. Mildly exponential lower bounds on tolerant testers for monotonicity, unateness, and juntas. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4321–4337. SIAM, 2024.

[CGG+19] Clément L. Canonne, Elena Grigorescu, Siyao Guo, Akash Kumar, and Karl Wimmer. Testing k-monotonicity: The rise and fall of Boolean functions. *Theory Comput.*, 15:1–55, 2019.

[CGR13] Andrea Campagna, Alan Guo, and Ronitt Rubinfeld. Local reconstructors and tolerant testers for connectivity and diameter. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 411–424. Springer, 2013.

[CS13] Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 419–428, 2013.

[CSVW22] Sílvia Casacuberta, Michael Shoemate, Salil P. Vadhan, and Connor Wagaman. Widespread underestimation of sensitivity in differentially private libraries and how to fix it. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 471–484. ACM, 2022.

[DGL+99] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 97–108, 1999.

[DJRT13] Kashyap Dixit, Madhav Jha, Sofya Raskhodnikova, and Abhradeep Thakurta. Testing the Lipschitz property over product distributions with applications to data privacy. In *Theory of Cryptography Conference (TCC)*, pages 418–436, 2013.

[DL09] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 371–380. ACM, 2009.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[DRTV18] Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin Varma. Erasure-resilient property testing. *SIAM Journal on Computing (SICOMP)*, 47(2):295–329, 2018.

[FF05] Eldar Fischer and Lance Fortnow. Tolerant versus intolerant testing for Boolean properties. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 135–140. IEEE Computer Society, 2005.

[FLN+02] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 474–483. ACM, 2002.

[FR10] Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Transactions on Algorithms (TALG)*, 6(3):52:1–52:37, 2010.

[Gha22] Mohsen Ghaffari. Local computation of maximal independent set. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 438–449. IEEE, 2022.

[GHK+16] Marco Gaboardi, James Honaker, Gary King, Kobbi Nissim, Jonathan R. Ullman, and Salil P. Vadhan. PSI

(Ψ): a private data sharing interface. *CoRR*, abs/1609.04340, 2016.
[HK05] Shirley Halevy and Eyal Kushilevitz. A lower bound for distribution-free monotonicity testing. In *APPROX-RANDOM*, pages 330–341, 2005.
[HK07] Shirley Halevy and Eyal Kushilevitz. Distribution-free property-testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007.
[HPN11] Andreas Haeberlen, Benjamin C. Pierce, and Arjun Narayan. Differential privacy under fire. In *20th USENIX Security Symposium (USENIX Security 11)*, San Francisco, CA, August 2011. USENIX Association.
[JR13] Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM Journal on Computing (SICOMP)*, 42(2):700–731, 2013.
[KL23] Nitin Kohli and Paul Laskowski. Differential privacy for black-box statistical analyses. *Proc. Priv. Enhancing Technol.*, 2023(3):418–431, 2023.
[KRV23] Iden Kalemaj, Sofya Raskhodnikova, and Nithin Varma. Sublinear-time computation in the presence of online erasures. *Theory of Computing*, 19(1):1–48, 2023.
[LRV22] Jane Lange, Ronitt Rubinfeld, and Arsen Vasilyan. Properly learning monotone functions via local correction. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 75–86. IEEE, 2022.
[LV23] Jane Lange and Arsen Vasilyan. Agnostic proper learning of monotone functions: beyond the black-box correction barrier. In *in 64rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023 (to appear)*. IEEE, 2023.
[LW19] Amit Levi and Erik Waingarten. Lower bounds for tolerant junta and unateness testing via rejection sampling of graphs. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.
[McD89] Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*. London Math. Soc. Lecture Notes 141, 1989.
[McS10] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9):89–97, 2010.
[PRR06] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006.
[PRW22] Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of Boolean functions. *Random Struct. Algorithms*, 60(2):233–260, 2022.
[RS06] Sofya Raskhodnikova and Adam D. Smith. A note on adaptivity in testing properties of bounded degree graphs. *Electron. Colloquium Computational Complexity*, 13(089), 2006.
[RSK+10] Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for mapreduce. In *NSDI*, pages 297–312, 2010.
[RTVX11] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *ICS*, pages 223–238, 2011.
[SS10] Michael E. Saks and C. Seshadhri. Local monotonicity reconstruction. *SIAM J. Comput.*, 39(7):2897–2926, 2010.
[Yao77] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.