Limitations of Local Filters of Lipschitz and Monotone Functions*

Pranjal Awasthi[1], Madhav Jha[2], Marco Molinaro[1], and Sofya Raskhodnikova[2]**

[1] Carnegie Mellon University, USA,
{pawasthi,molinaro}@cmu.edu.
[2] Pennsylvania State University, USA,
{mxj201,sofya}@cse.psu.edu.

Abstract. We study local filters for two properties of functions $f : \{0,1\}^d \to \mathbb{R}$: the Lipschitz property and monotonicity. A local filter with additive error a is a randomized algorithm that is given black-box access to a function f and a query point x in the domain of f. Its output is a value $F(x)$, such that (i) the *reconstructed function* $F(x)$ satisfies the property (in our case, is Lipschitz or monotone) and (ii) if the input function f satisfies the property, then for every point x in the domain (with high constant probability) the reconstructed value $F(x)$ differs from $f(x)$ by at most a. Local filters were introduced by Saks and Seshadhri (SICOMP 2010) and the relaxed definition we study is due to Bhattacharyya *et al.* (RANDOM 2010), except that we further relax it by allowing additive error. Local filters for Lipschitz and monotone functions have applications to areas such as data privacy.
We show that every local filter for Lipschitz or monotone functions runs in time exponential in the dimension d, even when the filter is allowed significant additive error. Prior lower bounds (for local filters with no additive error, i.e., with $a = 0$) applied only to more restrictive class of filters, e.g., *nonadaptive* filters. To prove our lower bounds, we construct families of hard functions and show that lookups of a local filter on these functions are captured by a combinatorial object that we call a c-connector. Then we present a lower bound on the maximum outdegree of a c-connector, and show that it implies the desired bounds on the running time of local filters. Our lower bounds, in particular, imply the same bound on the running time for a class of privacy mechanisms.

1 Introduction

In this work we study local reconstruction of properties of functions. Property-preserving data reconstruction [1] is a direction of research in sublinear algorithms that has its roots in property testing [14,10]. Some related notions include locally decodable codes [12], program checking [7] and, more generally, local computation [15,2].

To motivate the reconstruction model, consider an algorithm ALG that is computing on a large dataset and whose correctness is contingent upon the dataset satisfying a certain structural property. For example, ALG may require that its input array be sorted or

* All omitted proofs appear in the full version [3].

** P.A. is supported by NSF grant CCF-1116892. M.J. and S.R. are supported by NSF CAREER grant CCF-0845701 and NSF grant CCF-0729171. M.M. is supported by NSF grant CMMI-1024554.

that its input function be Lipschitz. In such situations, ALG could access its input via a *filter* that ensures that data seen by ALG always satisfies the desired property, modifying it at few places on the fly, if required. We can represent the input to ALG as a function f, where $f(x)$ represents the portion of the data that can be accessed on query x. Instead of accessing $f(x)$ directly, ALG makes a *query* x to the filter. The filter *looks up* the value of f on a small number of points and returns $F(x)$, where F satisfies the desired property and is as close to the original function f as possible. Thus, ALG is computing with reconstructed data F instead of its original input f.

Saks and Seshadhri [16] introduced the stronger notion of a *local filter*. It has an additional requirement that the reconstruction of $f(x)$ and $f(y)$ on two different queries x and y should be done independently. In particular, the output function F is independent of the order of the queries x made to the filter.

Local filters have many desirable features: for example, they can be implemented in a distributed setting, where several processes need to access different parts of the input, and the filter has to ensure that all the parts together are consistent with some function F that satisfies the desired property. This global consistency guarantee enables several applications of local filters described in previous work [16,5,11], including the application to data privacy that we explain below.

The main goal of this paper is to understand limitations of local filters. This is crucial in order to identify the types of tradeoffs (i.e., output quality vs. lookup complexity) available for a given application. Two natural candidate properties for this evaluation are the Lipschitz property and monotonicity of functions[3] $f : [n]^d \to \mathbb{R}$, studied in previous work [1,16,5,11]: the first is motivated by the privacy application explained below and the second is a 'benchmark' problem in property-preserving reconstruction and property testing. A function $f : [n]^d \to \mathbb{R}$ is *Lipschitz* (with respect to the ℓ_1 metric on $[n]^d$) if $|f(x) - f(y)| \leq \|x - y\|_1$ for all points x, y in the domain $[n]^d$. Intuitively, changing the argument to the Lipschitz function by a small amount does not significantly change the value of the function. A function $f : [n]^d \to \mathbb{R}$ is *monotone* if $f(x) \leq f(y)$ for all points $x \preceq y$ in the domain $[n]^d$, where $\preceq$ denotes the natural partial order on $[n]^d$: for $x = (x_1, \ldots, x_d) \in [n]^d$ and $y = (y_1, \ldots, y_d) \in [n]^d$, we have $x \preceq y$ iff $x_i \leq y_i$ for all coordinates $i \in [d]$. In other words, increasing the coordinates of the argument to a monotone function does not decrease the value of the function.

The original definition of local filters in [16] has a requirement that the filter be *distance-respecting*, that is, the reconstructed function F should not differ from the original function f on significantly more points than necessary. Bhattacharyya *et al.* [5] and Jha and Raskhodnikova [11] removed this requirement and demonstrated that it is not necessary in some applications. Their local filter is simply required to output $F = f$ if the original function has the property; otherwise, F can be an arbitrary function satisfying the property. We relax the notion of local filter further by allowing additive error. Our definition (see Definition 2.1) has an additional parameter a, and the function F can differ from f by a small amount on every point, even if f satisfies the property: namely, we require that for every x in the domain, with high constant probability $|F(x) - f(x)| \leq a$. Local filters considered in [5,11] are a special case of our local filters with $a = 0$. Our

[3] We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

goal is to determine (for small a) if there are local filters that make only $\mathrm{poly}(n, d)$ lookups in order to output the reconstructed function $F(x)$ at a given point x.

Privacy Application. We observe that local filters with small additive error can still be used in the privacy application described in [11]. Consider a server which has a private database with information about individuals, modeled as a point x in $\{0,1\}^d$, representing which of d possible types of people are present in the database. (More generally, x is modeled as a point in $[n]^d$ representing a histogram that captures how many people of each type are present.) A user who does not have direct access to x can ask the server for some information about this database by specifying a function f for the server to evaluate at the point x. The server's goal is to output a value which is close to $f(x)$ but which reveals almost no information about any single individual. Recently, the latter notion has been made precise via the concept of *differential privacy* [9]. A standard way of obtaining such guarantees is to ask users to *submit only Lipschitz* functions[4], and have the server output $f(x)$ plus some random noise depending on the desired privacy guarantee [9]. However, if a malicious user submits a function which is not Lipschitz, the differential privacy guarantee is lost. A local filter with the following properties can then be used between the server and the submitted function f to ensure the desired privacy: (i) the reconstructed function F is always Lipschitz; (ii) if f is already Lipschitz, then with high probability $|F(x) - f(x)| \leq a$ for all x, where a is a given parameter. This way, the server always evaluates a Lipschitz function F and thus has the desired privacy guarantees. Furthermore, if the user provides a valid Lipschitz function f, the mechanism outputs a value $F(x)$ in the range $f(x) \pm a$ plus a random noise; if a is reasonably small it is then absorbed in the noise. Thus, bounds on the running time and additive error of the local filter translate directly into bounds on the running time and accuracy of the corresponding privacy mechanism.

1.1 Previous Results on Local Filters

Despite the fact that local filters have been thoroughly studied, lower bounds for general (not necessarily *distance-respecting*) *adaptive* filters remained a big challenge.

Saks and Seshadhri [16] present a distance-respecting local filter for monotonicity of functions $f : [n]^d \to \mathbb{R}$ with running time $(\log n + 1)^{O(d)}$ per query. For monotonicity of functions $f : \{0,1\}^d \to \mathbb{R}$, no nontrivial (i.e., performing $o(2^d)$ lookups per query) filter is known. Saks and Seshadhri also show that a *distance-respecting* local filter for monotonicity on the domain $\{0,1\}^d$ must perform $2^{\Omega(d)}$ lookups per query. This lower bound crucially uses the fact that the filter is distance respecting, and does not apply to general local filters (even when no additive error is allowed).

As we explained, in many applications the extra requirement that the filter be distance-respecting is not necessary. Bhattacharyya *et al.* [5] studied lower bounds for local monotonicity filters which are not necessarily distance-respecting. However, their super-polynomial lower bounds only hold for *nonadaptive* filter. For the domain $\{0,1\}^d$, Bhattacharyya *et al.* show that nonadaptive filters must perform $\Omega(\frac{2^{\alpha d}}{d})$ lookups per query

[4] More generally, if a user wants to evaluate a function f with Lipschitz constant at most ℓ, where $\ell > 1$, then the Lipschitz function f/ℓ can be submitted to the server. When the noisy answer returned by the server is multiplied by ℓ, the effect is to add noise proportional to ℓ.

in the worst case, where $\alpha \geq 0.1620$. For adaptive filters, their bound quickly degrades with the number of lookups performed to *incomparable* points in the domain ($x, y \in [n]^d$ are *comparable* if $x \preceq y$ or $y \preceq x$ and *incomparable* otherwise). Specifically, their lower bounds for adaptive filters is $\Omega(\frac{2^{\alpha d - \ell}}{d})$, where ℓ is the number of lookups to points incomparable to x made on query x; for arbitrary adaptive filters, this degrades to $\Omega(d)$. Prior to our work, no super-polynomial lower bound for adaptive local monotonicity filter was known.

For the Lipschitz property, Jha and Raskhodnikova [11] obtain a deterministic nonadaptive local filter that runs in time $O((\log n + 1)^d)$ per query. They also show that the lower bound from [5] for *nonadaptive* filters, with the same statement, applies to *nonadaptive* local filters of the Lipschitz property.

Previous work left open whether it is possible to obtain (adaptive and not necessarily distance-respecting) local filters monotonicity and Lipschitz properties that make only $\mathrm{poly}(n, d)$ lookups per query.

1.2 Our Results and Techniques

We consider local a-filters, which is the relaxation of local filters that allows additive error a, as described above and formally stated in Definition 2.1. These filters do not need to be distance-respecting and can be fully adaptive. Our main results, stated in more detail in Section 2, are that even such relaxed filters need to perform a number of lookups exponential in the dimension d in order to reconstruct a Lipschitz (resp., monotone) function. (This applies even to functions on the domain $\{0,1\}^d$).

Theorem 1.1 (Limitations of Lipschitz filters). *Consider the Lipschitz property of functions $f : \{0,1\}^d \to \mathbb{R}$ and any (randomized) local (not necessarily distance-respecting) $\frac{d}{402}$-filter for this property. Then there is a function f and a query x where, with constant probability, this filter makes $2^{\Omega(d)}$ lookups.*

The additive error $a = d/402$ in the theorem above is as large as possible up to a constant factor: the trivial filter that outputs $F(x) = (f(\mathbf{0}) + f(\mathbf{1}))/2$, where $\mathbf{0}$ and $\mathbf{1}$ are all-0 and all-1 vectors, respectively, is a local $\frac{d}{2}$-filter.[5] To see this, note that (i) the reconstructed function $F(x)$ is Lipschitz and (ii) if the input function $f(x)$ is Lipschitz then $|F(x) - f(x)| = \frac{1}{2}|f(\mathbf{0}) + f(\mathbf{1}) - 2f(x)| \leq \frac{1}{2}(|f(\mathbf{0}) - f(x)| + |f(\mathbf{1}) - f(x)|) \leq \frac{1}{2}(\|\mathbf{0} - x\|_1 + \|\mathbf{1} - x\|_1) = \frac{d}{2}$ for every $x \in \{0,1\}^d$.

For monotonicity, we can prove an analogous theorem with no upper bound on a. This is explained by the fact that monotonicity is determined by the order of the values at different points and not their magnitudes. To calibrate the additive error, we state the next theorem for functions with bounded range, namely, $[0, 2a+1]$. The additive error in the theorem is also tight because for functions with that range, the trivial filter above that outputs $F(x) = (f(\mathbf{0}) + f(\mathbf{1}))/2$ is a local $(a + \frac{1}{2})$-filter.

Theorem 1.2 (Limitations of monotonicity filters). *Consider the monotonicity property of functions $f : \{0,1\}^d \to [0, 2a+1]$ and any (randomized) local a-filter for this*

[5] In order to simplify the presentation, we did not attempt to optimize this constant factor. In particular, the choice of weights $d/3$ and $2d/3$ in Definition 3.1 might not give the best factor.

property. Then there is a function f and query x where, with constant probability, this filter makes $2^{\Omega(d)}$ lookups.

To introduce the ideas used in the proofs, we focus for now on deterministic filters. To obtain lower bounds for *nonadaptive* filters in [5,11], the authors construct two collections of 'hard functions' $f^{(x,y)}$ and $f^{(\overline{x,y})}$ (satisfying the Lipschitz property) indexed by $x, y \in \{0,1\}^d$. They show that if a local filter works correctly on $f^{(x,y)}$ and $f^{(\overline{x,y})}$, as well as on a suitably defined function $h^{(x,y)}$ (violating the Lipschitz property on (x,y)), the lookups made on queries x and y need to have a structured interaction. (Note that in this case the lookups are independent of the input function because the filter is non-adaptive.) More precisely, they construct a graph over $\{0,1\}^d$ based on these interactions and show that it is a 2-transitive-closure-spanner (2-TC-spanner) for the hypercube. (TC-spanners were introduced in [6]; see Section 3 for definition and comparison with c-connectors that we introduce.) Using the lower bound on the size of a 2-TC-spanner for the hypercube from [5], it can be shown that any non-adaptive filter must use exponential lookups on one of the query points.

In the case of adaptive filters one cannot assume that the lookups made on a given query point are independent of the input function. One simple idea to try to overcome this obstacle is to consider, for each query x, the union of the lookups made on query x over all possible choice of hard functions. One can then try to apply the lower bound approach discussed in the previous paragraph. In fact this union of lookups still has strong interactions that imply a 2-TC-spanner. The problem is that this is clearly overcounting the number of lookups made by the filter on a single given function on query x. Due to the large number of 'hard functions' considered in [5,11], this overcounting makes the bound coming from the 2-TC-spanners vacuous for adaptive filters; this is where the factor 2^ℓ lost in [5] mentioned above comes from.

In order to remedy this, we build a collection of hard functions which are much 'smoother' than those from [5,11]. This allows us to use fewer functions. However, it comes at a cost: the interactions of the lookups caused by these functions are not as structured as before and do not imply a 2-TC-spanner. We introduce a type of directed graph called *c-connector* (Definition 3.2) which captures lookup interactions. When arc directions are ignored, a c-connector is a relaxation of 2-TC-spanners (as discussed in Section 3, our transformation to c-connectors preserves information on whether x is looked up on query y or vice versa, while this information is lost in the transformation to 2-TC-spanners in [5,11]). Nevertheless, we can argue that a c-connector has a large maximum outdegree, which relates to the lookup complexity. Indeed, one of the key ingredients for our lower bound is recognizing the limitations of 2-TC-spanners in this context and finding a combinatorial structure with the right amount of flexibility. Given the importance of TC-spanners (see [13] for a survey), c-connectors might find use outside of this work.

Organization. Section 2 gives basic definitions and a more detailed statement of our main results. In Section 3, we define c-connectors, the graph objects on which our lower bounds are based. In Sections 4 and 5, we develop a connection between c-connectors and local filters for the Lipschitz property and monotonicity. In Section 6, we bound the outdegree of c-connectors. Our lower bounds follow directly from putting these two parts together.

2 Definitions and Formal Statement of Results

Given a point $x \in \{0,1\}^d$, we use x_i to denote its ith coordinate and $|x|$ to denote its Hamming weight, that is, $|x| = \sum_i x_i$. We identify each point $x \in \{0,1\}^d$ with the subset of coordinates where it takes value 1, namely, $\{i : x_i = 1\}$. This gives meaning to expressions like $x \subseteq y$, $x \cap y$, $x \cup y$ and $x \setminus y$ for $x, y \in \{0,1\}^d$. For $x \in \{0,1\}^d$, the Hamming weight $|x|$ coincides with the cardinality of the set associated with x.

We now provide a formal definition of local a-filters that allow additive error a. It is stated for a general property P of functions with domain D; in our case, P will be either the Lipschitz property or monotonicity.

Definition 2.1 (Local a-filter). *Let* P *be a property of functions $f : D \to R$ for some $R \subseteq \mathbb{R}$. A local a-filter for* P *with error probability δ is a randomized algorithm which is given black-box access to a function $f : D \to R$ together with a* query *point $x \in D$. For each random seed σ in the algorithm's probability space $(\Omega, \Pr)$, the filter obtains the value of f on a sequence of points $L(\sigma, f, x) = \{y_1, y_2, \ldots, y_k\}$, called* lookups*, (where the choice of y_i depends only on x, σ and $f(y_1), f(y_2), \ldots, f(y_{i-1})$) and outputs a reconstructed value $F(\sigma, f, x)$ for x solely based on the values of f at $L(\sigma, f, x)$. The reconstructed function $F_{\sigma,f} : D \to R$ given by $F_{\sigma,f}(x) = F(\sigma, f, x)$ must obey two conditions: (i) $F_{\sigma,f}$ satisfies property* P *for all functions f and all random seeds σ; (ii) if f satisfies property* P *then for all $x \in D$ we have $\Pr_\sigma(F_{\sigma,f}(x) \in [f(x)-a, f(x)+a]) \geq 1-\delta$.*

Notice that requirement (ii) in this definition is weaker than requiring that "if f satisfies property P then $\Pr_\sigma(\forall x \in D, F_{\sigma,f}(x) \in [f(x)-a, f(x)+a]) \geq 1-\delta$"; therefore, we manage to obtain lower bounds for a more general class of filters. As a notational remark, we usually omit the probability space and denote a local a-filter by (L, F).

The next observation captures the structural rigidity of local filters exploited in our lower bounds. It states that if functions f and g are identical on the lookups performed on query x when the input function is f, then the filter will perform the same lookups on x for both f and g and, consequently, reconstruct the same value.

Observation 2.1 *Let (L, F) be a local a-filter. Then the following holds for every random seed σ and query point x: if f and g are functions such that $f|_{L(\sigma,f,x)} = g|_{L(\sigma,f,x)}$, then $F(\sigma, f, x) = F(\sigma, g, x)$.*

Now we restate Theorems 1.1 and 1.2, giving more details about parameters we obtain.

Theorem 2.1. *Fix a non-negative constant δ, consider a sufficiently large integer d (depending on δ) and let $a \in [0, d/402]$. Let (L, F) be a local a-filter for the Lipschitz property with error probability δ. Then there exists a function $f : \{0,1\}^d \to \mathbb{R}$ and a query $x \in \{0,1\}^d$ such that $\Pr_\sigma(|L(\sigma, f, x)| \geq 2^{0.009d}) \geq 1/2 - 1.1\delta$.*

Theorem 2.2. *Fix a non-negative constant δ, consider a sufficiently large integer d (depending on δ) and let $a \geq 0$. Let (L, F) be a local a-filter for monotonicity with error probability δ. Then there exists a function $f : \{0,1\}^d \to [0, 2a+1]$ and a query $x \in \{0,1\}^d$ such that $\Pr_\sigma(|L(\sigma, f, x)| \geq 2^{0.009d}) \geq 1/2 - 1.1\delta$.*

The proof of Theorem 2.1 (resp. Theorem 2.2) follows directly from Lemma 4.3 (resp. Lemma 5.3) and Theorem 6.1; details are given in the full version [3].

3 c-Connectors

In this section, we formally introduce the notion of c-connectors. This combinatorial structure can be represented as a directed graph on the vertex set $\{0,1\}^d$, where pairs of nodes need to share an out-neighbor with some prescribed properties. As we shall see next, c-connectors are related to 2-TC-spanners, although the full motivation for the exact definition will only become clear in Sections 4 and 5.

Definition 3.1. *Let X denote the set of points in $\{0,1\}^d$ with Hamming weight exactly $d/3$ and let Y denote the set of points in $\{0,1\}^d$ with Hamming weight exactly $2d/3$. Also let $\mathcal{P}$ denote the set of comparable pairs $(x,y) \in X \times Y$, namely, such that $x \prec y$.*

Definition 3.2 (c-connector). *Fix $c \in \mathbb{N}$. Given a subset $\mathcal{P}'$ of $\mathcal{P}$, a digraph G with the node set $\{0,1\}^d$ is a* c-connector *for $\mathcal{P}'$ if for every $(x,y) \in \mathcal{P}'$ there exists $z \in \{0,1\}^d$ with the following properties:*

- *(Connectivity) The arcs (x,z) and (y,z) belong to G.*
- *(Structure) $|z \setminus y| < c$ and $|z| > \frac{d}{3} - c$.*

A *2-TC-spanner* of the boolean hypercube (with the usual partial order) is a directed graph H on the node set $\{0,1\}^d$ with the property that for all $x \prec y$ there is a point z satisfying $x \preceq z \preceq y$, such that the arcs (x,z) and (z,y) belong to H [6]. If we reorient the arcs in a 2-TC-spanner of the hypercube, so that the nodes in Y only have outgoing arcs, we obtain a valid c-connector for every $c \geq 1$: this is because the requirement $x \preceq z \preceq y$ (in the definition of 2-TC-spanner) implies the structure requirement in a c-connector. Therefore, c-connectors relax 2-TC-spanners in two ways: first it requires that only pairs in $\mathcal{P}$ have a common neighbor with prescribed properties, and second it relaxes the required properties of this common neighbor. We remark that the direction of the arcs in c-connectors is important here, since in order to obtain the desired results we lower bound the outdegree. In contrast, in previous work [5,11] the information of whether point x was looked up on query y or vice versa was lost in the transformation to the corresponding 2-TC-spanner and the lower bound on the number of arcs, not the outdegree, was used. This is one of the changes that gives us stronger lower bounds.

4 Local Filters for the Lipschitz Property imply c-Connectors

In this section we focus on the Lipschitz property. We construct a family of functions such that a local a-filter that works correctly on functions from the family must preform lookups corresponding to a c-connector. The idea is to start with a Lipschitz function f^0 and then construct other Lipschitz functions f^c_y which agree with f^0 on most points, but where $f^c_y(y)$ is much larger than $f^0(y)$. We argue that if a purported local a-filter makes only 'local' lookups when reconstructing at queries x and y, then we can create a function that looks like f^c_y around y (so that the filter is fooled and returns $F(y)$ in the range $f^c_y(y) \pm a$) and looks like f^0 around x (so that the filter is fooled and returns $F(x)$ in the range $f^0(x) \pm a \ll f^c_y(y) \pm a$). Thus, for the returned function, $F(x)$ and $F(y)$ are too far apart, ensuring that it is not Lipschitz.

4.1 Hard functions for Filter

Recall from Definition 3.1 that Y denotes the set of points in $\{0,1\}^d$ with Hamming weight exactly $d/3$. In order to construct these hard functions, for a point $y \in Y$ let $T_y = \{x \in \{0,1\}^d : x \subseteq y, |x| \geq d/3\}$. Define the function f^0 by $f^0(z) = \max\{|z|, d/3\}$ for all $z \in \{0,1\}^d$. Intuitively, for $c \in \mathbb{N}$ and $y \in Y$, we define the function f_y^c as the smallest Lipschitz function which is at least $f^0 + c\chi_{T_y}$, where χ_{T_y} denotes the characteristic function of the set T_y. More specifically, we set $f_y^c(z) = \max\{|z| + c - |z \setminus y|, f^0(z)\}$ for all $z \in \{0,1\}^d$.

Clearly f^0 is Lipschitz, and the functions f_y^c can be shown to be Lipschitz as well.

Lemma 4.1. *For all $c \in \mathbb{N}$ and $y \in Y$ the function f_y^c is Lipschitz.*

For a point $y \in Y$ and a constant $c \in \mathbb{N}$, let $T_y^c \subseteq \{0,1\}^d$ be the set of points z, such that $f_y^c(z) \neq f^0(z)$. Then $T_y^1 = T_y$ and the set T_y^c gets larger as c increases: specifically, $T_y^c \subseteq T_y^{c'}$ for $c < c'$. The definitions of f_y^c and f^0 directly give the following observation, which justifies the specific structure used in the definition of a c-connector.

Observation 4.1 *All elements z in the set T_y^c satisfy $|z \setminus y| < c$ and $|z| > \frac{d}{3} - c$.*

4.2 Correct Reconstruction of Hard Functions implies c-Connector

Now we show that if a local a-filter is correct on the constructed functions, its lookups correspond to a c-connector for the interesting pairs $\mathcal{P}$ (recall that $\mathcal{P}$ is the set of pairs $(x,y) \in X \times Y$ such that $x \prec y$). We start by essentially focusing on deterministic filters or, alternatively, by looking at a 'good' seed of a randomized filter. The analysis for randomized filters is based on the ability to pick a few of these good seeds and then analyzing the 'union' of the behavior of the filter running with these seeds.

Consider a local a-filter (L, F). Given points $x \in X$ and $y \in Y$, we say that a random seed $\sigma \in \Omega$ is *good* for x and y if $F_{\sigma,f^0}(x) \in [f^0(x) - a, f^0(x) + a]$ and $F_{\sigma,f_y^c}(y) \in [f_y^c(y) - a, f_y^c(y) + a]$. Given a seed σ which is good for x and y, we define the digraph $G_\sigma^{xy} = (\{0,1\}^d, A_\sigma^{xy})$ that captures the lookups made on queries x and y. Specifically, the set A_σ^{xy} consists of all the arcs $\{(x,z) : z \in L(\sigma, f^0, x) \cup \{x\}\}$ and $\{(y,z) : z \in L(\sigma, f_y^c, y) \cup \{y\}\}$.

Lemma 4.2 (Local filter implies c-connector). *Consider a local a-filter (L, F) for the Lipschitz property and an integer $c > 2a$. For all $(x,y) \in \mathcal{P}$, if $\sigma \in \Omega$ is good for x and y then G_σ^{xy} is a c-connector for (x,y).*

Proof. For the sake of contradiction suppose not. Unraveling the definitions and using Observation 4.1 this means that the sets $(L(\sigma, f^0, x) \cup \{x\}) \cap T_y^c$ and $(L(\sigma, f_y^c, y) \cup \{y\}) \cap T_y^c$ do not intersect. Then let A, B be a partition of T_y^c such that A contains $(L(\sigma, f^0, x) \cup \{x\}) \cap T_y^c$ and B contains $(L(\sigma, f_y^c, y) \cup \{y\}) \cap T_y^c$. Define the function f such that $f|_A = f^0|_A$, $f|_B = f_y^c|_B$, and $f|_{\{0,1\}^d \setminus (A \cup B)} = f^0|_{\{0,1\}^d \setminus (A \cup B)} = f_y^c|_{\{0,1\}^d \setminus (A \cup B)}$ (the last equation follows from the definition of T_y^c). To reach a contradiction, we show that the filter does not reconstruct f correctly.

Notice that $f^0|_{L(\sigma,f^0,x)} = f|_{L(\sigma,f^0,x)}$, so Observation 2.1 gives that $F(\sigma,f,x) = F(\sigma,f^0,x)$. Similarly, $f^c_y|_{L(\sigma,f^c_y,y)} = f|_{L(\sigma,f^c_y,y)}$ and hence $F(\sigma,f,y) = F(\sigma,f^c_y,y)$.

Now since σ is good for x and y, we have that $F(\sigma,f,x) = F(\sigma,f^0,x) \leq f^0(x) + a = \frac{d}{3} + a$ and $F(\sigma,f,y) = F(\sigma,f^c_y,y) \geq f^c_y(y) - a = \frac{2d}{3} + c - a$. Since $c > 2a$ we get $F(\sigma,f,y) - F(\sigma,f,x) > d/3 = \|x-y\|_1$, and hence the function $F_{\sigma,f}$ is not Lipschitz; this contradicts that (L,F) is a local a-filter and concludes the proof. □

Consider two subsets $\mathcal{P}_1, \mathcal{P}_2$ of $\mathcal{P}$. Notice that if G_1 is a c-connector for $\mathcal{P}_1$ and G_2 is a c-connector for $\mathcal{P}_2$ then the graph formed by the union of (the arcs of) G_1 and G_2 is a c-connector for $\mathcal{P}_1 \cup \mathcal{P}_2$. We remark that when we take this union we do not add parallel arcs. This directly gives the following result.

Corollary 4.1. *Consider a local a-filter (L,F) for the Lipschitz property and an integer $c > 2a$. Suppose that for each $(x,y) \in \mathcal{P}$ there is a random seed $\sigma(x,y) \in \Omega$ which is good for x and y. Then the graph obtained as the union of the graphs $\{G^{xy}_{\sigma(x,y)}\}_{(x,y)\in\mathcal{P}}$ is a c-connector for $\mathcal{P}$. Moreover, this graph has outdegree at most*

$$\max\left\{\max_{x\in X}\left\{\left|\bigcup_y L(\sigma(x,y),f^0,x)\right|\right\}, \max_{y\in Y}\left\{\left|\bigcup_x L(\sigma(x,y),f^c_y,y)\right|\right\}\right\} + 1. \quad (1)$$

Using this corollary, we show that a local a-filter with small 'average' number of lookups implies a c-connector for $\mathcal{P}$ with a small outdegree. The idea is to construct, via the probabilistic method, a set $\bar{S} \subseteq \Omega$ of good seeds which attains a small value in (1); details are provided in the full version [3].

Lemma 4.3. *Consider a local a-filter (L,F) for the Lipschitz property with error probability δ and an integer $c > 2a$. Consider $\alpha > 0$ and let $M = \max_{f,x} \Pr_\sigma(|L(\sigma,f,x)| > \alpha)$. If $\delta + M < 1/2$ then there is a c-connector for $\mathcal{P}$ with maximum outdegree at most $2d\alpha / \log\left(\frac{1}{2\delta+2M}\right) + 1$.*

5 Local Filters for Monotonicity imply 1-Connectors

In this section, we consider the monotonicity property and show that again the lookups performed by a local a-filter give rise to a c-connector (in this case, with $c = 1$).

5.1 Hard Functions for Filter

Again, we start by defining functions f^0 and f^a_y, such that if a local filter is correct on these functions, its lookups correspond to a 1-connector. Recall that for a point $y \in Y$, we define $T_y = \{x \in \{0,1\}^d : x \subseteq y, |x| \geq d/3\}$. Define the function f^0 by $f^0(z) = 2a+1$ if $|z| \geq d/3$ and $f^0(z) = 0$ if $|z| < d/3$. For a point $y \in Y$, we define the function f^a_y equal to $f^0 - (2a+1)\chi_{T_y}$, namely, $f^a_y(z) = 2a+1$ if $z \geq d/3$ and $z \notin T_y$ and $f^a_y(z) = 0$ otherwise. It can be easily verified that these functions are monotone.

Lemma 5.1. *For all $y \in Y$ and $a \geq 0$, the functions f^0 and f^a_y are monotone.*

Notice that the functions f^0 and f^a_y differ exactly on points in T_y and that T_y is the set of points which satisfy the structure property in the definition of a 1-connector.

5.2 Correct Reconstruction of Hard Functions implies 1-Connector

Recall that $\mathcal{P}$ is the set of comparable pairs $(x, y) \in X \times Y$ or, equivalently, pairs where $x \in T_y$. Consider a local a-filter (L, F) for monotone functions. As before, given $x \in X$ and $y \in Y$, we say that a random seed $\sigma \in \Omega$ is *good* for x and y if $F_{\sigma,f^0}(x) \in [f^0(x) - a, f^0(x) + a]$ and $F_{\sigma,f^a_y}(y) \in [f^a_y(y) - a, f^a_y(y) + a]$. Given a seed σ which is good for x and y, we define the digraph $G^{xy}_\sigma = (\{0,1\}^d, A^{xy}_\sigma)$ in a way similar to what we did in the previous section: we add to A^{xy}_σ all the arcs $\{(x, z) : z \in L(\sigma, f^0, x) \cup \{x\}\}$ and $\{(y, z) : z \in L(\sigma, f^a_y, y) \cup \{y\}\}$.

Again the construction of our functions and the digraph G^{xy}_σ together with the behavior of local a-filters captured in Observation 2.1 give the following.

Lemma 5.2. *Take $a \geq 0$ and consider a local a-filter (L, F) for monotonicity. For any $(x, y) \in P$, if $\sigma \in \Omega$ is good for x and y then G^{xy}_σ is a 1-connector for (x, y).*

Finally, we can utilize the same technique for finding a set of good seeds which achieve small value in (1) as done in Lemma 4.3 to obtain the desired connection between local a-filters and 1-connectors for $\mathcal{P}$.

Lemma 5.3. *Take $a \geq 0$ and consider a local a-filter (L, F) for monotone functions with error probability δ. Consider $\alpha > 0$ and let $M = \max_{f,x} \Pr_\sigma \left(|L(\sigma, f, x)| > \alpha\right)$. If $\delta + M < 1/2$ then there is a 1-connector for $\mathcal{P}$ with maximum outdegree at most $2d\alpha / \log\left(\frac{1}{2\delta+2M}\right) + 1$.*

6 Lower Bound on the Maximum Outdegree of a c-Connector

Recall that $\mathcal{P}$ is the set of pairs $(x, y) \in X \times Y$ such that x and y are *comparable*. We show a lower bound on the maximum outdegree of a c-connector for $\mathcal{P}$. We remark that the constants in the bound are not optimized.

Theorem 6.1. *Consider $d \geq 40,200$ and let c be an integer in the range $[d/201, d/200]$. Then the maximum outdegree of any c-connector for $\mathcal{P}$ is at least $2^{0.01d}$.*

To prove this, let G be a c-connector for $\mathcal{P}$. Let $\tilde{T}^c_y = \{z : |z \setminus y| < c, |z| > d/3 - c\}$ be the points which satisfy the structure property in Definition 3.2. Then $T_y \subseteq T^c_y \subseteq \tilde{T}^c_y$ for all $y \in Y$, and for $x \in T_y$ and $z \in \tilde{T}^c_y$ we have $x \cup z \in \tilde{T}^c_y$. We say that a pair $(x, y) \in \mathcal{P}$ is *covered* by a point z if $z \in \tilde{T}^c_y$ and the arcs (x, z) and (y, z) belong to G.

Each pair in $\mathcal{P}$ needs to be covered by a point. For a fixed $x \in X$, the outdegree of x in G is at least the number of distinct points which are covering the pairs in $\mathcal{P}$ containing x (and similarly for a fixed $y \in Y$). The difficulty in lower bounding the outdegree of x is that many pairs containing it can be covered by the same point. The heart of the argument is to show that no point can cover too many such pairs. It relies on the fact that the sets $\tilde{T}^c_y$ are 'localized'. More precisely, consider a point z and let (x, y) be covered by it. Notice that $x \in T_y$ and $z \in \tilde{T}^c_y$, hence $x \cup z \in \tilde{T}^c_y$. If z is not near x, namely, $z \setminus x$ is large, then we argue that not too many points y satisfy $x \cup z \in \tilde{T}^c_y$, given the localization

of $\tilde{T}_y^c$. On the other hand, if z is near x then there are not too many possibilities for x itself. Our bound is derived by putting these observations together.

In order to make the above argument work we divide the pairs in $\mathcal{P}$ into two groups based on the covers they have. Let $\alpha \in [1/15, 1/14]$ be such that αd is an integer, which exists since d is sufficiently large. For $(x, y) \in \mathcal{P}$ and z that covers (x, y), if $|z \setminus x| \leq \alpha d$, then we say that z is *near* x and that z is a *nearby cover* of (x, y). Let $\mathcal{N}$ denote the set of pairs $(x, y) \in \mathcal{P}$ which have a nearby cover and let $\mathcal{F} = \mathcal{P} \setminus \mathcal{N}$ be the remaining pairs. For a fixed $y \in Y$, define $\mathcal{N}_y$ as the pairs in $\mathcal{N}$ containing y and for $x \in X$ define $\mathcal{F}_x$ as the pairs in $\mathcal{F}$ containing x.

Let $Z \subseteq \{0,1\}^d$ be the set of points which cover at least one pair in $\mathcal{P}$. For a given $x \in X$, we use Z_x to denote the set of points which cover at least one pair in $\mathcal{P}$ containing x. We define Z_y analogously. That is, Z is the union of sets Z_x and Z_y over all $x \in X$ and $y \in Y$.

Now we sketch the argument that upper bounds the number of pairs in $\mathcal{N}$ and $\mathcal{F}$; computations are presented in the full version [3]. In order to upper bound $\mathcal{N}$ we start by arguing that, for a fixed $y \in Y$, a point cannot be a nearby cover for many pairs (x, y) in $\mathcal{N}_y$. To see this, take $z \in Z_y$ and let $(x, y) \in \mathcal{P}$ be such that z is a nearby cover for it. Then notice that x and z are very similar: $|z \setminus x| \leq \alpha d$ and $|x \setminus z| \leq \alpha d + c$; the first bound follows from the definition of a nearby cover and the second uses $|z| \geq |x| - c$ from Observation 4.1. From these constraints, it follows that there are at most $d^2\binom{d/3+\alpha d}{\alpha d}\binom{2d/3+c}{\alpha d+c}$ possibilities for such x's. Thus, for all $y \in Y$ we have $|\mathcal{N}_y| \leq |Z_y| \cdot d^2\binom{d/3+\alpha d}{\alpha d}\binom{2d/3+c}{\alpha d+c}$. Adding over all y gives the desired bound.

Lemma 6.1. *Letting* $\Theta = d^2\binom{d/3+\alpha d}{\alpha d}\binom{2d/3+c}{\alpha d+c}$, *the number of pairs in* $\mathcal{N}$ *is at most* $|Y| \cdot \Theta \cdot \max_{y \in Y}\{|Z_y|\}$.

To upper bound the size of $\mathcal{F}$ we start by showing that, for a fixed $x \in X$, a point cannot be a (non-nearby) cover for too many pairs in $\mathcal{F}_x$. To see this, take $z \in Z_x$ and suppose $(x, y) \in \mathcal{F}_x$ is covered by z. Notice that $x \cup z$ and y are very similar: $|(x \cup z) \setminus y| \leq c$ and $|y \setminus (x \cup z)| \leq d/3 - \alpha d + c$; the first bound follows from $x \subseteq y$ and Observation 4.1, and the second further uses the fact that $|x \cup z| \geq d/3 + \alpha d$ (since z is not a nearby cover). Then it is easy to see that there are at most $d^2\binom{\frac{2d}{3}+c}{c}\binom{\frac{2d}{3}-\alpha d}{\frac{d}{3}-\alpha d+c}$ such y's. Thus, for each $x \in X$ we have $|\mathcal{F}_x| \leq |Z_x| \cdot d^2\binom{\frac{2d}{3}+c}{c}\binom{\frac{2d}{3}-\alpha d}{\frac{d}{3}-\alpha d+c}$ and adding over all x gives the desired bound on $\mathcal{F}$.

Lemma 6.2. *Letting* $\Phi = d^2\binom{\frac{2d}{3}+c}{c}\binom{\frac{2d}{3}-\alpha d}{\frac{d}{3}-\alpha d+c}$, *the number of pairs in* $\mathcal{F}$ *is at most* $|X| \cdot \Phi \cdot \max_{x \in X}\{|Z_x|\}$.

The maximum outdegree of the c-connector G is bounded from below by

$$M \triangleq \max\{\max_{x \in X}\{|Z_x|\}, \max_{y \in Y}\{|Z_y|\}\}.$$

Since $\mathcal{N}$ and $\mathcal{F}$ partition the set of pairs $\mathcal{P}$, we can add the bounds from Lemmas 6.1 and 6.2 and obtain that M is at least the size of $\mathcal{P}$ divided by $\binom{d}{d/3}(\Theta + \Phi)$, which gives $M \geq \binom{2d/3}{d/3}/(\Theta + \Phi)$. Standard computations can be used to lower bound the right-hand side of this expression by $2^{0.01d}$. This concludes the proof of Theorem 6.1.

7 Conclusion and Future work

We show that local filters for the Lipschitz property and monotonicity require exponentially many (in the dimension) lookups, even when allowed additive error. One can try to further relax the requirements on local filters in order to overcome these lower bounds.

One possibility is to consider local filters whose output does not satisfy the desired property P with small probability. Such weaker guarantees can still be useful for other definitions of privacy [4,8]. Another relaxation, specific to the Lipschitz property, is to allow the reconstructed function F to be b-Lipschitz, that is, to require only $|F(x) - F(y)| \leq b \cdot \|x - y\|_1$ for all $x, y \in \{0,1\}^d$. For the privacy application described, having a and b of order $O(\sqrt{d})$ is still acceptable. We remark that the techniques presented here yield similar lower bounds for b slightly larger than 1, but not for $b \geq 2$.

References

1. Ailon, N., Chazelle, B., Comandur, S., Liu, D.: Property-preserving data reconstruction. Algorithmica 51(2), 160–182 (2008)
2. Alon, N., Rubinfeld, R., Vardi, S., Xie, N.: Space-efficient local computation algorithms. In: Rabani, Y. (ed.) SODA. pp. 1132–1139. SIAM (2012)
3. Awasthi, P., Jha, M., Molinaro, M., Raskhodnikova, S.: Limitations of local filters of lipschitz and monotone functions. Electronic Colloquium on Computational Complexity (ECCC) TR12-075 (2012)
4. Bhaskar, R., Bhowmick, A., Goyal, V., Laxman, S., Thakurta, A.: Noiseless database privacy. In: ASIACRYPT. pp. 215–232 (2011)
5. Bhattacharyya, A., Grigorescu, E., Jha, M., Jung, K., Raskhodnikova, S., Woodruff, D.P.: Lower bounds for local monotonicity reconstruction from transitive-closure spanners. SIAM J. Discrete Math. 26(2), 618–646 (2012)
6. Bhattacharyya, A., Grigorescu, E., Jung, K., Raskhodnikova, S., Woodruff, D.P.: Transitive-closure spanners. In: SODA. pp. 932–941 (2009)
7. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. J. Comput. Syst. Sci. 47(3), 549–595 (1993)
8. Dwork, C., Kenthapadi, K., Mcsherry, F., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: In EUROCRYPT. pp. 486–503. Springer (2006)
9. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: TCC. pp. 265–284 (2006)
10. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. J. ACM 45(4), 653–750 (1998)
11. Jha, M., Raskhodnikova, S.: Testing and reconstruction of Lipschitz functions with applications to data privacy. In: IEEE FOCS. pp. 433–442 (2011), full version available at http://eccc.hpi-web.de/report/2011/057/.
12. Katz, J., Trevisan, L.: On the efficiency of local decoding procedures for error-correcting codes. In: STOC. pp. 80–86 (2000)
13. Raskhodnikova, S.: Transitive-closure spanners: A survey. In: Goldreich, O. (ed.) Property Testing. Lecture Notes in Computer Science, vol. 6390, pp. 167–196. Springer (2010)
14. Rubinfeld, R., Sudan, M.: Robust characterization of polynomials with applications to program testing. SIAM J. Comput. 25(2), 252–271 (1996)
15. Rubinfeld, R., Tamir, G., Vardi, S., Xie, N.: Fast local computation algorithms. In: ICS. pp. 223–238 (2011)
16. Saks, M.E., Seshadhri, C.: Local monotonicity reconstruction. SIAM J. Comput. 39(7), 2897–2926 (2010)