2016 IEEE 57th Annual Symposium on Foundations of Computer Science

Lipschitz Extensions for Node-Private Graph Statistics and the Generalized Exponential Mechanism

Sofya Raskhodnikova, Adam Smith
Computer Science and Engineering Department
Pennsylvania State University
University Park, PA USA
{sofya,asmith}@cse.psu.edu

***Abstract*— Lipschitz extensions were proposed as a tool for designing differentially private algorithms for approximating graph statistics. However, efficiently computable Lipschitz extensions were known only for 1-dimensional functions (that is, functions that output a single real value). We study efficiently computable Lipschitz extensions for multi-dimensional (that is, vector-valued) functions on graphs. We show that, unlike for 1-dimensional functions, Lipschitz extensions of higher-dimensional functions on graphs do not always exist, even with a non-unit stretch. We design Lipschitz extensions with small stretch for the sorted degree list and degree distribution of a graph, viewed as functions from the space of graphs equipped with the node distance into real space equipped with ℓ_1. Our extensions are from the space of bounded-degree graphs to the space of arbitrary graphs. The extensions use convex programming and are efficiently computable.**

We also develop a new tool for employing Lipschitz extensions in differentially private algorithms that operate with no prior knowledge of the graph (and, in particular, no knowledge of the degree bound). Specifically, we generalize the exponential mechanism, a widely used tool in data privacy. The exponential mechanism is given a collection of score functions that map datasets to real values. It returns the name of the function with nearly minimum value on the dataset. Our generalized exponential mechanism provides better accuracy than the standard exponential mechanism when the sensitivity of an optimal score function is much smaller than the maximum sensitivity over all score functions.

We use our Lipschitz extensions and the generalized exponential mechanism to design a node differentially private algorithm for approximating the degree distribution of a sensitive graph. Our algorithm is much more accurate than those from previous work. In particular, our algorithm is accurate on all graphs whose degree distributions decay at least as fast as those of "scale-free" graphs. Using our methodology, we also obtain more accurate node-private algorithms for 1-dimensional statistics.

The authors were supported by NSF awards CDI-0941553 and IIS-1447700, a Google Faculty Award, Boston University's Hariri Institute for Computing and RISCS Center and, while visiting the Harvard Center for Research on Computation and Society, by a Simons Investigator grant to Salil Vadhan.

I. Introduction

The area of *differential privacy* studies how to output aggregate information about a database while protecting privacy of individuals whose information it contains. Many datasets can be represented as graphs, where nodes correspond to individuals and edges capture relationships between them. There are two natural variants of differential privacy that are suitable for graphs: *edge* privacy and *node* privacy. Intuitively, the former protects relationships among individuals, while the latter protects each individual, together with all his/her relationships. Edge privacy has been extensively studied, with algorithms now known for the release of subgraph counts and related scalar-valued functions [32, 33, 17, 30, 24, 18], the degree distribution [11, 12, 16, 23, 15], cut densities [10, 3] and the parameters of generative graph models [30, 18, 24, 15, 35]. Node privacy is a much stronger guarantee, but is significantly harder to attain because it guards against larger changes in the input. Until recently, no known node private algorithms gave accurate answers on sparse graphs, even for extremely simple statistics. (Typically, graphs that contain sensitive information, such as friendships, sexual relationships, and communication patterns, are sparse.)

In 2013, Blocki et al. [4], Kasiviswanathan et al. [19], and Chen and Zhou [6] proposed two methods for obtaining node private algorithms: (i) using projections whose smooth sensitivity could be bounded (combined with mechanisms from [32] that are based on smooth sensitivity), and (ii) computing *Lipschitz extensions* (and releasing them via the Laplace mechanism of [9]). The former method is generic, i.e., works for releasing any graph statistics, while the latter method requires designing an efficiently computable Lipschitz extension for each desired graph statistics. However, the latter method yielded much more accurate algorithms. In particular, [19, 6] used it to obtain accurate node-private algorithms for computing subgraph counts and

0272-5428/16 $31.00 © 2016 IEEE
DOI 10.1109/FOCS.2016.60

related statistics. Subsequently to the initial version of this paper, Lipschitz extensions were also used to design node-private algorithms for fitting a large family of statistical models (graphons and stochastic block models) to network data [5].

Despite the success of efficiently computable Lipschitz extensions, they were known only for 1-dimensional functions (that is, functions that output a single real value). We study efficiently computable Lipschitz extensions for multi-dimensional (that is, vector-valued) functions on graphs. In the full version of this paper, we show that, unlike for 1-dimensional functions, Lipschitz extensions of higher-dimensional functions on graphs do not always exist, even with a non-unit stretch. We design Lipschitz extensions with small stretch for the sorted degree list and degree distribution of a graph, viewed as functions from the space of graphs equipped with the node distance into real space equipped with ℓ_1. Our extensions are from the space of bounded-degree graphs to the space of arbitrary graphs. The extensions can be computed in polynomial time.

We also develop a new tool for employing Lipschitz extensions in differentially private algorithms that operate with no a priori knowledge of the graph. The algorithms in [4, 19] need a publicly known bound on the degree of the input graph in order to choose an appropriate Lipschitz extension. Chen and Zhou [6] overcome this problem by showing how to choose the Lipschitz extension for the special case of their family of Lipschitz extensions for subgraph counts and related 1-dimensional functions while paying a factor of $O(\log n)$ for n-node graphs in the algorithm's additive error. We develop a general method for choosing a good Lipschitz extension with a smaller loss in accuracy (specifically, a factor of $O(\log \log n)$ in the additive error for subgraph counts). To achieve this, we generalize the exponential mechanism of McSherry and Talwar [29], a widely used tool in data privacy. Our generalized exponential mechanism provides better accuracy than the standard exponential mechanism when the sensitivity of an optimal score function is much smaller than the maximum sensitivity over all score functions.

We use our Lipschitz extension and the generalized exponential mechanism to design a node-private algorithm for releasing an approximation to the degree distribution of a graph. Our algorithm is much more accurate than those from previous work [19]. In particular, our algorithm is accurate on all graphs whose degree distributions decay at least as fast as those of "scale-free" graphs.

A. Lipschitz Extensions

Lipschitz extensions are basic mathematical objects studied in functional analysis.

Definition I.1 (Lipschitz constant)**.** *Let $f : X \to Y$ be a function from a domain X to a range Y with associated distance measures d_X and d_Y. Function f has* Lipschitz constant *c (equivalently, is c-Lipschitz) if $d_Y(f(x), f(x')) \leq c \cdot d_X(x, x')$ for all $x, x' \in X$. The smallest Lipschitz constant of f is sometimes referred to as the* (global) sensitivity *of f.*

Definition I.2 (Lipschitz extension)**.** *Consider a domain X and a range Y with associated distance measures d_X and d_Y, and let $Z \subset X$. Fix constants $c > 0$ and $s \geq 1$. Given a c-Lipschitz function $f : Z \to Y$, a function $\hat{f} : X \to Y$ is a* Lipschitz extension of f from Z to X with stretch s *if*

1) *$\hat{f}$ is an* extension of f*, that is, $\hat{f}(x) = f(x)$ on all $x \in Z$ and*
2) *$\hat{f}$ is $s \cdot c$-Lipschitz.*

If $s = 1$, then we call $\hat{f}$ a Lipschitz extension of f from Z to X *(omitting the stretch).*

Functional analysts have devoted considerable attention to determining, for given metric spaces X, Z and Y, whether Lipschitz extensions from Z to X with stretch 1 exist for all functions $f : Z \to Y$. In contrast to this work, the focus in functional analysis is mostly on continuous function spaces.

We study Lipschitz extensions of multi-dimensional functions on graphs. Let $\mathcal{G}$ denote the set of all finite, labeled, and unweighted undirected graphs. Given $D \in \mathbb{N}$, let $\mathcal{G}^D$ be the set of all *D-bounded* graphs in $\mathcal{G}$, that is, graphs of maximum degree at most D. Two graphs are called (node) *neighbors* if one can be obtained from the other by removing a node and its adjacent edges. This notion of neighbors induces a (node) distance measure d_{node} on $\mathcal{G}$. (Analogously, we define the edge distance measure by allowing two graphs be neighbors if they differ in exactly one edge.) We consider functions from $\mathcal{G}$, equipped with d_{node}, to $\mathbb{R}^p$, equipped with ℓ_1. We refer to p as the dimension of the function.

Lipschitz extensions of real-valued 1-dimensional functions (with arbitrary domain, not just $\mathcal{G}$) with stretch 1 always exist [28]. We show that it is not true, in general, for multi-dimensional functions on graphs, even with non-unit stretch. Our first technical contribution is the construction of efficiently computable, small-stretch extensions from $\mathcal{G}^D$ to $\mathcal{G}$ (equipped with d_{node}) of two related high-dimensional functions: the sorted degree list and the degree distribution.

Lipschitz Extensions as a Privacy Tool: Functions with low Lipschitz constant can be approximated very accurately by differentially private algorithms. The *Laplace mechanism* [9] can release a differentially private approximation of $f(G)$ for all functions $f : \mathcal{G} \to \mathbb{R}^p$ by adding noise proportional to the Lipschitz constant of f to each coordinate of $f(G)$ and publishing the result. The difficulty with employing the Laplace mechanism directly on graph data is that many useful functions on graphs are highly sensitive to the insertion or removal of a well-connected vertex. For example, the number of triangles (i.e., 3-cliques) in an n-node graph may go up by $\binom{n}{2}$ with the insertion of a single vertex. The degree distribution of a graph can also change drastically, shifting up by 1 in every coordinate (as one vertex can increase the degree of all other vertices). Therefore, these functions have high sensitivity. Mechanisms based on local notions of sensitivity (such as smooth sensitivity in [32]) also don't apply directly, since these functions are also sensitive in a local sense (roughly, interesting graphs such as those with low average degree are "near" other graphs with a vastly different value of the function).

One can get around the sensitivity issue by first considering a "nice" subset of the space $\mathcal{G}$ where the function f has low sensitivity [4, 19, 6]. Many functions of interest have low Lipschitz constant on $\mathcal{G}^D$, the set of bounded-degree graphs. The number of triangles in a graph, for instance, changes by at most $\binom{D}{2}$ among node-neighboring graphs of degree at most D, and the degree list changes by at most $2D$ in ℓ_1. Given a function f with low Lipschitz constant on "nice" graphs, if we find an efficiently computable Lipschitz extension $\hat{f}$ of f to all of $\mathcal{G}$, then we can use the Laplace mechanism to release $\hat{f}(G)$ with relatively small additive noise. The lower the stretch of the extension, the lower the noise. The result is accurate when the input indeed falls into, or near, the class of "nice" graphs. Interestingly, the class of "nice" graphs need not contain the input for the answer to be accurate—in our main application, we use $\mathcal{G}^D$ as the set of "nice" graphs, but D is much lower than the maximum degree of the input (the reduction in noise compensates for the distortion introduced by the extension.)

Existence and Computational Complexity of Lipschitz Extensions: Motivated by this methodology, we ask: when do Lipschitz extensions exist, and when do they admit efficient algorithms? The existence question has drawn interest from functional analysis and combinatorics for nearly a century [28, 20, 34, 26, 14, 27, 1, 2, 21, 31, 22, 25]; see Lee and Naor [22] for an overview. For any real-valued function $f : \mathcal{G}^D \to \mathbb{R}$, there exists an extension $\hat{f} : \mathcal{G} \to \mathbb{R}$ whose node sensitivity is the same as that of f [28]. Kasiviswanathan et al. [19] and Chen and Zhou [6] constructed *polynomial-time* computable Lipschitz extensions from $\mathcal{G}^D$ to $\mathcal{G}$ of subgraph counts (such as the number of triangles) and related 1-dimensional functions on graphs.

Prior constructions of higher-dimensional Lipschitz extensions focused on extending functions on a metric space X, where X is given explicitly as input (say, as a distance matrix) [22, 25]. Such constructions, at best, run in time polynomial in the size of X. The size of $\mathcal{G}^D$ is infinite, and even restricting to graphs on at most n vertices leaves a set $\mathcal{G}_n^D$ that is exponentially large in n. Moreover, generic constructions have stretch at least $\sqrt{n}$ (since the doubling dimension of $\mathcal{G}_n^D$ is large).

B. Our Contributions

Lipschitz Extension of the Degree List: Our main technical contribution is a polynomial-time, constant-stretch Lipschitz extension of the *sorted degree list*, viewed as a function from $\mathcal{G}^D$ to ℓ_1^*, to all of $\mathcal{G}$. Here ℓ_1^* denotes the ℓ_1 metric on the space of finite-length real sequences, where sequences of different length are padded with 0's to compute the distance.

Given any graph $G = (V, E)$, our function $\hat{f}_D(G)$ outputs a nonincreasing real sequence of length $|V|$. If the maximum degree of G is D or less, the output is the sorted list of degrees in G. The output can be thought of as a list of "fractional degrees", where "fractional edges" are real weights in $[0, 1]$ and the "fractional degree" of a vertex is the sum of the weights of its adjacent edges. The weights are selected by minimizing a quadratic function over the polytope of s-t flows in a directed graph closely related to G. Previous work [19] had shown that the *value* of the maximum flow in the graph has low sensitivity; by introducing the quadratic penalty, we give a way to select an optimal flow that changes slowly as the graph itself changes. Introducing a strongly convex penalty (or *regularizer*) to make the solution of an optimization problem stable to changes in the loss function is common in machine learning. In our setting, however, it is the *constraints* of the convex program that change with data, and not the loss function.

Theorem I.3. *There is a Lipschitz extension of the degree list, viewed as a function taking values in ℓ_1^*, from $\mathcal{G}^D$ to $\mathcal{G}$ with stretch 3/2 that can be computed in polynomial time.*

The sorted degree list has ℓ_1 sensitivity $2D$ on $\mathcal{G}^D$. The extension $\hat{f}_D(G)$ has ℓ_1 sensitivity at most $3D$ (the stretch is thus at most $3/2$).

We use our Lipschitz extension of the sorted degree list to get a Lipschitz extension of the degree distribution (a list of counts of nodes of each degree) and the degree CDF (a list of counts of nodes of at least each given degree). These functions condense the information to a D-dimensional vector (regardless of the size of the graph), making it easier to release with node-privacy.

Generalized Exponential Mechanism for Scores of Varying Sensitivity: One of the difficulties with using Lipschitz extensions in differentially private algorithms is selecting a good class of inputs from which to extend. For example, to apply our degree distribution extension, we need to select the degree bound D. More generally, we are given a collection of possible extensions $\hat{f}_1, ..., \hat{f}_k$, each of which agrees with f on a different set and has different sensitivity Δ_i.

We can abstract the task we are faced with as a private optimization problem: given a set of *real-valued* functions $q_1, ..., q_k$, the goal is to output the index $\hat{\imath}$ of a function with approximately minimal value on the dataset x (so that $q_{\hat{\imath}}(x) \approx \min_{i=1}^{k} q_i(x)$). (In our setting, the q_i functions are related to the error of the approximation $\hat{f}_i$ on dataset x). Suppose that each q_i has a known Lipschitz constant Δ_i. The *error* of an output $\hat{\imath}$ on input x is the difference $q_{\hat{\imath}}(x) - \min_{i=1}^{k} q_i(x)$.

The exponential mechanism [29] achieves error that scales with the *largest* Lipschitz constant. Specifically, for every $\beta > 0$, with probability $1 - \beta$, the output $\hat{\imath}$ satisfies $q_{\hat{\imath}}(x) \leq \min_{i=1}^{k} q_i(x) + \Delta_{max} \cdot \frac{2\ln(k/\beta)}{\epsilon}$ where $\Delta_{max} = \max_{i=1}^{k} \Delta_i$.

In contrast, we give an algorithm whose accuracy scales with the Lipschitz constant of the *optimal* score function Δ_{i^*} where $i^* = \operatorname{argmin}_{i=1}^{k} q_i(x)$. Our mechanism requires as input an upper bound $\beta > 0$ on the desired probability of a "bad" outcome; the algorithm's error guarantee depends on this β.

Theorem I.4 (Informal)**.** *For all settings of the input parameters $\beta \in (0, 1)$, $\epsilon > 0$, the Generalized Exponential Mechanism is ϵ-differentially private. For all inputs x, with probability at least $1 - \beta$, the output $\hat{\imath}$ satisfies*

$$q_{\hat{\imath}}(x) \leq \min_{i=1}^{k} \left(q_i(x) + \Delta_i \cdot \tfrac{4\ln(k/\beta)}{\epsilon} \right) .$$

When the maximum Δ_i is much larger than the sensitivity of the optimal score function, this guarantee is much better than that of the standard exponential mechanism. For instance, in our setting, the Δ_i's grow exponentially with i (that is, Δ_i is roughly 2^i for $i = 1, ..., k$). However, on sparse graphs, the best choice of Δ_i is for relatively small i. (Also, the issue is not merely with the error guarantee. The exponential mechanism provides bad outputs for many inputs where the true minimizer has low sensitivity.)

We can use our algorithm for selecting the sensitivity parameter for the Lipschitz extensions of graph functions in [4, 19, 6] and in this work. (These parameters are sometimes interpretable as a degree bound, as in the case of the degree distribution, but not always; for example, when computing the number of triangles, the parameter is a bound on the number of triangles involving any one vertex). This allows the algorithm to adapt to the specific input. The guarantee we get is that the error of the overall algorithm on an n-node graph is at most $O(\log \log n)$ times higher than one would get with the best Lipschitz constant. In contrast, the parameter selection method of Chen and Zhou [6] provides only a $O(\log n)$ guarantee on the error blow-up and is specific to the extensions they construct.

Differentially Private Algorithms for Releasing the Degree Distribution: We can combine the Lipschitz extension of the degree list and the parameter selection algorithm to get a differentially private mechanism for releasing the degree distribution of a graph that automatically adapts to the structure of the graph.

We show that our algorithm provides an accurate estimate on a large class of graphs, including graphs with low average degree whose degree distribution has a heavy tail. We measure accuracy in the ℓ_1 norm, normalized by the number of nodes in the graph—that is, we deem the algorithm accurate if the total variation distance between the true degree distribution and the estimate is small.

This measure goes to 0 for graphs of low average degree in which the tail of the degree distribution decreases slightly more quickly than what trivially holds for all graphs. If $\bar{d}$ is the average degree in a graph, Markov's inequality implies that the fraction of nodes with degree above $t \cdot \bar{d}$ is at most $1/t$. We assume that this fraction goes down as $1/t^\alpha$ for a constant $\alpha > 1$. The condition is called α-*decay* [19]. Our assumption is satisfied by all the well-studied social network models we know of, including *scale-free* graphs [7] (which satisfy the assumption with $\alpha \in (1, 2)$). Previous work [19] provided nontrivial accuracy only for $\alpha > 2$ (and, in particular, gave no guarantees for scale-free graphs).

Our algorithm need not be given α or the average degree of the graph; these are implicitly taken into account by parameter selection. Incontrast, the algorithm in [19] requires a lower bound on α as input.

C. Contents of the Short and Full Versions

This short version of the document includes background information (Section II), proofs of our two

main technical results: the Lipschitz extension of the degree list (Section III) and the generalized exponential mechanism (Section IV) and the statement of the utility guarantees we obtain for our node-private algorithm for releasing the degree distribution.

We defer to the full version our lower bounds on the stretch of extensions from $\mathcal{G}^D$ to $\mathcal{G}$, our low-stretch extension of the degree distribution (obtained from our extension of the degree list) and details on how to combine the main technical results to design differentially private algorithms.

II. Definitions and Basic Tools

We use $[n]$ to denote the set $\{1, \ldots, n\}$. For a graph $G = (V, E)$, the average degree is denoted by $\bar{d}(G) = 2|E|/|V|$. For simplicity of presentation, we assume that $n = |V|$, the number of nodes of the input graph G, is publicly known. This assumption is justified since, as we will see, one can get an accurate estimate of $|V|$ by running a node-private algorithm. The degrees of nodes in V are denoted by $\deg_1(G), \ldots, \deg_n(G)$. When the graph referenced is clear, we drop G in the notation.

A. Graphs Metrics and Differential Privacy

Suppose that datasets are members of a universe U equipped with a neighbor relation. For example, if datasets are graphs, we could use the notion of node or edge neighbors; for standard datasets (with no relationship information), two datasets can be considered neighbors if they are at Hamming distance 1 or if their set difference has size 1.

Definition II.1 (ϵ-differential privacy [9])**.** *A randomized algorithm $\mathcal{A}$ is* ϵ-differentially private *(with respect to the neighbor relation on U) if for all events S in the output space of $\mathcal{A}$ and all neighbors $x, x' \in U$,*

$$\Pr[\mathcal{A}(x) \in S] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(x') \in S]\,.$$

If $U = \mathcal{G}$ with node (respectively, edge) neighbor relationship, we call a differentially private algorithm simply node-private *(respectively,* edge-private*).*

In this paper, if node or edge privacy is not specified, we mean node privacy by default.

B. Basic Tools

Laplace Mechanism: In the most basic framework for achieving differential privacy, Laplace noise is scaled according to the *sensitivity* of the desired statistic f, measured with respect to an appropriate metric on datasets. This technique extends to node private analysis of graphs as long as we measure sensitivity with respect to the node distance. Let $\mathcal{G}$ denote the set of all graphs. In the sequel, the *sensitivity* of $f : \mathcal{G} \to \mathbb{R}^p$ denotes the smallest Lipschitz constant (Definition I.1) of f, viewed as a map from $(\mathcal{G}, d_{\text{node}})$ to ℓ_1^p.

For example, the number of edges in graphs with at most n nodes has sensitivity $n - 1$, since adding or deleting a node and its adjacent edges can add or remove at most $n - 1$ edges. In contrast, the number of nodes has sensitivity 1.

A *Laplace* random variable with mean 0 and standard deviation $\sqrt{2}\lambda$ has density $h(z) = (1/(2\lambda))e^{-|z|/\lambda}$. We denote it by $\mathrm{Lap}(\lambda)$.

Theorem II.2 (Laplace Mechanism [9])**.** *If $f : \mathcal{G} \to \mathbb{R}^p$ is Δ-Lipschitz (i.e., has sensitivity at most Δ), the algorithm $\mathcal{A}(G) = f(G) + \mathrm{Lap}(\Delta/\epsilon)^p$ (which adds i.i.d. noise $\mathrm{Lap}(\Delta f/\epsilon)$ to each entry of $f(G)$) is ϵ-node private.*

Thus, we can release the number of nodes $|V|$ in a graph with noise of expected magnitude $1/\epsilon$ while satisfying node privacy. Given a public bound n on the number of nodes, we can release the number of edges $|E|$ with additive noise of expected magnitude $(n-1)/\epsilon$.

Exponential Mechanism: Suppose we are given a collection of functions $q_1, ..., q_k$, from the universe U to $\mathbb{R}$ such that the function q_i is Δ_i-Lipschitz for each $i \in [k]$. Recall that $\Delta_{max} = \max_{i \in [k]} \Delta_i$. The exponential mechanism [29], denoted by $\mathcal{EM}$, takes a dataset $x \in U$ and returns an index $\hat{\imath}$ for which $q_{\hat{\imath}}(x)$ has nearly minimal value at x, that is, such that $q_{\hat{\imath}}(x) \approx \min_{i \in [k]} q_i(x)$. The algorithm $\mathcal{EM}$ has the following parameters: $\epsilon > 0$, score functions $q_i : U \to \mathbb{R}$ for all $i \in [k]$, and Δ_{max}. On input x, the algorithm $\mathcal{EM}$ samples and returns an index i from $[k]$ with probability proportional to $\exp\left(\frac{\epsilon}{2\Delta_{max}} q_i(x)\right)$, normalized so that probabilities for all $i \in [k]$ sum to 1.

Lemma II.3 (Exponential Mechanism [29])**.** *The algorithm $\mathcal{EM}$ is ϵ-differentially private. Moreover, for every $\beta \in (0, 1)$, with probability at least $1 - \beta$, its output $\hat{\imath}$ satisfies $q_{\hat{\imath}}(x) \leq \min_i \left(q_{i \in [k]}(x)\right) + \frac{2\Delta_{max} \ln(k/\beta)}{\epsilon}$.*

There is a simple, efficient implementation of the exponential mechanism that adds exponential noise to each score function and reports the maximizer of the noisy scores (see, e.g., [8, Sec. 3.4]).

III. Lipschitz Extensions of the Degree List

In this section, we give a Lipschitz extension of the degree list. For an n-node graph G, denote the list of degrees of G sorted in nonincreasing order by

$$\text{deg-list}(G) = sort(deg_1(G), ..., deg_n(G)).$$

We view the degree list as an element of $\mathbb{R}^*$ (the set of finite sequences of real numbers). We equip the space with the ℓ_1 distance, where the sequences of different lengths are padded with 0's to allow comparison. This representation is convenient for handling node additions and deletions.

The ℓ_1 sensitivity (under node distance) of the degree list on D-bounded graphs is $2D$ because the unsorted degree list has sensitivity $2D$ and, as Hay et al. [11] observed, sorting does not increase the ℓ_1 distance between vectors. We construct a $3D$-Lipschitz extension that agrees with deg-list on $\mathcal{G}^D$.

Before explaining our construction, we illustrate the difficulty of the problem with a simpler "straw man" attempt: suppose that, given the degree list deg-list(G), we obtain $\hat{f}_D(G)$ by rounding all degrees above D down to D. This will not affect the degrees in a graph with maximum degree D, but it is not $O(D)$-Lipschitz: consider an n-node star graph with one node of degree $n-1$ and $n-1$ nodes of degree 1. Rounding results in $\hat{f}(G) = (D, 1, ..., 1)$. But the graph has a neighbor G' with no edges at all, for which $\hat{f}(G') = (0, ..., 0)$. Vectors $\hat{f}(G)$ and $\hat{f}(G')$ differ by $n + D - 1$ in the ℓ_1 norm. Other simple ways of dropping very high-degree vertices considered in [4, 19] (as part of the method called "projection") also yield poor bounds on the sensitivity of the obtained degree sequence and result in too much noise being added for privacy.

Like in [19], our starting point is the construction of the flow graph $\mathsf{FG}(G)$ for graph G. In [19], it is shown that half the value of the maximum flow in $\mathsf{FG}(G)$ is a Lipschitz extension of the number of edges in G. We will use the flow values on certain edges as a proxy for degrees of related nodes. The main challenge is that, whereas the *value* of the maximum flow in $\mathsf{FG}(G)$ is unique, the flow on specific edges is not.

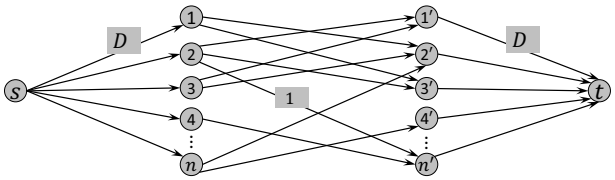

Figure 1. Flow graph illustration. Grey boxes indicate capacities.

Definition III.1 (Flow graph)**.** *Given a graph $G = (V, E)$, let $V_\ell = \{v_\ell \mid v \in V\}$ and $V_r = \{v_r \mid v \in V\}$ be two copies of V, called the* left *and the* right *copies, respectively. Let D be a natural number less than n. The* flow graph *of G with threshold D, a source s, and a sink t is a directed graph on nodes $V_\ell \cup V_r \cup \{s, t\}$ with the following capacitated edges: edges of capacity D from the source s to all nodes in V_ℓ and from all nodes in V_r to the sink t, and unit-capacity edges (u_ℓ, v_r) for all edges (u, v) of G. The flow graph of G is denoted $\mathsf{FG}(G)$. See Figure 1 for an illustration.*

We would like our extension function to output the sorted list of flows leaving the source vertex in some maximum flow. The challenge is that there may be many maximum flows. If we select a maximum flow arbitrarily, then the selected flow may be very sensitive to changes in the graph, even though its value changes little. We resolve this by selecting a flow that minimizes a strictly convex function of the flow values.

Definition III.2 (Lipschitz extension of degree list)**.** *Given a flow f of $\mathsf{FG}(G)$, let $f(e)$ denote the flow on an edge e. Also, let $f_{s\bullet}$ be the vector of flows on the edges leaving the source s, let $f_{\bullet t}$ be the vector of flows on the edges entering t, and let $f_{s\bullet,\bullet t}$ be the concatenation of the two vectors. We use $\vec{D}_{2n}$ to denote a vector of length $2n$, where all entries are D. Let $\Phi(f)$ be the squared ℓ_2 distance between $f_{s\bullet,\bullet t}$ and $\vec{D}_{2n}$, that is,*

$$\begin{aligned}\Phi(f) &= \|f_{s\bullet,\bullet t} - \vec{D}_{2n}\|_2^2 \\ &= \sum_{v \in V} \left((D - f(s, v_\ell))^2 + (D - f(v_r, t))^2\right) .\end{aligned}$$

Let f be the flow that minimizes the objective function Φ over all feasible flows in $\mathsf{FG}(G)$. Define $\hat{f}_D(G)$ to be the sorted list of flows along the edges leaving the source, that is, $\hat{f}_D(G) = sort(f_{s\bullet})$.

Observe that for each G, there is a unique $\hat{f}_D(G)$ because the objective Φ is strictly convex in the values $f_{s\bullet,\bullet t}$. We can approximate $\hat{f}_D(G)$ to arbitrary precision in polynomial time, since it is the minimum of a strongly convex function over a polytope with polynomially many constraints. The approximation may slightly increase the sensitivity; in our application, one can account for this by adding slightly more than $3D/\epsilon$ noise in each coordinate.

Theorem I.3 follows from the following theorem.

Theorem III.3. *The function $\hat{f}_D(G)$ is a Lipschitz extension of deg-list(G) from $\mathcal{G}^D$ to $\mathcal{G}$ of stretch 3/2. In other words,*

1) *If G is D-bounded, then $\hat{f}_D(G) =$ deg-list(G).*
2) *For any two graphs G_1, G_2 (not necessarily D-bounded) that are node neighbors,*

$$\|\hat{f}_D(G_1) - \hat{f}_D(G_2)\|_1 \leq 3D .$$

We do not know if the bound $3D$ in part (2) of the theorem is tight, even for this particular construction. It

could be that this extension has stretch 1 (which would correspond to a bound of $2D$ on the sensitivity).

Proof of Theorem III.3 (item 1): Suppose G is D-bounded. The flow f in $\mathsf{FG}(G)$ that assigns 1 to all edges (u_ℓ, v_r) and $deg(v)$ to all edges (s, v_ℓ) and (v_r, t) is feasible. Moreover, for any feasible flow f' and any edge e in $\mathsf{FG}(G)$, we have $f'(e) \leq f(e)$ because of capacity constraints on edges between V_ℓ and V_r and flow conservation. Finally, f minimizes Φ since, for $x \in [0, D]$, function $(D-x)^2$ is decreasing in x. ∎

There are two distinct notions of optimality of a flow in $\mathsf{FG}(G)$: optimality with respect to Φ, which we call Φ-*optimality*, and optimality of the net flow form s to t, called *net flow optimality*. Next, we show that Φ-optimality implies net flow optimality.

Lemma III.4. *For every graph G, if f is a feasible flow for the flow graph $\mathsf{FG}(G)$ that minimizes Φ, then f is a maximum net flow from s to t in $\mathsf{FG}(G)$.*

Proof: Suppose f is not a maximum net flow. Let p be a shortest augmenting s-t path and $c > 0$ be the minimal residual capacity of an edge in p. Consider the flow c_p that assigns c units of flow to each edge of p and 0 to all other edges. Since p is a shortest path, it is simple. Thus, adding c_p to f results in a feasible flow, but does not decrease the flow along any edge leaving s or entering t. This implies that $\Phi(f + c_p) < \Phi(f)$, since Φ is strictly decreasing in each argument. That is, f is not Φ-optimal. ∎

The flow graph $\mathsf{FG}(G)$ admits a simple symmetry: for any flow f, we can obtain a feasible flow $\pi(f)$ by swapping the roles of s and t and the roles of left and right copies of all vertices. That is, we define $\pi(f)(s, v_\ell) := f(v_r, t)$, $\pi(f)(u_r, t) := f(s, u_\ell)$, $\pi(f)(u_\ell, v_r) := f(v_\ell, u_r)$ for all vertices v, u in G. Flow f is *symmetric* if $\pi(f) = f$. For every graph G, there exists a symmetric Φ-optimal flow in $\mathsf{FG}(G)$: given any Φ-optimal flow f', the flow $f'' = \frac{1}{2}(f' + \pi(f'))$ is symmetric, feasible (because the set of feasible flows is convex) and satisfies $\Phi(f'') \leq \Phi(f')$ by convexity of Φ.

Proof of Theorem III.3 (item 2): Suppose a graph G_1 on $n-1$ nodes is obtained by removing a node v^{new} and its associated edges from a graph G_2 on n nodes.

Let f_1, f_2 be Φ-optimal symmetric flows for the flow graphs $\mathsf{FG}(G_1)$ and $\mathsf{FG}(G_2)$, respectively.

Observe that f_1 is a feasible flow in $\mathsf{FG}(G_2)$. Consider the flow $\Delta = f_2 - f_1$. Note that Δ is a flow in the residual graph of flow f_1 for $\mathsf{FG}(G_2)$. It satisfies flow conservation and capacity constraints, but not necessarily positivity. Since $\|\hat{f}_D(G_1) - \hat{f}_D(G_2)\|_1 = \|\Delta_{s\bullet}\|_1$, our goal is to prove $\|\Delta_{s\bullet}\|_1 \leq 3D$.

Next, we decompose Δ into three *subflows*. A *subflow* Δ' of a flow Δ is a flow that, for all edges e, satisfies $\Delta(e) \cdot \Delta'(e) \geq 0$ and $|\Delta'(e)| \leq |\Delta(e)|$. We start by decomposing Δ into subflows that form simple s-t paths and simple cycles. Then we group them as follows:

- Let Δ^s be the sum of all flows from the initial decomposition that form paths and cycles using the edge (s, v_ℓ^{new}).
- Let Δ^t be the sum of all flows from the initial decomposition that form paths and cycles using the edge (v_r^{new}, t), but not (s, v_ℓ^{new}).
- Let Δ^0 be the sum of the remaining flows, i.e., $\Delta^0 = \Delta - \Delta^s - \Delta^t$.

Since, by definition of the subflow decomposition, $\|\Delta_{s\bullet}\|_1 = \|{\Delta^s}_{s\bullet}\|_1 + \|{\Delta^t}_{s\bullet}\|_1 + \|{\Delta^0}_{s\bullet}\|_1$, it remains to bound the three values in the sum. We do it in the following three lemmas.

Lemma III.5. $\|{\Delta^s}_{s\bullet}\|_1 \leq 2D$.

Proof: Recall that Δ^s can be decomposed into simple s-t paths and simple cycles that use the edge (s, v_ℓ^{new}). Each such path contributes the value of its flow to $\|{\Delta^s}_{s\bullet}\|_1$, and each such cycle contributes at most twice the value of its flow. Since the total flow $\Delta^s(s, v_\ell^{\text{new}})$ is at most D, we get that $\|{\Delta^s}_{s\bullet}\|_1 \leq 2D$. ∎

Lemma III.6. $\|{\Delta^t}_{s\bullet}\|_1 \leq D$.

Proof: Recall that Δ^t can be decomposed into simple s-t paths and cycles that use the edge (v_r^{new}, t), but not (s, v_ℓ^{new}). Each such path contributes the value of its flow to $\|{\Delta^s}_{s\bullet}\|_1$. Any such cycle contributes 0 to $\|{\Delta^s}_{s\bullet}\|_1$ if it does not pass through s. We will show that no simple cycle in the initial decomposition of Δ^t passes through s.

Consider, for the sake of contradiction, a simple cycle that passes through s in the initial decomposition of Δ^t. By definition of Δ^t, the cycle uses the edge (v_r^{new}, t). So, it is of the form $s, v_\ell^i, \ldots, v_r^{\text{new}}, t, v_r^j, \ldots, v_\ell^k, s$ for some nodes i, j, k in G_1. Since Δ^t is a subflow of $\delta = f_2 - f_1$ and f_1 cannot use the edge (v_r^{new}, t), the cycle has positive flow on (v_r^{new}, t). So, the cycle has positive flow on $s, v_\ell^i, \ldots, v_r^{\text{new}}, t$ and negative flow on $s, v_\ell^k, \ldots, v_r^j, t$. But this implies that $s, v_\ell^k, \ldots, v_r^j, t$ is an augmenting path in $\mathsf{FG}(G_2)$ with respect to flow f_2, that is, f_2 is not net flow optimal in $\mathsf{FG}(G_2)$. By Lemma III.4, it contradicts Φ-optimality of f_2 in $\mathsf{FG}(G_2)$. Therefore, no simple cycle passes through s in the initial decomposition of Δ^t.

Since the total flow $\Delta^t(v_r^{\text{new}}, t)$ is at most D, we get that $\|{\Delta^t}_{s\bullet}\|_1 \leq D$. ∎

Lemma III.7. $\|{\Delta^0}_{s\bullet}\|_1 = 0$.

Proof: In this proof, to make notation less cumbersome, we think of flows for $\mathsf{FG}(G_1)$ as flows of $\mathsf{FG}(G_2)$, i.e., living in a higher-dimensional space than necessary to represent them.

The flow Δ^0 does not use the edges (s, v_ℓ^{new}) and (v_r^{new}, t) since all flow in Δ along (s, v_ℓ^{new}) and (v_r^{new}, t) has been used by $\Delta^s + \Delta^t$. Recall that Δ^0 is a subflow of $f_2 - f_1$, so it must carry nonnegative flow along all edges that are in $\mathsf{FG}(G_2)$, but not in $\mathsf{FG}(G_1)$, that is, edges adjacent to v_ℓ^{new} and v_r^{new}. Consequently, Δ^0 has no flow passing through v_ℓ^{new} and v_r^{new}. Therefore, Δ^0 is a feasible flow for the residual graph of f_1 in $\mathsf{FG}(G_1)$. We conclude that $f_1 + \Delta^0$ is feasible in $\mathsf{FG}(G_1)$.

Assume for the sake of contradiction that $\|{\Delta^0}_{s\bullet}\|_1 > 0$. Then we can use the convexity of Φ to prove the following inequalities:

$$\langle {\Delta^0}_{s\bullet,\bullet t}\, , \vec{D}_{2n} - (f_1)_{s\bullet,\bullet t} \rangle \;\leq\; 0. \qquad (1)$$

$$\langle {\Delta^0}_{s\bullet,\bullet t}\, , \vec{D}_{2n} - (f_2 - \Delta^0)_{s\bullet,\bullet t} \rangle \;>\; 0. \qquad (2)$$

To prove (1), consider the polytope $\mathcal{P}_1$ of feasible flows in $\mathsf{FG}(G_1)$. Both f_1 and $f_1 + \Delta^0$ are in $\mathcal{P}_1$. Let $\mathcal{P}_1'$ be the polytope obtained by projecting $\mathcal{P}_1$ onto the $2n$ coordinates corresponding to flows out of s and flows into t. Then $(f_1)_{s\bullet,\bullet t}$ is the unique vector in $\mathcal{P}_1'$ corresponding to Φ-optimal flows in $\mathsf{FG}(G_1)$. Since Φ is minimized at $\vec{D}_{2n}$, a tiny step from $(f_1)_{s\bullet,\bullet t}$ in the direction of $(f_1 + \Delta^0)_{s\bullet,\bullet t}$ takes us further from $\vec{D}_{2n}$. In other words, the angle between the vectors $\vec{D}_{2n} - (f_1)_{s\bullet,\bullet t}$ and $(f_1 + \Delta^0)_{s\bullet,\bullet t} - (f_1)_{s\bullet,\bullet t}$ is at least $90°$, implying (1).

To prove (2), consider the polytope $\mathcal{P}_2$ of feasible flows in $\mathsf{FG}(G_2)$. Both f_2 and $f_2 - \Delta^0$ are in $\mathcal{P}_2$. Let $\mathcal{P}_2'$ be the polytope obtained by projecting $\mathcal{P}_2$ onto the $2n$ coordinates corresponding to flows out of s and into t. Then $(f_2)_{s\bullet,\bullet t}$ is the unique vector in $\mathcal{P}_2'$ corresponding to Φ-optimal flows in $\mathsf{FG}(G_2)$. Since Φ is minimized at $\vec{D}_{2n}$, a tiny step from $(f_2 - \Delta^0)_{s\bullet,\bullet t}$ in the direction of $(f_2)_{s\bullet,\bullet t}$ takes us closer to $\vec{D}_{2n}$. In other words, the angle between the vectors $(f_2)_{s\bullet,\bullet t} - (f_2 - \Delta^0)_{s\bullet,\bullet t}$ and $\vec{D}_{2n} - (f_2 - \Delta^0)_{s\bullet,\bullet t}$ is less than $90°$, implying (2).

Subtracting (1) from (2) and using the fact that $\Delta = f_2 - f_1 = \Delta^s + \Delta^t + \Delta^0$, we get

$$\begin{aligned} &\langle {\Delta^0}_{s\bullet,\bullet t}\, , \vec{D}_{2n} - (f_2 - \Delta^0)_{s\bullet,\bullet t} \rangle \\ &\quad - \langle {\Delta^0}_{s\bullet,\bullet t}\, , \vec{D}_{2n} - (f_1)_{s\bullet,\bullet t} \rangle \;>\; 0; \\ &\langle {\Delta^0}_{s\bullet,\bullet t}, -(f_2 - f_1 - \Delta^0)_{s\bullet,\bullet t} \rangle \;>\; 0; \\ &\langle {\Delta^0}_{s\bullet,\bullet t}, (\Delta^s + \Delta^t)_{s\bullet,\bullet t} \rangle \;<\; 0. \qquad (3) \end{aligned}$$

But Δ^0 and $\Delta^s + \Delta^t$ are both subflows of Δ, so they cannot have opposite signs on any edge, contradicting (3). Therefore, $\|{\Delta^0}_{s\bullet}\|_1 = 0$. ∎

We now complete the proof of Theorem III.3 (Item 2). Recall that $\Delta = \Delta^s + \Delta^t + \Delta^0$ and that Δ^s, Δ^t, and Δ^0 are subflows of Δ. From Lemmas III.5–III.7, we get $\|\hat{f}_D(G_1) - \hat{f}_D(G_2)\|_1 = \|{\Delta_{s\bullet}}\|_1 = \|{\Delta^s}_{s\bullet}\|_1 + \|{\Delta^t}_{s\bullet}\|_1 + \|{\Delta^0}_{s\bullet}\|_1 \leq 3D$, as desired. ∎

IV. Exponential Mechanism For Scores With Varying Sensitivity

In this section, we generalize the exponential mechanism and prove Theorem I.4. A limitation of the utility guarantee of the exponential mechanism, stated in Lemma II.3, is that it depends on the maximum sensitivity of the score functions q_i. In the context of threshold selection for graph algorithms, such a guarantee is meaningless for sparse graphs. This poor utility bound is not merely an artifact of the analysis. The problem is inherent in the algorithm. For example, consider the setting with $k = 2$, where the two score functions have sensitivity $\Delta_1 = 1$ and $\Delta_2 \gg 1$. Further, consider a dataset x with $q_1(x) = 0$ and $q_2(x) = \Delta_2/\epsilon$. On input x, the exponential mechanism selects $\hat{\imath} = 2$ with constant probability, resulting in error of Δ_2/ϵ, which may be arbitrarily larger than Δ_1.

In contrast, we give an algorithm whose error scales with the sensitivity of the *optimal* score function Δ_{i^*}, where $i^* = \operatorname{argmin}_{i\in[k]} q_i(x)$. Our mechanism requires as input an upper bound β on the desired probability of a bad outcome; the algorithm's error guarantee depends on β.

Theorem I.4 (Formal)**.** *For all parameters $\beta \in (0,1)$, $\epsilon > 0$, the generalized exponential mechanism (Algorithm 1) is ϵ-differentially private (with respect to the neighbor relation on U). For all inputs x, with probability at least $1 - \beta$, the output $\hat{\imath}$ satisfies*

$$q_{\hat{\imath}}(x) \leq \min_{i\in[k]} \left(q_i(x) + \Delta_i \cdot \tfrac{4\ln(k/\beta)}{\epsilon} \right). \qquad (4)$$

In particular, our algorithm is competitive with the sensitivity of the true minimizer $i^* = \operatorname{argmin}_{i\in[k]} q_i(x)$ (since the right-hand side of (4) is at most $q_{i^*}(x) + \Delta_{i^*} \cdot \frac{4\ln(k/\beta)}{\epsilon}$). When all the Δ_i's are the same, our algorithm simplifies to running the usual exponential mechanism with $\epsilon' = \epsilon/2$; this justifies the "generalized" name.

The intuition behind the algorithm is as follows: since the score function q_i has different sensitivity for each i, we would like to find an alternative score function which is less sensitive. One simple score would be to compute, for each j, the *distance*, in the space of datasets, from the input x to the nearest dataset y in which $q_j(y)$ is

Algorithm 1: Generalized Exponential Mechanism

Input: Dataset x from universe U,
Parameters: $\epsilon > 0$ and $\beta \in (0,1)$,
$\forall i \in [k]$, a Δ_i-Lipschitz score $q_i : U \to \mathbb{R}$.

Set $t = \frac{2\ln(k/\beta)}{\epsilon}$;

$\forall i \in [k]$, define

$$s_i(x) \leftarrow \max_{j \in [k]} \frac{(q_i(x) + t\Delta_i) - (q_j(x) + t\Delta_j)}{\Delta_i + \Delta_j}$$

/* s_i is 1-Lipschitz. */

Run the exponential mechanism $\mathcal{EM}(x)$ with parameters $\epsilon, \Delta_{max} = 1$, and score functions s_i, i.e., sample an index i from $[k]$ with probability proportional to $\exp(\epsilon \cdot s_i(x)/2)$;

return $\hat{\imath}$, the output of $\mathcal{EM}$.

smallest among the values $\{q_i(y)\}_{i \in [k]}$. (This idea is inspired by the GWAS algorithms of [13].) This score has two major drawbacks: first, it is hard to compute in general; second, more subtly, it will tend to favor indices j with very high sensitivity (since they become optimal with relatively few changes to the data).

Instead, we use a substitute measure that (i) is easy to compute given the scores $q_i(x)$ for $i \in [k]$ and (i) appropriately penalizes scores with large sensitivity. Given a value $t > 0$ (to be set later), define the *normalized score* as

$$\begin{aligned} s_i(x) &= \max_{j \in [k]} \frac{(q_i(x) + t\Delta_i) - (q_j(x) + t\Delta_j)}{\Delta_i + \Delta_j} \\ &= \max_{j \in [k]} \left(\frac{q_i(x) - q_j(x)}{\Delta_i + \Delta_j} + t \cdot \frac{\Delta_i - \Delta_j}{\Delta_i + \Delta_j} \right). \end{aligned}$$

The first term inside the maximum on the right-hand side is an approximation to the Hamming distance from x to the nearest dataset y where score $q_j(\cdot)$ becomes smaller than $q_i(\cdot)$. The second term (containing t and independent of the dataset) penalizes indices i with larger sensitivity.

We obtain an index $\hat{\imath}$ by running the usual exponential mechanism on the normalized scores s_i. The privacy of the mechanism follows from the fact that the new scores have sensitivity at most 1.

Lemma IV.1. *For all $i \in [k]$ and all $t \in \mathbb{R}$, the normalized score $s_i(\cdot)$ has sensitivity at most 1.*

Proof of Theorem I.4: Regardless of the dataset x, Algorithm 1 uses $t = 2\ln(k/\beta)/\epsilon$. By Lemma IV.1, each new score $s_i(\cdot)$ is 1-Lipschitz. Thus, the application of the usual exponential mechanism (or its efficient implementation, "report noisy min") is ϵ-differentially private.

To analyze utility, let m denote the index that minimizes the penalized score $q_i(x) + t\Delta_i$. Then

$$s_m(x) = 0,$$

since each of the terms in the maximum defining s is nonpositive for m (and the term for $j = m$ is 0). By the usual analysis of the exponential mechanism, we have that with probability at least $1 - \beta$,

$$s_{\hat{\imath}}(x) \leq \underbrace{s_m(x)}_{0} + \frac{2\ln(k/\beta)}{\epsilon}.$$

Now consider an arbitrary index j. Since

$$s_{\hat{\imath}}(x) \geq \frac{(q_{\hat{\imath}}(x) + t\Delta_{\hat{\imath}}) - (q_j(x) + t\Delta_j)}{\Delta_{\hat{\imath}} + \Delta_j},$$

we can multiply by $\Delta_{\hat{\imath}} + \Delta_j$ to obtain:

$$\begin{aligned} q_{\hat{\imath}}(x) &\leq q_j(x) + t(\Delta_j - \Delta_{\hat{\imath}}) + \tfrac{2\ln(k/\beta)}{\epsilon} \cdot (\Delta_{\hat{\imath}} + \Delta_j) \\ &= q_j(x) + \Delta_j \left(\tfrac{2\ln(k/\beta)}{\epsilon} + t \right) + \Delta_{\hat{\imath}} \left(\tfrac{2\ln(k/\beta)}{\epsilon} - t \right). \end{aligned}$$

Substituting $t = \frac{2\ln(k/\beta)}{\epsilon}$ yields the desired result. ∎

In the full version, we provide several applications of the generalized exponential mechanism.

V. Degree Distributions and α-Decay

Our techniques provide a significantly more accurate way to release the degree distributions of graphs while satisfying node privacy. To illustrate this, we study the accuracy of our method on graphs that satisfy α-decay, a mild condition on the tail of the degree distribution.

Recall that $\bar{d}(G)$ denotes the average degree of G.

Assumption V.1. *Fix $\alpha \geq 1$. A graph G satisfies α-decay if for all real numbers $t > 1$, $P_G(t \cdot \bar{d}(G)) \leq t^{-\alpha}$.*

All graphs satisfy 1-decay (by Markov's inequality). The assumption is nontrivial for $\alpha > 1$, but it is nevertheless satisfied by almost all widely studied classes of graphs. So-called "scale-free" networks (those that exhibit a heavy-tailed degree distribution) typically satisfy α-decay for $\alpha \in (1,2)$. Regular graphs satisfy α-decay for all $\alpha > 1$.

Kasiviswanathan et al. [19] gave algorithms for releasing the degree distribution using a projection-based technique. Their algorithm required knowledge of the decay parameter α (which was used to select the projection threshold). They bounded the ℓ_1 error of their algorithm in estimating the degree distribution, and showed that it went to 0 as long as $\alpha > 2$ and $\bar{d}$ was polylogarithmic in n. More precisely, they gave an expected error bound of $\mathbb{E}\,\|\hat{p} - p_G\|_1 = \tilde{O}\left(\bar{d}^{\frac{3\alpha}{\alpha+1}} / \left(\epsilon^2 n^{\frac{\alpha-2}{\alpha+1}}\right)\right)$.

Combining the techniques of the previous sections with some additional tools, we design an algorithm $\mathcal{A}_{combo}$ which improves on the previous bound.

Theorem V.2. *Given inputs $G \in \mathcal{G}$ and $\epsilon > 0$, the algorithm $\mathcal{A}_{combo}$ produces an estimate $\hat{p}$ such that, if G satisfies α-decay for $\alpha > 1$, then*

$$\mathbb{E}\left\|\hat{p} - p_G\right\|_1 = O\left(\bar{d}(G)^{\frac{2\alpha}{\alpha+1}} / (\epsilon n)^{\frac{\alpha-1}{\alpha+1}}\right).$$

This error is $o(1)$ as $n \to \infty$ if $\bar{d}(G) = o(\epsilon n)^{\frac{\alpha-1}{2\alpha}}$.

ACKNOWLEDGMENTS

We thank Aleksandar Nikolov for helpful discussions and anonymous referees for useful comments.

REFERENCES

[1] K. Ball. Markov chains, Riesz transforms and Lipschitz maps. *Geometric & Functional Analysis*, 2(2):137–172, 1992.

[2] Y. Benyamini and J. Lindenstrauss. *Geometric nonlinear functional analysis*. American Mathematical Society, 1998.

[3] J. Blocki, A. Blum, A. Datta, and O. Sheffet. The Johnson-Lindenstrauss transform itself preserves differential privacy. In *53rd Symposium on Foundations of Computer Science (FOCS)*, pages 410–419, 2012.

[4] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Innovations in Theoretical Computer Science (ITCS)*, pages 87–96, 2013.

[5] C. Borgs, J. T. Chayes, and A. D. Smith. Private graphon estimation for sparse graphs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1369–1377, 2015.

[6] S. Chen and S. Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *International Conference on Management of Data (SIGMOD)*, pages 653–664, 2013.

[7] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-Law Distributions in Empirical Data. *SIAM Review*, 51(4):661–703, 2009.

[8] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Now Publishers Inc., 2014.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TCC)*, pages 265–284, 2006.

[10] A. Gupta, A. Roth, and J. Ullman. Iterative constructions and private data release. In *9th Theory of Cryptography Conference (TCC)*, pages 339–356, 2012.

[11] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *Int. Conf. Data Mining (ICDM)*, pages 169–178, 2009.

[12] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.

[13] A. Johnson and V. Shmatikov. Privacy-preserving data exploration in genome-wide association studies. In *19th International Conference on Knowledge Discovery and Data Mining*, pages 1079–1087. ACM, 2013.

[14] W. B. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of Lipschitz maps into Banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986.

[15] V. Karwa and A. Slavkovic. Inference using noisy degrees: Differentially private β-model and synthetic graphs. *Ann. Statist.*, 44(1):87–112, 2016.

[16] V. Karwa and A. B. Slavkovic. Differentially private graphical degree sequences and synthetic graphs. In *Privacy in Statistical Databases (PSD)*, pages 273–285, 2012.

[17] V. Karwa, S. Raskhodnikova, A. D. Smith, and G. Yaroslavtsev. Private analysis of graph structure. *ACM Trans. Database Syst.*, 39(3):22:1–22:33, 2014.

[18] V. Karwa, A. B. Slavkovic, and P. N. Krivitsky. Differentially private exponential random graphs. In *Privacy in Statistical Databases (PSD)*, pages 143–155, 2014.

[19] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing graphs with node-differential privacy. In *Theory of Cryptography Conference (TCC)*, pages 457–476, 2013.

[20] M. Kirszbraun. Über die zusammenziehende und Lipschitzsche transformationen. *Fundamenta Mathematicae*, 22(1):77–108, 1934.

[21] U. Lang, B. Pavlović, and V. Schroeder. Extensions of Lipschitz maps into hadamard spaces. *Geometric & Functional Analysis GAFA*, 10(6):1527–1553, 2000.

[22] J. R. Lee and A. Naor. Extending Lipschitz functions via random metric partitions. *Inventiones mathematicae*, 160(1): 59–95, 2005.

[23] B.-R. Lin and D. Kifer. Information preservation in statistical privacy and Bayesian estimation of unattributed histograms. In *International Conference on Management of Data (SIGMOD)*, pages 677–688, 2013.

[24] W. Lu and G. Miklau. Exponential random graph estimation under differential privacy. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 921–930. ACM, 2014.

[25] K. Makarychev and Y. Makarychev. Metric extension operators, vertex sparsifiers and Lipschitz extendability. In *51th Symposium on Foundations of Computer Science (FOCS)*, pages 255–264, 2010.

[26] M. Marcus and G. Pisier. Characterizations of almost surely continuous p-stable random Fourier series and strongly stationary processes. *Acta mathematica*, 152(1):245–301, 1984.

[27] J. Matoušek. Extension of Lipschitz mappings on metric trees. *Commentationes Mathematicae Universitatis Carolinae*, 31(1): 99–104, 1990.

[28] E. J. McShane. Extension of range of functions. *Bull. Amer. Math. Soc.*, 40(12):837–842, 1934.

[29] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2007.

[30] D. J. Mir and R. N. Wright. A differentially private estimator for the stochastic Kronecker graph model. In *EDBT/ICDT Workshops*, pages 167–176, 2012.

[31] A. Naor. A phase transition phenomenon between the isometric and isomorphic extension problems for Hölder functions between L_p spaces. *Mathematika*, 48(1/2; ISSU 95/96):253–272, 2001.

[32] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Symp. Theory of Computing (STOC)*, pages 75–84. ACM, 2007. Full paper: http://www.cse.psu.edu/~asmith/pubs/NRS07.

[33] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: output perturbation for queries with joins. In *Symp. Principles of Database Systems (PODS)*, pages 107–116, 2009.

[34] J. H. Wells and L. R. Williams. *Embeddings and extensions in analysis*, volume 84 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer, 1975.

[35] Q. Xiao, R. Chen, and K. Tan. Differentially private network data release via structural inference. In *20th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 911–920, 2014.