Analyzing Graphs with Node Differential Privacy

Shiva Prasad Kasiviswanathan[1⋆], Kobbi Nissim[2], Sofya Raskhodnikova[3⋆⋆], and Adam Smith[3⋆⋆⋆]

[1] General Electric Global Research, USA; kasivisw@gmail.com
[2] Ben-Gurion University, Israel; kobbi@cs.bgu.ac.il
[3] Pennsylvania State University, USA; {sofya,asmith}@cse.psu.edu

Abstract. We develop algorithms for the private analysis of network data that provide accurate analysis of realistic networks while satisfying stronger privacy guarantees than those of previous work. We present several techniques for designing *node* differentially private algorithms, that is, algorithms whose output distribution does not change significantly when a node and all its adjacent edges are added to a graph. We also develop methodology for analyzing the accuracy of such algorithms on realistic networks.

The main idea behind our techniques is to "project" (in one of several senses) the input graph onto the set of graphs with maximum degree below a certain threshold. We design projection operators, tailored to specific statistics that have low sensitivity and preserve information about the original statistic. These operators can be viewed as giving a fractional (low-degree) graph that is a solution to an optimization problem described as a maximum flow instance, linear program, or convex program. In addition, we derive a generic, efficient reduction that allows us to apply any differentially private algorithm for bounded-degree graphs to an arbitrary graph. This reduction is based on analyzing the smooth sensitivity of the "naive" truncation that simply discards nodes of high degree.

1 Introduction

Data from social and communication networks have become a rich source of insights in the social and information sciences. Gathering, sharing and analyzing these data is challenging, however, in part because they are often highly sensitive (your Facebook friends or the set of people you email reveal a tremendous amount of information about you, as in, e.g., Jernigan and Mistree [1]). This paper develops algorithms for the private analysis of network data that provide accurate analysis of realistic networks while satisfying stronger privacy guarantees than those of previous work.

A recent line of work, starting from Dinur and Nissim [2], investigates rigorous definitions of privacy for statistical data analysis. Differential privacy (Dwork *et al.* [3, 4]), which emerged from this line of work, has been successfully used in the context of

⋆ Part of this wok was done while the author was a postdoc at Los Alamos National Laboratory and IBM T.J. Watson Research Center.
⋆⋆ Supported by NSF CAREER grant CCF-0845701 and NSF grant CDI-0941553
⋆⋆⋆ Supported by NSF Awards CCF-0747294 and CDI-0941553 as well as Penn State Clinical & Translational Research Institute, NIH/NCRR Award UL1RR033184.

"tabular", or "array" data. Roughly, differential privacy guarantees that changes to one person's data will not significantly affect the output distribution of an analysis procedure.

For tabular data, it is clear which data "belong" to a particular individual. In the context of graph data, two interpretations of this definition have been proposed: *edge* and *node* differential privacy. Intuitively, edge differential privacy ensures that an algorithm's output does not reveal the inclusion or removal of a particular edge in the graph, while node differential privacy hides the inclusion or removal of a node together with all its adjacent edges.

Node privacy is a strictly stronger guarantee, but until now there have been no node-private algorithms that can provide accurate analysis of the sparse networks that arise in practice. One challenge is that for many natural statistics, node privacy is impossible to achieve while getting accurate answers in the worst case. The problem, roughly, is that node-private algorithms must be robust to the insertion of a new node in the graph, but the properties of a sparse graph can be altered dramatically by the insertion of a well-connected node. For example, for common graph statistics – the number of edges, the frequency of a particular subgraph – the change can overwhelm the value of the statistic in sparse graphs.

In this paper we develop several techniques for designing differentially *node*-private algorithms, as well as a methodology for analyzing their accuracy on realistic networks. The main idea behind our techniques is to "project" (in one of several senses) the input graph onto the set of graphs with maximum degree below a certain threshold. The benefits of this approach are two-fold. First, node privacy is easier to achieve in bounded-degree graphs since the insertion of one node affects only a relatively small part of the graph. Technically, the *sensitivity* of a given query function may be much lower when the function is restricted to graphs of a given degree. Second, for realistic networks this transformation loses relatively little information when the degree threshold is chosen carefully.

The difficulty with this approach is that the projection itself may be very sensitive to a change of a single node in the original graph. We handle this difficulty via two different techniques. First, for a certain class of statistics, we design tailored projection operators that have low sensitivity and preserve information about a given statistic. These operators can be viewed as giving a fractional (low-degree) graph that is a solution to a convex optimization problem, typically given by a maximum flow instance or linear program. Using such projections we get algorithms for accurately releasing the number of edges in a graph, and counts of small subgraphs such as triangles, k-cycles, and k-stars (used as sufficient statistics for popular graph models) in a graph, and certain estimators for power law graphs (see Sections 4 and 5).

Our second technique is much more general: we analyze the "naive" projection that simply discards high-degree nodes in the graph. We give efficient algorithms for bounding the "local sensitivity" of this projection, which measures how sensitive it is to changes in a particular input graph. Using this, we derive a generic, efficient reduction that allows us to apply any differentially private algorithm for bounded-degree graphs to an arbitrary graph. The reduction's loss in accuracy depends on how far the input

graph is from having low degree. We use this to design algorithms for releasing the entire degree distribution of a graph.

Because worst-case accuracy guarantees are problematic for node-private algorithms, we analyze the accuracy of our algorithms under a mild assumption on the degree distribution of the input graph. The simplest guarantees are for the case where a bound D on the maximum degree of the graph is known, and the guarantees typically relate the algorithms's accuracy to how quickly the query function can change when restricted to graphs of degree D (e.g., Corollary 6.1). However, real-world networks are not well-modeled by a graphs of a fixed degree, since they often exhibit influential, high-degree nodes. In our main results, we assume only that 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 $\alpha > 1$ or $\alpha > 2$, depending on the result.) Our assumption is satisfied by all the well-studied social network models we know of, including so-called *scale-free* graphs [5].

1.1 Related Work

The initial statements of differential privacy [3, 4] considered databases that are arrays or sets – each individual's information corresponds to an entry in the database, and this entry may be changed without affecting other entries. That paper also introduced the very basic technique for constructing differentially private function approximations, by the addition of Laplace noise calibrated to the *global sensitivity* of the function.[4] This notion naturally extends to the case of graph data, where each individual's information corresponds to an *edge* in the graph (edge privacy). The basic technique of Dwork *et al.* [3] continues to give a good estimate, e.g., for counting the number of edges in a graph, but it ceases to provide good analyses even for some of the most basic functions of graphs (diameter, counting the number of occurrences of a small specified subgraph) as these functions exhibit high global sensitivity.

The first differentially private computations over graph data appeared in Nissim *et al.* [6] where it was shown how to estimate, with differential edge privacy, the cost of the minimum spanning tree and the number of triangles in a graph. These computations employed a different noise addition technique, where noise is calibrated to a more local variant of sensitivity, called *smooth sensitivity*. These techniques and results were further extended by Karwa *et al.* [7]. Hay *et al.* [8] showed that the approach of [3] can still be useful when combined with a post-processing technique for removing some of the noise. They use this technique for constructing a differentially edge-private algorithm for releasing the degree distribution of a graph. They also proposed the notion of differential node privacy and highlighted some of the difficulties in achieving it.

A different approach to graph data was suggested by Rastogi *et al.* [9], where the privacy is weakened to a notion concerning a Bayesian adversary whose prior distribution on the database comes from a specified family of distributions. Under this notion of privacy, and assuming that the adversary's prior admits mainly negative correlations

[4] Informally, global sensitivity of a function measures the largest change in the function outcome than can result from changing one of its inputs.

between edges, they give an algorithm for counting the occurrences of a specified subgraph. The notion they use, though, is weaker than differential edge privacy. We refer the reader to [7] for a discussion on how the assumptions about an attacker's prior limit the applicability of the privacy definition.

The current work considers databases where nodes correspond to individuals, and edges correspond to relationships between these individuals. Edge privacy corresponds in this setting to a requirement that the properties of every relationship (such as its absence or presence) should be kept hidden, but the overall relationship pattern of an individual may be revealed. However, each individual's information corresponds to *all* edges adjacent to her node and a more natural extension of differential privacy for this setting would be that this entire information should be kept hidden. This is what we call *node privacy* (in contrast with *edge privacy* guaranteed in prior work). A crucial deviation from edge privacy is that a change in the information of one individual can affect the information of all other individuals. We give methods that provide node privacy for a variety of types of graphs, including very sparse graphs.

Finally, motivated by examples from social networks Gehrke *et al.* [10] suggest a stronger notion than differential node privacy – called *zero-knowledge privacy* – and demonstrate that this stronger notion can be achieved for several tasks in extremely dense graphs. Zero-knowledge privacy, as they employ it, can be used to release quantities that can be computed from small, random induced subgraphs of a larger graph. Their techniques are not directly applicable to sparse graphs (since a random induced subgraph will contain very few edges, with high probability).

We note that while node privacy gives a very strong guarantee, it may not answer all privacy concerns in a social network. Kifer and Machanavajjhala [11] criticize differential privacy in the context of social networks, noting that individuals can have a greater effect on a social network than just forming their own relationships (their criticism is directed at edge privacy, but it can also apply to node privacy).

Concurrent Work. In independent work, Blocki *et al.* [12] also consider node-level differential private algorithms for analyzing sparse graphs. Both our work and that of Blocki *et al.* are motivated by getting good accuracy on sparse graphs, and employ projections onto the set of low-degree graphs to do so. The two works differ substantially in the technical details. See Appendix A for a detailed comparison.

Organization. Section 2 defines the basic framework of node and edge privacy and gives background on sensitivity and noise addition that is needed in the remainder of the paper. Section 3 introduces a useful, basic class of queries that can be analyzed with node privacy, namely queries that are linear in the degree distribution. Section 4 gives our first projection technique based on maximum flow and applies it to privately estimate the number of edges in a graph (Section 4.2). Section 4.3 generalizes the flow technique to apply it to any concave function on degree. Section 5 provides a private (small) subgraph counting algorithm via linear programming. Finally, Section 6 describes our general reduction from privacy on all graphs to the design of algorithms that are private only on bounded-degree graphs, and applies it to privately release the (entire) degree distribution. Due to space constraints, all proofs are deferred to the full version of this paper.

2 Preliminaries

Notation. We use $[n]$ to denote the set $\{1, \ldots, n\}$. For a graph, (V, E), $\bar{d}(G) = 2|E|/|V|$ is the average degree of the graph G and $\deg_v(G)$ denotes the degree of node $v \in V$ in G. When the graph referenced is clear, we drop G in the notation. The asymptotic notation $O_n(\cdot), o_n(\cdot)$ is defined with respect to growing n. Other parameters are assumed to be functions independent of n unless specified otherwise.

Let $\mathcal{G}$ denote the set of unweighted, undirected finite *labeled* graphs, and let $\mathcal{G}_n$ denote the set of graphs on at most n nodes and $\mathcal{G}_{n,D}$ be the set of all graphs in $\mathcal{G}_n$ with maximum degree D.

2.1 Graphs Metrics and Differential Privacy

We consider two metrics on the set of labeled graphs: node and edge distance. The *node* distance $d_{\text{node}}(G, G')$ (also called *rewiring distance*) between graphs G and G' is the minimum number of nodes in G' that need to be changed ("rewired") to obtain G. Rewiring allows one to add a new node (with an arbitrary set of edges to existing nodes), remove it entirely, or change its adjacency lists arbitrarily. In particular, a rewiring can affect the adjacency lists of all other nodes. Equivalently, let k is the number of nodes in the largest induced subgraph of G which equals the corresponding induced subgraph of G'. The node distance is $d_{\text{node}}(G, G') = \max\{|V_G|, |V_{G'}|\} - k$. Graphs G, G' are node neighbors if their node distance is 1.

The *edge* distance $d_{\text{edge}}(G, G')$ is the minimum number of edges in G' that need to be changed (*i.e.,* added or deleted) to obtain G. We also count insertion or removal of an isolated node (to allow for graphs with different number of nodes). In this paper, distance between graphs refers to the node distance unless specified otherwise.

Definition 2.1 ((ϵ, δ)-differential Privacy [3, 4, 13]) *A randomized algorithm $\mathcal{A}$ is* (ϵ, δ)-node-private *(resp.* edge-private*) if for all events S in the output space of $\mathcal{A}$, and for all graphs G, G' at rewiring distance 1 (resp. edge-distance 1) we have:*

$$\Pr[\mathcal{A}(G) \in S] \leq \exp(\epsilon) \times \Pr[\mathcal{A}(G') \in S] + \delta\,.$$

When $\delta = 0$, the algorithm is ϵ-differentially private. In this paper, if node or edge privacy is not specified, we mean node privacy by default.

In this paper, 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 a very accurate estimate of $|V|$ via a node-private query. Moreover, given a publicly known value n, one can force the input graph $G = (V, E)$ to have n nodes without sacrificing differential node privacy: one either pads the graph with isolated nodes (if $|V| < n$) or discards the $|V| - n$ "excess" nodes with the largest labels (if $|V| > n$) along with all their adjacent edges. Changing one node of G corresponds to a change of at most one node in the resulting n-node graph as long as the differentially private algorithms being run on the data do not depend on the labeling (i.e., they should be symmetric in the order of the labels).

Differential privacy "composes" well, in the sense that privacy is preserved (albeit with slowly degrading parameters) even when the adversary gets to see the outcome of multiple differentially private algorithms run on the same data set.

Lemma 2.1 (Composition, Post-processing [14, 15]). *If an algorithm $\mathcal{A}$ runs t randomized algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_t$, each of which is (ϵ, δ)-differentially private, and applies an arbitrary (randomized) algorithm g to their results, i.e., $\mathcal{A}(G) = g(\mathcal{A}_1(G), \ldots, \mathcal{A}_t(G))$, then $\mathcal{A}$ is $(t\epsilon, t\delta)$-differentially private.*

2.2 Calibrating Noise to Sensitivity

Output Perturbation. One common method for obtaining efficient differentially private algorithms for approximating real-valued functions is based on adding a small amount of random noise to the true answer. In this paper, we use two families of random distributions to add noise: Laplace and Cauchy. 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 $\mathrm{Lap}(\lambda)$. A *Cauchy* random variable with median 0 and median absolute deviation λ has density $h(z) = 1/(\lambda\pi(1 + (z/\lambda)^2))$. We denote it by $\mathrm{Cauchy}(\lambda)$.

Global Sensitivity. The most basic framework for achieving differential privacy, Laplace noise is scaled according to the *global sensitivity* of the desired statistic f. This technique extends directly to graphs as long as we measure sensitivity with respect to the same metric as differential privacy. Below, we define these (standard) notions in terms of node distance and node privacy. Recall that $\mathcal{G}_n$ is the set of all n-node graphs.

Definition 2.1 (Global Sensitivity [3]). *The ℓ_1-global node sensitivity of a function $f : \mathcal{G}_n \to \mathbb{R}^p$ is:*

$$\Delta f = \max_{G, G' \text{ node neighbors}} \|f(G) - f(G')\|_1 .$$

For example, the number of edges in a graph has node sensitivity n (when we restrict our attention to n-node graphs), since rewiring a node can add or remove at most n nodes. In contrast, the number of nodes in a graph has node sensitivity 1, even when we consider graphs of all sizes (not just a fixed size n).

Theorem 2.2 (Laplace Mechanism [3]). *The algorithm $\mathcal{A}(G) = f(G) + \mathrm{Lap}(\Delta f/\epsilon)^p$ (i.e., adds i.i.d. noise $\mathrm{Lap}(\Delta f/\epsilon)$ to each entry of f), 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 differential 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 $(n-1)/\epsilon$ (the global sensitivity for releasing edge count is $n-1$).

Local Sensitivity. The magnitude of noise added by the Laplace mechanism depends on Δf and the privacy parameter ϵ, but not on the database G. For many functions, this approach yields high noise, not reflecting the function's typical insensitivity to individual inputs. Nissim *et al.* [6] proposed a local measure of sensitivity, defined next.

Definition 2.2 (Local Sensitivity [6]). *For a function $f : \mathcal{G}_n \to \mathbb{R}^p$ and a graph $G \in \mathcal{G}_n$, the local sensitivity of f at G is $LS_f(G) = \max_{G'} \|f(G) - f(G')\|_1$, where the maximum is taken over all node neighbors G' of G.*

Note that, by Definitions 2.1 and 2.2, the global sensitivity $\Delta f = \max_G LS_f(G)$. One may think of the local sensitivity as a discrete analogue of the magnitude of the gradient of f.

A straightforward argument shows that every differentially private algorithm must add distortion at least as large as the local sensitivity on many inputs. However, finding algorithms whose error matches the local sensitivity is *not* straightforward: an algorithm that releases f with noise magnitude proportional to $LS_f(G)$ on input G is not, in general, differentially private [6], since the noise magnitude itself can leak information.

Smooth Bounds on LS. Nissim *et al.* [6] propose the following approach: instead of using the local sensitivity, select noise magnitude according to a *smooth* upper bound on the local sensitivity, namely, a function S that is an upper bound on LS_f at all points and such that $\ln(S(\cdot))$ has low global sensitivity. The level of smoothness is parameterized by a number β (where smaller numbers lead to a smoother bound) which depends on ϵ.

Definition 2.3 (Smooth Bounds [6]). *For $\beta > 0$, a function $S : \mathcal{G}_n \to \mathbb{R}$ is a β-smooth upper bound on the local sensitivity of f if it satisfies the following requirements:*

$$\begin{aligned} \textit{for all } G \in \mathcal{G}_n : &\qquad S(G) \geq LS_f(G); \\ \textit{for all neighbors } G, G' \in \mathcal{G}_n : &\qquad S(G) \leq e^{\beta} S(G'). \end{aligned}$$

One can add noise proportional to smooth bounds on the local sensitivity using a variety of distributions. We state here the version based on the Cauchy distribution.

Theorem 2.3 (Calibrating Noise to Smooth Bounds [6]). *Let $f : \mathcal{G}_n \to \mathbb{R}^p$ be a real-valued function and let S be a β-smooth bound on LS_f. If $\beta \leq \epsilon/(\sqrt{2}p)$, the algorithm $\mathcal{A}(G) = f(G) + \mathrm{Cauchy}(\sqrt{2}S(G)/\epsilon)^p$ (adding i.i.d. $\mathrm{Cauchy}(\sqrt{2}S(G)/\epsilon)$ to each coordinate of f) is ϵ-differentially private.*

From the properties of Cauchy distribution, the algorithm of the previous theorem has median absolute error $(\sqrt{2}S(G))/\epsilon$ (the median absolute error is the median of the random variable $|\mathcal{A}(G) - f(G)|$, where $\mathcal{A}(G)$ is the released value and $f(G)$ is the query answer). Note that the *expected* error of Cauchy noise is not defined. One can get a similar result with an upper bound on any finite moment of the error using different heavy-tailed probability distributions [6]. We use Cauchy noise here for simplicity.

To compute smooth bounds efficiently, it is convenient to break the expression defining it down into tractable components. For every distance t, consider the largest local sensitivity attained on graphs at distance at most t from G. The *local sensitivity of f at distance t* is:

$$LS^{(t)}(G) = \max_{G' \in \mathcal{G}_n \,:\, d_{\mathrm{node}}(G,G') \leq t} LS_f(G') \ .$$

Now the smooth sensitivity is: $S^*_{f,\beta}(G) = \max_{t=0,\ldots,n} e^{-t\beta} LS^{(t)}(G)$. Many smooth bounds on the local sensitivity have a similar form, with $LS^{(t)}$ being replaced by some other function $C^{(t)}(G)$ with the property that $C^{(t)}(G) \leq C^{(t+1)}(G')$ for all pairs of neighbors G, G'. For example, our bounds on the sensitivity of naive truncation have this form (Proposition 6.1, Section 6).

2.3 Sensitivity and Privacy on Bounded-degree Graphs

A graph is D-bounded if it has maximum degree at most D. The degree bound D can be a function of the number of nodes in the graph. We can define a variant of differential privacy that constrains an algorithm only on these bounded-degree graphs.

Definition 2.4 (Bounded-degree (ϵ, δ)-differential Privacy) *A randomized algorithm $\mathcal{A}$ is $(\epsilon, \delta)_D$-node-private (resp. $(\epsilon, \delta)_D$-edge-private) if for all pairs of D-bounded graphs $G_1, G_2 \in \mathcal{G}_{n,D}$ that differ in one node (resp. edge), we have $\Pr[\mathcal{A}(G) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{A}(G') \in \mathcal{S}] + \delta$.*

In bounded-degree graphs, the difference between edge privacy and node privacy is relatively small. For example, an $(\epsilon, 0)_D$-edge-private algorithm is also $(\epsilon D, 0)_D$-node-private (and a similar statement can be made about (ϵ, δ) privacy, with a messier growth in δ).

The notion of global sensitivity defined above (from previous work) can also be refined to consider only how the function may change within $\mathcal{G}_{n,D}$, and we can adjust the Laplace mechanism correspondingly to add less noise while satisfying $(\epsilon, 0)_D$-differential privacy.

Definition 2.4 (Global Sensitivity on Bounded Graphs). *The ℓ_1-global node sensitivity on D-bounded graphs of a function $f : \mathcal{G}_n \to \mathbb{R}^p$ is:*

$$\Delta_D f = \max_{G, G' \in \mathcal{G}_{n,D}:\ d_{\text{node}}(G,G')=1} \|f(G) - f(G')\|_1 .$$

Observation 2.5 (Laplace Mechanism on Bounded Graphs) *The algorithm $\mathcal{A}(G) = f(G) + \text{Lap}\left(\Delta_D f/\epsilon\right)^p$ is $(\epsilon, 0)_D$-node-private.*

2.4 Assumptions on Graph Structure

Let p_G denote the degree distribution of the graph G, i.e., $p_G(k) = \left|\{v : \deg_v(G) = k\}\right|/|V|$. Similarly, P_G denotes the *cumulative* degree distribution, i.e., $P_G(k) = \left|\{v : \deg_v(G) \geq k\}\right|/|V|$. Recall that $\bar{d}(G) = 2|E|/|V|$ is the average degree of G.

Assumption 2.6 (α-decay) *Fix $\alpha \geq 1$. A graph G satisfies* α-decay *if for all*[5] *real numbers $t > 1$, $P_G(t \cdot \bar{d}) \leq t^{-\alpha}$.*

Note that *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 $\alpha \in (1, 2)$. Random graphs satisfy α-decay for essentially arbitrarily large α since their degree distributions have tails that decay exponentially (more precisely, for any α we can find a constant c_α such that, with high probability, α-decay holds when $t > c_\alpha$). Regular graphs satisfy the assumption with $\alpha = \infty$. Next we consider an implication of α-decay.

[5] Our results hold even when this condition is satisfied only for sufficiently large t. For simplicity, we use a stronger assumption in our presentation.

Lemma 2.2. *Consider a graph G on n nodes that satisfies α-decay for $\alpha > 1$, and let $D > \bar{d}$. Then the number of edges in G adjacent to nodes of degree at least D is $O(\bar{d}^{\alpha} n / D^{\alpha-1})$.*

3 Linear Queries in the Degree Distribution

The first, and simplest, queries we consider are functions linear in the degree distribution. In many cases, these can be released directly with node privacy, though they also highlight why bounding the degree leads to such a drastic reduction in sensitivity. Suppose we are given a function $h : \mathbb{N} \to \mathbb{R}^{\geq 0}$ that takes nonnegative real values. We can extend it to a function on graphs as follows:

$$F_h(G) \stackrel{\text{def}}{=} \sum_{v \in G} h(\deg_v(G)) \,,$$

where $\deg_v$ is the degree of the node v in G. We will drop the superscript in F_h when h is clear from the context. The query F_h can also be viewed as the inner product of $\boldsymbol{h} = (h(0), \ldots, h(n-1))$ with the degree distribution p_G, scaled up by n, i.e., $F_h(G) = n\langle \boldsymbol{h}, p_G \rangle$.

Several natural quantities can be expressed as linear queries. The number of edges in the graph, for example, corresponds to half the identity function, that is, $h(i) = i/2$ (since the sum of the degrees is twice the number of edges). The number of nodes in the graph is obtained by choosing the constant function $h(i) = 1$. The number of nodes with degrees in a certain range – say above a threshold D – also falls into this category. Less obviously, certain subgraph counting queries, namely, the number of k-stars for a given k, can be obtained by taking $h(i) = \binom{i}{k}$ for $i \geq k$ (and $h(i) = 0$ for $i < k$).

The sensitivity of these linear queries depends on the maximum value that h can take as well as the largest jump in h over the interval $\{0, \ldots, n-1\}$. Let

$$\|h'\|_\infty \stackrel{\text{def}}{=} \max_{0 \leq i < n-1} |h(i+1) - h(i)| \,.$$

We refer to $\|h'\|_\infty$ as the *maximum slope* of h. This quantity depends on n, though we leave n out of the notation for clarity. Let

$$\|h\|_\infty \stackrel{\text{def}}{=} \left(\max_{0 \leq i \leq n-1} |h(i)| \right) .$$

Lemma 3.1. *The sensitivity of F_h on $\mathcal{G}_n$ is at most $\Delta F_h \leq \|h\|_\infty + (n-1) \cdot \|h'\|_\infty$. If there is a value $j \in \{0, \ldots, n-1\}$ such that $h(j) = 0$, then $\Delta F_h \leq 2(n-1)\|h'\|_\infty$.*

This simple rule immediately gives us tight bounds on the sensitivity of several natural functions, such as the number of nodes, number of edges and the number of k-stars for a given k). We mention here two functions that come up later in the paper.

(1) Common Estimators for Power Law Coefficients: Many real-world networks exhibit heavy-tailed degree distributions, and a common goal of analysts is to identify the coefficient of a power law that best fits the data (we note that power laws

are not the only heavy-tailed distributions, but they are very popular). One well-studied approach to identifying the power law coefficient is to treat the degrees as n independent samples from a power law distribution (Clauset *et al.* [5]). In that case, the maximum likelihood estimator for the exponent is $1 + n/M(G)$ where $M(G) = \sum_{v \in V} \ln(\deg_v)$. Note that M is a linear function of the degree distribution (as $M(G) = F_h(G)$ with $h(i) = \ln(i)$ for $i \geq 1$ and $h(0) = 0$) with maximum slope $\ln(2) - \ln(1) = \ln(2)$ and maximum value $\ln(n-1)$. The sensitivity of M is $\Theta(n)$. Therefore, applying the Laplace mechanism directly is problematic, since the noise (of magnitude $O(n/\epsilon)$) will swamp the value of the query. In Section 4.3, we propose a different approach (based on convex programming) for privately releasing these estimators.

(2) Counting Nodes in an Interval: If $f = \chi_{[a,b]}$ where $\chi_{[a,b]}(i) = 1$ if $a \leq i \leq b$, and 0 otherwise, then F_f counts the number of nodes of degree between thresholds a and b. However, the sensitivity $\Delta F_f = \Theta(n)$, making the answer to this query useless once Laplace noise has been added.

We can reduce the sensitivity of this query by *tapering* the characteristic function of the interval. Given an interval $[a, b]$, consider the tapered step function $f_{t,a,b}(i) = \max\{0, 1 - t \cdot dist(i, [a, b])\}$, where $dist(i, [a, b])$ denotes the distance from i to the nearest point in the interval $[a, b]$. The maximum slope of $f_{t,a,b}$ is t, so $\Delta F_{f_{t,a,b}} = 2tn$. Answers to this query may be meaningful for any $t = o(1)$ (since then the sensitivity will be $o(n)$). We will find this sort of "smoothed" counting query to be useful when estimating how many nodes of high degree there are in a graph (see Proposition 6.1, Section 6).

The linear queries already give us a toolkit for analyzing graphs with node privacy, much as linear queries (over the data points) give a powerful basic toolkit for the differentially private analysis of conventional data sets (as in the SuLQ framework of Blum *et al.* [16]). The difference, of course, is that we need to consider slowly varying functions in order to keep the sensitivity low.

Graphs of Bounded Degree Notice that the techniques mentioned above for bounding the sensitivity of a linear query work better in bounded-degree graphs. Specifically, the sensitivity of F_h on D-bounded graphs is at most

$$\Delta F_h \leq \|h\|_\infty + D\|h'\|_\infty \,. \tag{1}$$

This motivates the approaches in the remainder of the paper, which seek to first bound the degree via a projection step.

4 Flow-based Lipschitz Extensions

We now present our flow-based technique. In Section 4.1, we define a flow function and show that it has low global node sensitivity and, on bounded-degree graphs, it correctly computes the number of edges in the graph. In Section 4.2, we design a node-private algorithm for releasing the number of edges in a graph based on this flow function.

4.1 Flow Graph

Definition 4.1 (Flow graph). *Given an (undirected) 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 parameter 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. Let $v_{\mathrm{fl}}(G)$ denote the value of the maximum flow in the flow graph of G.*

Lemma 4.1. *The global node sensitivity $\Delta v_{\mathrm{fl}} \leq 2D$.*

Lemma 4.2. *For all graphs G, the value $v_{\mathrm{fl}}(G) \leq 2f_e(G)$. Moreover, if G is D-bounded then $v_{\mathrm{fl}}(G) = 2f_e(G)$.*

4.2 Algorithm for Releasing the Number of Edges

In this section, we design a node-private algorithm for releasing the number of edges. The main challenge in applying the methodology from the previous section is that we need to select a good threshold D that balances two conflicting goals: keeping the sensitivity low and retaining as large a fraction of the graph as possible.

Given a graph G, let $f_e(G)$ be the number of edges in G. Observe that the global node sensitivity of the edge count, Δf_e, is at most n because rewiring (or adding/removing) a node can change this count by at most n. So releasing f_e with Laplace noise of the magnitude n/ϵ is ϵ-node-private. The resulting approximate count is accurate if the number of edges in the input graph G is large. The following algorithm allows us to release an accurate count even when this number is low, provided that G satisfies α-decay, a natural assumption discussed in Section 2.4.

Algorithm 1 ϵ-Node-Private Algorithm for Releasing $f_e(G)$

Input: parameters ϵ, D, n, and graph G on n nodes.

1: Let $\hat{e}_1 = f_e(G) + \mathrm{Lap}(\frac{2n}{\epsilon})$ and threshold $\tau = \frac{n \ln n}{\epsilon}$.
2: If $\hat{e}_1 \geq 3\tau$, **return** $\hat{e}_1$.
3: Else compute the flow value $v_{\mathrm{fl}}(G)$ given in Definition 4.1 with D.
4: **return** $\hat{e}_2 = v_{\mathrm{fl}}(G)/2 + \mathrm{Lap}(\frac{2D}{\epsilon})$.

Lemma 4.3. *Algorithm 1 is an ϵ-node-private algorithm that takes a graph G and parameters ϵ, n, D, and outputs an approximate count for $f_e(G)$ (number of edges in G).*

1. *If $f_e(G) \geq (5n \ln n)/\epsilon$, then with probability at least $1 - 1/\ln n$, Algorithm 1 outputs $\hat{e}_1$ with*
$$|\hat{e}_1 - f_e(G)| \leq (2n \ln \ln n)/\epsilon.$$
2. *If G satisfies α-decay for $\alpha > 1$, $D > \bar{d}$, and $f_e(G) < (n \ln n)/\epsilon$, then with probability at least $1 - 2/\ln n$, Algorithm 1 outputs $\hat{e}_2$ and*
$$|\hat{e}_2 - f_e(G)| = O\left(\frac{2D \ln \ln n}{\epsilon} + \frac{\bar{d}^\alpha}{D^{\alpha-1}}\right).$$

The algorithm runs in $O(nf_e(G))$ time.

Using this lemma, and setting $D = n^{1/\alpha}$, we get the following theorem about privately releasing edge counts.

Theorem 4.1 (Releasing Edge Counts Privately). *There is a node differentially private algorithm which, given constants $\alpha > 1$, $\epsilon > 0$, and a graph G on n nodes, computes with probability at least $1 - 2/(\ln n)$ an $(1 \pm o_n(1))$-approximation to $f_e(G)$ (the number of edges in G) if either of the following holds:*

1. *If $f_e(G) \geq (5n \ln n)/\epsilon$.*
2. *If G satisfies α-decay and $f_e(G) = \omega(n^{1/\alpha}(\ln n)^{\alpha+1})$.*

4.3 Extension to Concave Query Functions

The flow-based technique of the previous section can be generalized considerably. In this section, we look at linear queries in the degree distribution in which the function h specifying the query is itself concave, meaning that its increments $h(i+1) - h(i)$ are non-increasing as i goes from 0 to $n-2$. The number of edges in the graph is an example of such a query, since the increments of $h(i) = i/2$ are constant.[6]

For mathematical convenience, we assume that the function h is in fact defined on the real interval $[0, n-1]$ and is increasing and concave on that set (meaning that for all $x, y \in [0, n-1]$, we have $h((x+y)/2) \leq (h(x) + h(y))/2$. It is always possible to extend a (discrete) function on $\{0, \ldots, n-1\}$ with nonincreasing increments to a concave function on $[0, n-1]$ by interpolating linearly between each adjacent pair of values $h(i), h(i+1)$. Note that the maximum of h is preserved by this transformation, and the largest increment $|h(i+1) - h(i)|$ equals the Lipschitz constant of the new function (defined as $\sup_{x,y \in [0,n-1]} \frac{|h(x)-h(y)|}{|x-y|}$).

Given a graph G on at most n nodes, a concave function h on $[0, n-1]$ and a threshold D, we define an optimization problem as follows: construct the flow graph (Definition 4.1) as before, but make the objective to maximize $obj_h(Fl) = \sum_{v \in V} h(Fl(v))$, where $Fl(v)$ is the units of flow passing from s to v_ℓ in the flow Fl. Let $opt_h(G)$ denote the maximum value of the objective function over all feasible flows. The constraints of this optimization problem are all linear.

This new optimization problem is no longer a maximum flow problem (nor even a linear program), but the concavity of h ensures that it still a convex optimization problem and can be solved in polynomial time using convex programming techniques. Note that we need h be to concave only for computational efficiency purposes, and one could define the above flow graph and optimization problem for all h.

Proposition 4.1. *For every increasing function $h : [0, n-1] \to \mathbb{R}$,*

1. *If G is D-bounded, then $opt_h = F_h(G)$ (that is, the value of the optimization problem equals the correct value of the query).*

[6] There is some possible confusion here: any query of the form F_h described in Section 3 is linear *in the degree distribution of the graph*. Our additional requirement here is that the "little" function h be concave *in the degree argument i*.

2. *The optimum opt_h has global sensitivity at most $\|f\|_\infty + D\|f'\|_\infty$ on $\mathcal{G}_n$, where $\|f\|_\infty = \max_{0\le x\le D} h(x)$ and $\|f'\|_\infty$ is the Lipschitz coefficient of h on $[0, D]$ (that is, the global sensitivity of the optimization problem's value is at most the sensitivity of F_h on D-bounded graphs).*
3. *If h is concave then $opt_h(G)$ can be computed to arbitrary accuracy in polynomial (in n) time.*

Thus, as with the number of edges, we can ask a query which matches F_h on D-bounded graphs but whose global sensitivity on the whole space is bounded by its sensitivity of the set of D-bounded graphs.

The MLE for power laws described in Section 3 is an interesting example where Proposition 4.1 could be used. There is a natural concave extension for the power law MLE: set $f(x) = x$ for $0 \le x < 1$ and $f(x) = 1 + \ln(x)$ for $x \ge 1$. The sensitivity of F_f on D-bounded graphs is $\Delta_D f \le 1 + \ln(D) + D$ (this follows from (1)). In graphs with few high-degree nodes of degree greater than D, this leads to a much better private approximation to the power-law MLE in low-degree graphs than suggested in Section 3.

5 LP-based Lipschitz Extensions

In this section, we show how to privately release the number of (not necessarily induced copies) of a specified small template graph H in the input graph G. For example, H can be a triangle, a k-cycle, a length-k path, a k-star (k nodes connected to a single common neighbor), or a k-triangle (k nodes connected to a pair of common neighbors that share an edge). Let $f_H(G)$ denote the number of (not necessarily induced) copies of H in G, where H is a connected graph on k nodes.

5.1 LP-based Function

Definition 5.1 (Function $v_{\text{LP}}(G)$). *Given an (undirected) graph $G = ([n], E)$ and a number $D \in [n]$, consider the following LP. The LP has a variable x_C for every copy of the template graph H in G. Let $\Delta_D f$ denote the global node sensitivity of function f in D-bounded graphs. Then the LP corresponding to G is specified as follows:*

$$\begin{aligned} &\textit{maximize} \sum_{\textit{copies } C \textit{ of } H} x_C \textit{ subject to:} \\ &\quad 0 \le x_C \le 1 \textit{ for all variables } x_C \\ &\quad S_v \le \Delta_D f_H \textit{ for all nodes } v \in [n], \qquad \textit{where } S_v = \sum_{C: v \in V(C)} x_C. \end{aligned}$$

We denote the value that maximizes this linear program by $v_{\text{LP}}(G)$.

When the variable x_C takes values 1 or 0, it signifies the presence or absence of the corresponding copy of H in G. The first type of constraints restricts these variables to $[0, 1]$. The second type of constraints says that every node can participate in at most $\Delta_D f_H$ copies of H. This is the largest number of copies of H in which a node can participate in a D-bounded graph.

Observation 5.1 *$\Delta_D f_H \leq k \cdot D \cdot (D-1)^{k-2}$, where k is the number of nodes in H.*

Lemma 5.1. *The global node sensitivity $\Delta v_{\text{LP}} \leq \Delta_D f_H \leq k \cdot D \cdot (D-1)^{k-2}$.*

Lemma 5.2. *For all graphs G, the value $v_{\text{LP}}(G) \leq f_H(G)$. Moreover, if G is D-bounded then $v_{\text{LP}}(G) = f_H(G)$.*

5.2 Releasing Counts of Small Subgraphs

The LP-based function from the previous section can be used to privately release small subgraph counts. If $f_H(G)$ is relatively large then the Laplace mechanism will give an accurate estimate. Using the LP-based function, we can release $f_H(G)$ accurately when $f_H(G)$ is much smaller, provided that G satisfies α-decay. In this section, we work out the details of the algorithm for the special case when H has 3 nodes, i.e., is the triangle or the 2-star, but the underlying ideas apply even when H is some other small subgraph.

Algorithm 2 ϵ-Node-Private Algorithm for Releasing Subgraph Count $f_H(G)$

Input: parameters ϵ, D, n, template graph H on 3 nodes, and graph G on n nodes.

1: Let $\hat{f}_1 = f_H(G) + \text{Lap}(\frac{6n^2}{\epsilon})$ and threshold $\zeta = \frac{n^2 \ln n}{\epsilon}$.
2: If $\hat{f}_1 \geq 7\zeta$, **return** $\hat{f}_1$.
3: Compute the value $v_{\text{LP}}(G)$ given in Definition 5.1 using D.
4: **return** $\hat{f}_2 = v_{\text{LP}}(G) + \text{Lap}(\frac{6D^2}{\epsilon})$.

Lemma 5.3. *Algorithm 2 is an ϵ-node-private polynomial time algorithm that takes a graph G, parameters ϵ, D, n, and a connected template graph H on 3 nodes, and outputs an approximate count for $f_H(G)$ (the number of copies of H in G).*

1. *If $f_H(G) \geq (13n^2 \ln n)/\epsilon$, then with probability at least $1 - 1/\ln n$, Algorithm 2 outputs $\hat{f}_1$ and*
$$\left|\hat{f}_1 - f_H(G)\right| \leq (6n^2 \ln \ln n)/\epsilon.$$
2. *If G satisfies α-decay for $\alpha > 1$, $D > \bar{d}$, and $f_H(G) < (n^2 \ln n)/\epsilon$, then with probability at least $1 - 2/\ln n$, Algorithm 2 outputs $\hat{f}_2$ and*
$$|\hat{f}_2 - f_H(G)| \leq \frac{6D^2 \ln \ln n}{\epsilon} + t_h,$$
$$\text{where } t_h = \begin{cases} O\left(\bar{d}^\alpha n \cdot D^{2-\alpha}\right) & \text{if } \alpha > 2, \\ O\left(\bar{d}^\alpha n \cdot \ln n\right) & \text{if } \alpha = 2, \\ O\left(\bar{d}^\alpha n \cdot n^{2-\alpha}\right) & \text{if } 1 < \alpha < 2. \end{cases}$$

Lemma 5.4. *If H has 3 nodes and G satisfies α-decay for $\alpha > 1$ and $D \geq \bar{d}$ then $v_{\text{LP}}(G) \geq f_H(G) - t_h$, where $t_h = O\left(\bar{d}^\alpha n D^{2-\alpha}\right)$ if $\alpha > 2$, $t_h = O\left(\bar{d}^2 n \ln n\right)$ if $\alpha = 2$, and $t_h = O\left(\bar{d}^\alpha n n^{2-\alpha}\right)$ if $1 < \alpha < 2$.*

Using Lemmas 5.3 and 5.4 with a carefully chosen threshold degree D, we get the following theorem about privately releasing counts of subgraphs on 3 nodes. A private value of $\bar{d}$ can be obtained using Theorem 4.1.

Theorem 5.2 (Releasing Subgraph Counts Privately). *There is a node differentially private algorithm which, given constants $\alpha > 1$, $\epsilon > 0$, a connected template graph H on 3 nodes, and a graph G on n nodes, computes with probability at least $1 - 2/(\ln n)$ an $(1 \pm o_n(1))$-approximation to $f_H(G)$ (the number of copies of H in G) if either of the following holds:*

1. *If $f_H(G) \geq (13n^2 \ln n)/\epsilon$.*
2. *If G satisfies α-decay, has average degree at most $\bar{d} > 1$, and either of the following holds: (a) $f_H(G) = \omega(\bar{d}^2 n^{2/\alpha} \ln n)$ if $\alpha > 2$, (b) $f_H(G) = \omega(\bar{d} n \ln^2 n)$ if $\alpha = 2$, or (c) $f_H(G) = \omega(\bar{d}^\alpha n^{3-\alpha} \ln n)$ if $1 < \alpha < 2$.*

6 Generic Reduction to Node Privacy in Bounded-Degree Graphs

We now turn to another, more general approach to getting more the accurate queries by looking at bounded degree graphs. Recall that if we had a promise that all degrees were at most D, then for many natural queries we could add less noise and still satisfy differential privacy. The question is, how can we enforce such a promise? Given an input graph G, possibly of large maximum degree, it is tempting to simply answer all queries with respect to a "truncated" version $T(G)$, in which nodes of very large degree have been removed. This is delicate, however, since the truncated graph $T(G)$ may change a lot when a single node of G is changed. That is, it could be that the *local sensitivity* of the "truncation" operator (viewed as a map from $\mathcal{G}_n$ to $\mathcal{G}_{n,D}$) is very high, making queries on the truncated graph also high-sensitivity.

More generally, consider a projection operator $T : \mathcal{G}_n \to \mathcal{G}_{n,D}$ which takes an arbitrary graph and outputs a D-bounded graph. We may define the (local, global, smooth) sensitivity of T in terms of the node distance $d_{\text{node}}(T(G_1), T(G_2))$ where G_1 and G_2 differ in one node.

Given a query f defined on D-bounded graphs, it is easy to see that the local sensitivity of a *composed* query $f \circ T$ is bounded by the product $LS_T(G) \cdot \Delta_D f$ (one can see this as a discrete analogue of the chain rule from calculus). Our main lemma is that we can bound the smooth sensitivity similarly. We use the definition of β-smooth upper bound on local sensitivity from 2.3.

Lemma 6.1 (Smooth Bounds on Composed Functions). *Let $T : \mathcal{G}_n \to \mathcal{G}_{n,D}$. If $S_T(G)$ is a β-smooth upper bound on the local sensitivity of T (measured w.r.t. node distance), then $S_{f \circ T}(G) = S_T(G) \cdot \Delta_D F$ is a β-smooth bound on the local sensitivity of $f \circ T$.*

Given a smooth upper bound on the local sensitivity of $F_f \circ T$, we can use Theorem 2.3 to obtain a private algorithm for releasing F_f on all graphs in $\mathcal{G}_n$.

Instead of using smooth sensitivity, we can also use a differentially private upper bound on the local sensitivity, inspired by Dwork and Lei [15] and Karwa *et al.* [7]. This give a general technique to transform any algorithm that is private on D-bounded graphs to one which is private for all graphs.

Lemma 6.2 (Generic Reduction [7]). *Let $T : \mathcal{G}_n \to \mathcal{G}_{n,D}$. Suppose L_ϵ is an (ϵ, δ_1)-differentially private algorithm (on all graphs in $\mathcal{G}_n$) that outputs a real value such that $\Pr[L_\epsilon(G) > LS_T(G)] \geq 1 - \delta_2$ (where LS_T is measured w.r.t. node distance).*

Suppose that A is a $(\epsilon, 0)_D$-differentially private algorithm. Then the following algorithm is $(2\epsilon, e^\epsilon\delta_2 + \delta_1)$-differentially private: compute $\hat{L} = L_\epsilon(G)$, then run A on input $T(G)$ with privacy parameter $\epsilon' = \epsilon/\hat{L}$ and finally output the pair $\hat{L}, A(T(G))$.

Naive Truncation. This is the simplest truncation operator. Consider the operator T_{naive} that deletes all nodes of degree greater than D in $G = (V, E)$. This may have high local sensitivity (for example, rewiring one node may change the degrees of many nodes from D to $D+1$, resulting in a drastic increase in the number of nodes deleted by T_{naive}. This projector is computable in $O(n+m)$ time, where $n = |V|$ and $m = |E|$. The following simple lemma analyzes the sensitivity of this truncation operation.

Lemma 6.3. *Given a threshold D, the local sensitivity of naive truncation (w.r.t. node distance) is 1 plus the number of nodes with degree either D or $D+1$.*

The following proposition bounds the local and smooth sensitivity of naive truncation. The last two parts of this proposition allow us to employ Lemmas 6.1 and 6.2, respectively.

Proposition 6.1 (Bounding the Sensitivity of Naive Truncation). *Given a graph G, let $N_k(G)$ denote the number of nodes in G with degrees in the range $[D-k, D+k+1]$. Let $C_k(G) = 1 + k + N_k(G)$. Then*

1. *$C_0(G)$ is the local sensitivity of naive truncation at G.*
2. *For any graph G' within rewiring distance $k+1$ of G, the local sensitivity of naive truncation between G and G' is at most $C_k(G)$.*
3. *$S_{T_{naive}}(G) = \max_{k\geq 0} e^{-\beta k} C_k(G)$ is a smooth upper bound on the local sensitivity of naive truncation. Moreover, if $N_{\ln n/\beta}(G) \leq \ell$ (that is, if there are ℓ nodes in G with degrees in the range $D \pm \ln n/\beta$), then*

$$S_{T_{naive}}(G) \leq \ell + 1/\beta + 1\,.$$

4. *Consider the tapered interval query given by the function $f_{t,D,D+1}$ (defined in Section 3, Item (2)) for some $t \in (\frac{1}{n}, 1]$. The algorithm that returns*

$$L(G) = 1 + F_{f_{t,D,D+1}}(G) + \frac{2tn\log(1/\delta)}{\epsilon} + \text{Lap}\left(\frac{2tn}{\epsilon}\right)$$

is $(\epsilon, 0)$-node-private and returns a value larger than $LS_{T_{naive}}(G)$ with probability at least $1 - \delta$.

6.1 Using Naive Truncation: Deterministic and Randomized Cutoffs

The smooth sensitivity bound of Proposition 6.1 depends on the number of nodes immediately around the cutoff D. Thus, even if a graph G is D-bounded, truncating exactly at D may lead to a large smooth sensitivity bound. We get a much better bound on the noise by truncating slightly above the maximum degree. The following corollary follows by adding Cauchy noise as per Theorem 2.3.

Corollary 6.1. *For every $\epsilon > 0$, every threshold $D > \sqrt{2}(\ln n)/\epsilon$ and every real-valued function $f : \mathcal{G}_{n,D} \to \mathbb{R}$, there is a ϵ-node-private algorithm that outputs $f(G)$ with median error $O(\Delta_{\hat{D}} f/\epsilon^2)$, where $\hat{D} = D + 2\ln(n)/\epsilon \leq 2D$.*

Randomizing the Degree Threshold One obvious problem with the truncation technique is that we may not know the maximum degree in the graph, or the maximum degree may be very large. Indeed, as have seen in the algorithms for counting subgraphs, it often makes sense to project to a degree threshold well below the maximum degree in a graph. In that case, the smooth sensitivity bound of Proposition 6.1 could be large.

One can get a substantially better bound by *randomizing* the cutoff. Given a target threshold D, consider an algorithm that picks a random threshold in a range of bounded by a constant multiple of D (say, between $2D$ and $3D$). We show that the smooth sensitivity of naive truncation is (likely to be) close to the *average* number of nodes of a random degree in the range, saving a factor of roughly D in the introduced noise.

Lemma 6.4 (Randomized Cutoff Lemma). *Fix $\beta > 0$, a graph G on n nodes, and an integer $D > 0$. Let $P_G(D)$ be the fraction of nodes in G of degree greater than D, and let $\hat{D}$ be uniformly random in the range $\{D + 1 + \ln n/\beta, \ldots, 2D + \ln n/\beta\}$. If T_{naive} is the naive truncation at degree $\hat{D}$, then*

$$\mathop{\mathbb{E}}_{\hat{D}}[S_{T_{naive}}(G)] \leq 3\frac{nP_G(D)}{D} \cdot \frac{\ln n}{\beta} + \frac{1}{\beta} + 1\,.$$

6.2 Application of Naive Truncation for Releasing Degree Distribution

For concreteness, we work out one application of the naive truncation idea to releasing an approximation to the entire degree distribution (rather than releasing specific functions of that distribution). Our goal is to output a vector $\hat{p}$ that minimizes the ℓ_1-error $\|\hat{p} - p_G\|_1$, where p_G is the (true) degree distribution of the graph. If the error is $o(1)$, then $\hat{p}$ provides an estimate with vanishing error for all of the entries of degree distribution.

We use Lemma 6.1 to get a smooth bound on local sensitivity. The global sensitivity $\Delta_{\hat{D}}\|\hat{p} - p_G\|_1 \leq 2\hat{D}$.

Algorithm 3 ϵ-Node-Private Algorithm for Releasing Degree Distributions

Input: parameters ϵ, D, n, and a graph G on n nodes.

1: Pick $\hat{D} \in_R \{D + \frac{\ln n}{\beta} + 1, \ldots, 2D + \frac{\ln n}{\beta}\}$.

2: Compute the naive truncation $T_{\text{naive}}(G)$ with threshold $\hat{D}$ and the smooth bound $S_{T_{\text{naive}}}(G)$ with $\beta = \epsilon/(\sqrt{2}(\hat{D} + 1))$ (as in Proposition 6.1).

3: Output $\hat{p} = p_{T_{\text{naive}}(G)} + \text{Cauchy}\left(\frac{2\sqrt{2}\hat{D}}{\epsilon} S_{T_{\text{naive}}}(G)\right)^{\hat{D}+1}$ (that is, add i.i.d. Cauchy noise with median absolute deviation $\frac{2\sqrt{2}\hat{D}}{\epsilon} S_{T_{\text{naive}}}(G)$ to the entries of the degree distribution of $T_{\text{naive}}(G)$).

Theorem 6.1. *Algorithm 3 is an ϵ-node-private algorithm that takes a graph G and parameters n, D, ϵ, and outputs a vector $\hat{p}$ such that, if G satisfies α-decay for $\alpha > 1$ and $D > \frac{4}{\epsilon} \ln n$ and $D > \bar{d}$ where $\bar{d} = \bar{d}(G)$ is the average degree in G , then with probability at least $1/2$ we have*

$$\|\hat{p} - p_G\|_1 = O\left(\frac{\bar{d}^\alpha \ln n \ln(D)}{\epsilon^2 D^{\alpha-2}} + \frac{D^3 \ln(D)}{n\,\epsilon^2}\right) = \tilde{O}\left(\frac{1}{\epsilon^2}\left(\frac{\bar{d}^\alpha}{D^{\alpha-2}} + \frac{D^3}{n}\right)\right),$$

and the $\tilde{O}$ notation hides constants depending on α and polylogarithmic factors in n.

We note that one can get slightly better bounds on the error by considering an algorithm that uses different noise distributions other than Cauchy. We stick to Cauchy noise here for simplicity. For the following corollary, we set $D = \bar{d}^{\frac{\alpha}{\alpha+1}} n^{\frac{1}{\alpha+1}}$ in the previous theorem.

Corollary 6.2 (Releasing Degree Distribution Privately). *There is a node differentially private algorithm running in $O(|E|)$ time which, given $\alpha > 1$, $\epsilon > 0$, and a graph $G = (V, E)$ on n nodes, computes an approximate degree distribution with ℓ_1 error (with probability at least $1/2$)*

$$\|\hat{p} - p_G\|_1 = \tilde{O}\left(\bar{d}^{\frac{3\alpha}{\alpha+1}} / \left(\epsilon^2 n^{\frac{\alpha-2}{\alpha+1}}\right)\right)$$

if G satisfies α-decay and has average degree at most $\bar{d} > 1$. In particular, this error goes to 0 for any constant $\alpha > 2$ when $\bar{d}$ is polylogarithmic in n.

Acknowledgments

We thank Madhav Jha for pointing out an error in an earlier version of the Randomized Cutoff Lemma.

References

[1] Jernigan, C., Mistree, B.F.T.: Gaydar: Facebook Friendships Expose Sexual Orientation. First Monday **14**(10) (2009)
[2] Dinur, I., Nissim, K.: Revealing Information While Preserving Privacy. In: PODS, ACM (2003) 202–210
[3] Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: TCC. LNCS, Springer (2006) 265–284
[4] Dwork, C.: Differential Privacy. In: ICALP. LNCS, Springer (2006) 1–12
[5] Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-Law Distributions in Empirical Data. SIAM Review **51**(4) (2009) 661–703
[6] Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: Symp. Theory of Computing (STOC), ACM (2007) 75–84 Full paper: `http://www.cse.psu.edu/~asmith/pubs/NRS07`.
[7] Karwa, V., Raskhodnikova, S., Smith, A., Yaroslavtsev, G.: Private analysis of graph structure. PVLDB **4**(11) (2011) 1146–1157

[8] Hay, M., Li, C., Miklau, G., Jensen, D.: Accurate Estimation of the Degree Distribution of Private Networks. In: ICDM. (2009) 169–178
[9] Rastogi, V., Hay, M., Miklau, G., Suciu, D.: Relationship Privacy: Output Perturbation for Queries with Joins. In: PODS. (2009) 107–116
[10] Gehrke, J., Lui, E., Pass, R.: Towards Privacy for Social Networks: A Zero-Knowledge Based Definition of Privacy. In: TCC. (2011) 432–449
[11] Kifer, D., Machanavajjhala, A.: No Free Lunch in Data Privacy. In: SIGMOD. (2011) 193–204
[12] Blocki, J., Blum, A., Datta, A., Sheffet, O.: Differentially Private Data Analysis of Social Networks via Restricted Sensitivity. In: ITCS (To appear). (2013)
[13] Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our Data, Ourselves: Privacy Via Distributed Noise Generation. In: EUROCRYPT. LNCS, Springer (2006) 486–503
[14] McSherry, F., Mironov, I.: Differentially Private Recommender Systems: Building Privacy into the Net. In: KDD, ACM New York, NY, USA (2009) 627–636
[15] Dwork, C., Lei, J.: Differential Privacy and Robust Statistics. In: STOC. (2009) 371–380
[16] Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical Privacy: The SuLQ Framework. In: PODS, ACM (2005) 128–138
[17] Orlin, J.B.: Max flows in $O(nm)$ time, or better (2012) `http://jorlin.scripts.mit.edu/docs/papersfolder/O(nm)MaxFlow.pdf`.

A Comparison to Concurrent Work

Blocki *et al.* [12] provide algorithm for analyzing graph data with node-level differential privacy. They proceed from a similar intuition to ours, developing low-sensitivity projections onto the set of graphs of a given maximum degree. However, the results of the two papers are not directly comparable. This section discusses the differences between the two works.

Specifically, Blocki *et al.*have two main results on node privacy, both of which are incomparable to our corresponding results.

- First, Blocki *et al.*show that for every function $f : \mathcal{G}_{n,D} \to \mathbb{R}$, there exists an extension $g : \mathcal{G}_n \to \mathbb{R}$ that agrees with f on $\mathcal{G}_{n,D}$ and that has global sensitivity $\Delta g = \Delta_D f$. The resulting function need not be computable efficiently.
  In contrast, we give explicit, *efficient* constructions of such extensions for several families of functions (the number of edges, linear functions of the degree distribution defined by concave queries, and subgraph counting queries).
- Second, Blocki *et al.*give a specific projection from arbitrary graphs to graphs of a particular degree $\mu : \mathcal{G}_n \to \mathcal{G}_{n,D}$, along with a smooth upper bound on its local sensitivity. They propose to use this for answering queries which have low node sensitivity on $\mathcal{G}_{n,D}$.
  We give a similar result for a different projection (naive truncation). As in their work, we propose to compose this projection with queries that have low sensitivity when restricted to graphs of bounded degree (Lemma 6.1), though we also observe that more general types of composition are also possible (Lemma 6.2).
  The results for these different projections are similar in that both techniques have low smooth sensitivity (depending only on ϵ) when the input graph has degree less than the input threshold D.

To the best of our understanding, the accuracy results are nevertheless incomparable. The Blocki *et al.*projection has a bicriteria approximation guarantee: on input D and G, their projection function is guaranteed to output a graph of degree at most D such that the distance $d_{\text{node}}(G, \mu(G)) \leq 4d_{\text{node}}(G, \mathcal{G}_{n,D/2})$. (No such guarantee is possible for naive truncation, which may be arbitrarily worse than the optimal projection even onto graphs of degree smaller than D.) Nonetheless, the sensitivity bound for μ can be quite a bit higher than the one we present for naive truncation, resulting in lower noise added for privacy (similarly, there are graphs for which the other projection is less sensitive).
Our approach has a considerable efficiency advantage: the naive truncation procedure we propose runs in $O(n + m)$ time for a graph with n vertices and m edges, whereas the projection of Blocki *et al.*seems to require solving a linear program with $n + \binom{n}{2}$ variables and $\Theta(n^2)$ constraints.

The final accuracy guarantees for our algorithms are stated for graphs that satisfy a mild tail bound on the degree distribution, called α-decay. In contrast, Blocki *et al.*only give accuracy guarantees for graphs with bounded degree.

Finally, Blocki *et al.*also consider *edge* privacy, and give a simple, elegant projection operator that has constant edge sensitivity. There is no analogue of that result in this paper, which focuses on node privacy.