Private Analysis of Graph Structure

Vishesh Karwa Sofya Raskhodnikova Adam Smith Grigory Yaroslavtsev
Computer Science and Engineering Department
Pennsylvania State University
University Park, Pennsylvania 16802
vishesh@psu.edu, {sofya, asmith, grigory}@cse.psu.edu

ABSTRACT

We present efficient algorithms for releasing useful statistics about graph data while providing rigorous privacy guarantees. Our algorithms work on data sets that consist of relationships between individuals, such as social ties or email communication. The algorithms satisfy *edge differential privacy*, which essentially requires that the presence or absence of any particular relationship be hidden.

Our algorithms output approximate answers to *subgraph counting queries*. Given a query graph H, e.g., a triangle, k-star or k-triangle, the goal is to return the number of edge-induced isomorphic copies of H in the input graph. The special case of triangles was considered by Nissim, Raskhodnikova and Smith (STOC 2007), and a more general investigation of arbitrary query graphs was initiated by Rastogi, Hay, Miklau and Suciu (PODS 2009). We extend the approach of [NRS] to a new class of statistics, namely, k-star queries. We also give algorithms for k-triangle queries using a different approach, based on the higher-order local sensitivity. For the specific graph statistics we consider (i.e., k-stars and k-triangles), we significantly improve on the work of [RHMS]: our algorithms satisfy a stronger notion of privacy, which does not rely on the adversary having a particular prior distribution on the data, and add less noise to the answers before releasing them.

We evaluate the accuracy of our algorithms both theoretically and empirically, using a variety of real and synthetic data sets. We give explicit, simple conditions under which these algorithms add a small amount of noise. We also provide the average-case analysis in the Erdős-Rényi-Gilbert $G(n, p)$ random graph model.

Finally, we give hardness results indicating that the approach NRS used for triangles cannot easily be extended to k-triangles (and hence justifying our development of a new algorithmic approach).

1. INTRODUCTION

Data privacy has become a fundamental problem of the modern information infrastructure. Increasing volumes of personal and sensitive data are collected and archived by social networking systems, health networks, financial organizations, search engines, intrusion detection systems, retailers and other enterprises. Many of the resulting databases contain information not only on individuals, but also on relationships between them, *e.g.*, personal contacts, financial transactions, and romantic relationships and electronic communication. The potential social benefits from analyzing these databases are enormous, but access to the information they contain is constrained by privacy concerns.

We study the problem of releasing useful statistics on networks while protecting against disclosure of relationships they contain. All our algorithms for releasing graph statistics satisfy *differential privacy* [4], a notion that provides meaningful privacy in the presence of a strong, realistic adversary. It makes assumptions neither about what kind of attack might be perpetrated against the released statistics nor about what additional information the attacker might possess. It limits the *incremental* information the attacker might learn in addition to what he knew before seeing the released statistics. It guarantees that databases that differ in one entry—in our case, graphs that differ in one edge[1]—induce similar distributions on the statistics released by our (randomized) algorithms. (See Section 2.)

Our algorithms output approximate answers to *subgraph counting queries*. Given a query graph H, e.g., a triangle, the goal is to return the number of edge-induced (not necessarily vertex-induced) isomorphic copies of H in the input graph. We exclude automorphisms[2], e.g., in a complete graph on 3 vertices, our triangle count is 1, not 3!.

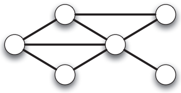
In addition to triangles, we consider k-triangles and k-stars. A k-triangle consists of k triangles, all of which share a common edge. A k-star consists of a central vertex connected to k other vertices. Note that we count a copy of a subgraph, even if it is not induced. For instance, each node of degree $d \geq k$ contributes $\binom{d}{k}$ to the k-star count. See Figure 1 for examples. The number of triangles, k-stars and k-triangles in the input graph G is denoted by $f_{\triangle}(G)$, $f_{k\star}(G)$ and $f_{k\triangle}(G)$, respectively.

Proceedings of the VLDB Endowment, Vol. 4, No. 11
Copyright 2011 VLDB Endowment 2150-8097/11/08... $ 10.00.

[1] We call this privacy guarantee *edge privacy*, to distinguish it from *node privacy* (see [5]), which protects each node together with all its adjacent edges instead of protecting each edge. It is open whether any nontrivial graph statistics can be released with node differential privacy—a qualitatively stronger privacy guarantee than edge differential privacy.

[2] Including automorphisms would simply change the count by a factor independent of the input graph, namely, the number of automorphisms of the query graph.

An example graph G



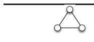
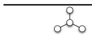
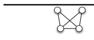
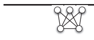
Subgraph	Name & Abbrv.	Count in G
	Triangle $\triangle$	3
	2-star $2\star$	18
	3-star $3\star$	12
	2-triangle $2\triangle$	2
	3-triangle $3\triangle$	0

Figure 1: Example subgraphs and counts.

Importance of subgraph counts. Analysis of social networks is a rapidly growing field, and new models of network data are constantly being proposed. Subgraph counts play a prominent role in many of these models. For example, the subgraph counts $f_{2\star}$, $f_{3\star}$, and $f_{\triangle}$, as well as the number of edges, f_{edge}, are sufficient statistics for several popular exponential random graph models (ERGM) (e.g., [11, 12, 6]). Moreover, many descriptive statistics of graphs are functions of subgraph counts (*e.g.*, the *clustering coefficient* is the ratio $3f_{\triangle}/f_{2\star}$).

Previous Work. Dwork *et al.* [4] showed that, to ensure differential privacy when releasing a query function f of a data set, it suffices to perturb the value of f with random noise of magnitude proportional to the *global sensitivity* of f—the maximum amount by which changing one database entry (in our case, an edge) can change the query answer. (See Section 2.1.) This implies that one can release the number of edges in a graph with constant additive perturbation, since adding or removing an edge alters the number of edges by 1. However, the counts of subgraphs with more than one edge have high global sensitivity: there are contrived graphs on which the query answer changes tremendously if a single edge is added.

Nissim *et al.* [9] ("NRS") introduced the idea of instance-dependent noise. They defined the *local sensitivity* of a query at a particular data set (in our case, a graph G) to be the amount by which the query answer can change if an edge is added to or removed from G. Unfortunately, they showed that adding noise proportional to the local sensitivity is *not*, in general, differentially private. However, they proved that one can instead use the *smooth sensitivity* of the query, which upper bounds the local sensitivity and which is very close to the local sensitivity as long as the query function varies smoothly in a neighborhood of the input graph. (See Definition 2.5.) NRS gave algorithms for computing the smooth sensitivity of statistics in a variety of domains. In the context of subgraph counts, they showed how to efficiently compute the smooth sensitivity of the number of triangles. (Details appear in the online full version [9].) However, they left open whether one can compute this quantity efficiently for other subgraph counts.

Rastogi *et al.* [10] ("RHMS") considered releasing general subgraph counts. They studied a relaxed version of edge differential privacy, called *(edge) adversarial privacy*, which considers a Bayesian attacker; the presence or absence of any given edge is concealed as long as the attacker's prior distribution on the graph comes from a specified family of distributions. Rastogi *et al.* gave a general algorithm for releasing the count of any specified subgraph assuming, roughly, that the adversary's prior admits mainly negative correlations between edges (i.e., the presence of a set of edges does not make other edges more likely to be present). Their algorithm works by first computing a high-probability upper bound on the local sensitivity (before looking at the data) and then adding noise proportional to that bound.

RHMS are able to release vastly more general graph statistics than NRS (who only deal with triangles). However, the RHMS results suffer from two limitations. First, as we show in this paper, the NRS method is much more accurate on the specific problems to which it applies. Second, assumptions about an attacker's prior limit the applicability of a privacy definition. Social networks generally have positive correlations between edges (*e.g.*, people who share a friend are more likely than random to be friends). Thus, considering attackers who think of correlations as mainly negative limits the settings in which the privacy definition makes sense. Our results can be seen as extending the basic approach of NRS (along with its advantages of better accuracy and stronger privacy guarantees) to a much wider class of graph statistics.

Finally, Hay *et al.* [5] gave a differentially private algorithm for releasing an approximation to the degree distribution of a graph. Their algorithm is a clever combination of the global sensitivity approach of [4] with post-processing of the released output to remove some of the added noise. The number of k-stars in a graph can be expressed as a function of the degree distribution, and so the algorithm of [5] implies an algorithm for releasing the number of k-stars. In terms of error, the resulting algorithm is incomparable to our algorithm for releasing k-stars. However, our algorithm is never worse by more than a constant factor, and on some graphs is much more accurate (by a factor of $\Omega(\sqrt{n})$). A detailed comparison is deferred to the full version of this paper.

1.1 Our Contributions

We give new algorithms for releasing subgraph counts differentially privately and evaluate them and the NRS algorithm for releasing triangle counts both theoretically and empirically. Our algorithms apply to two key families of subgraphs: k-stars and k-triangles. We significantly improve on the work of RHMS [10] for these subgraphs: our algorithms satisfy a stronger notion of privacy, which does not rely on the adversary having a particular prior distribution on the data, and give more accurate answers.

Algorithms. We give two new algorithms for releasing graph counts with instance-dependent noise.

$\triangleright$ We extend the approach of NRS to k-star queries. Specifically, we give an algorithm for computing the *smooth sensitivity* of the number of k-stars, denoted $S^*_{k\star,\beta}(G)$, on a given input graph. Our algorithm runs in time $O(n \log n + m)$, where n is the number of nodes and m is the number of edges in the input. Following the general framework of NRS, one can release the number of k-stars after adding random noise (from, e.g., the Gaussian, Laplace or Cauchy distribution) with expected magnitude proportional to $S^*_{k\star,\beta}(G)$.

$\triangleright$ We use a different approach, based on the higher-order local sensitivity, to give a differentially private algorithm for releasing k-triangle counts. Our approach is inspired by the "propose-test-release" framework of Dwork and Lei [3]. Specifically, we develop a two-phase algorithm. First, it finds a differentially-private estimate of the *local sensitivity* of the query function on the input graph G. Second, it releases the query answer plus random noise with expected magnitude proportional to this estimate. Our algorithm runs in time $O(md)$ in graphs of degree at most d.

Computational Hardness. We show that computing the smooth sensitivity is NP-hard for two subgraph counts: 2-triangles and cycles of length 4. Thus, the approach we used for releasing k-stars cannot be extended directly to k-triangles unless $P = NP$.

Evaluation. Any differentially private (or adversarially private) algorithm must be randomized and cannot always return an exact answer to the query. Evaluating the usefulness of a differentially private algorithm therefore requires understanding how much distortion it introduces. The NRS triangles algorithm and our two algorithms add instance-dependent noise to the value of the query before releasing it and satisfy differential privacy; we henceforth refer to all three algorithms in this group as *instance-dependent*.

The instance-dependent and RHMS algorithms aim, in different ways, to add noise close in magnitude to the *local sensitivity* of a query. We therefore seek to understand both the difference between a given algorithm's noise bounds and the local sensitivity and also the overall relative error (the ratio of noise magnitude to the value of the query function). The instance-dependent algorithms are especially hard to evaluate since their worst-case accuracy differs sharply from their accuracy on "typical" inputs. NRS provided no evaluation of the accuracy of their triangle algorithm. One of our contributions is to evaluate their algorithm.

▷ We give explicit, simple conditions under which the instance-dependent algorithms add noise proportional to the local sensitivity of the input. These conditions provide a sense of the inputs for which the algorithms are most useful.

▷ We analyze the accuracy of the instance-dependent algorithms in the Erdős-Rényi-Gilbert $G(n, p)$ model, in which each edge in a graph appears independently with probability p. For moderately dense graphs, that is, for distributions where $p = \omega(1/\sqrt{n})$, we show that the instance-dependent algorithms are "useful" in that they have vanishing *relative* error (roughly $1/n^2p$) with high probability. For triangles and two-triangles, we extend this result to much sparser graphs (where $np \geq \log^2 n$).

▷ We analyze and compare the instance-dependent and RHMS algorithms empirically, using synthetic graphs and real data sets. RHMS provided no empirical evaluation. We consider four subgraphs: 2- and 3-stars, triangles, and 2-triangles.

On both real and synthetic data, we found that the instance-dependent algorithms have noise magnitude close to their "target", the local sensitivity. Moreover, the local sensitivity of 2- and 3-stars (and hence the added noise) was always small relative to the query answer. Thus, the instance-dependent algorithms for 2- and 3-stars seem broadly applicable.

For triangles and 2-triangles, the picture was more nuanced. In relatively dense graphs, we found the local sensitivity was low relative to the query answer, making the algorithms useful. In very sparse graphs, however, the local sensitivity was sometimes higher than the query answer (the phenomenon was less acute for triangles than for 2-triangles). It means that any algorithm which adds distortion on the order of the local sensitivity (and this is necessary for differential or adversarial privacy) basically erases the query answer. Of course, it also means that in such graphs the counts of triangles and (especially) 2-triangles are highly sensitive to small perturbations in the input, making analyses based on these counts non-robust.

Evaluating the RHMS algorithm is delicate since their privacy guarantee makes sense only when the adversary's prior distribution on graphs has expected average degree approximately $\log n$ and (mostly) nonpositive correlations between edges. For comparisons we therefore used graphs drawn from Erdős-Rényi-Gilbert distributions with $p = \log(n)/n$ (which would be valid prior distributions) and various values of n. For all four subgraphs, we find that the instance-dependent algorithms perform better than the RHMS algorithm. For 2-stars, both algorithms have low relative error. For the three other subgraphs, however, the RHMS noise magnitude dwarfs the actual statistic. Thus, the instance-based algorithms are preferable to (the current formulation of) the RHMS algorithm: they satisfy a stricter notion of privacy and perform better or as well as the RHMS algorithm for all subgraph queries to which they apply.

1.2 Discussion

Our results, together with those of [9, 10], raise several interesting questions beyond those mentioned so far. First, the techniques developed here proceed one statistic at a time. Because the amount of distortion increases as more statistics are released (see Lemma 2.1), these techniques are not appropriate for releasing a large number of statistics simultaneously (such as the edgewise shared partner distribution used by [6]).

Hay *et al.* [5] discuss algorithms for releasing the degree distribution, but their techniques do not apply to more complicated vectors of statistics. Releasing synthetic data that reflects many different statistics is, potentially, an even harder task. As noted in "Previous Work", above, the algorithm of [5] implies an algorithm for releasing the number of k stars whose performance is incomparable to that of our algorithms. See the full version for a detailed comparison.

Along different lines, one might ask for a qualitatively stronger privacy guarantee, such as *node privacy* (discussed in Footnote 1). Our algorithms do not apply directly to node privacy, but the general technique developed for our k-triangles algorithm offers some promise.

Other Related Work. The general topic of privacy in social networks has been studied extensively, and we do not attempt to summarize the literature here. See [13] for a survey of techniques for anonymizing social networks that work along the lines of k-anonymity (and do not carry the strong semantics of differential privacy). Attacks on anonymized social network data are studied, for example, in [1, 8].

1.3 Organization of This Paper

After a brief introduction to differential privacy and noise addition, we present our algorithms (Sections 3 and 4) and our experimental results (Section 5). Results on the hardness and average-case analysis of our algorithms are deferred to Appendices A and B, respectively.

2. PRELIMINARIES

In this work, a statistical database is a graph G on n vertices, representing relationships between n individuals. The database is held by a trusted curator who answers users' queries about the database. A user query is a function $f : \mathcal{G}_n \to \mathbb{R}$ to be evaluated on the database G, where $\mathcal{G}_n$ is a set of all n-vertex (undirected) graphs. In response to each query f, the curator runs an algorithm $\mathcal{A}_f$ on the database G and sends back $\mathcal{A}_f(G)$. For example, $\mathcal{A}_f(G)$ could be $f(G)$ with random noise added according to some agreed upon distribution or a pair of values, specifying an approximation to $f(G)$ and the magnitude of the added noise. Our goal is to make $\mathcal{A}_f(G)$ (or, in the second example, the first component output by $\mathcal{A}_f(G)$) as close to $f(G)$ as possible, thus enabling the users to learn their target value as accurately as possible, while preserving the privacy of individuals whose information is stored in the database.

Definition 2.1. *The distance between n-vertex graphs G and G', denoted $d(G, G')$, is the number of edges on which they differ. Graphs G and G' are* neighbors *if $d(G, G') = 1$.*

A randomized algorithm is private if neighboring databases induce nearby distributions on its outcomes:

Definition 2.2. (Differential Privacy, [4, 2]) *Let ϵ and δ be small constants. A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for all neighboring databases G, G', and for all sets $\mathcal{S}$ of possible outputs, $\Pr[\mathcal{A}(G) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{A}(G') \in \mathcal{S}] + \delta$. The probability is taken over the random coins of $\mathcal{A}$. When $\delta = 0$, the algorithm is ϵ-differentially private.*

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 [7, 3]) *If an algorithm $\mathcal{A}$ runs t randomized algorithms $\mathcal{A}_1, ..., \mathcal{A}_t$, each of which is (ϵ, δ)-differentially private, and applies an arbitrary randomized algorithm g to their results (that is, $\mathcal{A}(G) = g(\mathcal{A}_1(G), \mathcal{A}_2(G), ..., \mathcal{A}_t(G))$), then $\mathcal{A}$ is $(t\epsilon, t\delta)$-differentially private. This holds even if for each $i > 1$, $\mathcal{A}_i$ is selected adaptively, based on $\mathcal{A}_1(G), \ldots, \mathcal{A}_{i-1}(G)$.*

2.1 Calibrating Noise to Sensitivity

Output Perturbation. One 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 $h(z) = \frac{1}{2\lambda} e^{-|z|/\lambda}$. We denote it by $\mathrm{Lap}(\lambda)$. A *Cauchy* random variable with median 0 and median absolute value λ has density $h(z) = \frac{1}{\lambda\pi(1+(z/\lambda)^2)}$. We denote it by $\mathrm{Cauchy}(\lambda)$.

Global Sensitivity. In the most basic framework for achieving differential privacy, Laplace noise is scaled according to the *global sensitivity* of the desired statistic f.

Definition 2.3. (Global Sensitivity, [4]) *The* global sensitivity *of a function $f : \mathcal{G}_n \to \mathbb{R}$ is:*

$$GS_f = \max_{G, G' \, neighbors} |f(G) - f(G')| .$$

Theorem 2.2 (Laplace Mechanism, [4]). *The algorithm $\mathcal{A}(G) = f(G) + \mathrm{Lap}(GS_f/\epsilon)$ is ϵ-differentially private.*

Local Sensitivity. The magnitude of noise added by the Laplace mechanism depends on GS_f and the privacy parameter ϵ, but not on the database G. For all functions considered in this paper, this approach yields high noise, not reflecting the function's typical insensitivity to individual inputs. NRS [9] proposed a local measure of sensitivity:

Definition 2.4. (Local Sensitivity, [9]) *For a function $f : \mathcal{G}_n \to \mathbb{R}$ and a graph $G \in \mathcal{G}_n$, the local sensitivity of f at G is $LS_f(G) = \max_{G'} |f(G) - f(G')|$, where the maximum is taken over all neighbors G' of G.*

Note that, by Definitions 2.3 and 2.4, $GS_f = \max_G LS_f(G)$. One may think of the local sensitivity as a discrete analogue of the magnitude of the gradient of f.

An algorithm that releases f with noise magnitude proportional to $LS_f(G)$ on input G is not, in general, differentially private [9], since the noise magnitude can leak information. The question becomes: when can one release an approximation to $f(G)$ whose error is close to $LS_f(G)$?

Smooth Sensitivity. NRS 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 tightest such bound is called the *smooth sensitivity* of f. Roughly, the smooth sensitivity is the maximum local sensitivity attained among graphs "near" to G. More precisely, it is the maximum over all possible graphs G' of a "scaled down" local sensitivity, in which the scaling factor shrinks exponentially with the distance $d(G, G')$ (the number of edges on which G and G' differ). The level of smoothness is parametrized by a number β (where smaller numbers lead to a smoother bound) that is usually comparable to ϵ.

Definition 2.5 ([9]). *The* β-smooth sensitivity *of f at G is $S^*_{f,\beta}(G) = \max_{G' \in \mathcal{G}_n} \left(LS_f(G') \cdot e^{-\beta d(G,G')} \right)$.*

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

Theorem 2.3. ([9]) *Let $f : \mathcal{G}_n \to \mathbb{R}$ be a real-valued function and let $S^*_{f,\beta}$ be its β-smooth sensitivity. If $\beta \leq \frac{\epsilon}{6}$, the algorithm $\mathcal{A}(G) = f(G) + \mathrm{Cauchy}(6S^*_{f,\beta}(G)/\epsilon)$ is ϵ-differentially private.*

To compute smooth sensitivity efficiently, one can break down the expression defining it 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:

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

Lemma 2.4. (Computing smooth sensitivity, [9]) *The smooth sensitivity can be expressed in terms of $LS^{(t)}$:*

$$S^*_{f,\beta}(G) = \max_{t=0,1,\ldots,\binom{n}{2}} e^{-t\beta} LS^{(t)}(G) .$$

We can further break down the expression for smooth sensitivity by separately considering graphs G' that differ from G on a particular edge. Specifically, the *local sensitivity of*

f over an edge (i,j) is $LS_{ij}(G) = |f(G) - f(G')|$, where G' is obtained from G by flipping (adding or deleting) the edge (i,j). Then $LS_f(G) = \max_{i \neq j} LS_{ij}(G)$. The *local sensitivity of f over an edge (i,j) at distance t* is defined analogously:

$$LS_{ij}^{(t)}(G) = \max_{G' \in \mathcal{G}_n:\ d(G,G') \leq t} LS_{ij}(G').$$

2.2 Graph Statistics We Consider

Our algorithms output approximate answers to *subgraph counting queries*. Given a query graph H, e.g., a triangle, k-star or k-triangle, the goal is to return an approximation to the number of edge-induced isomorphic copies of H in the data set graph G. The number of triangles in a graph G is denoted by $f_\triangle(G)$, and the corresponding local sensitivity and smooth sensitivity functions (see Definitions 2.4 and 2.5) are denoted by $LS_\triangle(G)$ and $S^*_{\triangle,\beta}(G)$, respectively.

Similarly, we denote the number of edge-induced subgraphs of G isomorphic to a k-star by $f_{k\star}(G)$, and the number of edge-induced subgraphs of G isomorphic to a k-triangle by $f_{k\triangle}(G)$. As in the notation for sensitivity functions of $f_\triangle$, we omit f from the names of sensitivity functions of $f_{k\star}$ and $f_{k\triangle}$: e.g., $LS_{k\star}(G)$ and $LS_{k\triangle}(G)$.

Our algorithms rely on the following simple statistics about the neighborhood of individual edges. Below, $[n]$ denotes the set $\{1, 2, \ldots, n\}$ and $[a,b]$ denotes $\{a, a+1, \ldots, b\}$.

Definition 2.6. (Edge statistics, [9]) *Consider an undirected graph on n nodes, represented by a (symmetric) adjacency matrix $X = (x_{ij})$, where $x_{ii} = 0$ for all $i \in [n]$. Let a_{ij} denote the number of common neighbors shared by a particular pair of vertices i,j, that is, $a_{ij} = \sum_{\ell \in [n]} x_{i\ell} \cdot x_{\ell j}$. Let b_{ij} denote the number of vertices connected to exactly one of the two vertices i,j, that is, $b_{ij} = \sum_{\ell \in [n]} x_{i\ell}$ XOR $x_{\ell j}$.*

If an edge (i,j) is present, then a_{ij} denotes the number of triangles involving that edge. One can think of b_{ij} as the number of "half-built" triangles involving the edge (i,j), since adding one more edge completes a triangle.

Given the adjacency matrix X, we can compute the matrices of values a_{ij} and b_{ij} in time $O(M(n))$, where $M(n)$ is the time needed to multiply two $n \times n$ matrices, since the matrix with entries a_{ij} is equal to X^2 and $b_{ij} = degree(i) + degree(j) - 2a_{ij} - 2x_{ij}$. In sparse graphs (represented as adjacency lists), one can compute the list of all non-zero values a_{ij} in time $O(m \cdot d_{max})$, where d_{max} is the maximum degree in the graph and m is the number of edges. The values b_{ij} can be then be computed in time $O(n^2)$.

2.3 Asymptotics

We state the performance of our algorithms in terms of the parameters of the input graph. Our asymptotic statements hold for every infinite sequence of graphs of size $n = 1, 2, \ldots$; the asymptotic notation $(O, \Omega, o, \omega, \Theta)$ is defined with respect to n. Other parameters (such as subgraph counts or ϵ and δ of differential privacy) are implicitly functions of n.

3. COMPUTING SMOOTH SENSITIVITY

In this section, we present and analyze algorithms that use the smooth sensitivity framework of [9] to release graph statistics. We state efficient algorithms for computing the smooth sensitivity of the triangle count, $f_\triangle(G)$, and the k-star count, $f_{k\star}(G)$. These algorithms can be used, in conjunction with Theorem 2.3, to obtain efficient differentially private algorithms for releasing these statistics. We also formulate explicit conditions that imply that the local sensitivity of the count is equal to its smooth sensitivity. Under these conditions, the resulting differentially private algorithms add noise proportional to the local sensitivity of the graph counts.

3.1 Triangles

In Theorem 3.1, we state known facts, from the full version of [9], about computing the local sensitivity and the smooth sensitivity of $f_\triangle$. Theorem 3.2 (proved in Appendix C.1) highlights when these quantities are equal.

Theorem 3.1. (Full version of [9]) *The local sensitivity of $f_\triangle$ is $LS_\triangle(G) = \max_{i,j \in [n]} a_{ij}$. The local sensitivity of $f_\triangle$ at distance t is $LS_\triangle^{(t)}(G) = \max_{i \neq j; i,j \in [n]} c_{ij}(t)$, where $c_{ij}(t) = \min\left(a_{ij} + \left\lfloor \frac{t + \min(t, b_{ij})}{2} \right\rfloor,\ n-2\right)$. The β-smooth sensitivity of $f_\triangle$ is computable in time $O(M(n))$.*

We show that the β-smooth sensitivity of $f_\triangle$ is at most the maximum of $1/\beta$ and the local sensitivity of $f_\triangle$.

Theorem 3.2. *For a given graph G, if $LS_\triangle(G) \geq \frac{1}{\beta}$, then $S^*_{\triangle,\beta}(G) = LS_\triangle(G)$.*

3.2 k-Stars

In this section, we explain how to compute the local sensitivity and the smooth sensitivity of $f_{k\star}(G)$ and highlight, in Theorem 3.6, when these two quantities are equal.

Recall that $f_{k\star}(G)$, the number of k-stars in G, is equal to $\sum_{i \in [n]} \binom{d_i}{k}$, where d_i is the degree of node i, for all $k \geq 2$. The following lemma is easy to verify.

Lemma 3.3. *The local sensitivity of $f_{k\star}$ is*

$$LS_{k\star}(G) = \max_{i \neq j; i,j \in [n]} \left(\binom{d_i - x_{ij}}{k-1} + \binom{d_j - x_{ij}}{k-1} \right).$$

The following lemma (proved in Appendix C.2) gives a formula for computing the local sensitivity of $f_{k\star}$ at distance t.

Lemma 3.4. *Let $C(a)$ denote $\binom{a}{k-1}$. Let $d_i' = d_i - x_{ij}$, $B_i = n - 2 - d_i'$, and define d_j' and B_j analogously. Then $LS_{k\star}^{(t)}(G) = \max_{(i,j):\ d_i \geq d_j} LS_{ij}^{(t)}(G)$, where $LS_{ij}^{(t)}(G)$ is the local sensitivity of $f_{k\star}$ at distance t over an edge (i,j). If $d_i \geq d_j$ then $LS_{ij}^{(t)}(G)$ is*

$$\begin{cases} C(d_i' + t) + C(d_j') & \text{if } t \leq B_i, \\ C(n-2) + C(d_i' + t - B_j) & \text{if } t \in (B_i, B_i + B_j), \\ 2C(n-2) & \text{if } t \geq B_i + B_j. \end{cases}$$

In Appendix C.2, we use Lemma 3.4 to give an efficient algorithm for computing the smooth sensitivity of $f_{k\star}$. Next, we state its performance.

Theorem 3.5. *Given a graph G with n nodes and m edges, $S^*_{k\star,\beta}(G)$ can be computed in time $O(n \log n + m)$.*

The final theorem of this section, proved in the full version, shows that the local sensitivity and the smooth sensitivity of the k-star count are equal in graphs with moderately large maximum degree (roughly, at least $\frac{k-1}{\beta}$).

THEOREM 3.6. *Let d_{max} be the the largest degree in G. If $d_{max} \geq \max\{k, (k-1)(\frac{1-\beta}{\beta})\}$, then $S^*_{k\star,\beta}(G) = LS_{k\star}(G)$.*

4. BOUNDING LOCAL SENSITIVITY OF LOCAL SENSITIVITY: k-TRIANGLES

The approach of Section 3 does not extend to k-triangles even for $k = 2$, since computing the smooth sensitivity of this statistic is NP-hard (Theorem A.1). In this section, we present a different approach that yields an efficient (ϵ, δ)-differentially private algorithm (Algorithm 1) for releasing the number of k-triangles with a small amount of Laplace noise, together with the magnitude of the added noise.

The main idea behind Algorithm 1 is that one can get differential privacy by adding noise proportional to a *differentially private* upper bound on the local sensitivity (instead of a "smooth" upper bound, as in NRS). We show this in Lemma 4.4. The better a differentially private bound we get on the local sensitivity, the more accurate our final answer. If $LS_{k\triangle}$ were itself an *in*sensitive function, we could release it using the Laplace mechanism (Theorem 2.2) and add a small offset to it to get a (high-probability) upper bound. Unfortunately, $LS_{k\triangle}$ has high global sensitivity. Instead, we consider LS', the local sensitivity of $LS_{k\triangle}$. Just as we may think of LS as a discrete derivative, we may think of LS' as a second order derivative. We show that LS' is a deterministic function of a quantity with global sensitivity 1, which we can release with little noise using the Laplace mechanism. That allows us to compute an accurate and private approximation to LS'. We use this approximation in turn to add noise to $LS_{k\triangle}(G)$ and finally we use the noisy version of $LS_{k\triangle}(G)$ to release $f_{k\triangle}(G)$.

To analyze the privacy and performance of Algorithm 1, we first give closed form expressions for the local sensitivity of the k-triangle count and for a simple upper bound on the second-order sensitivity LS' (Lemmas 4.1 and 4.2). Next, we prove the key privacy lemma, 4.4, which states that adding noise proportional to a differentially private bound on LS is indeed differentially private. This allows us to prove that the algorithm is differentially private (Theorem 4.3). Finally, Theorem 4.5 shows that the algorithm has low relative error for a wide range of inputs. The missing proofs of the results below are collected in Appendix D.

Computing Local and Second-order Sensitivity. Let N_{ij} be the set of common neighbors of vertices i and j in graph G, that is, $N_{ij} = \{\ell \in [n] \mid x_{i\ell} \cdot x_{\ell j} = 1\}$. Using the notation of Definition 2.6, $|N_{ij}| = a_{ij}$.

LEMMA 4.1. *The local sensitivity of $f_{k\triangle}$ is $LS_{k\triangle}(G) = \max\limits_{i,j \in [n]; i \neq j} LS_{ij}(G)$, where $LS_{ij}(G)$ is*

$$\binom{a_{ij}}{k} + \sum_{\ell \in N_{ij}} \left(\binom{a_{i\ell} - x_{ij}}{k-1} + \binom{a_{\ell j} - x_{ij}}{k-1} \right). \qquad (1)$$

LEMMA 4.2. *Let LS' denote the local sensitivity of the local sensitivity function $LS_{k\triangle}(G)$ and let $a_{\max} = \max\limits_{i,j \in [n]; i \neq j} a_{i,j}$. Then*

$$LS'(G) \leq 3\binom{a_{\max}}{k-1} + a_{\max}\binom{a_{\max}}{k-2}. \qquad (2)$$

Our Algorithm and Privacy Analysis. Our algorithm (see Algorithm 1) releases $f_{k\triangle}(G)$ and the amount of Laplace noise that was added to it.

Algorithm 1 (ϵ, δ)-Differentially Private Algorithm for Releasing $f_{k\triangle}(G)$

```
Input: graph $G$, parameters $\epsilon, \delta$ and $k$.
Set $\epsilon' = \epsilon/3$ and $\delta' = \delta/3$; let $a_{\max} = \max_{i,j \in [n]; i \neq j} a_{ij}$.
$\tilde{a}_{\max} = a_{\max} + \mathrm{Lap}(\frac{1}{\epsilon'}) + \frac{\ln(1/\delta')}{\epsilon'}$.
$\widetilde{LS} = LS_{k\triangle}(G) + \mathrm{Lap}(\frac{B(\tilde{a}_{\max})}{\epsilon'}) + \ln(\frac{1}{\delta'}) \cdot \frac{B(\tilde{a}_{\max})}{\epsilon'}$, where
    $B(a) = 3\binom{a}{k-1} + a\binom{a}{k-2}$ is the expression from (2).
**return** $(f_{k\triangle}(G) + \mathrm{Lap}(\frac{\widetilde{LS}}{\epsilon'}), \widetilde{LS})$
```

THEOREM 4.3. *For all $\epsilon \in (0, \frac{3}{2}\ln(\frac{3}{2})]$ and $\delta \in (0, 1)$, Algorithm 1 is (ϵ, δ)-differentially private.*

Our proof of Theorem 4.3 relies on the following lemma, inspired by the "propose-test-release" framework of [3].

LEMMA 4.4. *Let $\mathcal{B}$ be an (ϵ_1, δ_1)-differentially private algorithm, such that $\Pr(\mathcal{B}(x) \geq LS_f(x)) > 1 - \delta_2$ for all x. Consider the algorithm $\mathcal{A}$ that runs $\mathcal{B}(x)$ to obtain an estimate $\widetilde{LS}$ of the local sensitivity, and releases both $\widetilde{LS}$ and a noisy estimate of f,*

$$\mathcal{A}(x) = (\widetilde{LS}, f(x) + \mathrm{Lap}(\widetilde{LS}/\epsilon_2)), \text{ where } \widetilde{LS} = \mathcal{B}(x)$$

and where $\mathrm{Lap}(\lambda)$ is a Laplace random variable with mean 0 and scale parameter λ. Then $\mathcal{A}$ is $(\epsilon_1 + \epsilon_2, \delta_1 + e^{\epsilon_1}\delta_2)$-differentially private.

To apply Lemma 4.4 to Algorithm 1, we argue that the values $\tilde{a}_{\max}$ and $\widetilde{LS}$ it computes are indeed high-probability upper bounds on the quantities they approximate, $a_{\max}$ and $LS_{k\triangle}(G)$. See Appendix D for details.

Error/Time Analysis. The error of our algorithm depends on the specific instance, and it is difficult to give a clean characterization of the inputs on which it performs well. We show, however, that the error closely tracks the local sensitivity of the number of k-triangles as long as some pair of vertices has a reasonable number of common neighbors (significantly larger than $\log(1/\delta)/\epsilon$). Specifically:

THEOREM 4.5. *Let $\tilde{f}_{k\triangle}$ be the approximation to $f_{k\triangle}$ returned by Algorithm 1. Set $\epsilon' = \epsilon/3$ and $a_{\max} = \max_{i \neq j} a_{ij}$. With probability at least $1 - \delta/3$, the scale parameter of the Laplace noise added to $f_{k\triangle}$ in order to get $\tilde{f}_{k\triangle}$ is at most $\left(LS_{f_{k\triangle}}(G)/\epsilon'\right) \cdot (1 + o(1))$, as long as $\frac{\ln(1/\delta)}{\epsilon} = o(a_{\max})$.*

The asymptotic statements in the theorem should be interpreted as in Section 2.3.

Finally, we note that Algorithm 1 can be implemented to run in time $O(m \cdot d_1)$ on graphs with m edges and maximum degree d_1. (See Appendix D.)

5. EXPERIMENTAL EVALUATION

Our experiments have two goals: (1) to compare instance-dependent algorithms (namely, the triangles algorithm of [9] and our algorithms for k-stars and k-triangles) with the algorithm from [10] ("RHMS") and (2) to evaluate the performance of the instance-dependent algorithms on an absolute scale. Along the way, we examine the relationship between the magnitude of noise added and the quantity $\frac{LS}{\epsilon}$. The subgraph counts included in our experiments are the number of triangles, 2-triangles, 2-stars and 3-stars.

We measure the accuracy of the algorithms on a given graph G by the *median absolute error*, that is, the median of the random variable $|A(G) - f(G)|$, where $A(G)$ is the released value and $f(G)$ is the query answer, and by the *median relative error*, which is the median absolute error divided by $f(G)$.

5.1 Algorithms Used in Our Experiments

Instance-Dependent Algorithms. Given a statistic for which one can compute the smooth sensitivity, Theorem 2.3 states that it suffices to add Cauchy noise scaled up by a factor of $\frac{6S^*(G)}{\epsilon}$, where S^* is the $\frac{\epsilon}{6}$-smooth sensitivity of f. The median absolute value of the Cauchy distribution is 1, so the median absolute error of this approach is $\frac{6S^*(G)}{\epsilon}$. We use it for releasing $f_\triangle$, $f_{2\star}$ and $f_{3\star}$.

To release $f_{2\triangle}$, we run Algorithm 1 (from Section 4). We do not have a closed form expression for the median error of this approach, but it is easy to evaluate numerically.

RHMS Algorithm. The RHMS algorithm (see Algorithm 2 in the appendix), roughly speaking, adds noise proportional to $\frac{\lambda}{\epsilon}$, where λ is a high-probability upper bound on the local sensitivity, taken over the adversary's prior distribution (which is assumed to come from a known class). The algorithm's median absolute error is about $\frac{\lambda}{\sqrt{2}\epsilon}$, but is tricky to state precisely. The proof of the following claim is deferred to the full version of this paper.

Claim 5.1. *The median absolute error M of Algorithm 2 satisfies* $-\frac{\lambda}{\sqrt{2}\epsilon}\ln\frac{0.5}{1-\theta} \leq M \leq -\frac{\lambda}{\sqrt{2}\epsilon}\ln\frac{0.5-\theta}{1-\theta}$, *where the parameters θ and λ are those used in Algorithm 2.*

Setting Parameters. The instance-dependent algorithms and the RHMS algorithm satisfy different definitions of privacy. The smooth-sensitivity-based algorithms (for releasing $f_\triangle$, $f_{2\star}$ and $f_{3\star}$) satisfy ϵ-differential privacy. Algorithm 1 for 2-triangles satisfies (ϵ, δ)-differential privacy, a relaxation with similar semantics. One can think of δ as the probability of a significant privacy compromise. In contrast, RHMS provide a different relaxation, (ϵ, γ)-*adversarial privacy for* $(\log(n)/n, \log n)$-*bounded adversaries*, which has a similar interpretation to differential privacy, but requires specific assumptions on the attacker's prior beliefs about the data set (see [10, Theorem 2.6]). When $\gamma = \delta = 0$, the two relaxations coincide with ϵ-differential privacy, but the algorithms are meaningless in that case. When γ and δ are nonzero, the definitions are incomparable (but both get weaker as γ and δ increase). For concreteness, we set $\gamma = \delta = 0.1$ and $\epsilon = 0.5$.

5.2 Graphs Used in Our Experiments

We performed three sets of experiments. In the first two sets, we used synthetic graphs drawn from the Erdős-Rényi-Gilbert model $G(n, p)$, in which each edge in a graph appears independently with probability p.

The first set was used to compare the RHMS algorithm with instance-dependent algorithms, as well as to evaluate the instance-dependent algorithms on sparse graphs. The RHMS algorithm assumes that the attacker's prior distribution on the data has expected average degree at most $\log n$ (see discussion in Section 1.1). In order to test the algorithm on data consistent with the attack model, we set $p = \log n/n$. (The algorithm fares worse on sparser graphs, so we chose the densest distribution consistent with the model.) The number of vertices, n, varied from 100 to 1000 in steps of 100. The results are plotted in Figure 2.

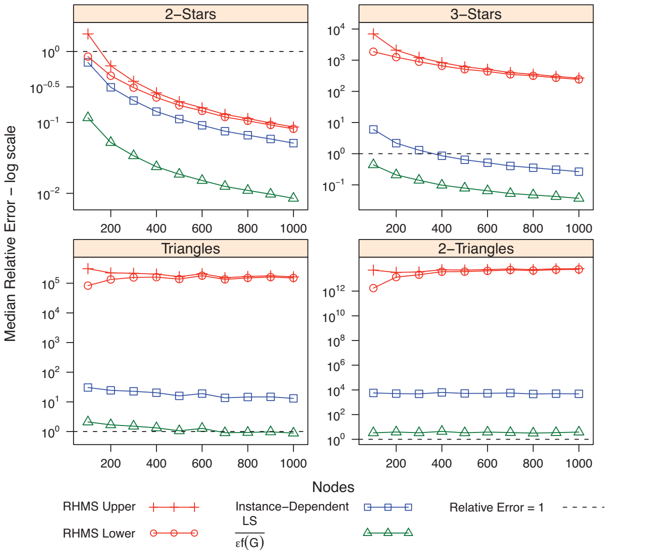


Figure 2: Comparison of instance-dependent algorithms and RHMS for $G(n, p)$ with $p = \frac{\log n}{n}$. Upper and lower bounds for the median relative error of RHMS in red (plus signs/circles), the median relative error of instance-dependent algorithms in blue (squares), and $\frac{LS}{\epsilon f(G)}$ in green (triangles).

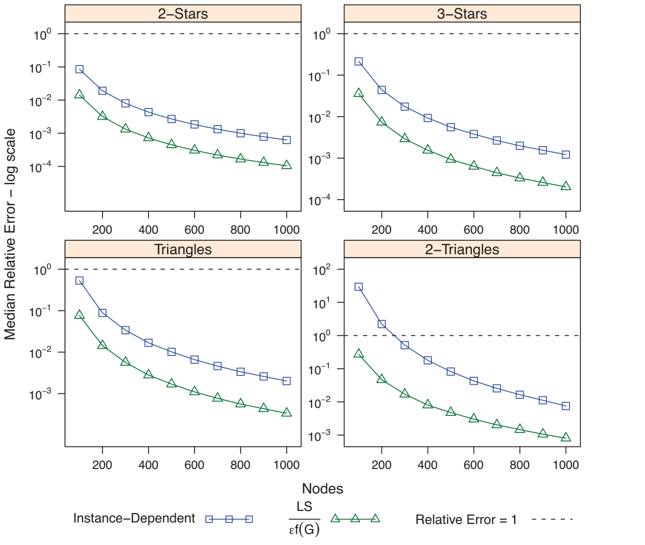


Figure 3: Evaluation of instance-dependent algorithms on $G(n, p)$ graphs for $p = 0.1$. Relative median error in blue (squares), and $\frac{LS}{\epsilon f(G)}$ in green (triangles).

The second set was used to evaluate the instance-dependent algorithms on synthetic graphs for ranges of parameters where the RHMS guarantees do not apply. We generated graphs from the Erdős-Rényi-Gilbert model with

p varying from 0.1 to 0.8 in steps of 0.1 and n between 100 and 1000 in steps of 100. We report on all experiments in the full version. Figure 3 shows the results only for $p = 0.1$, since they are representative of the results for larger values of p.

In the third set, we evaluated the instance-dependent algorithms on 5 collaboration networks "GrQc","HepTh", "CondMat", "HepPh","AstrpPh" and 1 email network, "Enron", obtained from the Stanford Large Network Dataset Collection. The results (and the graphs' parameters) are shown in Figure 4, with additional details in Table 1. We did not include the RHMS algorithm for comparison in the latter two sets because the graphs do not satisfy the average degree condition discussed above.

5.3 Results

Evaluation of Instance-Dependent Algorithms. On both real and synthetic data, we find that the instance-dependent algorithms have noise magnitude close to their "target", the value LS/ϵ. (They were farthest apart for 2-triangles. The gap reflects the fact that the differentially private upper bound on LS used by the algorithm is necessarily loose.) Understanding the accuracy of these algorithms therefore comes down to understanding how sensitive different graph statistics are on various types of graphs.

The local sensitivity of 2- and 3-stars was always small relative to the query answer, except for one setting ($n = 100, p = \frac{\log n}{n}$ for 3-stars), reflected by the fact that the green lines (with triangles) in Figures 2 and 3, representing $\frac{LS}{\epsilon f(G)}$, are lower than 1 and close to 0. These results suggest that the instance-dependent algorithms for 2- and 3-stars are broadly applicable.

For triangles and 2-triangles, the picture was more nuanced. In relatively dense graphs (the second set of experiments, where $p = 0.1$), we found the local sensitivity of triangles was low relative to the query answer. This was also true for 2-triangles, except for $n = 100$. In very sparse synthetic graphs ($p = \log(n)/n$), the local sensitivity was generally higher than the query answer for both triangles and 2-triangles, which means that all of the algorithms considered here are doomed to fare poorly on such data. Finally, in the real data sets (which are fairly sparse), the triangle counts were fairly insensitive on all the data sets considered, and the noise magnitude reflects this fact. The sensitivity of 2-triangle counts, in contrast, varied substantially between data sets (the relative local sensitivity ranged from 1.6% to 25%, see Table 1 in the appendix). The fact that the counts of 2-triangles are sensitive on sparse graphs means that releasing them privately, at least on sparse graphs, requires a significantly different approach. It also means that in sparse graphs, *regardless of privacy*, analyses based on the counts of triangles and 2-triangles may be highly skewed by even small errors in the data.

Comparison of Instance-Dependent Algorithms with RHMS. For all four subgraphs, the instance-dependent algorithms perform better than the RHMS algorithm (see Figure 2).

For 2-stars, both algorithms have low relative error and add noise close to LS/ϵ. For the other three subgraphs, the RHMS noise magnitude dwarfs both the local sensitivity and the query value. In contrast, the noise magnitude of the instance-dependent algorithms remains close to the local sensitivity. (They are still ineffectual for triangles and 2-triangles on such sparse graphs since the local sensitivity is itself quite high.)

The results indicate that with its current formulation, the RHMS approach is of limited use, though perhaps the approach can be modified (say based on tighter probabilistic bounds and more careful characterization of adversarial priors) to add less noise.

Alternatively, it may be possible to extend the approach we use for k-triangles, based on the higher-order local sensitivity, to get (ϵ, δ)-differentially private algorithms for a much broader class of subgraphs, thus combining the generality of the RHMS results with the more robust definitions and performance of our approach.

Acknowledgments

This research was supported in part by National Science Foundation awards CCF-0729171, BCS-941553, and CCF-0747294. We thank Ishan Behoora for contributing initial ideas on the NP-hardness proofs. We are grateful to David Hunter, Dan Kifer, Gerome Miklau, Vibhor Rastogi and a VLDB shepherd for helpful comments.

6. REFERENCES

[1] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In *Proc. 16th Intl. World Wide Web Conference*, pages 181–190, 2007.

[2] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

[3] C. Dwork and J. Lei. Differential privacy and robust statistics. In *Symp. Theory of Computing (STOC)*, pages 371–380, 2009.

[4] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284. Springer, 2006.

[5] 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.

[6] D. Hunter. Curved exponential family models for social networks. *Social Networks*, 29(2):216–230, 2007.

[7] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *Symp. Knowledge Discovery and Datamining (KDD)*, pages 627–636. ACM New York, NY, USA, 2009.

[8] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symp. Security and Privacy*, pages 173–187, 2009.

[9] 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`.

[10] 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.

[11] G. Robins, P. Pattison, Y. Kalish, and D. Lusher. An introduction to exponential random graph (p*) models for social networks. *Social Networks*, 29(2):173–191, 2007.

[12] S. Wasserman and G. Robins. An introduction to random graphs, dependence graphs, and p*. *Models and methods in social network analysis*, pages 148–161, 2005.

[13] B. Zhou, J. Pei, and W. Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explor. Newsl.*, 10:12–22, December 2008.

APPENDIX

Algorithm 2 RHMS Output Perturbation

```
Input: data graph $G$ with $n$ vertices, query graph $H = (V_H, E_H)$, parameters $\gamma$ and $\epsilon$.
1: Set $\lambda = (8(|V_H|+1)|E_H|^2 \log n)^{|E_H|-1}$.
2: With probability $\theta = \frac{|E_H|}{n\gamma}$, return a uniformly random
   element from $\{0, 1, \ldots, n^{|V_H|}\}$.
3: With the remaining probability,
   return $f_H(G) + \mathrm{Lap}(\frac{\lambda}{\sqrt{2}\epsilon})$.
```

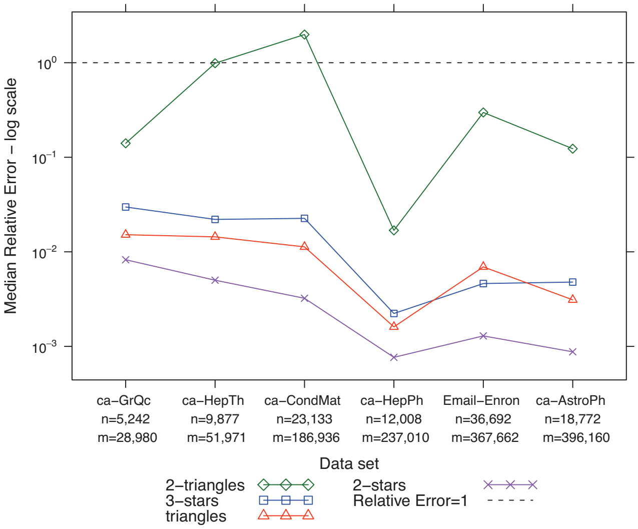


Figure 4: Relative median error of instance-dependent algorithms on several real, sparse data sets. The lines are to help readability; they are not meant to imply a trend. For each data set, n and m denote the number of vertices and edges. For triangles and 2- and 3-stars, $LS(G) = S^*(G)$ for all data sets G, so the median absolute error was $6LS(G)/\epsilon$. For 2-triangles, error and $LS(G)/\epsilon$ are compared in Table 1.

A. COMPUTATIONAL HARDNESS

In this section, we show that computing the smooth sensitivity of some graph statistics, *e.g.,* the number of 2-triangles, is NP-hard. Thus, the NRS approach used here to release the counts of k-stars and triangles does not extend directly to k-triangles even for $k = 2$ (unless $P = NP$).

For query graphs H with 3 or fewer vertices, the smooth sensitivity of f_H can be computed efficiently. (This was shown in Section 3 for 2-stars and triangles. The other connected graph on ≤ 3 vertices consists of a single edge. The local sensitivity of the number of edges is 1 on all input graphs, hence its smooth sensitivity is always 1 for all β.) In contrast, we show that computing the smooth sensitivity is hard for some graphs on four vertices, namely, 4-cycles and 2-triangles. Specifically, for a fixed subgraph H, consider the NP language SMOOTHSENS$_H$ $= \{\langle G, \beta, \nu\rangle \mid S^*_{H,\beta}(G) \geq \nu\}$. Let C_k denote the cycle of length k.

Data set	$\frac{LS_{2\triangle}}{\epsilon f_{2\triangle}}$	$\frac{\text{Error}}{f_{2\triangle}}$	$\frac{\epsilon\cdot\text{Error}}{LS_{2\triangle}}$
ca-GrQc	0.031	0.140	4.47
ca-HepTh	0.163	0.988	6.06
ca-CondMat	0.505	1.987	3.93
ca-HepPh	0.007	0.017	2.58
Email-Enron	0.104	0.298	2.86
ca-AstroPh	0.042	0.123	2.92

Table 1: Further details on the performance of the 2-triangles algorithm on real data sets.

THEOREM A.1. SMOOTHSENS$_{2\triangle}$ *and* SMOOTHSENS$_{C_4}$ *are NP-complete.*

We prove Theorem A.1 in the full version of this paper. Here we present a proof of a weaker statement (Lemma A.2), demonstrating some of the ideas used to prove Theorem A.1. We show that computing $LS^{(t)}_{ij}(G)$ for given distance t, edge (i, j) and graph G is hard. Specifically, let EDGEDISTLS$_H$ $= \{\langle G, t, (i, j), \nu\rangle \mid$ the local sensitivity of f_H at distance t over edge (i, j) on graph G is at most $\nu\}$.

LEMMA A.2. EDGEDISTLS$_{2\triangle}$ *and* EDGEDISTLS$_{C_4}$ *are NP-complete.*

We prove the lemma above separately for 2-triangles and 4-cycles in the next two sections.

A.1 Proof of Lemma A.2 for 2-triangles

We show NP-hardness of computing $LS^{(t)}_{ij}(G)$ by giving a reduction from CLIQUE. Given an instance $\langle G, k\rangle$ of CLIQUE, where G is an undirected graph and k is an integer, we produce a new graph G' by adding two new isolated vertices (numbered, say, 1 and 2) to G. The reduction outputs G' together with $t = 2k$, $(i, j) = (1, 2)$ and $\nu = 5\binom{k}{2}$. It can easily be implemented to run in polynomial time.

The reduction's correctness follows from the next claim. The idea is to show that the local sensitivity at distance $t = 2k$ over the edge $(1, 2)$ would be maximized by connecting vertices 1 and 2 to all the vertices of a k-clique inside of G if one existed. Testing if the local sensitivity over $(1, 2)$ at distance $2k$ achieves its maximum is thus equivalent to testing for a clique in G.

CLAIM A.3. *For every graph G, if G' is obtained by adding isolated vertices $1, 2$ to G, then $LS^{(2k)}_{1,2}(G') \leq 5\binom{k}{2}$. Moreover, equality holds if and only if G contains a k-clique.*

PROOF. Let $G = (V, E)$ and let $G' = (V', E)$ with $V' = V \cup \{1, 2\}$. Consider some graph $G''(V', E'')$ at distance $2k$ from G'.

Recall that $LS_{1,2}(G'')$ is the number of 2-triangles in G'' that would contain $(1, 2)$ if it were present. This number is maximized when the original graph G is complete (that is, a clique of size $|V|$).

Let a be the number of vertices in V connected to vertex 1 but not 2, b be the number of vertices in V connected to both 1 and 2 and c, the number of vertices in V connected to 2 but not 1. By (1) from Section 4, the local sensitivity over $(1, 2)$ is then $\binom{b}{2} + b(a+b-1+c+b-1) = 5\binom{b}{2} + b(a+c)$.

The local sensitivity is maximized by setting $b = k$ and $a = c = 0$, that is, by connecting vertices 1 and 2 to the same set of k vertices in the original graph G. The local sensitivity over $(1, 2)$ is thus at most $5\binom{k}{2}$.

The bound is achieved if 1 and 2 are both connected to a k-clique. Moreover, the proof of the upper bound shows that equality can only be achieved if 1 and 2 are connected to the same set of size k, and all possible edges within that set are present. ∎

A.2 Proof of Lemma A.2 for 4-cycles

This is very similar to the proof of the previous section, except that we give a reduction from the balanced complete bipartite subgraph problem, which we call Biclique. A *k-biclique* is a complete bipartite graph with k vertices in each part. Biclique $= \{\langle G, k\rangle \mid G$ is a bipartite graph that contains a k-biclique$\}$. Given an instance $\langle G, k\rangle$ of Biclique, we construct a graph $G' = (V', E)$ as before, by adding two isolated vertices 1 and 2. The reduction outputs G' together with $t = 2k$, $(i, j) = (1, 2)$ and $\nu = k^2$.

Claim A.4. *For every graph G, if G' is obtained by adding isolated vertices $1, 2$ to G, then $LS^{(2k)}_{1,2}(G') \leq k^2$. Moreover, equality holds if and only if G contains a k-biclique.*

Proof. Let $G = (V, E)$ and let $G' = (V', E)$ with $V' = V \cup \{1, 2\}$. As for 2-triangles, consider some graph $G''(V', E'')$ at distance $2k$ from G' and note that $LS_{1,2}(G'')$ is maximized when G is a clique of size $|V|$.

As before, let a, b, c denote the vertices connected to only vertex 1, both 1 and 2, and only vertex 2, respectively. The number of 4-cycles involving $(1, 2)$ is then $a(b+c)+b(b+c-1) = ac+b(2k-b-1)$. Unlike in the case of 2-triangles, this expression is maximized only when $a = c = k$ and $b = 0$, that is, when 1 and 2 are connected to disjoint sets.

The resulting upper bound of k^2 is achieved when 1 and 2 are each connected to one size of a k-biclique. Moreover, the bound can only be achieved in that case since all edges are necessary to get k^2 cycles. ∎

B. AVERAGE-CASE ANALYSIS

There is currently no single, parsimonious family of probability models that seems to describe the behavior of a broad range of real-world networks. Thus, an average-case analysis of our algorithms is necessarily limited by the choice of probability model. Nevertheless, such analysis provides a crude prediction of how performance depends on the graph parameters which the model incorporates.

In this section, we analyze our algorithms in the Erdős-Rényi-Gilbert $G(n, p)$ model. This provides a simplistic prediction of how our algorithms' accuracy might depend on the density of a network (quantified in the model by the parameter p).

We first show that the relative error of our algorithms is very low in moderately dense random graphs, where p grows faster than $1/\sqrt{n}$.

Theorem B.1. (Moderately Dense Graphs) *Suppose an undirected graph G on n vertices is selected by including each possible edge independently with probability p. If the expected average degree np is $\omega(\sqrt{n}\log n)$ and if ϵ is $o(\log n)$ (as n goes to ∞) then the algorithms we describe for releasing $f_\triangle$ (from [9]),$f_{2\star}, f_{3\star}$ and $f_{2\triangle}$ (from this paper) with high probability have relative error*

$$\frac{|A_H(G) - f_H(G)|}{f_H(G)} = O\left(\frac{1}{\epsilon n^2 p}\right) = O\left(\frac{1}{\epsilon n^{3/2}\log n}\right), \text{ where}$$

f_H is the number of occurrences of H as a subgraph in G and A_H is the corresponding differentially private algorithm.

We defer a detailed proof to the full version. The intuition behind the proof is that two basic phenomena coincide in the regime where $p \gg 1/\sqrt{n}$. First, for all the statistics we consider, the statistic and its local sensitivity are each highly concentrated around their means. For a query graph $H = (V_H, E_H)$, the expected value of the statistic is $\Theta(n^{|V_H|}p^{|E_H|})$, since there are $\Theta(n^{|V_H|})$ possible images for the vertices of H and each one contains a copy of H with probability $p^{|E_H|}$. Similarly, the expectation of the local sensitivity $LS_{ij}(G)$ for any given vertex pair (i, j) is $\Theta(n^{|V_H|-2}p^{|E_H|-1})$, since fixing the edge (i, j) effectively "pins down" two vertices and one edge from H. The fact that the subgraph count and its local sensitivity are both near their means with high probability implies that the ratio of the local sensitivity to the statistic, $\frac{LS_H}{f_H}$, is on the order of $n^{|V_H|-2}p^{|E_H|-1}/(n^{|V_H|}p^{|E_H|}) = 1/(n^2p)$.

Second, when $p \gg 1/\sqrt{n}$, the noise added by our algorithms is proportional to LS_H/ϵ with high probability. This means that the ratio $\frac{LS_H}{\epsilon f_H}$ is a good estimate of the relative error of the algorithm. Putting these statements together, we see that the relative error is $O(\frac{1}{\epsilon n^2 p})$.

In sparser graphs, the performance of the algorithms is harder to analyze theoretically, but we can nonetheless get meaningful (if messier) statements for relatively sparse graphs, *e.g.,* with polylogarithmic expected degree.

Theorem B.2. (Sparse Graphs) *If the expected average degree np is $\Omega(\log^2 n)$ (as n goes to ∞) then the algorithms we describe for releasing $f_\triangle$ (from [9]) and $f_{2\triangle}$ (from this paper) with high probability have relative error $O(\max\left\{\frac{1}{\epsilon^2 n^3 p^3}, \frac{1}{\epsilon n^2 p}\right\})$.*

C. MISSING PROOFS FROM SECTION 3

C.1 Triangles

Proof of Theorem 3.2. Let $c_{ij}(t)$ be as in Theorem 3.1. Let $U^{(t)}(G) = t + \max_{i\neq j; i,j\in[n]} a_{ij} = LS_\triangle(G) + t$. Then $LS^{(t)}_\triangle(G) \leq U^{(t)}(G)$ for all t, since $c_{ij}(t) \leq a_{ij} + t$. Thus, by Lemma 2.4, the smooth sensitivity $S^*_{\triangle,\beta}(G)$ is at most

$$S_\beta(G) = \max_{t\in[0,\binom{n}{2}]} e^{-t\beta}U^{(t)}(G) = \max_{t\in[0,\binom{n}{2}]} e^{-t\beta}(LS_\triangle(G) + t).$$

We can obtain an upper bound on S_β by taking the maximum over all real numbers in the interval $[0, \binom{n}{2}]$ instead of only integers. For any number A, consider the function $h(t) = e^{-\beta t}(A+t)$. Its derivative, $h'(t) = e^{-\beta t}(1-\beta(A+t))$, tells us that h is strictly concave and has a unique maximum at $t = \frac{1}{\beta} - A$. When $A \geq \frac{1}{\beta}$, the function h is decreasing on $[0, \infty)$. Thus, when $LS_\triangle(G) \geq \frac{1}{\beta}$, the maximum in the expression for $S_\beta(G)$ is attained at $t = 0$, and is equal to $LS_\triangle(G)$, so $S_\beta(G) = S^*_{\triangle,\beta}(G) = LS_\triangle(G)$. ∎

C.2 k-Stars

Proof of Lemma 3.4. First, recall that, by definition of $LS^{(t)}_{k\star}(G)$ and $LS^{(t)}_{ij}(G)$ (at the end of Section 2.1), $LS^{(t)}_{k\star}(G) = \max_{(i,j):\, d_i\geq d_j} LS^{(t)}_{ij}(G)$. Using the notation in

Lemma 3.4, the expression for the local sensitivity $LS_{k\star}(G)$, stated in Lemma 3.3, becomes $\max\limits_{i \neq j; i,j \in [n]} C(d'_i) + C(d'_j)$, *i.e.*,

$$LS_{ij}(G) = C(d'_i) + C(d'_j). \tag{3}$$

In order to understand $LS^{(t)}_{ij}(G)$ for positive t, we modify t edges in G to obtain a graph G' with maximum $LS_{ij}(G')$. The only modification to G that increases $LS_{ij}(G)$ is the addition of edges adjacent to either i or j. (Note that the presence of an edge (i, j) does not affect $LS_{ij}(G)$.) Adding an edge adjacent to i (resp., j) increments d'_i (resp., d'_j). Thus, G' should be obtained from G by adding t edges, and it remains to decide how to allocate them between nodes i and j.

Recall that we assumed $d'_i \geq d'_j$. Since $\binom{a+1}{k-1} - \binom{a}{k-1} = \binom{a}{k-2}$ for all integer $a \geq 0$ and $k \geq 2$ and since the binomial coefficient $\binom{a}{k-2}$ is monotonically increasing in a, we increment d'_i greedily for every unit of t, while possible, i.e., while $d'_i \leq n-2$. This ensures the largest increase in (3) and the largest increases for subsequent increments. This gives rise to the three cases in Lemma 3.4: (a) If $t \leq B_i$ then d'_i is increased by t. (b) If $t \in (B_i, B_i + B_j)$ then d'_i is increased until it becomes the maximum possible, $n-2$, and then the remaining $t - B_i$ edges are used to increase d'_j. (c) Finally, if $t \geq B_i + B_j$, we increase both d'_i and d'_j to $n-2$, the attaining maximum possible local sensitivity $LS_{ij}(G')$ with B_i and B_j edge modifications. ∎

Proof of Theorem 3.5. To compute the smooth sensitivity of $f_{k\star}$, we start by expressing it as the maximum of the local sensitivities at distance t over edges in G, as suggested in Lemma 2.4 and the subsequent discussion. We get:

$$\begin{aligned} S^*_{k\star,\beta}(G) &= \max_{t \in [0, \binom{n}{2}]} e^{-t\beta} LS^{(t)}_{k\star}(G) \\ &= \max_{t \in [0, 2n-2]} e^{-t\beta} \max_{(i,j):\, d_i \geq d_j} LS^{(t)}_{ij}(G). \end{aligned}$$

In the last line, we can take the maximum only over $t \leq 2n-2$ because, by Lemma 3.4, the value of $LS^{(t)}_{ij}(G)$ is the same for all $t \geq 2n-2$.

Suppose that the vertices of G are numbered in the order of non-increasing degree, i.e., $d_1 \geq \ldots \geq d_n$. (Sorting them takes time $O(n \log n)$.) Consider the expression $\max_{(i,j):\, d_i \geq d_j} LS^{(t)}_{ij}(G)$ for $LS^{(t)}_{k\star}(G)$. Lemma 3.4 implies that if $x_{12} = 0$ then for all t, the maximum of this expression is attained on $(1, 2)$ because $d_1 - x_{12} \geq d_i - x_{ij}$ and $d_2 - x_{12} \geq d_j - x_{ij}$ for all pairs of nodes (i, j) with $d_i \geq d_j$.

Now consider the case when $x_{12} = 1$. We will show that the maximum of this expression must be attained on one of three pairs of vertices. Let D_1 be the set of vertices of degree d_1 and D_2 be the set of vertices of the second largest degree. If the maximum is not attained on (1,2) then $d_1 - x_{12} < d_i - x_{ij}$ or $d_2 - x_{12} < d_j - x_{ij}$ for some pair (i, j) with $d_i \geq d_j$. If $d_1 - x_{12} < d_i - x_{ij}$ then $i \in D_1$ and $x_{ij} = 0$. If $d_2 - x_{12} < d_j - x_{ij}$ then $j \in D_2$.

Define v_1 to be a vertex with maximum degree d_{v_1} such that there is a vertex $u_1 \in D_1$ not adjacent to v_1 (if v_1 and u_1 exist). Define v_2 and u_2 analogously with respect to D_2. Then the maximum of the expression $\max_{(i,j):\, d_i \geq d_j} LS^{(t)}_{ij}(G)$ is attained on $(1, 2)$, (u_1, v_1) or (u_2, v_2). This gives:

$$S^*_{k\star,\beta}(G) = \max_{t \in [0, 2n-2]} e^{-t\beta} \max_{(i,j) \in \{(1,2),(u_1,v_1),(u_2,v_2)\}} LS^{(t)}_{ij}(G).$$

We use this formula to compute $S^*_{k\star,\beta}(G)$. It remains to prove that it takes $O(m + n \log n)$ time.

Once the vertices are sorted by degree, we can find sets D_1 and D_2 in time $O(n)$. The vertices v_1 and v_2, if they exist, can be found in time $O(m)$ by scanning the list of edges and counting for each vertex the number of adjacent edges reaching D_1 and D_2. Once we know (u_1, v_1) and (u_2, v_2), we can compute $LS^{(0)}_{12}(G)$, $LS^{(0)}_{u_1 v_1}(G)$, and $LS^{(0)}_{u_2 v_2}(G)$ in time $O(1)$. Using Lemma 3.4, computing $LS^{(t)}_{ij}(G)$ from $LS^{(t-1)}_{ij}(G)$ for any pair (i, j) takes time $O(1)$, and taking the maximum over $t \leq 2n-2$ requires time $O(n)$. Thus, the total time is $O(m + n \log n)$. ∎

D. MISSING PROOFS FROM SECTION 4

Proof of Lemma 4.1. We start by analyzing $LS_{ij}(G)$. When we add a new edge (i, j) to G, we form $\binom{a_{ij}}{k}$ k-triangles with base (i, j). In addition, for each node $\ell \in N_{ij}$, we form $\binom{a_{i\ell}}{k-1}$ new k-triangles with base (i, ℓ) and $\binom{a_{\ell j}}{k-1}$ new k-triangles with base (ℓ, j). No other new k-triangles are formed. Thus, the change in the number of k-triangles is given by (1) with $x_{ij} = 0$. The case when an edge (i, j) is deleted from G is symmetric. Taking the maximum over all $LS_{ij}(G)$, by definition, gives $LS_{k\triangle}(G)$. ∎

Proof of Lemma 4.2. Note that for every pair (i, j), flipping (that is, adding or removing) the edge (i, j) does not change $LS_{ij}(G)$. If some edge adjacent to either i or j is flipped, suppose w.l.o.g. that this edge is adjacent to i. Then $a_{ij} = |N_{ij}|$ changes by at most one and, consequently, at most one vertex (say, v) is added to N_{ij}. Also, for each $\ell \in N_{ij}$, where N_{ij} is the set of common neighbors of i and j before flipping the edge, $a_{i\ell}$ changes by at most one. Overall, $LS_{ij}(G)$ changes by at most

$$\begin{aligned} &\binom{a_{ij}+1}{k} - \binom{a_{ij}}{k} + \binom{a_{iv} - x_{ij}}{k-1} + \binom{a_{vj} - x_{ij}}{k-1} \\ &\qquad + \sum_{\ell \in N_{ij}} \left(\binom{a_{i\ell} + 1 - x_{ij}}{k-1} - \binom{a_{i\ell} - x_{ij}}{k-1} \right) \\ \leq\ & \binom{a_{ij}}{k-1} + \binom{a_{iv}}{k-1} + \binom{a_{vj}}{k-1} + \sum_{\ell \in N_{ij}} \binom{a_{i\ell}}{k-2} \\ \leq\ & 3\binom{a_{\max}}{k-1} + a_{\max}\binom{a_{\max}}{k-2}. \end{aligned}$$

Finally, if an edge (ℓ_1, ℓ_2) adjacent to neither i nor j is flipped then $LS_{ij}(G)$ changes by at most

$$\begin{aligned} &\sum_{\substack{u \in \{i,j\} \\ v \in \{\ell_1,\ell_2\}}} \left(\binom{a_{uv} + 1 - x_{ij}}{k-1} - \binom{a_{uv} - x_{ij}}{k-1} \right) \\ &= \sum_{\substack{u \in \{i,j\} \\ v \in \{\ell_1,\ell_2\}}} \binom{a_{uv} - x_{ij}}{k-1} \\ &\leq 4\binom{a_{\max}}{k-2} \leq 3\binom{a_{\max}}{k-1} + a_{\max}\binom{a_{\max}}{k-2}. \qquad ∎ \end{aligned}$$

Proof of Lemma 4.4. Consider two neighboring data sets x and y. We wish to compare the random variables

$$\mathcal{A}(x) = (\widetilde{LS}_x, f(x) + Lap(\widetilde{LS}_x/\epsilon_2)) \text{ and}$$
$$\mathcal{A}(y) = (\widetilde{LS}_y, f(y) + Lap(\widetilde{LS}_y/\epsilon_2)),$$

where $\widetilde{LS}_x = \mathcal{B}(x)$ and $\widetilde{LS}_y = \mathcal{B}(y)$. We consider a hybrid random variable $\mathcal{A}_{mix} = (\widetilde{LS}_x, f(y) + Lap(\widetilde{LS}_x/\epsilon_2))$. Let P_x, P_y and P_{mix} be the probability distributions of $\mathcal{A}(x), \mathcal{A}(y)$ and $\mathcal{A}_{mix}$, respectively.

First, consider the difference between $\mathcal{A}(y)$ and $\mathcal{A}_{mix}$. They differ only in the initial estimate $\widetilde{LS}$ (either $\mathcal{B}(y)$ or $\mathcal{B}(x)$). Since $\mathcal{B}$ is (ϵ_1, δ_1)-differentially private and since, by Lemma 2.1, post-processing does not affect differential privacy, it follows that for every event E,

$$P_y(E) \leq e^{\epsilon_1} P_{mix}(E) + \delta_1 \,. \tag{4}$$

Second, consider the difference between $\mathcal{A}_{mix}$ and $\mathcal{A}(x)$. Both random variables use the same estimate $\widetilde{LS}_x = \mathcal{B}(x)$ for the local sensitivity. Let F be the event that $\widetilde{LS}_x > LS_f(x)$. By hypothesis, $\Pr(F) \geq 1 - \delta_2$. Conditioned on F, changing the argument of f from y to x increases the probability of any event by at most e^{ϵ_2} (this follows from the differential privacy of the Laplace mechanism (Theorem 2.2) and the fact that $|f(x) - f(y)| < LS_f(x)$). Thus, for every event E, we have: $P_{mix}(E|F) \leq e^{\epsilon_2} P_x(E|F)$. Since the probability of F is the same under both P_{mix} and P_x, we can strengthen this to $P_{mix}(E \wedge F) \leq e^{\epsilon_2} P_x(E \wedge F)$. Since $\Pr(\bar{F}) \leq \delta_2$,

$$\begin{aligned} P_{mix}(E) \leq P_{mix}(E \wedge F) + P_{mix}(E \wedge \bar{F}) \leq& \\ e^{\epsilon_2} P_x(E \wedge F) + P_{mix}(E \wedge \bar{F}) \leq& \quad e^{\epsilon_2} P_x(E) + \delta_2 \,. \end{aligned}$$

Putting this together with (4), we get: $P_y(E) \leq e^{\epsilon_1} P_{mix}(E) + \delta_1 \leq e^{\epsilon_1 + \epsilon_2} P_x(E) + e^{\epsilon_1}\delta_2 + \delta_1$, as desired. ∎

Proof of Theorem 4.3. Before proving the theorem, we establish that the sensitivity bounds computed in Algorithm 1 are indeed high-probability upper bounds on the appropriate quantities.

Lemma D.1. *With probability at least* $1 - \frac{\delta'}{2}$, *the values computed by Algorithm 1 satisfy* $\tilde{a}_{\max} \geq a_{\max}$ *and* $\widetilde{LS} \geq LS_{k\triangle}(G)$.

Proof. Taking $\lambda = 1/\epsilon'$ and $t = \frac{\ln(1/\delta')}{\epsilon'}$ in Theorem E.2 gives that $\tilde{a}_{\max} < a_{\max}$ with probability $\delta'/2$. Similarly, taking $\lambda = \frac{B(\tilde{a}_{\max})}{\epsilon'}$ and $t = \ln(1/\delta') \cdot \frac{B(\tilde{a}_{\max})}{\epsilon'}$ gives that $\widetilde{LS} < LS_{k\triangle}(G)$ with probability $\delta'/2$. ∎

We now prove the theorem. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the algorithms that output $B(\tilde{a}_{\max})$ and $(B(\tilde{a}_{\max}), LS_{k\triangle}(G) + Lap(\frac{B(\tilde{a}_{\max})}{\epsilon'}))$, respectively. First, note that the global sensitivity of $a_{\max}$ is 1. Hence, by Theorem 2.2, the algorithm that releases $a_{\max} + Lap(1/\epsilon')$ is ϵ'-differentially private. Consequently, by Lemma 2.1, so is $\mathcal{A}_1$.

Second, observe that if $\tilde{a}_{\max} \geq a_{\max}$ then, by Lemma 4.2, $B(\tilde{a}_{\max}) \geq B(a_{\max}) \geq LS'(G)$. By Lemma D.1, this occurs with probability $1 - \frac{\delta'}{2}$. Hence, Lemma 4.4 implies that $\mathcal{A}_2$ is $(2\epsilon', e^{\epsilon'}\frac{\delta'}{2})$-differentially private.

Finally, by Lemma D.1, $\widetilde{LS} \geq LS_{k\triangle}(G)$ with probability $1 - \frac{\delta'}{2}$. Then Lemma 4.4 implies that Algorithm 1 is $(3\epsilon', e^{\epsilon'}\delta'/2 + e^{2\epsilon'}\delta'/2)$-differentially private. Since $\epsilon' = \epsilon/3$, $\delta' = \delta/3$ and $\epsilon \leq \frac{3}{2}\ln(\frac{3}{2})$, we get

$$e^{\epsilon'}\delta'/2 + e^{2\epsilon'}\delta'/2 \leq 2e^{2\epsilon'}\delta/3 \leq \delta$$

and, hence, Algorithm 1 is (ϵ, δ)-differentially private. ∎

Proof of Theorem 4.5. By Theorem E.2, $\tilde{a}_{\max} \leq a_{\max} + \frac{2\ln(1/\delta')}{\epsilon'}$ with probability $1 - \frac{\delta'}{2}$ and also $\widetilde{LS} \leq LS_{f_{k\triangle}}(G) + \frac{2\ln(1/\delta')}{\epsilon'}B(\tilde{a}_{\max})$ with probability $1 - \frac{\delta'}{2}$, where $B(a) = 3\binom{a}{k-1} + a\binom{a}{k-2}$. By a union bound, with probability at least $1 - \delta'$ both of these inequalities hold. By monotonicity of B,

$$\begin{aligned} \widetilde{LS} \leq& LS_{f_{k\triangle}}(G) + \frac{2\ln(1/\delta')}{\epsilon'}B(\tilde{a}_{\max}) \\ \leq& LS_{f_{k\triangle}}(G) + \frac{2\ln(1/\delta')}{\epsilon'}B\big(a_{\max} + \frac{2\ln(1/\delta')}{\epsilon'}\big). \end{aligned} \tag{5}$$

For constant k, we have $B(a_{\max}) = \Theta\left(a_{\max}^{k-1}\right)$ and $LS_{f_{k\triangle}}(G) = \Theta\left(a_{\max}^{k}\right)$. Then the theorem follows, because if $\frac{\ln(1/\delta)}{\epsilon} = o(a_{\max})$, then also $\frac{2\ln(1/\delta')}{\epsilon'} = o(a_{\max})$ and so the right-hand side of (5) is $o(a_{\max}^k)$, which is $LS_{k\triangle}(G) \cdot o(1)$. Therefore the scale parameter of the Laplace noise added, which is $\widetilde{LS}/\epsilon'$, is at most $(LS_{k\triangle}(G)/\epsilon') \cdot (1 + o(1))$. ∎

Runtime Analysis of Algorithm 1. We can compute a_{ij} for all $i, j \in [n]$ in time $O(f_{2\star}(G)) = O(n(m+n))$. After that, $a_{\max}$ and $f_{k\triangle}(G)$ can be computed in time $O(n^2)$, using the formulas $a_{\max} = \max_{i,j\in[n], i\neq j} a_{ij}$ and $f_{k\triangle}(G) = \sum_{(i,j)\in E}\binom{a_{ij}}{k}$. We can compute $LS_{k\triangle}(G)$ in $O(n(m+n))$ time, using Lemma 4.1, since for all pairs (i,j), the first and the second terms of (1) can be computed in total time $O(n^2)$ and $O(f_{2\star}(G)) = O(nm)$, respectively. The remaining steps of Algorithm 1 take time $O(1)$.

E. USEFUL CONCENTRATION BOUNDS

We make repeated use of several standard tail bounds.

Theorem E.1. (Chernoff Bounds) *Let* $X_1, \ldots, X_n$ *be i.i.d. Bernoulli random variables with* $\Pr[X_i = 1] = \mu$. *Then for every* $\phi \in (0, 1]$,

$$\begin{aligned} \Pr\left[\frac{\sum_i X_i}{n} \geq (1+\phi)\mu\right] \quad&\leq\quad \exp\left(-\frac{\phi^2\mu n}{3}\right) \quad \textit{and} \\ \Pr\left[\frac{\sum_i X_i}{n} \leq (1-\phi)\mu\right] \quad&\leq\quad \exp\left(-\frac{\phi^2\mu n}{2}\right)\,. \end{aligned}$$

Theorem E.2. (Tail bounds for Laplace) *Let* z *be drawn from the Laplace distribution with scale parameter* λ *and mean* 0, *denoted* $Lap(\lambda)$. *Then for all* $t > 0$,

$$\Pr(z > t) = \Pr(z < -t) = \frac{1}{2}e^{-t/\lambda}.$$