Private Release of Graph Statistics using Ladder Functions

Jun Zhang[1] Graham Cormode[2] Cecilia M. Procopiuc[3]* Divesh Srivastava[4] Xiaokui Xiao[1]

[1]Nanyang Technological University
{jzhang027, xkxiao}@ntu.edu.sg

[2]University of Warwick
g.cormode@warwick.ac.uk

[3]Google Inc.
mpro@google.com

[4]AT&T Labs – Research
divesh@research.att.com

ABSTRACT

Protecting the privacy of individuals in graph structured data while making accurate versions of the data available is one of the most challenging problems in data privacy. Most efforts to date to perform this data release end up mired in complexity, overwhelm the signal with noise, and are not effective for use in practice. In this paper, we introduce a new method which guarantees differential privacy. It specifies a probability distribution over possible outputs that is carefully defined to maximize the utility for the given input, while still providing the required privacy level. The distribution is designed to form a 'ladder', so that each output achieves the highest 'rung' (maximum probability) compared to less preferable outputs. We show how our ladder framework can be applied to problems of counting the number of occurrences of subgraphs, a vital objective in graph analysis, and give algorithms whose cost is comparable to that of computing the count exactly. Our experimental study confirms that our method outperforms existing methods for counting triangles and stars in terms of accuracy, and provides solutions for some problems for which no effective method was previously known. The results of our algorithms can be used to estimate the parameters of suitable graph models, allowing synthetic graphs to be sampled.

Categories and Subject Descriptors

H.2.7 [**Database Administration**]: Security, integrity & protection

Keywords

Differential privacy; subgraph counting; local sensitivity

1. INTRODUCTION

A large amount of private data can be well-represented by graphs. Social network activities, communication patterns, disease transmission, and movie preferences have all been encoded as graphs, and studied using the tools of graph theory. Given the private nature of the information stored in graphs, and the increasing pressure to allow controlled release of private data, there has been much interest in ways to do so with some guarantee of anonymity.

*Author worked on this result while she was at AT&T Labs.

The current locus of privacy work is around differential privacy. The model is popular, since it offers both provable properties on the results and practical algorithms to provide them. It is a statistical model of privacy which ensures that the output of the process undergoes sufficient perturbation to mask the presence or absence of any individual in the input. This model is very effective for releasing data in the form of histograms or counts, since the magnitude of the statistical noise is often dominated by random variation in the data. Applying this model to graph data has proven much more fraught. The main technical roadblock is that direct application of standard methods seems to require such a large scale of random modification of the graph input as to completely lose all its properties. Concretely, any method which tries to directly output a modified version of the input graph under differential privacy is indistinguishable from random noise.

Instead, progress has been made by focusing on releasing properties of the input graph under differential privacy, rather than the graph itself. For example, it is straightforward to release the number of nodes and edges with this guarantee, by appealing to standard techniques. It is valuable to find statistical properties of the graph, since there are many random graph models which take these as parameters (e.g., Kronecker graph models [26] and Exponential Random Graph Models [17]), and allow us to sample graphs from this family, which should have similar properties to the input graph. The properties of the graph are also important in their own right, determining, for instance, measures of how much clustering there is in the graph, and other characteristics of group behavior. See the discussion in [17], and references therein.

The problem then becomes one of providing effective ways to release statistics on the input graph privately. The most important statistics are *subgraph counts*: the number of occurrences of particular small subgraphs within the input graph. This remains a challenging problem, since standard differential privacy techniques still add a large amount of noise to the result. Take as a canonical problem the question of counting the number of triangles within a graph. The presence or absence of one relationship (represented by an edge) in the graph can contribute to a large number of potential triangles, and so the noise added to the count has to be scaled up by this amount. There have been numerous prior attempts to privately release statistics on graphs via more complex methods, but these still suffer from high noise, and high running time.

In this paper, we introduce a new technique for producing differentially private output, which is applied to the problem of producing subgraph counts. The technique builds upon ideas from differential privacy, specifically notions of the "sensitivity" of a function, and sampling possible output values using the exponential mecha-

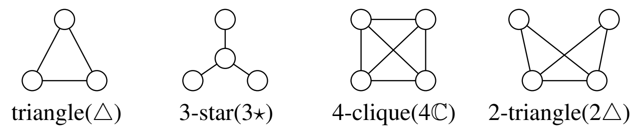


Figure 1: Examples of subgraphs

nism and a carefully tuned quality function. We combine these in a new way to define a class of "ladder functions", which provide the optimal sampling distribution for outputs. When applied to subgraph counting queries, the results are efficient to compute, and improve substantially over prior work on these problems, where applicable.

1.1 Our Contributions

We show how to answer subgraph counting queries in a differentially private manner [7]. Given a subgraph H, e.g., a triangle, we aim to release the number of isomorphic copies of H in the sensitive graph, while protecting individual privacy in the meantime. We write $f_H(g)$ to denote the function that computes the number of copies of subgraph H in graph g. We focus on a number of important classes of H. The function $f_\triangle$ counts the number of triangles in the graph: this is the most heavily studied subgraph counting query, as the triangle is the smallest non-trivial subgraph of interest. Other important subgraph counting functions include: $f_{k\star}$, for counting k-stars (one node of degree k connected to k nodes of degree 1); $f_{k\mathbb{C}}$, for counting k-cliques; and $f_{k\triangle}$ for counting k-triangles (a pair of adjacent nodes i, j that have k common neighbors). Figure 1 shows the smallest non-trivial example of each of these classes: note that the 3-clique and the 1-triangle are simply the standard triangle.

We draw up the following list of criteria that our desired solution for differentially private subgraph counting should satisfy:

1. The solution should achieve pure differential privacy;

2. It should be applicable to a wide range of queries, e.g., triangle counting, k-star counting, k-triangle counting and k-clique counting for a constant k;

3. Its time complexity should be no larger than that of computing the true query answer for the queries of interest;

4. It should have high accuracy on its output, bettering that of any previous applicable solution.

Outline of Ladder Framework Development. Our ladder framework achieves all the above criteria. It relies on the careful design of a probability distribution for outputting answers to count queries that tries to maximize the probability of outputting the true answer, or one that is near to it, and minimize the probability of outputting answers that are far from the true answer. The framework of differential privacy places constraints on how these probabilities should behave: the probability of providing output x for input graph g should be "close" (as determined by the parameter ε) to that for input graph g', if g and g' are close (differ by one edge). Rather than go right back to first principles to design these probability distributions, we make use of the exponential mechanism [25], since this approach is general: any differentially private mechanism can be expressed as an exponential mechanism.

Instantiating the exponential mechanism still takes some careful design. We arrive at the design of a symmetric "ladder function" specific to the input graph g, so-called because it is formed from a series of rungs. The first rung groups together a number of outputs that are close to the true answer to $f_H(g)$ (i.e. $f_H(g) \pm 1, f_H(g) \pm 2 \ldots$). The second rung groups together outputs that are a little further away, and so on (see Figure 3). The height of each rung determines the probability of outputting the corresponding values, while the width is the number of output values sharing the same height. Ideally, we would make each rung as narrow as possible, to maximize the dropoff in probabilities away from f_H. However, to guarantee differential privacy, we need to make the rungs wide enough so that the ith rung of the ladder for g overlaps with the $i-1$st, ith or $i+1$st rung of the ladder for all neighboring g' (in the language of the exponential mechanism, we seek bounded 'sensitivity' of the derived quality function). We call this the "small leap property".

It turns out that we can design such ladder functions which have rungs that are not too wide but which satisfy the small leap property. Moreover, these can be computed efficiently from the input graph g and used to perform the required output sampling. Our development of these objects takes multiple steps. First, we fill in the necessary details of differential privacy (Section 2). Then we show how the use of ladder functions can lead to a practical differentially private solution, if we assume the small leap property, and that the functions converge to a constant rung width (Section 3). We still have to show that we can create such functions, and this is the focus of Section 4, where we show that the concept of "local sensitivity at distance t" can be used to create a ladder function with the required properties. This provides a general solution which satisfies privacy; however, some of the quantities required can be time consuming to compute for certain subgraph counting queries. Hence, Section 5 considers how to instantiate our framework for specific queries, and provides alternate ladder functions for some queries to enable efficient computation. Our experimental evaluation of this approach, and comparison with previous works (where applicable) is detailed in Section 6.

1.2 Previous Work on Subgraph Counting

An obvious first attempt to differentially private subgraph counting is to apply the Laplace mechanism [11]. The Laplace mechanism answers a (numeric) query f by adding Laplace noise to the true answer, where the scale of noise is proportional to the *global sensitivity* of f. The global sensitivity is defined as the maximum possible change of query answer if one tuple is added to/deleted from an arbitrary database. This method fails since the global sensitivity of subgraph counting queries is very large, making the noise large enough to overwhelm the true answer. In the case of triangles, for example, adding or deleting an edge can affect a number of triangles proportional to the number of nodes, and hence the global sensitivity is very high.

Nissim et al. [29] and Karwa et al. [17] present an idea of utilizing a local version of global sensitivity, called *local sensitivity*, to build the noise distribution. The local sensitivity is a function of the input database, which is equal to the maximum change of query answer if one tuple is added in/deleted from the input database. The local sensitivity can be dramatically smaller than its global counterpart, but cannot be used directly to guarantee privacy. Instead, an upper bound on local sensitivity called *smooth sensitivity* is computed as a part of the noise scale. In [17], the authors give algorithms for computing smooth sensitivity of $f_\triangle$ and $f_{k\star}$. However, their idea fails when the smooth sensitivity of a query is hard to compute, e.g., $f_{k\triangle}$. In recompense, they propose an approach tailored specifically for $f_{k\triangle}$, which only achieves a restricted version of differential privacy.

Instead of answering the subgraph counting query directly, Chen et al. [5] release a lower bound on the result, whose global sensitivity is relatively low. This method suffers from a bias between the true query answer and the lower bound, in exchange for less noise. The authors claim that their method can be extended to a variety of subgraph counting queries, but it involves solving a linear program

Table 1: Table of notation

Notation	Description
f, g	the graph query and its sensitive input
$\mathcal{G}_n$	the set of simple graphs with n nodes
$d(g, g')$	minimum edit distance between two graphs
GS	global sensitivity of f
$LS(g)$	local sensitivity of f, given g
$LS(g,t)$	local sensitivity of f at distance t, given g
$q(g,k)$	quality of output k, given input g
Δ_q	sensitivity of quality function q
$I_t(g)$	the ladder function

with $O(L)$ variables where L equals the number of subgraphs H in the input times the number of edges in H. So the time complexity is super-quadratic in the answer value, making this method infeasible even for an input graph of moderate size (see details in Section 6).

Finally, Hay et al. [15] give a differentially private algorithm for publishing degree sequence of a sensitive graph. Since the number of k-stars is a function of the degree sequence, this approach implicitly allows k-star counting. It first utilizes the Laplace mechanism to generate a private version of the degree sequence, then applies a postprocessing step to enforce consistency of the sequence in order to reduce noise. Though better than the naive Laplace mechanism, it is reported to be less accurate than the solutions in [17] on k-star counting (see Section 6.3 of [17]). We note that this method was not designed for k-star counting; also it does *not* support other counting queries discussed in this paper.

2. PRELIMINARIES

In this section, we review the definition of differential privacy, two conventional mechanisms to achieve differential privacy, and some basic concepts of local sensitivity. The notational conventions of this paper are summarized in Table 1.

2.1 Differential Privacy

Let $g \in \mathcal{G}_n$ be a sensitive simple graph with n nodes, and f be the subgraph counting query of interest. Differential privacy requires that, prior to $f(g)$'s release, it should be modified using a randomized algorithm $\mathcal{A}$, such that the output of $\mathcal{A}$ does not reveal much information about any edge in g. The definition of differential privacy that we adopt in the context of graphs is as follows:

DEFINITION 2.1 (ε-DIFFERENTIAL PRIVACY [11, 15]). *A randomized algorithm $\mathcal{A}$ satisfies ε-differential privacy, if for any two graphs g and g' that differ by at most one edge, and for any possible output O of $\mathcal{A}$, we have*

$$\Pr[\mathcal{A}(g) = O] \le e^{\varepsilon} \cdot \Pr[\mathcal{A}(g') = O], \tag{1}$$

where $\Pr[\cdot]$ *denotes the probability of an event.*

In what follows, we say that two graphs g, g' which differ by at most one edge are *neighboring*, i.e., their minimum edit distance [4] $d(g,g') \le 1$. While there are many approaches to achieving differential privacy, the best known and most-widely used two for this purpose are the *Laplace mechanism* [11] and the *exponential mechanism* [25].

The Laplace mechanism releases the result of a function f that takes as input a dataset and outputs a vector of numeric values. Given f, the Laplace mechanism transforms f into a differentially private algorithm by adding i.i.d. noise (denoted as η) into each output value of f. The noise η is sampled from a *Laplace distribution* $\mathrm{Lap}(\lambda)$ with the following pdf: $\Pr[\eta = x] = \frac{1}{2\lambda} e^{-|x|/\lambda}$. Dwork et al. [11] show that the Laplace mechanism achieves ε-differential privacy if $\lambda \ge GS^f/\varepsilon$, where GS^f is the *global sensitivity* of f. In our problem setting (i.e., f is a subgraph counting query), GS^f is defined as follows:

DEFINITION 2.2 (GLOBAL SENSITIVITY [11]). *Let f be a function that maps a graph into a real number. The global sensitivity of f is defined as*

$$GS^f = \max_{g,g' | d(g,g') \le 1} \left| f(g) - f(g') \right|, \tag{2}$$

where g and g' are any two neighboring graphs.

Intuitively, GS^f measures the maximum possible change in f's output when we modify one arbitrary edge in f's input.

For an analysis task f with a categorical output (e.g., a zipcode), injecting random noise no longer yields meaningful results, but the exponential mechanism [25] can be used instead. The exponential mechanism releases a differentially private version of f, by sampling from f's output domain Ω. The sampling probability for each $k \in \Omega$ is determined based on a user-specified *quality function* q, which takes as input any dataset g and any element $k \in \Omega$, and outputs a numeric score $q(g,k)$ that measures the quality of k: a larger score indicates that k is a better output with respect to g. More specifically, given a graph g, the exponential mechanism samples $k \in \Omega$ with probability:

$$\Pr[k \text{ is selected}] \propto \exp\left(\frac{\varepsilon}{2\Delta_q} \cdot q(g,k)\right), \tag{3}$$

where Δ_q is the sensitivity of the quality function, i.e.,

$$\Delta_q = \max_{g,g',k \in \Omega} \left| q(g,k) - q(g',k) \right| \text{ for } g, g' \text{ s.t. } d(g,g') \le 1.$$

McSherry and Talwar [25] prove that the exponential mechanism ensures ε-differential privacy.

Both mechanisms can be applied quite generally; however, to be effective we seek to ensure that (i) the noise introduced does not outweigh the signal in the data, and (ii) it is computationally efficient to apply the mechanism. This requires a careful design of the functions to use in the mechanisms.

2.2 Local Sensitivity

The scale of noise added by the Laplace mechanism is proportional to GS^f of the query, when the privacy budget ε is fixed. For all queries discussed in this paper, this approach is not effective since they all have large GS^f relative to the size of the query answer. In [29], the authors seek to address this problem by presenting a local measurement of sensitivity, as in Definition 2.3:

DEFINITION 2.3 (LOCAL SENSITIVITY [29]). *Let f be a function that maps a graph into a real number. The local sensitivity of f is defined as*

$$LS^f(g) = \max_{g' | d(g,g') \le 1} \left| f(g) - f(g') \right|, \tag{4}$$

where g' is any neighboring graph of g.

Note that global sensitivity can be understood as the maximum of local sensitivity over the input domain, i.e., $GS^f = \max_g LS^f(g)$. For simplicity, we write GS and $LS(g)$ instead in the remainder of the paper, if there is no ambiguity about the subgraph counting query of interest in context.

We refine the definition of local sensitivity, by defining the sensitivity for a particular pair of nodes (i, j), denoted by $LS_{ij}(g)$, to

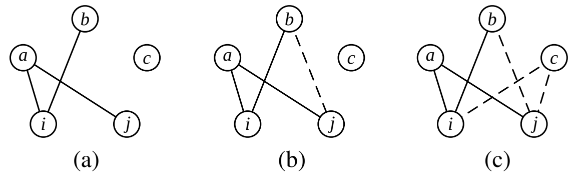


Figure 2: Examples of local sensitivity at distance t

indicate the magnitude of change of f when the connection between i and j is modified (added if absent; deleted if present) in the input graph. This then satisfies $LS(g) = \max_{i,j} LS_{ij}(g)$.

Although tempting, it is not correct to simply substitute $LS(g)$ for GS in the Laplace mechanism. This is because $LS(g)$ itself conveys information from the sensitive data. To address this problem, Nissim et al. [29] introduce *smooth sensitivity*, an upper bound of $LS(g)$, and show that differential privacy can be achieved by injecting Cauchy noise (instead of Laplace noise) calibrated to smooth sensitivity. As we show in Section 6, however, the adoption of Cauchy noise considerably degrades the quality of the output from a differentially private algorithm. This issue could be alleviated by combining smooth sensitivity with Laplace noise, but the resulting solution would only achieve a weakened version of ε-differential privacy (referred to as (ε, δ)-differential privacy [10]).

In this paper, we do not use smooth sensitivity, but adopt an intermediate notion of *local sensitivity at distance t* as a crucial part of our solution.

Definition 2.4 (Local Sensitivity at Distance t [29]). *The local sensitivity of f at distance t is the largest local sensitivity attained on graphs at distance at most t from g. The formal definition is:*

$$LS(g,t) = \max_{g^* | d(g,g^*) \leq t} LS(g^*). \tag{5}$$

Note that $LS(g,t)$ is a monotonically increasing function of t.

By analogy with $LS(g,t)$ and $LS(g)$, we further define $LS_{ij}(g,t)$ as the largest $LS_{ij}(\cdot)$ attained on graphs at distance at most t from g. Then $LS(g,t) = \max_{i,j} LS_{ij}(g,t)$.

Figure 2 gives an example of local sensitivity at distance t for triangle counting. The initial graph g in Figure 2(a) contains five nodes, three edges and no triangle. Its local sensitivity equals 1, achieved by adding edge (i, j) to the graph, i.e., $LS(g) = LS_{ij}(g) = 1$. Thereafter, $LS(g,1)$ allows one extra modification before calculating LS. In Figure 2(b), we show how the dashed edge (b, j) affects $LS_{ij}(\cdot)$, making $LS(g,1) = LS_{ij}(g,1) = 2$. Furthermore, we find that LS could never rise to 3 within 2 steps of modifications, and Figure 2(c) illustrates one example achieving $LS(g,3) = 3$.

3. OVERVIEW OF SOLUTION

This section presents the steps for using our ladder framework for answering subgraph counting queries under ε-differential privacy, given an appropriate function with some assumed properties (these functions are instantiated and proved to possess these properties in subsequent sections). Like the Laplace mechanism, we add a certain amount of noise to the true answer to prevent inference of sensitive information. However, to achieve better results, we aim to exploit the fact that most realistic graphs can tolerate a lower amount of noise while still providing the needed privacy. Towards this end, we need to tailor the noise to the given input, without revealing anything about the input by our choice of noise distribution. We introduce how to build such a distribution through the exponential mechanism in Section 3.1, then give an algorithm for efficient sampling in Section 3.2.

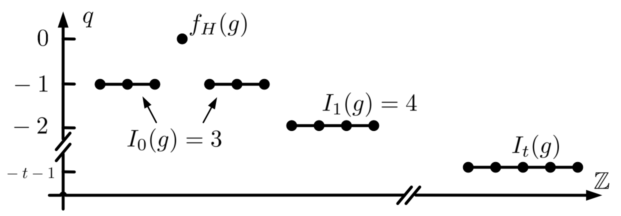


Figure 3: An example of a ladder quality

3.1 Building a Solution Distribution

Assume that we are given a query $f_H(g)$ that counts the number of subgraphs H in g. Let η be the distribution of the solution that we return as a private version of $f_H(g)$. In a nutshell, η is a discrete distribution defined over the integer domain, such that $\eta(k)$ increases with $q(g,k)$, where $q(g,k) \leq 0$ is a non-positive integer that quantifies the loss of data utility when we publish k instead of $f_H(g)$. We refer to q as a *quality function*, since we will use it as a quality function in the exponential mechanism. The expression of $\eta(k)$ as a function of $q(g,k)$ is given in equation (3) of Section 2.1.

Our choice of q is determined by a function $I_t(g)$, which defines how q varies. Figure 3 gives a working example of the q and $I_t(g)$. In particular, q is a symmetric function over the entire integer domain, centered at $f_H(g)$. We define the quality function to take its maximum value at $f_H(g)$. Without loss of generality, we set $q(g, f_H(g)) = 0$. For other possible answers, the quality decreases as the distance to the true answer increases. The decreasing speed is controlled by function $I_t(g)$. We refer to each quality level $(0, -1, -2, \ldots)$ as a *rung* of the ladder. In the figure, the first rung, corresponding to a quality level of -1, consists of the $I_0(g)$ integers on each side of $f_H(g)$, whose distances to $f_H(g)$ are no larger than $I_0(g)$. The next rung is formed of the next $I_1(g)$ integers on each side with quality -2, and so on. We say that I_t gives the "width" of the $t+1$st rung.

Definition 3.1 (Ladder Quality). *Formally, given function $I_t(g)$ we define the ladder quality function $q(g,k)$ by*

(i) $q(g, f_H(g)) = 0$;

(ii) $\forall k \in f_H(g) \pm \left(\sum_{t=0}^{u-1} I_t(g), \sum_{t=0}^{u} I_t(g)\right]$, *set* $q(g,k) = -u-1$.

After assigning each integer a quality score, we need to determine the sensitivity of the quality function,

$$\Delta_q = \max_{k,g,g' | d(g,g') \leq 1} |q(g,k) - q(g',k)|,$$

in order to apply the exponential mechanism. For arbitrary query class f_H and function $I_t(g)$, there is no simple answer: it can be the case that Δ_q is unbounded. Consequently, we restrict our attention to a class of well-behaved functions $I_t(g)$ that we will refer to as *ladder functions*. By analogy, we require that if we place two ladders corresponding to neighboring graphs g, g' next to each other and stand on the t'th rung of g's ladder, we can easily "leap across" to g''s ladder, since the rung at the corresponding position is at almost the same height. Formally,

Property 3.1 (Small leap property). *The ladder quality $q(g,k)$ defined by $I_t(g)$ has bounded Δ_q.*

In Section 4, we give a class of functions that meet this property since they have a constant $\Delta_q = 1$. For now, we simply assume that Δ_q is bounded and known to us, therefore, we can build the solution distribution using the exponential mechanism.

Algorithm 1 NoiseSample (f_H, g, ε, I_t): returns f^*

```
1: Compute true answer $f_H(g)$;
2: $range[0] = \{f_H(g)\}$; $weight[0] = 1 \cdot \exp\left(\frac{\varepsilon}{2\Delta_q} \cdot 0\right)$;
3: Initialize $dst = 0$;
4: for $t = 1$ to $M$ do
5:     $range[t] = f_H(g) \pm (dst, dst + I_{t-1}(g)]$;
6:     $weight[t] = 2I_{t-1}(g) \cdot \exp\left(\frac{\varepsilon}{2\Delta_q} \cdot -t\right)$;
7:     $dst = dst + I_{t-1}(g)$;
8: end for
9: $weight[M+1] = \frac{2C \cdot \exp\left(\frac{\varepsilon}{2\Delta_q} \cdot -(M+1)\right)}{1 - \exp\left(-\frac{\varepsilon}{2\Delta_q}\right)}$;
10: Sample an integer $t \in [0, M+1]$ with a probability $weight[t]$
    over sum of weights;
11: if $t \leq M$ then
12:     return a uniformly distributed random integer in $range[t]$;
13: else
14:     Sample an integer $i$ from the geometric distribution with parameter $p = 1 - \exp\left(-\frac{\varepsilon}{2\Delta_q}\right)$;
15:     return a uniformly distributed random integer in $f_H(g) \pm \{dst + i \cdot C + (0, C]\}$;
16: end if
```

3.2 Efficient Sampling from the Distribution

Our solution proceeds by using the ladder quality function to instantiate the exponential mechanism. This results in a solution which is guaranteed to provide differential privacy. However, the output of the mechanism can range over a very large, if not unbounded, number of possibilities, which we must sample from. The naive approach of computing the probability associated with each output value is not feasible: there are too many to calculate. Instead, we need a way to perform sampling in a tractable way. To reduce the time complexity of working with this infinite distribution, we assume a convergence property of $I_t(g)$, that there is a "bottom rung" of our ladder below which all rungs are the same width.

PROPERTY 3.2 (REGULAR STEP PROPERTY). *$I_t(g)$ converges to a constant C within M steps, i.e., $I_t(g) = C$ for any $t \geq M$.*

With these two properties, we can more effectively use the exponential mechanism, since there is a bounded number of sampling probabilities. Algorithm 1 gives pseudocode for the sampling process under these assumptions, which we now explain. Our algorithm aggregates all integers with the same quality, say $-t$, as t'th rung. The range function (Line 5) describes the domain of each rung, and the weight function (Line 6) gives the sum of weights within the domain. We illustrate how to calculate the range and weight for the first few rungs, e.g., rung 0 (the center) to rung M (Lines 2-8). For other rungs with indices $t > M$, we observe that their weights $2C \cdot \exp(\varepsilon/2\Delta_q \cdot -t)$ form a geometric sequence of t with common ratio $\exp(-\varepsilon/2\Delta_q)$. Therefore, we can write down a closed form sum of weights for all remaining rungs, as in Line 9, considered as one large rung indexed by $M+1$.

Lines 10-16 perform the random sampling. First, one rung is sampled with a probability proportional to its weight. If it is rung $M+1$, then we need to further specify the rung index by sampling from a geometric distribution (Line 14). At the end, we return a random integer within the range of the sampled rung as the final result. The time complexity of Algorithm 1 except Line 1 is $O(M)$.

Privacy guarantee. The correctness of Algorithm 1 follows from the correctness of the exponential mechanism [25]. Therefore, we have the following theorem (proof in Appendix B).

THEOREM 3.1. *Algorithm 1 satisfies ε-differential privacy, provided I_t has Properties 3.1 and 3.2.*

4. LADDER FUNCTIONS

The previous section explains how to build and sample a solution distribution with Algorithm 1, given a suitable function $I_t(g)$. In what follows, we consider how best to define such functions in general; we then apply this insight for the functions of interest that count various types of subgraph in the next section.

Some requirements for a good function are as follows. $I_t(g)$ should be a function of input graph g and step t, such that (i) $I_t(g)$ satisfies Properties 3.1 and 3.2, to guarantee the correctness of Algorithm 1; (ii) $I_t(g)$ is small. The rationale is that $I_t(g)$ controls the shape of the ladder. The smaller $I_t(g)$ the faster the quality decreases away from the true answer, and hence, the exponential mechanism tends to generate a distribution with higher probabilities for outputs closer to the true answer (smaller noise). In what follows, we first introduce a class of *ladder functions*, and give several examples. Then we prove that ladder functions always achieve a constant value for Δ_q, the sensitivity of the quality function (providing Property 3.1). Last, we construct a convergent ladder function for subgraph counting queries which always converges to the global sensitivity within $\binom{n}{2}$ steps (meeting Property 3.2).

We now formalize our notion of a ladder function.

DEFINITION 4.1 (LADDER FUNCTION). *$I_t(g)$ is said to be a* ladder function *of query f if and only if*

(a) $LS^f(g) \leq I_0(g)$, for any g;

(b) $I_t(g') \leq I_{t+1}(g)$, for any pair of neighboring graphs g, g', and any nonnegative integer t.

A straightforward example of a ladder function is the constant function $I_t(g) = GS$, since $LS(g) \leq GS$ for any g, and a constant always satisfies the second requirement. However, as discussed in Section 2.2, GS can be extremely large for subgraph counting, which does not adapt to typical graphs, which may not require so much noise. The local sensitivity $I_t(g) = LS(g)$ could be much smaller than GS, but unfortunately is not a ladder function because $LS(g') \leq LS(g)$ does *not* hold for all pairs of neighboring graphs. Our insight for designing good ladder functions is that we can use a relaxation of local sensitivity instead. Specifically, we adopt the notion of local sensitivity at distance t, i.e., $LS(g,t)$. We now prove that this choice is optimal within our framework: local sensitivity at distance t is the *minimum ladder function*, i.e. it is the lower envelope of all possible ladder functions.

THEOREM 4.1. *$LS(g,t)$ is the minimum ladder function that satisfies $LS(g,t) \leq I_t(g)$ for any ladder function $I_t(g)$.*

PROOF. We first prove that $LS(g,t)$ is a ladder function.

(i) $LS(g) = LS(g,0)$ (meeting Definition 4.1(a));

(ii) $\{g^* \mid d(g', g^*) \leq t\}$ is a subset of $\{g^* \mid d(g, g^*) \leq t+1\}$, given a pair of neighboring graphs g, g' such that $d(g, g') \leq 1$. Therefore, it meets Definition 4.1(b), since:

$$\begin{aligned} LS(g',t) &= \max_{g^* \mid d(g', g^*) \leq t} LS(g^*) \\ &\leq \max_{g^* \mid d(g, g^*) \leq t+1} LS(g^*) = LS(g, t+1). \end{aligned}$$

Next we prove $LS(g,t)$ is no larger than any ladder function $I_t(g)$ by induction on t.

Basis: when $t=0$, $LS(g,0)=LS(g)\le I_0(g)$ for all g;

Inductive step: suppose that $LS(g,t)\le I_t(g)$ holds for all g. It must be shown that $LS(g,t+1)\le I_{t+1}(g)$ holds for all g. For any pair of neighboring graphs g,g', we have $I_{t+1}(g)\ge I_t(g')$ given $I_t(g)$ is a ladder function. Thus, $I_{t+1}(g)\ge \max_{g'|d(g,g')\le 1} I_t(g')$ holds for all g. Then given the assumption that $LS(g,t)\le I_t(g)$ holds for all g,

$$\begin{aligned} I_{t+1}(g) &\ge \max_{g'|d(g,g')\le 1} I_t(g') \ge \max_{g'|d(g,g')\le 1} LS(g',t) \quad \text{by hypothesis}\\ &= \max_{g'|d(g,g')\le 1}\ \max_{g^*|d(g',g^*)\le t} LS(g^*)\\ &= \max_{g^*|d(g,g^*)\le t+1} LS(g^*) = LS(g,t+1) \end{aligned}$$

thus showing that indeed $LS(g,t+1)\le I_{t+1}(g)$ holds for all g. By induction, $LS(g,t)\le I_t(g)$ holds for all g and all nonnegative t. □

$LS(g,t)$ plays an important role in our framework. For certain subgraph counting queries, e.g., $f_\triangle, f_{k\star}$, it is the best ladder function that we can ever find. However, it is not always the preferred choice: as shown in Section 5, $LS(g,t)$ can be very hard to compute for some queries. To tackle this problem, we will present how to construct an alternate ladder function (basically, a relaxation of $LS(g,t)$) which is (i) computationally efficient and (ii) still much smaller than GS.

4.1 Bounded Δ_q

This subsection proves the most important property of ladder functions, that they satisfy Property 3.1.

THEOREM 4.2. *If $I_t(g)$ is a ladder function then the resulting quality function q has $\Delta_q=1$.*

PROOF. To prove this theorem, we must show that for any pair of neighboring graphs g,g', and any integer $k\in\mathbb{Z}$, $|q(g,k)-q(g',k)|\le 1$ always holds. That is, the quality ascribed to providing k as an output differs by at most 1 for neighboring graphs. In what follows, we discuss three different cases based on how close the value of k is to the top of the ladder.

Special case 1, the center: $k=f_H(g)\Rightarrow q(g,k)=0$. Given the properties of local sensitivity and ladder function (Definition 4.1(a)) $|k-f_H(g')|\le LS(g')$ and $LS(g')\le I_0(g')$, we have

$$f_H(g')-I_0(g')\le f_H(g)\le f_H(g')+I_0(g'). \tag{6}$$

That is, since k is the query answer for a neighboring graph of g', it must fall within the local sensitivity of the query answer for g'. From Definition 3.1(i) and (ii) with $u=0$ we have $q(g',k)\in\{0,-1\}$. Thus, the difference of quality is bounded by 1.

Special case 2, first rung: If $k\in f_H(g)+(0,I_0(g)]$, then $q(g,k)=-1$: k is on the first rung. By the construction of the ladder function, $q(g',k)$ should be on the center, the first or the second rung, and so the quality changes by at most 1. Formally, the lower bound of k as a function of g' is

$$k>f_H(g)\ge f_H(g')-I_0(g') \qquad \text{(from (6))};$$

and the upper bound is

$$k\le f_H(g)+I_0(g)\le f_H(g')+I_0(g')+I_1(g'),$$

using Definition 4.1(b) to show $I_0(g)\le I_1(g')$, combined with (6). Applying Definition 3.1, the value of the quality function satisfies $q(g',k)\in\{0,-1,-2\}$, and so the bound holds.

General case: Consider $k\in f_H(g)+(\sum_{t=0}^{u-1}I_t(g),\sum_{t=0}^{u}I_t(g)]$ so $q(g,k)=-u-1$, where $u>0$—we are on the $u+1$st rung on the ladder for g. We argue that we can only climb or descend a single rung when we move to the ladder for g'. The lower bound on k as a function of g' can be obtained using similar steps to case 2 above, from (6) and $u-1$ invocations of Definition 4.1(b):

$$\begin{aligned} k>f_H(g)+\sum_{t=0}^{u-1}I_t(g) &= f_H(g)+I_0(g)+\sum_{t=1}^{u-1}I_t(g)\\ &\ge f_H(g')+\sum_{t=0}^{u-2}I_t(g'). \end{aligned}$$

The upper bound uses a similar argument:

$$\begin{aligned} k&\le f_H(g)+\sum_{t=0}^{u}I_t(g)\\ &\le f_H(g')+I_0(g')+\sum_{t=1}^{u+1}I_t(g') \ = \ f_H(g')+\sum_{t=0}^{u+1}I_t(g'). \end{aligned}$$

Therefore, $q(g',k)$ is close to $q(g,k)=u-1$:

$$\begin{aligned} k\in f_H(g')+\left(\sum_{t=0}^{u-2}I_t(g'),\sum_{t=0}^{u+1}I_t(g')\right]\\ \Rightarrow q(g',k)\in\{-u,-u-1,-u-2\}. \end{aligned}$$

The analysis above proves the result for all $k\ge f_H(g)$. For integers $k<f_H(g)$, the proofs of special case 2 and the general case are symmetric, and yield the required result. □

4.2 Convergence

This subsection shows how to design a *convergent* ladder function for subgraph counting queries that meets Property 3.2. First, we state a useful lemma for our subsequent analysis. The proof is immediate given the definition of ladder functions, so we omit it.

LEMMA 4.1. *If $f(g,t)$ and $h(g,t)$ are ladder functions, $\min(f(g,t),h(g,t))$ is a ladder function.*

Using this result, one can easily design a convergent ladder function, e.g., $\min(I_t(g),GS)$, if the original function $I_t(g)$ is unbounded. This is critical since Algorithm 1 requires a convergent function. Unlike $LS(g,t)$ which converges to GS naturally, some ladder functions may need to be explicitly bounded by GS.

THEOREM 4.3. *For any subgraph counting query f_H, $\min(I_t(g),GS)$ is a ladder function of f_H that converges to GS within $\binom{n}{2}$ steps, given $I_t(g)$ is a ladder function of f_H.*

PROOF. $\min(I_t(g),GS)$ is a ladder function: since $I_t(g)$ and GS are both ladder functions we can invoke Lemma 4.1.

As the distance between any two graphs in $\mathcal{G}_n$ is no larger than $\binom{n}{2}$, we have that $\left\{g^*\mid d(g,g^*)\le\binom{n}{2}\right\}=\mathcal{G}_n$ holds for any graph $g\in\mathcal{G}_n$. Thus, the local sensitivity at distance $\binom{n}{2}$ equals GS, i.e.,

$$LS\left(g,\tbinom{n}{2}\right)=\max_{g^*|d(g,g^*)\le\binom{n}{2}} LS(g^*)=\max_{g^*\in\mathcal{G}_n} LS(g^*)=GS.$$

Given $LS(g,t)$ is the minimum ladder function and is monotonically increasing in t, it follows $I_t(g)\ge LS(g,t)\ge LS\left(g,\binom{n}{2}\right)=GS$ for any $t\ge\binom{n}{2}$. Hence, $\min(I_t(g),GS)=GS$ for any $t\ge\binom{n}{2}$. □

Recall that the time complexity of Algorithm 1 (except Line 1) is linear in M. Therefore we can conclude that the algorithm terminates in $O(n^2)$ time for any subgraph counting query, if a ladder function $I_t(g)$ is given in advance. Quadratic time can still be large, so in the next section, we find ladder functions for classes of subgraphs which converge in $O(n)$ steps.

5. APPLICATIONS

In this section, we show how to apply our framework for a variety of subgraph counting queries, including $f_{\triangle}, f_{k\star}, f_{k\mathbb{C}}$ and $f_{k\triangle}$. $LS(g,t)$ of the functions $f_{\triangle}$ and $f_{k\star}$ are carefully studied in [17,29], which can serve as ladder functions for these two queries directly. However, as we will show in this section, computing $LS(g,t)$ can be hard for some queries, e.g., $f_{k\mathbb{C}}$ and $f_{k\triangle}$. Instead of using $LS(g,t)$, we present an efficient method to build a convergent upper bound of $LS(g,t)$, which is shown to satisfy the requirements in Definition 4.1. Such an upper bound could be used as the ladder function for $f_{k\mathbb{C}}$ and $f_{k\triangle}$.

Graph statistics. Our detailed analysis of global sensitivity, local sensitivity and its upper bound rely on some simple graph statistics related to the (common) neighborhood of nodes.

DEFINITION 5.1 (GRAPH STATISTICS). *Let $(x_{ij})_{n\times n}$ be the adjacency matrix of an undirected, simple graph on n nodes. $x_{ij} = 1$ when there exists an edge between nodes i and j, 0 otherwise. Let d_i denote the degree of node i, and $d_m = \max_i d_i$ be the maximum degree of the graph.*

Let A_{ij} be the set of common neighbors of i and j. Node l belongs to A_{ij} if and only if $x_{il}x_{lj} = 1$. Let $a_{ij} = |A_{ij}|$ be the number of common neighbors of i, j, and $a_m = \max_{i,j} a_{ij}$ be the maximum number of common neighbors of a pair of nodes in the graph. Let $b_{ij} = d_i + d_j - 2a_{ij} - 2x_{ij}$ denote the the number of nodes connected to exactly one of the two nodes i, j.

5.1 $LS(g,t)$ as Ladder Function

Triangle counting. In Lemma 5.1, we give the global sensitivity and local sensitivity at distance t of triangle counting.

LEMMA 5.1 (CLAIM 3.13 OF FULL VERSION OF [29]). *The global sensitivity of $f_{\triangle}$ is $GS = n-2$; The local sensitivity at distance t of $f_{\triangle}$ is $LS(g,t) = \max_{i,j} LS_{ij}(g,t)$, where*

$$LS_{ij}(g,t) = \min\left(a_{ij} + \left\lfloor \frac{t + \min(t, b_{ij})}{2} \right\rfloor, n-2\right).$$

It is easy to prove that $LS(g,t)$ converges to GS when $t \geq 2n$. The time complexity of computing $LS(g,t)$ for $t \in [0,2n]$ is $O(M(n))$, where $M(n)$ is the time needed to multiply two $n \times n$ matrices.

k-star counting. Another important problem of subgraph counting studied in [5, 17] is k-star counting. Lemma 5.2 shows its global sensitivity and local sensitivity at distance t.

LEMMA 5.2 (LEMMA 3.4 OF [17]). *The global sensitivity of $f_{k\star}$ is $GS = 2\binom{n-2}{k-1}$; The local sensitivity at distance t of $f_{k\star}$ is $LS(g,t) = \max_{i,j} LS_{ij}(g,t)$, where*

$$LS_{ij}(g,t) = \begin{cases} \binom{\bar{d}_i+t}{k-1} + \binom{\bar{d}_j}{k-1}, & \text{if } t \leq B_i; \\ \binom{n-2}{k-1} + \binom{\bar{d}_j+t-B_i}{k-1}, & \text{if } B_i < t \leq B_i + B_j; \\ 2\binom{n-2}{k-1}, & \text{if } B_i + B_j < t. \end{cases}$$

Here $\bar{d}_i = d_i - x_{ij}$ and $B_i = n-2-\bar{d}_i$. $\bar{d}_j$ and B_j are defined analogously. Without loss of generality, assume that $d_i \geq d_j$.

As with $f_{\triangle}$, $LS(g,t)$ of $f_{k\star}$ converges to GS when $t \geq 2n$. It takes $O(n\log n + m)$ time to compute $LS(g,t)$ for $t \in [0,2n]$.

5.2 Customized Ladder Function

Although $LS(g,t)$ works well for triangle and k-star counting, it cannot be extended to the class of queries whose $LS(g,t)$ are NP-complete to compute, e.g., k-clique counting for a constant $k > 3$ (the reduction is shown for completeness in Appendix A) and k-triangle counting for $k > 1$ (proved in [17]). To address this problem, the authors of [17] propose to use a *differentially private* version of local sensitivity, to replace the inefficient $LS(g,t)$. However, this method is quite restricted because (i) it is specifically for k-triangle counting; (ii) it achieves (ε,δ)-differential privacy only, and ε is limited to $(0, \frac{3}{2}\ln\frac{3}{2} \approx 0.6]$. In this section, we illustrate how to avoid using $LS(g,t)$ by designing a new customized ladder function.

5.2.1 *k-clique counting*

We first emphasize that we are only interested in a small constant k here, otherwise the counting query $f_{k\mathbb{C}}$ itself is hard to compute. 1- and 2-clique counting are trivial in our setting, and 3-clique (triangle) counting is already well addressed in Lemma 5.1. For other constant $k > 3$, we aim to design a function $I_t(g)$ which satisfies all the requirements in Definition 4.1.

First of all, we introduce some building blocks of $I_t(g)$, i.e., a_m (see Definition 5.1), and the global and local sensitivity of $f_{k\mathbb{C}}$ in the next lemma

LEMMA 5.3. *The global sensitivity of $f_{k\mathbb{C}}$ is $GS = \binom{n-2}{k-2}$; the local sensitivity of $f_{k\mathbb{C}}$ is*

$$LS(g) = \max_{i,j} \mathbb{C}(g(A_{ij}), k-2),$$

where $g(S)$ denotes the subgraph induced on g by the node subset S, and $\mathbb{C}(g,k)$ is the number of k-cliques in graph g.

The global sensitivity is achieved when deleting one edge from a complete graph with n nodes. The local sensitivity at (i,j), i.e., $\mathbb{C}(g(A_{ij}), k-2)$, indicates the number of k-cliques that contain nodes i and j when edge (i,j) is added.

Next we give our ladder function for k-clique counting in Theorem 5.1 and explain the intuition of this design in the proof.

THEOREM 5.1.

$$I_t(g) = \min\left(LS(g) + \binom{a_m+t}{k-2} - \binom{a_m}{k-2}, GS\right)$$

is a ladder function for $f_{k\mathbb{C}}$.

PROOF. By Lemma 4.1, we learn that proving $LS(g) + \binom{a_m+t}{k-2} - \binom{a_m}{k-2}$ is a ladder function is sufficient to prove this theorem. The proof contains the following three steps.

(1) $LS(g) \leq I_0(g)$, for any g. This step is trivial since $I_0(g) = LS(g)$;

(2) $I_0(g') \leq I_1(g)$, for any pair of neighboring graphs g, g'. Let A_{ij} (A'_{ij}) be the set of common neighbors of nodes i,j in g (g'). Let a_m (a'_m) be maximum number of common neighbors in g (g'). Note that the size of A_{ij} is bounded by a_m.

$$\begin{aligned} LS(g') - LS(g) &= \max_{i,j} \mathbb{C}(g'(A'_{ij}), k-2) - \max_{i,j} \mathbb{C}(g(A_{ij}), k-2) \\ &\leq \max_{i,j} \left(\mathbb{C}(g'(A'_{ij}), k-2) - \mathbb{C}(g(A_{ij}), k-2)\right). \end{aligned}$$

Recall that there is only one edge difference between g and g'. To increase the number of $(k-2)$-cliques in $g(A_{ij})$ by editing one edge, one can either (i) add one edge within $g(A_{ij})$ while keeping the set A_{ij} unchanged, or (ii) expand A_{ij} by one new node. The maximum increase, i.e., $\binom{a_m}{k-3}$ is achieved in the case that $g(A_{ij})$ is a complete graph of size a_m and $g'(A'_{ij})$ is a complete graph of size $a_m + 1$. However, observe that since

$$\binom{a_m}{k-3} = \binom{a_m+1}{k-2} - \binom{a_m}{k-2}$$

we have

$$LS(g') \le LS(g) + \binom{a_m+1}{k-2} - \binom{a_m}{k-2} \quad \Leftrightarrow \quad I_0(g') \le I_1(g).$$

(3) $I_t(g') \le I_{t+1}(g)$, for any neighboring graphs g, g' and integer $t > 0$. We make use of the combinatorial identity $\sum_{k=0}^{n} \binom{k}{m} = \binom{n+1}{m+1}$ and obtain

$$\begin{aligned} LS(g') + \binom{a'_m+t}{k-2} - \binom{a'_m}{k-2} &= LS(g') + \sum_{i=0}^{t-1} \binom{a'_m+i}{k-3} \\ \le LS(g) + \binom{a_m}{k-3} &+ \sum_{i=0}^{t-1} \binom{a_m+i+1}{k-3} \\ = LS(g) + \sum_{i=0}^{t} \binom{a_m+i}{k-3} &= LS(g) + \binom{a_m+t+1}{k-2} - \binom{a_m}{k-2}, \end{aligned}$$

which is equivalent to $I_t(g') \le I_{t+1}(g)$. The central step of the proof relies on an important property of the maximum number of common neighbors: its global sensitivity equals 1. So it holds that $a'_m - a_m \le 1$, and so $\binom{a'_m+i}{k-3} \le \binom{a_m+i+1}{k-3}$. $\square$

This ladder function converges to GS when $t \ge n$. The major computational overhead of $I_t(g)$ for $t \in [0,n]$ is $LS(g)$, which can be computed within $O(n^k)$ time via exhaustive search. In our experiment, we implement a sophisticated algorithm which utilizes the sparsity of the input to improve efficiency. It returns $LS(g)$ within a few seconds for all graphs tested in the next section.

5.2.2 *k-triangle counting*

Besides k-clique counting, we also present k-triangle counting as another example of using customized ladder function. We state the currently known results about k-triangle counting as Lemma 5.4.

LEMMA 5.4 (LEMMA 4.1 AND 4.2 OF [17]). *The global sensitivity of $f_{k\triangle}$ is $GS = \binom{n-2}{k} + 2(n-2)\binom{n-3}{k-1}$. The local sensitivity of $f_{k\triangle}$ is*

$$LS(g) = \max_{i,j} \left(\binom{a_{ij}}{k} + \sum_{l \in A_{ij}} \binom{a_{il} - x_{ij}}{k-1} + \binom{a_{lj} - x_{ij}}{k-1} \right);$$

We also have $LS(g') \le LS(g) + 3\binom{a_m}{k-1} + a_m\binom{a_m}{k-2}$, given a pair of neighboring graphs g and g'.

Similarly, one can design a ladder function for k-triangle counting, as shown in Theorem 5.2. The proof follows the same lines as that of Theorem 5.1.

THEOREM 5.2.

$$I_t(g) = \min \left(LS(g) + \sum_{i=0}^{t-1} U(a_m + i), GS \right)$$

is a ladder function for $f_{k\triangle}$, where $U(a) = 3\binom{a}{k-1} + a\binom{a}{k-2}$.

It takes $t \ge 3n$ steps for this ladder function to converge to GS, and the time complexity of computing $I_t(g)$ for $t \in [0, 3n]$ would be $O(n^3)$ using a naive method.

To summarize, we observe some similarities between k-clique and k-triangle counting. First, their $LS(g,t)$ functions are both NP-complete to compute, which motivates our customized ladder functions. Second, there exist efficient ways to compute the local sensitivities $LS(g)$, and $LS(g') - LS(g)$ is bounded by a function of a variable whose global sensitivity is constant, e.g., a_m or d_m. Any subgraph counting query with these two properties could be processed by our framework with a customized ladder function as in Theorems 5.1 and 5.2.

6. EXPERIMENTS

6.1 Experimental Settings

Datasets. We make use of six real-world graph datasets in our experiments: AstroPh, HepPh, HepTh, CondMat and GrQc are collaboration networks from the e-print arXiv, which cover scientific collaborations between authors who submitted papers to Astro Physics, High Energy Physics, High Energy Physics Theory, Condensed Matter and General Relativity categories, respectively. In particular, if a paper is co-authored by k authors, the network generates a completed connected subgraph (k-clique) on k nodes representing these authors. Enron is an email network obtained from a dataset of around half a million emails. Nodes of the network are email addresses and if an address i sent at least one email to address j, the graph contains an undirected edge from i to j. Tabel 2 illustrates the properties of the datasets and their results to four subgraph counting queries. All datasets can be downloaded from Stanford Large Network Dataset Collection [21].

Baselines. To justify the performance of our algorithm (denoted as Ladder) in answering subgraph counting queries, we compare it with other four approaches: (i) Laplace [11], which directly injects Laplace noise with scale GS/ε to the true subgraph counts. (ii) Smooth [17, 29], which first computes a smooth upper bound SS of the local sensitivity, then adds Cauchy noise scaled up by a factor of $6SS/\varepsilon$ to the answer. It is only evaluated on two queries $f_\triangle$ and $f_{k\star}$ due to the hardness of computing SS for $f_{k\mathbb{C}}$ and $f_{k\triangle}$. (iii) NoisyLS [17], which achieves differential privacy by adding noise proportional to a *differentially private* upper bound on local sensitivity (instead of the smooth upper bound used in Smooth). This approach is designed for $f_{k\triangle}$, and is the only (ε, δ)-differentially private baseline algorithm. The parameter ε is restricted to $(0, \frac{3}{2}\ln\frac{2}{3}]$ while δ is set to a fixed value 0.01 in our experiments. (iv) Recursive [5], which answers the query by releasing a differentially private lower bound (which has low-sensitivity) of the counting result. The time complexity of this algorithm is super-quadratic to the number of subgraphs. All experiments using the Recursive mechanism failed to complete within 4 days except for two cases: HepTh-$\triangle$ and GrQc-$\triangle$ since their counting results are relatively small. In contrast, all local sensitivity based solutions (i.e., Ladder, Smooth, NoisyLS) are rather efficient, and in fact share the same time complexity.

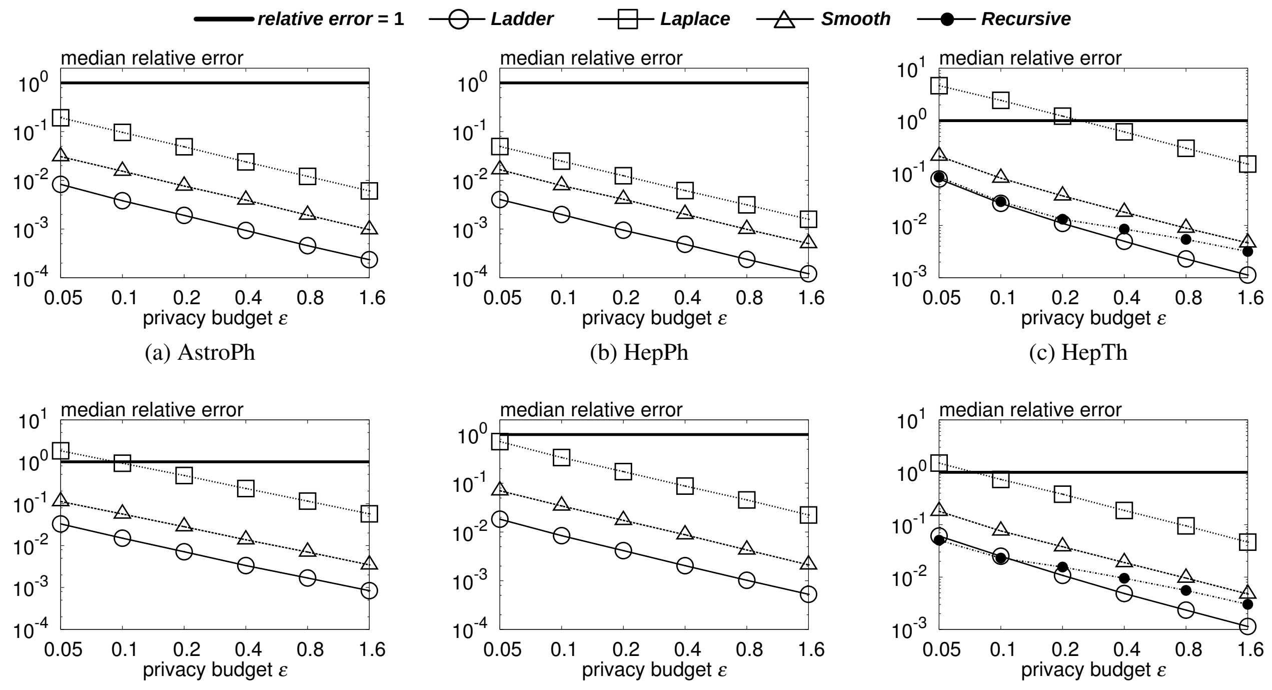
Evaluation. We evaluate the performance of Ladder and four baselines on four subgraph counting queries over all six datasets. We measure the accuracy of each method by the *median relative error* [17], i.e., the median of the random variable $|\mathcal{A}(g) - f(g)|/f(g)$ where $\mathcal{A}(g)$ is the differentially private output and $f(g)$ is the true answer. The reason of choosing this measurement is that the mean of Cauchy noise (used in Smooth) is undefined. In other words, the mean error of the Smooth method is never stable, no matter how large the sampling set is. Therefore, in accordance with prior works [5, 17], we choose to report median error. We also include a bold line to indicate a relative error of 1. Any result above this line is of no practical value. For each result reported, we repeat each experiment 10,000 times to get the median, except for Recursive (where we perform 100 repetitions).

6.2 Counting Queries

Figures 4-7 show all experimental results on counting queries, varying the privacy budget ε from 0.05 to 1.6. The title of each subfigure (e.g., AstroPh) indicates the reported dataset.

Table 2: Datasets properties

datasets	nodes	edges	avg. deg.	density	$\triangle$	$3\star$	$4\mathbb{C}$	$2\triangle$
AstroPh	18,772	198,050	21.101	0.1124%	1,351,440	545,677,550	9,580,415	72,355,715
HepPh	12,008	118,491	19.735	0.1644%	3,358,499	1,276,967,000	150,281,372	936,890,335
HepTh	9,877	25,975	5.260	0.0533%	28,339	2,098,336	65,592	429,013
CondMat	23,133	93,439	8.078	0.0349%	173,361	37,115,060	294,008	2,349,650
Enron	36,692	183,831	10.020	0.0273%	727,044	4,909,606,844	2,341,639	36,528,276
GrQc	5,242	14,485	5.527	0.1054%	48,260	2,482,748	329,297	2,041,499



(a) AstroPh (b) HepPh (c) HepTh

(d) CondMat (e) Enron (f) GrQc

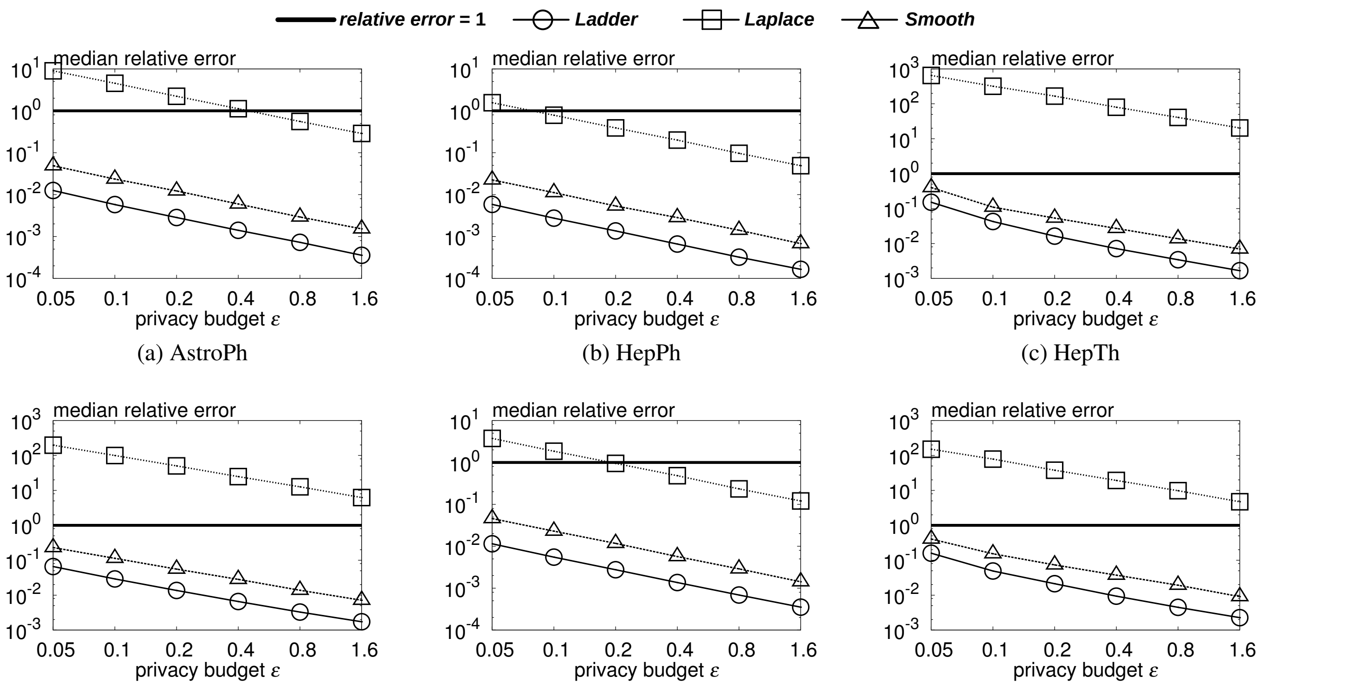
Figure 4: Triangle counting

Triangle counting. The first query that we evaluate is the triangle counting $f_\triangle$ (see Figure 4). In summary, Ladder achieves good accuracy on $f_\triangle$ over all datasets. When privacy budget is relatively large, e.g., $\varepsilon = 1.6$, its median relative error always stays below or close to 0.1%. With the decrease of ε, the accuracy drops but it is still smaller than 10% even when $\varepsilon = 0.05$. Compared with other differentially private methods, Ladder clearly outperforms Laplace and Smooth in all cases, simply because it injects less noise to the true results. The improvement is significant since the y-axis in shown on a log scale. As for Recursive, it is rather competitive when ε is small, say $\varepsilon \leq 0.1$. However, the gap between Recursive and Ladder begins to grow when ε increases. To explain, recall that the error of Recursive comes from two parts: bias of the lower bound and noise injected to the lower bound. As ε increases, the scale of noise reduces, while the bias is less sensitive to the change of ε then becomes the dominating factor of the error.

k-**star counting.** Next we evaluate different methods for 3-star counting $f_{3\star}$ in Figure 5. Ladder keeps returning extremely accurate results for large ε, and reasonably good results for small ε. Meanwhile, it is still the best solution with an ε-differential privacy guarantee in all settings. In contrast to the case of triangle counting, the performance of Laplace degrades significantly compared to other two local sensitivity based solutions. The main reason is that *GS* of triangle counting is linear to *n* while that of 3-star counting is quadratic. On the other hand, the counting results do not increase so dramatically due to the locality of inputs. Therefore the relative error of Laplace increases. Local sensitivity, however, can capture the locality of inputs, leading to a stable relative error when the query is changed.

k-**clique counting.** Figure 6 presents the results of 4-clique counting ($f_{4\mathbb{C}}$). Ladder is the only private solution besides the naive Laplace, and the former is orders of magnitude better than the latter in terms of accuracy. The lines of Ladder always stay below the bold lines except for two points where ε is extremely small, and the gaps increase markedly with ε. In contrast, the quality of results from Laplace tends to be rather poor, if it is usable at all. Thus we conclude that Ladder is the only effective and efficient algorithm for releasing private *k*-clique counting.

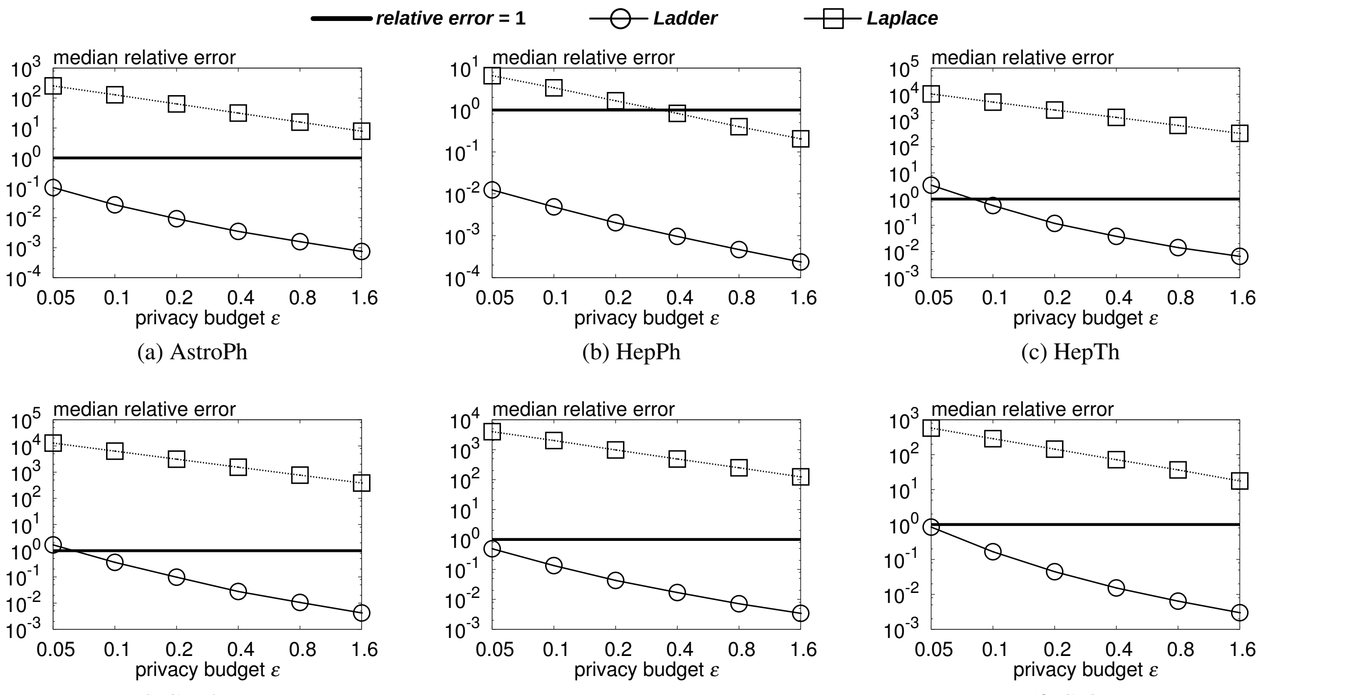
k-**triangle counting.** The last query of interest in our experiment is 2-triangle counting $f_{2\triangle}$. We have a new baseline NoisyLS for this query and compare it with Ladder in Figure 7. The superiority of Ladder is three-fold: (i) it is always more accurate than NoisyLS when ε varies from 0.05 to 0.4; (ii) it has no extra constraint on privacy budget ε; (iii) it provides stronger privacy protection than



(a) AstroPh (b) HepPh (c) HepTh

(d) CondMat (e) Enron (f) GrQc

Figure 5: 3**-star counting**



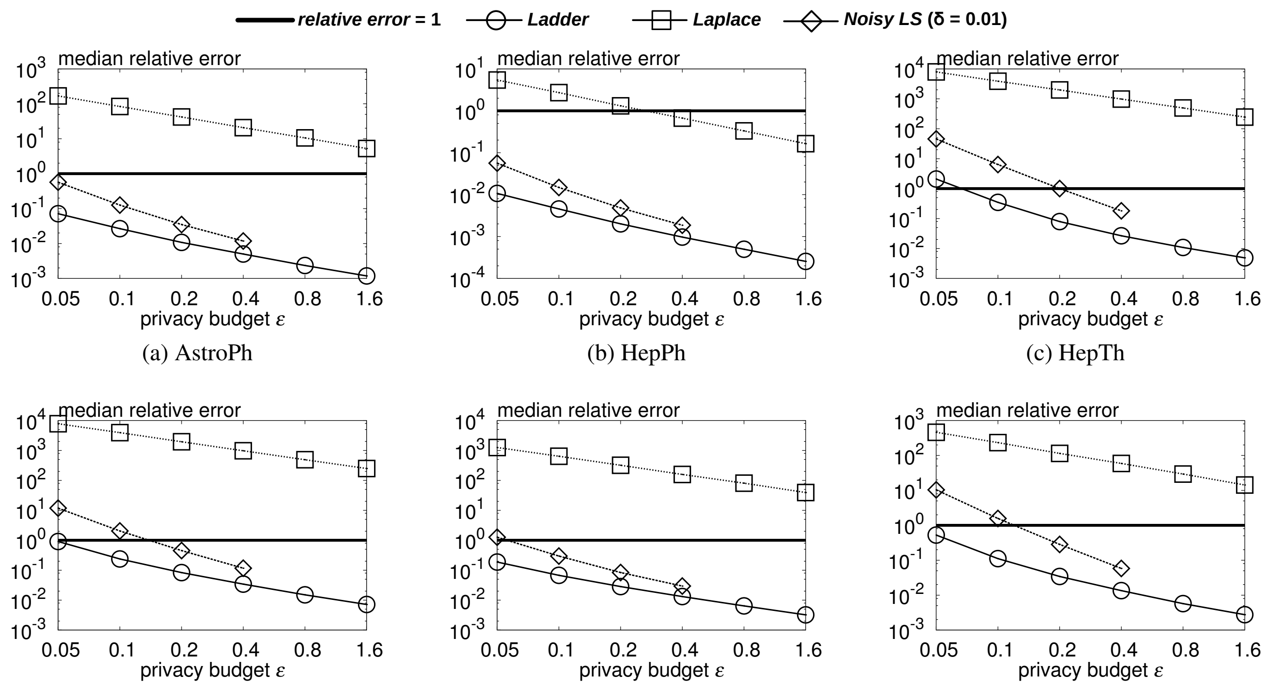
(a) AstroPh (b) HepPh (c) HepTh

(d) CondMat (e) Enron (f) GrQc

Figure 6: 4**-clique counting**

NoisyLS. In summary, Ladder is a more preferable solution for private k-triangle counting.

Average degree and density. Besides the comparison among private algorithms, we are also interested in the impact of input graphs to the private counting results. Intuitively, a graph with more edges (assuming a fixed number of nodes) is likely to have more copies of subgraphs and a larger local sensitivity, but the global sensitivity is never a function of number of edges. Thus, we can expect relatively good performance of Laplace on denser graphs. Nevertheless, the impact of graph density to local sensitivity is still unclear. From Table 2, we can learn that AstroPh and HepPh are denser graphs with higher average degree, on which Laplace does



(a) AstroPh (b) HepPh (c) HepTh

(d) CondMat (e) Enron (f) GrQc

Figure 7: 2-triangle counting

have better overall accuracy. Interestingly, local sensitivity based algorithms like Ladder and Smooth also benefit notably from high density inputs, implying that local sensitivity does not increase as drastically as the counting results as the graph gets denser.

6.3 Stochastic Kronecker Models

To further justify the importance of subgraph counts, we show how accurate counts lead to accurate graph models. Here we adopt the stochastic Kronecker model [20], a sophisticated method which can simulate large real-world graphs with only a few parameters. The algorithm in [13, 26] provides a way to estimate parameters of the simplest Kronecker model from a small set of subgraph counts: f_e (the number of edges), $f_\triangle$, $f_{2\star}$ and $f_{3\star}$. Therefore, one can easily build a Kronecker model then generate synthetic Kronecker graphs with differential privacy guarantees, if provided four *private* counts in advance. We aim to evaluate how the noise in private counts affects the accuracy of Kronecker models. In this set of experiments, we build three types of Kronecker models: (i) Kron-True is a non-private model, and tuned from the true subgraph counts of the graph; (ii) Kron-Ladder satisfies ε-differential privacy. The privacy budget ε is split into four parts equally, and each is used to generate a private subgraph count. The Laplace mechanism is effective enough to release f_e since it has a constant $GS = 1$. For $f_\triangle$, $f_{2\star}$ and $f_{3\star}$, we employ Ladder. (iii) Kron-Smooth is identical to Kron-Ladder except that $f_\triangle$, $f_{2\star}$ and $f_{3\star}$ are released by the Smooth mechanism.

To measure the distortion of Kron-Ladder from Kron-True, we look into the difference of their synthetic graphs. For Kron-True, we simply sample 100 synthetic graphs from the model. However, given that Ladder is a stochastic process, we first generate 100 independent Kron-Ladder models, and from each we sample 100 graphs. Therefore, Kron-Ladder is represented by a set of $10,000$ graphs. Then the error between each pair of graphs from these two sets is measured, and an aggregate value is reported as the final result. The evaluation of Kron-Smooth follows the same procedures. We report the error of five graph queries including median relative error of f_e, $f_\triangle$, $f_{2\star}$, $f_{3\star}$ and average Euclidian error of the sorted degree sequence (as in [15]).

Table 3 shows the empirical results with two datasets (Enron and GrQc) and two privacy budgets ($\varepsilon = 0.2$ and 1.6). Kron-Ladder outperforms Kron-Smooth by considerable margins in all settings, which implies that Kron-Ladder generates more accurate Kronecker models. It is interesting to observe that the synthetic graphs of Kron-Ladder perform better in answering f_e (recall that both methods use Laplace mechanism for f_e). Note that the parameter estimation algorithm [13, 26] is designed to find a Kronecker model that fits *all* counts well. Hence the noise in any count could affect the choice of Kronecker model, then introduce bias to other properties of the synthetic graphs. This explains how the large scale of noise in $f_\triangle$, $f_{2\star}$ and $f_{3\star}$ propagates to f_e in Kron-Smooth.

7. RELATED WORK

The question of being able to release information on graphs in a privacy-preserving way has been of interest for a number of years, driven by recent efforts around privacy-preserving data publishing. It is interesting to note that within the computer science community, the recent slew of interest in graph anonymization began with publications that emphasized the difficulty of this problem. Backstrom *et al.* [2] showed that an adversary could easily learn neighborhood information of targeted nodes from a "deidentified" (unlabeled) released graph, if they could insert a small number of edges. Hay *et al.* [16] showed that the neighborhood around a node in an unlabeled graph is often sufficiently distinctive to identify it. Narayanan and Shmatikov [27, 28] "weaponized" these observations into effective attacks able to extract sensitive link information from deidentified and lightly perturbed graphs—most famously, by linking individuals from the Netflix prize data set to public IMDB profiles.

Table 3: Accuracy of Kronecker models tuned from noisy counts

datasets	methods	$\varepsilon = 0.2$					$\varepsilon = 1.6$				
		f_e	$f_\triangle$	$f_{2\star}$	$f_{3\star}$	deg. seq.	f_e	$f_\triangle$	$f_{2\star}$	$f_{3\star}$	deg. seq.
Enron	Kron_Ladder	0.306%	0.595%	0.511%	0.691%	0.4628	0.043%	0.083%	0.071%	0.096%	0.4388
	Kron_Smooth	1.347%	2.791%	2.394%	3.247%	1.2383	0.171%	0.331%	0.285%	0.385%	0.6027
GrQc	Kron_Ladder	2.195%	9.849%	4.490%	14.52%	0.9048	0.187%	0.728%	0.298%	1.061%	0.3993
	Kron_Smooth	5.908%	24.32%	11.56%	31.70%	1.5223	0.854%	3.088%	1.620%	4.469%	0.5744

The response of the research community to these demonstrations of the weakness of deidentification was the design of numerous methods which aimed to be more robust to attack. Drawing inspiration from concurrent efforts on k-anonymity for tabular data, several approaches aimed to modify the input graph structure to produce an output graph that was not susceptible to the techniques used to defeat deidentification. Of particular note was the work of Liu and Terzi [23] which aimed to make graphs *k-degree anonymous*, so that an adversary possessed of the knowledge of the degree of a target node would find at least k nodes in the released graph sharing that degree. Many subsequent works built on this foundation by proposing stronger structural requirements on the released graph. Zhou and Pei [37] considered graphs with nodes of certain types, and sought to ensure that for every set of neighborhood links around a target node, there are multiple possible matches in the released graph. Zou *et al.* [38] proposed *k-automorphism*, requiring that every node in the released graph has at least $k-1$ structurally indistinguishable counterparts (i.e. have the same pattern of links between neighbors, and neighbors-of-neighbors, etc.). A parallel set of works were based on “clustering” within graphs, where groups of nodes are replaced by a “cluster” in the output graph, and only aggregate statistics of the cluster are revealed [6, 16, 34].

A survey of the state of the art in 2008 provides a snapshot of several dozen contributions in this area; in the intervening years that have been many more efforts in this direction. Nevertheless, a number of criticisms of this philosophy towards data anonymization have emerged. Principal amongst these is that methods which seek to prevent a specific attack by an adversary with a certain type of information and an assumed approach to reasoning are not robust against “real-world” adversaries, who can bring unanticipated knowledge and inference techniques to circumvent these. Secondarily, there is the perception that these methods provide only ad hoc rather than principled guarantees, and do not adequately preserve utility for their inputs. Instead, there has been growing interest in ways to provide a more formal guarantee of privacy. The main such model is Differential Privacy, which is not without its critics or identified limitations, but which nevertheless represents the current focus of efforts to provide workable privacy solutions.

Differential privacy [7] is rigorously formulated based on statistics and offers an extremely strong privacy protection. A plethora of efforts has been made to apply differential privacy to a broad range of problems. It is beyond the scope of this paper to give full coverage of all differential privacy methods; instead, see recent surveys [8,9,12,30] and references therein. A brief survey of differentially private algorithms for releasing subgraph counts is given in Section 1.2. In what follows, we review some other related efforts to realize privacy protection for graph data.

Hay et al. [15] translated the language of differential privacy to the graph context, and give the formal definitions of edge and node differential privacy. This paper and subsequent related approaches [18, 22] provide effective solutions to releasing the degree sequence of a sensitive graph, and use some sophisticated post-processing techniques to reduce noise. Proserpio et al. [31] and Sala et al. [33] further extend the problem to higher-order joint degree sequences, which model more detailed information within the sensitive graph. A later work by Wang and Wu [35] improves Sala’s solution by calibrating noise based on smooth sensitivity, rather than the large global sensitivity. Moreover, the authors illustrate that synthetic graphs generated from the noisy joint degree sequence have a series of appealing properties. In addition, Mir and Wright [26] show how to bridge differentially private subgraph counts with Kronecker graph models, providing another method to generate synthetic graphs with privacy guarantees. In a different vein, [1, 14, 36] investigate differentially private spectral analysis (e.g., singular value decomposition and random projection) for graph data. Their results imply a great potential to process/release large scale sensitive graphs like the Facebook friendship graph and call/SMS/email networks. Recently, Lu and Miklau [24] describe the use of a chain mechanism to release alternating k-star and k-triangle counting with an (ε, δ)-differentially private guarantee. They also show how the published counts can be used to estimate the parameters of exponential random graph models.

The above discussion principally focuses on methods that provide edge differential privacy, where two graphs are considered neighbors if they differ only in the presence of a single edge. There is also a branch of work studying other variants of graph differential privacy. For example, [3, 5, 19] are the first few methods to provide *node* differential privacy with non-trivial utility guarantees. Node differential privacy is a qualitatively stronger privacy guarantee than edge differential privacy, where two graphs are considered neighbors if they differ in the presence of a node and all its incident edges. As a consequence, it generally requires substantially more noise to protect the released information. Meanwhile, Rastogi et al. [32] consider a relaxed version of edge differential privacy, called edge *adversarial* privacy, which relies on some assumptions about prior knowledge of the malicious user. As those assumptions might plausibly be violated in practice, this new definition actually provide less privacy assurance, compared to the conventional edge differential privacy.

8. CONCLUDING REMARKS

The problem of releasing information about private data is a timely one, and there continues to be pressures to make relevant information available in a privacy-respecting fashion. We have shown that the widely-respected differential privacy guarantee can be obtained for graph statistics, specifically subgraph counts, in a way that is efficient and accurate. Future work includes extending the solution to a large scale graph like Facebook. Moreover, the ladder method developed, although tailored for subgraph counts, can be applied more generally, to functions outside the domain of graphs. The main challenge in lifting the notion to other domains is to ensure that concepts such as local sensitivity at distance t are well-defined. It will be natural to study the usefulness of the ladder framework for other structured mathematical objects, such as matrix representations of data, as well as special cases of graphs, such as trees or directed acyclic graphs.

Acknowledgments

This work was supported in part by European Commission Marie Curie Integration Grant PCIG13-GA-2013-618202, by Nanyang Technological University under SUG Grant M4080094.020, by Microsoft Research Asia under a gift grant, by the Ministry of Education (Singapore) under an AcRF Tier-2 grant, and by AT&T under an unrestricted gift grant.

9. REFERENCES

[1] F. Ahmed, R. Jin, and A. X. Liu. A random matrix approach to differential privacy and structure preserved social network graph publishing. *CoRR*, abs/1307.0475, 2013.

[2] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, pages 181–190, 2007.

[3] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96. ACM, 2013.

[4] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recogn. Lett.*, 18(9):689–694, Aug. 1997.

[5] S. Chen and S. Zhou. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *SIGMOD*, pages 653–664, 2013.

[6] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. *VLDB J.*, pages 115–139, 2010.

[7] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[8] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

[9] C. Dwork. The differential privacy frontier (extended abstract). In *TCC*, pages 496–502, 2009.

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

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

[12] C. Dwork and A. Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2):2, 2010.

[13] D. F. Gleich and A. B. Owen. Moment-based estimation of stochastic kronecker graph parameters. *Internet Mathematics*, 8(3):232–256, 2012.

[14] M. Hardt and A. Roth. Beyond worst-case analysis in private singular vector computation. In *STOC*, 2013.

[15] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178, 2009.

[16] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and C. Li. Resisting structural re-identification in anonymized social networks. *VLDB J.*, pages 797–823, 2010.

[17] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev. Private analysis of graph structure. *TODS*, 39(3):22, 2014.

[18] V. Karwa and A. B. Slavković. Differentially private graphical degree sequences and synthetic graphs. In *Privacy in Statistical Databases*, pages 273–285. Springer, 2012.

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

[20] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

[21] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[22] B.-R. Lin and D. Kifer. Information preservation in statistical privacy and bayesian estimation of unattributed histograms. In *SIGMOD*, pages 677–688, 2013.

[23] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD*, pages 93–106, 2008.

[24] W. Lu and G. Miklau. Exponential random graph estimation under differential privacy. In *KDD*, pages 921–930, 2014.

[25] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[26] D. Mir and R. N. Wright. A differentially private estimator for the stochastic kronecker graph model. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 167–176. ACM, 2012.

[27] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

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

[29] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.

[30] M. M. Pai and A. Roth. Privacy and mechanism design. *ACM SIGecom Exchanges*, 12(1):8–29, 2013.

[31] D. Proserpio, S. Goldberg, and F. McSherry. Calibrating data to sensitivity in private data analysis. *PVLDB*, 7(8), 2014.

[32] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: output perturbation for queries with joins. In *PODS*, pages 107–116, 2009.

[33] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In *IMC*, pages 81–98. ACM, 2011.

[34] T. Tassa and D. Cohen. Anonymization of centralized and distributed social networks by sequential clustering. *TKDE*, 25(2), 2013.

[35] Y. Wang and X. Wu. Preserving differential privacy in degree-correlation based graph generation. *Trans. Data Privacy*, 6(2):127–145, Aug. 2013.

[36] Y. Wang, X. Wu, and L. Wu. Differential privacy preserving spectral graph analysis. In *Advances in Knowledge Discovery and Data Mining*, pages 329–340. 2013.

[37] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, pages 506 –515, 2008.

[38] L. Zou, L. Chen, and M. T. Özsu. K-automorphism: A general framework for privacy preserving network publication. In *PVLDB*, pages 946–957, 2009.

APPENDIX

A. NP-COMPLETENESS

THEOREM A.1. *Computing $LS(g,t)$ of $f_{k\mathbb{C}}$ for a constant $k > 3$ is NP-complete.*

PROOF. We first prove that it is NP-complete to compute $LS_{ij}(g,t)$ given a particular pair of nodes (i,j), to which the well-known Clique Problem is polynomial-time reducible.

Given an instance of Clique Problem $\langle g,t\rangle$ (i.e., whether a given graph g has a clique of at least size t), we extend g to g^+ by adding two extra nodes i and j. Then we make i connected to all other nodes in g^+ while keeping j isolated (except i). This construction can be done in time that is polynomial in the size of Clique Problem instance. Next, we prove that

$$LS_{ij}(g^+,t) = \binom{t}{k-2} \Leftrightarrow g \text{ contains a } t\text{-clique},$$

where k can be set to any *constant* integer within $(3,t)$.

Recall from Lemma 5.3 that local sensitivity of $f_{k\mathbb{C}}$ over (i,j) is the number of $(k-2)$-cliques in the subgraph induced by their common neighbors A_{ij}. In g^+, A_{ij} is empty since j is isolated from g. After t modification steps on the graph, the size of A_{ij} could reach t if we connect j to t nodes in g (i is already connected to all nodes in g); and the maximum number of $(k-2)$-cliques, i.e., $\binom{t}{k-2}$, is achieved if and only if new neighbors of i and j form a t-clique. Therefore, it implies the existence of a t-clique in g.

We remark that this proof does not apply in the case of $k = 3$ (triangle counting), since the t-clique is not the only subgraph of size t that maximizes the number of 1-cliques (nodes). Indeed, computing $LS_{ij}(g,t)$ of triangle counting is polynomial-time solvable as shown by Lemma 5.1.

Last, we extend the result to $LS(g,t)$ through a reduction from the same NP-complete problem, Clique Problem. For any instance $\langle g,t\rangle$ where g has n nodes, we construct g^+ from g by adding a n-clique and a hub node i which is connected to all other nodes in g^+. Then we show that

$$LS(g^+,t) = \binom{n-1}{k-2} + \binom{t}{k-2} \Leftrightarrow g \text{ contains a } t\text{-clique}.$$

Let j be an arbitrary node in the n-clique. It is easy to prove that $LS(g^+,t) = LS_{ij}(g^+,t)$. Therefore, $LS(g^+,t)$ equals the number of $(k-2)$-cliques in the subgraph of A_{ij}. At the beginning, subgraph of A_{ij} is a $(n-1)$-clique. To extend it, one can use one modification to either (i) add a node of g to A_{ij} by connecting it to j, or (ii) add an edge into the subgraph of A_{ij}. The optimum is only attained by adding t nodes of g to A_{ij}, who form a t-clique in g. Thus, the new subgraph of A_{ij} contains two separated components: a $(n-1)$-clique and a t-clique, and it has $\binom{n-1}{k-2} + \binom{t}{k-2}$ different $(k-2)$-cliques. □

B. PROOF OF THEOREM 3.1

In what follows, we rewrite the proof of Theorem 6 of [25] using the concepts and notations developed in this paper.

PROOF. The first step is to observe that Algorithm 1 draws from a probability distribution over the integers where the probability associated with output k is proportional to $\exp\left(\varepsilon q(g,k)/2\Delta_q\right)$, normalized by the sum of this quantity over all integers. Note that we do not compute this probability directly; instead, for the sake of efficiency, we group together all integers which share the same probability (in our terminology, which are on the same rung) for

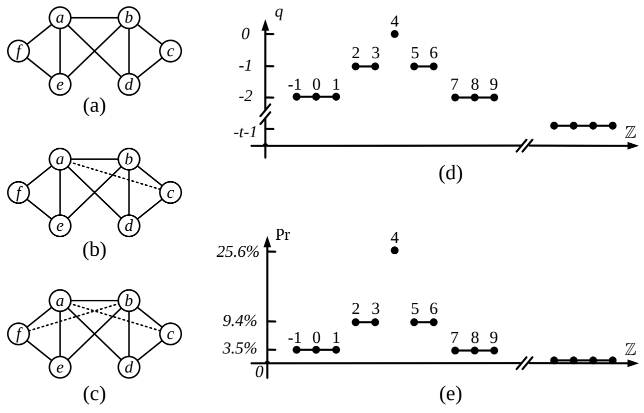


Figure 8: An end-to-end example

the rungs 0 to M. Additionally, we compute the aggregate probability for all outputs on rungs below M as a single value captured by $M+1$, as described after the description of the algorithm. It is straightforward to check that the actions of Algorithm 1 in the creation of the *weight* array give these sums of (unnormalized) probabilities. The action of the algorithm can then be seen as following two steps: first, we select a rung of the ladder (where rung $M+1$ is considered as a special case) according to the relative value of the *weight* of the rung. Then, we pick an integer from the corresponding rung. For rungs 0 to M, this is simple: all integers on the rung have the same probability, so we just pick one uniformly. For rung $M+1$, which corresponds to an unbounded tail of possible outputs, we take advantage of the structure to first pick how many further rungs down the ladder to go, and then to again pick uniformly from these. This gives the desired probability distribution of producing each possible value.

Hence as argued above, the output probability distribution of Algorithm 1 at integer k is equal to

$$\Pr[k] = \frac{\exp\left(\frac{\varepsilon}{2\Delta_q}\cdot q(g,k)\right)}{\sum_{k\in\mathbb{Z}}\exp\left(\frac{\varepsilon}{2\Delta_q}\cdot q(g,k)\right)}. \quad (7)$$

Now if the input g is replaced by its neighboring graph g', the quality of k will be changed by at most Δ_q, i.e., $|q(g,k) - q(g',k)| \le \Delta_q$. Therefore, the numerator of (7) can change at most by a factor of $\exp(\varepsilon\Delta_q/2\Delta_q) = \exp(\varepsilon/2)$ and the denominator minimum by a factor of $\exp(-\varepsilon/2)$. Thus the ratio of the new probability of k (with g') and the original one is at most $\exp(\varepsilon)$. □

C. AN END-TO-END EXAMPLE

In this section, we review the end-to-end workflow of our algorithm with an example input in Figure 8(a). The query in use is the triangle counting $f_\triangle$. Our algorithm consists of the following steps:

1. Compute the number of triangles $f_\triangle(g) = 4$, i.e., $\triangle_{abd}$, $\triangle_{abe}$, $\triangle_{bcd}$ and $\triangle_{aef}$.

2. Generate the ladder function of query $f_\triangle$, input g and step t, i.e., $I_t(g)$, for each $t \in [0,M]$. We describe the definition and key properties of ladder functions in Section 4, then illustrate how to compute ladder functions for specific queries in Section 5. For triangle counting query, we use its local sensitivity at distance t as the ladder function. $I_0(g) = LS(g,0) = 2$ achieved by removing edge

(a,b) from the graph. $LS(g,1)$ allows one extra change before calculating LS. In Figure 8(b), we show how the dashed edge (a,c) affects LS_{ab}, making $LS(g,1) = LS_{ab}(g,1) = 3$. Then Figure 8(c) shows an example achieving $LS(g,2) = 4$, where the value converges to the global sensitivity. Therefore, this step terminates with $M = 2$.

3. Build the ladder quality function $q(g,k)$ for $k \in \mathbb{Z}$, by combining the true answer $f_{\triangle}(g)$ and ladder functions $I_t(g)$; see Definition 3.1 of Section 3.1. Figure 8(d) gives a part of the quality function q over the integer domain. The true answer $k = 4$ takes the maximum value 0, which is the center of the function. Next to it are two first rungs, each consisting of $I_0(g) = 2$ integers on one side of the center. All integers on first rungs are assigned quality score -1. The next rung is formed of the next $I_1(g) = 3$ integers on each side with quality -2 and so on. Note that every integer will be given a quality score.

4. Generate the output distribution from the exponential mechanism that takes $q(g,k)$ as quality function; see equation (3) of Section 2.1. As the distribution changes with the privacy budget ε, we fix ε to 2 in the following discussion. Now the sampling probability for $k = 4$ is proportional to $\exp\left(\varepsilon q(g,4)/2\Delta_q\right) = e^0$, given $\Delta_q = 1$ (see Theorem 4.2) and $q(g,4) = 0$. Similarly, the probability for $k = 5$ is proportional to e^{-1}. After normalization, we have $\Pr[k = 4] = 25.6\%$ and $\Pr[k = 5] = 9.4\%$, as shown in Figure 8(e).

5. Sample the distribution and return. Algorithm 1 and Section 3.2 are dedicated to detailing the efficient sampling technique.