Published in final edited form as:
Soc Netw Anal Min. 2012 ; : 395–402. doi:10.1109/ASONAM.2012.71.

On Learning Cluster Coefficient of Private Networks

Yue Wang,
University of North Carolina at Charlotte, USA, Tel.: +1-704-6878586, Fax: +1-704-687-4893

Xintao Wu,
University of North Carolina at Charlotte, USA, Tel.: +1-704-6878586, Fax: +1-704-687-4893

Jun Zhu, and
University of North Carolina at Charlotte, USA, Tel.: +1-704-6878586, Fax: +1-704-687-4893

Yang Xiang
Tongji University, Shanghai, China

Yue Wang: ywang91@uncc.edu; Xintao Wu: xwu@uncc.edu; Jun Zhu: jzhu16@uncc.edu; Yang Xiang: shxiangyang@tongji.edu.cn

Abstract

Enabling accurate analysis of social network data while preserving differential privacy has been challenging since graph features such as clustering coefficient or modularity often have high sensitivity, which is different from traditional aggregate functions (e.g., count and sum) on tabular data. In this paper, we treat a graph statistics as a function f and develop a divide and conquer approach to enforce differential privacy. The basic procedure of this approach is to first decompose the target computation f into several less complex unit computations $f_1, \ldots, f_m$ connected by basic mathematical operations (e.g., addition, subtraction, multiplication, division), then perturb the output of each f_i with Laplace noise derived from its own sensitivity value and the distributed privacy threshold ε_i, and finally combine those perturbed f_i as the perturbed output of computation f. We examine how various operations affect the accuracy of complex computations. When unit computations have large global sensitivity values, we enforce the differential privacy by calibrating noise based on the smooth sensitivity, rather than the global sensitivity. By doing this, we achieve the strict differential privacy guarantee with smaller magnitude noise. We illustrate our approach by using clustering coefficient, which is a popular statistics used in social network analysis. Empirical evaluations on five real social networks and various synthetic graphs generated from three random graph models show the developed divide and conquer approach outperforms the direct approach.

1 Introduction

The privacy preserving data mining community has expended great effort in developing sanitization techniques to effectively anonymize data so that the sanitized data can be published or shared with others. Researchers have proposed various privacy models such as *k*-anonymity [35], *l*-diversity [28], and *t*-closeness [27] and developed various sanitization approaches including suppression, generalization, randomization, permutation, and synthetic data generation. Refer to a recent survey book [2] for details. The aim is that an honest analyst should be able to perform a variety of ad hoc analysis and derive accurate results whereas a malicious attacker should be unable to exploit the published data to infer private information about individuals. All these sanitization approaches adopt the idea of pre-processing the raw data such that each individual's record or her sensitive attribute values

are hidden within a group of other individuals. However there is no guarantee to achieve strict privacy protection since they could not completely prevent adversaries from exploiting various auxiliary information (e.g., via background knowledge attacks [11, 29] and composition attacks [19]) to breach privacy.

Differential privacy [12, 15] is a paradigm of post-processing the output of queries such that the inclusion or exclusion of a single individual from the data set make no statistical difference to the results found. Differential privacy provides formal privacy guarantees that do not depend on an adversary's background knowledge (including access to other databases) or computational power. Differential privacy is achieved by introducing randomness into query answers. Most differential privacy research focused on theoretical studies on enforcing differential privacy in relational databases [4–6,8,13,15,16,18–20,22,24–26,36,42,43]. The applicability of enforcing differential privacy in real world applications has also been studied, e.g., the application of differential privacy to collaborative recommendation system [30], logistic regression [8], publishing contingency tables [4, 41] or data cubes [10], privacy preserving integrated queries [31], computing graph properties such as degree distributions [21] and clustering coefficient [33], and spectral analysis [40] in social network analysis.

Differential privacy is usually achieved by directly adding calibrated laplace noise on the output of the computation f. The calibrating process of this approach (denoted as *direct*) includes the calculation of the global sensitivity of the computation f that bounds the possible change in the computation output over any two neighboring databases. The added noise is generated from a Laplace distribution with the scale parameter determined by the global sensitivity of f and the user-specified privacy threshold ε. This approach works well for traditional aggregate functions (often with low sensitivity values) over tabular data.

In social network analysis, various graph features such as cluster coefficient and modularity often have a high sensitivity (proportional to the number of nodes), which is different from traditional aggregate functions (e.g., count and sum) on tabular data. Furthermore, for some computations such as spectral decomposition, we may not have explicit formula to calculate global sensitivity. A divide and conquer approach has been suggested in the literature [15]. The basic procedure of this approach (denoted as *D&C*) is to first decompose the target computation f into several less complex unit computations $f_1, \ldots, f_m$ connected by basic mathematical operations (e.g., addition, subtraction, multiplication, division), then perturb the output of each f_i with Laplace noise derived from its own sensitivity value and the distributed privacy threshold ε_i, and finally combine those perturbed f_i as the perturbed output of computation f. However, this straightforward adaptation could lead to poor performance especially when multiplication or division operations are involved. Furthermore, there is no theoretical study on calculating the unbiased estimate of f from perturbed results of f_is. In this paper, we theoretically examine how various operations affect the accuracy of complex computations. When unit computations have large global sensitivity values, we enforce the differential privacy by calibrating noise based on the smooth sensitivity [32], rather than the global sensitivity. By doing this, we achieve the strict differential privacy guarantee with smaller magnitude noise. We illustrate our approach by learning clustering coefficient (a popular graph feature used in social network analysis) from private networks. Empirical evaluations on five real social networks and various synthetic graphs generated from three random graph models show the developed divide and conquer approach outperforms the direct approach.

2 Background

We first revisit the formal definition differential privacy and the classic mechanism of enforcing differential privacy by calibrating Laplace noise based on global sensitivity in Section 2.1. We then introduce the smooth sensitivity framework [32] when global sensitivity yields unacceptable high noise levels in Section 2.2. The smooth sensitivity framework can calibrate the instance-specific noise with smaller magnitude than the worst-case noise based on the global sensitivity. In Section 2.3, we introduce complex graph models by which we generate various synthetic graphs for our empirical evaluation in Section 5.

2.1 Differential Privacy

In prior work on differential privacy, a database is treated as a collection of *rows*, with each row corresponding to the data of a different individual. Here we focus on how to compute graph features from private network topology described as its adjacency matrix. We aim to ensure that the inclusion or exclusion of a link between two individuals from the graph make no statistical difference to the results found.

Definition 1 (*Differential Privacy* [12]) *A graph analyzing algorithm* Ψ *that takes as input a graph G, and outputs* $\Psi(G)$*, preserves* (ε, δ)*-differential edge privacy if for all closed subsets S of the output space, and all pairs of neighboring graphs G and G′ from* $\Gamma(G)$*,*

$$Pr[\,\Psi(G) \in S] \leq e^{\varepsilon} \cdot Pr[\,\Psi(G') \in S] + \delta, \quad (1)$$

where

$$\Gamma(G) = \{G'(V, E') | \exists!(u, v) \in G but (u, v) \notin G'\}. \quad (2)$$

A differentially private algorithm provides an assurance that the probability of a particular output is almost the same whether or not any individual edge is included. The privacy parameter pair (ε, δ) controls the amount by which the distributions induced by two neighboring graphs may differ (smaller values enforce a stronger privacy guarantee).

A general method for computing an approximation to any function *f* while preserving ε-differential privacy is given in [15]. The mechanism for achieving differential privacy computes the sum of the true answer and random noise generated from a Laplace distribution. The magnitude of the noise distribution is determined by the sensitivity of the computation and the privacy parameter specified by the data owner. The sensitivity of a computation bounds the possible change in the computation output over any two neighboring graphs (differing at most one link).

Definition 2 (*Global Sensitivity* [15]) *The global sensitivity of a function* $f : D \rightarrow \boldsymbol{R}^d$ ($G \in D$)*,in the analysis of a graph G, is*

$$GS_f(G) := \max_{G, G' s.t. G' \in \Gamma(G)} \|f(G) - f(G')\|_1 \quad (3)$$

Theorem 1 (*The Mechanism of Adding Laplace noise* [15]) *An algorithm A takes as input a graph G, and some* $\varepsilon > 0$*, a query Q with computing function* $f : D^n \rightarrow \boldsymbol{R}^d$*, and outputs*

$$\boldsymbol{A}(G) = f(G) + (Y_1, \ldots, Y_d) \quad (4)$$

where the Y_i are drawn i.i.d from Lap($GS_f(G)/\varepsilon$). The Algorithm satisfies (ε, 0)*-differential privacy.*

Differential privacy maintains composability, i.e., differential privacy guarantees can be provided even when multiple differentially-private releases are available to an adversary.

Theorem 2 (*Composition Theorem* [14]) *If we have n numbers of* (ε, δ)*-differentially private mechanisms M_1, …, M_n, computed using graph G, then any composition of these mechanisms that yields a new mechanism M is* ($n\varepsilon$, $n\delta$)*-differentially private.*

Differential privacy can extend to group privacy as well: changing a group of *k* edges in the data set induces a change of at most a multiplicative $e^{k\varepsilon}$ in the corresponding output distribution. In this paper, we focus on the edge privacy. We can extend the algorithm to achieve the node privacy by using the above composition theorem [14].

2.2 Smooth Sensitivity

It may be hard to derive the global sensitivity of a complex function or global sensitivity yields unacceptable high noise levels. Nissim et al. [32] introduces a framework that calibrates the instance-specific noise with smaller magnitude than the worst-case noise based on the global sensitivity.

Definition 3 (*Local Sensitivity* [15, 32]) *The local sensitivity of a function $f : D \to R^d$, ($G \in D$) is*

$$LS_f(G) := \max_{G' s.t. G' \in \Gamma(G)} \|f(G) - f(G')\|_1. \quad (5)$$

Under the definition of *local sensitivity*, we only consider the set of G' for a given and predetermined *G*, such that the inclusion or exclusion of a single link between individuals cannot change the output distribution appreciably. We would emphasize that the release $f(G)$ with noise proportional to $LS_f(G)$ cannot achieve rigorous differential privacy as the noise magnitude might reveal information about the database. Refer to Example 1 in [32] for an illustrative example. To satisfy the strict differential privacy, Nissim et al. [32] proposes the β-smooth sensitivity and shows that adding noise proportional to a smooth upper bound on the local sensitivity yields a private output perturbation mechanism.

Definition 4 (*Smooth Sensitivity* [32]) *For $\beta > 0$, the β-smooth sensitivity of $f : D \to \boldsymbol{R}^d$ ($G \in D$), in the analysis of a given graph G, is*

$$S^*_{f,\beta}(G) = \max_{G' \in D} \left(LS_f(G') \cdot e^{-\beta d(G,G')} \right) \quad (6)$$

where $d(G,G')$ is the distance between graphs G and G'.

Nissim et al. [32] introduces how to compute smooth sensitivity based on the local sensitivity at distance *s* (measuring how much the sensitivity can change when up to *s* entries of *G* are modified).

Definition 5 (*Computing Smooth Sensitivity*)*The sensitivity of f at distance s is*

$$LS^{(s)}_f(G) = \max_{G' \in D: d(G,G') \le s} LS_f(G') \quad (7)$$

The β*-smooth sensitivity can be expressed in terms of* $LS_f^{(s)}(G)$ *as*

$$S^*_{f,\beta}(G)=\max_{s=0,1,\cdots,n} e^{-s\beta}\left(\max_{G':d(G,G')=s} LS_f(G')\right)=\max_{s=0,1,\cdots,n} e^{-s\beta}LS_f^{(s)}(G) \quad (8)$$

Theorem 3 shows the mechanism of calibrating noise to the smooth bound to achieve (ε, δ)-differential privacy.

Theorem 3 (*Mechanism to Add Noise Based on Smooth Sensitivity* [32]) *For a function f : D* $\rightarrow \boldsymbol{R}^d$(*G* ∈ *D*), *the following mechanism achieves* (ε, δ)-*differential privacy* (ε > 0, δ ∈ (0, 1)):

$$\boldsymbol{A}(G)=f(G)+\frac{S^*_{f,\beta}(G)}{\alpha}\cdot(Z_1,\cdots,Z_d) \quad (9)$$

where α = ε/2, $\beta=\frac{\varepsilon}{4(d+ln(2/\delta))}$, *and* Z_i (*i* = 1, …, *d*) *is drawn i.i.d from Lap*(0, 1).

Specifically when d=1, β *can be reduced as* $\beta=\frac{\varepsilon}{2ln(2/\delta)}$.

2.3 Graph Generation Models

Several network models have been proposed for studying the topological properties of real networks. Among them, the Erdös and Rényi random graphs, the Watts and Strogatz small world model, and the Barabási-Albert scale-free networks have been widely used [9]. In our work, we use the above three models with various parameters to generate synthetic graphs for empirical evaluation.

The Erdös and Rényi Random Graph [17] is the most basic model of complex networks which defines a graph with *n* vertices and a probability *p* of connecting each pair of vertices. In this model, the average degree of each node is *p*(*n* − 1) and the degree distribution is a Poisson distribution. The global cluster coefficient of the graph equals *p*. We refer to this model as the ER model in our paper.

The Small-World Model of Watts and Strogatz [38] is the most popular model of random networks with the small-world property, i.e., most vertices can be reached from others through a small number of edges. The Watts and Strogatz model can also generate graphs with the presence of a large number of triangles. In contrast, ER networks have the small world property but a small average clustering coefficient. We can construct a small-world network by staring with a regular lattice of *n* nodes in which each node is connected to *k* nearest neighbors in each direction. Next each edge is randomly rewired with probability *p*. When *p* near zero, the generated graph tends to have a high number of triangles but large distances; when *p* gets close to 1, the generated graph becomes a random graph with short distances but few triangles. The degree distribution for small-world networks is similar to that of random networks, with average degree 2*k*. The cluster coefficient of the graph is correlated to $\frac{3(k-1)}{2(2k-1)}(1-p)^3$ [9]. Scale-free small world networks have received much attention recently. For example, the authors in [7] presented an algorithm to generate graphs with small world properties by replacing each node of a random graph with cliques of different sizes. The authors in [34] developed a model for generating social networks having community structures with small-world and scale-free properties. In [44], small world

properties were examined in a snapshot of the facebook social network. We refer to the Small-World Model of Watts and Strogatz as the WS model in our paper.

The Scale-free Networks of Barabási and Albert [3] was proposed after the WS model in order to capture the characteristics of some networks whose degree distributions follow a power law. In scale-free networks, some vertices are highly connected while others have few connections. The Barabási and Albert model generates a graph by starting with a set of m_0 nodes, afterwards, at each step of the construction the network grows with the addition of new nodes. For each new node, m_1 new edges are added between the new node and some previous nodes. The nodes which receive the new edges are picked following a linear preferential attachment rule so that the most connected nodes have greater probability to receive new nodes as neighbors. The degree distribution of graphs generated by this model is $P(d) \sim d^{-3}$. The average degree is $2m_1$. The cluster coefficient of the graph is correlated to $n^{-0.75}$. We refer to this model as the BA model in our paper.

3 A Divide and Conquer Algorithm

Our divide and conquer approach is to express a function f in terms of unit computations f_1,. …, f_m such that f can be calculated from results of f_i (i = 1, …,m) via basic mathematical operations ⊙. In this paper, we limit ⊙ as linear combination, multiplication, and division. In our future work, we will extend our study to other mathematical operations such as root square and logarithm which are often used in data mining algorithms. For each unit computation f_i, we introduce noise in order to maintain its differential privacy requirement (ε_i, δ_i). Specifically, we can run the randomization mechanism with noise distribution on each f_i to achieve (ε_i, δ_i)-differential privacy. Using Theorem 2, we can achieve (ε, δ)-differential privacy of f where $\varepsilon=\sum_{i=1}^{m}\varepsilon_i$ and $\delta=\sum_{i=1}^{m}\delta_i$.

Algorithm 1 illustrates our divide and conquer approach. In Line 2, the total privacy budget (ε, δ) is distributed among unit computations such that each f_i has a privacy threshold (ε_i, δ_i). It is a challenging problem to determine the optimal distribution of privacy budget such that the combined output $\tilde{f}$ achieves the optimal approximation of f. In our paper, we simply distribute privacy budget equally among all unit computations, i.e., $\varepsilon_i=\frac{1}{m}\varepsilon$ and $\delta_i=\frac{1}{m}\delta$. In our evaluation, we show that the accuracy of the output $\tilde{f}$ varies significantly when we have different privacy budget distributions among f_i. In Lines 3–12, we enforce (ε_i, δ_i)-differential privacy on each f_i. For f_i that has small global sensitivity $GS_{f_i}(G)$, we apply Theorem 1 directly (Line 10). For f_i that may still have large global sensitivity or may not have an explicit formula for deriving global sensitivity, we first calculate its local sensitivity at distance s (Line 5), derive the smooth sensitivity parameter (β, α) based on the (ε_i, δ_i) (Line 6), compute its β-smooth sensitivity (Line 7), and finally enforce (ε_i, δ_i)-differential privacy on f_i by following Theorem 3 to calibrate noise based on the derived smooth sensitivity (Line 8). In Line 13, we output (ε, δ)-differential private $\tilde{f}$ by integrating $\tilde{f}_i$ (i = 1, …, m).

Algorithm 1

Differentially private graph statistics learning: *D&C* Approach

Require: Graph G, a target graph statistic function f, privacy parameters (ε, δ)

Ensure: $\tilde{f}(G)$ satisfies (ε, δ)-differential privacy

1: *Decompose* f into unit computations f_1, …, f_m connected by basic mathematical operations ⊙

2: *Distribute* (ε_i, δ_i) for each f_i such that $\varepsilon = \sum\varepsilon_i$ and $\delta = \sum\delta_i$

3: **for** $i = 1 \rightarrow m$ **do**

4: **if** $GS_{f_i}(G)$ is uncomputable or too large for perturbation **then**

5: Derive the formula of $LS_{f_i}^{(s)}(G)$

6: Using the $(\varepsilon_i, \delta_i)$ to compute (β, α) //Theorem 3

7: Compute the β-smooth sensitivity $S_{f_i,\beta}^{*}(G)$ using β, $LS_{f_i}^{(s)}(G)$ //Equation 8

8: Compute $\tilde{f_i}(G)$ using α, $S_{f_i,\beta}^{*}(G)$ //Equation 9

9: **else**

10: Compute $\tilde{f_i}(G)$ using $GS_{f_i}(G)$ // Theorem 1

11: **end if**

12: **end for**

13: *Integrate* $\tilde{f_1}(G), \ldots, \tilde{f_m}(G)$ into the (ε, δ)-differential private estimator of f: $\tilde{f}(G)$.

Next we show that our divide and conquer approach achieves unbiased estimate of f when operations $\odot$ contain linear combination, multiplication, and division. For simplicity, we choose one pair of functions, f_i and f_j. We assume the true value of f_i (f_j) on a given data set is a (b) and f_i (f_j) is perturbed by a Laplace noise $Lap(0, a')$ ($Lap(0, b')$). In other words, $\tilde{f_i} = a + Lap(0, a')$ and $\tilde{f_j} = b + Lap(0, b')$. Lemma 1 shows the linear combination of Laplace noise perturbed results ($\tilde{f_i}$ and $\tilde{f_j}$) is an unbiased estimate for the linear combination of the original variables (f_i and f_j). This lemma covers the mathematical operations of addition and subtraction. Similarly, Lemma 2 and Lemma 3 shows the result for multiplication and division, respectively. We leave all proof details in Appendix.

Lemma 1 *The linear combination of two perturbed values with Laplace noise is an unbiased estimate for the linear combination of the two original values without the perturbations.*

$$E(u \cdot (a+Lap(0, a')) + v \cdot (b+Lap(0, b'))) = E(u \cdot a + v \cdot b) \quad (10)$$

Assuming that $a, b, a', b' \in \mathbf{R}$; and $u, v \in \mathbf{R}$ *are parameters of the linear combination.*

Lemma 2 *The product of two perturbed values with independent Laplace noise is an unbiased estimate for the product of the two original values without the perturbations.*

$$E((a+Lap(0, a')) \cdot (b+Lap(0, b'))) = E(a \cdot b) \quad (11)$$

Assuming that $a, b, a', b' \in \mathbf{R}$ *and* a, b *are independently perturbed.*

Lemma 3 *The quotient of two perturbed results with Laplace noise is an unbiased estimate for the quotient of the two original values without the perturbation.*

$$E\left(\frac{a+Lap(0, a')}{b+Lap(0, b')}\right) = E\left(\frac{a}{b}\right) \quad (12)$$

Assuming that $a, b, a', b' \in \mathbf{R}$ *and* $b \neq 0$.

4 Learning Vertex Clustering Coefficient

In this section, we illustrate our D&C-based differential privacy preserving approach by learning vertex clustering coefficient, a widely used graph metric in social network analysis. Specifically, we show how to derive the local sensitivity at distance *s* (a key step in Algorithm 1 Line 5) for vertex clustering coefficient. We would emphasize here that our approach works naturally with other graph metrics such as graph modularity and data mining tasks (where functions can be decomposed by unit computations connected by basic mathematical operations).

The vertex clustering coefficient of vertex *i* in a graph quantifies how close *i*'s neighbors are to being a clique (complete graph). This measure was first introduced by Watts and Strogatz in 1998 [39] to determine whether a graph is a small-world graph.

$$C_i=\frac{N_\Delta(i)}{N_3(i)}=\frac{N_\Delta(i)}{d_i(d_i-1)/2} \quad (13)$$

where $N_\Delta(i)$ is the number of triangles involving vertex *i*, $N_3(i)$ is the number of connected triples having *i* as the central vertex, and d_i is the degree of vertex *i*.

We can see that C_i can be naturally expressed as a quotient of two unit computations $N_\Delta(i)$ and $N_3(i)$ or a quotient of $N_\Delta(i)$ and $d_i(d_i-1)/2$. In social network analysis, data miners often query for the vector $C=(C_1, \ldots, C_n)'$, which contains the clustering coefficients of all the vertices. For example, the average vertex clustering coefficient among all the vertices, which is defined as $\tilde{C}=\frac{1}{n}\sum_i C_i$, is a widely used metric for graph analysis. We can see that *C* can also be expressed by two vectors, $N_\Delta=(N_\Delta(1), \ldots, N_\Delta(n))'$ and $N_3=(N_3(1), \ldots, N_3(n))'$. Similarly, N_3 could be further decomposed to $D=(d_1, \ldots, d_n)'$. Table 1 shows the notations used in our paper.

Table 2 shows the global sensitivity and local sensitivity for the vertex clustering coefficient C_i (as well as its decomposed unit computations $N_\Delta(i)$, $N_3(i)$, d_i) and all vertices's clustering coefficients *C* (as well as its decomposed unit computations N_Δ, N_3, *D*). We skip the proof details in this paper since most of them are either well known or can be easily derived. We would point out that degree sequence *D* has a low global sensitivity while other functions such as N_Δ have very high global sensitivity value.

To apply our *D&C* algorithm, we need to derive the formulas of the local sensitivity at distance *s* for all above computations. We show our derived results in the remainder of this section and leave all proof details in Appendix. Result 1 shows the formula of the local sensitivity at distance s for the vertex clustering coefficient C_i and Result 2 shows the formulas of the local sensitivity at distance *s* for $N_\Delta(i)$ and $N_3(i)$.

Result 1 *The local sensitivity at distance s for the vertex clustering coefficient C_i is*

$$LS^{(s)}_{C_i}(G)=\begin{cases}\frac{2}{d_i-s} & for\, d_i-s>2\\ 1 & otherwise\end{cases} \quad (14)$$

Result 2 ([32]) *The local sensitivity at distance s for $N_\Delta(i)$ is*

$$LS^{(s)}_{N_\Delta(i)}(G)=\max_{i\neq j; j\in[n]} c_{ij}(s) \quad (15)$$

where

$$c_{ij}(s)=\min(|Ngb(i)\cap Ngb(j)|+\left\lfloor\frac{s+min(s,b_{ij})}{2}\right\rfloor,n-2)$$

and $b_{ij}=\sum_{k\in[n]}a_{ik}\oplus a_{kj}$ *is the number of half-built triangles involving edge(i,j).*

The local sensitivity at distance s for $N_3(i)$ *is*

$$LS^{(s)}_{N_3(i)}(G)=\min(d_i+s-1,n-2) \quad (16)$$

Result 3 shows the formula of the local sensitivity at distance *s* for the clustering coefficient vector *C* and Result 4 shows the formulas of the local sensitivity at distance *s* for vector N_Δ and N_3.

Result 3 *The local sensitivity at distance s for* $C=(C_1,\ldots,C_n)'$ *is*

$$LS^{(s)}_{C}(G)=\max_{i\neq j;i,j\in[n]}\{\min(d_{max}+\left\lfloor\frac{s+min(s,b_{ij})}{2}\right\rfloor,n-1)\} \quad (17)$$

where $b_{ij}=\sum_{k\in[n]}a_{ik}\oplus a_{kj}$ *is the number of half-built triangles involving edge(i,j)*

Result 4 *The local sensitivity at distance s for* N_Δ *is*

$$LS^{(s)}_{N_\Delta}(G)=3\cdot\max_{i\neq j;j\in[n]}c_{ij}(s)$$

where

$$c_{ij}(s)=\min(|Ngb(i)\cap Ngb(j)|+\left\lfloor\frac{s+min(s,b_{ij})}{2}\right\rfloor,n-2)$$

and $b_{ij}=\sum_{k\in[n]}a_{ik}\oplus a_{kj}$ *is the number of half-built triangles involving edge(i,j) (This result was appeared in* [32]).

The local sensitivity of N_3 *at distance s is*

$$LS^{(s)}_{N_3}(G)=\max_{i\neq j;i,j\in[n]}\{2n-4,\min(d_{max}+d_{sencondmax}-2+\left\lfloor\frac{s+min(s,b_{ij})}{2}\right\rfloor)$$

where $b_{ij}=\sum_{k\in[n]}a_{ik}\oplus a_{kj}$ *is the number of half-built triangles involving edge(i,j).*

For vertex clustering coefficient C_i, we have two decomposition strategies: ($N_\Delta(i)$, $N_3(i)$) or ($N_\Delta(i)$, d_i). Similarly for clustering coefficient vector *C*, we can also have two decomposition strategies: (N_Δ, N_3) or (N_Δ, *D*). When we apply the second decomposition strategy, we use the global sensitivity of d_i or *D* because they are very small. However, we should adjust our estimate of $d_i(d_i-1)/2$, as shown in Lemma 4, if we use the same $\tilde{d}_i$ twice in the calculation. Of course, we can query twice to get two perturbed values of d_i and calculate the unbiased estimate of $d_i(d_i-1)/2$ based on Lemma 2. In this case, two queries of d_i should split the

privacy budget assigned to d_i. This example illustrates the importance of deriving unbiased estimate of f from its perturbed values of unit computations.

Lemma 4 *The unbiased estimate for the product of the linear combinations of the same perturbed value with Laplace noise is*

$$E((u_1 \cdot a+v_1)(u_2 \cdot a+v_2))=E((u_1 \cdot \tilde{a}+v_1)(u_2 \cdot \tilde{a}+v_2)) - u_1 \cdot u_2 \cdot a'^2$$

Assuming that $a \in \mathbf{R}$ *and* $\tilde{a} = a + Lap(0, a')$.

The time complexity for computing the cluster coefficient C_i for node i is $O(d_i^2)$ where d_i denotes the degree of node i. In the worst case where the node degree may be as large as $n - 1$, the time complexity is $O(n^2)$ for computing the cluster coefficient of one node and $O(n^3)$ for computing the cluster coefficients of all vertices. Since the time complexity of perturbing one output with the Laplacian noise is $O(1)$, enforcing differential privacy by the direct approach has no influence on the time complexity of the original algorithm of computing cluster coefficient. Furthermore, our divide and conquer approach, which decomposes the computation f into m less complex unit computations $f_1, \ldots, f_m$ connected by basic mathematical operations, has the same time complexity as the direct approach because m is often a small constant. Note that it takes constant time for the algorithm to compute β-smooth sensitivity $S^*_{f_i,\beta}(G)$ (Line 7 in Algorithm 1). This is because we can compute the maximum value by calculating the derivative of the right side of Equation 8 rather than iteratively calculate the value for each distance s. In the characteristics of the function $e^{-s\beta}LS^s_f(G)$, the $e^{-s\beta}$ part decreases exponentially as s increases while the $LS^s_f(G)$ part increases polynomially, which ensures the strategy of computing the derivative feasible.

5 Empirical Evaluation

In this section, we conduct evaluations to compare the utility of the *direct* approach and the $D\&C_D$ approach on five real graphs and several synthetic graphs generated with three graph models, ER model, WS model and BA model (refer to section 2.3).

The five real graphs are denoted respectively as GrQc, Enron, Polbooks, Polblogs, YesIWell. GrQc is the General Relativity and Quantum Cosmology collaboration network from the SNAP Stanford [23]. Table 3 gives some published statistics of the GrQc dataset. We mainly use the GrQc graph data when comparing the utility preservations of different approaches in Section 5.1 and exploring the privacy budget distribution in Section 5.2. Enron[1] is an email network collected and prepared by the CALO Project and it has 148 nodes and 869 edges; Polbooks[2] is a network of books about US politics published around the time of the 2004 presidential election and sold by Amazon.com and it has 105 nodes and 441 edges; Polblog [1] is a network of hyperlinks between weblogs on US politics; YesIWell is a human physical activities related social network dataset with 185 nodes and 684 edges, which is part of the data gained from the YesIWell study[3] conducted in 2010–2011 as collaboration among several health laboratories and universities to help people maintain active lifestyles and lose weight.

[1] http://www.cs.cmu.edu/enron/
[2] http://www–personal.umich.edu/mejn/netdata/
[3] http://aimlab.cs.uoregon.edu/smash/

Synthetic graphs are generated using the software Gephi[4] with the Complex Generators plugin. For each graph model (ER, WS, and BA), we generate several graphs by varying graph generation parameters. Table 4 shows some basic characteristics of the generated graphs, where n denotes the number of nodes, m denotes the number of edges, $\bar{c}$ is the average cluster coefficient, n_Δ is the total number of triangles in the graph, fr_Δ is the fraction of the closed triangles, which is defined as the total number of triangles divided by the total number of length two paths. The generation parameters for the ER model are listed as (n, p), those for the WS model as (n, k, p) and those for the BA model as (n, m_0, m_1) (refer to Section 2.3). For all synthetic graphs, we fix the number of nodes as 1000.

5.1 Utility

We compare our divide and conquer approach with the *direct* approach that directly adds calibrated laplace noise on the output of the computation of f. For vertex cluster coefficient C_i, to examine how different decomposition strategies affect the accuracy of the final output $\tilde{C}(i)$, we include evaluation results on two decomposition strategies: $(N_\Delta(i), N_3(i))$ and $(N_\Delta(i), d_i)$.

Table 5 shows comparisons of these three methods, denoted as *direct*, $D\&C_{N_3(i)}$, and $D\&C_{di}$ respectively. In our experiments, we fix $\delta = 0.01$ and vary $\varepsilon \in \{0.01, 0.1, 1, 10\}$. We choose the node with the largest degree ($d_i = 81$) for the vertex cluster coefficient. For each of three methods with every privacy setting, we repeat the randomization process for 3,000 times. We report the the mean and standard deviation of the absolute error between $\tilde{C}_i$ and C_i in Table 5.

Similarly for clustering coefficient vector C, we use $D\&C_{N_3}$ and $D\&C_D$ to denote divide and conquer approaches based on two decomposition strategies. We set ε' here with the magnitude of $n \cdot \varepsilon$, where $\varepsilon \in \{0.01, 0.1, 1, 10\}$. As a result, each entry of the vector achieves the same (ε, δ)-differential privacy as the previous experiment. Table 6 shows our comparisons.

Note that in our experiments, we also use the β-smooth sensitivity in the *direct* approach. This is because the utility is significantly lost if we use the global sensitivity. For example, if we use the the global sensitivity for C_i when $\varepsilon = 1$, the error is 0.3558 ± 0.1915, which is significantly larger than 0.0338 ± 0.0332 (shown in Table 5) of the *direct* approach using the smooth sensitivity.

We have following observations from our evaluation results. First, in general, the D&C approach achieves equivalent utility as, if not better than, that of the *direct* approach. This result indicates that we can still enforce differential privacy by decomposing a complex function into unit computations even though the complex function may have a large global sensitivity or may not have an explicit formula of its global sensitivity. Second, for the $D\&C$ approach, querying for the degree sequence D instead of the N_3 vector will probably lead to better utility. This is because the degree sequence has a low global sensitivity. Third, under the same privacy threshold, it is much better to query for the vector of all the clustering coefficients at once rather than to query for the vertex clustering coefficient one by one.

5.2 Distribute Privacy Budget

Note that in our previous experiments, we adopted a simple strategy, i.e., distributing privacy budget equally among unit computations. One conjecture is that the $D\&C$ approach would achieve much better utility preservation if we have a better strategy of distributing

[4] https://gephi.org/

privacy budget. For example, in our $D\&C_D$ method that obtains the clustering coefficient vector C by querying for the vectors N_Δ and D, the sensitivity magnitude of the vector N_Δ is much larger than that of the vector D. Hence we expect to achieve better utility if we distribute more privacy budget to N_Δ than to D. On the other hand, one characteristic of the division is that the denominator is more sensitive than the numerator, having more influence on the quotient result. As a result, the denominator vector D may need more privacy budget under certain conditions.

Figure 1 shows how preservation of utility (in terms of approximation error shown as X-axis) varies when we change the ratio of the privacy budget on N_Δ (numerator) and the privacy budget on D (shown as Y-axis) when we apply our $D\&C_D$ method with the total budget $\varepsilon = 0.1, 1.0, 10$. The red lines correspond to the *direct* method and the blue curves correspond to the $D\&C_D$ method. The points of those blue curves that are under the red line show the privacy budget distributions with which the $D\&C_D$ method outperforms the *direct* method. Our evaluation results (shown as the third column in Table 6) correspond to the first point (with ratio=1) in each figure. In our future work, we will study the use of Newton iterative method to find out the optimal ratio so that we can achieve optimal utility preservation in our divide and conquer approach.

5.3 Evaluation on other real graphs and synthetic graphs

In this section, we conduct evaluations to compare the utility of the *direct* approach and the $D\&C_D$ approach on the other four real graphs and various synthetic graphs generated by the three graph models (ER, WS, and BA). In all our experiments, we fix $\delta = 0.01$ and set ε' here with the magnitude of $n \cdot \varepsilon$, where $\varepsilon \in \{0.01, 0.1, 1, 10\}$. As a result, each entry of the vector achieves the same (ε, δ)-differential privacy as the previous experiment. For the $D\&C_D$ approach, we simply distribute privacy budget equally, i.e., setting the distribution ratio of privacy budget as 1. As illustrated in Section 5.2, we would achieve even better utility preservation for the $D\&C$ approach when we adopt a better strategy of distributing privacy budget.

The evaluation results are shown in Table 7. Specifically, we have the following observations.

- For the ER model with parameters n and p, ER1, ER2, and ER3 are generated with the same number of nodes $n = 1000$ with increasing p values, 0.05, 0.1, 0.5. The number of edges (m), the average cluster coefficient ($\bar{c}$), and the fraction of triangle (fr_Δ) of these three graphs increase with the increasing of p. For the *direct* approach, we can see that the entrywise absolute error increases with p. On the contrary, the entrywise absolute error decreases for the $D\&C_D$ approach. Thus we can conclude that the $D\&C_D$ approach tends to achieve much better utility preservation than the *direct* approach for ER random graphs. The utility of $D\&C_D$ approach increases as p increases.
- For the WS model with parameters n, k and p, from Section 2.3, we know that the graph tends to have more triangles and larger distance when p is near 0 and the graph is more random with less triangles and shorter distance when p approaches 1. WS1, WS2, and WS3 are generated with the same parameters n and k but decreasing p as 0.7, 0.5 and 0.2 respectively. These three graphs have the same density but increasing $\bar{c}$ and fr_Δ. We see no clear trend of the absolute entrywise error change for both approaches. WS3, WS4, and WS5 are generated with the same parameters n and p (0.2) but increasing k as 50, 100, and 500 respectively. As a result, these three graphs have increasing m, $\bar{c}$, and fr_Δ. For the *direct* approach, we can see that the entrywise absolute error increases with k

whereas the entrywise absolute error decreases for the $D\&C_D$ approach. Thus we can conclude that the $D\&C_D$ approach tends to achieve much better utility preservation than the *direct* approach among WS small world graphs and the utility of the $D\&_{C_D}$ approach increases as k increases.

- For the BA model with parameters n, m_0, and m_1, BA1, BA2, and BA3 are generated with the same parameter n and increasing m_1 as 25, 50, and 250 respectively. Recall that in the BA model m_1 denotes the number of newly added edges between each new node and existing nodes. Note that when the ratio between m_0 and m_1 is fixed, the increase of m_1 indicates the increases of m, $\bar{c}$, and fr_Δ. For the *direct* approach, we can see that the entrywise absolute error increases with the increase of m_1 whereas the entrywise absolute error decreases for the $D\&C_D$ approach. Thus we can conclude that the $D\&C_D$ approach tends to achieve much better utility preservation than the *direct* approach among BA small world graphs with large density.
- For the BA model with parameters n, m_0 and m_1, BA4, BA5, BA6, and BA7 are generated with the same parameters n and $m_1 = 5$ but increasing m_0 as 10, 50, 100, and 500 respectively. Recall that in the BA model m_0 denotes the number of nodes in the starting set. When m_1 is fixed, the increase of m_0 indicates the increases of $\bar{c}$ and fr_Δ. We can see that the utility of the $D\&C_D$ approach is still better than that of the *direct* approach but not as significant as the graphs BA1, BA2, and BA3. The absolute entrywise error for the $D\&C_D$ approach is increasing as parameter m_0 increases, which shows the same trend as that of the *direct* approach. Thus we can conclude that the $D\&C_D$ approach still tends to achieve better utility preservation than the *direct* approach among BA small world graphs with low density but its advantage decreases when m_0 is larger.
- For all four real networks (Enron, Polbook, Polblog and YesIWell), we can see that the $D\&C_D$ approach outperforms the *direct* approach. The extent of the advantage of the $D\&C_D$ approach is similar as that observed in BA graphs with low density (BA4, BA5, BA6, and BA7).

In summary, we draw the following conclusions. First, the $D\&C_D$ approach outperforms the *direct* approach in all graphs (15 synthetic graphs and four real graphs). Second, the $D\&C_D$ approach shows overwhelming superiority in all three graphs generated by the ER model, all five graphs generated the WS model, and the first three graphs (BA1, BA2, BA3) from the BA model. In the above eleven graphs, the entrywise error of the output of $D\&C_D$ approach tends to much smaller (by several orders of magnitude) than that of the *direct* approach. Third, for real graphs (Enron, Polbook, Polblog, and YesIWell), and the last four graphs (BA4, BA5, BA6 and BA7) generated with the BA model, the advantage of the $D\&C_D$ approach is still obvious and the entrywise error of the output of $D\&C_D$ approach is still smaller (by 10% with small ε or by one order of magnitude with large ε than that of the *direct* approach. We observe that in the four real networks and the last four BA graphs are relatively sparse and most nodes tend to have a very small magnitude of degree (less than 10) and hence a small number of triangles. When calibrating the noises to N_Δ and D in our $D\&C_D$ approach, the influence due to the distortion is large. As a result, the advantage of our $D\&C_D$ approach decreases when the graphs have low density.

6 Conclusion and Future Work

Enabling accurate analysis of graph data while preserving differential privacy is of great importance and poses great challenge due to potential high global sensitivity. In this paper, we have presented a divide and conquer approach that can be used to enforce differential privacy for complex functions. We have conducted theoretical analysis and extensive

empirical evaluations to show that the developed divide and conquer approach generally outperforms the approach of directly enforcing differential privacy in terms of utility preservation. This result is especially promising for data mining or exploration tasks with interactive processes, in which a user can adaptively query the system about the data. The user now has options of reusing previous intermediate query results rather than submitting to the system "new" queries that can be expressed by previous ones.

There are some other aspects of this work that merit further research. Among them, we will continue the line of this research by investigating how to enforce differential privacy for other complex functions or social network analysis tasks. For functions that we cannot compute the smooth sensitivity efficiently or explicitly, Nissim et al. proposed an approximation method that computes the β-smooth upper bound on the local sensitivity of these functions and developed a sample-aggregation framework for a large class of functions [32]. We will evaluate those functions based on the sample-aggregation framework. We will exploit the use of correlations among unit computations to further reduce noise and enhance accuracies of computation outputs. Our goal is to identify (optimal) decomposition strategies and (optimal) budget privacy distribution. Finally, we will study non-interactive graph data release mechanisms, i.e., we use the derived differentially private graph statistics to generate synthetic graphs for release.

Acknowledgments

The conference version of this work was published in [37]. This journal version contains significant extensions including detailed theoretical proofs and extensive empirical evaluations. We would like to thank anonymous reviewers for their invaluable comments. This work was supported in part by U.S. National Science Foundation (0546027, 0831204, 0915059, 1047621), U.S. National Institutes of Health (1R01GM103309-01A1), and the Shanghai Magnolia Science & Technology Talent Fund (11BA1412600).

References

1. Adamic LA, Glance N. The political blogosphere and the 2004 us election. WWW-2005 Workshop on the Weblogging Ecosystem. 2005
2. Aggarwal, CC.; Yu, PS. Privacy-preserving data mining: models and algorithms. New York Inc: Springer-Verlag; 2008.
3. Barabási AL, Albert R. Emergence of scaling in random networks. science. 1999; 286(5439):509–512. [PubMed: 10521342]
4. Barak, B.; Chaudhuri, K.; Dwork, C.; Kale, S.; McSherry, F.; Talwar, K. Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. ACM; 2007. Privacy, accuracy, and consistency too: a holistic solution to contingency table release; p. 273-282.
5. Blum, A.; Dwork, C.; McSherry, F.; Nissim, K. Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. ACM; 2005. Practical privacy: the SuLQ framework; p. 128-138.
6. Blum, A.; Ligett, K.; Roth, A. Proceedings of the 40th annual ACM symposium on Theory of computing. ACM; 2008. A learning theory approach to non-interactive database privacy; p. 609-618.
7. Caci, Barbara; Cardaci, Maurizio; Tabacchi, Marco Elio. Facebook as a Small World: a topological hypothesis. Socail Network Analysis and Mining. 2012; 2(2):163–167.
8. Chaudhuri, K.; Monteleoni, C. Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS). Citeseer; 2008. Privacy-preserving logistic regression; p. 289-296.
9. Costa LF, Rodrigues FA, Travieso G, Boas PRV. Characterization of complex networks: A survey of measurements. Advances in Physics. 2007; 56(1):167–242.

10. Ding, Bolin; Winslett, Marianne; Han, Jiawei; Li, Zhenhui. Differentially private data cubes: optimizing noise sources and consistency; SIGMOD Conference; 2011. p. 217-228.
11. Du W, Teng Z, Zhu Z. Privacy-MaxEnt: integrating background knowledge in privacy quantification. ACM SIGMOD. 2008
12. Dwork C. A firm foundation for private data analysis. Communications of the ACM. 2011; 54(1): 86–95.
13. Dwork C, Kenthapadi K, McSherry F, Mironov I, Naor M. Our data, ourselves: Privacy via distributed noise generation. Advances in Cryptology-EUROCRYPT 2006. 2006:486–503.
14. Dwork, C.; Lei, J. Proceedings of the 41st annual ACM symposium on Theory of computing. ACM; 2009. Differential privacy and robust statistics; p. 371-380.
15. Dwork C, McSherry F, Nissim K, Smith A. Calibrating noise to sensitivity in private data analysis. Theory of Cryptography. 2006:265–284.
16. Dwork C, Smith A. Differential privacy for statistics: What we know and what we want to learn. Journal of Privacy and Confidentiality. 2010; 1(2):2.
17. P. ERDdS and A. R&WI. On random graphs i. Publ. Math. Debrecen. 1959; 6:290–297.
18. Friedman, A.; Schuster, A. Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2010. Data mining with differential privacy; p. 493-502.
19. Ganta, SR.; Kasiviswanathan, SP.; Smith, A. Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2008. Composition attacks and auxiliary information in data privacy; p. 265-273.
20. Ghosh, A.; Roughgarden, T.; Sundararajan, M. Proceedings of the 41st annual ACM symposium on Theory of computing. ACM; 2009. Universally utility-maximizing privacy mechanisms; p. 351-360.
21. Hay, M.; Li, C.; Miklau, G.; Jensen, D. ICDM. IEEE; 2009. Accurate estimation of the degree distribution of private networks; p. 169-178.
22. Hay M, Rastogi V, Miklau G, Suciu D. Boosting the Accuracy of Differentially Private Histograms Through Consistency. Proceedings of the VLDB Endowment. 2010; 3(1)
23. Leskovec J. Snap: Stanford network analysis platform.
24. Kifer, Daniel; Machanavajjhala, Ashwin. No free lunch in data privacy; SIGMOD Conference; 2011. p. 193-204.
25. Lee, Jaewoo; Clifton, Chris. Differential identificablity. KDD. 2012
26. Li, C.; Hay, M.; Rastogi, V.; Miklau, G.; McGregor, A. Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data. ACM; 2010. Optimizing linear counting queries under differential privacy; p. 123-134.
27. Li N, Li T, Venkatasubramanian S. t-closeness: Privacy beyond k-anonymity and l-diversity. ICDE. 2007
28. Machanavajjhala, A.; Gehrke, J.; Keifer, D.; Venkitasubramanian, M. l-diversity: privacy beyond k-anonymity; Proceedings of the IEEE ICDE Conference; 2006.
29. Martin DJ, Kifer D, Machanavajjhala A, Gehrke J, Halpern JY. Worst-Case Background Knowledge for Privacy-Preserving Data Publishing. ICDE. 2007
30. McSherry, F.; Mironov, I. KDD. ACM; 2009. Differentially Private Recommender Systems.
31. McSherry, FD. Proceedings of the 35th SIGMOD international conference on Management of data. ACM; 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis; p. 19-30.
32. Nissim, K.; Raskhodnikova, S.; Smith, A. Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing. ACM; 2007. Smooth sensitivity and sampling in private data analysis; p. 75-84.
33. Rastogi, V.; Hay, M.; Miklau, G.; Suciu, D. Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM; 2009. Relationship privacy: Output perturbation for queries with joins; p. 107-116.

34. Sallaberry, Arnaud; Zaidi, Faraz; Melancon, Guy. Model for generating artificial social networks having community structures with small-world and scale-free properties. Socail Network Analysis and Mining. 2013
35. Samarati, P.; Sweeney, L. Protecting privacy when disclosing information: k-anonymty and its enforcement through generalization and suppression; Proceedings of the IEEE Symposium on Research in Security and Privacy; 1998.
36. Sarathy, R.; Muralidhar, K. Joint UNECE/Eurostat work session on statistical data confidentiality. Bilbao, Spain: 2009. Differential Privacy for Numeric Data.
37. Wang, Yue; Wu, Xintao; Zhu, Jun; Xiang, Yang. On learning cluster coefficient of private networks. ASONAM. 2012
38. Watts D, Strogatz S. The small world problem. Collective Dynamics of Small-World Networks. 1998; 393:440–442.
39. Watts DJ, Strogatz SH. Collective dynamics of small-world networks. Nature. 1998; 393(6684): 440–442. [PubMed: 9623998]
40. Wang, Yue; Wu, Xintao; Wu, Leting. Differential privacy preserving spectral graph analysis. PAKDD. 2013
41. Xiao, X.; Wang, G.; Gehrke, J. Data Engineering (ICDE), 2010 IEEE 26th International Conference on. IEEE; 2010. Differential privacy via wavelet transforms; p. 225-236.
42. Xiao, Xiaokui; Bender, Gabriel; Hay, Michael; Gehrke, Johannes. ireduct: differential privacy with reduced relative errors; SIGMOD Conference; 2011. p. 229-240.
43. Ying, Xiaowei; Wu, Xintao; Wang, Yue. On linear refinement of differential privacy-preserving query answering. PAKDD. 2013
44. Zaidi, Faraz. Small world networks and clustered small world networks with random connectivity. Socail Network Analysis and Mining. 2013; 3(1):51–63.

A Proof

Proof of Lemma 1.

$$E(u\cdot(a+Lap(0,a'))+v\cdot(b+Lap(0,b')))=E(u\cdot a+v\cdot b)+u\cdot E(Lap(0,a'))+v\cdot E(Lap(0,b'))$$

Since $E(Lap(0, a')) = 0$ and $E(Lap(0, b')) = 0$, we have

$$E(u\cdot(a+Lap(0,a'))+v\cdot(b+Lap(0,b')))=E(u\cdot a+v\cdot b)$$

Proof of Lemma 2.

$$\begin{aligned}
&E((a+Lap(0,a'))\\
&\quad\cdot(b+Lap(0,b')))\\
&=E(a\\
&\cdot b+b\cdot Lap(0,a')\\
&+a\cdot Lap(0,b')\\
&+Lap(0,a')\\
&\quad\cdot Lap(0,b'))\\
&=E(a\\
&\quad\cdot b)+b\cdot E(Lap(0,a'))\\
&+a\cdot E(Lap(0,b'))\\
&+E(Lap(0,a')\cdot Lap(0,b'))
\end{aligned}$$

$E(Lap(0, a')) = E(Lap(0, b')) = 0$; besides, $E(Lap(0, a') \cdot Lap(0, b')) = E(Lap(0, a')) \cdot E(Lap(0, b'))$ since a, b are independently perturbed with Laplace noise; hence

$$E((a+Lap(0,a'))\cdot(b+Lap(0,b')))=E(a\cdot b)$$

Proof of Lemma 3.

$$E(\frac{a+Lap(0,a')}{b+Lap(0,b')})=E(\frac{a}{b})+E(\frac{a+Lap(0,a')}{b+Lap(0,b')}-\frac{a}{b})$$

Since $E(Lap(0, a')) = 0$ and $E(0, Lap(b')) = 0$, we have

$$E(\frac{a+Lap(0,a')}{b+Lap(0,b')}-\frac{a}{b})=E(\frac{(a+Lap(0,a')b-a(b+Lap(0,b')))}{b\cdot(b+Lap(0,b'))})=E(\frac{b\cdot Lap(0,a')}{b\cdot(b+Lap(0,b'))})-E(\frac{a\cdot Lap(0,b')}{b\cdot(b+Lap(0,b'))})=0$$

Proof of Result 1.

In order to derive $LS^{(s)}_{C_i}(G)$, we first consider the case for $s = 0$, i.e., $LS_{C_i}(G)$.

Let G and G' respectively denote the original graph G and its neighbor graph by deleting an edge from G. For a given node i: let $N_\Delta(i)$ and $N_3(i)$ denote the attributes of i in G; while $N'_\Delta(i)$ and $N'_3(i)$ denote the same attributes in G'. By definition, we have $0 \le N_\Delta(i) \le N_3(i) = 2/(d_i(d_i - 1))$, When deleting an edge from G, $N_\Delta(i)$ would be decreased by at most $d_i - 1$; while $N_3(i)$ would be decreased by exactly $d_i - 1$. Therefore,

$$LS_{C_i}(G)=\frac{N_\Delta(i)}{d_i(d_i-1)/2}-\frac{N'_\Delta(i)}{d_i(d_i-1)/2-(d_i-1)} \quad (18)$$

When $0 \le N_\Delta(i) \le d_i - 1$, we have

$$LS_{C_i}(G)\le\frac{N_\Delta(i)}{d_i(d_i-1)/2}\le\frac{d_i-1}{d_i(d_i-1)/2}=2/d_i$$

When $d_i - 1 \le N_\Delta(i) \le d_i(d_i - 1)/2$, we have

$$LS_{C_i}(G)\le\frac{N_\Delta(i)}{d_i(d_i-1)/2}-\frac{N_\Delta(i)-(d_i-1)}{d_i(d_i-1)/2-(d_i-1)}=\frac{2}{d_i-2}-\frac{4N_\Delta(i)}{d_i(d_i-1)(d_i-2)}\le\frac{2}{d_i-2}-\frac{4(d_i-1)}{d_i(d_i-1)(d_i-2)}=2/d_i$$

So that $LS_{C_i}(G)=\frac{2}{d_i}$.

In general case, for $s > 0$, we have (Equation 7),

$$LS^{(s)}_{C_i}(G)=\max_{G'\in D:d(G,G')\le s}LS_{C_i}(G')=\frac{2}{d_i-s}$$

for $d_i - s > 2$; and $LS_{C_i}^{(s)}(G) = GS_{C_i}(G) = 1$ otherwise.

Proof of Lemma 4.

Since $E(Lap(0, a')) = 0$ and $E(Lap^2(0, a')) = a'^2$, $\tilde{a} = a + Lap(0, a')$,therefore

$$\begin{aligned} &E((u_1 \cdot \tilde{a} + v_1)(u_2 \cdot \tilde{a} + v_2)) \\ &= E((u_1 \cdot a + v_1)(u_2 \cdot a + v_2)) \\ &+ (u_1 \cdot (u_2 \cdot a + v_2) + u_2 \cdot (u_1 \cdot a + v_1)) E(Lap(0, a')) \\ &+ u_1 \cdot u_2 \cdot E(Lap^2(0, a')) \\ &= E((u_1 \cdot a + v_1)(u_2 \cdot a + v_2)) \\ &+ u_1 \cdot u_2 \cdot a'^2 \end{aligned}$$

Proof of Result 3.

We first consider the situation of $s = 0$,

$$LS_C(G) = \max_{a_{ij}=1} \left\{ \frac{2}{d_i} + \frac{2}{d_j} + \sum_{a_{ik}a_{jk}=1} \frac{1}{d_k(d_k-1)/2} \right\} \leq \max_{a_{ij}=1} \left\{ \frac{2}{d_i} + \frac{2}{d_j} + \frac{2(d_{max}-1)}{d_k(d_k-1)} \right\} \leq 2\left(\frac{1}{2} + \frac{1}{2} + \frac{2(d_{max}-1)}{2*(2-1)} \right) = d_{max}$$

$LS_C^{(s)}(G) = LS_C(G) + s$ for $s \leq b_{ij}$ because we may add one edge to complete a half-built triangle involving edge(i, j) which makes the sensitivity increased by at most one; meanwhile, $LS_C^{(s)}(G) = LS_C(G) + \left\lfloor \frac{s + b_{ij}}{2} \right\rfloor$ for $s > b_{ij}$ because we have to add two edges to form a triangle to make the sensitivity increased by one, after completing all the b_{ij} half built triangles involving edge(i, j). Besides, $LS_C^{(s)}(G) \leq GS_C(G) = n - 1$. So we have Equ. 17.

Proof of Result 4.

In addition to the proof of Result. 2 given in [32], $LS_{N_\Delta}^{(s)}(G) = 3 \cdot \max_{i \neq j; j \in [n]} c_{ij}(s)$ because each of the two entries corresponding to vertex i and j will be decreased by at most $\max_{i \neq j; j \in [n]} c_{ij}(s)$, when edge ($i$, j) is deleted from G. Besides, there are $\max_{i \neq j; j \in [n]} c_{ij}(s)$ other entries whose value will be decreased by one, corresponding to the neighbours in common by vertex i and j.

For N_3, we first consider the situation of $s = 0$,

$$LS_{N_3}(G)=\max_{a_{ij}=1}\left\{\frac{d_i(d_i-1)}{2}-\frac{(d_i-1)(d_i-2)}{2}+\frac{d_j(d_j-1)}{2}-\frac{(d_j-1)(d_j-2)}{2}\right\}\leq \max_{a_{ij}=1}\{d_i-1+d_j-1\}\leq d_{max}+d_{secondmax}-2$$

In general case, $LS^{(s)}_{N_3}(G)=LS_{N_3}(G)+s$ for $s \leq b_{ij}$ and $LS^{(s)}_{N_3}(G)=LS_{N_3}(G)+\left\lfloor\frac{s+b_{ij}}{2}\right\rfloor$ for $s > b_{ij}$ which are similar as those of $LS^{(s)}_{C}(G)$, and $LS^{(s)}_{N_3}(G) \leq GS_{N_3}(G)=2n-4$. So we have the form for $LS^{(s)}_{N_3}(G)$.

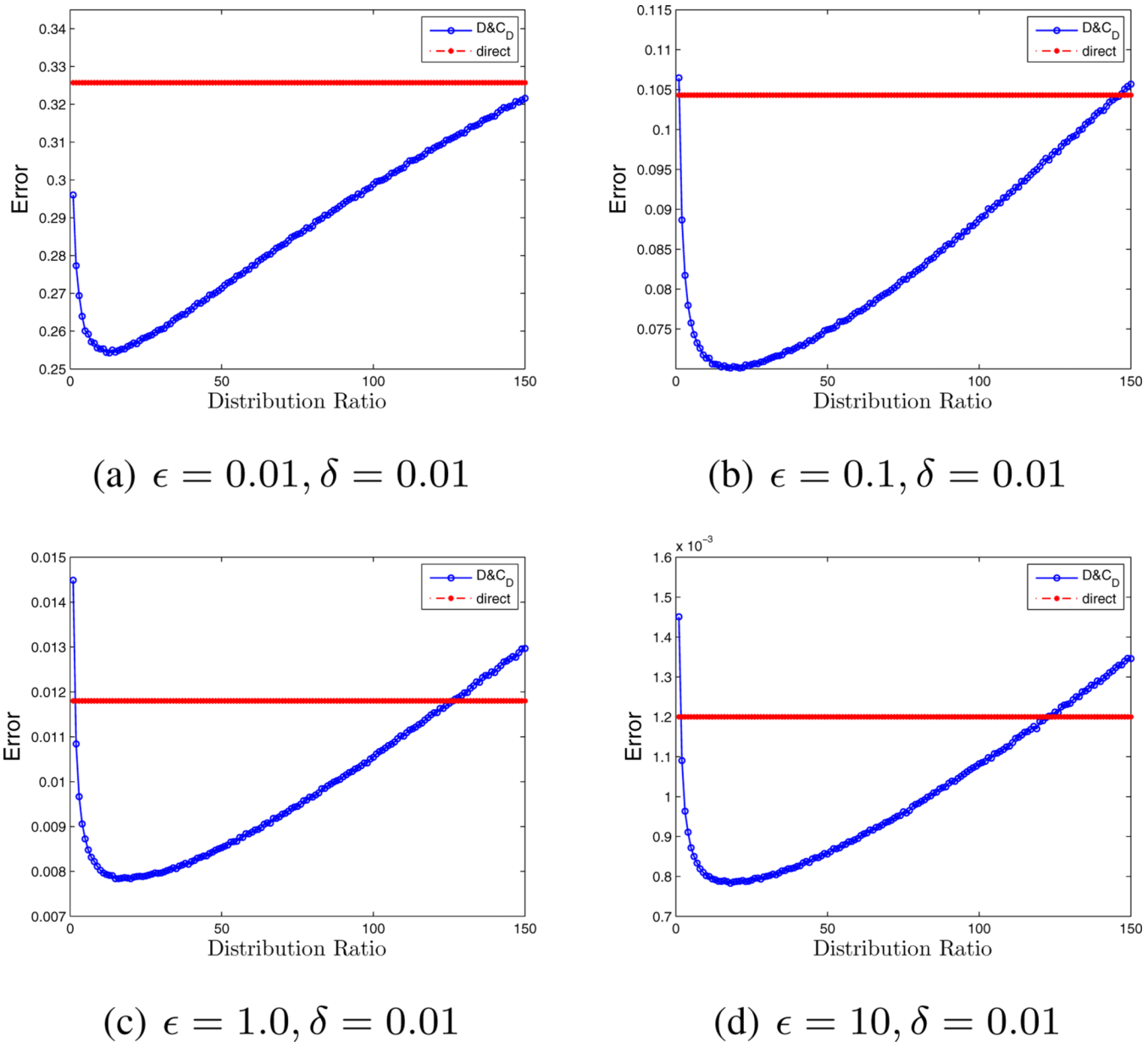

Fig. 1.
Average entrywise absolute error change with the distribution ratio of ε′, given varying, ε, δ

Table 1

Notations of graph metrics

The original graph with n nodes and m edges	$G(n,m)$
Adjacent matrix of G	A
An entry in A	a_{ij}
The degree of a node i	d_i
The maximum node degree in G	d_{max}
The clustering coefficient of a node i	C_i
The set of the neighboring nodes of node i	$N\ gb(i)$
Number of triangles involving node i	$N_\Delta\ (i)$
Number of connected triples (i as the central node)	$N_3(i)$
The vector of all C_i	C
The vector of all d_i	D
The vector of all $N_\Delta(i)$	N_Δ
The vector of all $N_3(i)$	N_3

Table 2

Global sensitivity and local sensitivity of graph metrics

Function f	GS_f	LS_f
C_i	1	$2/d_i$
$N_\Delta(i)$	$n-2$	$\max_{a_{ij}=1} \lvert Ngb(i) \cap Ngb(j) \rvert$
$N_3(i)$	$n-2$	$d_i - 1$
d_i	1	1
C	$n-1$	d_{max}
N_Δ	$3(n-2)$	$3\max_{a_{ij}=1} \lvert Ngb(i) \cap Ngb(j) \rvert$
N_3	$2n-4$	$2d_{max}-2$
D	2	2

Table 3

General Relativity and Quantum Cosmology Collaboration Network Dataset Statistics

Number of nodes	5242
Number of edges	28980
Nodes in the largest connected component	4158
Edges in the largest connected component	26850
Average clustering coefficient	0.5296
Number of triangles	48260
Fraction of closed triangles	0.6298
Diameter	17

Table 4

Dataset Statistics

Graphs	n	m	$\bar{c}$	n_Δ	fr_Δ	Generation parameters
ER1	1000	25094	0.0496	20787	0.0496	(1000, 0.05)
ER2	1000	50129	0.1002	167671	0.1002	(1000, 0.1)
ER3	1000	249755	0.5000	20768657	0.4999	(1000, 0.5)
WS1	1000	25000	0.0661	27153	0.0659	(1000, 50, 0.7)
WS2	1000	25000	0.1280	52354	0.1272	(1000, 50, 0.5)
WS3	1000	25000	0.3886	158301	0.3863	(1000, 50, 0.2)
WS4	1000	50000	0.4080	672623	0.4069	(1000, 100, 0.2)
WS5	1000	250000	0.5679	23694389	0.5696	(1000, 500, 0.2)
BA1	1000	24975	0.0759	64660	0.1274	(1000, 50, 25)
BA2	1000	49950	0.1462	443582	0.2189	(1000, 100, 50)
BA3	1000	249750	0.6171	31524992	0.6748	(1000, 500, 250)
BA4	1000	4995	0.0156	617	0.0308	(1000, 10, 5)
BA5	1000	5975	0.0483	20355	0.4162	(1000, 50, 5)
BA6	1000	9450	0.1068	162689	0.7614	(1000, 100, 5)
BA7	1000	127250	0.6459	20711072	0.9861	(1000, 500, 5)
Enron	151	869	0.5018	1700	0.3441	–
Polbook	105	441	0.4875	560	0.3484	–
Polblog	1222	16714	0.3203	101043	0.2260	–
YesIWell	185	342	0.2018	223	0.1710	–

Table 5

Mean and standard deviation (*mean* ± *std*) of the absolute error for the clustering coefficient of one node (δ = 0.01)

ε	*direct*	$D\&\ C_{N3(i)}$	$D\&C_{d_i}$
0.01	0.4986 ± 0.1369	0.3656 ± 0.1568	0.4651 ± 0.1510
0.1	0.4695 ± 0.1594	0.3578 ± 0.1800	0.3772 ± 0.1942
1.0	0.0338 ± 0.0332	0.0629 ± 0.0611	0.0549 ± 0.0533
10	0.0036 ± 0.0036	0.0062 ± 0.0057	0.0055 ± 0.0053

Table 6

Mean and standard deviation (*mean* ± *std*) of all the absolute error for the clustering coefficient vector (δ = 0.01)

ε	*direct*	$D\&C_D$	$D\&C_{N_3}$
0.01	0.3257 ± 0.0045	0.2971 ± 0.0045	0.3269 ± 0.0040
0.1	0.1043 ± 0.0021	0.1069 ± 0.0028	0.1398 ± 0.0029
1.0	0.0118 ± 0.0002	0.0145 ± 0.0005	0.0182 ± 0.0006
10	0.0012 ± 0.0001	0.0015 ± 0.0001	0.0019 ± 0.0001

Table 7

The average entrywise absolute error for the clustering coefficient vector ($\delta = 0.01$)

Graphs	$\varepsilon = 0.01$		$\varepsilon = 0.1$		$\varepsilon = 1.0$		$\varepsilon = 10$	
	direct	*D&C$_D$*	*direct*	*D&C$_D$*	*direct*	*D&C$_D$*	*direct*	*D&C$_D$*
ER1	0.4920	0.0753	0.2499	0.0016	0.0438	4.08*e*-5	0.0055	4.07*e*-6
ER2	0.4947	0.0281	0.3402	2.56*e*-4	0.0793	2.58*e*-5	0.0096	2.57*e*-6
ER3	0.4968	0.0013	0.4697	1.27*e*-4	0.2832	1.26*e*-5	0.0393	1.27*e*-6
WS1	0.4730	0.0472	0.2633	0.0011	0.0423	5.78*e*-5	0.0049	5.81*e*-6
WS2	0.4844	0.0904	0.2522	0.0010	0.0455	9.09*e*-5	0.0047	9.10*e*-6
WS3	0.4830	0.0721	0.2881	0.0015	0.0424	1.55*e*-4	0.0042	1.54*e*-5
WS4	0.4864	0.0162	0.3729	7.55*e*-4	0.0823	7.55*e*-5	0.0083	7.57*e*-6
WS5	0.4967	0.0015	0.4682	1.54*e*-4	0.2769	1.55*e*-5	0.0382	1.55*e*-6
BA1	0.4856	0.1270	0.3389	0.0042	0.0754	4.26*e*-4	0.0102	4.53*e*-5
BA2	0.4890	0.0189	0.4005	0.0019	0.1274	1.93*e*-4	0.0164	1.93*e*-5
BA3	0.4969	0.0033	0.4739	3.28*e*-4	0.3071	3.29*e*-5	0.0499	3.30*e*-6
BA4	0.1899	0.1377	0.0623	0.0141	0.0094	5.51*e*-4	0.0011	5.48*e*-5
BA5	0.2136	0.1433	0.1110	0.0180	0.0182	0.0025	0.0023	2.45*e*-04
BA6	0.2473	0.1573	0.1634	0.0370	0.0334	0.0057	0.0043	5.86*e*-04
BA7	0.4792	0.2297	0.4357	0.1813	0.2012	0.0550	0.0284	0.0063
Enron	0.4758	0.4535	0.4283	0.1873	0.2026	0.0436	0.0253	0.0048
Polbook	0.4968	0.4860	0.4263	0.3442	0.1492	0.0438	0.0163	0.0045
Polblog	0.4025	0.2808	0.3537	0.1118	0.1392	0.0336	0.0161	0.0040
YesIWell	0.2253	0.2210	0.1866	0.1586	0.0583	0.0247	0.0065	0.0026