The VLDB Journal (2014) 23:653–676
DOI 10.1007/s00778-013-0344-8

REGULAR PAPER

# Correlated network data publication via differential privacy

**Rui Chen · Benjamin C. M. Fung · Philip S. Yu · Bipin C. Desai**

Received: 29 November 2012 / Revised: 22 October 2013 / Accepted: 24 October 2013 / Published online: 15 November 2013


**Abstract** With the increasing prevalence of information networks, research on privacy-preserving network data publishing has received substantial attention recently. There are two streams of relevant research, targeting different privacy requirements. A large body of existing works focus on preventing node re-identification against adversaries with structural background knowledge, while some other studies aim to thwart *edge disclosure*. In general, the line of research on preventing edge disclosure is less fruitful, largely due to lack of a formal privacy model. The recent emergence of *differential privacy* has shown great promise for rigorous prevention of edge disclosure. Yet recent research indicates that differential privacy is vulnerable to *data correlation*, which hinders its application to network data that may be inherently correlated. In this paper, we show that differential privacy could be tuned to provide provable privacy guarantees even in the correlated setting by introducing an extra parameter, which measures the extent of correlation. We subsequently provide a holistic solution for *non-interactive* network data publication. First, we generate a private vertex labeling for a given network dataset to make the corresponding adjacency matrix form dense clusters. Next, we adaptively identify dense regions of the adjacency matrix by a data-dependent partitioning process. Finally, we reconstruct a noisy adjacency matrix by a novel use of the exponential mechanism. To our best knowledge, this is the first work providing a practical solution for publishing real-life network data via differential privacy. Extensive experiments demonstrate that our approach performs well on different types of real-life network datasets.

**Keywords** Network data · Differential privacy · Data correlation · Non-interactive publication

R. Chen (✉)
Hong Kong Baptist University, Kowloon, Hong Kong
e-mail: ruichen@comp.hkbu.edu.hk

B. C. M. Fung
McGill University, Montreal, Canada
e-mail: ben.fung@mcgill.ca

P. S. Yu
University of Illinois at Chicago, Chicago, IL, USA
e-mail: psyu@cs.uic.edu

B. C. Desai
Concordia University, Montreal, Canada
e-mail: BipinC.Desai@concordia.ca

## 1 Introduction

In the last few years, information networks in various application domains, such as social networks, communication networks and transportation networks, have experienced vigorous developments. In particular, online social networks, such as Facebook, LinkedIn and Twitter, have become very prevalent. With the growth of information networks, a large volume of network data has been generated, which enables a wide spectrum of data analysis tasks. Network data is typically represented as graphs, where nodes represent a set of individuals with their attributes, and edges represent connections between them. In this paper, we use the terms *network data* and *graph* interchangeably.

It has been shown that, with naively sanitized network data (i.e., merely replacing explicit identifiers by pseudo-identifiers), an adversary is able to launch different types of privacy attacks that re-identify nodes, reveal edges between nodes or expose node attributes [20]. Therefore, network data needs to be sanitized with formal, provable privacy guarantees before it can be released to the public.

In addressing privacy issues in network data publication, there has been a series of research [4,6,8,16,17,26,27,34,37–40] based on different privacy models targeting different privacy requirements. Most of these works [6,17,26,38–40] focus on preventing node re-identification and/or associated attribute disclosure against adversaries with structural background knowledge. Some other papers [4,6,8,27,34,37] consider privacy threats due to *edge disclosure*, which lets an adversary learn the sensitive relationships between individuals (e.g., sexual relationships in a population [2]). However, these works on preventing edge disclosure lack a formal privacy definition for edge anonymity. Among all privacy models for network data, $k$-isomorphism [6] provides relatively strong privacy protection for edge disclosure (i.e., an adversary cannot determine if two individuals are connected via a path with probability $\geq \frac{1}{k}$). However, we show that, with moderate background knowledge, an adversary is able to ascertain a direct link between two individuals on a $k$-isomorphic graph, as illustrated in Example 1.

*Example 1* Consider the 3-isomorphic graph in Fig. 1. Suppose that the adversary has successfully identified Bob as one of $\{v_1, v_2, v_3\}$ and Ann as one of $\{v_4, v_5, v_6\}$ and seeks to learn if there is a direct link between Bob and Ann, which is considered sensitive. With the background knowledge that Bob and Ann are connected via a common friend, the adversary can ascertain that Bob and Ann are in the same subgraph, and therefore learn that there is a direct link between them.

The vulnerability of $k$-isomorphism is largely due to its deterministic nature. In contrast, *$\epsilon$-differential privacy* [11], a well-acknowledged privacy notion emerged recently, demands *inherent randomness* of a sanitization algorithm. It requires that the outcome of any analysis should be insensitive to the change of a single data record. It follows that even if a user had opted in a database, there would not be a significant change in any computation based on the database. Hence it assures every record owner that any privacy breach will not be a result of participating in the database.

$\epsilon$-differential privacy provides rigorous privacy guarantees on a database whose records are generated *independently*; however, its application to network data is hindered by the fact that network data may be inherently *correlated*.

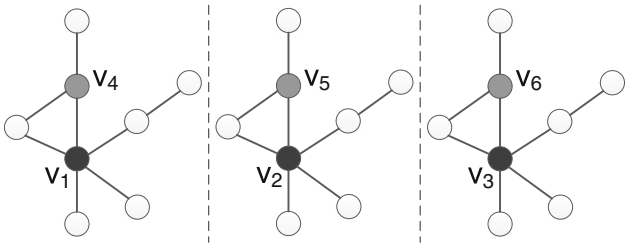



**Fig. 1** $k$-isomorphism is insufficient for preventing edge disclosure

In the context of network data, the *evidence of participation* [24] of a record (e.g., an edge) may be reflected by several other records. For example, the presence of an edge in a network database can be inferred by the existence of several other edges. The deletion of a single edge will not be able to fully mask its presence in the database. Consequently, $\epsilon$-differential privacy fails to provide the claimed privacy guarantee in the correlated setting.

In this paper, we analyze the property of differential privacy under correlation and show that $\epsilon$-differential privacy could be tuned to provide provable privacy guarantees even in the correlated setting by introducing an extra parameter, which measures the extent of correlation. Based on this parameter, we interpret differential privacy in the correlated setting in terms of *group differential privacy* [11] and demonstrate that differential privacy exhibits a monotonic property with respect to correlation. Analogous to the notion of *edge differential privacy* [16,18,22,31,32] that defines neighboring graphs as graphs differing in at most one edge, we consider its counterpart in the correlated setting. This notion prevents an adversary from learning the existence of *any single edge* in the correlated setting while allowing to preserve essential network structural properties for important data analysis tasks, such as degree distribution, cut query and shortest path length.

In addition to correlation, other major challenges of applying differential privacy to network data include scalability and utility. This is confirmed by the recent paper [16] that studies network data publication in the *non-interactive* setting under differential privacy. It requires the input graph to be *dense*, which is unlikely to be satisfiable on real-life network datasets, and leaves finding an efficient algorithm as an open problem. In this paper, we follow the line of *data-dependent* solutions [5,30], which adaptively make use of noisy information from the underlying database to improve efficiency and effectiveness. We first identify a good vertex labeling to make the adjacency matrix of an input graph form clustered dense regions, then explore these dense regions using an adapted quadtree, which avoids the high complexity of graph operations and leads to operations with smaller sensitivities, and finally design an efficient use of the exponential mechanism to reconstruct the quadtree's leaf nodes while satisfying $\epsilon$-differential privacy.

*Contributions* In this paper, we consider the problem of publishing useful network structures while preventing an attacker from learning the existence of any single edge even when the underlying edges are correlated. Our major contributions are summarized as follows.

– We analyze the privacy guarantee of $\epsilon$-differential privacy in the correlated setting (Sect. 4). We indicate that differential privacy could still provide provable guarantees by

introducing an extra correlation parameter to measure the extent of correlation. We unravel the privacy guarantee of differential privacy under correlation in terms of group differential privacy [11].
- We propose a practical *non-interactive* solution for network data publication, which prevents an adversary from learning the existence of a direct link between any two individuals (Sect. 5). Compared with the previous work [16], our improvements are significant: (1) we achieve $\epsilon$-differential privacy in the correlated setting, whereas Gupta et al. [16] achieve $(\epsilon, \delta)$-differential privacy, a weaker variant of $\epsilon$-differential privacy; (2) we lift the impractical assumption that the input graph has to be dense. We show that theoretically our approach obtains high utility as long as sufficient edge information is contained in some dense subgraphs and that experimentally our approach performs well on many different types of real-life network datasets; (3) most importantly, our approach is efficient to handle large real-life datasets.
- We conduct an extensive experimental study over various types of real-life network datasets, including social network, collaboration network and transportation network (Sect. 6). We examine the utility of sanitized network data for three different data analysis tasks, namely *degree distribution, cut query* and *shortest path length*. We demonstrate that our approach maintains high utility and scales to large real-life network data.
- We perform an experimental comparison between the $k$-isomorphic algorithm [6] and our approach from a utility-driven perspective (Sect. 6). The experimental results provide important insights for a data publisher to select a proper privacy model in different data publishing scenarios.

The rest of the paper is organized as follows. Section 2 reviews the related works under both partition-based privacy models and differential privacy. Section 3 presents the preliminaries of our approach and our utility requirements. Section 4 discusses the privacy guarantee of $\epsilon$-differential privacy in the correlated setting. Section 5 gives the main sanitization algorithm. Comprehensive experimental results and an experimental comparison with $k$-isomorphism are reported in Sect. 6. Section 7 concludes the paper.

## 2 Related work

Since Backstrom et al.'s [1] pioneering study of privacy attacks on social networks, the problem of *privacy-preserving network data publishing* has received increasing attention.

### 2.1 Network data sanitization under partition-based privacy models

A large line of research studies how to prevent a node from re-identification against an adversary with background knowledge on the network structure (and node attributes). Liu and Terzi [26] propose the notion of $k$-degree anonymity, which requires that, for every node $v$, there exist at least $k-1$ other nodes with the same degree as $v$. They achieve $k$-degree anonymity by a two-step approach, which first generates a $k$-anonymous degree sequence with the minimum number of edge modifications and then constructs a graph satisfying this degree sequence while making sure that the anonymized graph is a supergraph of the original graph.

Zhou and Pei [39] demand that any vertex cannot be re-identified in an anonymized graph with probability greater than $\frac{1}{k}$ by an adversary equipped with 1-neighborhood background knowledge. They propose a neighborhood component coding technique to compare the neighborhoods of all vertices and organize vertices with similar neighborhoods into the same group. Anonymization is then conducted within each group by generalizing vertex labels and adding edges.

Hay et al. [17] study privacy risks in network data by modeling adversarial knowledge as three types of *knowledge queries*, namely vertex refinement queries, subgraph queries and hub fingerprint queries. They propose an anonymization technique based on generalization. They generalize a graph by grouping nodes into partitions with size at least $k$, and only release information in the generalized level (e.g., the size of each partition and the density of edges within and across partitions).

Zou et al. [40] propose $k$-automorphism, which resists *any* structural attack by enforcing $k-1$ automorphic functions in the published data. This notion ensures that an adversary with any structural background knowledge cannot identify a node from the anonymized data with probability higher than $\frac{1}{k}$. They achieve $k$-automorphism by graph partitioning, graph alignment and edge copy.

Cheng et al. [6] present the notion of $k$-security based on $k$-isomorphism. It requires an input graph to be transformed into $k$ *disjoint* isomorphic subgraphs. The strong point of $k$-isomorphism is that it prevents not only node re-identification but also edge disclosure. However, as shown in Sect. 1, $k$-isomorphism's ability in preventing edge disclosure is limited.

Yuan et al. [38] introduce a framework that provides personalized privacy protection for labeled social networks. They define three levels of privacy protection requirements by modeling gradually increasing adversarial background knowledge. The framework combines label generalization and other structure protection techniques (e.g., adding nodes or edges) in order to achieve improved utility.

Another line of research aims at obfuscating an adversary's certainty of the presence of a link between two target nodes. Cormode et al. [8] consider the problem of anonymizing bipartite graph data based on a privacy model called $(k, l)$-grouping. They propose to preserve the entire graph structure by perturbing the mapping from entities to nodes.

Bhagat et al. [4] model a rich interaction graph as a bipartite graph over the set of entities and interactions. Their privacy requirement is to limit an attacker's confidence on an entity's participation in an interaction. They anonymize graphs using label lists based on a critical safety condition and then release only the number of edges among different node classes. Their idea is similar to that of Hay et al. [17], but differs in more rigorous privacy protection in the presence of node attributes.

Ying and Wu [37] study how random edge perturbation strategies affect both real and spectral characteristics of graphs and consequently develop two spectrum-preserving randomization mechanisms, *Spctr Add/Del* and *Spctr Switch*, which can better preserve graph spectral characteristics. They conduct privacy analysis on random edge perturbation strategies; however, it is not clear what privacy guarantees the two spectrum-preserving randomization mechanisms provide.

Liu et al. [27] consider a special situation where edges are weighted. They propose two privacy-preserving strategies, one based on a Gaussian randomization multiplication and the other based on a greedy perturbation algorithm, in order to preserve shortest paths between pairs of nodes. Wu et al. [34] present a low-rank-approximation-based reconstruction algorithm, which recovers spectral properties of a randomized graph. In general, all these works lack a formal privacy definition and are only resistant to certain types of privacy attacks.

### 2.2 Network data sanitization under differential privacy

With the recent wide acknowledgment of differential privacy, some papers [16,18,22,31,32] have started to apply differential privacy to network data from different perspectives. Hay et al. [18] propose two alternative formulations of differential privacy for network data, namely *edge differential privacy* and *node differential privacy* (see Sect. 3.3 for more discussion). They point out that node differential privacy may be too strong to provide desirable utility, and therefore, in reality, edge differential privacy is a more meaningful privacy notion. Based on edge differential privacy, they provide an efficient implementation of the constrained inference technique in [19] to release a private estimate of a network's degree distribution.

Similarly, other works [16,22,31,32] are also based on edge differential privacy. Karwa et al. [22] consider to output approximate answers to subgraph counting queries (e.g., triangles, $k$-triangles and $k$-stars). Their approaches rely on *local sensitivity* and *smooth sensitivity*, instead of global sensitivity, to add instance-dependent noise and, therefore, only satisfy the weaker $(\epsilon, \delta)$-differential privacy.

Sala et al. [32] propose to generate synthetic graphs by feeding differentially private degree correlation statistics (i.e., $dK$-2-series) into known graph generators. Due to the large sensitivity of $dK$-2-series, a partitioning approach is employed, which divides the $dK$-2-series into subseries and, for each subseries, adds Laplace noise of magnitude proportional to its local maximum degree. The resulting synthetic graphs are shown to be useful for degree-based metrics and node separation metrics; however, it is not clear whether this approach supports cut queries, which is the main utility metric considered in our paper. Though this paper opens an interesting direction for achieving differential privacy over network data, it is not clear how well it performs under stringent privacy parameter settings (e.g., $\epsilon \leq 1.0$) as it is only validated under relatively large privacy parameters (e.g., $\epsilon \in [5, 100]$).

Following the idea of adding *non-uniform* noise proposed in [32], Proserpio et al. [31] present a new workflow for differentially private graph synthesis. They develop the weighted PINQ language to produce private graph statistics and search for synthetic graphs whose accurate measurements fit those private statistics based on Markov chain Monte Carlo (MCMC). Similarly, the current version of their approach focuses on degree distribution, edge multiplicity and joint degree distribution and does not support cut queries.

The work closest to ours is by Gupta et al. [16]. They give new algorithms for approximately releasing the cut function of a graph in both interactive and non-interactive settings. The key idea is to pair an *iterative database construction* algorithm with a *distinguisher*, which returns a cut query with a significantly different value on an intermediate database, in order to give increasingly accurate approximations with respect to cut queries. The efficiency of the proposed approach counts on an efficient, private algorithm for computing accurate rank-1 matrix approximation; however, it is still unclear how to find such an algorithm. As mentioned in Sect. 1, there are several limitations that obstruct the application of the proposed approach to real-life network datasets.

## 3 Preliminaries

### 3.1 Adjacency matrix

In this paper, we follow the convention of modeling an input network dataset as a *simple graph* (i.e., an undirected graph with no loops or multiple edges), $G = (V, E)$, where $V$ is the set of vertices and $E \subseteq V \times V$ is the set of edges. For a graph $G$, we use $V(G)$ and $E(G)$ to respectively denote

the vertex set and the edge set of $G$. When the graph is clear from the context, we omit $G$ from the notation.

For the same graph, different *vertex labeling* schemes (i.e., different vertex orders) may lead to different adjacency matrices. For now, we assume that a vertex labeling has been given in a way that is independent of the underlying edge set (e.g., a random vertex labeling). Given a vertex labeling, the *adjacency matrix* of a simple graph $G = (V, E)$, denoted by $A(G)$, is a square $|V| \times |V|$ matrix satisfying:

$$A(G)_{ij} = \begin{cases} 1 & \text{if}(v_i, v_j) \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

It is evident that, for any simple graph $G$, $A(G)$ is a symmetric matrix with a zero diagonal. Figure 2 presents two possible adjacency matrices of a given simple graph under two different vertex labelings (ignore the bold lines in Fig. 2b for now). A (0, 1)-matrix is called a *graphic* matrix if it is an adjacency matrix of some simple graph. Apparently, a (0, 1)-matrix is graphic if and only if it is a symmetric matrix with a zero diagonal.

We define the density of a region $R \subseteq A$ with size $|R| = m \times l$ to be $\text{den}(R) = \sum_{i=1}^{m} \sum_{j=1}^{l} R_{ij}/ml$. It gives the fraction of elements in $R$ which are equal to 1. The region in $A$ formed by rows $i, i+1, \ldots, j$ and columns $m, m+1, \ldots, n$ is denoted by $A[i, j; m, n]$. For example, in Fig. 2b the densities of $A$ and $A[1, 3; 6, 8]$ are $20/64 = 31.25\,\%$ and $8/9 = 88.89\,\%$, respectively.

### 3.2 Differential privacy

$\epsilon$-differential privacy is defined based on the concept of *neighboring* databases. Two databases, $D_1$ and $D_2$, are neighbors if they differ on *at most* one record, denoted by $|D_1 \Delta D_2| = 1$. $\epsilon$-differential privacy requires that anything that is learnable from a database $D$ is also learnable from any of its neighbors. This implies that any computation is insensitive to the removal or addition of a single database record. Formally, $\epsilon$-differential privacy [11] is defined below.

**Definition 1** ($\epsilon$-*differential privacy*) A privacy mechanism $\mathcal{A}$ gives $\epsilon$-differential privacy if for any databases $D_1$ and $D_2$ s. t. $|D_1 \Delta D_2| = 1$, and for any possible output $O \in \text{Range}(\mathcal{A})$,

$$Pr\left[\mathcal{A}(D_1) = O\right] \leq e^{\epsilon} \times Pr\left[\mathcal{A}(D_2) = O\right]$$

where the probability is taken over the randomness of $\mathcal{A}$.

Two standard techniques, the *Laplace mechanism* [11] and the *exponential mechanism* [29], have been proposed in the literature for achieving $\epsilon$-differential privacy. A fundamental concept of both mechanisms is the *global sensitivity* [11] of a function that maps an underlying database to real values.

**Definition 2** (*Global sensitivity*) For any function $f : D \rightarrow \mathbb{R}^d$, the global sensitivity of $f$ is

$$GS(f) = \max_{D_1, D_2} ||f(D_1) - f(D_2)||_1$$

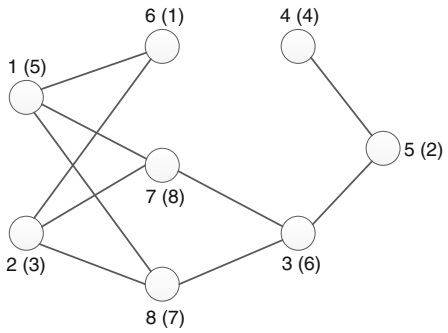
for all $D_1, D_2$ s. t. $|D_1 \Delta D_2| = 1$.

*Laplace mechanism* For the analysis whose outputs are real, Dwork et al. [11] propose the Laplace mechanism, which takes as inputs a database $D$, a function $f$, and the privacy parameter $\epsilon$ and returns the true output of $f$ plus some Laplace noise. The noise is drawn from a Laplace distribution with the probability density function $p(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}$, where $\lambda$ is determined by both $GS(f)$ and the desired privacy level $\epsilon$.

**Theorem 1** *For any function $f : D \rightarrow \mathbb{R}^d$ over an arbitrary domain $D$, the mechanism $\mathcal{A}$*

$$\mathcal{A}(D) = f(D) + Laplace(GS(f)/\epsilon)$$

*gives $\epsilon$-differential privacy.*




**(a)** A sample graph $G$ with two vertex labelings

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 6 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 8 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

**(b)** An adjacency matrix of $G$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 8 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

**(c)** Another adjacency matrix of $G$

**Fig. 2** A sample graph and its two adjacency matrices under different vertex labelings

*Exponential mechanism* For the analysis whose outputs are not real or make no sense after adding noise, McSherry and Talwar [29] propose the exponential mechanism that selects an output from the output domain, $r \in \mathcal{R}$, by taking into consideration its score of a given utility function $q$ in a differentially private manner. The exponential mechanism assigns exponentially greater probabilities of being selected to outputs of higher scores so that the final output would be close to the optimum with respect to $q$. The chosen utility function $q$ should be insensitive to changes in any particular record, that is, has a low sensitivity. Let the sensitivity of $q$ be $GS(q) = \max_{\forall r, D_1, D_2} |q(D_1, r) - q(D_2, r)|$. A formal definition follows.

**Definition 3** (*Exponential mechanism*) Given a utility function $q : (D \times \mathcal{R}) \to \mathbb{R}$ for a database instance $D$, the mechanism $\mathcal{A}$,

$$\mathcal{A}(D, q) = \left\{ \text{return } r \text{ with probability} \propto \exp\left(\frac{\epsilon q(D, r)}{2GS(q)}\right) \right\}$$

gives $\epsilon$-differential privacy.

*Composition properties* $\epsilon$-differential privacy exhibits desirable composition properties for a sequence of computations. Any sequence of computations that each provides differential privacy in isolation also provides differential privacy in sequence, which is known as *sequential composition* [28].

**Theorem 2** *Let $\mathcal{A}_i$ each provide $\epsilon_i$-differential privacy. A sequence of $\mathcal{A}_i(D)$ over the database $D$ provides $\sum \epsilon_i$-differential privacy.*

In some special cases, in which a sequence of computations is conducted on *disjoint* databases, the privacy cost does not accumulate, but depends only on the worst guarantee of all computations. This is known as *parallel composition* [28].

**Theorem 3** *Let $\mathcal{A}_i$ each provide $\epsilon_i$-differential privacy. A sequence of $\mathcal{A}_i(D_i)$ over a set of disjoint databases $D_i$ provides $\max(\epsilon_i)$-differential privacy.*

### 3.3 Edge differential privacy

The privacy protection provided by $\epsilon$-differential privacy rests on the interpretation of *neighboring databases*. Since $\epsilon$-differential privacy guarantees that an attacker is not able to distinguish neighboring databases, what is being protected is precisely the difference between two neighboring databases [18]. For relational data, the definition of a neighboring database is based on a single record difference, which entirely encapsulates an individual's private information. For network data, an analogous definition could be obtained by defining two neighboring graphs as two differing in at most one node and *all* its incident edges. In this case, a node's participation in a graph could be totally hidden from an attacker. Hay et al. [18] call the privacy model based on this notion of neighboring graphs *node differential privacy*. While node differential privacy provides desirable privacy protection, it is, in general, infeasible to provide guaranteed utility due to the huge sensitivity (consider the extreme case that the additional node connects to all other nodes).

Another formulation of neighboring databases for network data is defined as graphs differing in at most an edge (or one *isolated* node). This definition leads to *edge differential privacy* [18]. Intuitively, edge differential privacy protects any single edge from being unveiled. Though weaker than node differential privacy, this adaptation still provides meaningful privacy protection for many applications. An example was given in [18] based on the study of Kossinets and Watts [25], where direct e-mail communication between students and faculty in a large university is considered sensitive. For this reason, edge differential privacy has been considered a reasonable adaptation of differential privacy for network data and has been consequently employed in subsequent works [16,22,31,32]. In this paper, we also consider an analogous notion of edge differential privacy in the correlated setting.

A natural extension to edge differential privacy is also introduced in [18], known as *k-edge differential privacy*. It allows neighboring graphs to differ by at most $k$ edges and therefore prevents leakage of aggregate information of any subset of $k$ edges. We stress that the purpose of $k$-edge differential privacy is to protect $k$ edges' collective information and is *not* to hide the presence of *any single edge* in the correlated setting, which is the privacy goal of this paper.

### 3.4 Utility metrics

Our general goal is to generate a sanitized graph $\widetilde{G}$, whose adjacency matrix $\widetilde{A}$ minimizes $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |A_{ij} - \widetilde{A}_{ij}|$ (i.e., $\widetilde{A}$ is as close to $A$ as possible). When $\widetilde{A}$ is identical to $A$, $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |A_{ij} - \widetilde{A}_{ij}| = 0$; when $\widetilde{A}$ is totally different from $A$, $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |A_{ij} - \widetilde{A}_{ij}| = |V|^2 - |V|$.

Minimizing $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |A_{ij} - \widetilde{A}_{ij}|$ naturally makes the published network data useful for many analysis tasks, including *degree distribution, cut query* and *shortest path length*.

#### *3.4.1 Degree distribution*

Given a graph $G$, we use a vector $F(G)$ of size $|V(G)|$ to denote the *degree frequency sequence* of $G$. The $i$-th value in $F(G)$ is

$$\frac{|\{v \in V : \deg(v) = i\}|}{|V|},$$

where $\deg(v)$ is the degree of $v$. For example, the degree frequency sequence of the graph in Fig. 2 is {0, 0.125, 0.25, 0.625, 0, 0, 0, 0}.

Given the degree frequency sequences of the original graph and the sanitized graph, $F(G)$ and $F(\widetilde{G})$, we measure their difference by *Kullback–Leibler divergence* (KL-divergence) [23]:

$$D_{KL}(F(G)||F(\widetilde{G})) = \sum_{i=0}^{|V|-1} F(G)[i]\log\frac{F(G)[i]}{F(\widetilde{G})[i]}.$$

If $F(G) = F(\widetilde{G})$, $D_{KL}(F(G)||F(\widetilde{G})) = 0$. We follow the standard convention that $0\log 0 = 0$.

### 3.4.2 Cut query

In this paper, a *cut* of a graph $G$ is defined by any two subsets of vertices $S, T \subseteq V(G)$ [16]. A *cut query* returns the number of edges in the *cut-set* of a cut, that is, $Q_{S,T}(G) = |\{(u, v) \in E(G) : u \in S, v \in T\}|$. For example, given the graph in Fig. 2, $S = \{v_1, v_2\}$ and $T = \{v_6, v_7, v_8\}$, we have $Q_{S,T}(G) = 6$.

We measure the utility loss of a cut query over a sanitized graph $\widetilde{G}$ by its *relative error*, with respect to the true count over the original graph $G$, which is computed as:

$$\text{error}(Q_{S,T}(\widetilde{G})) = \frac{|Q_{S,T}(\widetilde{G}) - Q_{S,T}(G)|}{\max\{Q_{S,T}(G), s\}},$$

where $s$ is a *sanity bound* that mitigates the effect of the queries with extremely small *selectivities* [5,35,36].

### 3.4.3 Shortest path length

In a simple graph $G = (V, E)$, a path between two vertices $v_1$ and $v_n$ is defined to be a sequence of vertices $P = (v_1, v_2, \ldots, v_n) \in V \times V \times \cdots \times V$ such that $(v_i, v_{i+1}) \in E(G)$ for $1 \leq i < n$. The *length* of a path is the number of edges involved in its sequence of vertices. For example, the path between vertices $v_1$ and $v_4$ in Fig. 2, $(v_1, v_7, v_3, v_5, v_4)$, is of length 4.

The *shortest path length* between two vertices $v_i$ and $v_j$ is the minimum length of *all* paths between them. For example, the shortest path length between vertices $v_1$ and $v_4$ in Fig. 2 is 4. If there is no path between two vertices, the shortest path length is defined to be $\infty$. The shortest path between two vertices can be efficiently computed by *breadth-first search* (BFS) [7].

## 4 Differential privacy under correlation

$\epsilon$-differential privacy is built on the assumption that all underlying records are independent of each other. In the context of network data, this assumption does not always hold. Kifer and Machanavajjhala [24] indicate that, in the *correlated setting*, $\epsilon$-differential privacy cannot provide the claimed privacy protection because the removal of a single record cannot hide its *evidence of participation* (e.g., its presence in the database could still be inferred by the existence of some other records to which it is correlated). We first revisit the example provided in [24] to see how correlation could enhance an adversary's ability in differentiating two neighboring databases.

*Example 2* [1]Bob or one of his 9 immediate family members may have contracted a highly infectious disease, in which case the entire family would have been infected. An attacker asks the query "how many in Bob's family have this disease?" to infer if Bob has been infected. The true answer is of high probability to be either 0 or 10. Suppose the noisy answer returned is 12. If this answer is obtained by adding Laplace noise calibrated by the specified privacy parameter $\epsilon$, the attacker learns that the probability of 10 being the true answer is $e^{10\epsilon}$ times larger than the probability of 0 being the true answer, violating the expected privacy guarantee.

Example 2 suggests that the attacker's ability of distinguishing two neighboring databases (with and without Bob's record) is augmented ten times due to the presence of correlation. A natural question to ask is that *if the extent of correlation could be measured or upper bounded, can differential privacy provide guaranteed privacy under correlation*? Below we give a positive answer to this question, which provides a promise for publishing correlated network data under differential privacy. To this end, we introduce a new correlation parameter $k$ to measure the extent of a record's correlation to other records.

A database $D$ with a correlation parameter $k$ means that any record in $D$ is correlated to *at most* $k - 1$ other records. We assume that the database size is greater than $k$. The introduction of the correlation parameter $k$ allows us to generalize $\epsilon$-differential privacy [11], where $k = 1$, and free-lunch privacy [24], where $k$ equals the database size, into the same framework. The $k$ value may vary from application to application. Typically, the $k$ value is domain specific and does *not* depend on a specific database. In Sect. 6, we experimentally show that our solution can preserve useful information even when $k$ is relatively large (e.g., $k = 25$), demonstrating its usefulness in many real-life applications.

Given a privacy parameter $\epsilon$ and a correlation parameter $k$, our goal is to provide a similar differential privacy guarantee in the correlated setting, leading to the following definition.

[1] The strong attacker mentioned in [24] cannot be prevented as his prior knowledge (without accessing any database) has allowed him to succeed in a privacy attack. Such a privacy attack is not caused by data sharing and therefore beyond the scope of this paper.

**Definition 4** (*$\epsilon$-differential privacy under correlation*) A privacy mechanism $\mathcal{A}$ satisfies $\epsilon$-differential privacy if for any two databases $D_1$ and $D_2$ with a correlation parameter $k$ and $|D_1 \Delta D_2| = 1$, and for any possible output $O \in \text{Range}(\mathcal{A})$,

$$Pr[\mathcal{A}(D_1) = O] \leq e^{\epsilon} \times Pr[\mathcal{A}(D_2) = O]$$

where the probability is taken over the randomness of $\mathcal{A}$.

As illustrated in Example 2, adding noise calibrated by the global sensitivity and the privacy parameter $\epsilon$ cannot guarantee $\epsilon$-differential privacy under correlation any more. The correlation in a database makes a function more "sensitive" to the change of a single record because its output is affected by not only this record but also all of its correlated records. Here we make the key observation that, no matter how records are correlated, it is sufficient to cancel out the effect of data correlation on any computation by considering all correlated records (*at most* $k$ records) as if they were totally removed from the underlying database. This observation allows differential privacy under correlation to be interpreted in terms of *group differential privacy* [11]: any $\frac{\epsilon}{k}$-differentially private mechanism also guarantees $\epsilon$-differential privacy for databases that differ in $k$ records. This implies that a privacy mechanism that achieves $\frac{\epsilon}{k}$-differential privacy in the *non-correlated setting* is sufficient to give $\epsilon$-differential privacy over a database with a correlation parameter $k$.

Following this idea, we now show how calibrating noise by taking into consideration the correlation parameter $k$ (i.e., decreasing $\epsilon$ proportional to $k$) helps thwart the privacy risk due to data correlation.

*Example 3* Continue with Example 2. Consider a privacy mechanism that achieves $\frac{\epsilon}{k}$-differential privacy (in this case, $k \geq 10$). We have

$$\frac{Pr[c = 10|\widetilde{c} = 12]}{Pr[c = 0|\widetilde{c} = 12]} \leq \frac{\exp(-\frac{\epsilon|10-12|}{10})}{\exp(-\frac{\epsilon|0-12|}{10})} \leq \exp(\epsilon).$$

That is, it enjoys $\epsilon$-differential privacy over a database with a correlation parameter $k$.

Differential privacy exhibits a nice monotonic property with respect to correlation.

**Theorem 4** *An $\epsilon$-differentially private mechanism $\mathcal{A}$ over a database with a correlation parameter $k$ also gives $\epsilon$-differential privacy for all databases with a correlation parameter $k'$, where $1 \leq k' \leq k$.*

This is true because $\frac{Pr[\mathcal{A}(D_1)=O]}{Pr[\mathcal{A}(D_2)=O]} \leq e^{\frac{\epsilon}{k}} \leq e^{\frac{\epsilon}{k'}}$. Theorem 4 indicates that if $k$ is specified as the upper bound of correlation in a series of computation, the sequential composition property of differential privacy still holds in the correlated setting.

**Algorithm 1** DER Algorithm

**Input:** Raw graph $G$
**Input:** Privacy budget $\epsilon$
**Input:** Correlation parameter $k$
**Output:** Sanitized graph $\widetilde{G}$
1: $\frac{\epsilon}{k} = \epsilon_I + \epsilon_E + \epsilon_A$;
2: Vertex labeling $\mathcal{L} \leftarrow$ *IdentifyVertexLabeling*$(G, \epsilon_I)$;
3: Generate the adjacency matrix $A$ from $G$ based on $\mathcal{L}$;
4: Noisy quadtree $\mathcal{QT} \leftarrow$ *ExploreDenseRegion*$(A, \epsilon_E)$;
5: Sanitized matrix $\widetilde{A} \leftarrow$ *ArrangeEdge*$(\mathcal{QT}, A, \epsilon_A)$;
6: Generate $\widetilde{G}$ from $\widetilde{A}$;
7: **return** $\widetilde{G}$;

*Edge differential privacy under correlation* In the rest of this paper, we consider the counterpart of edge differential privacy (explained in Sect. 3.3) in the correlated setting. Recall that in this case two neighboring databases are defined as two databases that differ on at most one *edge*. This instantiation of differential privacy prevents an adversary from learning the presence of any *single* edge (i.e., if two individuals are directly connected) even when its existence can be inferred by $k - 1$ other edges. This notion hence provides "equal" privacy protection to every vertex, regardless of its degree. That is, every edge of every vertex is protected. Again we stress that the privacy requirement of edge differential privacy under correlation is different from $k$-edge differential privacy. To protect $k$ edges' collective information in the correlated setting, extra noise needs to be injected.

## 5 Private network publication

### 5.1 Overview

We first provide an overview of our solution, called *density-based exploration and reconstruction* (*DER*), in Algorithm 1. It takes as inputs a graph $G$, a privacy budget $\epsilon$ and a correlation parameter $k$ and returns a sanitized graph $\widetilde{G}$ satisfying $\epsilon$-differential privacy over databases with a correlation parameter $\leq k$. Our solution consists of three main steps. As explained in Sect. 4, we adjust the real privacy budget to $\frac{\epsilon}{k}$ to cancel out the effect of correlation and then divide $\frac{\epsilon}{k}$ into three portions, $\epsilon_I$, $\epsilon_E$ and $\epsilon_A$, each being used in a step.

In the first step *IdentifyVertexLabeling*, we aim to identify a good vertex labeling that makes the corresponding adjacency matrix form dense clusters of 1s. Existing approaches that can serve this purpose are highly sensitive to the presence of an edge and therefore are difficult to achieve differential privacy with acceptable utility. We develop an effective greedy algorithm that iteratively permutes pairs of vertices from a random vertex labeling for better density contrast.

In the second step *ExploreDenseRegion*, we design a differentially private and data-dependent partitioning process by adapting a standard quadtree to explore dense regions of the adjacency matrix $A$ of $G$, which can be reconstructed later with high accuracy. This process results in a noisy quadtree $\mathcal{QT}$ whose nodes represent a region of $A$ and are associated with a noisy count. The major technical challenges in this step include the design of stop conditions based on an accurate estimation of the height of $\mathcal{QT}$, the selection of splitting points based on the exponential mechanism, an adaptive privacy budget allocation scheme and an efficient implementation, each of which is key to the success of the entire algorithm.

In the third step *ArrangeEdge*, we propose an efficient edge arrangement algorithm to reconstruct a noisy, graphic matrix $\widetilde{A}$ that minimizes $\sum_{i=1}^{|V|}\sum_{j=1}^{|V|}|A_{ij} - \widetilde{A}_{ij}|$. Our method is based on a novel use of the exponential mechanism, which provides an efficient solution for an extremely large output domain. It successfully reduces the run-time complexity to $O(|V|^2)$, in contrast to the *factorial complexity* of a naive implementation.

We illustrate the general idea of Algorithm 1 in the example below.

*Example 4* Consider the graph given in Fig. 2. Our approach starts by generating a random vertex labeling $\mathcal{L}$. Algorithm 1 iteratively swaps pairs of vertices in $\mathcal{L}$ to form dense regions in the adjacency matrix. Suppose the final vertex labeling is the one given in Fig. 2a (outside the brackets). Algorithm 1 generates the corresponding adjacency matrix, as illustrated in Fig. 2b. It then employs a quadtree structure to explore the dense regions and calculate the number of 1s in them. This process is illustrated by the bold lines in Fig. 2b. Finally, Algorithm 1 makes use of the exponential mechanism to reconstruct the regions. A possible sanitized adjacency matrix and the corresponding graph are given in Fig. 3. The effect of correlation is diminished in this process by adding extra Laplace noise that always hides the presence of *any* $k$ correlated edges.

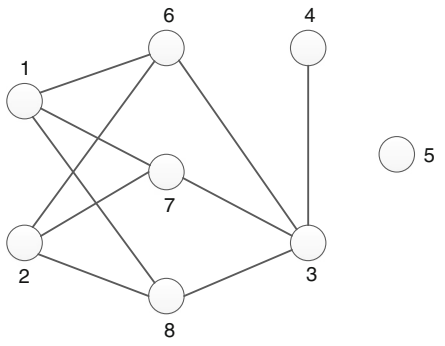
## 5.2 Vertex labeling identification

Real-life graphs often have dense clusters, that is, they satisfy the assumption of our approach that sufficient edge information is contained in some dense subgraphs. However, the reflection of these dense clusters in the corresponding adjacency matrix relies on a good vertex labeling. As an illustration, in Fig. 2b, c, we give two adjacency matrices of the same graph based on two different vertex labelings. We can observe that the elements of value 1 (i.e., the edges) in Fig. 2b cluster better than those in Fig. 2c. Since our approach needs to explore the dense regions of an adjacency matrix, our first objective is to identify a vertex labeling that makes the adjacency matrix exhibit high density contrast.

Without the privacy requirement, this task could be achieved by, for example, the Cuthill–McKee algorithm [10] and many other methods proposed for graph clustering [33] and community detection [14]. However, most of these approaches face a major challenge when adapted to a differentially private version: They are overly sensitive to the presence/absence of an edge. For example, the Cuthill–McKee algorithm relies on BFS, whose result could be substantially different under the change of an edge. Hence their differentially private variants are difficult to achieve desirable utility under small privacy parameters. In this paper, we propose a simple greedy algorithm that performs well under differential privacy.

Given a vertex labeling, we need to quantify how well its corresponding adjacency matrix forms dense clusters. Note that this quality metric is of purpose different from that of either graph clustering or community detection. It suffices if Algorithm 1 as a whole achieves good utility for the analysis tasks introduced in Sect. 3.4. For this reason, we aim to simply find a vertex labeling that places all 1s in the adjacency matrix as close as possible to a designated area (e.g., the central point or the main diagonal). The designated area could be arbitrary, but its resultant global sensitivity is an important factor to consider.

**Fig. 3** A possible sanitized graph and its adjacency matrix




**(a)** The sanitized graph $\widetilde{G}$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 3 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 8 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

**(b)** The adjacency matrix $A$ ($\widetilde{G}$)

**Procedure 1** *IdentifyVertexLabeling* Procedure

**Input:** Raw graph $G$
**Input:** Privacy budget $\epsilon_I$
**Output:** Vertex labeling $\mathcal{L}$
1: i = 0;
2: Generate a random vertex labeling $\mathcal{L}$;
3: **while** $i < t$ **do**
4: Generate a candidate set $\mathcal{C}$ of swaps;
5: **for** each $\chi(v_m, v_l) \in \mathcal{C}$ **do**
6: **if** $NoisyCount(q(\chi(v_m, v_l)), \frac{\epsilon_I}{t}) \geq 0$ **do**
7: Perform $\chi(v_m, v_l)$ over $\mathcal{L}$;
8: $i++$;
9: **return** $\mathcal{L}$;

In this paper, we choose to use the central point $\left(\left\lceil \frac{|V|}{2} \right\rceil, \left\lceil \frac{|V|}{2} \right\rceil\right)$ of the adjacency matrix, which leads to a smaller sensitivity (half of that of the main diagonal). Formally, the quality of a vertex labeling $\mathcal{L}$ for the graph $G = (V, E)$ is measured by:

$$q(\mathcal{L}) = \sum_{i,j} \frac{A_{ij}}{|V| - 2} \cdot \left( \left| i - \left\lceil \frac{|V|}{2} \right\rceil \right| + \left| j - \left\lceil \frac{|V|}{2} \right\rceil \right| \right)$$

where $\left| i - \left\lceil \frac{|V|}{2} \right\rceil \right| + \left| j - \left\lceil \frac{|V|}{2} \right\rceil \right|$ is the Manhattan distance of $A_{ij}$ to the center $(\left\lceil \frac{|V|}{2} \right\rceil, \left\lceil \frac{|V|}{2} \right\rceil)$, and $|V| - 2$ is a normalization constant. We choose Manhattan distance, instead of Euclidean distance, for the reason of simplicity. Slightly abusing the term, we call this quality metric *centrality*. A smaller centrality $q(\mathcal{L})$ indicates better quality, that is, the 1s in the adjacency matrix are closer to the center.

For a graph $G = (V, E)$, there are a total of $|V|!$ possible vertex labelings. It is computationally infeasible to exhaustively search for the best labeling. Here we propose a swap-based greedy algorithm that finds a reasonably good labeling from a random vertex labeling.[2] The idea is to iteratively swap pairs of vertices in order to achieve a smaller centrality. We denote the swap operation between two vertices $v_i$ and $v_j$ by $\chi(v_i, v_j)$. Note that for a simple graph $\chi(v_i, v_j)$ is essentially the same as $\chi(v_j, v_i)$. The *IdentifyVertexLabeling* procedure is detailed in Procedure 1.

The number of iterations $t$ (Line 3) involves a trade-off between the number of swaps to execute and the magnitude of Laplace noise added to each iteration. A larger $t$ value allows more swaps to be performed, but makes each swap less accurate. Since here Laplace noise dominates the quality of the resultant labeling, we prefer to select a small $t$ value (e.g., 5–10). In the following, we set $t = 5$.

In each iteration, ideally, we should consider all pairwise swaps (of number $\frac{|V|(|V|-1)}{2}$) and perform those improving centrality. However, this method introduces a huge sensitivity because a single edge could impact $2(|V| - 1)$ swaps. To reduce the sensitivity, we only consider a restricted set of swaps $\mathcal{C}$ such that every vertex is only involved in one swap in $\mathcal{C}$ (Line 4). We randomly select pairs of vertices to form $\mathcal{C}$. Essentially, this could be considered as a sampling process over all possible pairwise swaps. It makes sure that a single edge can influence *at most* two swaps in $\mathcal{C}$. For each swap $\chi(v_m, v_l) \in \mathcal{C}$, we calculate the centrality change between the labelings before and after the swap, denoted by $q(\chi(v_m, v_l))$. To satisfy differential privacy, Laplace noise is added to $q(\chi(v_m, v_l))$ (Line 6). We quantify the sensitivity of $q(\chi(v_m, v_l))$ below.

**Theorem 5** *Let $Q$ be the set of $q(\chi(v_m, v_l))$ queries for all $\chi(v_m, v_l) \in \mathcal{C}$. $GS(Q) = 2$.*

*Proof* Let two neighboring graphs $G_1$ and $G_2$ differ in edge $(v_i, v_j)$. Without loss of generality, let $E(G_1) = E(G_2) \cup (v_i, v_j)$. Due to the construction of $\mathcal{C}$, at most two queries involving either $v_i$ or $v_j$ in $Q$ will be affected by edge $(v_i, v_j)$.[3] We denote these two queries by $q_i$ and $q_j$, respectively. Therefore, the global sensitivity of $Q$ is:

$$\begin{aligned} GS(Q) &= \max_{G_1,G_2} \sum_{q \in Q} |q(G_1) - q(G_2)| \\ &= \max_{G_1,G_2} \left( |q_i(G_1) - q_i(G_2)| + |q_j(G_1) - q_j(G_2)| \right) \end{aligned}$$

Let $\chi(v_i, v_x) \in \mathcal{C}$ and $\chi(v_j, v_y) \in \mathcal{C}$. For ease of exposition, let $\left\lceil \frac{|V|}{2} \right\rceil$ be denoted by $c$. We have:

$$\begin{aligned} &\max_{G_1,G_2} \left( |q_i(G_1) - q_i(G_2)| + |q_j(G_1) - q_j(G_2)| \right) \\ &= \frac{2}{|V| - 2} \max_{G_1,G_2} \left( |(|i - c| + |x - c|) - (|i - c| + |j - c|)| \right. \\ &\quad \left. + |(|j - c| + |y - c|) - (|i - c| + |j - c|)| \right) \\ &= \frac{2}{|V| - 2} \max_{G_1,G_2} \left( ||x - c| - |j - c|| + ||y - c| - |i - c|| \right) \end{aligned}$$

The multiple 2 comes from the fact that each of $\chi(v_i, v_x)$ and $\chi(v_j, v_y)$ affects two rows and two columns. Since $\{x, y, i, j\} \in [1, |V|]$ and $v_i \neq v_j \neq v_x \neq v_y$,

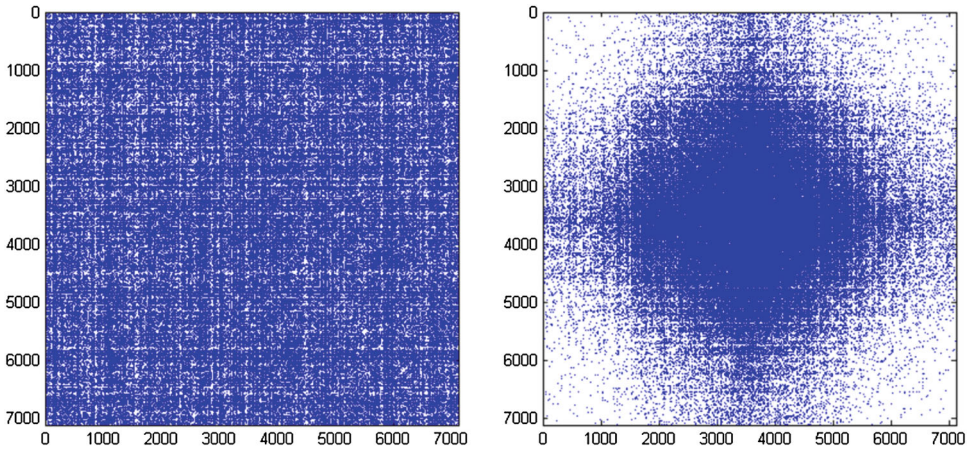
$$\begin{aligned} &\frac{2}{|V| - 2} \max_{G_1,G_2} \left( ||x - c| - |j - c|| + ||y - c| - |i - c|| \right) \\ &\leq \frac{2}{|V| - 2} \left( \frac{|V|}{2} + \frac{|V|}{2} - 2 \right) \\ &= 2 \end{aligned}$$

This completes the proof. □

In Line 7, we perform the swaps that improve the centrality of the vertex labeling $\mathcal{L}$. We illustrate the efficacy of

[2] A random vertex labeling satisfies 0-differential privacy because it can be viewed as an application of the exponential mechanism with privacy parameter 0.

[3] If $v_i$ and $v_j$ are selected to swap with each other, only one query will be affected. This leads to a lower global sensitivity.




**(a)** An adjacency matrix **(b)** The permuted adjacency matrix

**Fig. 4** The adjacency matrices over a random vertex labeling and the permuted vertex labeling, where each blue dot represents an element with value 1. $\epsilon = 1.0$ is used

*IdentifyVertexLabeling* over a real-life dataset *wiki-Vote* (see Sect. 6 for a description of *wiki-Vote*) in Fig. 4.

It is interesting to point out that, in spite of the noise added, *IdentifyVertexLabeling* itself does *not* incur any utility loss with respect to the three data analysis tasks introduced in Sect. 3.4. This is because no matter what the vertex labeling is, the corresponding adjacency matrix still represents the same graph. All the purpose of *IdentifyVertexLabeling* is to help improve the utility resulted from the subsequent procedures.

*Run-time complexity* The run-time complexity of Procedure 1 is $O(|V|^2)$. This is because, for each iteration, we generate $\frac{|V|}{2}$ candidate swaps, and for each swap, the calculation of its centrality change involves exactly two rows and two columns, which is of complexity $O(|V|)$.

**Procedure 2** *ExploreDenseRegion* Procedure

**Input:** Raw adjacency matrix $A$
**Input:** Privacy budget $\epsilon_E$
**Output:** Noisy quadtree $\mathcal{QT}$
1: i = 0;
2: $\mathcal{QT} \leftarrow \emptyset$;
3: Calculate the height $h$ of $\mathcal{QT}$;
4: **while** $i < h$ **do**
5: **if** $i = 0$ **then**
6: Insert a node representing $A$ to $\mathcal{QT}$;
7: **for** each non-leaf node $u \in level(i, \mathcal{QT})$ **do**
8: Calculate privacy budget portion $\epsilon_u^c$ and $\epsilon_u^p$;
9: Subregions $\mathcal{R} \leftarrow partition(u, \epsilon_u^p)$;
10: **for** each $R \in \mathcal{R}$ **do**
11: $\widetilde{c} = NoisyCount(R, \epsilon_u^c)$;
12: Insert a node $v$ representing $R$ to $\mathcal{QT}$;
13: **if** $v$ meets *stop condition* **then**
14: Mark $v$ as leaf;
15: $i++$;
16: **return** $\mathcal{QT}$;

## 5.3 Dense region exploration

After constructing the adjacency matrix $A$ based on the vertex labeling identified, we perform a recursively partitioning process guided by *density* in order to identify dense regions (and, implicitly, sparse regions) of $A$, which can be reconstructed accurately. This process could be supported by many popular space-partitioning data structures, such as *kd-tree* [3], *quadtree* [13] and *Hilbert R-tree* [21]. In this paper, we employ *quadtree* as the basic data structure for the exploration because it achieves the best trade-off between utility and efficiency.

A standard quadtree decomposes a given two-dimensional region into four *equal* quadrants, subquadrants, and so on until each leaf node meets certain *stop condition*. The splitting point of a standard quadtree is independent of the input data. It always selects the midpoint of each dimension to split. Each node in a quadtree represents a region of $A$. For our task, we adapt a quadtree in a data-dependent, differentially private manner. Each node (except the root) in a quadtree records not only the region it represents, but also the noisy count of the number of 1s in its region. We slightly abuse the term *count* to mean the number of 1s in a region. Procedure 2 presents the details of *ExploreDenseRegion*.

*Stop condition* One key problem in the partitioning process is to determine the height of the quadtree. Previous works [9, 30] normally require a data publisher to specify the height. In our paper, we calculate a good estimate of the height based on other inputs (Line 3). It is very difficult to calculate a very precise height of a *data-dependent* quadtree under our *adaptive* privacy budget allocation scheme. Instead, we use a standard quadtree with the *geometric budget scheme* [9] to derive a reasonably good estimate. A geometric budget scheme assigns all nodes on the same level $i$ the same privacy budget $\epsilon_i$ and increases the budget by a factor of $2^{1/3}$ with the increase of nodes' depth.

**Theorem 6** [9] *Given the privacy budget $\epsilon_{cnt}$, the geometric budget scheme, assigning $\frac{2^{i/3}(\sqrt[3]{2}-1)\epsilon_{cnt}}{2^{(h+1)/3}-1}$ to nodes with depth $i$ in a quadtree of height $h$, achieves maximum accuracy for range queries.*

According to Theorem 6, the privacy budget allocated to leaf regions is $\epsilon_h = \frac{2^{h/3}(\sqrt[3]{2}-1)\epsilon_{cnt}}{2^{(h+1)/3}-1}$. Observing that the size of a leaf region in a standard quadtree is $\frac{|V|^2}{4^h}$, we calculate $h$ by requiring the size of a leaf region to be greater than $\mu$ times the noise's standard deviation because we cannot get useful noisy counts on overly small regions, that is:

$$\frac{|V|^2}{4^h} \geq \frac{\mu\sqrt{2}GS(f)}{\epsilon_h} = \frac{\mu\sqrt{2}(2^{(h+1)/3}-1)GS(f)}{2^{h/3}(\sqrt[3]{2}-1)\epsilon_{cnt}}$$

where $f$ is a count query. For a count query over a region $R$ of $A$, *in the worst case*, the global sensitivity $GS(f) = 2$.[4] From the above equation, we get:

$$\sqrt[3]{2}(2^h)^2 - (2^h)^{5/3} \leq \frac{(\sqrt[3]{2}-1)|V|^2\epsilon_{cnt}}{\mu\sqrt{2}GS(f)} \quad (1)$$

Thus, our goal is to calculate the maximal $h$ value, $h_{\max}$, that satisfies Eq. 1. Theorem 7 implies that there is a unique solution for $h_{\max}$.

**Theorem 7** *$f(h) = \sqrt[3]{2}(2^h)^2 - (2^h)^{5/3}$ monotonically increases on $[0, +\infty)$.*

*Proof* Let $t = 2^h$. $\forall h \geq 0,\ t \geq 1$. Plugging $t$ to the equation, we get:

$$f'(t) = 2\sqrt[3]{2}t - \frac{5}{3}t^{\frac{2}{3}} \geq \frac{5}{3}t^{\frac{2}{3}}(t^{\frac{1}{3}} - 1) > 0$$

on $(1, +\infty)$. This completes the proof. □

Since the left-hand side (LHS) of Eq. 1 increases monotonically, we can always find $h_{\max}$ by attempting increasing $h$ values. Then, $h_{\max}$ is used as the height of the quadtree. In our experiments, setting $\mu = 5$ gives good estimates for different types of real-life datasets.

In addition to the major stop condition, we propose another two heuristic stop conditions to improve the efficiency and utility of our approach. First, if a region is dense enough (the density is calculated based on noisy counts), then there is no need to further partition it because we can already reconstruct its noisy version with high accuracy. In practice, we consider a region $R$ with $\text{den}(R) \geq 80\,\%$ to be dense (experiments show that there is no significant utility difference among the density thresholds in the range [75 %, 90 %]).

[4] For a region $R$ not containing elements on the diagonal, $GS(f) = 1$ because a single edge difference cannot change two elements in this region.

Second, we can stop partitioning a region $R$ if the number of elements with value 1 in $R$ is small enough. Note that the determination of a sparse region is based on its noisy count, *not* its density. Specifically, we set the threshold for determining a sparse region to be $80\,\% \times \frac{|V|^2}{4^h}$, the number of 1s needed to form *at least* one dense region. Any region with number of 1s $< 80\,\% \times \frac{|V|^2}{4^h}$ is not worth further partitioning as it will only lead to excessive noise. As a result, only regions that are neither dense nor sparse will be iteratively partitioned; otherwise, they are marked as leaf (Lines 13–14).

*Partitioning* For a non-leaf region $R$, we employ the exponential mechanism to find the best splitting point that divides $R$ into four subregions with the maximal *density contrast* among all possible splitting points (Line 9). Intuitively, such a split best distinguishes the dense and sparse subregions. Recall that a splitting point is composed of two coordinates, each from a dimension. For example, a possible splitting point in Fig. 2b is (3, 5), which means to split by the 3rd row and the 5th column. The corresponding split operation is illustrated by the two boldest lines in Fig. 2b.

For a non-leaf region $R$ of size $m \times l$, there could be *at most* $(m-1)(l-1)$ possible splitting points. We denote the set of all possible splitting points by $\mathcal{P}$. The utility function of selecting a splitting point $p \in \mathcal{P}$ over a region $R$ is designed to be

$$q(R, p) = \max_{\forall R' \in \mathcal{R}} (\text{den}(R')) - \min_{\forall R' \in \mathcal{R}} (\text{den}(R')),$$

where $\mathcal{R}$ is the set of subregions of $R$ resulted by $p$. Intuitively, this utility function finds the point that results in the maximal density contrast.

Since $q(R, p)$ is defined by density, to obtain a reasonably low sensitivity, we constrain the minimum size of a region in level $i$ (except the root and leaves) of $\mathcal{QT}$ to be $\frac{|V|^2}{4^{i+1}}$. This guarantees that, depending on the location of the region $R$ in $A$, $GS(q) = \frac{2 \cdot 4^{i+1}}{|V|^2}$ (if $R$ contains elements on the main diagonal) or $GS(q) = \frac{4^{i+1}}{|V|^2}$ (otherwise). Due to this constraint, we can apply the exponential mechanism on a smaller set of possible splitting points $\mathcal{P}$, because there is no need to consider the splitting points on level $i$ resulting in a subregion of size $< \frac{|V|^2}{4^{i+1}}$. Then, the exponential mechanism is used to select a splitting point $p_i$ on $R$ with the following probability,

$$\frac{\exp(\frac{\epsilon_{par}}{2hGS(q)}q(R, p_i))}{\sum_{p_j \in \mathcal{P}} \exp(\frac{\epsilon_{par}}{2hGS(q)}q(R, p_j))}$$

where $\frac{\epsilon_{par}}{h}$ is the privacy budget assigned to the mechanism, as explained below.

*Example 5* Consider the graph and its adjacency matrix in Fig. 2a, b. Suppose the height of $\mathcal{QT}$ is calculated to be 2. The

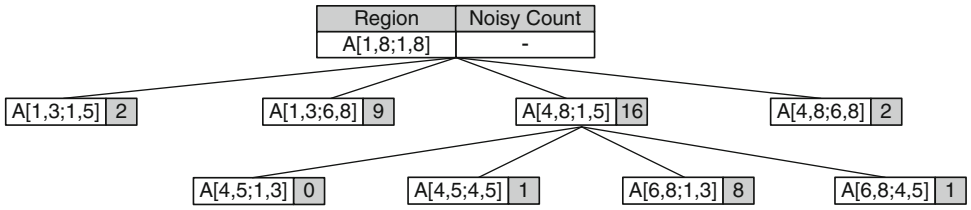



**Fig. 5** A sample quadtree structure for density-based partitioning

first possible partition operation is illustrated by the boldest lines, resulting in four subregions: $R_1 = A[1, 3; 1, 5]$, $R_2 = A[1, 3; 6, 8]$, $R_3 = A[4, 8; 1, 5]$ and $R_4 = A[4, 8; 6, 8]$. Assume that the noisy counts of $R_1$ and $R_4$ indicate that they are sparse and the noisy count of $R_2$ indicates that it is dense. *DER* only needs to further partition $R_3$. After that, the height has been reached and $\mathcal{QT}$ ends with seven leaf nodes. The corresponding quadtree is illustrated in Fig. 5.

*Privacy budget allocation* Broadly, the adjusted total privacy budget $\frac{\epsilon}{k}$ is divided into three portions: $\epsilon_I$, $\epsilon_E$ and $\epsilon_A$, each being used in a step. $\epsilon_E$ is further divided for two tasks, $\epsilon_{cnt}$ for calculating noisy counts of all (sub)regions and $\epsilon_{par}$ for selecting splitting points on all internal nodes of $\mathcal{QT}$.

The first problem is to determine the values of $\epsilon_I$, $\epsilon_{cnt}$, $\epsilon_{par}$ and $\epsilon_A$. In general, we assign larger budgets to $\epsilon_{cnt}$ and $\epsilon_A$ because: (1) *IdentifyVertexLabeling* does *not* directly incur any utility loss; (2) as shown later in Theorem 15, as long as we can obtain relatively accurate noisy counts, we can always find denser (or sparser) subregions, which can be reconstructed with better accuracy. Between $\epsilon_{cnt}$ and $\epsilon_A$, more budget is given to $\epsilon_{cnt}$ because a sufficiently dense (or sparse) leaf region can be recovered with reasonable accuracy regardless of the privacy budget (see Theorem 14). Since it is difficult to theoretically quantify the values, we experimentally choose proper portions for each of them, complying with the analysis above.

Once $\epsilon_I$, $\epsilon_{cnt}$, $\epsilon_{par}$ and $\epsilon_A$ are fixed, we employ the following allocation scheme to distribute them to each node of $\mathcal{QT}$ (Line 8). To obtain noisy counts, we employ an *adaptive* privacy budget allocation scheme based on the geometric budget scheme [9]. Initially, we assume that each root-to-leaf path in $\mathcal{QT}$ will be of the same length $h$ (e.g., $\mathcal{QT}$ is perfect) and assign $\frac{2^{i/3}(\sqrt[3]{2}-1)\epsilon_{cnt}}{2^{(h+1)/3}-1}$ to each node with depth $1 \le i < h$. Since an input dataset is always non-empty, there is no need to assign any budget to get the noisy count of the root. Hence we add the portion of the root level, $\frac{(\sqrt[3]{2}-1)\epsilon_{cnt}}{2^{(h+1)/3}-1}$, to the leaves, that is, a node with depth $h$ receives $\frac{(2^{h/3}+1)(\sqrt[3]{2}-1)\epsilon_{cnt}}{2^{(h+1)/3}-1}$. After that, we adaptively adjust privacy budgets during the partitioning process.

Due to the stop conditions, $\mathcal{QT}$ may not be perfect (i.e., some root-to-leaf paths may have a length $< h$) and, therefore, we want to reallocate the remaining privacy budget on these paths to fully make use of the total budget. For a leaf node $v$ whose depth $i < h$, let $\widetilde{c_1}$ be the noisy count obtained by privacy parameter $\epsilon_1 = \frac{2^{i/3}(\sqrt[3]{2}-1)\epsilon_{cnt}}{2^{(h+1)/3}-1}$ (the initial privacy parameter assigned to $v$). We can calculate another noisy count $\widetilde{c_2}$ of $v$ with $\epsilon_2 = \frac{(2^{h/3}+1)(\sqrt[3]{2}-1)\epsilon_{cnt}}{2^{(h+1)/3}-1}$ (the privacy budget initially reserved for level $h$). Obviously, $\widetilde{c_2}$ has a better accuracy than $\widetilde{c_1}$ because $Var(\widetilde{c_2}) < Var(\widetilde{c_1})$. We can replace $\widetilde{c_1}$ by $\widetilde{c_2}$ as a more precise estimate of the true count, but this simple strategy essentially wastes the privacy parameter used for generating $\widetilde{c_1}$. We propose a strategy that combines both $\widetilde{c_1}$ and $\widetilde{c_2}$ to calculate a more accurate estimate $\widetilde{c}$ than *both* $\widetilde{c_1}$ and $\widetilde{c_2}$ without extra privacy budget.

**Theorem 8** *Let* $\widetilde{c} = \frac{\epsilon_1^2}{\epsilon_1^2+(\gamma\epsilon_2)^2}\widetilde{c_1} + \frac{(\gamma\epsilon_2)^2}{\epsilon_1^2+(\gamma\epsilon_2)^2}\widetilde{c_2}$, *where* $\gamma = \frac{\epsilon_2}{\epsilon_1} = 2^{\frac{h-i}{3}}$. *Then,* $Var(\widetilde{c}) < Var(\widetilde{c_2}) < Var(\widetilde{c_1})$.

*Proof* Since $Var(\widetilde{c_1}) = \frac{2}{{\epsilon_1}^2}$ and $Var(\widetilde{c_2}) = \frac{2}{{\epsilon_2}^2}$,

$$Var(\widetilde{c}) = \frac{\epsilon_1^4 Var(\widetilde{c_1})}{(\epsilon_1^2+(\gamma\epsilon_2)^2)^2} + \frac{(\gamma\epsilon_2)^4 Var(\widetilde{c_2})}{(\epsilon_1^2+(\gamma\epsilon_2)^2)^2} = \frac{2}{\frac{(\epsilon_1^2+(\gamma\epsilon_2)^2)^2}{{\epsilon_1}^2+\gamma^4{\epsilon_2}^2}}$$

Hence we need to prove that

$$\frac{(\epsilon_1^2+(\gamma\epsilon_2)^2)^2}{{\epsilon_1}^2+\gamma^4{\epsilon_2}^2} > {\epsilon_2}^2 > {\epsilon_1}^2.$$

This is equivalent to prove that

$$\frac{\gamma^4{\epsilon_2}^2 + 2\gamma^2{\epsilon_1}^2 + \frac{{\epsilon_1}^4}{{\epsilon_2}^2}}{\gamma^4{\epsilon_2}^2 + {\epsilon_1}^2} > 1.$$

Since $\gamma = 2^{\frac{h-i}{3}} > 1$, we have $2\gamma^2{\epsilon_2}^2 + \frac{{\epsilon_1}^4}{{\epsilon_2}^2} > {\epsilon_1}^2$. Therefore, $Var(\widetilde{c}) < \frac{2}{{\epsilon_2}^2} = Var(\widetilde{c_2})$. Since $\epsilon_1 < \epsilon_2$, we get

$$Var(\widetilde{c}) < Var\,(\widetilde{c_2}) < Var\,(\widetilde{c_1})\,.$$

□

For a leaf node $v$ whose depth $i < h - 1$, the portion of privacy budget left from partitioning (that is the sum of the privacy parameters initially assigned to levels $i + 1, i + 2, \ldots, h - 1$), $\frac{(2^{h/3}-2^{(i+1)/3})\epsilon_{cnt}}{2^{(h+1)/3}-1}$, is added to $\epsilon_A$ so that we can make full use of the privacy budget.

For selecting splitting points by the exponential mechanism, we use a uniform budget allocation scheme that equally distributes $\frac{\epsilon_{par}}{h}$ to each internal node in $\mathcal{QT}$. For reconstructing leaf regions, each leaf node in $\mathcal{QT}$ receives $\epsilon_A$ plus the privacy budget left from partitioning.

*Efficient implementation* In order to apply the exponential mechanism, for *every* internal node of $\mathcal{QT}$, we need to compute the densities of the four subregions resulted from *every* possible splitting point. A naive implementation takes run-time $O(|V|^4)$ to calculate the densities for all possible splitting points for all nodes on the same level of $\mathcal{QT}$. We propose a data structure, called *count summary matrix*, which improves the run-time complexity of calculating all densities for a level of $\mathcal{QT}$ from $O(|V|^4)$ to $O(|V|^2)$.

**Definition 5** (*Count summary matrix*) Given an adjacency matrix $A$ of a simple graph $G = (V, E)$, the count summary matrix $C$ of $A$ is a $|V| \times |V|$ matrix, where $\forall 1 \leq i, j \leq |V|$, $C[i, j]$ equals the number of 1s in the region $A[1, i; 1, j]$, that is, $C[i, j] = \sum_{m=1}^{i} \sum_{l=1}^{j} A_{ml}$.

A count summary matrix $C$ can be constructed with run-time complexity $O(|V|^2)$ based on the following theorem.

**Theorem 9** *$C[i, j] = C[i-1, j] + C[i, j-1] - C[i-1, j-1] + A_{ij}$, where $C[i, j] = 0$ if $i < 1$ or $j < 1$.*

*Proof* By the definition of a count summary matrix, we have the following:

$$
\begin{aligned}
&C[i-1, j] + C[i, j-1] - C[i-1, j-1] + A_{ij} \\
&= \sum_{m=1}^{i-1} \sum_{l=1}^{j} A_{ml} + \sum_{m=1}^{i} \sum_{l=1}^{j-1} A_{ml} - \sum_{m=1}^{i-1} \sum_{l=1}^{j-1} A_{ml} + A_{ij} \\
&= \sum_{m=1}^{i-1} A_{mj} + \sum_{m=1}^{i} \sum_{l=1}^{j-1} A_{ml} + A_{ij} \\
&= \sum_{m=1}^{i} A_{mj} + \sum_{m=1}^{i} \sum_{l=1}^{j-1} A_{ml} \\
&= \sum_{m=1}^{i} \sum_{l=1}^{j} A_{ml} \\
&= C[i, j]
\end{aligned}
$$

This establishes the theorem. □

Note that the count summary matrix $C$ just needs to be computed *once* for the entire sanitization process. Once $C$ is constructed, the density of any region can be computed by Theorem 10 in $O(1)$.

**Theorem 10** *The density of a region $A[k, l; m, n]$ is*

$$\frac{C[l, n] - C[l, m-1] - C[k-1, n] + C[k-1, m-1]}{(n - m + 1)(l - k + 1)}.$$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 |
| 2 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 6 |
| 3 | 0 | 0 | 0 | 0 | 1 | 3 | 6 | 9 |
| 4 | 0 | 0 | 0 | 0 | 2 | 4 | 7 | 10 |
| 5 | 0 | 0 | 1 | 2 | 4 | 6 | 9 | 12 |
| 6 | 1 | 2 | 3 | 4 | 6 | 8 | 11 | 14 |
| 7 | 2 | 4 | 6 | 7 | 9 | 11 | 14 | 17 |
| 8 | 3 | 6 | 9 | 10 | 12 | 14 | 17 | 20 |

**Fig. 6** The count summary matrix of the adjacency matrix in Fig. 2b

*Proof* Similarly, from the definition of a count summary matrix, we have

$$
\begin{aligned}
&C[l, n] - C[l, m-1] - C[k-1, n] + C[k-1, m-1] \\
&= \sum_{p=1}^{l} \sum_{q=1}^{n} A_{pq} - \sum_{p=1}^{l} \sum_{q=1}^{m-1} A_{pq} - \sum_{p=1}^{k-1} \sum_{q=1}^{n} A_{pq} + \sum_{p=1}^{k-1} \sum_{q=1}^{m-1} A_{pq} \\
&= \sum_{p=1}^{l} \sum_{q=m}^{n} A_{pq} - \sum_{p=1}^{k-1} \sum_{q=m}^{n} A_{pq} \\
&= \sum_{p=k}^{l} \sum_{q=m}^{n} A_{pq}
\end{aligned}
$$

Since $den(A[k, l; m, n]) = \frac{\sum_{p=k}^{l} \sum_{q=m}^{n} A_{pq}}{(n-m+1)(l-k+1)}$, this completes the proof. □

*Example 6* The count summary matrix of the adjacency matrix in Fig. 2b is illustrated in Fig. 6. The density of the region $den(A[4, 6; 4, 7]) = (C[6, 7] - C[6, 3] - C[3, 7] + C[3, 3])/12 = (11 - 3 - 6 + 0)/12 = 1/6$.

In addition, when the input dataset is extremely large, sampling (i.e., checking the splitting points with a step larger than 1) could be used at the cost of slightly worse utility. We experimentally study the effect of sampling on data utility and scalability in Sect. 6.

*Run-time complexity* The run-time complexity of *ExploreDenseRegion* is given in Theorem 11.

**Theorem 11** *The run-time complexity of Procedure 2 is $O(|V|^2)$.*

*Proof* The complexity of Procedure 2 is dominated by the application of the exponential mechanism to select the splitting points. Suppose the size of a node $v_i$ in level $j$ of $\mathcal{QT}$ is $m_i \times l_i$. A single application of the exponential mechanism needs to consider at most $(m_i - 1)(l_i - 1)$ possible splitting positions. Due to the count summary matrix, each position can be checked in constant time. Since $m_i l_i >$

$(m_i - 1)(l_i - 1)$ is the area of the region represented by $v_i$, we have $\sum_{v_i \in \text{level}(j,\mathcal{QT})} m_i l_i \leq |V|^2$ because the sum of the areas represented by all nodes on level $j$ cannot be greater than the total area $|V|^2$. So the complexity of building level $j$ is $O(|V|^2)$. Therefore, the total complexity of building $\mathcal{QT}$ of height $h$ must be bounded by $O(h|V|^2)$. Since $h \ll |V|^2$, the run-time complexity of Procedure 2 can be further considered as $O(|V|^2)$.

Finally, in the exploration process, we can conduct a simple post-processing step by rounding the noisy count $\widetilde{c}$ of a region $R$ with size $m \times l$ into the range of $[0, ml]$ because the number of 1s in a region cannot be larger than its size.

### 5.4 Edge arrangement

In this section, we denote an original region in $A$ by $R$ and its reconstructed counterpart in $\widetilde{A}$ by $\widetilde{R}$. Since our utility requirement is to build a differentially private $\widetilde{A}$ such that $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |A_{ij} - \widetilde{A}_{ij}|$ is minimized, it naturally requires to reconstruct each leaf region $\widetilde{R}$ of size $m \times l$ with $\sum_{i=1}^{m} \sum_{j=1}^{l} |R_{ij} - \widetilde{R}_{ij}|$ minimized.

A simple method to reconstruct $R$ is to randomly place 1s in $\widetilde{R}$. Unfortunately, our experiments suggest that the performance of this scheme is highly sensitive to the quality of vertex labeling (see Sect. 6). In addressing this drawback, given a leaf region $\widetilde{R}$ of size $m \times l$ with a noisy count $\widetilde{c} \leq ml$, we design an exponential mechanism to select an edge arrangement $r$ by the following utility function:

$$q(\widetilde{R}, r) = ml - \sum_{i=1}^{m} \sum_{j=1}^{l} |R_{ij} - \widetilde{R}_{ij}|. \tag{2}$$

Intuitively, the utility function measures how many elements of $\widetilde{R}$ are correctly assigned with respect to $R$. The global sensitivity of $q(R, r)$ is $GS(q) = 2$ or $GS(q) = 1$, depending on the location of $R$ in $A$.

However, there is a major technical challenge: A naive implementation of the exponential mechanism needs to explicitly consider a total of $\binom{ml}{\widetilde{c}}$ possible arrangements, which is of *factorial complexity*. Instead, we propose an efficient implementation, which takes run-time complexity of only $O(ml)$ for assigning edges in a single leaf region. We first implicitly group all arrangements with the same score into a group. At first glance, for any region with size $m \times l$, there can be *at most* $ml + 1$ groups because there are at most $ml + 1$ possible score values (from 0 to $ml$ as defined in Eq. 2). Now we show that the actual number of groups to consider is $\leq \lceil \frac{ml+1}{2} \rceil$ by giving the sufficient and necessary condition of a possible score below.

**Theorem 12** *For a leaf region $\widetilde{R}$ of size $m \times l$ with a noisy count $\widetilde{c}$ and the true count $c$, a score $s$ is possible if and only if $s \in [\max\{\widetilde{c}+c-ml, ml-c-\widetilde{c}\}, \min\{ml+c-\widetilde{c}, ml+\widetilde{c}-c\}]$ and $\frac{s+c+\widetilde{c}-ml}{2}$ is an integer. The total number of possible scores is less than or equal to*

$$\left\lceil \frac{\min\{ml + c - \widetilde{c}, ml + \widetilde{c} - c\} - \max\{\widetilde{c} + c - ml, ml - c - \widetilde{c}\}}{2} \right\rceil$$

$$\leq \tfrac{ml+1}{2}.$$

*Proof* We first calculate the lower and upper bounds of a possible score by considering all possible cases. (1) $\widetilde{c} \geq c$ and $\widetilde{c} \leq ml - c$: the maximum score is achieved when $c$ 1s are assigned to the elements where $R_{ij} = 1$ and the rest $\widetilde{c} - c$ 1s are assigned to the elements with $R_{ij} = 0$, which gives the score $ml + c - \widetilde{c}$; the minimal score is achieved when $\widetilde{c}$ 1s are assigned to the elements with $R_{ij} = 0$, which gives the score $ml - c - \widetilde{c}$. (2) $\widetilde{c} \geq c$ and $\widetilde{c} > ml - c$: Similarly, the maximum score is $ml + c - \widetilde{c}$, while the minimum is $\widetilde{c} + c - ml$. (3) $\widetilde{c} < c$ and $\widetilde{c} \leq ml - c$: The maximum is $ml + \widetilde{c} - c$, while the minimum is $ml - c - \widetilde{c}$. (4) $\widetilde{c} < c$ and $\widetilde{c} > ml - c$: The maximum is $ml + \widetilde{c} - c$, and the minimum is $\widetilde{c} - (ml - c)$. Combining these four cases, we get the bounds of $s$.

Next, we prove that $\frac{s+c+\widetilde{c}-ml}{2}$ must be an integer in order to make $s$ possible. Consider the allocation of $\widetilde{c}$ 1s in $\widetilde{R}$. Suppose the numbers of elements where $R_{ij} = 0 \wedge \widetilde{R}_{ij} = 0$, $R_{ij} = 0 \wedge \widetilde{R}_{ij} = 1$, $R_{ij} = 1 \wedge \widetilde{R}_{ij} = 0$, $R_{ij} = 1 \wedge \widetilde{R}_{ij} = 1$, are, respectively, $x$, $y$, $z$ and $w$. For an arrangement with a score $s$, we have:

$$\begin{aligned} x + y + z + w &= ml \\ x + w &= s \\ y + w &= \widetilde{c} \\ z + w &= c \end{aligned}$$

Solving these equations, we get $w = \frac{s+c+\widetilde{c}-ml}{2}$. Apparently, only if $x$, $y$, $z$ and $w$ are non-negative integers, $s$ is possible. Since $s \in [\max\{\widetilde{c} + c - ml, ml - c - \widetilde{c}\}, \min\{ml + c - \widetilde{c}, ml + \widetilde{c} - c\}]$, $x$, $y$, $z$ and $w$ must be non-negative. So we just need to require $\frac{s+c+\widetilde{c}-ml}{2}$ to be an integer, which consequently guarantees that $x$, $y$, $z$ are also integers. Finally, since $\frac{s+c+\widetilde{c}-ml}{2}$ has to be an integer, all possible scores have to be either all even or all odd. We complete the proof. □

We call a group of arrangements with a possible score a *valid* group. We can calculate the size of each valid group by Theorem 13.

**Theorem 13** *Given a leaf region $\widetilde{R}$ of size $m \times l$ with a noisy count $\widetilde{c}$ and the true count $c$, the size of a valid group $G_s$ with score $s$ is*

$$|G_s| = \binom{c}{\frac{s+c+\widetilde{c}-ml}{2}} \binom{ml - c}{\frac{ml+\widetilde{c}-s-c}{2}},$$

*where $\binom{0}{0}$ is defined to be 1.*

*Proof* Following the proof of Theorem 12, we have $w = \frac{s+c+\widetilde{c}-ml}{2}$, which means that we need to assign $\frac{s+c+\widetilde{c}-ml}{2}$ 1s to the elements where $R_{ij} = 1$ and $\widetilde{c} - \frac{s+c+\widetilde{c}-ml}{2}$ 1s to the elements where $R_{ij} = 0$. For the former case, there are a total of $\binom{c}{\frac{s+c+\widetilde{c}-ml}{2}}$ possible combinations; for the latter case, there are a total of $\binom{ml-c}{\frac{ml+\widetilde{c}-s-c}{2}}$ possible combinations. Therefore, combining these two cases, we get $|G_s| = \binom{c}{\frac{s+c+\widetilde{c}-ml}{2}}\binom{ml-c}{\frac{ml+\widetilde{c}-s-c}{2}}$. □

Then, the exponential mechanism can be used to select a group $G_i$ with the following probability,

$$\frac{\exp\left(\frac{i\bar{\epsilon}}{2GS(q)}\right) \times |G_i|}{\sum_{j=0}^{ml}\left(\exp(\frac{j\bar{\epsilon}}{2GS(q)}) \times |G_j|\right)},$$

where $\bar{\epsilon}$ equals $\epsilon_A$ plus the privacy budget left from the exploration process, $GS(q) = 2$ or $GS(q) = 1$, and the size of an *invalid* group is 0.

Finally, conditional on that the group $G_i$ is selected, we can uniformly generate a random arrangement within $G_i$ by randomly assigning $\frac{i+c+\widetilde{c}-ml}{2}$ 1s to the elements $R_{ij}$ with $R_{ij} = 1$ and $\frac{ml+\widetilde{c}-i-c}{2}$ 1s to the elements $R_{ij}$ with $R_{ij} = 0$. Obviously, generating such an arrangement could be done with run-time complexity $O(1)$. In particular, if a generated arrangement makes $\widetilde{A}_{ii} = 1$ for any $1 \le i \le |V|$, an alternative arrangement could be generated because a graphic matrix contains a zero diagonal.

We give the utility guarantee of our edge arrangement method below.

**Theorem 14** *Given a leaf region $\widetilde{R}$ of size $m \times l$ with a noisy count $\widetilde{c}$ and the true count $c$, with probability $1-\beta$,*

$$\forall \widetilde{c} < c,$$
$$q(\widetilde{R}, r^*) \ge \max\{\widetilde{c} + c - ml, ml - c - \widetilde{c},$$
$$ml - c + \widetilde{c} - \frac{2GS(q)}{\bar{\epsilon}}(log\binom{ml}{\widetilde{c}} - log\binom{c}{\widetilde{c}} - ln\beta)\}$$

*and*

$$\forall \widetilde{c} \ge c,$$
$$q(\widetilde{R}, r^*) \ge \max\{\widetilde{c} + c - ml, ml - c - \widetilde{c},$$
$$ml + c - \widetilde{c} - \frac{2GS(q)}{\bar{\epsilon}}(log\binom{ml}{c} - log\binom{\widetilde{c}}{c} - ln\beta)\}$$

*where $r^*$ is the arrangement selected by our approach.*

*Proof* Let $\mathrm{OPT}_q(\widetilde{R}) = \max_{r\in\mathcal{R}} q(\widetilde{R}, r)$, $\mathcal{R}_{\mathrm{OPT}} = \{r \in \mathcal{R} : q(\widetilde{R}, r) = \mathrm{OPT}_q(\widetilde{R})\}$ and $r^* = \mathrm{Exponential}(\widetilde{R}, \mathcal{R}, q, \bar{\epsilon})$. In [15,29], it has been proven that

$$Pr\left[q(\widetilde{R}, r^*) \le \mathrm{OPT}_q(\widetilde{R}) - \frac{2GS(q)}{\bar{\epsilon}}(\log\frac{|\mathcal{R}|}{|\mathcal{R}_{\mathrm{OPT}}|} + t)\right] \le e^{-t}$$

when $\widetilde{c} < c$, $\mathrm{OPT}_q(\widetilde{R})$ is achieved when all $\widetilde{c}$ 1s are assigned to the elements with $R_{ij} = 1$, and $\mathrm{OPT}_q(\widetilde{R}) = ml - c + \widetilde{c}$. The total number of possible arrangements is $\binom{ml}{\widetilde{c}}$, and the number of arrangements achieving $\mathrm{OPT}_q(\widetilde{R})$ is $\binom{c}{\widetilde{c}}$. Therefore, setting $t = \ln(1/\beta)$, we obtain

$$q(\widetilde{R}, r^*) \ge ml - c + \widetilde{c} - \frac{2GS(q)}{\bar{\epsilon}}(\log\binom{ml}{\widetilde{c}} - \log\binom{c}{\widetilde{c}} - \ln\beta).$$

Combining the lower bound of a score given in Theorem 12, we get the lower bound of $q(\widetilde{R}, r^*)$.

When $\widetilde{c} \ge c$, $\mathrm{OPT}_q(\widetilde{R})$ is achieved when all the elements with $R_{ij} = 1$ are assigned 1s and the rest $\widetilde{c} - c$ 1s are assigned to the elements with $R_{ij} = 0$. We get $\mathrm{OPT}_q(\widetilde{R}) = c + (ml - c) - (\widetilde{c} - c) = ml + c - \widetilde{c}$. The number of arrangements achieving $\mathrm{OPT}_q(\widetilde{R})$ is $\binom{ml-c}{\widetilde{c}-c}$. Hence we have, with probability $1-\beta$,

$$\begin{aligned} q(\widetilde{R}, r^*) &\ge ml + c - \widetilde{c} - \frac{2GS(q)}{\bar{\epsilon}}\left(\log\frac{\binom{ml}{\widetilde{c}}}{\binom{ml-c}{\widetilde{c}-c}} - \ln\beta\right) \\ &= ml + c - \widetilde{c} - \frac{2GS(q)}{\bar{\epsilon}}\left(\log\frac{\binom{ml}{\widetilde{c}}}{\frac{\binom{ml}{\widetilde{c}}\binom{\widetilde{c}}{c}}{\binom{ml}{c}}} - \ln\beta\right) \\ &= ml + c - \widetilde{c} - \frac{2GS(q)}{\bar{\epsilon}}(\log\binom{ml}{c} - \log\binom{\widetilde{c}}{c} \\ &\quad -\ln\beta) \end{aligned}$$

Similarly, the lower bound in Theorem 12 also applies. This completes the proof. □

Specifically, when $\widetilde{c} = c$, we have

$$q(\widetilde{R}, r^*) \ge \max\left\{2c - ml, ml - 2c, ml - \frac{2GS(q)}{\bar{\epsilon}}(\log\binom{ml}{c} - ln\beta)\right\}.$$

We can observe that when $c$ is either relatively large or relatively small with respect to $ml$ (that is, either $den(R)$ is large enough or small enough), the reconstructed $\widetilde{R}$ could be very close to $R$. The worst utility occurs when $c = \frac{ml}{2}$. However, this case can always be avoided by further partitioning, as confirmed by Theorem 15.

**Theorem 15** *Given a region $R$, any partitioning of $R$ results in subregions $R'$ satisfying either $den(R') \le den(R)$ or $den(R') \ge den(R)$, with equality attained if and only if 1s are uniformly distributed in $R$.*

The proof is straightforward and is therefore omitted here. According to the *power law distribution* [12], $R$ is very unlikely to have a uniform distribution. Therefore, Theorem 15 suggests that keeping partitioning a region leads to a more precise reconstruction. This observation is based on the assumption that $\widetilde{c}$ is accurate. However, when subregions become smaller, the accuracy of $\widetilde{c}$ decreases, which causes extra utility lost in edge assignment. Therefore, it justifies our design of the stop condition that takes into consideration both the accuracy of noisy counts and the size of a leaf region.

After reconstructing each leaf region, we perform a simple step to make $\widetilde{A}$ *graphic*: $\forall 1 \leq i \leq j \leq |V|$, set $\widetilde{A}_{ji} = \widetilde{A}_{ij}$. We can see that the complexity of reconstructing all leaf regions is $O(|V|^2)$ because $\sum_i \frac{m_i l_i + 1}{2} < |V|^2$. Therefore, the total complexity of *DER* is $O(|V|^2)$.

### 5.5 Privacy analysis

In this section, we prove that Algorithm 1 satisfies $\epsilon$-differential privacy over correlated network data.

**Theorem 16** DER *is $\epsilon$-differentially private over network databases with a correlation parameter $k$.*

*Proof* (Sketch) By definition, any subset of a dataset with a correlation parameter $k$ can be of a correlation parameter at most $k$. Then, according to Theorem 4, proving this theorem is equivalent to proving that *DER* satisfies $\frac{\epsilon}{k}$-differential privacy over non-correlated network databases.

Recall that *DER* is composed of three procedures. *IdentifyVertexLabeling* iteratively permutes pairs of vertices based on the noisy centralities. In each iteration, we employ the Laplace mechanism to obtain the noisy centralities with privacy budget $\frac{\epsilon_I}{t}$. By Theorem 5, each iteration preserves $\frac{\epsilon_I}{t}$-differential privacy. Since there are $t$ iterations, *IdentifyVertexLabeling* is $\epsilon_I$-differentially private due to the *sequential composition* property (Theorem 2).

For *ExploreDenseRegion*, all regions in the same level of $\mathcal{QT}$ are *disjoint* to each other, and hence the *parallel composition* property (Theorem 3) applies. That is, the privacy budget used in each root-to-leaf path of $\mathcal{QT}$ is independent of each other. We focus on analyzing the privacy budget used for each path. For any path, in each level (except the leaf level), we perform two differentially private operations: using the Laplace mechanism to calculate noisy counts and employing the exponential mechanism to select a splitting point; in the leaf level, only the Laplace mechanism is applied. The privacy budget within a path follows the sequential composition property. Under our adaptive privacy budget allocation scheme, the privacy budget used in a single path is *at most*

$$\sum_{i=0}^{h} \frac{2^{i/3}(\sqrt[3]{2}-1)\epsilon_{cnt}}{2^{(h+1)/3}-1} + \sum_{i=0}^{h-1} \frac{\epsilon_{par}}{h} = \epsilon_E.$$

Thus, *ExploreDenseRegion* guarantees $\epsilon_E$-differential privacy.

For *ArrangeEdge*, it selects an edge arrangement to recover a leaf region. The algorithm first uses the exponential mechanism to select a group of arrangements. It guarantees $\epsilon_A$-differential privacy. For a selected group, we then randomly select an arrangement within it. Randomly selecting an arrangement is 0-differentially private because this is a special application of the exponential mechanism with privacy budget 0. Hence *ArrangeEdge* enjoys $\epsilon_A$-differential privacy.

**Table 1** Experimental dataset statistics

| Datasets | $\|V\|$ | $\|E\|$ | Edge density |
|---|---|---|---|
| ca-GrQc | 5,242 | 14,484 | 0.00106 |
| ca-HepTh | 5,000 | 17,120 | 0.00137 |
| wiki-Vote | 7,115 | 100,762 | 0.00398 |
| STM | 1,012 | 7,860 | 0.01536 |

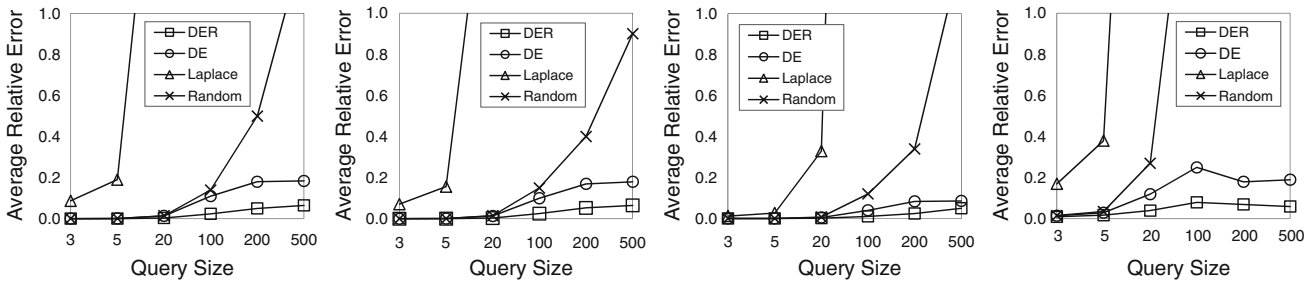
Applying the sequential composition property again, we can conclude that *DER* is $\frac{\epsilon}{k}$-differentially private because $\frac{\epsilon}{k} = \epsilon_I + \epsilon_E + \epsilon_A$. This establishes the proof. □

## 6 Experimental evaluation

In this section, we experimentally evaluate the performance of our sanitization algorithm (*DER*) in terms of both data utility and efficiency. As a reference point, we compare the utility of *DER* with a random graph of the same numbers of nodes and edges [6,17] (referred to as *Random*) and a sanitized graph generated by a simple Laplace mechanism-based approach proposed in [16] (referred to as *Laplace*). In addition, we consider a variant of *DER* in which we replace the *ArrangeEdge* procedure by randomly placing edges in each leaf region based on its noisy count. We refer to this variant as *DE*. In *DE*, the privacy budget for *ArrangeEdge* is allocated to the rest procedures. In all figures, the results reported are the average of 10 runs. Our implementation was done in C++, and all experiments were performed on an Intel Core 2 Quad 2.40 GHz PC with 16 GB RAM.

Four real-life datasets from three different types of networks are used in our experiments.[5] *ca-GrQc* is a subset of the collaboration network of Arxiv general relativity category. Two authors are connected if they coauthored at least one paper. *ca-HepTh* is extracted from the collaboration network of Arxiv high energy physics theory category. Similarly, there is an edge if two authors coauthored at least one paper. The *wiki-Vote* dataset contains social network information about Wikipedia voting on promotion to administratorship. An edge is created between two persons if one voted on or was voted by the other. *STM* provides the transportation network information of the Montreal transportation system. Two stations are considered connected if there are more than 500 passengers commuting between them within one week. The detailed characteristics of the datasets are summarized in Table 1. All loops are removed from the datasets.

[5] *ca-GrQc*, *ca-HepTh* and *wiki-Vote* are publicly available in the Stanford large network dataset collection (http://snap.stanford.edu/data/index.html). *STM* is provided by the Société de transport de Montréal (http://www.stm.info).




**(a)** ca-GrQc **(b)** ca-HepTh **(c)** wiki-Vote **(d)** STM

**Fig. 7** Average relative error versus query size

## 6.1 Data utility

We examine the utility of sanitized network data for three common data analysis tasks introduced in Sect. 3.4, namely *cut query*, *degree distribution* and *shortest path length*.

*Cut query* In the first set of experiments, we examine the utility for cut queries in terms of average relative error. We examine different query sizes (i.e., the number of vertices in a cut query) and report the results of 11 representative query sets with sizes spanning over the full spectrum in Fig. 7 and Table 2. Each query set consists of queries with sizes that are uniformly randomly distributed between 1 and the specified maximal size. For each query set, we randomly generate 20,000 queries. The sanity bound is set to 0.1 % of $|E|$, the same as [5,35].

Figure 7 presents the average relative errors of cut queries of relatively small query sizes while fixing $\epsilon = 1.0$ and $k = 1$. Six query sets (with maximal query sizes 3, 5, 20, 100, 200 and 500, respectively) are used to represent the general trends of the four approaches. As one can observe, the average relative errors of *DER* are consistently small under all query sizes. It is worth mentioning that the relative errors of *DER* do not monotonically increase with the increase of query sizes. *DE* also achieves small average relative errors on all datasets. This suggests that exploring dense regions is critical for cut queries. However, though not directly visible from Fig. 7, *DE* is more sensitive to different vertex labelings. Moreover, later we will see that, without the *ArrangEdge* procedure, *DE* cannot obtain desirable utility for degree distribution and shortest path length.

It is surprising to see that *Laplace* performs much worse than *Random*. This is because Laplace noise generated under a small privacy budget can easily make the original value of an element (either 0 or 1) indistinguishable. With the increase in query sizes, both *Laplace* and *Random* provide very poor utility. Another interesting observation is that the utility of *Random* is subtly related to the edge density: Its performance deteriorates quickly with the increase in edge density.

Table 2 inspects the performance of *DER* under large query sizes with $\epsilon = 1.0$ and $k = 1$, where the query sets have the maximal sizes $0.2 \cdot |V|$, $0.4 \cdot |V|$, $0.6 \cdot |V|$, $0.8 \cdot |V|$ and $|V|$, respectively. We can observe that *DER* also performs stably well under all large query sizes.

**Table 2** Average relative error of large query sizes

| Datasets | $0.2\|V\|$ | $0.4\|V\|$ | $0.6\|V\|$ | $0.8\|V\|$ | $\|V\|$ |
|---|---|---|---|---|---|
| ca-GrQc | 0.056 | 0.064 | 0.072 | 0.062 | 0.075 |
| ca-HepTh | 0.056 | 0.055 | 0.054 | 0.059 | 0.068 |
| wiki-Vote | 0.059 | 0.047 | 0.055 | 0.058 | 0.084 |
| STM | 0.084 | 0.07 | 0.057 | 0.053 | 0.033 |

In Fig. 8, we present relative errors of *DER*, *DE* and *Random* under varying privacy budgets from 0.6 to 1.0 while fixing the maximal query size to be $0.4 \cdot |V|$ and $k = 1$ (*Laplace*'s relative errors are too large to fit into the figures). As expected, the relative error increases when the privacy budget decreases. Nevertheless, *DER* achieves relative errors less than 13 % on all datasets even when $\epsilon = 0.6$.

Next we study how average relative errors vary under different correlation parameters while fixing $\epsilon = 1.0$ and the maximal query size to $0.4 \cdot |V|$ in Fig. 9. In general, the relative error increases with the increment of $k$ because larger noise has to be injected to hide stronger correlation. We can observe that *DER* can still provide some useful information even when $k$ is relatively large. In practice, many types of networks (e.g., transportation networks) have relatively small correlation, and thus, our approach can provide meaningful data utility without sacrificing privacy.

In Sect. 5.3, we point out that sampling (i.e., checking the splitting points using a step greater than 1) could be an effective means to speed up our approach at slight cost of data utility. In Fig. 10, we experimentally study the utility loss due to sampling while fixing $k = 1$ and the maximal query size to $0.4 \cdot |V|$. We show the average relative errors of different sampling steps (1, 2, 4 and 6). Generally, the increase in average relative error due to sampling is relatively small for all datasets under different privacy budgets. We can observe two interesting trends in Fig. 10. First, the influence of sampling is smaller when the privacy budget is larger. Second, the influence of sampling is smaller when the underlying network dataset is sparser. These two

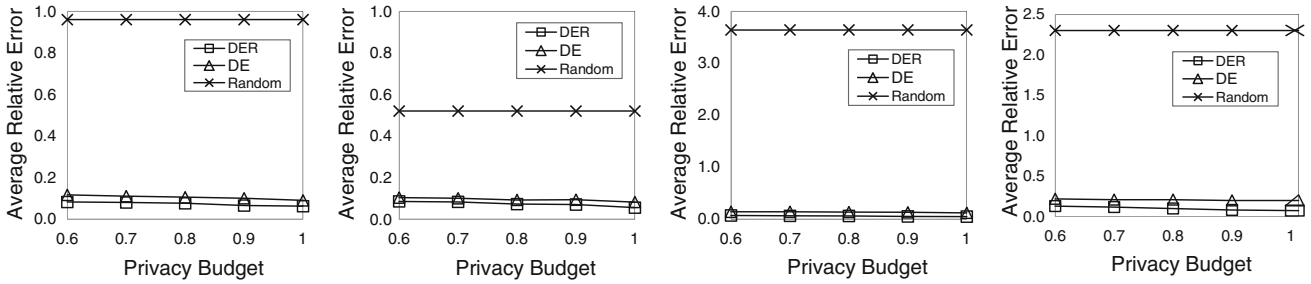



**Fig. 8** Average relative error versus privacy budget $\epsilon$

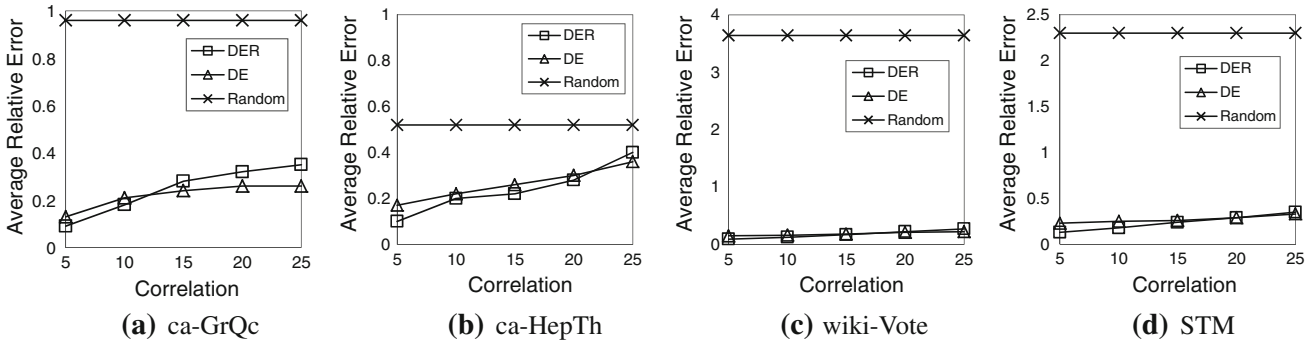



**Fig. 9** Average relative error versus correlation parameter $k$

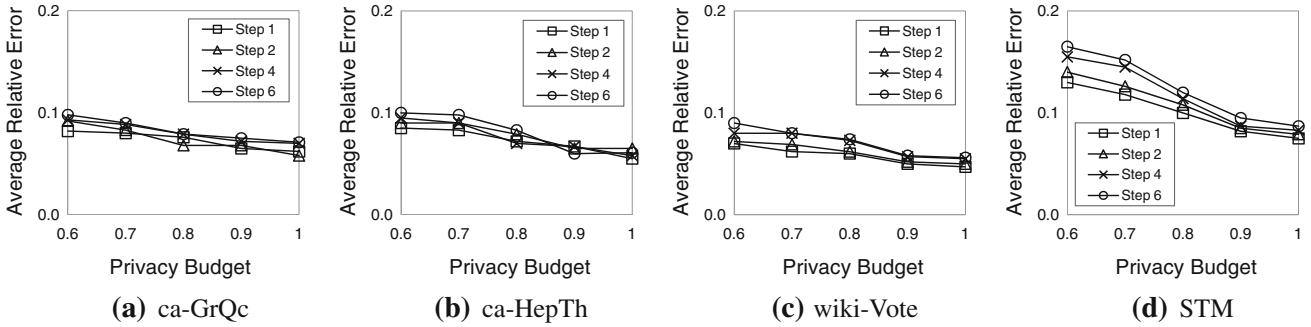



**Fig. 10** Average relative error versus sampling step

trends could provide a data publisher useful guidance for selecting a reasonable sampling step to trade for better efficiency.

*Degree distribution* In the second set of experiments, we demonstrate the utility of sanitized data for degree distribution, measured by KL-divergence. Figure 11 presents the KL-divergences for all datasets under different privacy budgets with $k = 1$. Due to the large KL-divergence resulted from *Laplace*,[6] it is excluded from the figures for better visibility. We can observe that our approach is suitable for preserving degree distributions. The KL-divergences of *DER* are small in all settings. Without the *ArrangeEdge* procedure, *DE* is not able to accurately reconstruct the leaf regions and, therefore, leads to less accurate degree distributions.

Figure 12 examines the KL-divergence for varying correlation parameters with $\epsilon = 1.0$. Though the KL-divergence of *DER* grows quickly with the increase in correlation, it is still able to preserve the general degree distributions on some datasets (e.g., *ca-GrQC* and *ca-HepTh*) even when $k = 25$.

In Fig. 13, we present the KL-divergence of the four datasets under different sampling steps while fixing $\epsilon = 1.0$ and $k = 1$. We can observe again that the impact of sampling on degree distribution is low. Moreover, similar trends to those of count queries can also be observed. Given a not too small privacy budget, if the underlying dataset is sparse, it is usually beneficial to employ the sampling technique.

[6] *Laplace* can barely provide any useful information in terms of degree distribution because its KL-divergence is almost the same as an empty graph (i.e., $|E| = 0$).

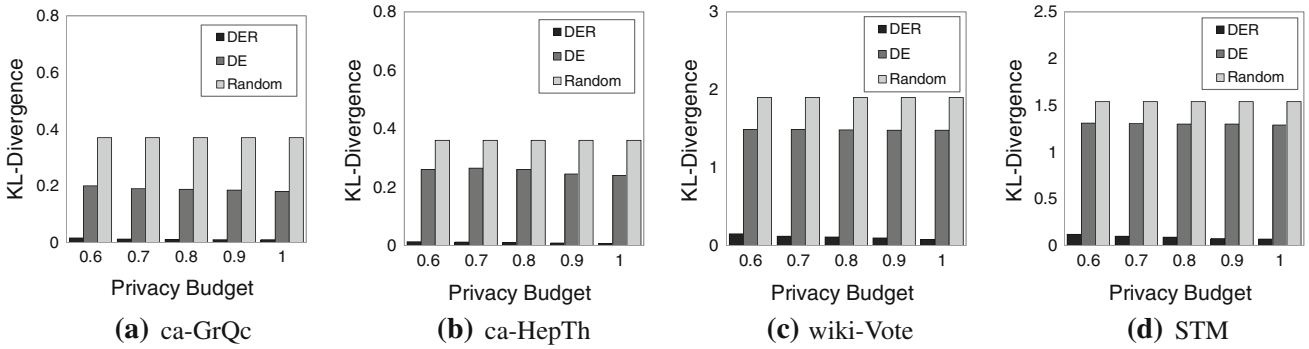



**Fig. 11** Degree distribution versus privacy budget $\epsilon$

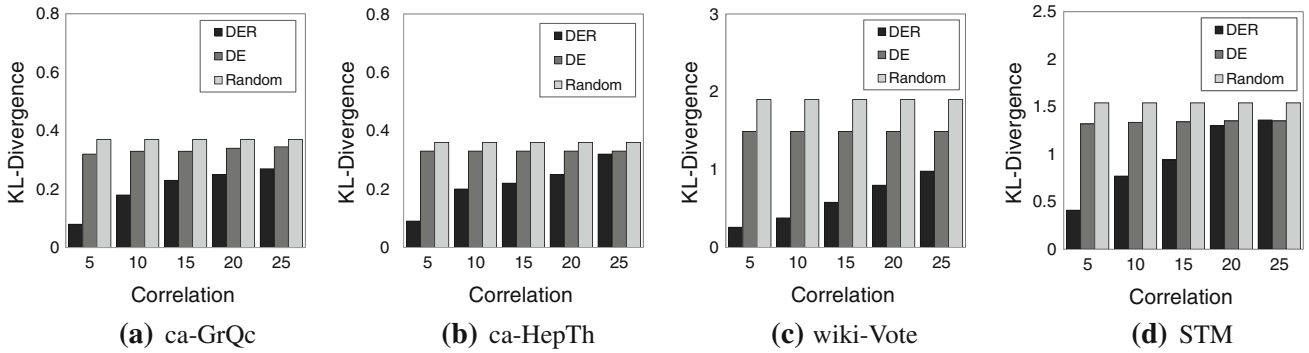



**Fig. 12** Degree distribution versus correlation parameter $k$

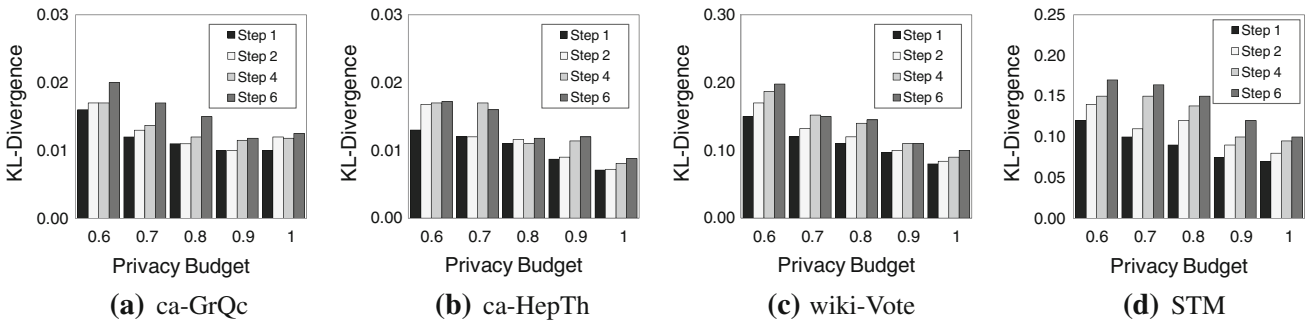
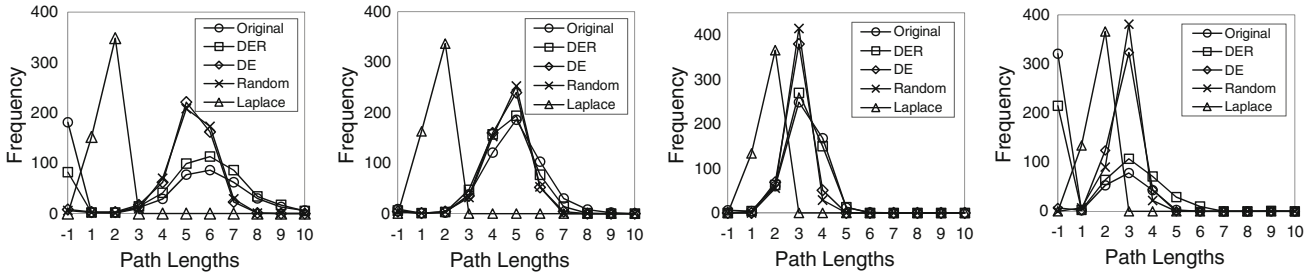



**Fig. 13** Degree distribution versus sampling step

*Shortest path length* Shortest path length is one of the most robust measures of network topology. Our next set of experiments demonstrate that *DER* is also effective for preserving shortest path lengths. Identical to the experimental setting of [6], we randomly select 500 pairs of vertices and compare the distributions of their shortest path lengths between the sanitized dataset and the original dataset.

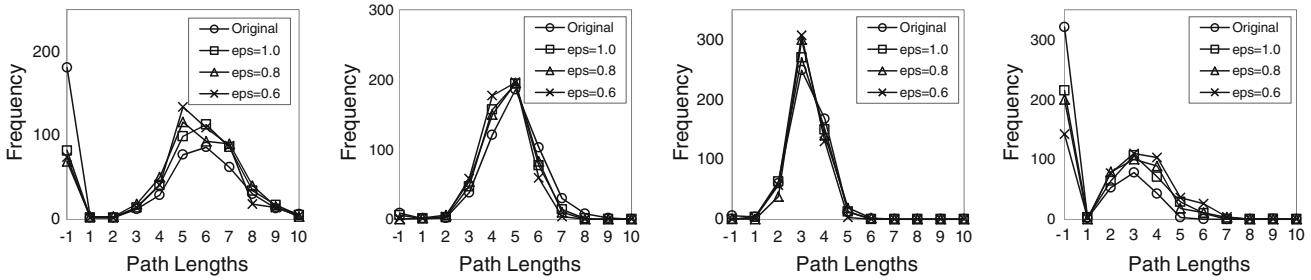
Figure 14 shows the distributions of shortest path lengths of different approaches, where $\epsilon = 1.0$ and $k = 1$. "$-1$" on $X$-axis means that the pairs of vertices are not connected (i.e, the length is $\infty$). As can be observed, *DER* obtains close distributions on all sanitized datasets. Similar to the previous observations, neither *Laplace* nor *Random* can preserve useful shortest path length distributions. *Laplace* gives similar distributions over all datasets. Without the *ArrangeEdge* procedure, *DE* cannot preserve the shortest path length distributions either. Its performance is similar to *Random* on all datasets. Our experimental results conclude that *ArrangeEdge* is critical to achieving desirable utility on different data analysis tasks.

We also study the performance of *DER* on shortest path length with respect to other parameters. Figure 15 presents the shortest path length distributions under different privacy budgets, from which we can observe that a larger privacy budget gives a more similar distribution. Figure 16 gives the distributions under varying correlation parameters, where we can observe that larger correlation leads to worse utility. Figure 17 shows the distributions under various sampling steps.



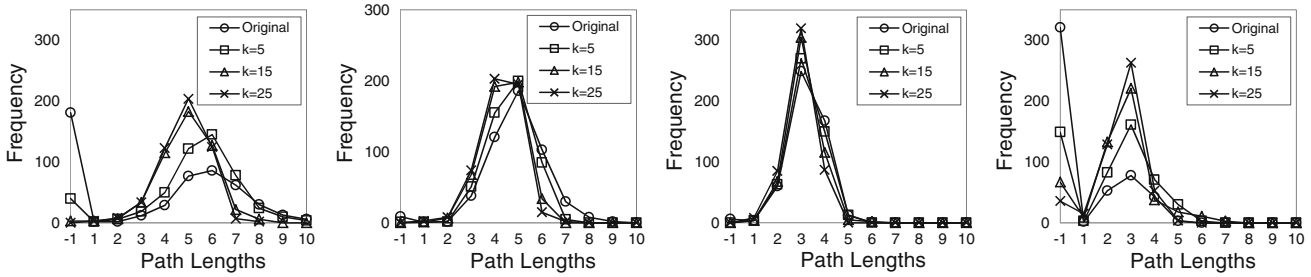

**(a)** ca-GrQc **(b)** ca-HepTh **(c)** wiki-Vote **(d)** STM

**Fig. 14** Distributions of shortest path lengths of different approaches



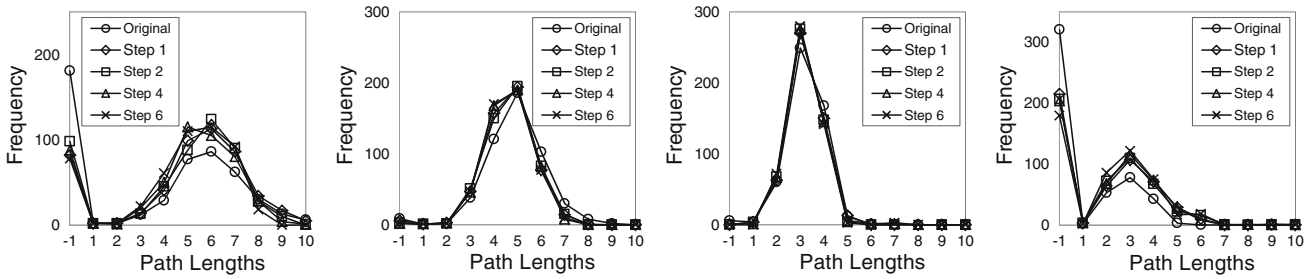

**(a)** ca-GrQc **(b)** ca-HepTh **(c)** wiki-Vote **(d)** STM

**Fig. 15** Distribution of shortest path lengths versus privacy budget $\epsilon$



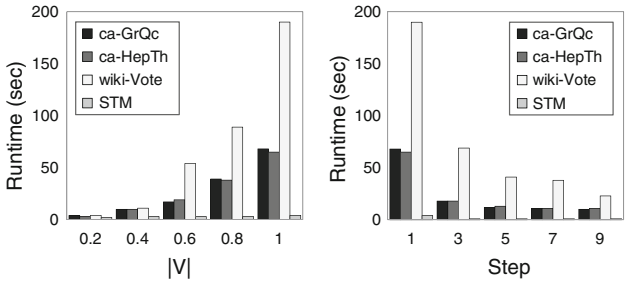

**(a)** ca-GrQc **(b)** ca-HepTh **(c)** wiki-Vote **(d)** STM

**Fig. 16** Distribution of shortest path lengths versus correlation parameter $k$



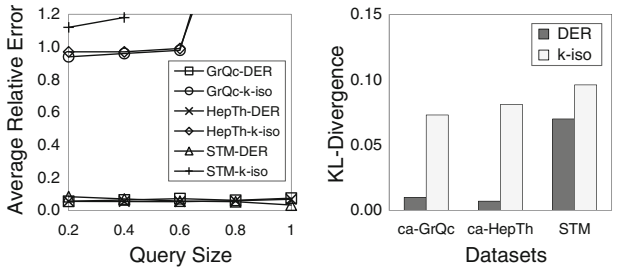

**(a)** ca-GrQc **(b)** ca-HepTh **(c)** wiki-Vote **(d)** STM

**Fig. 17** Distribution of shortest path lengths versus sampling step




**(a)** Number of vertice **(b)** Sampling step

**Fig. 18** Run-time versus different parameters




**(a)** Cut query **(b)** Degree distribution

**Fig. 19** Comparisons on cut query and degree distribution

In general, *DER* performs stably well under different parameter settings.

### 6.2 Efficiency

According to the complexity analysis of *DER*, its run-time is dominated by $|V|$. Thus, we present the run-time of *DER* under different $|V|$ values in Fig. 18a. The test sets are generated by randomly extracting a subset from the original datasets. The *X*-axis represents the percentage of the test sets' $|V|$ values with respect to the original datasets. It can be observed that roughly the run-time grows quadratically with $|V|$, which confirms our theoretical analysis.

Since our approach is used in the *non-interactive* setting, it meets the scalability requirement of most real-life applications. In the few extreme cases, we note that *DER* can substantially speed up by sampling (see Sect. 5.3) at the cost of slight utility degradation. In Fig. 18b, we report the run-time of *DER* for processing the four datasets using different sampling steps.[7] The run-time decreases quickly with the increase of the sampling step. Using the step of 3 makes *DER* approximately 3 times faster. When we increase the step to 6, *DER* becomes roughly 7 times faster. When the sampling step becomes larger than 5, the speedup becomes less obvious because in these cases I/O costs (i.e., the costs of loading the datasets from hard drive into main memory) dominate the run-time.

### 6.3 An experimental comparison with *k*-isomorphism

The core in privacy-preserving data publishing is the trade-off between privacy and data utility. A data publisher typically chooses a solution that results in the best trade-off. *k*-isomorphism and edge differential privacy are not directly comparable in terms of privacy because they aim at different types of privacy guarantee: *k*-isomorphism provides strong privacy protection for node re-identification while edge differential privacy prevents edge disclosure. However, it is beneficial to conduct a comparison from a utility-driven perspective. There are scenarios where the data publisher has predetermined utility requirements. In this section, we aim to experimentally answer the question "*given a utility requirement, which privacy model, k-isomorphism or edge differential privacy, should be used*."

In the following experiments, we set $k = 5$ for *k*-isomorphism and $\epsilon = 1.0$ for edge differential privacy. Both parameters roughly correspond to medium privacy protection under their privacy models. In Fig. 19a, we show the average relative errors of $DER$ and the *k*-isomorphic algorithm [6] (referred to as *k-iso*) for cut queries. Since *k-iso* was not able to process *wiki-Vote*, we only report the experimental results of the other three datasets. It can be observed that *k-iso* is not ideal for answering cut queries. The average relative errors are high in all cases.

Figure 19b suggests that both methods perform well for preserving degree distribution. $DER$ is slightly better than *k-iso*. We present the comparisons on shortest path length distributions in Fig. 20. Again both methods can preserve essential information of shortest path length distribution. *DER* performs better on *ca-GrQc* and *ca-HepTh*, while *k-iso* works better on *STM*.

In addition to utility concerns, scalability is also an important factor for selecting a proper sanitization solution. Figure 21 gives the run-time of both *DER* and *k-iso*. Note that the *Y*-axis is in log scale. In general, *DER* is significantly faster than *k-iso*. In particular, it takes 17,155 seconds for *k-iso* to process *STM*, whereas it takes only 4 seconds for *DER*.

In summary, for cut queries, *DER* is a better choice; for degree distribution and shortest path length, the data publisher could select a solution based on his privacy requirement (either node re-identification or edge disclosure). If the run-time requirement is critical, then *DER* is a better choice.

## 7 Conclusions

In this paper, we analyze the properties of differential privacy in the correlated setting and indicate that if the extent of

[7] Step 1 means that sampling is not employed.

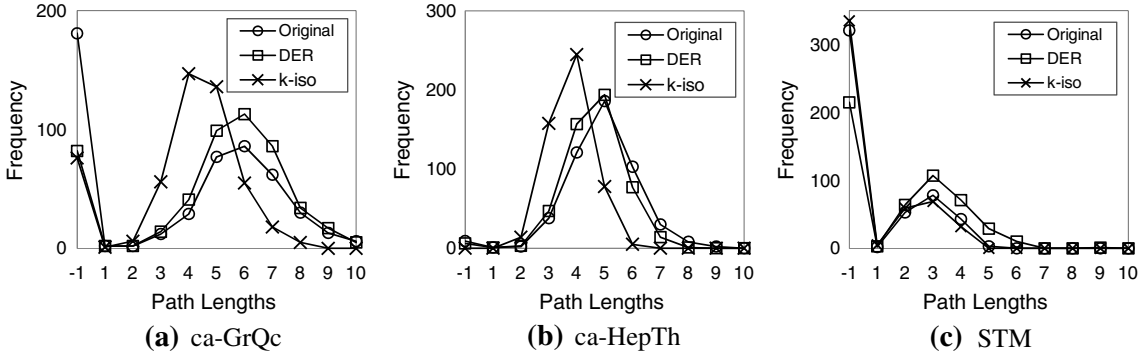



**Fig. 20** Comparison on shortest path length distributions

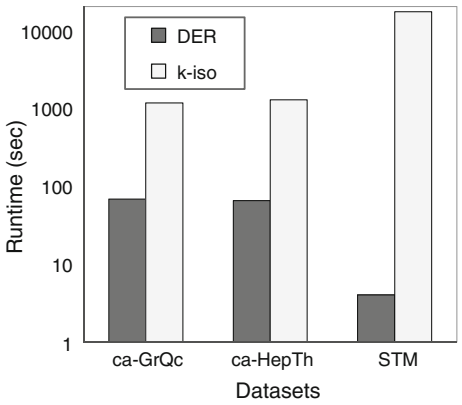



**Fig. 21** Comparison on scalability

correlation can be measured, differential privacy can still provide provable privacy guarantees. Consequently, we present an efficient non-interactive approach for publishing correlated network data. Our approach first identifies a good vertex labeling to make the corresponding adjacency matrix form dense clusters, then conducts a data-dependent exploration for the dense regions in the adjacency matrix and finally reconstructs these regions based on a novel use of the exponential mechanism. This is the first work that gives a practical solution for network data publication via differential privacy. Extensive experiments demonstrate that our solution performs well for various data analysis tasks on different types of real-life network datasets. In addition, we conduct a utility-driven comparison between $k$-isomorphism and edge differential privacy, which guides a data publisher to select a proper privacy model for a given utility requirement.

**Acknowledgments** We sincerely thank the reviewers for their insightful comments. We thank James Cheng, Ada Wai-Chee Fu and Jia Liu for providing the source code of $k$-isomorphism. The research is supported in part by NSERC through Discovery Grants (356065-2013), US NSF through grants CNS-1115234, DBI-0960443 and OISE-1129076 and US Department of Army through grant W911NF-12-1-0066.

## References

1. Backstrom, L., Dwork, C., Kleinberg, J.: Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In: Proceedings of the 16th International Conference on World Wide Web (WWW), pp. 181–190 (2007)
2. Bearman, P.S., Moody, J., Stovel, K.: Chains of affection: the structure of adolescent romantic and sexual networks. Am. J. Sociol. **110**(1), 44–91 (2004)
3. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
4. Bhagat, S., Cormode, G., Krishnamurthy, B., Srivastava, D.: Class-based graph anonymization for social network data. Proc. VLDB Endow. **2**(1), 766–777 (2009)
5. Chen, R., Mohammed, N., Fung, B.C.M., Desai, B.C., Xiong, L.: Publishing set-valued data via differential privacy. Proc. VLDB Endow. **4**(11), 1087–1098 (2011)
6. Cheng, J., Fu, A.W.C., Liu, J.: K-isomorphism: privacy preserving network publication against structural attacks. In: Proceedings of the 36th ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 459–470 (2010)
7. Cormen, T.H., Stein, C., Rivest, R.L., Leiserson, C.E.: Introduction to Algorithms, 3rd edn. McGraw-Hill Higher Education, New York (2009)
8. Cormode, G., Srivastava, D., Yu, T., Zhang, Q.: Anonymizing bipartite graph data using safe groupings. Proc. VLDB Endow. **1**(1), 833–844 (2008)
9. Cormode, G., Procopiuc, M., Shen, E., Srivastava, D., Yu, T.: Differentially private spatial decompositions. In: Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE), pp. 20–31 (2012)
10. Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: Proceedings of the 1969 24th National Conference, pp. 157–172 (1969)
11. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Proceedings of the 3rd Theory of Cryptography Conference (TCC), pp. 265–284 (2006)
12. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. In: Proceedings of the 23rd ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pp. 251–262 (1999)
13. Finkel, R.A., Bentley, J.L.: Quad trees: a data structure for retrieval on composite keys. Acta Informatica **4**(1), 1–9 (1974)
14. Fortunato, S.: Community Detection in Graphs. CoRR abs/0906.0612 (2009)
15. Gupta, A., Ligett, K., McSherry, F., Roth, A., Talwar, K.: Differentially private combinatorial optimization. In: Proceedings of the

21st ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1106–1125 (2010)
16. Gupta, A., Roth, A., Ullman, J.: Iterative constructions and private data release. In: Proceedings of the 9th Theory of Cryptography Conference (TCC), pp. 339–356 (2012)
17. Hay, M., Miklau, G., Jensen, D., Towsley, D.F., Weis, P.: Resisting structural re-identification in anonymized social networks. Proc. VLDB Endow. **1**(1), 102–114 (2008)
18. Hay, M., Li, C., Miklau, G., Jensen, D.: Accurate estimation of the degree distribution of private networks. In: Proceedings of the 9th IEEE International Conference on Data Mining (ICDM), pp. 169–178 (2009)
19. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially private histograms through consistency. Proc. VLDB Endow. **3**(1), 1021–1032 (2010)
20. Hay, M., Liu, K., Miklau, G., Pei, J., Terzi, E.: Privacy-aware data management in information networks. In: Proceedings of the 37th ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 1201–1204 (2011)
21. Kamel, I., Faloutsos, C.: Hilbert R-tree: an improved R-tree using fractals. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), pp. 500–509 (1994)
22. Karwa, V., Raskhodnikova, S., Smith, A., Yaroslavtsev, G.: Private analysis of graph structure. Proc VLDB Endow. **4**(11), 1146–1157 (2011)
23. Kifer, D., Gehrke, J.: Injecting utility into anonymized datasets. In: Proceedings of the 32nd ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 217–228 (2006)
24. Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Proceedings of the 37th ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 193–204 (2011)
25. Kossinets, G., Watts, D.: Empirical analysis of an evolving social networks. Science **311**(5757), 88–90 (2006)
26. Liu, K., Terzi, E.: Towards identity anonymization on graphs. In: Proceedings of the 34th ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 93–106 (2008)
27. Liu, L., Wang, J., Liu, J., Zhang, J.: Privacy preservation in social networks with sensitive edge weights. In: Proceedings of the 9th SIAM International Conference on Data Mining (SDM), pp. 954–965 (2009)
28. McSherry, F.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Proceedings of the 35th ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 19–30 (2009)
29. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: Proceedings of the 48th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 94–103 (2007)
30. Mohammed, N., Chen, R., Fung, B.C.M., Yu, P.S.: Differentially private data release for data mining. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 493–501 (2011)
31. Proserpio, D., Goldberg, S., McSherry, F.: A workflow for differentially-private graph synthesis. In: Proceedings of the 2012 ACM Workshop on Online Social Networks (WOSN), pp. 13–18 (2012)
32. Sala, A., Zhao, X., Wilson, C., Zheng, H., Zhao, B.Y.: Sharing graphs using differentially private graph models. In: Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement (IMC), pp. 81–98 (2011)
33. Schaeffer, S.E.: Graph clustering. Comput. Sci. Rev. **1**(1), 27–64 (2007)
34. Wu, L., Ying, X., Wu, X.: Reconstruction from randomized graph via low rank approximation. In: Proceedings of the 10th SIAM International Conference on Data Mining (SDM), pp. 60–71 (2010)
35. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. In: Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE), pp. 225–236 (2010)
36. Xiao, X., Bender, G., Hay, M., Gehrke, J.: iReduct: differential privacy with reduced relative errors. In: Proceedings of the 37th ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 229–240 (2011)
37. Ying, X., Wu, X.: Randomizing social networks: a spectrum preserving approach. In: Proceedings of the 8th SIAM International Conference on Data Mining (SDM), pp. 739–750 (2008)
38. Yuan, M., Chen, L., Yu, P.S.: Personalized privacy protection in social networks. Proc. VLDB Endow. **4**(2), 141–150 (2011)
39. Zhou, B., Pei, J.: Preserving privacy in social networks against neighborhood attacks. In: Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE), pp. 506–515 (2008)
40. Zou, L., Chen, L., Ozsu, M.T.: K-automorphism: a general framework for privacy preserving network publication. Proc. VLDB Endow. **2**(1), 946–957 (2009)