PEG: Local Differential Privacy for Edge-Labeled Graphs

André L. C. Mendonça
andre.luis@lsbd.ufc.br
Universidade Federal do Ceará
Fortaleza, Ceará, Brazil

Felipe T. Brito
felipe.timbo@lsbd.ufc.br
Universidade Federal do Ceará
Fortaleza, Ceará, Brazil

Javam C. Machado
javam.machado@lsbd.ufc.br
Universidade Federal do Ceará
Fortaleza, Ceará, Brazil

ABSTRACT

Edge-labeled graphs are a particular class of graphs designed to represent networks in which the edge content indicates a type of relationship between two nodes. The study of edge-labeled graphs finds applications in diverse fields, such as anomaly detection, mobility analysis, and community search. Since edge-labeled graphs usually contain sensitive information, preserving privacy when releasing this data type for graph analytics becomes an important issue. In this context, local differential privacy (LDP) has emerged as a robust definition for data release under solid privacy guarantees. However, existing graph LDP techniques in the literature primarily focus on traditional graph structures without considering the nuanced labels associated with edges in labeled graphs. In this paper, we introduce PEG, a novel approach designed to release edge-labeled graphs with local differential privacy guarantees. By combining partitioning and clustering techniques, we enable more effective noise distribution among similar nodes, which preserves the inherent structure and relationships within the released graph. Extensive experiments on real-world datasets show that PEG can effectively release useful and private edge-labeled graphs, enabling subsequent computation of various graph analysis metrics with high utility, including applications in community detection.

1 INTRODUCTION

Graphs are fundamental data structures that represent relationships between entities. Edge-labeled graphs emerge as a special class of graphs designed to represent networks in which the edge content indicates a type of relationship between two nodes. Edge-labeled graphs have been widely adopted in many fields to explain why and how users make connections to each other. Examples include communication networks [39], co-authorship networks [3], protein-protein networks [24], and heterogeneous information networks [37]. The study of edge-labeled graphs has become a prosperous research area, finding applications in anomaly detection [36], mobility analysis [27], and community search [29]. Figure 1 shows an example of an edge-labeled graph G, where the topic of exchanged text messages is considered as the edge label. For this particular case, AM indicates administrative matters, that is, emails related to the management and organization of business operations, and WR denotes work-related topics, such as project updates, meeting requests, collaboration requests, and task assignments.

Due to the sensitive nature of the information found in edge-labeled graphs, releasing such data for analysis and statistical purposes to machine learning practitioners and data scientists without adequate privacy guarantees could put individuals' privacy at risk. For instance, in Figure 1, let's say an adversary knows that user "g" only sends administrative matters (topic AM) to user

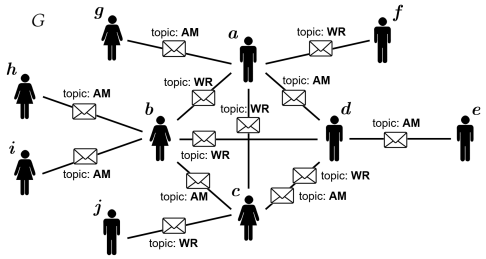

Figure 1: An example of an edge-labeled graph G where nodes represent users and edges are emails exchanged among them. The topic of the emails represents the edge label. AM denotes administrative matters, and WR depicts work-related topics.

"a". The adversary can infer that user "g" likely holds a position of authority or responsibility within the organization, particularly concerning administrative tasks or decision-making processes.

Differential privacy (DP) [13] has emerged as a robust privacy definition that has become the standard for data release under strong privacy guarantees. The main idea behind DP is that an analysis is determined by a randomized algorithm, also known as a mechanism, that computes private information and returns a randomized answer sampled from a probability distribution. In the literature, the primary DP setups are the global DP [13] and the local DP (LDP) [12]. The key difference between them consists in the nature of the data curator. In the global setting, it is assumed that a trusted data curator has indiscriminate access to the complete data and is responsible for releasing it after a differentially private procedure. Conversely, in the local setting, the data curator is assumed to be untrustworthy. In this case, each user is responsible for applying privacy to their own data before sending it to the data curator. Compared to the global DP, local DP has a stronger notion of privacy since it keeps the individuals' sensitive data private, even from untrustworthy data curators.

The standard DP models (global and local) have been initially defined to attend to tabular data. However, studies have been developed over the years in the field of the differentially private release of graph data [6]. Within the graph scope and following the definition of neighboring graphs, there are two main DP settings: the edge differential privacy (edge-DP) [23] and the node differential privacy (node-DP) [26]. In the DP model, two datasets are neighbors if they differ in at most one single record. In the graph context, the edge-DP model states that two graphs are neighbors if they differ in at most one single edge. In contrast, the node-DP model states that two graphs are neighbors if they differ in exactly one node and all its incident edges. However, for labeled graphs, neither edge-DP nor node-DP privacy models are adequate since they ignore the presence of attributes on the edges.

Many efforts have already been made to protect individuals' privacy in edge-weighted graphs, i.e., graphs that contain numerical attributes on their edges [5, 8, 17, 33, 35, 40]. However, these

works often face limitations when dealing with non-numeric attributes [10, 30]. Some studies [9, 25, 43, 47] have applied DP in node-attributed graphs, focusing on methods that consider node attributes (instead of edge attributes) for privacy-preserving data releases. On the other hand, Liu et al. [31] proposes a method that specifically addresses local differential privacy for edge-labeled graphs. However, it only provides privacy for a few statistics rather than releasing the entire labeled graph for comprehensive graph analytics.

In this paper, we present PEG (**P**rivacy for **E**dge-labeled **G**raphs), an approach for releasing entire edge-labeled graphs under local differential privacy guarantees. In summary, the main contributions of this paper are as follows:

(1) We first introduce the Randomized Attribute Neighbor List (RANL), a novel data structure for encoding edge-labeled graphs in the LDP setting.
(2) We then present a new method that combines partitioning and clustering techniques to achieve more effective noise distribution among similar nodes, which improves data utility when applying the RANL data structure.
(3) We also improve the accuracy of the released graph by developing a post-processing technique to guarantee graph consistency.
(4) Finally, we conduct an extensive experimental analysis on four real-world edge-labeled graphs to evaluate the performance of PEG. We show that our approach achieves high utility for various graph analysis metrics on the released graph, including applications in community detection.

The rest of the paper is structured as follows: Section 2 provides the theoretical background overview. In Section 3, we review the existing literature. We then present PEG in Section 4. Section 5 empirically evaluates PEG. Finally, Section 6 concludes the paper and provides future research directions.

2 THEORETICAL BACKGROUND

Edge-labeled graphs are a specific type of graph where non-numeric attributes are assigned to the edges. These graphs can model various types of relationships between nodes, where each edge belongs to a category. We denote an undirected edge-labeled graph as $G = (V, E, X)$, where V is the set of nodes, E is the set of edges, and X is the set of labels associated with each edge in E. The set of nodes is defined as $V = \{v_1, \ldots, v_n\}$, such as $|V| = n$. The set of labels is defined as $X = \{x_1, \ldots, x_t\}$, and t is the number of possible edge labels. The set of edges is defined as $E \subseteq V \times V \times X = \{e_{i,j,k}, \ldots, e_{o,p,q}\}$, where $e_{i,j,k}$ refers to an undirected edge between nodes v_i and $v_j \in V$ associated with the label $x_k \in X$ ($e_{i,j,k} \equiv e_{j,i,k}$), and $|E| = m$. Additionally, in this work, we consider that G holds the multigraph property, meaning that for any nodes $v_i, v_j \in V$, there may exist multiple edges $\{e_{i,j,k}, \ldots e_{i,j,l}\}$ with $x_k, x_l \in X$, such that $x_k \neq x_l$ for any $x_k, x_l \in X$.

2.1 Differential Privacy

Differential privacy (DP) [13] is a robust privacy definition that has become the standard for data release under strong privacy guarantees. In summary, it states that any answer to a query occurs with similar probability regardless of the presence or absence of any individual in the dataset, as defined as follows:

Definition 2.1. (ε-Differential Privacy [13]). A randomized algorithm $\mathcal{A}$ satisfies ε-differential privacy if for any two neighboring datasets D, D' and for any output $O \subseteq Range(\mathcal{A})$,

$$Pr[\mathcal{A}(D) = O] \leq e^{\varepsilon} Pr[\mathcal{A}(D') = O]. \quad (1)$$

Note that D and D' are neighbors if they differ in at most one single record. The parameter ε, known as the privacy budget, is an input parameter that controls the level of privacy protection in DP. Specifically, it determines the trade-off between the accuracy of the data output and the level of privacy protection for individuals in the dataset. A smaller ε corresponds to a stronger privacy-preserving guarantee, while a higher ε provides better data utility. The randomized algorithm $\mathcal{A}$ is also referred to as a mechanism, which is a way of achieving DP. For numerical queries, DP can be achieved by various mechanisms, such as Laplace [13] (for continuous data) and geometric [21] (for discrete data) mechanisms. Our approach follows the geometric mechanism based on the global sensitivity of a query.

Definition 2.2. (Global Sensitivity [14]). The global sensitivity of a query Q is the maximum l_1 distance between the outputs of Q on any two neighboring datasets D and D', given by

$$\Delta Q = \max_{D,D'} ||Q(D) - Q(D')||_1. \quad (2)$$

The geometric mechanism adds integer noise to the true query answers following the two-sided geometric distribution, as defined below.

Definition 2.3. (Two-sided Geometric Distribution). A random variable X distributed as a two-sided geometric distribution, with mean 0 and $\alpha \in [0, 1]$, has a probability mass function

$$P(X = x) = \frac{1 - \alpha}{1 + \alpha} \alpha^{|x|}. \quad (3)$$

We denote $Geom(\frac{\varepsilon}{\Delta Q})$ the two-sided geometric distribution with mean 0 and $\alpha = e^{-\frac{\varepsilon}{\Delta Q}}$.

Theorem 2.4. *(Geometric Mechanism [21]) Given any query $Q : \mathbb{N}^{|\mathcal{D}|} \rightarrow \mathbb{Z}^k$, the geometric mechanism defined as*

$$\mathcal{A}_G(D, Q, \varepsilon) = Q(D) + (Y_1, \ldots, Y_k), \quad (4)$$

where Y_i are i.i.d. random variables draw from $Geom(\frac{\varepsilon}{\Delta Q})$ and $\mathcal{D}$ is the set of all possible datasets, satisfies ε-DP with $\alpha = e^{-\frac{\varepsilon}{\Delta Q}}$.

Various useful properties are present in differentially private mechanisms. When combined, these properties offer the flexibility to aggregate multiple differentially private steps into a single mechanism that satisfies differential privacy. These properties are:

Theorem 2.5. *(Post-processing [15]) Let $\mathcal{A}$ be any randomized algorithm such that $\mathcal{A}(D)$ is ε-differentially private, and let f be any function. Then, $f(\mathcal{A}(D))$ also satisfies ε-DP.*

Theorem 2.6. *(Sequential Composition [15]) Let $\mathcal{A}_i$ provide ε_i-differential privacy. A sequence of differentially private algorithms $\mathcal{A}_i(D)$ provides $\sum \varepsilon_i$-DP.*

Theorem 2.7. *(Parallel Composition [15]) Let each D_i be disjoint data and $\mathcal{A}$ an algorithm that provides ε_i-differential privacy for data D_i. A sequence of differentially private algorithm execution $\mathcal{A}(D_i)$ provides $\max(\varepsilon_i)$-DP.*

2.2 Local Differential Privacy

Differential privacy typically involves a trusted curator (third party) responsible for collecting, perturbing, and publishing data through a mechanism that satisfies differential privacy (DP). Alternatively, Local Differential Privacy (LDP) offers a private approach that eliminates the need for a trusted data curator. In the LDP setting, instead of centralizing the data flow to a single, supposedly reliable entity, each user locally perturbs their data with a differentially private mechanism before sending it to the data curator. In this context, the data curator is commonly called an aggregator. Compared to global DP, LDP provides a stronger notion of privacy:

Definition 2.8. (ε-Local Differential Privacy [12]). A randomized algorithm $\mathcal{A}$ satisties ε-local differential privacy if for any pair of values $v, v' \in D$ and for any possible output $O \subseteq Range(\mathcal{A})$,

$$Pr[\mathcal{A}(v) = O] \leq e^{\varepsilon} Pr[\mathcal{A}(v') = O]. \tag{5}$$

2.2.1 Local Differential Privacy Protocols. In the LDP setting, mechanisms are commonly called protocols, i.e., techniques designed to modify the user's data to ensure LDP. The standard flow of an LDP protocol consists of (i) encoding the user's data, (ii) perturbing the user's data, and (iii) reporting the noisy data to the data curator. *Encoding:* The user's input data v is mapped into a binary bit vector B composed of 0's and 1's, such that $B[v] = 1$. *Perturbation:* The bits of B are modified according to probabilities p and q established by the protocol, as depicted in Equation 6. *Reporting:* The user's noisy data is sent to the data curator for analysis.

Several protocols have been proposed for different purposes [2, 4, 11, 16, 41, 44], with *Randomized Response* (RR) [14] and *Optimized Unary Encoding* (OUE) [41] being among the most commonly used.

$$\Pr[B'[i] = 1] = \begin{cases} p, & \text{if } B[i] = 1 \\ q, & \text{if } B[i] = 0 \end{cases} \tag{6}$$

Randomized Response Protocol [14]. The RR protocol allows an input value v to be encoded into a bit vector B such that $B[v]$ may be represented by more than one bit assigned to 1. It was proved in [16] that the RR protocol satisfies ε-LDP if the bit vector B is perturbed according to probabilities $p = \frac{e^{\varepsilon}}{1+e^{\varepsilon}}$ and $q = 1 - p$.

Optimized Unary Encoding Protocol [41]. The OUE differs from the RR primarily in how the data is encoded and in the probabilities p and q. In the OUE protocol, an input value v is encoded into a bit vector B such that $B[v]$ is represented by only one bit set to 1. The protocol proposes optimized values for p and q that reduce the variance of the reported data, thereby improving the data utility. It was proven in [41] that the OUE protocol satisfies ε-LDP if the bit vector B is perturbed according to probabilities $p = \frac{1}{2}$ and $q = \frac{1}{e^{\varepsilon}+1}$.

2.3 Local Differential Privacy For Graphs

In the context of LDP, the literature focuses on attacks where an adversary attempts to infer the presence or absence of nodes or edges in graphs. In this sense, there are two main settings of LDP for graphs: Edge Local Differential Privacy (edge-LDP) [23] and Node Local Differential Privacy (node-LDP) [26]. Given an undirected graph $G = (V, E)$, for each node $v_i \in V$, let $B_i = \{b_1, b_2, \ldots, b_n\}$ be the adjacency bit vector of v_i, where $b_j = 1$ if and only if $e_{i,j} \in E$, otherwise $b_j = 0$. Then, both definitions are stated as:

Definition 2.9. (ε-Edge Local Differential Privacy [23]). A randomized algorithm $\mathcal{A}$ satisfies ε-edge local differential privacy if and only if for any two adjacency bit vectors B, B' that differ only in one bit, and for any output $O \subseteq Range(\mathcal{A})$,

$$Pr[\mathcal{A}(B) = O] \leq e^{\varepsilon} Pr[\mathcal{A}(B') = O]. \tag{7}$$

Definition 2.10. (ε-Node Local Differential Privacy [26]). A randomized algorithm $\mathcal{A}$ satisfies ε-node local differential privacy if for any two adjacency bit vectors B, B' and for any output $O \subseteq Range(\mathcal{A})$,

$$Pr[\mathcal{A}(B) = O] \leq e^{\varepsilon} Pr[\mathcal{A}(B') = O]. \tag{8}$$

Achieving privacy under node-LDP is much harder than it is in edge-LDP since it requires protecting the privacy of the entire node's data, including all its connections. Therefore, designing algorithms that ensure node-LDP and simultaneously provide accurate graph analytics may not be feasible. Nonetheless, edge-LDP can still achieve strong privacy protection regarding the existence of edges, which is sufficient for most graph applications, such as community search [29], and anomaly detection [36] while preserving high data utility. Therefore, this paper focuses on the edge-LDP setting. In the rest of the paper, we use the terms edge attribute and edge label interchangeably to refer to the information on the edges.

3 RELATED WORK

The concept of DP has been extensively studied and applied across various domains, including graph data analysis. In this context, edge-DP [23] focuses on preserving privacy in graph data by perturbing the edges of the graph. Early works in this area primarily centered around techniques, such as edge addition or removal [38, 46], and synthetic graphs generation [20, 34], to achieve local differential privacy guarantees. On the other hand, node-DP [26] extends the principles of DP to individual nodes within a graph. Instead of perturbing entire edges, node-DP aims to protect the privacy of specific nodes and their associated edges. However, due to the complexities involved in achieving satisfactory levels of utility while adhering to LDP, only a few works consider node-DP in the context of local differential privacy [18, 19]. In recent years, the differential privacy setting for edge-weighted graphs was formally introduced [35]. Particularly, edge-weight-DP techniques introduce noise to the numerical values associated with edges, representing strengths, distances, or other quantitative measures [5, 42]. However, while edge-DP, node-DP, and edge-weight-DP offer valuable techniques for privacy-preserving graph analysis, none of these methods were specifically developed to handle attributed graphs.

In the field of attributed graphs, two main types are particularly relevant: node-attributed and edge-attributed graphs. Jorgensen et al. [25] proposed the TriCycLe model for releasing synthetic node-attributed graphs, i.e., graphs with attributes in the nodes, under DP guarantees. In summary, TriCycLe collects information about the original graph through differentially private analyses and generates a synthetic graph according to them. It rewrites the graph connections until the graph structure closely resembles the original. In the same context, Chen et al. [9] proposed the Community-Preserving Attributed Graph Model (C-AGM), a method that captures the properties of the communities from a node-attributed graph and releases a synthetic graph with

DP guarantees. Additionally, Wei et al. [43] proposes AsgLDP, a local approach for releasing synthetic node-attributed graphs. AsgLDP is a two-phase framework based on the local-DP setting. In the first phase, the users report some properties related to their local graphs, while in the second phase, the data collector performs an unbiased estimation of the reported data to sample a private synthetic graph.

Liu et al. [31] proposed the PrivAG framework under a novel neighboring definition for the LDP model. The authors define the attribute-wise LDP, which considers two graphs to be neighbors if they differ in one attribute and all related edges associated with that attribute. Although PrivAG addresses LDP for edge-labeled graphs with non-numerical attributes, the approach can cause severe data distortion, as in the worst case, it is equivalent to the node-LDP notion. Additionally, PrivAG has a limited analysis scope, releasing only specific graph metrics rather than a full DP graph. On the other hand, our approach aims to release an entire edge-labeled graph under LDP guarantees, enabling subsequent computations for various graph analyses.

4 THE PEG APPROACH

PEG is a multiphase approach in which each user is responsible for privatizing their data locally employing LDP. PEG is divided into four main phases: (i) *Partitioning & Clustering*; (ii) *Partition-Cluster Mapping*; (iii) *RANL Reporting*; and (iv) *Graph Post-Processing*. Before explaining each phase, we introduce the *Randomized Attribute Neighbor List (RANL)*, a novel data structure for encoding edge-labeled graphs in the LDP setting. RANL is a key contribution for understanding PEG.

The Randomized Attribute Neighbor List consists of an encoding structure through which each user can report their neighborhood locally, i.e., all the edges and their labels that form the user's local graph. The RANL combines the key features of two existing encoding methods, the Randomized Neighbor List (RNL) [34] and the Randomized Attribute List (RAL) [43] to suit our context of edge-labeled graphs. In summary, the RNL consists of a n-length bit vector, where n is the number of users in V, i.e. $n = |V|$. The RNL of a user v_i is given by $\text{RNL}_{v_i} = [e_{i,1}, \ldots, e_{i,n}]$, where $e_{i,j} \in \{0, 1\}$ denotes whether exists the edge $e_{i,j} \in E$, i.e., if exists a connection between users v_i and v_j. Note that only the connection is considered, disregarding the existence of any property on the edge. Consequently, the RAL is adequate for the context of node-attributed graphs. It consists of a w-length bit vector, where w is the number of possible attributes. Then, the RAL of a user v_i is given by $\text{RAL}_{v_i} = [x_{i,1}, \ldots, x_{i,w}]$, where $x_{i,j} \in \{0, 1\}$ denotes whether exists the j-th attribute of the user v_i. It has been proven [34, 43] that given a privacy budget ε, each user can perturb its RNL or RAL through the RR protocol and send the perturbed data to the data collector through an LDP mechanism while satisfying ε-edge-LDP. Thus, the RANL definition is formally stated as:

Definition 4.1. (Randomized Attribute Neighbor List (RANL)). Given an edge-labeled graph $G = (V, E, X)$ and an user v_i, the RANL of an user v_i is given by a h-length bit vector in the form of $\text{RANL}_{v_i} = [e_{i,1,1}, \ldots, e_{i,1,t}, \ldots, e_{i,n,1}, \ldots, e_{i,n,t}]$, where $n = |V|$ is the number of users in V, $t = |X|$ is the edge label domain size, $h = n \cdot t$, and $e_{i,j,k} \in \{0, 1\}$ denotes whether exists or not the edge between nodes $v_i, v_j \in V$ associated with the label $x_k \in X$.

Given a privacy budget ε within the RR protocol, the process of perturbing and reporting the users' RANLs with probabilities p and q satisfies ε-edge-LDP since adding or removing a single labeled edge will make two neighboring RANLs differ in only one bit. The probabilities are given by $p = \frac{e^{\varepsilon}}{1+e^{\varepsilon}}$ and $q = 1 - p$, where p denotes the probability of not flipping a bit and q is the probability of flipping a bit, respectively.

Another aspect is that the size of the RANL is proportional to the number of users n and the edge label domain size t. Also, graphs usually have a long-tailed degree distribution, meaning that users have low degrees, i.e., few connections. In this scenario, the length of the RANL is large, and the list contains significantly more zero values than ones. Consequently, reporting the RANL through the RR protocol may significantly increase the number of ones. To overcome this issue, we have to devise a way to shorten the length of the RANL. The primary solution involves reducing the user population to limit the number of connections each user can have. Then, we propose a partitioning and clustering strategy where each user within a partition reports their RANL with a length equal to $n^* \cdot t$, where n^* is the number of users among the clusters in their partition, rather than $n \cdot t$. This approach enables more effective noise distribution among similar nodes and consequently preserves the inherent structure and relationships within the released graph.

4.1 Partitioning & Clustering

In this phase, the untrusted data curator first splits all users $V = \{v_1, \ldots, v_n\}$ into p random disjoint sets $\mathcal{P} = \{P_1, \ldots, P_p\}$ of the same size, such that each user $v_i \in V$ belongs to one, and only one, partition $P_j \in \mathcal{P}$, and $|P_j| = \lfloor \frac{n}{p} \rfloor \ \forall_{P_j \in \mathcal{P}}$. Let $|P_j|$ be the size of the partition P_j. It is important to mention that, in this work, we assume the data curator has information on the number of users (n) but does not have knowledge about any user, except that each user is identified by a random identifier. In the cases where the disjoint sets could not be of the same size (due to particularities of the values of n and p), consider $|P_j| = \lfloor \frac{n}{p} \rfloor$ $\forall_{P_j \in \mathcal{P}}$ except for one of the partitions that will be chosen to accommodate the remaining users, given by $(n \bmod p)$.

The next step performed by the data curator is the degree-based clustering. The main idea behind this step is that users with high degrees, meaning many connections, tend to connect with other users who also have high node degrees. In short, consider a node v_i, the degree of v_i consists of the number of connections involving v_i. In the context of edge-labeled graphs, it may be desirable to know not only the degree of v_i but also the degree of v_i considering only the connections with a specific label. We detail these different notions of degree as follows.

Edge Label Degree. Let $Y_i^k = (y_{i,1}^k, \ldots, y_{i,n}^k)$ be the relationship vector of a node $v_i \in V$ regarding the label $x_k \in X$ in an edge-labeled graph $G = (V, E, X)$. If a node v_i is connected to a node v_j and a label x_k is associated to this connection, then $y_{i,j}^k = 1$, otherwise $y_{i,j}^k = 0$. We define $d_{v_i}^{x_k}$ as the edge label degree of a node v_i with label x_k given by $\sum_{j=1}^{n} y_{i,j}^k$. In summary, in an undirected graph, the edge label degree represents the number of edges associated with a specific label connected to a given node. Then, we denote $\delta_{v_i} = (d_{v_i}^{x_1}, \ldots, d_{v_i}^{x_t})$ as the edge label degree vector of a node v_i.

Node Degree. Given an edge-labeled graph $G = (V, E, X)$, We define d_{v_i} as the node degree of a node $v_i \in V$ given by $\sum_{x_k \in X} d_{v_i}^{x_k}$. In summary, in an undirected graph, the node degree represents the number of edges connected to a given node.

Releasing users' degrees without privacy concerns can compromise their privacy. The geometric mechanism [21] is an effective technique for perturbing discrete function values. Then, to ensure edge-LDP, each user $v_i \in V$ adds to their degree d_{v_i} a random noise drawn from the two-sided geometric distribution $Geom(\frac{\varepsilon_1}{2})$, where ε_1 is the privacy budget allocated to this phase and 2 is the sensitivity of the degree function [45]. However, instead of requesting each user to report only their degrees, our approach captures their edge label degree vectors. The edge label degree vectors hold much more relevant information than merely the node degree. It holds information about the node degree per edge label while also allowing us to derive the original node degree by summing up the edge label degrees.

Instead of sharing the degrees with the data curator, each user $v_i \in V$ shares a perturbed version of their edge label degree vectors δ_{v_i}, given by $\tilde{\delta}_{v_i} = (d^{x_1}_{v_i} + Geom(\frac{\varepsilon_1}{2}), \ldots, d^{x_t}_{v_i} + Geom(\frac{\varepsilon_1}{2}))$. Since the edges related to each edge label degree are non-overlapping, adding or removing one edge from a user would change one, and only one, edge label degree of δ_{v_i} by 1. Once the data curator collects all $\tilde{\delta}_{v_i}$, he can derive the node degree of each user v_i by calculating $\tilde{d}_{v_i} = \sum_{x_k \in X} \tilde{\delta} v_i[x_k]$. However, once every edge label degree has been queried through the geometric mechanism, where the noise sample can assume positive or negative values, the original edge label degree may be converted to a value lower than zero, which is not plausible in practical scenarios. In this work, we consider that every user has at least one connection, which leads to a node degree of at least one. For this purpose, the data curator has to prior post-process the collected data before deriving the users' degrees.

To prevent cases where a user's node degree could be estimated as a value lower than zero, the data curator adjusts the perturbed edge label degrees. This process consists of calculating the expected edge label degrees sum of each edge label and, then, using this information to adjust the edge label degrees such that each edge label degree will have a non-negative value and the sum of the adjusted edge label degrees will be the same as the expected edge label degrees sum. For instance, consider $\tilde{s}_k$ as the expected edge label degrees sum of the edges labeled with $x_k \in X$, given by $\tilde{s}_k = \sum_{v_i \in V} \tilde{\delta}_{v_i}[x_k]$. Then, for a user $v_i \in V$, the adjusted edge label degree vector is given by $\bar{\delta}_{v_i}$, such that $\bar{\delta}_{v_i}[x_k] \geq 0 \; \forall_{x_k \in X}$. Additionally, the adjusted edge label degrees sum is given by $\bar{s}_k = \tilde{s}_k \; \forall_{x_k \in X}$. Finally, the data curator can derive the perturbed degree sequence $\bar{\phi}$, where the perturbed degree of a user v_i is given by $\bar{\phi}_{v_i} = \max(1, \bar{d}_{v_i})$.

After collecting and adjusting users' node degrees, the data curator sorts them in descending order and groups users into c clusters based on their degrees. Instead of equal-sized clusters, each cluster is formed to have a similar degree mass, i.e., the sum of the node degrees of its members. Let $s_{\bar{\phi}} = \sum_{v_i \in V} \bar{\phi}_{v_i}$ be the degree mass of the perturbed degrees, i.e., the sum of the degrees. We define the maximum degree mass of each cluster $s_{max} = \frac{s_{\bar{\phi}}}{c}$. Finally, we form the clusters by allocating the users according to the descending degree order until the cluster's mass constraint is not violated. When the degree mass of a cluster reaches s_{max}, or it is not possible to add the next available user with the highest degree into the cluster without exceeding the s_{max} limitation, the current cluster stops receiving new users, and the next cluster starts being populated. Then, we define the set of clusters $C = \{C_1, \ldots, C_c\}$, such that each user $v_i \in V$ belongs to one, and only one, cluster $C_j \in C$, and $s_{C_j} \leq s_{max} \; \forall_{C_j \in C}$, where s_{C_j} is the degree mass of the cluster C_j. However, some clusters

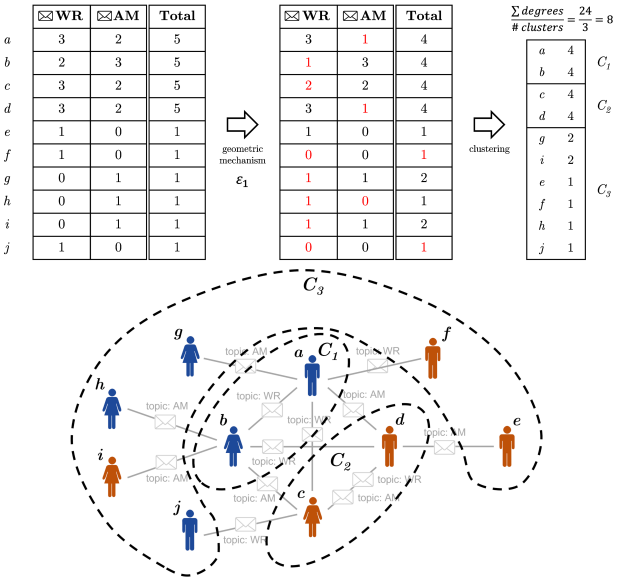

Figure 2: Partitioning & Clustering phase.

may not reach the exact degree mass of s_{max}. Consequently, the last cluster may need to accommodate more users than expected, causing its degree mass to surpass s_{max}. Example 1 illustrates the Partitioning & Clustering phase.

Example 1. Initially, consider the edge-labeled graph G in Figure 1 with $n = 10$, where $V = \{v_a, \ldots, v_j\}$. Figure 2 presents how the partitioning and clustering procedures are performed over the original graph. Suppose that the data curator desires to partition the users into $p = 2$ groups. In this example, all partitions have a size of $\frac{n}{p} = 5$, meaning that all users were allocated into equal-sized partitions. Since the users of each partition are randomly selected, a possible partition set $\mathcal{P}$ is given by $\mathcal{P} = \{P_1, P_2\}$, where $P_1 = \{v_a, v_b, v_g, v_h, v_j\}$ (blue nodes) and $P_2 = \{v_c, v_d, v_e, v_f, v_i\}$ (orange nodes). Now, consider that the data curator desires to cluster the users into $c = 3$ groups. First, each user reports its edge label degrees through the geometric mechanism. Then, the data curator estimates the users' degrees. Note that the users v_f and v_j have reported all their edge label degrees as zero. In these cases, the user degree is assigned to 1 since it is supposed that each user has at least one connection. Thus, the data curator calculates the $s_{max} = \frac{s_{\bar{\phi}}}{c} = \frac{24}{3} = 8$, to get the maximum degree mass that each cluster may have. Finally, the clusters $C_1 = \{v_a, v_b\}, C_2 = \{v_c, v_d\}$ and $C_3 = \{v_e, v_f, v_g, v_h, v_i, v_j\}$ are formed according to the descending order of the noisy degrees and the s_{max} constraint.

4.2 Partition-Cluster Mapping

In this phase, the untrusted data curator aims to determine the cluster each partition belongs to. The users within the partitions and clusters are already known. Each participant within a partition is asked to indicate the cluster they are most likely to belong to based on their connections. After collecting responses, a count is performed to determine the most suitable cluster for the partition based on the majority vote. Instead of assigning clusters to individual nodes, which would introduce excessive noise and destroy information, we consider the majority cluster for the entire partition. This approach enables unbiased estimation of noisy counts, enabling us to infer the majority clusters with high fidelity and assign consistent clusters to the partition.

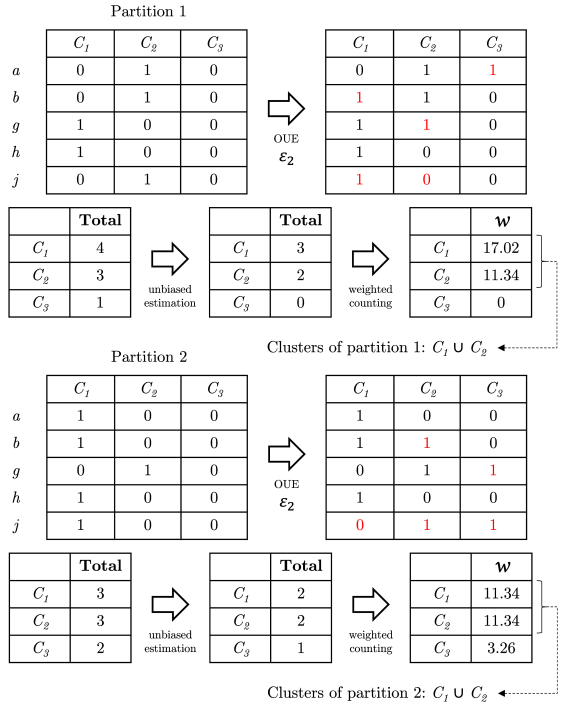

Figure 3: Partition-Cluster Mapping phase.

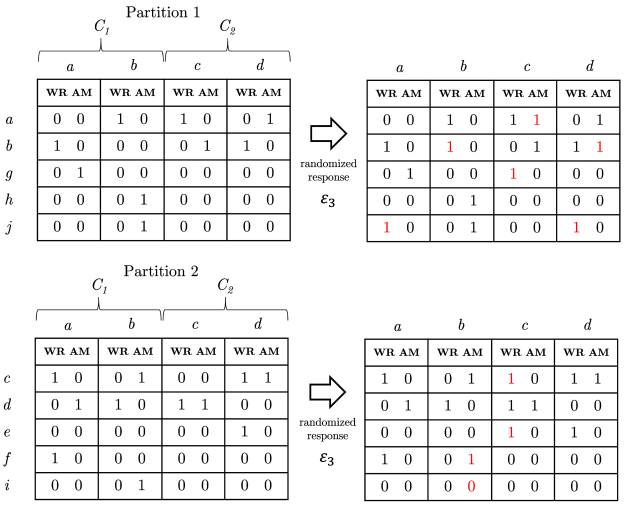

Figure 4: RANL Reporting phase.

The partitions' clusters are privately chosen using an LDP mechanism. The OUE protocol is appropriate in this context, as it utilizes unary encoding, where each user's bit vector contains a single bit set to one. The parameters p and q in OUE are optimized to preserve information by maximizing the accuracy of zero bits reported as zero in the original data.

For each partition $P_j \in \mathcal{P}$, each user $v_i \in P_j$ sends a bit vector B_{v_i} of length c (the number of clusters), indicating the cluster C_k to which they are most likely to belong. The k-th bit vector position denotes whether v_i belongs or not to the cluster C_k. Then, $B_{v_i}[k] = 1$ when v_i states that belongs to C_k, and $B_{v_i}[k] = 0$ otherwise. Finally, the bit vector is perturbed and sent to the data curator through the OUE protocol, ensuring ε_2-edge-LDP.

The data curator then calculates the counts of each cluster by summing up the bits of each vector cluster-wise. We denote $\tilde{count}^{C_k}_{P_j} = \sum_{v_i \in p_j} B_{v_i}[C_k]$ the perturbed count of the cluster C_k in partition P_j. However, simply summing these perturbed bits does not reflect the true counts, as the randomized bit vectors may contain more than one bit set to one. Then, an unbiased estimation is applied to reduce bias and produce more accurate counts. We denote $\bar{count}^{C_k}_{P_j} = \frac{\tilde{count}^{C_k}_{P_j} - (q \cdot |P_j|)}{p \cdot q}$ the estimated count of the cluster C_k in partition P_j, where $|P_j|$ is the number of users in the partition P_j. Similarly to the previous phase, some of the $\bar{count}^{C_k}_{P_j}$ may present negative values. In those situations, we adjust the overall counts such the negative values become ≥ 0, but the sum of the estimated counts remains unchanged.

After estimating the counts, the data curator assigns each partition to a cluster. Selecting the cluster with the highest count can be problematic for long-tailed degree distributions, as it often favors less dense, choosing low-degree clusters that misrepresent dominant relationships in the graph. To address this issue, we introduce a weighting function to adjust the counts based on cluster density using a percentile-based selection method. In Equation 9, we present the weighting function, where $w^{C_k}_{P_j}$ is the weighted count of the cluster C_k in partition P_j, s_{C_k} is the degree mass of C_k, and $|C_k|$ is the number of elements in C_k. Choosing the square root prevents larger clusters from gaining additional advantage, ensuring that even a slightly larger cluster with a much higher estimated count maintains a higher weighted count.

$$w^{C_k}_{P_j} = \bar{count}^{C_k}_{P_j} \cdot \sqrt{\frac{s_{C_k}}{|C_k|}} \tag{9}$$

Finally, after weighting the counts, the clusters for the partition are determined by selecting those where the weighted count reaches the y-th percentile. This method allows for the assignment of more than one cluster to a partition, addressing the uncertainty associated with clusters that have similar counts. This process ensures a more accurate and representative clustering. The procedure is repeated until the clusters for all partitions are properly defined. Example 2 shows how this phase is executed.

Example 2. Consider the edge-labeled graph G in Figure 1 and the partitions $\mathcal{P}$ and clusters C in Figure 2, respectively. Suppose that the data curator desires to define the clusters of each partition according to the 50th percentile. Figure 3 presents how the partition-clustering phase is performed. First, for each partition, each user reports to which cluster it has more connections through the OUE protocol. Then, the data curator estimates and weights the counts according to Equation 9. Finally, the data curator selects the clusters with a weighted count at least equal to the 50th percentile of all weighted counts. For partition P_1, we set $P_{1clus} = \{C_1, C_2\}$, since the 50th percentile of the weighted counts $w_{P_1} = [17.02, 11.34, 0] = 11.34$. Similarly, for partition P_2, we set $P_{2clus} = \{C_1, C_2\}$, since the 50th percentile of the weighted counts $w_{P_2} = [11.34, 11.34, 3.26] = 11.34$. Cluster C_3 was not allocated to represent any partition since its count did not meet the minimum value stated by the 50th percentile.

4.3 RANL Reporting

In this phase, the untrusted data curator aims to gather each user's connections within the graph. Each user reports their neighborhood locally, including all edges and their labels that form the

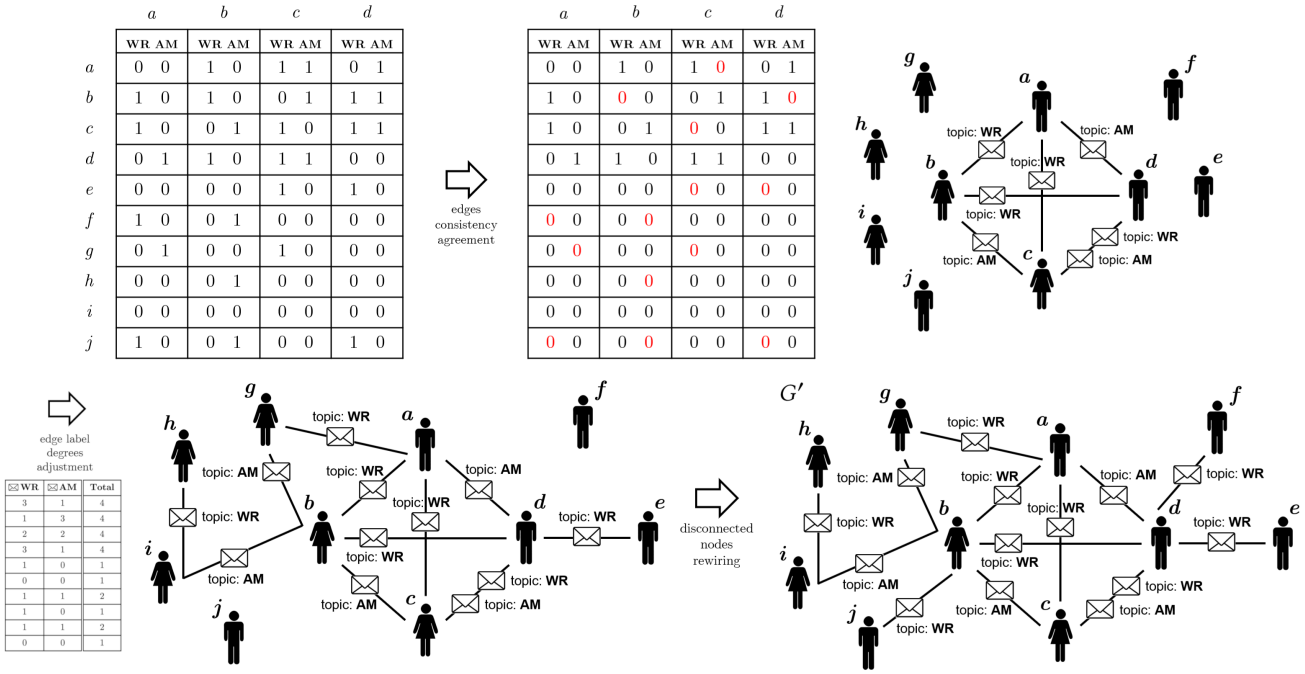

Figure 5: Post-Processing phase.

user's local graph. The local graph of a user comprises only the user node and its adjacent nodes and edges. This approach allows the data curator to use the user reports to reconstruct a graph that closely resembles the original one.

Once the users are partitioned and clustered, each user is supposed to encode its RANL according only to the users present in the clusters of the user's partition. Now, the new size of a user's RANL will be proportional to the size of the clusters of the user's partition. This solution avoids the addition of excessive bits flipped to one. Then, each user encodes its RANL according to their respective partition's clusters, randomizes it, and sends it to the data curator, which gathers all the users' RANLs to form a perturbed graph G'. The running example 3 illustrates this RANL reporting phase.

Example 3. Consider the edge-labeled graph G in Figure 1, the partitions $\mathcal{P}$ and clusters $\mathcal{C}$ in Figure 2, and their respective partition-cluster mapping in Figure 3. Figure 4 presents how the users of each partition build and report their RANLs through the RR protocol according to the clusters of the partitions where they belong. For example, consider the user $v_a \in V$ that belongs to the partition P_1. As the clusters of P_1 are $C_1 \cup C_2 = \{v_a, v_b, v_c, v_d\}$, the $RANL_{v_a}$ is formed only by regarding the connections between v_a and the elements of $C_1 \cup C_2$.

4.4 Graph Post-Processing

In this last phase, the data curator performs post-processing techniques over the perturbed graph G' to fix users' connection inconsistencies. The post-processing techniques are enumerated as follows: (i) *Edges Consistency Agreement*; (ii) *Edge Label Degrees Adjustment*; and (iii) *Disconnected Nodes Rewiring*.

4.4.1 Edges Consistency Agreement. We initiate this stage by removing the self-edges from G' that may have arisen in the users' perturbed RANLs. In this work, we assume that edge-labeled graphs do not have edges that connect a node to itself. The next step consists of validating whether an edge truly exists or not. For instance, consider two users $v_i, v_j \in V$ and an edge label $x_k \in X$. The edge $e_{i,j,k}$ is only considered to exist if $e_{i,j,k}$ is present in $RANL_{v_i}[e_{i,j,k}] = RANL_{v_j}[e_{j,i,k}] = 1$. Otherwise, if the edge is present only in one of these RANLs, the edge is removed from the released DP graph G'. This double-check is crucial for ensuring graph consistency and enhancing data utility, particularly in the context of undirected graphs. As we are dealing with undirected graphs, we must ensure that both related nodes have reported the existence of the same edge. Also, since the probability of keeping a truly bit one is higher than flipping a bit from zero to one, it is much more plausible that an edge only exists when it appears in the RANLs of both nodes.

4.4.2 Edge Label Degrees Adjustment. In this stage, we use the noisy edge label degrees obtained in the first phase of PEG (Section 4.1) to adjust the users' edge label degrees according to the noisy information. This adjustment is necessary because the RR protocol tends to add extra edges to the users' RANLs and, consequently, to the perturbed graph G'.

4.4.3 Disconnected Nodes Rewiring. It is important to note that some users may exhibit all edge label degrees as zero after reporting their data to the curator. This outcome is particularly expected in datasets with long-tailed degree distributions, where many users have degrees approaching zero. Consequently, certain users may have their degrees estimated as zero after sending it to the data curator. However, in practical scenarios, there are no disconnected nodes, i.e., each user is expected to have at least one connection. Thus, for each user $v_i \in V$ with all edge label degrees equal to zero, a random edge $e_{i,j,k}$ is added to G', ensuring that $v_i \neq v_j$ and the edge label $x_k \in X$ is sampled proportionally to the edge labels present in G'.

Figure 5 presents how the graph is created through the RANLs as well as how the post-processing step is applied.

4.5 The PEG Algorithm

The PEG algorithm[1], detailed in Algorithm 1, expects as input an edge-labeled graph $G = (V, E, X)$, the number of partitions and clusters, p and c, the y-th *percentile_value*, and the privacy budget ε. The output of PEG is a private version of G, given by $G' = (V, E', X)$.

In line 1, the privacy budget ε is split into ε_1 (clustering), ε_2 (partition-cluster mapping) and ε_3 (RANL reporting), such that $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 = \varepsilon$. In line 3, the set of partitions $\mathcal{P}$ are created, such that $|\mathcal{P}| = p$. Next, each user begins to report its edge label degrees through the geometric mechanism, with ε_1 and $\Delta Q = 2$. In lines 4-10, the data curator aggregates and estimates the users' noisy edge label degrees to build the set of clusters C, such that $|C| = c$. Afterward, the data curator adopts the partition and cluster information to perform the partition-cluster mapping. For each partition, each user reports its preferred cluster through the OUE protocol with ε_2 in lines 11-13. Subsequently, the data curator aggregates and estimates the cluster counts using a weighting function and percentile selection, as described in lines 14-18, to determine the partition's clusters. Then, in lines 19-22, each user builds its RANL according to the partition clusters to which it belongs and reports the RANL through the RR protocol with ε_3. Additionally, in lines 23-26, the data curator aggregates the users' RANLs to construct the edges of the DP graph. Additionally, it performs all the post-processing steps on these edges. Finally, the DP edge-labeled graph $G' = (V, E', X)$ is released in line 27.

4.6 Computational Cost

The computational cost of our approach is determined by the composition of the multiple phases of PEG. In the partitioning & clustering phase (Algorithm 1 – lines 3-10) it primarily involves partitioning the users V in the original graph G into a set of partitions $\mathcal{P}$. This phase also includes reporting and estimating the users' degrees, followed by building the set of clusters C based on their sorted node degrees. The sorting procedure is the most time-consuming step, which results in an expected time complexity of $O(|V| \cdot \log |V|)$.

The running time complexity of the partition-cluster mapping phase (Algorithm 1 – lines 11-18) is $O(|V| \cdot |C|)$ since each user has to indicate to which cluster it is more connected within the existing clusters. The RANL reporting phase (Algorithm 1 – lines 19-22) runs in $O(|V| \cdot |C_*| \cdot |X|)$ where $|C_*|$ is the size of the largest cluster and $|X|$ is the number of possible labels. Finally, the post-processing phase (Algorithm 1 – lines 23-26) complexity is given by $O(|V| \cdot |C_*| \cdot |X|)$. Therefore, the overall complexity of PEG is given by $O(|V| \cdot |C_*| \cdot |X|)$.

4.7 Privacy Analysis

The threat model for PEG considers that the adversary may possess background information about the edges and their labels and may use that information to infer private details from the released graph G'. PEG aims to ensure the adversary cannot determine with high confidence whether a specific edge (with its label) is present or absent. We accomplish this by employing local differential privacy and implementing adjustment steps. The adjustments made by the untrusted curator are considered post-processing steps, which still maintain the formal guarantees of LDP (Theorem 2.5) and consequently do not leak information.

[1]https://github.com/andreluiscm/peg-ldp

Algorithm 1: PEG

Input: Edge-labeled graph $G = (V, E, X)$, # partitions p, # clusters c, y-th percentile p_value, privacy budget ε
Output: Perturbed edge-labeled graph $G' = (V, E', X)$

1 $\varepsilon_1, \varepsilon_2, \varepsilon_3 \leftarrow$ SplitPrivacyBudget(ε);
2 $n \leftarrow |V|; t \leftarrow |X|$;
// Partitioning & Clustering phase
3 $\mathcal{P} \leftarrow$ BuildPartitions(V, p);
4 **for** $v_i \in V$ **do**
5 **for** $x_k \in X$ **do**
6 $\tilde{d}_{v_i}^{x_k} \leftarrow d_{v_i}^{x_k} + Geom(\frac{\varepsilon_1}{2})$;
7 **for** $x_k \in X$ **do**
8 $\tilde{d}^{x_k} \leftarrow$ AggregateEdgeLabelDegrees($\tilde{d}_{v_i}^{x_k}, \ldots, \tilde{d}_{v_n}^{x_k}$);
9 $\bar{\phi} \leftarrow$ EstimateNodeDegrees($\bar{d}^{x_1}, \ldots, \bar{d}^{x_t}$);
10 $C \leftarrow$ BuildClusters($\bar{\phi}, c$);
// Partition-Cluster Mapping phase
11 **for** $P_j \in \mathcal{P}$ **do**
12 **for** $v_i \in P_{j\,elems}$ **do**
13 $\tilde{clus}_{v_i}^{P_j} \leftarrow$ OUE_Protocol(v_i, C, ε_2);
14 $\tilde{clus}^{P_j} \leftarrow$ AggregateClusters($\tilde{clus}_{v_i}^{P_j}$ **for** $v_i \in P_{j\,elems}$);
15 $\bar{clus}^{P_j} \leftarrow$ EstimateClusters($\tilde{clus}^{P_j}$);
16 $\bar{clus}^{P_j} \leftarrow$ WeighClusters($\bar{clus}^{P_j}$);
17 $\bar{clus}^{P_j} \leftarrow$ GetTopPercentile($\bar{clus}^{P_j}, p\_value$);
18 $P_{j\,clus} \leftarrow \bar{clus}^{P_j}$;
// RANL Reporting phase
19 **for** $P_j \in \mathcal{P}$ **do**
20 **for** $v_i \in P_{j\,elems}$ **do**
21 $RANL_{v_i} \leftarrow$ BuildRANL($v_i, P_{j\,clus}$);
22 $\tilde{RANL}_{v_i} \leftarrow$ RR_Protocol($RANL_{v_i}, \varepsilon_3$);
// Post-Processing phase
23 $E' \leftarrow$ AggregateRANLs($\tilde{RANL}_{v_i}, \ldots, \tilde{RANL}_{v_n}$);
24 $E' \leftarrow$ AdjustEdgesConsistency(E');
25 $E' \leftarrow$ AdjustEdgeLabelDegrees($E', \bar{d}^{x_1}, \ldots, \bar{d}^{x_t}$);
26 $E' \leftarrow$ AdjustDisconnectedNodes($E', \bar{d}^{x_1}, \ldots, \bar{d}^{x_t}$);
27 **return** $G' = (V, E', X)$;

As previously mentioned, PEG is divided into four main phases. Not all of them (such as partitioning and post-processing steps) consume a privacy budget. The partitioning step utilizes the known number of users n, which is considered public information. On the other hand, the post-processing steps modify the released graph G' and do not compromise its privacy.

The remaining steps of PEG require privacy protection, as users must send their data privately via LDP mechanisms. The clustering step uses ε_1 to report users' edge label degrees through the geometric mechanism. Although each user submits multiple reports, one per edge label, these are performed in parallel since the degrees of different edge labels are independent, satisfying ε_1-edge-LDP. In the partition-cluster mapping, users in each partition report their cluster membership via the OUE protocol, using a privacy budget ε_2. As each user reports only once, this step consumes ε_2 and satisfies ε_2-edge-LDP. In the RANL reporting step, each user sends its RANL through the RR protocol, which also requires a privacy budget ε_3. As each user reports this information just once, this step consumes only ε_3 and satisfies ε_3-edge-LDP. Finally, by the sequential composition of DP, we can state that PEG satisfies ε-edge-LDP, where $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3$.

5 EXPERIMENTAL EVALUATION

In this section, we empirically evaluate the effectiveness of PEG on four real-world edge-labeled graphs. The experiments were conducted in Linux 64-bit, Intel(R) Core(TM) i7-7820X CPU and 128GB RAM. We implemented PEG in Python with the graph-tool [32] and Gurobi [22] packages. We repeated each experiment 10 times for each dataset and reported the average results. The number of partitions p and the number of clusters c were defined according to some known heuristics, as there is no immediate solution for determining optimal values for these parameters. We set $p = \max(1, \lfloor \frac{n}{1{,}000} \rfloor)$ since we need a reasonable amount of users to perform a good estimation of the reported values [16]. In turn, we used a similar heuristic [25] to set $c = \lfloor \sqrt[3]{n} \rfloor$, which states that the highest users' degree within a graph will not exceed $\lfloor \sqrt[3]{n} \rfloor$. Then, we assume that in our clustering scenario, the extreme case consists of a user with connections with users of every other cluster.

We set the y-th percentile to the 70th since it is suitable for gathering the most representative nodes of the graph. We varied the privacy budget ε in the experiments from 0.1 to 1.0, aligning with the range commonly used in other studies in this field, which assures a significant level of privacy. We argue that choosing the most adequate ε for an application is a challenging task that demands efforts from several experts [7] and is out of the scope of this work. Furthermore, as PEG is a multiphase algorithm, we had to split the privacy budget among the phases that use private mechanisms to ensure that the overall privacy constraint is not violated. We set the allocation to $[20\%, 20\%, 60\%]$ of the privacy budget ε to the *clustering* (ε_1), *partition-cluster mapping* (ε_2), and *RANL reporting* (ε_3) phases, respectively. The reason for giving more budget to the RANL reporting phase relies on the fact that the transmitted information of this phase is more sensitive to smaller privacy budgets.

5.1 Datasets

We conducted experiments over four real-world undirected edge-labeled network datasets from different domains and characteristics. *DBLP* [2] and *Netscience* [3] (NS) are co-authorship datasets, while *Yeast Landscape* [3] (YL) and *Pierre Auger* [3] (PA) are genetic datasets. Table 1 summarizes their characteristics. "ELP" is the abbreviation for edge label proportions, detailed in Section 5.3.2.

5.2 Baselines

To the best of our knowledge, no prior work exists on the DP release of entire edge-labeled graphs. Therefore, we compare our approach with three other methods based on PEG. We propose the following baselines: (i) RANL-random, (ii) RANL-consensus, and (iii) PEG-random. We did not compare PEG with PrivAG [31] as this approach is based on another privacy definition, denoted *attribute-wise LDP*, and also does not release the entire graph, only a few graph statistics.

RANL-random and RANL-consensus are similar approaches, which build a perturbed graph based only on the reported users' RANLs. In these approaches, the whole privacy budget is used to report the h-length RANL of each user, where $h = |V| \cdot |X|$. They differ only in the edges consistency agreement post-processing step. The RANL-consensus uses the same idea as PEG, while RANL-random chooses randomly from which RANL the edge information is true. For example, consider two users $v_i, v_j \in V$ and their respective RANLs, given by $RANL_{v_i}$ and $RANL_{v_j}$, such that both users have reported their connections between each other in their respective RANLs. Then, in the RANL-random approach, the true connection information between v_i and v_j has come from any of their RANLs. The PEG-random approach is quite similar to PEG, but it is different because it does not consider any degree information. Thus, the clustering is made randomly, the same way as in the partitioning. Finally, since there are only two private phases in PEG-random, the privacy budget ε allocation is split into 20% and 80% of ε for the partition-cluster mapping and RANL reporting phases, respectively. Also, only the cluster with the highest count (top 1) is chosen to be the cluster of the partition. Note that there are no degree adjustments in any of these baselines.

[2] https://github.com/supriya-gdptl/HCODA/tree/master/data
[3] https://manliodedomenico.com/data.php

Table 1: Characteristics of the edge-labeled graph datasets.

	DBLP	NS	YL	PA
# Nodes	41,427	14,065	4,458	514
# Edges	124,214	59,026	8,450,408	7,153
# Edge Labels	4	13	4	16
Degree$_{avg}$	5.99	8.39	3,791.12	27.83
Degree$_{max}$	358	361	5,044	123
St. Deviation$_{ELP}$	0.13	0.07	0.24	0.18

5.3 Utility Analysis

In this section, we conduct various analyses to evaluate the effectiveness of the graphs released by PEG in terms of utility. For the following analyses, consider an input graph $G = (V, E, X)$ and its perturbed version $G' = (V, E', X)$ produced by PEG.

5.3.1 Degree Distribution. To evaluate how well G' – the released graph – captures the degree distribution of the original edge-labeled graph G, we applied the Kolmogorov-Smirnov (KS) statistic, which quantifies the maximum distance between two-degree distributions. Let Cum_{SD} and $Cum_{\tilde{SD}}$ denote the cumulative distribution functions estimated from the sorted degrees of the G and G', respectively. Then, the KS$(SD, \tilde{SD})$ can be calculated according to Equation 10. The lower the KS statistic, the higher the data utility.

$$\text{KS}(SD, \tilde{SD}) = max_d |Cum_{SD}(d) - Cum_{\tilde{SD}}(d)| \quad (10)$$

Figure 6 shows the results for the degree distribution. We can observe that PEG outperforms all baselines in almost all datasets. This behavior is comprehensive since PEG is the only approach with an additional degree correction step. The exception occurs in the Netscience and Pierre Auger datasets, where PEG-random surpasses PEG when $\varepsilon = 0.1$. In Netscience, it occurs because this dataset is a sparse graph composed of 13 possible edge labels, and it has an average degree smaller than the number of edge labels. Thus, many edge-label degrees that are originally zero are estimated to different values after being perturbed, harming the graph's degree distribution. Also, many non-zero edge-label degrees have low values that become zero after being perturbed. Although the Pierre Auger dataset also has many edge labels (16 in total), this graph is much denser than Netscience, making it less sensitive to the estimation of noisy degrees. However, it still suffers from estimating the degree distribution under small ε values due to the reduced number of nodes.

5.3.2 Edge Label Proportions. To evaluate how well the edge label proportions (ELP) of G are being maintained in G', we

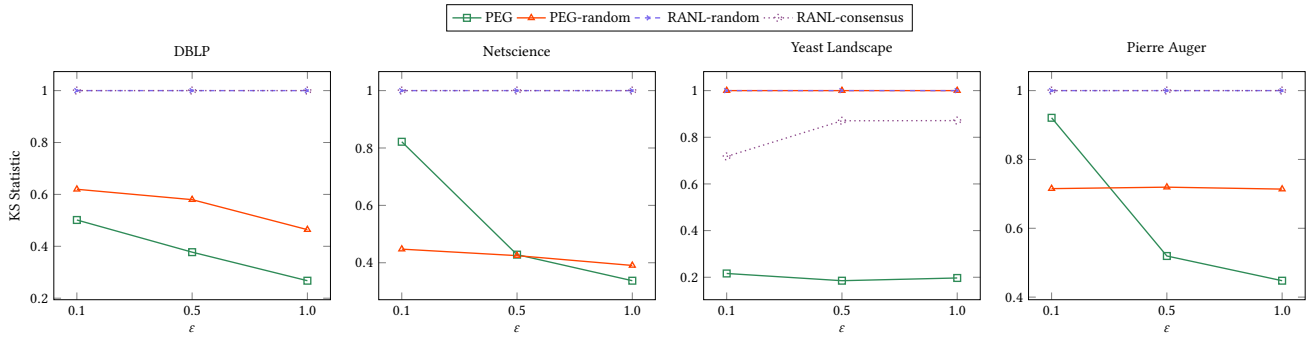

Figure 6: Kolmogorov-Smirnov (KS) comparison of PEG and baseline approaches for the degree distribution analysis after 10 runs, where $\varepsilon \in \{0.1, 0.5, 1.0\}$. The x-axis is the ε, while the y-axis is the KS statistic. The lower the KS statistic, the higher the data utility.

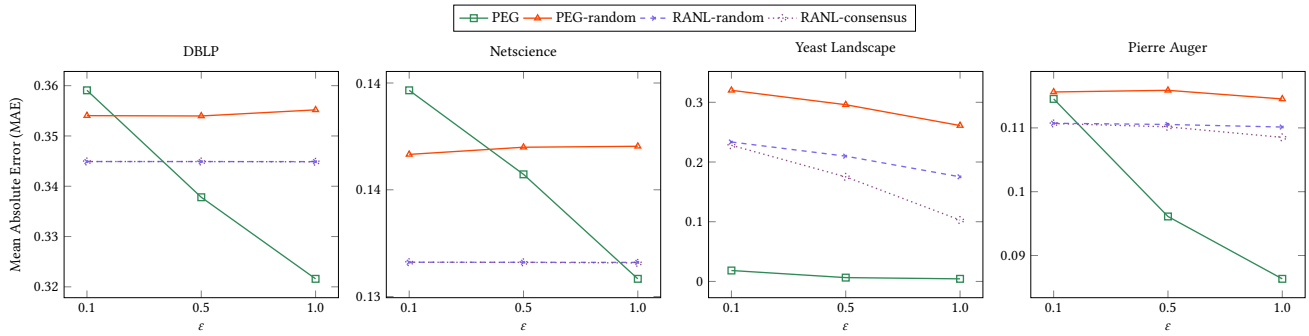

Figure 7: Mean absolute error (MAE) comparison of PEG and baseline approaches for the edge label proportions analysis after 10 runs, where $\varepsilon \in \{0.1, 0.5, 1.0\}$. The x-axis is the ε, while the y-axis is the MAE. The lower the MAE, the higher the data utility.

measured the Mean Absolute Error (MAE) of the edge label proportions between G and G'. Let $prop_{v_i}^{G} = (prop_{v_i}^{x_1}, \ldots, prop_{v_i}^{x_k})$ be the edge label proportions of the node $v_i \in V$, such that $prop_{v_i}^{x_k}$ denotes the proportion of adjacent edges of v_i associated with the label $x_k \in X$. The proportion is calculated by dividing the number of adjacent edges associated with x_k by the degree of v_i. Then, we calculate the $\text{ELP}_{MAE}(G, G')$ according to Equation 11. The lower the ELP_{MAE}, the higher the data utility.

$$\text{ELP}_{MAE}(G, G') = \frac{\sum_{v_i \in V} \frac{||prop_{v_i}^{G} - prop_{v_i}^{G'}||_1}{|X|}}{|V|} \tag{11}$$

Figure 7 presents the results for edge label proportions, showing that PEG outperforms almost all baselines. Similar to the degree distribution analysis, the exception is the Netscience dataset due to its low standard deviation in edge label proportions and its sparsity, which makes it harder to maintain accurate edge label degrees. For the other datasets, there is more information available to improve edge label proportions (ELP). Although the DBLP dataset is sparse, it has only 4 edge labels, a dominant label, and a large number of nodes, allowing for more accurate results. The Yeast Landscape and Pierre Auger datasets benefit from a higher density, a clear dominant edge label, and a significantly higher average degree relative to the number of edge labels.

5.3.3 Number of Edges. This analysis is extremely useful for evaluating whether the released graph G' maintains the magnitude of the edges of the input graph G. For this purpose, we measured the Mean Relative Error (MRE) of the number of edges (NE) between G and G'. We define $\text{NE}_{MRE}(G, G')$ according to Equation 12, where $|E(G)|$ and $|E(G')|$ denote the number of edges in G and G', respectively. The lower the NE_{MRE}, the higher the data utility.

$$\text{NE}_{MRE}(G, G') = \frac{||E(G)| - |E(G')||}{|E(G)|} \tag{12}$$

Figure 8 shows the results for the number of edges. PEG outperforms all baselines on datasets with a small number of edge labels due to its post-processing of perturbed graphs based on users' noisy edge label degrees. However, for datasets like Netscience and Pierre Auger with more edge labels, the noise significantly affects users' edge label degrees, causing inaccuracies when these degrees are adjusted during post-processing. In contrast, PEG-random, despite lacking an edge label degree adjustment step, performs better on these datasets (when $\varepsilon = 0.1$) due to its clustering step, which reduces the length of the RANL. This reduction leads to fewer noisy edges being added, resulting in a perturbed graph with a more accurate number of edges. Meanwhile, RANL-random and RANL-consensus baselines are hindered by the length of the RANL, where $h = |V| \cdot |X|$, causing excessive noise.

5.3.4 Graph Similarity. This analysis consists of a general metric that compares two graphs with different edges and labels. We applied the Jaccard Similarity (JS), which quantifies how similar two graphs are regarding their corresponding connections. We define $\text{JS}(G, G')$ according to Equation 13, where $E(G)$ and $E(G')$ denote the set of edges in G and G', respectively. The

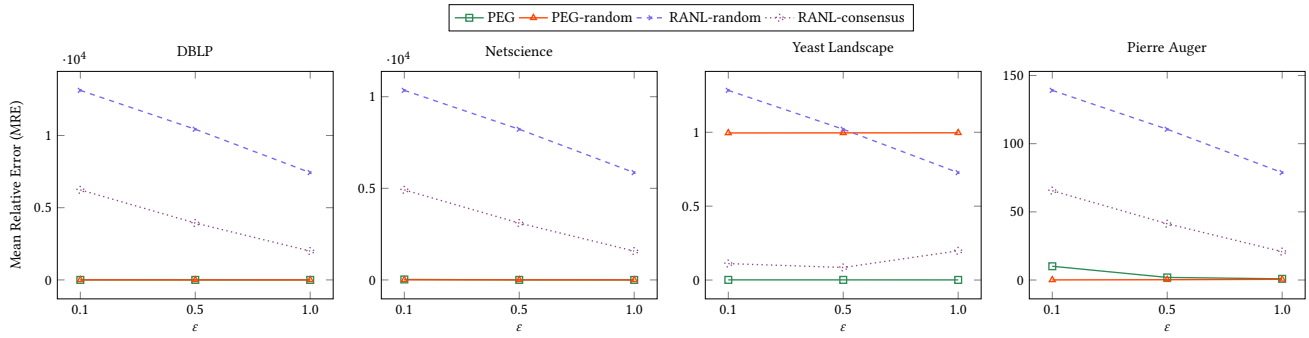

Figure 8: Mean relative error (MRE) comparison of PEG and baseline approaches for the number of edges analysis after 10 runs, where $\varepsilon \in \{0.1, 0.5, 1.0\}$. The x-axis is the ε, while the y-axis is the MRE. The lower the MRE, the higher the data utility.

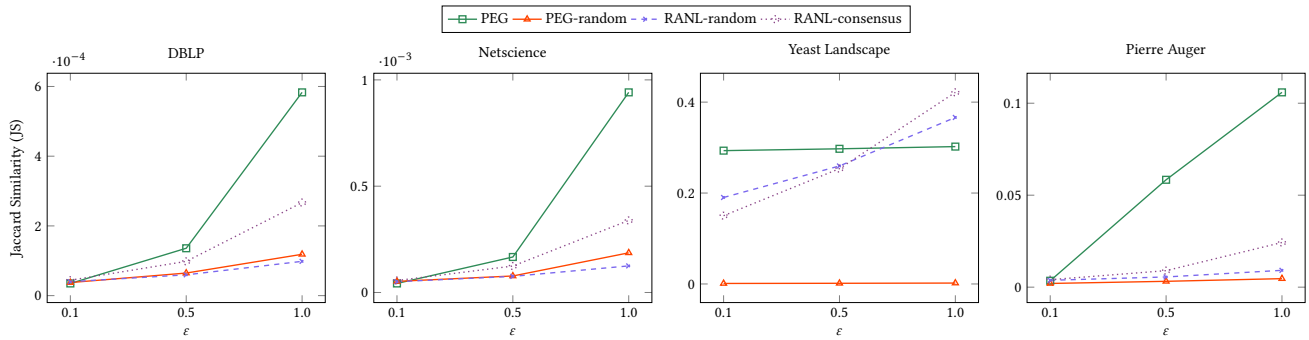

Figure 9: Jaccard similarity (JS) comparison of PEG and baseline approaches for the graph similarity analysis after 10 runs, where $\varepsilon \in \{0.1, 0.5, 1.0\}$. The x-axis is the ε, while the y-axis is the JS. The higher the JS, the higher the data utility.

higher the JS, the higher the similarity and, consequently, the data utility.

$$\mathrm{JS}(G, G') = \frac{|E(G) \cap E(G')|}{|E(G) \cup E(G')|} \tag{13}$$

Figure 9 shows that PEG outperforms all baselines on most datasets. The exception is the Yeast Landscape graph, where PEG falls behind RANL-random and RANL-consensus at $\varepsilon = 1.0$. This occurs because the Yeast Landscape consists of a dense graph with a high average degree, and PEG's clustering phase may exclude many inter-cluster edges by only selecting clusters above the 70th percentile. In contrast, RANL-random and RANL-consensus perform better as ε increases since the dense graph structure means that most of the RANLs content consists of true edges, leaving little room for false edges. For less dense datasets, PEG excels by reducing the RANL length during clustering, minimizing false edges, and better preserving original edges compared to the baselines.

5.3.5 Community Similarity. In graph analytics, communities are extremely relevant since they help us understand network complexities. To evaluate the community similarity between G and G', we define an optimization function that maximizes the number of nodes in G and G' that belong to the same communities. The motivation for using an optimization function relies on the fact that (i) G and G' may have a different number of communities and (ii) let $CM^{G}_{v_i}$ and $CM^{G'}_{v_i}$ denote the label of the community assigned to the user $v_i \in V$ in G and G', respectively, there are no guarantees that v_i remained in the same community, even though $CM^{G}_{v_i} = CM^{G'}_{v_i}$. It may happen since the communities in G and G' may be labeled differently.

However, community detection algorithms were not originally designed for edge-labeled multigraphs. These algorithms expect a graph with node attributes or one attributed edge. We then redesigned our graphs so that each edge has a weight. Let $prop^{G} = (prop_{x_1}, \ldots, prop_{x_k})^{|X|}$ be the edge label proportions of G, such that $prop_{x_k}$ denotes the proportion of edges associated with the label $x_k \in X$ in G. Also, let $conn^{G}_{v_i,v_j} = (conn^{x_1}_{v_i,v_j}, \ldots, conn^{x_k}_{v_i,v_j})^{|X|}$ be the connection intentions of v_i, $v_j \in V$ in G, such that $conn^{x_k}_{v_i,v_j} = 1$ if the edge $e_{i,j,k} \in E$ in G, and 0 otherwise. Thus, we can define $weight^{G}_{e_{i,j}} = \sum(prop^{G} \odot conn^{G}_{v_i,v_j})$ to assign a weight to each edge $e_{i,j}$, where $e_{i,j}$ refers to the connection between nodes v_i and v_j.

Once each pair $e_{i,j}$ has been assigned with their corresponding weights, we apply the stochastic block model [1] to find graph communities. This process of weighting and discovering the communities is repeated for the graphs G and its private version G'. However, as mentioned above, there are no guarantees that the number of communities of G and G' is the same. For this purpose, we model an optimization function to maximize the number of nodes that belong to the same community in G and G' simultaneously.

Given the constraints of our problem, we modeled it as a variant of the assignment problem [28], where the aim is to optimally match workers to tasks, ensuring each worker is assigned to exactly one task and vice versa while minimizing or maximizing the objective function. In our context, the communities of G represent the workers, and the communities of G' represent the tasks.

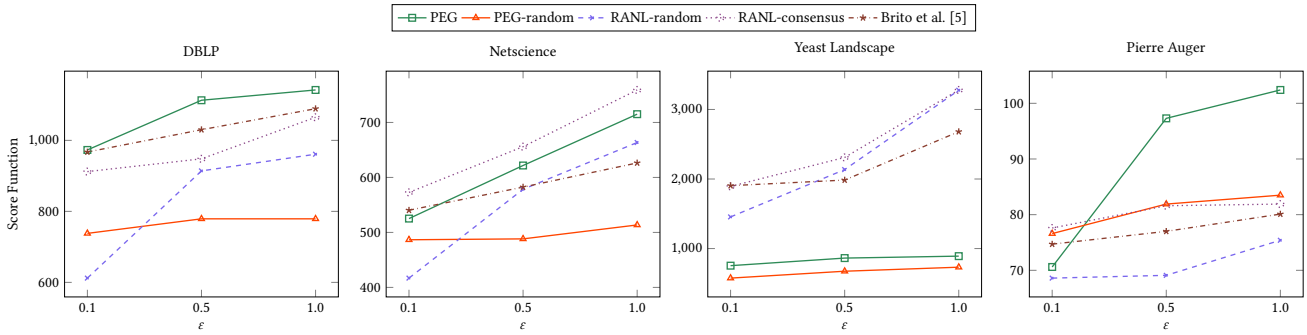

Figure 10: Score function comparison of PEG and baseline approaches for the community similarity analysis after 10 runs, where $\varepsilon \in \{0.1, 0.5, 1.0\}$. The x-axis is the ε, while the y-axis is the score. The higher the score, the higher the data utility.

$$\textbf{maximize} \quad Z = \sum_{i=1}^{z} \sum_{j=1}^{\tilde{z}} score_{i,j} \cdot x_{i,j}$$

$$\textbf{s.t.} \quad (1) \sum_{i=1}^{z} x_{i,j} = 1 \quad \forall_{j \leq \tilde{z}} \tag{14}$$

$$(2) \sum_{j=1}^{\tilde{z}} x_{i,j} = 1 \quad \forall_{i \leq z}$$

Our goal is to determine which pair of communities of G and G' maximize the objective function Z in Equation 14. We denote z and $\tilde{z}$ as the number of communities of G and G', respectively. In addition, $score_{i,j}$ refers to the number of nodes that belong to the i-th and j-th communities of G and G', simultaneously, given by $CM_{s_i}^{G}$ and $CM_{s_j}^{G'}$, respectively. Since $CM_{s_i}^{G}$ and $CM_{s_j}^{G'}$ are subsets of V, we applied an adapted Jaccard Similarity (JS) to calculate the $score_{i,j} = |CM_{s_i}^{G} \cap CM_{s_j}^{G'}|$. The higher the Z, the higher the similarity and, consequently, the data utility.

Since we transform edge-labeled graphs into edge-weighted graphs for community detection, we also incorporate the local approach by Brito et al. [5] as a baseline, which is designed to release edge-weighted graphs with LDP guarantees. To handle real-number weights, we adapted this baseline approach to utilize the Laplace mechanism instead of the geometric mechanism. Figure 10 shows the results for the community similarity. We can observe that PEG does not dominate the baselines in all datasets. It happens due to the particularities of some datasets. For the Netscience dataset, PEG slightly loses for the RANL-consensus since this dataset is very sparse and also has an average degree smaller than the number of edge labels. This characteristic leads PEG to query edge label degrees inaccurately, impacting the adjustment step and forming unexpected communities. In contrast, PEG is significantly affected by the dense Yeast Landscape dataset. Despite accurate querying, the clustering phase may lose many original connections, which are then randomly rewired based on degrees.

5.4 Summary

Our results consistently demonstrate that PEG introduces less noise compared to the baseline approaches in most of the presented scenarios, resulting in higher data utility and more accurate analyses. This can be evidenced mainly in datasets with a long-tailed degree distribution. This occurs due to the clustering step that reduces the dimensionality of the RANLs, which helps to maintain the true edges while reducing the noise injection that produces noisy edges.

PEG occasionally underperforms in certain analyses, especially with dense graphs like the Yeast Landscape. To enhance the performance in these graphs, one potential solution consists of adjusting the number of clusters c and the y-th percentile value used in the analyses. The higher c and y in a dense graph, the more clusters tend to be excluded from the partition-cluster mapping phase. Consequently, the reported RANL of individuals in the partition may contain less information, as it will not report connections with individuals in the non-selected clusters. Another key point is that datasets with varying characteristics may require different privacy budget allocations across PEG's phases. Specifically, for dense graphs, allocating more privacy budget to the RANL reporting phase seems more appropriate, as it is expected to maintain more information across the user's RANL and, consequently, in the entire released graph.

6 CONCLUSION AND FUTURE WORK

In this paper, we addressed the problem of releasing edge-labeled graphs under local differential privacy guarantees. Initially, we proposed the Randomized Attribute Neighbor List, which complies with edge-LDP to report the users' local edge-labeled graphs. Then, we developed PEG, a novel decentralized dynamic degree-based clustering approach designed for privately releasing edge-labeled graphs under the notion of edge-LDP. Additionally, we have improved the accuracy of the proposed approach by adopting several post-processing techniques to tune the released graph structure according to heuristics present in real-world applications. Our experiments demonstrated through an extensive evaluation that our approach outperforms the baselines in almost all presented scenarios. In future work, we plan to extend PEG to more complex applications such as anomaly detection, mobility analysis, and link prediction in real-world edge-labeled graphs. Additionally, we would extend the notion of node-LDP to edge-labeled graphs since it is a stronger notion of privacy in the graph context. Finally, we aim to explore combining PEG with other privacy-preserving techniques, such as federated learning or secure multi-party computation, to enhance both privacy and performance in distributed environments.

ACKNOWLEDGEMENTS

This work was supported by CAPES/Brazil under grant number 88882.454571/2019-01 and by CNPq/Brazil under grant number 316729/2021-3.

REFERENCES

[1] Emmanuel Abbe. 2018. Community detection and stochastic block models: recent developments. *Journal of Machine Learning Research* 18, 177 (2018), 1–86.

[2] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. 2019. Hadamard response: Estimating distributions privately, efficiently, and with little communication. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1120–1129.

[3] Izzat Alsmadi and Ikdam Alhami. 2015. Clustering and classification of email contents. *Journal of King Saud University-Computer and Information Sciences* 27, 1 (2015), 46–57.

[4] Raef Bassily and Adam Smith. 2015. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 127–135.

[5] Felipe T Brito, Victor AE Farias, Cheryl Flynn, Subhabrata Majumdar, Javam C Machado, and Divesh Srivastava. 2023. Global and Local Differentially Private Release of Count-Weighted Graphs. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–25.

[6] Felipe T. Brito, André L. C. Mendonça, and Javam C. Machado. 2024. A Differentially Private Guide for Graph Analytics. In *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy*. 850–853.

[7] United States Census Bureau. 2021. Census Bureau Sets Key Parameters to Protect Privacy in 2020 Census Results. https://www.census.gov/newsroom/press-releases/2021/2020-census-key-parameters.html. [Online; accessed 08 May 2024].

[8] Lang Chen, Kai Han, Qing Xiu, and Dazheng Gao. 2022. Graph clustering under weight-differential privacy. In *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 1457–1464.

[9] Xihui Chen, Sjouke Mauw, and Yunior Ramírez-Cruz. 2019. Publishing community-preserving attributed social graphs with a differential privacy guarantee. *arXiv preprint arXiv:1909.00280* (2019).

[10] Robin Christensen. 2020. An Analysis of Notions of Differential Privacy for Edge-Labeled Graphs.

[11] José S Costa Filho and Javam C Machado. 2023. FELIP: A local Differentially Private approach to frequency estimation on multidimensional datasets. (2023).

[12] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 429–438.

[13] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.

[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 265–284.

[15] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[16] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[17] Chenglin Fan and Ping Li. 2022. Distances release with differential privacy in tree and grid graph. In *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2190–2195.

[18] Nan Fu, Weiwei Ni, Lihe Hou, Dongyue Zhang, and Ruyu Zhang. 2024. Community detection in decentralized social networks with local differential privacy. *Information Sciences* 661 (2024), 120164.

[19] Nan Fu, Weiwei Ni, Sen Zhang, Lihe Hou, and Dongyue Zhang. 2023. GC-NLDP: A graph clustering algorithm with local differential privacy. *Computers & Security* 124 (2023), 102967.

[20] Tianchong Gao, Feng Li, Yu Chen, and XuKai Zou. 2018. Local differential privately anonymizing online social networks under hrg-based model. *IEEE Transactions on Computational Social Systems* 5, 4 (2018), 1009–1020.

[21] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. 2009. Universally utility-maximizing privacy mechanisms. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 351–360.

[22] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[23] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. 2009. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 169–178.

[24] Allen L Hu and Keith CC Chan. 2013. Utilizing both topological and attribute information for protein complex identification in PPI networks. *IEEE/ACM transactions on computational biology and bioinformatics* 10, 3 (2013), 780–792.

[25] Zach Jorgensen, Ting Yu, and Graham Cormode. 2016. Publishing attributed social graphs with formal privacy guarantees. In *Proceedings of the 2016 international conference on management of data*. 107–122.

[26] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*. Springer, 457–476.

[27] Mehdi Kaytoue, Marc Plantevit, Albrecht Zimmermann, Anes Bendimerad, and Céline Robardet. 2017. Exceptional contextual subgraph mining. *Machine Learning* 106 (2017), 1171–1211.

[28] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[29] Ling Li, Yuhai Zhao, Siqiang Luo, Guoren Wang, and Zhengkui Wang. 2023. Efficient Community Search in Edge-Attributed Graphs. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[30] Yang Li, Michael Purcell, Thierry Rakotoarivelo, David Smith, Thilina Ranbaduge, and Kee Siong Ng. 2023. Private graph data release: A survey. *Comput. Surveys* 55, 11 (2023), 1–39.

[31] Zichun Liu, Liusheng Huang, Hongli Xu, Wei Yang, and Shaowei Wang. 2020. PrivAG: Analyzing attributed graph data with local differential privacy. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 422–429.

[32] Tiago P. Peixoto. 2014. The graph-tool python library. *figshare* (2014). https://doi.org/10.6084/m9.figshare.1164194

[33] Rafael Pinot, Anne Morvan, Florian Yger, Cédric Gouy-Pailler, and Jamal Atif. 2018. Graph-based clustering under differential privacy. *arXiv preprint arXiv:1803.03831* (2018).

[34] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2017. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 425–438.

[35] Adam Sealfon. 2016. Shortest paths and distances with differential privacy. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 29–41.

[36] Neil Shah, Alex Beutel, Bryan Hooi, Leman Akoglu, Stephan Gunnemann, Disha Makhija, Mohit Kumar, and Christos Faloutsos. 2016. Edgecentric: Anomaly detection in edge-attributed networks. In *2016 IEEE 16Th international conference on data mining workshops (ICDMW)*. IEEE, 327–334.

[37] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2016. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 17–37.

[38] Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Hui Wang, and Ting Yu. 2019. Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 703–717.

[39] Chang-Dong Wang, Jian-Huang Lai, and S Yu Philip. 2013. NEIWalk: Community discovery in dynamic content-based networks. *IEEE transactions on knowledge and data engineering* 26, 7 (2013), 1734–1748.

[40] Dan Wang and Shigong Long. 2019. Boosting the accuracy of differentially private in weighted social networks. *Multimedia tools and applications* 78 (2019), 34801–34817.

[41] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*. 729–745.

[42] Yuye Wang, Jing Yang, and Jianpei Zhang. 2020. Differential privacy for weighted network based on probability model. *IEEE Access* 8 (2020), 80792–80800.

[43] Chengkun Wei, Shouling Ji, Changchang Liu, Wenzhi Chen, and Ting Wang. 2020. AsgLDP: collecting and generating decentralized attributed graphs with local differential privacy. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3239–3254.

[44] Min Ye and Alexander Barg. 2018. Optimal schemes for discrete distribution estimation under locally differential privacy. *IEEE Transactions on Information Theory* 64, 8 (2018), 5662–5676.

[45] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. 2020. LF-GDPR: A framework for estimating graph metrics with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2020), 4905–4920.

[46] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. 2020. Towards locally differentially private generic graph metric estimation. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1922–1925.

[47] Nannan Zhou, Shigong Long, Hai Liu, and Hai Liu. 2022. Structure–Attribute Social Network Graph Data Publishing Satisfying Differential Privacy. *Symmetry* 14, 12 (2022), 2531.