UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANDRÉ LUÍS DA COSTA MENDONÇA

PEG: LOCAL DIFFERENTIAL PRIVACY FOR EDGE-ATTRIBUTED GRAPHS

FORTALEZA

2024

ANDRÉ LUÍS DA COSTA MENDONÇA

PEG: LOCAL DIFFERENTIAL PRIVACY FOR EDGE-ATTRIBUTED GRAPHS

Thesis submitted to the Programa de Pós-graduação em Ciência da Computação of the Centro de Ciências of the Universidade Federal do Ceará, as a partial requirement for obtaining the title of PhD in Ciência da Computação. Concentration Area: Banco de Dados

Advisor: Prof. Dr. Javam de Castro Machado

FORTALEZA

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M494p Mendonça, André Luís da Costa.
PEG: Local Differential Privacy for Edge-Attributed Graphs / André Luís da Costa Mendonça. – 2024.
103 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2024.
Orientação: Prof. Dr. Javam de Castro Machado.

1. Local Differential Privacy. 2. Edge-Attributed Graphs. 3. Graph Analytics. I. Título.
CDD 005

ANDRÉ LUÍS DA COSTA MENDONÇA

PEG: LOCAL DIFFERENTIAL PRIVACY FOR EDGE-ATTRIBUTED GRAPHS

Thesis submitted to the Programa de Pós-graduação em Ciência da Computação of the Centro de Ciências of the Universidade Federal do Ceará, as a partial requirement for obtaining the title of PhD in Ciência da Computação. Concentration Area: Banco de Dados

Approved on: 23 August 2024

EXAMINATION BOARD

Prof. Dr. Javam de Castro Machado (Advisor)
Universidade Federal do Ceará (UFC)

Profa. Dra. Ana Karolinna Maia de Oliveira
Universidade Federal do Ceará (UFC)

Prof. Dr. José Maria da Silva Monteiro Filho
Universidade Federal do Ceará (UFC)

Prof. Dr. José Soares de Andrade Júnior
Universidade Federal do Ceará (UFC)

Dr. Felipe Timbó Brito
Laboratório de Sistemas e Bancos de Dados
(LSBD)

Prof. Dr. Sérgio Lifschitz
Pontifícia Universidade Católica do Rio de
Janeiro (PUC-Rio)

To my beloved parents, Antônio and Aparecida,

I dedicate this thesis.

ACKNOWLEDGEMENTS

Firstly, I thank God, whose constant presence in my life has been a beacon in moments of uncertainty. His strength and spiritual guidance have helped me to maintain faith and perseverance, even in the face of the most challenging times.

To my parents, Antônio and Aparecida, whose unconditional love, sacrifice, and support have been fundamental to achieving this dream. You have provided me with the necessary foundation to walk this path with confidence and determination.

To my girlfriend, Aline Fernandes, whose patience, understanding, and encouragement were essential throughout this journey. Your constant love and support gave me the strength to face every obstacle and keep moving forward.

To my entire family, especially my grandmother, Antônia, who always encouraged me to dedicate myself to my studies, always saying: ‘‘My son, study to be a doctor’’. Today, I wish I could tell you that I did it, and I dedicate this achievement to you, wherever you are.

To my advisor, Prof. Javam Machado, for his trust, support, guidance, and patience throughout this academic journey. His dedication to my academic and professional development was crucial to completing this work. I will be eternally grateful for having had the opportunity to learn and grow under his guidance.

To my friend Felipe Timbó, whose contribution was essential throughout this work. On numerous occasions, he acted as a true co-advisor, offering technical support, constant help, and valuable insights. His generosity in sharing knowledge and dedication throughout this process was fundamental to the development and conclusion of this research.

To my friends: Iago Chaves, Eduardo Rodrigues, Daniel Praciano, Malu Maia, Paulo Amora, Serafim Costa, Ítalo Abreu, Victor Farias, Tallita Silveira, Antônio José (AJ), Tiago Nascimento, João Almeida, Israel Vidal, Diogo Martins, Fernando Dione, Bruno Leal, Felipe Monteiro, and Leonardo Linhares, I extend my deepest gratitude. Your support was crucial throughout this journey. You provided me with moments of relaxation that made the whole process lighter and more bearable. In the toughest moments, when exhaustion and discouragement almost made me give up, you were the ones who encouraged me to stay on course. I am deeply grateful for every laugh, every piece of advice, and above all, for each of your friendships.

To all the collaborators of the Systems and Databases Laboratory (LSBD). The support, discussions, and collaboration of each of you were fundamental to the development of this work. The constant exchange of ideas, the valuable suggestions, and the cooperative

environment created in the laboratory greatly contributed to my academic and personal growth. I am deeply grateful to have had the opportunity to work with such a talented and dedicated group.

To all who, directly or indirectly, contributed to the completion of this challenge. Whether through academic support, encouragement, or even any gesture of friendship and understanding throughout this process, each of you played a crucial role in this journey.

To the Systems and Databases Laboratory (LSBD) for providing a prosperous work environment for the development of this thesis. The infrastructure provided was essential for conducting the research. Additionally, the financial support provided by the LSBD allowed me to fully dedicate myself to my work without additional concerns. I am immensely grateful for all the support and resources that contributed to the realization of this work.

To CAPES for the financial support under grant number 88882.454571/2019-01, which was fundamental for conducting this research.

AGRADECIMENTOS

Primeiramente, agradeço a Deus, cuja presença constante em minha vida foi um farol nos momentos de incerteza. Sua força e orientação espiritual me ajudaram a manter a fé e a perseverança, mesmo diante dos desafios mais difíceis.

Aos meus pais, Antônio e Aparecida, cujo amor incondicional, sacrifício e apoio foram fundamentais para a realização deste sonho. Vocês me proporcionaram as bases necessárias para trilhar este caminho com confiança e determinação.

À minha namorada, Aline Fernandes, cuja paciência, compreensão e incentivo foram essenciais ao longo desta jornada. Seu amor e apoio constante me deram forças para enfrentar cada obstáculo e continuar avançando.

À toda minha família, em especial à minha avó, Antônia, que em vida sempre me incentivou a dedicar-me aos estudos, sempre dizendo-me: ‘‘Meu filho, estude para ser doutor’’. Hoje, queria poder te dizer que eu consegui, e dedico esta conquista à senhora, onde quer que esteja.

Ao meu orientador, Prof. Javam Machado, pela sua confiança, apoio, orientação e paciência ao longo desta jornada acadêmica. Sua dedicação ao meu desenvolvimento acadêmico e profissional foi fundamental para a conclusão deste trabalho. Serei eternamente grato por ter tido a oportunidade de aprender e crescer sob sua orientação.

Ao meu amigo Felipe Timbó, cuja contribuição foi essencial durante toda a realização deste trabalho. Em diversas ocasiões, atuou como um verdadeiro coorientador, oferecendo suporte técnico, apoio constante, além de *insights* valiosos. Sua generosidade em compartilhar conhecimento e sua dedicação ao longo deste processo foram fundamentais para o desenvolvimento e conclusão desta pesquisa.

Aos meus amigos: Iago Chaves, Eduardo Rodrigues, Daniel Praciano, Malu Maia, Paulo Amora, Serafim Costa, Ítalo Abreu, Victor Farias, Tallita Silveira, Antônio José (AJ), Tiago Nascimento, João Almeida, Israel Vidal, Diogo Martins, Fernando Dione, Bruno Leal, Felipe Monteiro e Leonardo Linhares, deixo aqui minha mais profunda gratidão. O apoio de vocês foi fundamental ao longo desta jornada. Vocês me proporcionaram momentos de descontração que tornaram todo o processo mais leve e suportável. Nos momentos mais difíceis, quando a exaustão e o desânimo quase me fizeram desistir, foram vocês que me incentivaram a continuar firme no caminho. Sou profundamente grato por cada risada, cada conselho e, acima de tudo, pela amizade de cada um de vocês.

A todos os colaboradores do Laboratório de Sistemas de Bancos de Dados (LSBD). O apoio, as discussões e a colaboração de cada um de vocês foram fundamentais para o desenvolvimento deste trabalho. A troca constante de ideias, as sugestões valiosas e o ambiente de cooperação criado no laboratório contribuíram imensamente para o meu crescimento acadêmico e pessoal. Sou profundamente grato por ter tido a oportunidade de trabalhar com um grupo tão talentoso e dedicado.

A todos que, direta ou indiretamente, contribuíram para a conclusão deste desafio. Seja através de apoio acadêmico, incentivo ou até mesmo por qualquer gesto de amizade e compreensão ao longo deste processo, cada um de vocês desempenhou um papel crucial nesta jornada.

Ao Laboratório de Sistemas de Bancos de Dados (LSBD) por proporcionar um ambiente de trabalho propício ao desenvolvimento desta tese. A infraestrutura oferecida foi essencial para a realização das pesquisas. Além disso, o suporte financeiro fornecido pelo LSBD permitiu que eu me dedicasse integralmente ao meu trabalho, sem preocupações adicionais. Sou imensamente grato por todo o apoio e recursos que contribuíram para a concretização deste trabalho.

À CAPES pelo suporte financeiro concedido sob o número de processo 88882.454571/2019-01, o qual foi fundamental para a realização desta pesquisa.

‘‘Success is not the key to happiness. Happiness is the key to success. If you love what you are doing, you will be successful.’’

(Albert Schweitzer)

ABSTRACT

Edge-attributed graphs are a particular class of graphs designed to represent networks whose edge content indicates a relationship between two nodes. The study of edge-attributed graphs finds applications in diverse fields, such as anomaly detection, mobility analysis, and community search. Since edge-attributed 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 attributes associated with edges in attributed graphs. This paper introduces PEG, a novel approach designed to release edge-attributed graphs with local differential privacy guarantees. Combining partitioning and clustering techniques enables 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-attributed graphs, enabling subsequent computation of various graph analysis metrics with high utility, including applications in community detection.

Palavras-chave: local differential privacy; edge-attributed graphs; graph analytics.

RESUMO

Grafos com atributos nas arestas são uma classe particular de grafos projetados para representar redes nas quais o conteúdo das arestas indica um tipo de relacionamento entre dois nós. O estudo de grafos com atributos nas arestas encontra aplicações em diversos campos, como detecção de anomalias, análise de mobilidade e busca de comunidades. No entanto, como os grafos com atributos nas arestas geralmente contêm informações sensíveis, a preservação da privacidade ao liberar esse tipo de dado para análise de grafos torna-se uma questão importante. Nesse contexto, a privacidade diferencial local (PDL) emergiu como uma definição robusta para a liberação de dados sob garantias sólidas de privacidade. No entanto, as técnicas existentes de PDL para grafos na literatura se concentram principalmente em estruturas de grafos tradicionais, sem considerar os atributos associados às arestas em grafos com atributos. Neste trabalho, introduzimos o PEG, uma abordagem inovadora projetada para liberar grafos com atributos nas arestas com garantias de privacidade diferencial local. Combinando técnicas de particionamento e agrupamento, possibilitamos uma distribuição mais eficaz do ruído entre nós similares, preservando a estrutura e os relacionamentos inerentes dentro do grafo liberado. Experimentos extensivos em conjuntos de dados do mundo real mostram que o PEG pode liberar de forma eficaz grafos com atributos nas arestas que são úteis e privados, permitindo a subsequente computação de várias métricas de análise de grafos com alta utilidade, incluindo aplicações na detecção de comunidades.

Keywords: privacidade diferencial local; grafos com atributos nas arestas; análise de grafos.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CONTENTS

1 INTRODUCTION

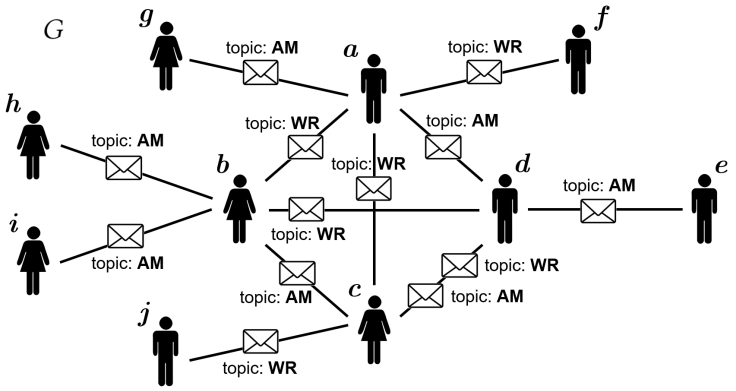
Graphs, also known as networks, are fundamental structures that represent entities and relationships between the entities. A graph consists of nodes (or vertices) and edges (or links) that connect pairs of nodes. They can be classified into several types based on their properties and the nature of the relationships. For instance, in undirected graphs, edges have no orientation, meaning the relationship is bidirectional. Conversely, edges have a direction in directed graphs (digraphs), indicating a one-way relationship. Furthermore, graphs can be weighted or unweighted, depending on whether the edges carry numerical values (weights) representing the relationships' strength or capacity. Graphs are used extensively in various fields due to their ability to model complex interactions and dependencies (BERGE, 2001).

Edge-attributed 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-attributed graphs have been widely adopted in many fields to explain why and how users make connections to each other. These graphs enable the modeling of complex relationships and interactions in various domains, providing a richer representation of the underlying data than traditional graphs. Examples include communication networks (WANG *et al.*, 2013), co-authorship networks (ALSMADI; ALHAMI, 2015), protein-protein networks (HU; CHAN, 2013), and heterogeneous information networks (SHI *et al.*, 2017).

The study of edge-attributed graphs has become a thriving research area, finding applications in many fields. For instance, edge-attributed graphs are employed in anomaly detection to identify unusual patterns and deviations within network data, which can indicate potential security breaches or system malfunctions (SHAH *et al.*, 2016). In mobility analysis, these graphs help to understand and predict movement patterns and behaviors in transportation networks and urban planning (KAYTOUE *et al.*, 2017). Finally, in community search, edge-attributed graphs enable the identification and exploration of connected groups or communities within larger networks, facilitating a better understanding of social structures and interactions (LI *et al.*, 2023).

Figure 1 illustrates an edge-attributed graph G, where the edge labels represent the topics of exchanged emails. In this example, graph G comprises ten nodes and twelve edges, with two distinct labels for the topics. The label ''AM'' indicates administrative matters, referring to messages related to managing and organizing business operations. On the other hand, the label ''WR'' denotes work-related topics, including messages such as project updates, meeting

Figure 1 – An edge-attributed G where nodes represent users and edges are emails exchanged among them. The topic of the emails represents the edge attribute. ‘‘AM’’ denotes administrative matters, and ‘‘WR’’ depicts work-related topics.



Source: Elaborated by the author.

requests, collaboration inquiries, and task assignments.

Due to the sensitive nature of the information found in edge-attributed 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 ‘‘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. This inference could lead to targeted attacks on user ‘‘g’’ for private information. Additionally, if other edges in the graph reveal patterns of communication or collaboration, an adversary might deduce further details about the organizational structure and relationships between users. Consequently, it is important to implement robust privacy-preserving techniques when sharing such sensitive graph data to prevent unintended disclosures and protect individuals' privacy.

In this context, differential privacy (DP) (DWORK, 2006) 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. This ensures that the mechanism's output does not significantly change when any single individual's data is added or removed from the dataset. DP achieves this by introducing a controlled amount of randomness, which effectively masks the contribution of any individual data point, making it difficult for an attacker to infer specific information about individuals. This trade-off between accuracy and privacy is carefully balanced

through parameters that dictate the level of noise added to the data.

In the literature, the primary DP setups are the global DP (DWORK, 2006) and the local DP (LDP) (DUCHI *et al.*, 2013). 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 exists who has indiscriminate access to the complete data and is responsible for releasing it after a differentially private procedure. In other words, the data curator is the entity responsible for ensuring privacy. 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, several studies have been developed over the years in the field of the differentially private release of graph data (BRITO *et al.*, 2024). Within the graph scope and following the definition of neighboring graphs, there are two main DP settings: the edge differential privacy (edge-DP) (HAY *et al.*, 2009) and the node differential privacy (node-DP) (KASIVISWANATHAN *et al.*, 2013). In the DP model, two datasets are neighbors if they differ in at most one single record. But, 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 attributed graphs, neither edge-DP nor node-DP privacy models are adequate since they ignore the presence of attributes in 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 (SEALFON, 2016; PINOT *et al.*, 2018; WANG; LONG, 2019; CHEN *et al.*, 2022; FAN; LI, 2022; BRITO *et al.*, 2023). However, these works often face limitations when dealing with non-numeric attributes. Some studies (JORGENSEN *et al.*, 2016; CHEN *et al.*, 2020; WEI *et al.*, 2020; ZHOU *et al.*, 2022) 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, Lie et al. (LIU *et al.*, 2020) proposes a method that specifically addresses local differential privacy for edge-attributed graphs with non-numerical attributes. However, it only provides privacy for a few statistics rather than releasing the entire attributed graph for comprehensive graph analytics.

1.1 Problem Statement

Given a network, represented as an edge-attributed graph, connected through edges that store information about the relationship of the related nodes, our problem is defined as ‘‘*how to privately release a meaningful edge-attributed graph while preserving as much of the original graph structure as possible?*’’.

Consider an undirected edge-attributed graph $G = (V, E, X)$ representing a network, where V is the set of nodes (or users), E is the set of edges that means the relationships between nodes from V, and X is the set of the possible edge attributes. We aim to release an edge-attributed graph $G' = (V, E', X)$, such that G' denotes a perturbed version of G, which is composed of E', a perturbed version of E, ensuring local DP guarantees and maintaining as many characteristics of the original graph structure as possible.

In this problem, we have two main characteristics: (i) the number of nodes is publicly known, and (ii) the edges that represent the connections between nodes are attributed, i.e., store some information about its relationship. Then, the information that must be protected is the edges and their attributes. In terms of utility, many useful metrics, such as the node degree distribution, number of edges, edge attribute proportions, graph similarity, and community similarity, should perform in the private graph similarly to the original graph. In particular, we measure the Kolmogorov–Smirnov divergence, the mean absolute error (MAE), the mean relative error (MRE), the Jaccard similarity, and a maximization objective function to evaluate the error introduced by our approach.

1.2 Hypothesis

Given an edge-attribute graph, there is an LDP approach that is able to publish a perturbed version of this graph without adding an amount of noise that makes the published data useless. Moreover, there are protocols associated with this DP setting that have low variance, i.e., they achieve better data utility levels.

1.3 Contributions

To address the mentioned concerns, we propose PEG (**P**rivacy for **E**dge-attributed **G**raphs), an approach for releasing entire edge-attributed graphs under local differential privacy guarantees. In summary, the main contributions of this work are as follows:

1. We first introduce the Randomized Attribute Neighbor List (RANL), a novel data structure for encoding edge-attribute 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 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-attributed graphs to evaluate the performance of PEG. We show that our approach achieves high utility for a variety of graph analysis metrics on the released graph, including applications in community detection.

The contributions of this thesis resulted in the submission of the following paper, which is still under revision:

- MENDONÇA, A. L. C.; BRITO, F. T.; MACHADO, J. C. **PEG: Local Differential Privacy for Edge-Labeled Graphs**. 2024. Submitted for publication to the International Conference on Extending Database Technology (EDBT, 2025).

Additionally, we highlight side contributions, which were developed and disseminated at various conferences throughout this Ph.D. These contributions significantly broadened the scope and impact of our research:

- MENDONÇA, A. L.; BRITO, F. T.; MACHADO, J. C. Análise de dados privada em redes sociais. **Jornadas de Atualização em Informática**, 2024.
- BRITO, F. T.; MENDONÇA, A. L. C.; MACHADO, J. C. A differentially private guide for graph analytics. In: **Proceedings 27th International Conference on Extending Database Technology (EDBT)**. Paestum, Italy: OpenProceedings.org, 2024. p. 850–853.
- MENDONÇA, A. L.; BRITO, F. T.; MACHADO, J. C. Privacy-preserving techniques for social network analysis. In: **Anais Estendidos do XXXVIII Simpósio Brasileiro de Bancos de Dados**. Belo Horizonte, MG, Brazil: SBC, 2023. p. 174–178.
- VIDAL, I. C.; MENDONÇA, A. L.; ROUSSEAU, F.; MACHADO, J. de C. Protecting: An application of local differential privacy for iot at the edge in smart home scenarios. In: **XXXVIII Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)**. Rio de Janeiro, RJ, Brazil: SBC, 2020. p. 547–560.
- NETO, E. R. D.; MACHADO, J. C.; MENDONÇA, A. L. Privlbs: Preserving privacy in

location based services. **Journal of Information and Data Management**, v. 10, n. 2, p. 81–96, 2019.

- NETO, E. R. D.; MENDONÇA, A. L. C.; BRITO, F. T.; MACHADO, J. C. Privlbs: uma abordagem para preservação de privacidade de dados em serviços baseados em localização. In: **XXXIII Simpósio Brasileiro de Banco de Dados (SBBD)**. Rio de Janeiro, RJ, Brazil: SBC, 2018. p. 109–120.

1.4 Organization

The rest of this document is described as follows. Chapter 2 presents an overview of several concepts to help understand this work, such as some graph concepts and the properties and settings of DP. Chapter 3 presents some studies under the edge-DP model and its variations to address the problem of releasing general graphs and attributed graphs under DP guarantees. We then detail the RANL data structure and present our approach PEG in Chapter 4. Furthermore, Chapter 5 shows the experimental results. Finally, Chapter 6 concludes our work by presenting the conclusion and future research directions.

2 THEORETICAL BACKGROUND

In this chapter, we describe the main concepts and components of this thesis, such as differential privacy, including its variation for graphs, along with its key principles. We also present a few notions of graph theory and graph analytics.

2.1 Graph Theory

We start by presenting important aspects of understanding the graph data structure, which includes the notion of edge-weighted graphs, the notion of edge-weighted graphs, and attributed graphs, which consist of graph variations such that additional information may be present in the edges or nodes.

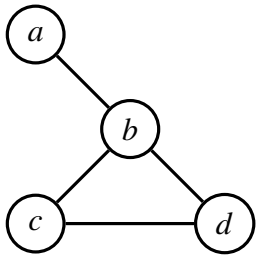
2.1.1 Graphs

Graphs are mathematical representations consisting of vertices (nodes) connected by edges that can have attributes like direction, weight, or labels, which can be applied in several applications, such as social network analysis.

Let $G = (V, E)$ be an undirected graph composed of a set of nodes V and a set of edges E representing the relationships between nodes. The set of nodes is defined as $V = \{v_1, \ldots, v_n\}$, such as $|V| = n$. The set of edges is defined as $E \subseteq V \times V = \{e_{i,j}, \ldots, e_{o,p}\}$, where $e_{i,j}$ refers to an undirected edge between nodes v_i and $v_j \in V$ ($e_{i,j} \equiv e_{j,i}$), and $|E| = m$.

Figure 2 presents an example of a graph $G = (V, E)$ with 4 nodes ($n = 4$) and 4 edges ($m = 4$), and its corresponding sets of vertices and edges, composed of $V = \{v_a, v_b, v_c, v_d\}$ and $E = \{e_{a,b}, e_{b,c}, e_{b,d}, e_{c,d}\}$, respectively.

Figure 2 – An example of a graph $G = (V, E)$ with $n = 4$ and $m = 4$.



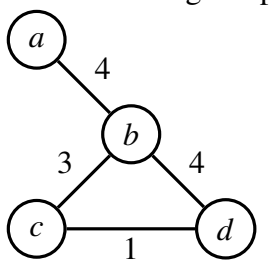
Source: Elaborated by the author.

2.1.2 Edge-Weighted Graphs

Edge-weighted graphs are graphs in which numerical values, i.e., weights, are assigned to the edges, representing attributes such as distance, cost, or capacity. This kind of graph is widely used in applications like network routing, shortest path algorithms, and resource optimization. We denote an edge-weighted graph as $G = (V, E, w)$, where V is the set of nodes, E is the set of edges, and w is a weight function w that maps each edge in E to a real number. The set of nodes is defined as $V = \{v_1, \ldots, v_n\}$, such as $|V| = n$. The set of edges is defined as $E \subseteq V \times V = \{e_{i,j}, \ldots, e_{o,p}\}$, where $e_{i,j}$ refers to an undirected edge between nodes v_i and $v_j \in V$ ($e_{i,j} \equiv e_{j,i}$), and $|E| = m$. The weight function $w : E \rightarrow \mathbb{R}$ assigns a real number to each edge in E, such that $w_{i,j}$ denotes the weight of the edge $e_{i,j} \in E$.

Figure 3 presents an example of a graph $G = (V, E, w)$ with 4 nodes ($n = 4$) and 4 edges ($m = 4$), and its corresponding sets of vertices and edges, composed of $V = \{v_a, v_b, v_c, v_d\}$ and $E = \{e_{a,b}, e_{b,c}, e_{b,d}, e_{c,d}\}$, respectively. Additionally, the edge weights are given by $w_{a,b} = 4$, $w_{b,c} = 3$, $w_{b,d} = 4$, and $w_{c,d} = 1$.

Figure 3 – An example of an edge-weighted graph $G = (V, E, w)$ with $n = 4$ and $m = 4$, where the numerical values near edges represent the edge weights.



Source: Elaborated by the author.

2.1.3 Edge-Attributed Graphs

Edge-attributed 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-attributed 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 attributes 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 edge attributes is defined as $X = \{x_1, \ldots, x_t\}$, and t is the number of possible edge attributes. 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 attribute $x_k \in X$ ($e_{i,j,k} \equiv e_{j,i,k}$), 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 $x_k, x_l \in X$.

Figure 1 presents an example of an edge-attributed graph $G = (V, E, X)$ composed of 10 nodes ($n = 10$) and 13 edges ($m = 13$), where nodes represent users and edges are emails exchanged among them. The topic of the emails represents the edge attribute. ‘‘AM’’ denotes administrative matters, and ‘‘WR’’ depicts work-related topics.

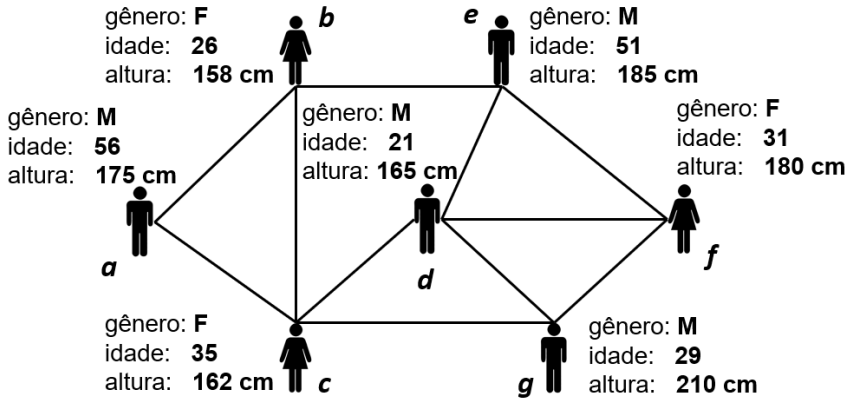
2.1.4 Node-Attributed Graphs

On the other hand, node-attributed graphs are graphs in which attributes are associated with the nodes. Nodes-attribute graphs have the capacity to characterize the nodes, being extremely useful for prediction tasks, especially by the presence of the homophily property, i.e., similar nodes may be more likely to attach to each other based on their attributes (PEROZZI, 2016).

We denote an undirected node-attribute 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 attributes associated with each node in V. The set of nodes is defined as $V = \{v_1, \ldots, v_n\}$, such that $|V| = n$. The set of edges is defined as $E \subseteq V \times V = \{e_{i,j}, \ldots, e_{o,p}\}$, where $e_{i,j}$ refers to an undirected edge between nodes v_i and $v_j \in V$ ($e_{i,j} \equiv e_{j,i}$), and $|E| = m$. The set of node attributes defined as $X = \{x_1, ..., x_n\}$ is the set of w-dimensional node attribute vectors, where w is the number of attributes and the vector $x_i = (x_{i,1}, ..., x_{i,w})$ contains the attributes associated with the node $v_i \in V$.

Figure 4 presents an example of a node-attributed graph $G = (V, E, X)$ composed of 7 nodes ($n = 7$) and 11 edges ($m = 11$), with 3 attributes ($w = 3$): gender, age, and height, associated with the nodes (or users) and the edges representing the relationships among users. For example, the attribute vectors of the users v_a and v_b are given by x_a = {gender: M, age: 56, height: 175 cm} and x_b = {gender: F, age: 26, height: 158 cm}, respectively.

Figure 4 – An example of a node-attributed graph $G = (V, E, X)$, where 3 attributes (gender, age, and height) are associated with the nodes (i.e., users) and edges represent the relationships among users.



Source: Elaborated by the author.

2.2 Graph Analytics

In the digital age, data has transformed into important assets for organizations, mainly due to its high value and importance. Currently, large volumes of data of various kinds are available and have become great strategic allies for companies in their decision-making processes through the analyses performed. In short, data analysis consists of the process of inspecting, treating, transforming, and modeling data to discover useful information, insights, and conclusions that assist in the decision-making of companies and organizations.

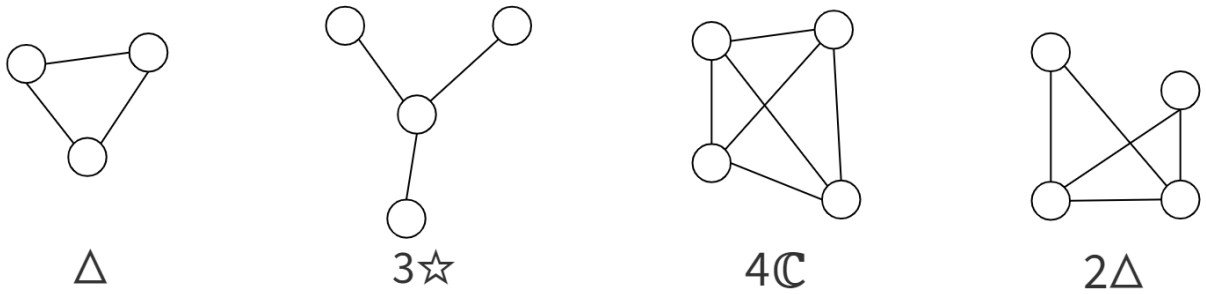
Commonly, data analyses occur on tabular data, represented through records, which limits analyses on more complex structures, such as graphs. Unlike analyses on tabular data, analyses on graphs prioritize the relationships between the nodes and their respective relationships or connections. Due to the inherent characteristics of graphs, various statistics can be extracted from analyses on them. The most common statistics include the degrees of nodes, along with their respective degree distributions, centrality metrics, and other pertinent measures. Additionally, subgraph counts, as well as various distance metrics, are also examples of metrics frequently examined in graph analysis.

2.2.1 Subgraph Counts

Subgraph analysis plays a crucial role in understanding networks, providing insights into the structure and underlying patterns of the interactions between nodes (RIBEIRO *et al.*, 2022). Among the different types of subgraphs, triangles, stars, and cliques stand out, each providing relevant information about the connectivity and structure of the network. Figure 5

exemplifies four different types of subgraphs that can be present in a graph: triangles, k-stars, k-cliques, and k-triangles.

Figure 5 – Examples of subgraphs that can be present in a graph: triangles, 3-stars, 4-cliques, and 2-triangles.



Source: Elaborated by the author.

Triangles are subgraphs composed of three interconnected nodes forming a triangular structure. Counting triangles in a graph is important for identifying densely connected clusters of three individuals. The occurrence of triangles indicates local proximity relationships, which can suggest the existence of communities within the network.

On the other hand, a k-star consists of a central node connected to k peripheral nodes, forming a star-like structure. This concept is particularly valuable for identifying and characterizing the most influential nodes and interaction patterns within a network. Additionally, the distribution of k-stars helps to understand the dynamics of graphs. In communication networks, for example, a central node with many connections can act as a communication hub, where information is aggregated and distributed.

A k-clique is a complete subgraph composed of k nodes, where each pair of nodes is directly connected by an edge. Counting k-cliques is essential for identifying cohesive and highly connected groups in the graph, such as close friend groups or collaborative work teams. The existence of k-cliques suggests that the members of these subgraphs share common interests or have a high frequency of communication.

Finally, k-triangles are an extension of the concept of triangles in graphs. Specifically, this subgraph is formed by a set of k triangles that share a common vertex. This structure is used to identify areas of high interconnectivity in a network where multiple relationships converge on a single node. The analysis of k-triangles is particularly useful for identifying central or highly influential nodes in the network. In scientific collaboration networks, for example, a central node with many k-triangles may represent a researcher who collaborates with various distinct groups, serving as an intersection point between different research communities.

2.2.2 Histograms

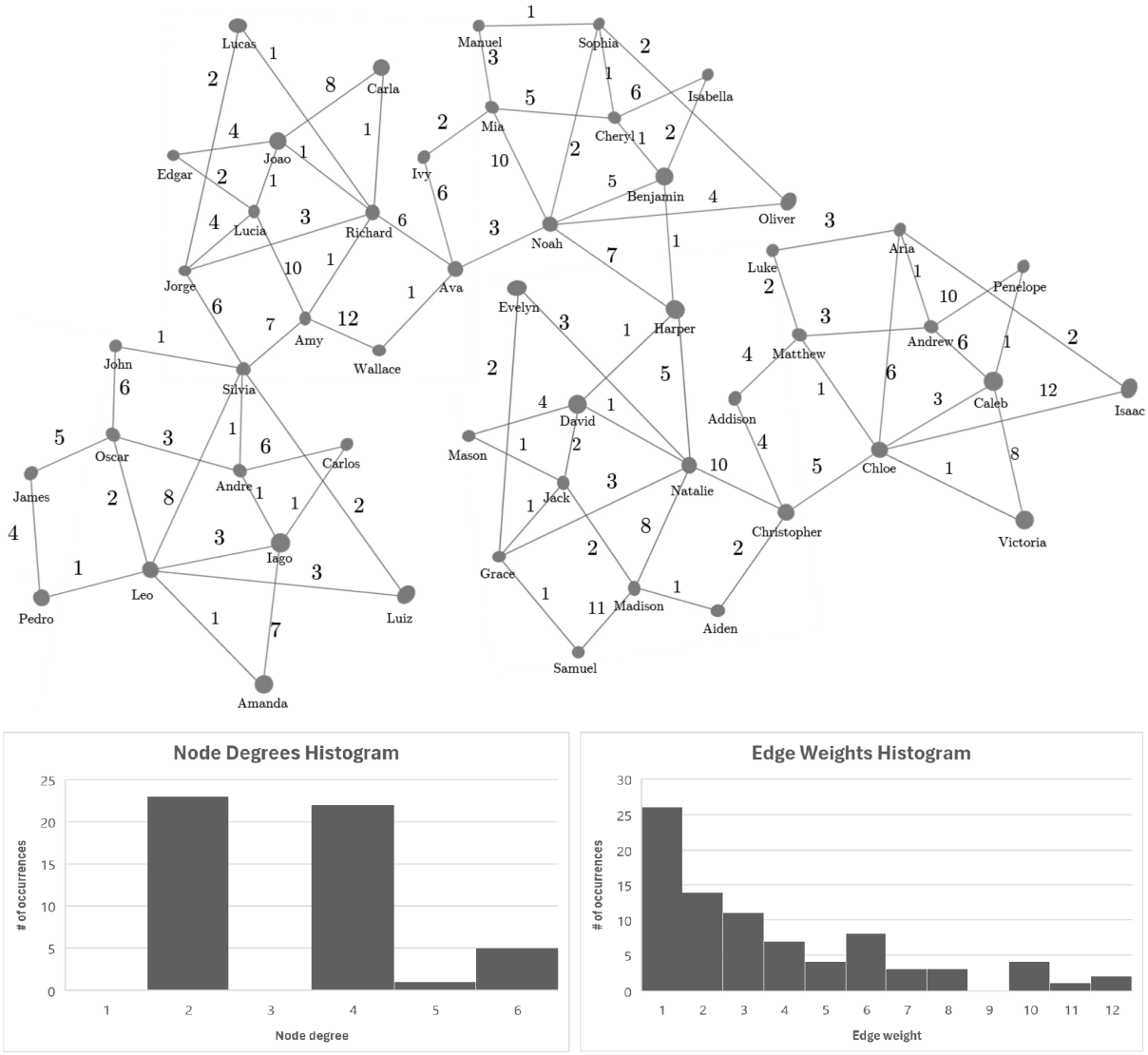
Histograms are graphical representations that illustrate the distribution of a dataset. Composed of rectangular bars, each bar represents the frequency of data within specific intervals, called bins. The height of the bars corresponds to the number of occurrences of the data in each interval. By visually representing the distribution of certain characteristics of a graph, histograms allow a clear understanding of how connectivity is structured within the network. They help to identify recurring patterns and trends that may be difficult to see (COOK; HOLDER, 2006). For example, a histogram of the degree distribution of a graph reveals how connections are distributed among the nodes, highlighting those that are highly connected (hubs) and indicating whether the network follows a power-law distribution or is more homogeneous. Another example would be a histogram of edge weights in edge-weighted graphs, where it is possible to identify which edges carry greater relevance or influence in the network's dynamics. In this scenario, edges with consistently high weights, evidenced by peaks in the histogram, suggest critical connections or robust relationships between nodes. Figure 6 illustrates two histograms, one of node degrees and another of edge weights, from an edge-weighted graph.

Additionally, histograms are useful for examining the structural properties of graphs, such as the distribution of clustering coefficients, which measure the tendency of nodes to form clusters. A histogram of coefficients can show whether the network has well-defined communities or if it is more dispersed. Another critical aspect of histograms is their ability to detect anomalies. By comparing histograms from different periods or subsets of the network, it is possible to identify sudden changes or unusual patterns that may indicate anomalous behaviors, such as cyberattacks, system failures, or even the unexpected formation of new social groups. For example, a significant change in the degree distribution might suggest the massive addition or removal of nodes or edges, pointing to extraordinary events in the network.

2.2.3 Shortest Paths and Distances

A shortest path between two nodes in a graph is defined as the sequence of edges that connects these two nodes with the smallest sum of edges or weights. If the graph is unweighted, the shortest path is simply the path with the fewest number of edges. In edge-weighted graphs, the shortest path is the one that minimizes the sum of the edge weights along the path (CORMEN *et al.*, 2022). The definition of a shortest path is presented below:

Figure 6 – Histograms of node degrees and edge weights from a given edge-weighted graph.



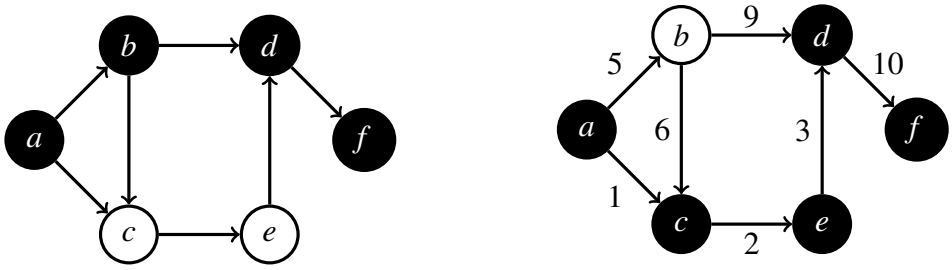
Source: Elaborated by the author.

Definition 1. *(Shortest Path (CORMEN et al., 2022)). Let $G = (V, E)$ be a graph, where V is the set of vertices and E is the set of edges, each edge $(u, v) \in V$ having a weight $w(u, v) \geq 0$. Given a source vertex $s \in V$ and a target vertex $t \in V$, the shortest path from s to t is a path $P = \{s, v_1, \ldots, t\}$ such that the sum of the weights of the edges in P is minimized. Formally, the shortest path minimizes the function $w(P) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$, where $P = \{v_0 = s, v_1, \ldots, v_k = t\}$ is the sequence of vertices along the path and $w(v_i, v_{i+1})$ is the weight of the edge connecting v_i and v_{i+1}.*

Figure 7 illustrates a network with the shortest path in an unweighted graph (Figure 7a) and in an edge-weighted graph (Figure 7b).

Another important metric involving distances in graphs is the average shortest path

Figure 7 – Example of shortest paths between the nodes a and f.



(a) Unweighted graph. (b) Edge-weighted graph.

Source: Elaborated by the author.

length. It is calculated as follows: for each shortest path between all pairs of nodes, all distances are summed, and this value is divided by the total number of pairs of nodes. In networks, a low average shortest path length suggests a more cohesive network where individuals are closer to each other, facilitating the dissemination of information, influences, and interactions. On the other hand, a high average shortest path length indicates a more dispersed network, where the distances between individuals are greater. Therefore, this can make it difficult to spread information.

Finally, the diameter metric in a graph is another important measure used to characterize the maximum distance between vertices. This metric measures the longest distance found when traversing all pairs of vertices in the graph, representing the longest distance that must be traveled to connect any pair of nodes in the network. Diameters in social networks are related to the concept of ‘‘six degrees of separation’’ (SAMOYLENKO *et al.*, 2023), which suggests that any person in the world can be connected to any other person through at most six intermediaries.

2.3 Differential Privacy

Differential privacy (DP) (DWORK, 2006) is a robust privacy definition that has become the standard for data release under strong privacy guarantees. Much is attributed to the existing limitations in more traditional or synthetic privacy models, such as k-anonymity, l-diversity, t-closeness, δ-presence, among others, which are not robust enough to provide a desirable level of privacy to the individuals (BRITO; MACHADO, 2017).

2.3.1 Intuition and Definition

In the original definition of differential privacy, private data is viewed as a collection of records, where each record corresponds to an individual. Essentially, differential privacy ensures individual privacy protection by injecting noise into the results of queries applied to

individuals' data, that is, by modifying the original data through the introduction of randomness (DWORK *et al.*, 2006). The fundamental premise of differential privacy is that the outcome of any query is almost equally likely to occur, regardless of the presence or absence of any individual in the dataset. It is important to mention that differential privacy is not a simple tool but a paradigm capable of quantifying and managing the risks of privacy breaches. Therefore, differential privacy can be applied from simple statistical estimates to machine learning.

Let Q be a query to be performed on a dataset D, which contains sensitive information about a group of individuals. DP is defined through a randomized algorithm M, also called a mechanism, which is executed on D. DP ensures that the output of $M(D)$ should be similar to the output of $Q(D)$, i.e., the goal of DP is to make the output of $M(D)$ as close as possible to the output of $Q(D)$ to ensure data utility while simultaneously preserving the privacy of all individuals in the dataset.

To preserve the privacy of all individuals through a mechanism M, DP establishes the notion of neighboring datasets. Two datasets D and D' are said to be neighbors if they differ by at most one record, denoted as $D \sim D'$. D' can be obtained from D by adding or removing a single record. Figure 8 presents an example of two neighboring datasets D and D', initially composed of 5 records, corresponding to users labeled from ''A'' to ''E''. Note that the dataset D' was obtained after removing the record of the user ''E'' from the dataset D. DP ensures that regardless of whether the input is D or D', the probability of a given output occurring from $M(D)$ or $M(D')$ is almost the same. This property is denoted as the indistinguishability of neighboring datasets. In other words, DP states that any query response occurs with a similar probability, regardless of the presence or absence of any individual in the dataset.

Figure 8 – Example of two neighboring datasets D and D' after removing the record of user E from D.

Dataset D

User	Gender	Age	Income ($)
A	Male	20	40,000
B	Male	30	55,000
C	Female	25	50,000
D	Female	35	70,000
E	Male	50	60,000

Dataset D'

User	Gender	Age	Income ($)
A	Male	20	40,000
B	Male	30	55,000
C	Female	25	50,000
D	Female	35	70,000

Source: Elaborated by the author.

Before presenting the definition of DP, consider that $Range(M)$ consists of all possible outputs of M, that is, its output domain. For example, if M calculates the number of records in a

dataset, then $Range(M)$ is equal to the set of non-negative integers. Finally, the definition of DP is presented below:

Definition 2. *(ε-Differential Privacy (DWORK, 2006)). A mechanism M satisfies ε-differential privacy if, for any two neighboring datasets D and D', and for any possible output $O \subseteq Range(M)$,*

$$\Pr[M(D) = O] \leq \exp(\varepsilon) \times \Pr[M(D') = O], \tag{2.1}$$

where $\Pr[\cdot]$ *denotes the probability of the mechanism $\mathcal{M}$ to output O.*

2.3.2 Privacy Budget

The parameter ε that appears in Definition 2 is called the privacy budget. This parameter is responsible for controlling the differences between the probabilities of the outputs of a mechanism that is executed on two neighboring datasets, i.e., the privacy budget ensures that these differences are limited to at most ε.

The privacy budget consists of a positive real number that controls the level of privacy that a mechanism M provides. A smaller ε offers stronger privacy guarantees, with more indistinguishable probability distributions, but lower data utility since more noise must be added to the result. Similarly, a larger ε provides weaker privacy guarantees but higher data utility.

Defining the appropriate value of ε for an application is a very challenging task. This task requires the effort of various parties, such as privacy experts, stakeholders, and data owners (i.e., individuals who share their data), to provide continuous feedback to ensure individuals' privacy while also releasing meaningful information. However, the privacy budget typically assumes small values, making the mechanism's output probabilities almost the same, regardless of whether the input to the mechanism is D or D'. Several studies have already addressed the problem of determining a desirable value for ε (HSU *et al.*, 2014; LI *et al.*, 2016). Nonetheless, it has been widely argued that $0.1 \leq \varepsilon \leq 1$ provides strong privacy guarantees and acceptable levels of utility, while $\varepsilon \geq 5$ is acceptable only in some specific applications (BRITO, 2023).

2.3.3 Sensitivity

As previously mentioned, DP can be achieved by adding an appropriate amount of noise to the query results. However, adding excessive noise can drastically reduce the data utility, reducing the accuracy of analyses, while an insufficient amount of noise may not provide adequate

privacy guarantees. Therefore, the noise added to a query Q depends on the global sensitivity of Q (DWORK *et al.*, 2006). The definition of global sensitivity is given below:

Definition 3. *(Global Sensitivity (DWORK et al., 2006)). 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*

$$\Delta Q = \max_{D,D'} ||Q(D) - Q(D')||_1. \tag{2.2}$$

The global sensitivity, also simply called sensitivity, measures the maximum impact on the query results from adding or removing any record in the dataset. The sensitivity serves as an essential parameter for determining the appropriate amount of noise to be added to the query. It is important to mention that sensitivity is related only to the query function and is independent of the dataset. For example, a query with low sensitivity requires only a small amount of noise to be added to the query results to mask the impact of adding or removing a record. On the other hand, when the sensitivity is high, a significant amount of noise must be added to the query results to ensure the privacy of individuals, compromising the data utility.

For some queries, the sensitivity is straightforward to calculate. For example, the sensitivity of counting queries is 1 since adding or removing a record in the dataset will affect the query results by, at most, 1. On the other hand, the sensitivity of more complex queries, such as maximum and sum queries, is not as simple to calculate as for counting queries.

For example, consider a query that calculates the sum of the weights of people in a given dataset. The inclusion of a new record in the dataset will increase the query result by an amount equivalent to the weight of the individual added. Therefore, the sensitivity will depend on the weight value of the individual added to the dataset. Note that the same reasoning applies to the removal of a record from the dataset. Then, we desire to assign a specific value to represent the sensitivity of this query, as the query should be independent of the dataset. For the specific domain of weights, there is a known rational upper bound for the maximum weight that an individual can have. According to (ALLARDYCE, 2012), Jon Brower Minnoch was the heaviest known person ever documented, weighing an impressive 635 kilograms. Thus, it is plausible to assign a value of 635 to the sensitivity of this query. However, this does not serve as definitive proof, as it is impossible to guarantee that another person will not weigh 635 kilograms or more in the future. Therefore, in some domains, determining a reasonable sensitivity can be a challenging task (NEAR; ABUAH, 2021).

2.3.4 Differential Privacy Mechanisms

Mechanisms are algorithms capable of ensuring the properties of DP. For numerical queries, DP can be achieved through various mechanisms, such as the Laplace mechanism (DWORK, 2006) and the geometric mechanism (GHOSH *et al.*, 2009). Although both mechanisms are designed for numerical queries, they differ in the type of noise added to the query result. The Laplace mechanism is recommended for queries that generate real values, as this mechanism produces noise values $\in \mathbb{R}$. In turn, the geometric mechanism is recommended for queries that generate integer values, as this mechanism produces noise values $\in \mathbb{Z}$. However, not all queries generate numerical values. For such queries, also known as categorical queries, the exponential mechanism (MCSHERRY; TALWAR, 2007) is more suitable.

2.3.4.1 Laplace Mechanism

As briefly mentioned earlier, the Laplace mechanism adds real-valued noise to the query results. As the name suggests, the mechanism relies on the Laplace distribution to generate random values, which will be added to the query result. Let x be the noise added to the result of a query Q, the Laplace distribution is defined as follows:

Definition 4. *(Laplace Distribution). The Laplace distribution with mean 0 and scale b is the distribution with probability density function*

$$Lap(x|b) = \frac{1}{2b} \cdot \exp(-\frac{|x|}{b}). \tag{2.3}$$

Consider $Lap(b)$ as the Laplace distribution with scale b, Q as a query, and D as a dataset. The Laplace mechanism works by calculating the result of $Q(D)$ and perturbing this result by adding noise generated from the Laplace distribution. The scale b of the generated noise is calibrated through the relation between the sensitivity of the query and the privacy budget, such that $b = \frac{\Delta Q}{\varepsilon}$.

Theorem 1 (Laplace Mechanism (DWORK *et al.*, 2006))**.** *The Laplace mechanism that adds noise drawn from $Lap(\frac{\Delta Q}{\varepsilon})$ satisfies ε-DP.*

2.3.4.2 Geometric Mechanism

The geometric mechanism (GHOSH *et al.*, 2009) is the discrete version of the Laplace mechanism, i.e., it adds integer noise to the query results according to the two-sided geometric

distribution. This ensures that the final query result is an integer. Therefore, the geometric mechanism is specialized in improving the performance of counting queries (GHOSH *et al.*, 2009). The two-sided geometric distribution is defined as follows:

Definition 5. *(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*

$$P(X = x) = \frac{1 - \alpha}{1 + \alpha}\alpha^{|x|}. \tag{2.4}$$

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

Theorem 2 (Geometric Mechanism (GHOSH *et al.*, 2009))**.** *Given any query* $Q : \mathbb{N}^{|\mathcal{D}|} \rightarrow \mathbb{Z}^k$*, the geometric mechanism defined as*

$$\mathcal{M}_G(D, Q, \varepsilon) = Q(D) + (Y_1, \ldots, Y_k), \tag{2.5}$$

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* $\alpha = e^{-\frac{\epsilon}{\Delta Q}}$.

2.3.4.3 Exponential Mechanism

As mentioned earlier, the exponential mechanism arises as a solution for queries that do not return numerical values. For this purpose, McSherry et al. (MCSHERRY; TALWAR, 2007) proposed the exponential mechanism, which ensures DP for categorical queries. Its main idea consists of choosing an output O from the output space $\mathcal{O}$, according to a utility function u. This utility function assigns exponentially higher probabilities to the outputs with higher utilities. Moreover, the choice of u depends on the application. Thus, different applications lead to different utility functions. Unlike the previous mechanisms, Laplace and geometric, which generate their noise proportional to the sensitivity of the query Q, the exponential mechanism uses the concept of sensitivity of the utility function to provide the mechanism's output. The sensitivity of the utility function is defined below:

Definition 6. *(Global Sensitivity of the Utility Function (MCSHERRY; TALWAR, 2007)). The global sensitivity of a utility function u is given by*

$$\Delta u = \max_{O \in \mathcal{O}} \max_{D, D' neighbors} |u(D, O) - u(D', O)|. \tag{2.6}$$

Theorem 3 (Exponential Mechanism (MCSHERRY; TALWAR, 2007)). *Given a utility function* $u : (D \times O \rightarrow \mathbb{Z})$ *for a dataset* D*, the mechanism* M *that samples an output* $O \in O$ *with probability proportional to* $\exp\frac{\epsilon \cdot u(D,O)}{2\Delta U}$ *satisfies* ϵ*-DP.*

2.3.5 Differential Privacy Properties

Several useful properties are integrated into the DP mechanisms, such as post-processing, sequential composition, and parallel composition. The post-processing property assumes that any function applied to the output of a differentially private mechanism also satisfies DP. In turn, the sequential composition property assumes that any sequence of differentially private mechanisms that satisfies DP in isolation also provides DP in sequence. Finally, the parallel composition assumes that the same differentially private mechanism, when applied to disjoint datasets satisfying DP in isolation, also provides DP. These properties are formally stated below:

Theorem 4 (Post-processing (DWORK; ROTH, 2014)). *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 5 (Sequential Composition (DWORK; ROTH, 2014)). *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 6 (Parallel Composition (DWORK; ROTH, 2014)). *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.*

When combined, these properties provide the flexibility to devise a way to aggregate several differentially private steps into a sole mechanism that satisfies DP.

2.4 Local Differential Privacy

Differential privacy, as previously presented, considers the existence of a trusted curator (third-party) who is responsible for collecting the data, perturbing the query results through a mechanism that satisfies DP, and providing noisy results. This setup of DP is generally referred to as global DP, centralized DP, or simply DP. However, finding a trustworthy curator to collect and process the data can be a challenging task in practical scenarios. Therefore, the lack

of reliable curators limits the applicability of global DP. As a result, Local Differential Privacy (LDP) (DUCHI *et al.*, 2013) has been proposed as a differentially private approach that eliminates the need for a trusted data curator. Instead of centralizing the data flow in a single, supposedly trustworthy external entity, each individual is responsible for protecting their data by perturbing it locally through a differentially private mechanism before sending it to the data curator. In LDP, the data curator is also commonly referred to as an aggregator.

When compared to the global DP, LDP is a stronger notion of privacy, which keeps individuals' sensitive data private, even from untrusted data curators. However, as it is a stronger notion of privacy, LDP is expected to introduce more noise to the results under the same circumstances, i.e., using the same privacy budget when compared to the global DP. The formal definition of LDP is presented below:

Definition 7. *(ε-Local Differential Privacy (DUCHI et al., 2013)). A mechanism $\mathcal{M}$ satisfies ϵ-local differential privacy if for any pair of values $v, v' \in D$ and for any possible output $O \subseteq Range(\mathcal{M})$,*

$$\Pr[M(v) = O] \leq \exp(\varepsilon) \times \Pr[M(v') = O]. \tag{2.7}$$

The main difference between DP and LDP lies in the input data that the mechanisms receive. A global DP mechanism receives a dataset D as input, i.e., the data of all individuals, and ensures that the output is indistinguishable. In contrast, LDP receives only the data of a single individual v as input and independently generates noisy responses per individual.

2.4.1 Local Differential Privacy Protocols

As mentioned earlier, mechanisms are ways to ensure the properties of DP. In LDP, these properties are achieved through the use of protocols. Therefore, in LDP, mechanisms are referred to as protocols. In summary, protocols are techniques that modify an individual's data to ensure the properties of DP. The standard flow of an LDP protocol consists of (I) encoding the individual's data, (II) perturbing the individual's data, and (III) sending the individual's noisy data to the data curator.

(I) Encoding: In this step, the individual's data v is encoded into a bit vector B of size d consisting of 0's and 1's, such that the value 1 is assigned to the positions in the vector B that corresponds to v, and 0 to the remaining positions. Thus, the function $\texttt{Encode}(v) = B$ is defined such that $B[v] = 1$ e $B[i] = 0$ for all $i \neq v$.

(II) Perturbation: In this step, the encoded bit vector B is perturbed according to two main parameters: p and q, resulting in a new bit vector B', as shown in Equation 2.8. The parameter p is the probability that a bit i in B assigned with the value 1 remains 1 even after being perturbed, i.e., $B[i] = 1 \rightarrow B'[i] = 1$. On the other hand, q is the probability that a bit i in B assigned with the value 0 becomes 1 after being perturbed, i.e., $B[i] = 0 \rightarrow B'[i] = 1$. Intuitively, $(1-p)$ and $(1-q)$ represent the probabilities of $B[i] = 1 \rightarrow B'[i] = 0$ and $B[i] = 0 \rightarrow B'[i] = 0$, respectively. This step is performed in a differentially private manner, using the privacy budget parameter ε to determine the probability values p and q. Additionally, the values of p and q depend not only on the value of ε but also on the chosen protocol. Once perturbed, the vector B' is reported to the aggregator.

$$\Pr[B'(i) = 1] = \begin{cases} p, \text{ if } B[i] = 1 \\ q, \text{ if } B[i] = 0 \end{cases} \tag{2.8}$$

(III) Aggregation: In this step, the aggregator collects all the perturbed bit vectors B' reported by the individuals and performs the analysis of these data based on the aggregated information. The basis for performing the analyses consists of identifying the number of occurrences of each possible input value v from the vectors B', using a `Support` function. For example, a vector B' supports an input value v if $B'[v] = 1$, i.e., $\texttt{Support}(B') = \{v \mid B'[v] = 1\}$ is the set of values present in B'. Similarly, $\texttt{Support}(v)$ is defined as the number of occurrences of the value v in the reported vectors B'.

It is important to mention that the `Encode` and `Support` functions are directly dependent on the LDP protocol used. Thus, different protocols may implement these functions differently. Additionally, some relevant information is publicly known, i.e., known by the aggregator. In summary, the number of responses n, the size of the encoded vector d, the privacy budget ε, and the LDP protocol used are known by the aggregator. Therefore, the aggregator is able to understand from which LDP protocol the data was sanitized and then calculate the respective probability values p and q. Finally, the aggregator can perform an unbiased estimation of the reported values according to Theorem 7. When each individual submits their data only once, the number of responses n can be treated as the number of individuals equivalently.

Theorem 7 (Unbiased Estimation (WANG *et al.*, 2017))**.** *Given an LDP protocol, the number of occurrences of a value v, given by* $\tilde{c}(v) = \frac{\texttt{Support}(v) - n \cdot q}{p - q}$*, is unbiased, where* $\texttt{Support}(v)$ *is the number of responses containing the value v and n is the number of responses.*

The problem of frequency estimation, where the aggregator estimates the frequencies of values in a pre-established domain, is one of the most fundamental problems that LDP aims to solve. Problems of this nature are commonly known as Frequency Oracles (FO). Several studies have already been conducted to develop FO protocols (FILHO; MACHADO, 2023; ACHARYA *et al.*, 2019; BASSILY; SMITH, 2015; YE; BARG, 2018), where the Randomized Response (RR) protocol (DWORK *et al.*, 2006) and the Unary Encoding (UE) (ERLINGSSON *et al.*, 2014) protocol are among the most disseminated in the literature.

2.4.1.1 Randomized Response Protocol (RR)

The RR protocol was one of the first FO protocols proposed in the literature. Among its main characteristics, it allows a value v to be encoded into a bit vector B in such a way that B has more than one representative bit assigned with 1. This representation can be quite useful in various domains. For instance, consider the context of social networks, where an individual u wants to report his connections with other individuals. In this case, the value v to be reported consists of a list containing the other individuals connected to u. Therefore, one way to encode v would be to transform it into a bit vector B so that $B[i] = 1$ indicates that there is a connection between individuals u and i, while $B[i] = 0$ indicates the absence of that connection.

To ensure the properties of LDP, it has been proven that the RR protocol satisfies ε-LDP if the perturbation step is performed with specific values of p and q, such that $p = \frac{e^{\varepsilon}}{1+e^{\varepsilon}}$ and $q = 1 - p$ (ERLINGSSON *et al.*, 2014).

2.4.1.2 Unary Encoding Protocol (UE)

The UE protocol differs from the RR protocol primarily in the way that the data is encoded. As the name suggests, in the UE protocol, the encoding of any value is done through a single representative bit. Thus, for a given value v, it will be encoded into a bit vector B such that B has a single bit assigned with 1, while all other bits are set to 0. This representation is also quite useful in various domains, especially in simpler domains with a smaller dimension d.

As the RR protocol, it is also necessary to establish the values of p and q that allow the UE protocol to satisfy ε-LDP and ensure the properties of LDP. Therefore, some protocols based on the UE protocol with particular characteristics have emerged. Among them, the Symmetric Unary Encoding (SUE) protocol (ERLINGSSON *et al.*, 2014) and the Optimized Unary Encoding (OUE) (WANG *et al.*, 2017) protocol stand out. Both protocols are quite similar in terms of

the executed steps. The main difference lies in the choice of the parameters p and q used in the perturbation step of each protocol.

In the SUE protocol, the values of p and q are chosen to treat the bits 0 and 1 symmetrically. The values of p and q are given by $p = \frac{e^{\frac{\varepsilon}{2}}}{e^{\frac{\varepsilon}{2}}+1}$ and $q = \frac{1}{e^{\frac{\varepsilon}{2}}+1}$, such that $p + q = 1$. On the other hand, the OUE protocol is an improvement over the SUE protocol, which proposes optimal values for p and q. Assigning $p = \frac{1}{2}$ and $q = \frac{1}{e^{\varepsilon}+1}$ improves the utility of the estimated frequencies. Note that the value of $p + q$ will never be equal to 1 since the value of ε is always positive. In summary, the idea of the OUE protocol is to ensure that the bits reported as 1 are indeed those that were originally 1, just as the bits reported as 0 are those that were originally 0.

Finally, it is worth mentioning that do not exist a protocol that is best for everything. There are various protocols in the literature, each with different characteristics and purposes. Therefore, determining which protocol is most recommended for a particular task becomes quite challenging (WANG *et al.*, 2017).

2.5 Differential Privacy for Graphs

The fundamental concept of DP relies on the definition of neighboring datasets. In previous definitions, a neighboring dataset is defined as a dataset obtained by adding or removing a single record. However, regarding graph data, which mainly focuses on the relationship between individuals, the association between private data and dataset records becomes less clear. Therefore, to apply DP to graphs, it is necessary to establish a new definition for neighboring graphs that considers the graph structure and the privacy semantics associated with the graph.

2.5.1 Differential Privacy Models 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) (HAY *et al.*, 2009) and Node Local Differential Privacy (node-LDP) (KASIVISWANATHAN *et al.*, 2013). 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 8. *(ϵ-Edge Local Differential Privacy (HAY et al., 2009)). A mechanism $\mathcal{M}$ 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 $O \subseteq Range(\mathcal{M})$,

$$\Pr[\mathcal{M}(B) = O] \leq \exp(\epsilon) \times \Pr[\mathcal{M}(B') = O]. \tag{2.9}$$

Definition 9. *(ϵ-Node Local Differential Privacy (KASIVISWANATHAN et al., 2013)). A mechanism $\mathcal{M}$ satisfies ϵ-node local differential privacy if for any two adjacency bit vectors B, B' and for any output $O \subseteq Range(\mathcal{M})$,*

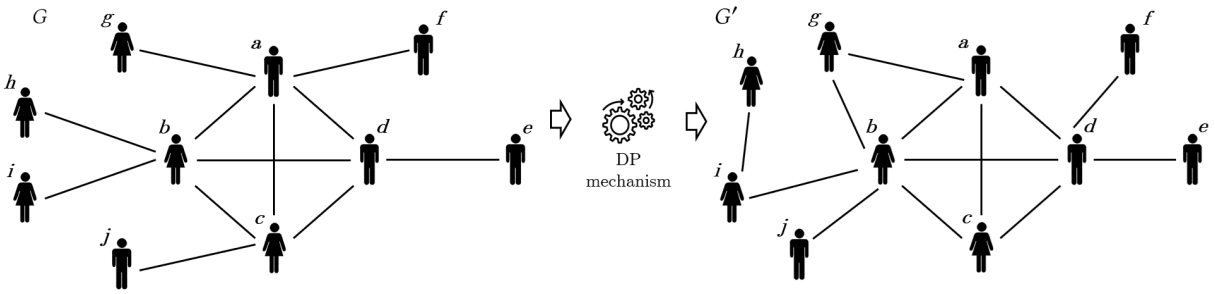
$$\Pr[\mathcal{M}(B) = O] \leq \exp(\epsilon) \times \Pr[\mathcal{M}(B') = O]. \tag{2.10}$$

Both edge-LDP and node-LDP satisfy the DP properties stated in Section 2.3.5. 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 (LI *et al.*, 2023), and anomaly detection (SHAH *et al.*, 2016) while preserving high data utility. Therefore, this thesis focuses on the edge-LDP setting.

2.5.2 Release of Graph Information Under Differential Privacy

Graphs are a powerful representation used to represent complex relationships and interactions in various domains such as social networks, biological networks, transportation systems, and more. Analyzing these graphs can reveal significant insights, from community structures and influential nodes to patterns of connectivity and paths of interaction. However, these analyses often involve sensitive information, necessitating robust privacy-preserving techniques. In this sense, DP provides strong privacy guarantees for such analyses. This section discusses the differences between the main approaches for applying DP to graphs: Entire Graph Release and Graph Statistics Release.

Figure 9 – An example of an input graph G and its respective private version G', generated after applying a hypothetical DP mechanism on G.



Source: Elaborated by the author.

2.5.2.1 Entire Graph Release

The entire graph release approach involves applying a DP mechanism to the input graph and releasing its private version. This method aims to preserve the privacy of individuals within the graph while allowing comprehensive analyses in the private graph.

Generally, the process consists of perturbing the graph, which may involve adding or removing nodes and edges or modifying edge and node attributes. This approach has several advantages, such as allowing researchers to perform a wide range of analyses on the perturbed graph, such as community detection, centrality measures, and pathfinding, providing flexibility in the types of studies conducted. However, it is quite challenging to balance utility and privacy while keeping valuable analysis in the perturbed graph.

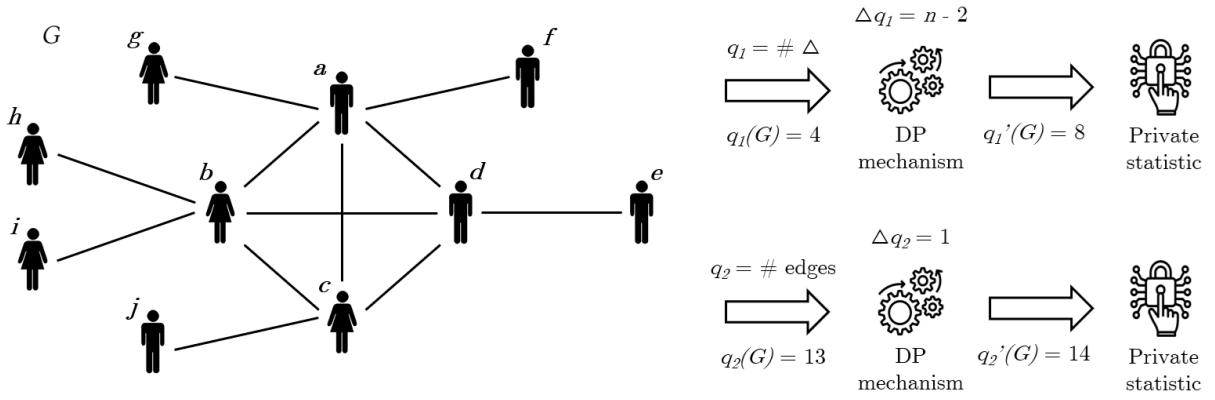
Figure 9 presents an example of an input graph G and its respective private version G', generated after applying a hypothetical DP mechanism on G. In this example, it is possible to observe that the nodes remained unchanged while some edges were modified.

2.5.2.2 Graph Statistics Release

Differently from the previous approach, the graph statistics release approach focuses on releasing specific statistical information about the graph rather than the entire graph structure. This method applies DP mechanisms to individual statistics derived from the input graph.

Specific graph statistics, such as number of edges and nodes, degree distributions, clustering coefficients, shortest paths, or subgraph counts, are computed with added noise to ensure the DP guarantees. The sensitivity of the graph metrics plays a crucial role in the amount of noise that needs to be added. This approach can achieve a good balance between utility and privacy in some scenarios, especially in those where the metrics under interest present low

Figure 10 – An example of an input graph G and two statistics q_1 and q_2 queried on G under the edge-DP model.



Source: Elaborated by the author.

sensitivities, potentially providing good-quality analyses.

On the other hand, metrics with inherently high sensitivity, such as certain centrality measures, require substantial noise addition to achieve DP guarantees. This significant perturbation can make the results less useful or even meaningless. Additionally, for some complex metrics, calculating their sensitivity can be a non-trivial and computationally demanding task. This complexity can make it challenging to apply DP accurately, leading to either over or under-estimation of the required noise.

Figure 10 presents an example of an input graph G and two statistics queried on G. The first query (q_1) consists of the number of triangles in G, while the latter (q_2) consists of the number of edges in G. For this example, consider that edge-DP is the graph DP model under interest. q_1 is a high-sensitivity query, being proportional to n, i.e., the number of nodes of the graph, with $\Delta q_1 = n - 2$, which leads to an excessive amount of noise added to the outcome. Differently, q_2 is a low-sensitivity query, with $\Delta q_2 = 1$, requiring only a small amount of noise to ensure DP.

The choice between releasing the entire graph or specific statistics involves a trade-off between flexibility, utility, privacy, and computational efficiency. Understanding these trade-offs is crucial for researchers and practitioners to apply DP mechanisms to graph data in a manner that aligns with their specific analytical goals and privacy requirements.

2.6 Summary

In this chapter, we provided an essential foundation for understanding this thesis by delving into crucial concepts and definitions. We began with an introduction to graph theory,

explaining the various graph analyses and the different types of graphs such as edge-weighted, edge-attributed, and node-attributed graphs, and their structural properties. In the following, we explored the principles of differential privacy, presenting its core definitions and properties, and significant mechanisms like the Laplace, geometric, and exponential. Additionally, we discussed the concept of local differential privacy, detailing various protocols that ensure privacy at the data collection level. Finally, we introduced the notion of differential privacy for graphs, addressing the challenges and presenting the main notions of neighboring graphs to adopt differential privacy on graph data, as well as the different analysis purposes on graphs.

3 RELATED WORK

In this chapter, we review the studies made in the last years towards the differentially private release of graph data. Various algorithms that satisfy the definition of DP have been developed to release the entire graph. However, since we do not consider the existing works related to node-DP, we focus only on the edge-DP-based approaches and present them in Section 3.1. Additionally, global DP techniques are more frequent in the literature than local DP settings. Additionally, we review the general studies on global DP for graphs in Section 3.1.1 and on LDP for graphs in Section 3.1.2. Finally, we present the studies in the field of DP applied to attributed graphs. These works are detailed in Section 3.2, where we cover recent works regarding weighted, node, and edge-attributed graphs. We summarize the entire section within a comparative table and position ourselves in relation to the presented studies in Section 3.3.

3.1 Approaches for Edge-DP

Several studies have already been conducted to release differentially private graph statistics. This problem focuses on releasing specific subgraph statistics, such as k-triangles, k-stars, and k-cliques (KARWA *et al.*, 2011; ZHANG *et al.*, 2015). However, it appears to be more limited when compared to the studies that aim to release the entire graph. Releasing the entire graph allows a huge variety of analyses, not limited to some specific subgraph statistics. Yet, simultaneously releasing entire graphs that provide high-fidelity analysis for different statistics may become challenging. We describe these works below, dividing them into sections according to the DP setting: global DP and local DP.

3.1.1 Global DP Approaches

The differentially private release of entire graphs within global DP has been extensively studied for over a decade. As mentioned, the main advantage of these approaches is that they are agnostic to the kind of analysis performed on the released graph since they allow the computation of any statistics.

Sala et al. (SALA *et al.*, 2011) proposed a differentially private graph model called Pygmalion. The purpose of Pygmalion consists of discovering the graph topology under edge-DP by transforming the provided graph structure into a private dK-graph and subsequently generating a synthetic graph. Later, Wang et al. (WANG; WU, 2013) proposed an improvement in the

accuracy of the *dK*-graph model by adjusting the noise levels according to the smooth sensitivity (NISSIM *et al.*, 2007). The smooth sensitivity consists of a new definition of sensitivity that considers the smoothness of the query function being computed, resulting in mechanisms that could achieve the same level of privacy with less noise added. The authors first extracted various parameters from the original graph, such as the degree correlations, and used them in the *dK*-graph model. This process ensures edge-DP for the learned parameters and, consequently, for the graph generation.

Xiao et al. (XIAO *et al.*, 2014) proposed an approach that adopts the Hierarchical Random Graph (HRG) model (CLAUSET *et al.*, 2006). A hierarchical network is naturally divided into groups and these groups themselves divide into subgroups, and so on until reaching the level of individual nodes. This structure is commonly represented as a dendrogram. The authors noted that estimating the connection probabilities between nodes reduced the noise scale imposed by DP. Then, while classical graph models are built based on the observed edges, the HRG uses the connection probabilities between nodes to create dendrograms.

Differently, some approaches focus on perturbing the original adjacency matrix of the graph through matrix perturbation strategies. An adjacency matrix is a type of graph representation in a tabular form, in which each cell (i, j) denotes the existence of an edge between nodes i and j. For this purpose, Chen et al. (CHEN *et al.*, 2014) designed a Density-based Exploration and Reconstruction (DER) mechanism to release the perturbed version of the adjacency matrix of the original graph. However, HRG and DER approaches are harmed regarding time complexity since they present quadratic time according to the number of nodes. Subsequently, Nguyen et al. (NGUYEN *et al.*, 2015) proposed the top-m filter (TmF) to overcome the scalability problems faced by the previous approaches. Its approach adds Laplace noise to each cell of the adjacency matrix and leverages the high-pass filtering technique (CORMODE *et al.*, 2012) to avoid the whole adjacency matrix manipulation. The authors proved that releasing a graph under DP using this approach has a time complexity upper bound of $O(\log n)$.

More recently, Iftikhar et al. (IFTIKHAR *et al.*, 2020) developed a microaggregation-based framework that perturbs the graph by adding noise to the distributions of the original graph. The framework works through a distance-constrained algorithm that approximates the *dK*-distributions of the graph via microaggregation. The authors demonstrated that the approach is robust enough to preserve the original graph's topological structures under different granularity levels and significantly reduce the amount of noise added to the released graph. Finally, Huang

et al. (HUANG *et al.*, 2020) proposed the Privacy Preserving Approach Based on Clustering and Noise (PBCN) method, which combines various algorithms to release noisy graphs under edge-DP. The algorithms that compose the PBCN method include pre-processing, clustering, degree sequence disturbing, reconstruction of nodes, post-processing, and more.

3.1.2 Local DP Approaches

Unlike the global DP, studies regarding the differentially private release of entire graphs within the local DP have only been deepened in recent years. For this purpose, we encounter a huge difference in the number of strategies that use the local DP compared to the ones that use the global DP.

Qin et al. proposed the (QIN *et al.*, 2017) LDPGen, a multi-phase technique for privately generating synthetic graphs. Its main idea consists of capturing the original graph's structure within an incremental process that clusters the users, based on their connections, to different partitions of the whole population. This process ensures that users with similar structures are clustered together. The method works by iteratively partitioning the nodes into groups and collecting information related to the node-to-group connectivity under LDP guarantees. Then, it clusters the nodes according to this information. Finally, once the clusters are defined, the LDPGen applies a graph generation model that uses these clusters to generate a private synthetic graph.

Gao et al. (GAO *et al.*, 2018) proposed a technique that generates a synthetic graph under LDP, which also adopted the Hierarchical Random Graph (HRG) model (CLAUSET *et al.*, 2006) to extract some graph features. The authors grouped the nodes with similar features by designing two heuristic methods. They also defined a novel notion of group-based local differential privacy according to the notion of 1-neighborhood local graphs that reduce the noise scale. Finally, the authors show that the grouping strategy ensures that each user within the same group is indistinguishable while the privacy level is enhanced without losing too much information.

3.2 Approaches for Attributed Graphs

When graphs have attributes attached to their edges or nodes, the aforementioned edge-DP and node-DP models may offer inappropriate privacy guarantees. Alternatively, a more

suitable setting consists of adapting the DP definition for the context of attributed graphs. In general, several types of attributes can be considered, such as boolean, categorical, or numerical (BENDIMERAD, 2019). Graphs with numerical attributes are commonly referred to as weighted graphs, while graphs with categorical attributes are referred to as attributed graphs. This section provides a detailed review of the latest techniques to ensure DP for edge-weighted, node-attributed, and edge-attribute graphs.

3.2.1 Edge-Weighted Graphs

The concept of differential privacy for weighted graphs was formally introduced by Sealfon (SEALFON, 2016). In this model, the graph topology is public, and only the edge weights are private. This approach is particularly relevant in scenarios like road networks, where the structure of the graph is fixed, and the sensitive information lies in the edge weights. Several recent works (SEALFON, 2016; FAN; LI, 2022; LI *et al.*, 2017; WANG; LONG, 2019; PINOT *et al.*, 2018; CHEN *et al.*, 2022) have considered this model to perform differentially private analysis over weighted graphs. These works are summarized below.

Sealfon (SEALFON, 2016) aimed to release weighted shortest paths and approximate distances between node pairs without revealing sensitive edge weights. He introduced theoretical foundations for weight differential privacy, assuming individuals influence edge weights. Two weight functions are considered neighbors if their l_1 distance is one or less, as stated in Definition 10.

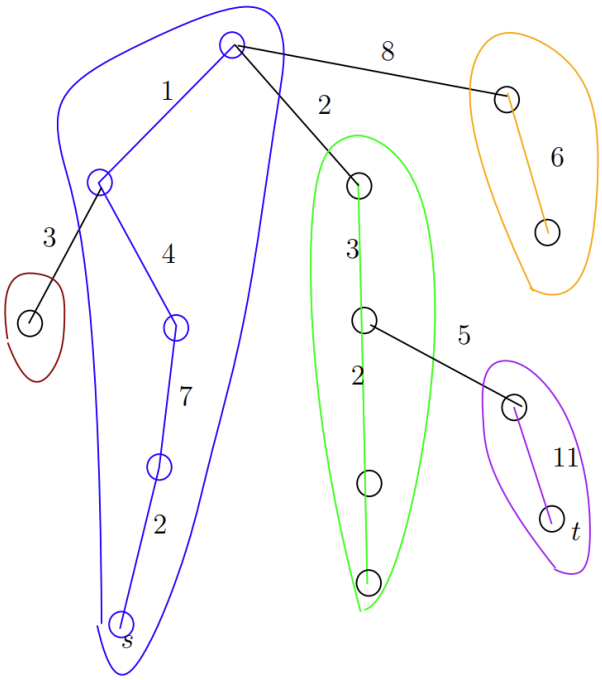
Definition 10. *(Neighboring Weight Functions (SEALFON, 2016)). Two weight functions ω, $\omega' : V^2 \rightarrow \mathbb{R}^+$ are neighboring, denoted $\omega \sim \omega'$, if:*

$$||\omega - \omega'||_1 = \sum_{u,v \in V} |\omega(u,v) - \omega'(u,v)| \leq 1. \tag{3.1}$$

For privately releasing shortest paths, Sealfon established a lower bound, showing it is impossible to release a path with less than $\Omega(|V|)$ additive error under differential privacy. He demonstrated that using the Laplace mechanism, an algorithm could approach this bound, with the released path's weight exceeding the optimal by at most $O(|V|\ log|V|)/\varepsilon$.

For releasing the all-pair shortest paths under DP, standard techniques yield an error of $O(|V|\ log|V|)/\varepsilon$ per query. Yet, Sealfon developed improved algorithms for special graph

Figure 11 – An example of a tree partitioned into three heavy paths. A unique color is assigned to every heavy path.



Source: (FAN; LI, 2022).

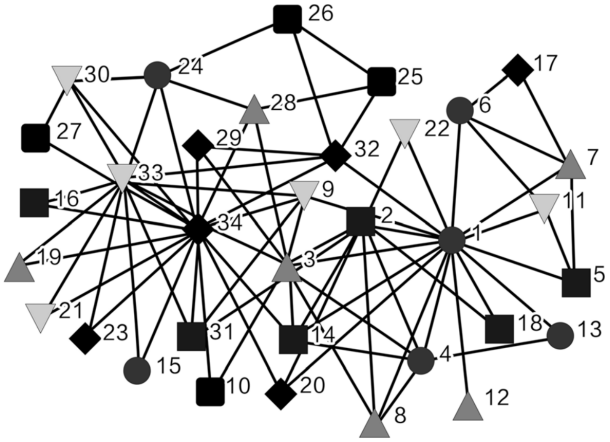
classes. For trees, he proposed a recursive algorithm with an error of $O(log^{2.5}|V|)/\varepsilon$. For bounded-weighted graphs, he showed that selecting a subset of vertices allows for estimating all-pair distances with relatively small errors.

Fan and Li (FAN; LI, 2022) revisited the problem of privately releasing approximate distances between all pairs of nodes and improved Sealfon's results. They divided a tree into disjoint heavy paths, where each non-leaf node selects the edge to its deepest child. This method decomposes the tree into several paths, as illustrated in Figure 11.

The authors showed that the unique path between any pair of vertices intersects at most $lov|V|$ heavy paths. They proved that releasing approximate all-pair distances is equivalent to handling multiple heavy path subqueries. For instance, in Figure 11, the shortest path between s and t is decomposed into sub-paths within disjoint heavy paths. For trees with depth h, they proposed a new algorithm that releases all-pair distances with an error of $O((log^{1.5}h)\cdot(log^{1.5}|V|)$, improving upon the previous error of $O(log^{2.5}|V|)$ (SEALFON, 2016).

Fan and Li also outperformed Sealfon's results for bounded-weighted graphs. They noted that some graphs, like those representing Manhattan's grid layout, can be divided into blocks. This division helps to separate the distances into three categories: within-block distances, boundary distances, and other distances. Then, the authors presented a method to release all-pair distances on general grid graphs with low error.

Figure 12 – An example of segmentation in a social network with 34 nodes and 78 edges.



Source: (WANG; LONG, 2019).

Li et al. (LI *et al.*, 2017) proposed the Merging Barrels and Consistency Inference (MBCI) approach for releasing weighted graphs under differential privacy guarantees. They proposed creating a histogram of edge weights, called merging barrels, to minimize the noise added to the weights. Instead of applying the Laplace mechanism directly to the weights, it is applied to groups of edges within the histogram.

The authors noted that merging all barrels with the same count into one group might violate differential privacy. Thus, to address this issue, they introduced a technique to achieve k-indistinguishability, ensuring that each group requires the same amount of noise. The groups satisfy k-indistinguishability for an integer ≥ 1 if there are at least k groups with the same number of barrels.

Finally, the authors also proposed an algorithm to preserve most of the shortest paths based on the original order of the edge-weight sequence. It is worth mentioning that this process is only based on the known order without accessing the private dataset, and, therefore, does not harm privacy.

Wang and Long (WANG; LONG, 2019) proposed the Lifted Merging Barrels and Consistency Inference (LMBCI) algorithm to reduce the error introduced by the MBCI (LI *et al.*, 2017) strategy. They segmented the original weighted graph into several sub-graphs without altering any edge weights. The segmentation process involves four main steps: (I) clustering nodes based on the number of common neighbors; (II) grouping nodes based on clustering results; (III) creating sub-graphs from the groups; and (IV) completing the sub-graph segmentation. An example of this process is shown in Figure 12, resulting in five sub-graphs.

Finally, the authors applied the MBCI algorithm to each sub-graph. Theoretical

analyses and experimental results indicated that LMBCI preserved most shortest paths and improved the accuracy of the published graph. However, both MBCI and LMBCI techniques introduce significant errors, achieving low errors only for $\varepsilon > 20$, which limits the efficiency of these approaches.

Pinot et al. (PINOT *et al.*, 2018) proposed PTClust, a differentially private method for node clustering in weighted graphs based on the Minimum Spanning Tree (MST) algorithm. The authors argued that the MST is an effective and intuitive way to summarize a graph and is useful for clustering non-convex shapes (GRYGORASH *et al.*, 2006). Also, they adapted the existing MST-based clustering algorithm, DBMSTClu (MORVAN *et al.*, 2017), to meet privacy requirements.

However, since the DBMSTClu algorithm takes weights only in the range of (0,1], the authors introduced the normalizing parameters τ and p to set lower and upper bounds for the weights. They developed a weight-release mechanism that normalizes the weights, transforms them to a new scale s, and adds noise from $Lap(0, s)$, proving it to be ε-differentially private.

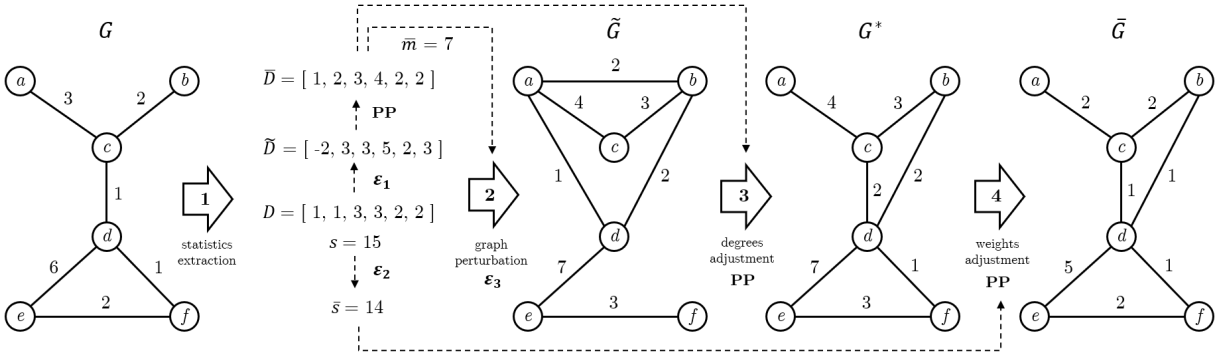
Finally, the PTClust solution involves generating a differentially private spanning tree topology, which releases randomized edge weights using the authors' weight-release mechanism. These weights are then input into the DBMSTClu algorithm to perform the clustering. The evaluations demonstrated that PTClust is robust enough to preserve the accuracy of the final clustering partition.

Chen et al. (CHEN *et al.*, 2022) also continued investigating privacy in weighted graph clustering, focusing on the k-median and k-center problems with weight-differential privacy. Their goal was to partition the vertices into k clusters to minimize the average and maximum distances between vertices and cluster centers.

For the k-median problem, the authors reformulated it as a submodular maximization problem (KRAUSE; GOLOVIN, 2014) and proposed a greedy differentially private algorithm with optimal approximation. They calculated the true shortest path distances and applied the exponential mechanism to the objective functions, which are indirectly determined by the edge weights. Additionally, the authors applied a sampling technique (MIRZASOLEIMAN *et al.*, 2015) to reduce the time complexity in large graphs.

For the k-center problem, the authors applied a similar approach and provided the best approximation guarantees with a greedy differentially private algorithm, improving the number of evaluations of the objective functions and adopting the same sampling technique.

Figure 13 – A running example of the proposed global approach.



Source: (BRITO *et al.*, 2023).

Differently from previous works, Brito et al. (BRITO *et al.*, 2023) assume that graph topology is not known. For some real-world applications, the assumption that the graph topology is public is misleading, and the existing works may not be effective in providing the desired privacy guarantees. To address this limitation, the authors propose a novel definition for neighboring weighted graphs with unknown topology, as stated in Definition 11.

Definition 11. *(Neighboring Weight Functions with Unknown Topology (BRITO et al., 2023)). Two weight functions ω, $\omega' : V^2 \rightarrow \mathbb{Z}_{\geq 0}$ are neighboring, denoted $\omega \sim \omega'$, if:*

$$||\omega - \omega'||_1 := \sum_{u,v \in V} |\omega(u,v) - \omega'(u,v)| = 1. \tag{3.2}$$

Therefore, the authors proposed a global and local approach to release weighted graphs via DP while keeping the graph topology and edge weights private. Several techniques, such as priority sampling and post-processing methods, were applied along the process to preserve the original node degrees and the sum of all edge weights as much as possible. An extensive experimental analysis demonstrated that the proposed approaches outperform the state-of-the-art in terms of utility and performance. Figure 13 depicts a running example of the proposed global approach.

3.2.2 Node-Attributed Graphs

Jorgensen et al. (JORGENSEN *et al.*, 2016) proposed the TriCycLe model for releasing synthetic node-attributed graphs, i.e., graphs with attributes in the nodes, under DP guarantees. TriCycLe is an extension of the Attributed Graph Model (AGM) (III *et al.*, 2014). In summary, the AGM models a graph using three sets of parameters that describe (I) the distribution

of the attributes over the nodes, (II) the correlations between node attributes and edges, and (III) the modeling parameters for an underlying generative structural model. These set of parameters are denoted as Θ_X, Θ_F, and Θ_M, respectively. The third parameter Θ_M contains the structural information, such as the degree sequence, the number of triangles, etc, of the graph that will be generated according to the generative random graph model. The generative random graph model could be any, including the existing in the literature like Chung-Lu (III *et al.*, 2012) and Erdős-Rényi (SESHADHRI *et al.*, 2012). It is important to note that the standard AGM does not provide any privacy guarantee. Then, TriCycLe appears by introducing privacy in its graph-generating process. However, since DP graph models were formerly proposed for graphs without attributes in the edges and nodes, a novel definition of neighboring graphs had to be established to attend DP for this kind of graph. The authors defined the notion of Edge-Adjacent Attributed Graphs, which is formally stated below:

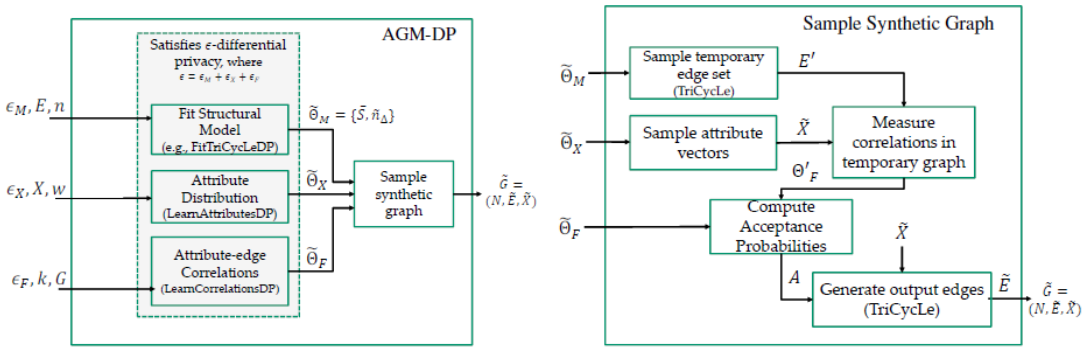
Definition 12. *(Edge-Adjacent Attributed Graphs (JORGENSEN et al., 2016)). Two attributed graphs G, G' are said to be edge-adjacent (or neighboring) if they differ in the presence of a single edge or in the attribute vector associated with a single node.*

An attribute vector consists of a vector that holds the attribute values of each user. Then, the authors could design TriCycLe within a differentially private mechanism according to this neighboring definition. In summary, TriCycLe collects information about the original graphs through various differentially private processes. First, it adds Laplace noise to the Θ_X and Θ_F parameters. It is important to mention that the Θ_F parameter is obtained after performing a projection in the original graph to reduce the sensitivity and, hence, the noise scale factor. Then, the Θ_M parameters are composed of the degree sequence and the number of triangles. The degree sequence is obtained by adding Laplace noise. However, since the number of triangles is a much more complex query, leading to extremely high sensitivity, the author applied the Ladder mechanism (ZHANG *et al.*, 2015), which is based on the local sensitivity, providing a perturbed version of the number of triangles with elevated accuracy. Finally, once these parameters are retrieved through DP mechanisms, TriCycLe generates a synthetic graph according to these parameters, rewiring the graph connections until the graph structure reaches a disposition similar to the expected by Θ_X, Θ_F, and Θ_M according to an acceptance probability. The whole process performed by TriCycLe ensures ϵ-DP. Although the approach is robust enough to provide highly accurate synthetic graphs for some metrics, it focuses on performing better in the metrics related

to the attributes that form the Θ_M parameter, i.e., the degree sequence and the number of triangles. Then, it may not perform very well for metrics of different nature.

Figure 14 presents the overview of the TriCycLe model. On the left side, we encounter the steps for computing the AGM-DP, which consists of computing the Θ_X, Θ_F, and Θ_M in a differentially private manner through a DP mechanism. Then, these parameters are used to sample a private synthetic graph. The steps of sampling a synthetic graph are presented on the right side.

Figure 14 – Workflow of the TriCycLe model.



Source: (JORGENSEN *et al.*, 2016).

In the same context, Chen et al. (CHEN *et al.*, 2020) proposed the Community-Preserving Attributed Graph Model (C-AGM), also based on the AGM and in the same neighboring definition proposed in (JORGENSEN *et al.*, 2016). Since this approach uses the same neighboring definition, it also lies on node-attributed graphs. The main objective of the C-AGM model consists of preserving as much information as possible about the communities of the original graph in the released synthetic graph. The main idea of the C-AGM model is to capture the properties of the communities of the node-attributed graph. Each existing community in the original graph is denoted as a community partition C. Then, the C-AGM model captures a variety of properties regarding the community partitions. These properties include:

1. The number and the size of the communities;
2. The number of intra-community edges in every community;
3. The number of inter-community edges;
4. The distributions of the attribute vectors in every community;
5. The distribution of the attribute-edge correlations (the same as (JORGENSEN *et al.*, 2016)) for the set of inter-community edges and the set of intra-community edges in every community.

Then the authors computes the AGM parameters Θ_X, Θ_F, and Θ_M for every commu-

nity partition C. The rest of the approach works similarly to the TriCycLe model. The recently computed parameters are used to generate a private synthetic graph. Also, similarly to the TriCycLe model, the results obtained by the C-AGM model are better for a specific variety of metrics. In this case, the method was mainly proposed to maintain as much information as possible about the communities and the clustering coefficients. Then, it also suffers from accuracy problems for some metrics. It is important to mention that the C-AGM model ensures ϵ-DP since all its steps are made through differentially private mechanisms.

Differently from the previous approaches, Wei et al. (WEI *et al.*, 2020) proposed a local approach for releasing synthetic node-attributed graphs called AsgLDP. To the best of our knowledge, it is the first and unique approach that tackles the problem of releasing node-attributed graphs under LDP guarantees. AsgLDP is a two-phase framework based on the edge LDP 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.

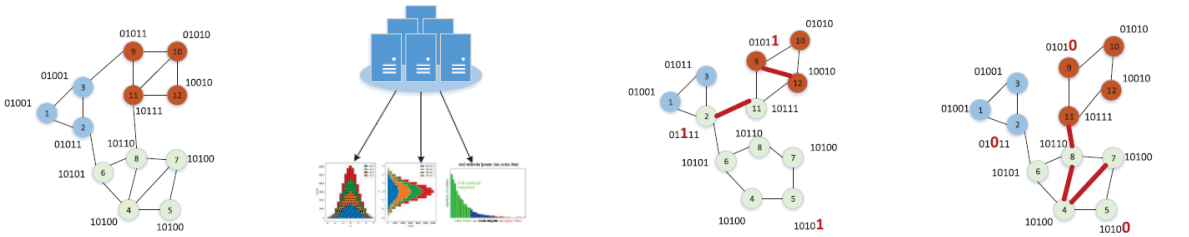
The properties that each user reports are the degree and the Randomized Attribute List (RAL). The RNL is similar to the Randomized Neighbor List (RNL) proposed in (QIN *et al.*, 2017). The RNL was defined in the context of general graphs, where given a graph $G = (V, E)$, the edges E wire nodes from the same set V. The RNL consists of a n-length bit vector, where n is the number of nodes (users) in V, i.e. $n = |V|$. In summary, the RNL of a user i is given by $\text{RNL}_i = [x_{i,j}, ..., x_{i,|V|}]$, where $x_{i,j} \in \{0, 1\}$ denotes whether exists the edge $e_{i,j} \in E$, i.e., if exists a connection between users v_i and v_j. Then, the RAL structure is an adaptation of the RNL structure, which is adequate for the problem of releasing node-attributed graphs. The RAL consists of a w-length bit vector, where w is the number of attributes. In summary, the RAL of a user i is given by $\text{RAL}_i = [x_{i,1}, ..., x_{i,w}]$, where $x_{i,j} \in \{0, 1\}$ denotes the value of the j-th attribute of the user i.

Both properties are reported through LDP mechanisms. However, the degree suffers from the dimensionality problem, i.e., the bit vector that reports the degree is defined according to the number of nodes, which leads to excessive noise addition. To solve this problem, the author proposed the Random Jump (RJ) method to perturb the degree in a decentralized manner to reduce the noise while still satisfying the LDP properties. The main idea of the RJ algorithm is that it considers that not all users agreed to report their data, so the number of users that agreed to report their data, denoted as n_v, is smaller than the initial number of users. Then, the RJ uses this

number of users n_v combined with the Generalized Randomized Response (GRR) (WARNER, 1965) to perturb the user degree.

Finally, the data collector performs an unbiased estimation of the reported data to estimate the degree distribution and the attribute joint distribution. With this distribution, the collector samples an initial synthetic graph, called a seed graph, that respects these distributions using the accept-reject sampling method (LIANG *et al.*, 2011). Yet, the collector clusters the seed graph to detect edge and attribute anomalies to optimize the graph further. The optimization step consists of keeping the nodes more closely related in the same community than those outside the community. The author demonstrated that the approach presents good results for some metrics, especially the ones related to the communities and the clustering coefficients of the graph.

Figure 15 – Workflow of the AsgLDP framework.



(a) Initial node-attributed graph. (b) Unbiased information collection. (c) Seed graph creation. (d) Seeg graph optimization.

Source: (WEI *et al.*, 2020).

Figure 15 presents the overview of the AsgLDP framework model from the original graph until the release of the differentially private graph after being optimized.

3.2.3 Edge-Attributed Graphs

Differently from all the previously presented approaches, Liu et al. (LIU *et al.*, 2020) presented the PrivAG, a framework that tackles the problem of DP in edge-attributed graphs, i.e., the attributes belong to the edge instead of the node. Also, the authors release only a set of statistics instead of an entire synthetic graph. Additionally, to the best of our knowledge, this is the first and unique work that investigates the privacy concerns in edge-attribute graphs. Similarly to the TriCycLe approach (JORGENSEN *et al.*, 2016), the authors designed a novel neighboring definition for the LDP model to address the problem of edge-attributed graphs. The authors defined the Attribute-wise Local Differential Privacy, which is formally stated below:

Definition 13. *(Attribute-wise Local Differential Privacy (LIU et al., 2020)). A randomized algo-*

rithm $\mathcal{A}$ satisfies ϵ-attribute-wise local differential privacy, if and only if for any two neighboring attributed local graph data G, G' differing in one attribute and related edges and for any output $O \in Range(\mathcal{A})$,

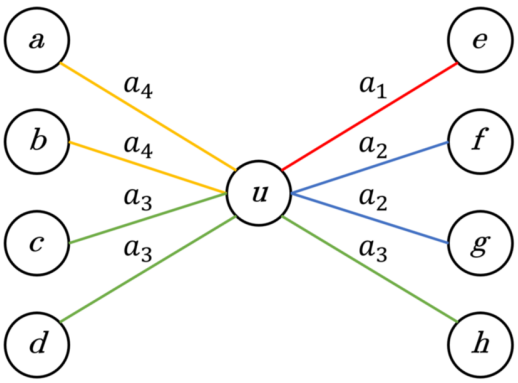
$$Pr[\mathcal{A}(G) = O] \leq e^{\varepsilon} Pr[\mathcal{A}(G') = O]. \tag{3.3}$$

The authors ensure this notion of DP by reporting the properties of the local graph under LDP guarantees. The properties consist of the edge attributes in the local graph and the degree of each edge attribute, i.e., the number of edges with a given attribute. These properties are denoted as attribute vectors and attribute-degree vectors, respectively. Both properties are encoded into a bit vector. However, the LDP mechanism applied to each property is different. Given an d-length attribute vector and a value $k \leq m$, where m is the number of attributes, only a k-length attribute vector is considered. Then, the exponential mechanism is applied to sample a perturbed version of the k-length initial attribute vector according to a utility function. With respect to the edge-degree vectors, the degree of each attribute present in the k-length attribute vector is reported through a θ-length vector, where θ is a truncation parameter over the graph G that limits the attribute degrees. This whole process ensures $(\epsilon_1 + \epsilon_2)$-LDP, where ϵ_1 is the privacy budget used to report the attribute vector and ϵ_2 is the privacy budget used to report each attribute-degree vector.

Once the data collector receives the data, it estimates the properties: the attributes and their degrees. However, after estimating the counts and the frequencies, the aggregator may face some inconsistencies between the attribute counts and the reported degree counts, which is fixed through an optimization step. Finally, data analyses are performed. However, the scope of the analyses is very limited since the approach does not release an entire graph for general-purpose analysis. Instead, only two analyses are performed: the attribute frequency estimation and the attribute-degree distribution estimation.

Figure 16 presents a local graph G of a supposed user ‘‘u’’, composed of 8 neighbor nodes and 8 edges with their corresponding attributes. On its side, we encounter the graph properties reported to the data collector, i.e., the attribute vector and the attribute-degree vectors, both already exhibited in the encoded form.

Figure 16 – An example of a local graph followed by its encoded information in the form of attribute vector and attribute-degree vectors.



attribute vector

a_1	a_2	a_3	a_4	a_5
1	1	1	1	0

attribute-degree vector

	$d=1$	$d=2$	$d=3$	$d=4$
a_1	1	0	0	0
a_2	0	1	0	0
a_3	0	0	1	0
a_4	0	1	0	0
a_5	0	0	0	0

Source: Adapted from (LIU *et al.*, 2020).

3.3 Summary

Once detailing a variety of studies of DP in the field of edge-DP and attributed graphs, we finish this section by presenting Table 1, which summarizes these existing works. It compares the purpose of release, the differentially private graph model, the DP setting, and the graph type under analysis.

Table 1 – Summary of the existing works.

Work	**Purpose of Release**	**Graph DP Model**	**DP Setting**	**Graph Type**
(SALA *et al.*, 2011)	Entire graph	Edge-DP	Global	Normal
(WANG; WU, 2013)	Entire graph	Edge-DP	Global	Normal
(XIAO *et al.*, 2014)	Entire graph	Edge-DP	Global	Normal
(CHEN *et al.*, 2014)	Entire graph	Edge-DP	Global	Normal
(NGUYEN *et al.*, 2015)	Entire graph	Edge-DP	Global	Normal
(IFTIKHAR *et al.*, 2020)	Entire graph	Edge-DP	Global	Normal
(HUANG *et al.*, 2020)	Entire graph	Edge-DP	Global	Normal
(QIN *et al.*, 2017)	Entire graph	Edge-DP	Local	Normal
(GAO *et al.*, 2018)	Entire graph	Edge-DP	Local	Normal
(SEALFON, 2016)	Graph statistics	Edge-Weight-DP	Global	Edge-Weighted
(LI *et al.*, 2017)	Graph statistics	Edge-Weight-DP	Global	Edge-Weighted
(PINOT *et al.*, 2018)	Graph statistics	Edge-Weight-DP	Global	Edge-Weighted
(WANG; LONG, 2019)	Graph statistics	Edge-Weight-DP	Global	Edge-Weighted
(FAN; LI, 2022)	Graph statistics	Edge-Weight-DP	Global	Edge-Weighted
(CHEN *et al.*, 2022)	Graph statistics	Edge-Weight-DP	Global	Edge-Weighted
(BRITO *et al.*, 2023)	Entire graph	Edge-Weight-DP	Global/Local	Edge-Weighted
(WEI *et al.*, 2020)	Entire graph	Edge-DP	Local	Node-Attributed
(JORGENSEN *et al.*, 2016)	Entire graph	Edge-Adjacent-DP	Global	Node-Attributed
(CHEN *et al.*, 2020)	Entire graph	Edge-Adjacent-DP	Global	Node-Attributed
(LIU *et al.*, 2020)	Graph statistics	Attribute-wise-DP	Local	Edge-Attributed
This thesis	*Entire graph*	*Edge-DP*	*Local*	*Edge-Attributed*

Several studies (SALA *et al.*, 2011; WANG; WU, 2013; XIAO *et al.*, 2014; CHEN *et al.*, 2014; NGUYEN *et al.*, 2015; IFTIKHAR *et al.*, 2020; HUANG *et al.*, 2020; QIN *et al.*,

2017; GAO *et al.*, 2018) have been made with either global DP or local DP to release a synthetic graph under edge-DP guarantees. However, none of them consider the existence of attributes in the graph structure.

In the context of graphs with attributes in its structure, (SEALFON, 2016; LI *et al.*, 2017; PINOT *et al.*, 2018; WANG; LONG, 2019; FAN; LI, 2022; CHEN *et al.*, 2022; BRITO *et al.*, 2023) proposed approaches for releasing either graph statistics or the entire graph for weighted graphs under multiple scenarios, through global and local DP settings, and for known and unknown graph topologies. Yet, Wei et al. (WEI *et al.*, 2020) proposed the AsgLDP framework to privately release node-attributed graphs under the LDP model, which may not provide the desirable level of privacy to attributed graphs. Jorgensen et al. (JORGENSEN *et al.*, 2016) and Chen et al. (CHEN *et al.*, 2020) proposed the TriCycLe and the C-AGM frameworks, respectively, for releasing attribute graphs under a novel neighboring notion called Edge-Adjacent Graphs within the global DP model. This neighboring definition considers that two graphs are neighbors if they differ in the presence of a single edge or the attribute vector associated with a single node. However, the privacy subject under this notion is the nodes' edges and attributes, which do not apply to edge-attributed graphs.

Finally, Liu et al. (LIU *et al.*, 2020) proposed the PrivAG framework under a novel neighboring definition for the LDP model. The authors defined the Attribute-wise Differential Privacy notion, which considers that two graphs are neighbors if they differ in one attribute and all related edges associated with this attribute. Although the privacy subject is the attributed edges, this approach may cause severe data distortion since, in the worst case, it will be equivalent to a node-DP notion. Additionally, PrivAG suffers from a limited range of analysis since the framework does not release an entire synthetic graph. Instead, it releases only specific graph statistics.

Differently, our work proposes PEG, a decentralized dynamic degree-based clustering approach designed for privately releasing edge-attributed graphs under the notion of edge-LDP. Our approach combines the characteristics of a novel encoding structure, called RANL, and the clustering to improve the utility of the released edge-attributed graphs.

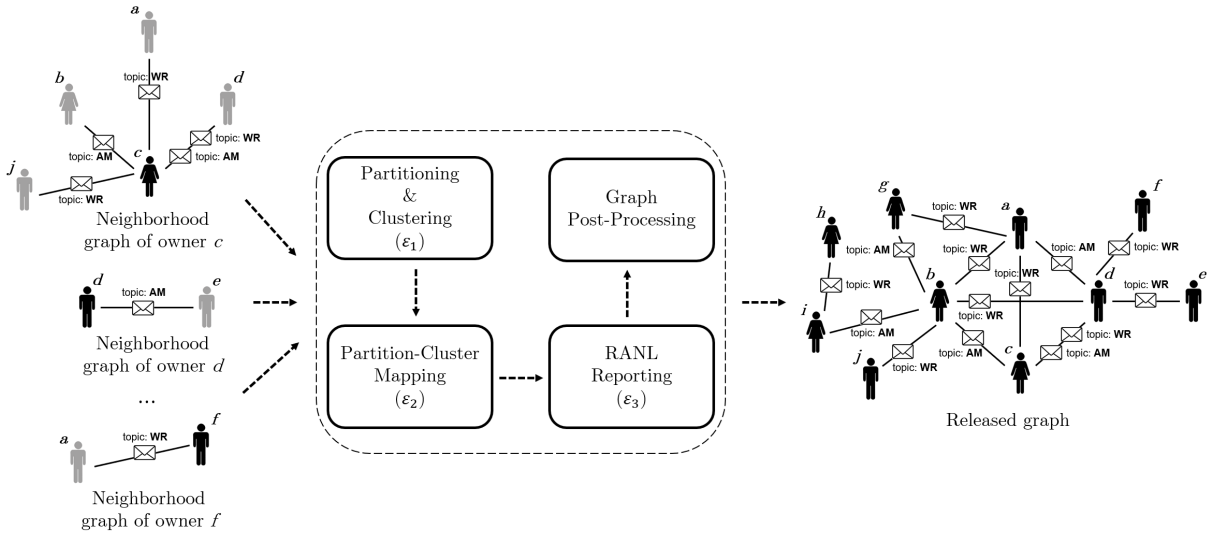
4 THE PEG APPROACH

In this chapter, we present PEG (**P**rivacy for **E**dge-attributed **G**raphs), our approach for releasing edge-attributed graphs under local differential privacy guarantees while maintaining the private graph useful for several analytics tasks. PEG is a multiphase approach in which each user is responsible for privatizing their data locally using local differential privacy before sharing it with the data curator. PEG is divided into four main phases: (i) *Partitioning & Clustering*; (ii) *Partition-Cluster Mapping*; (iii) *RANL Reporting*; and (iv) *Graph Post-Processing*.

However, before explaining each phase, we introduce the Randomized Attribute Neighbor List (RANL), a novel data structure for encoding edge-attribute graphs in the LDP setting. The RANL is one of the main contributions of this thesis and is crucial for a complete understanding of PEG.

Figure 17 presents an overview of the pipeline of the PEG approach. PEG receives the users' local graph as input and outputs a perturbed version of the original graph containing all the users. Note that only the three initial phases consume the privacy budget since the graph post-processing phase only modifies the private data, which does not compromise the users' privacy. Detailed information regarding the RANL and each phase of PEG will be presented below.

Figure 17 – An overview of the pipeline of the PEG approach.



Source: Elaborated by the author.

4.1 Randomized Attribute Neighbor List

Some approaches have already been proposed to encode graph information under the local DP setting, such as the Randomized Neighbor List (RNL) (QIN *et al.*, 2017) and the Randomized Attribute List (RAL) (WEI *et al.*, 2020). In summary, the RNL is applied to encode the users' local neighborhoods in general graphs, i.e., composed only of nodes and edges, without any additional information, like numerical or categorical, in either nodes or edges. In turn, the RAL is applied to encode users' attributes in the context of node-attributed graphs. However, none of these strategies are suitable for encoding the users' information, i.e., the relationships along with their attributes, in the context of edge-attributed graphs. For this purpose, we propose the Randomized Attribute Neighbor List (RANL).

The Randomized Attribute Neighbor List (RANL) consists of an encoding structure through which each user can report its neighborhood locally, i.e., all the edges and their attributes that form the user's local graph. The RANL combines the key features of the Randomized Neighbor List (RNL) and the Randomized Attribute List (RAL) encoding methods to suit our context of edge-attributed graphs. 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 (QIN *et al.*, 2017; WEI *et al.*, 2020) 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 14. *(Randomized Attribute Neighbor List (RANL)). Given an edge-attributed 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 attribute 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 attribute $x_k \in X$.*

Given a privacy budget ε within the RR protocol, the process of perturbing and

Figure 18 – An example of the RANLs of the users v_d and v_e in the edge-attributed graph G depicted in Figure 1.

RANL_{v_d}

v_a		v_b		v_c		v_d		v_e		v_f		v_g		v_h		v_i		v_j	
AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR
1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0

RANL_{v_e}

v_a		v_b		v_c		v_d		v_e		v_f		v_g		v_h		v_i		v_j	
AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR	AM	WR
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Source: Elaborated by the author.

reporting the users' RANLs with probabilities p and q satisfies ϵ-edge-LDP since adding or removing a single attributed 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. 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 long, and the list contains significantly more zero values than values equal to one. 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 h 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.

Figure 18 presents an example of the RANLs of the users v_d and v_e in the edge-attributed graph G depicted in Figure 1. Note that this graph has 10 nodes ($n = 10$) and two possible edge attributes ($t = 2$), given by ''AM'' and ''WR''. Therefore, the length h of the presented RANLs is given by $h = n \cdot t = 20$. Only the array positions that correspond to a user's existing edges in G are assigned to 1, while the remaining positions are assigned to 0.

4.2 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 know about any user, except that each user is identified by a random identifier. In the cases where disjoint sets could not be of the same size (due to particularities of the values of n and p), consider $|P_j| = \lfloor \frac{n}{k} \rfloor \; \forall_{P_j \in \mathcal{P}}$, except for one of the partitions that will be chosen to accommodate the remaining users, given by (n mod 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 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-attributed 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 attribute. We detail these different notions of degree as follows.

Edge Property 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 attribute $x_k \in X$ in an edge-attributed graph $G = (V, E, X)$. If a node v_i is connected to a node v_j and an attribute x_k is associated to this connection, i.e., $e_{i,j,k} \in E$, then $y_{i,j}^k = 1$, otherwise $y_{i,j}^k = 0$. We define $d_{v_i}^{x_k}$ as the edge property degree of a node v_i with property given by $\sum_{j=1}^{n} y_{i,j}^k$. In summary, in an undirected graph, the edge property degree represents the number of edges associated with a specific attribute 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 property degree vector of a node v_i.

Node Degree. Given an edge-attributed 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 (GHOSH *et al.*, 2009) 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 (YE *et al.*, 2022).

To prove that the sensitivity of the degree function is 2, consider that the data curator desires the degrees of two nodes v_i, v_j that report their degrees independently, such that $v_i, v_j \in V$, $v_i \neq v_j$, but v_i and v_j share the same edge. Given that sharing the existence or absence of that edge will contribute to both d_{v_i} and d_{v_j}. In the most extreme case, where there are only nodes v_i and v_j, and only one edge that connects them in G, $d_{v_i} = 1$ and $d_{v_j} = 1$ indicate the existence of this edge. If this edge is removed, both d_{v_i} and d_{v_j} will decrease by 1, causing the sensitivity of the node degree function to be 2. As DP and LDP consider that an adversary may possess any background knowledge, we must consider the extreme case in which the data curator already knows all edges except this one.

However, instead of requesting users to report only their degrees, our approach captures their edge property degree vectors. The edge property degree vector holds much more relevant information than merely the node degree. It holds information about the node degree per edge property while also allowing us to derive the original node degree by summing up the edge property degrees.

Instead of sharing the degrees with the data curator, each user $v_i \in V$ shares a perturbed version of their edge property degree vectors δ_{v_i}, given by $\tilde{\delta}_{v_i} = (d_{v_i}^{x_1} + Geom(\frac{\varepsilon_1}{2}), \ldots, d_{v_i}^{x_t} + Geom(\frac{\varepsilon_1}{2}))$. Since the edges related to each edge property degree are non-overlapping, adding or removing one edge from a user would change one, and only one, edge property degree of δ_{v_i} by 1. Therefore, by the DP parallel composition property, sharing the perturbed edge property degree vector $\tilde{\delta}_{v_i}$ still satisfies ε_1-edge-LDP.

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 degree property has been queried through the geometric mechanism, where the noise sample can assume positive or negative values, the original edge property 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. Note that the post-processing property of DP (DWORK; ROTH, 2014) ensures that any function can further modify any data perturbed through a DP or LDP mechanism without harming the users' privacy.

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 property degrees. This process consists of

calculating the expected edge property degrees sum of each edge property and, then, using this information to adjust the edge property degrees such that each edge property degree will have a value higher than zero and the sum of the adjusted edge property degrees will be the same as the expected edge property degrees sum. For instance, consider $\tilde{s}_k$ as the expected edge property degrees sum of the edges attributed 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 property 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 property 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 $\bar{\phi}_{v_i} = \max(1, \bar{d}_{v_i})$.

Finally, after collecting and adjusting the users' node degrees, the data curator sorts these degrees in descending order and groups the users into c clusters according to their corresponding degrees. This approach ensures that users with similar degrees are grouped in the same cluster. However, the users are not grouped into clusters of the same size. Instead, the criteria for determining the size of each cluster is to ensure that each cluster has a similar degree mass. The degree mass of a cluster consists of the sum of the node degrees of its belonging users.

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} by dividing the $s_{\bar{\phi}}$ by the number of desired clusters c, such that $s_{max} = \frac{s_{\bar{\phi}}}{c}$. 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, in some cases, it may occur that some clusters could not reach the exact degree mass of s_{max}. As a consequence, the last cluster may have to accommodate more users than expected to compensate for the underutilized degree mass by the other clusters, causing its degree mass to surpass the s_{max}. Example 1 illustrates the Partitioning & Clustering phase.

Example 1. *Initially, consider the edge-attributed graph G in Figure 1 with* $n = 10$*, where* $V = \{v_a, \ldots, v_j\}$*. Figure 19 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 equal to* $\frac{n}{p} = 5$*, meaning that*

Figure 19 – Partitioning & Clustering phase.

	✉WR	✉AM	Total
a	3	2	5
b	2	3	5
c	3	2	5
d	3	2	5
e	1	0	1
f	1	0	1
g	0	1	1
h	0	1	1
i	0	1	1
j	1	0	1

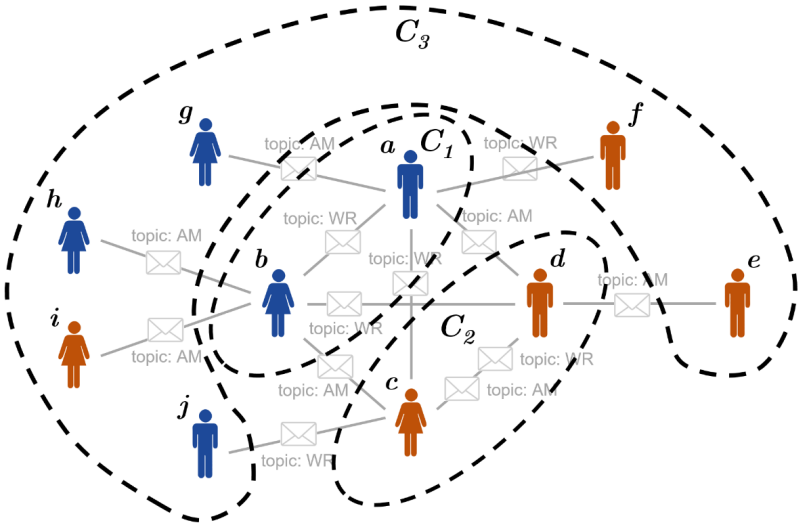
ε_1 geometric mechanism

	✉WR	✉AM	Total
	3	1	4
	1	3	4
	2	2	4
	3	1	4
	1	0	1
	0	0	1
	1	1	2
	1	0	1
	1	1	2
	0	0	1

clustering

$\frac{\sum degrees}{\#\ clusters} = \frac{24}{3} = 8$

a	4	C_1
b	4	
c	4	C_2
d	4	
g	2	C_3
i	2	
e	1	
f	1	
h	1	
j	1	



Source: Elaborated by the author.

all users were perfectly allocated into partitions of the same size. Since the users of each partition are selected randomly, a possible partition set $\mathcal{P}$ *is given by* $\mathcal{P} = \{P_1, P_2\}$*, where* $P_1 = \{v_a, v_b, v_g, v_h, v_j\}$ *and* $P_2 = \{v_c, v_d, v_e, v_f, v_i\}$*, such that users of partition* P_1 *were marked with blue nodes, while users of partition* P_2 *were marked with orange nodes. Now, consider that the data curator desires to cluster the users into* $c = 3$ *groups. First, each user reports its edge property 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 property 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* $s_{max} = \frac{s_{\tilde{\phi}}}{c} = \frac{24}{3} = 8$*, 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.*

4.3 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 allows us to perform an unbiased estimation of noisy counts, enabling us to infer the majority clusters with high fidelity and assign consistent clusters to the partition.

The process of choosing the partitions' clusters is done privately through an LDP mechanism. In this case, the OUE is a suitable protocol since it is based on the unary encoding principle, where each user's d-length bit vector will contain only one bit signed with one. Also, the OUE parameters p and q are optimized to maintain as much information as possible in the transferred information. Since only one bit is set to one and the remaining $d - 1$ bits are set to zero, the OUE utilizes probability values of p and q that maximize the number of bits reported as zero that were initially zero.

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.

After that, the data curator calculates the counts of each cluster by summing up the bits of each vector cluster-wise. We denote $\tilde{count}_{P_j}^{C_k} = \sum_{v_i \in p_j} B_{v_i}[C_k]$ the perturbed count of the cluster C_k in partition P_j, where $B_{v_i}[C_k]$ is the bit in the vector of the user v_i that states the presence, or absence, of v_i in C_k. However, simply summing the perturbed bits does not reflect the real counts since the perturbed bit vectors may contain more than one bit marked as one after being randomized. Then, an unbiased estimation is applied to eliminate bias and obtain counts that are closer to the actual values. We denote $\bar{count}_{P_j}^{C_k} = \frac{\tilde{count}_{P_j}^{C_k} - (q \cdot |P_j|)}{p \cdot q}$ 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 $\bar{count}_{p_j}^{C_k}$ 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.

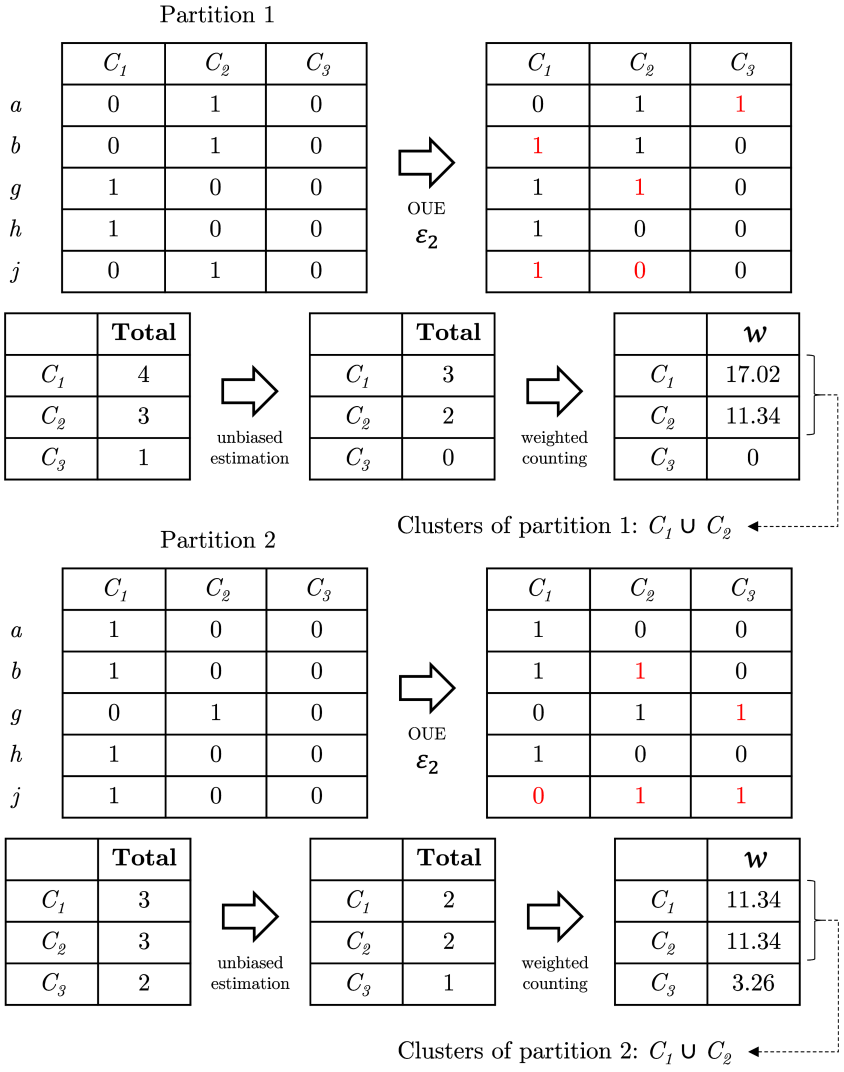
Once the counts are estimated, the data curator can assign the partition to a cluster. Choosing the cluster with the highest count may not be suitable for long-tailed degree distribution datasets, as this often results in selecting the least dense cluster formed by many users with the lowest degrees. Such clusters do not adequately reflect the graph's dominant relationships. To overcome this issue, we propose a weighting function to adjust the estimated counts based on the density of each cluster, combined with a percentile selection method. In Equation 4.1, we present the weighting function, where $w_{P_j}^{C_k}$ 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 clusters from gaining additional advantages based on their size, ensuring that a slightly larger cluster with a significantly higher estimated count still achieves a higher weighted count.

$$w_{P_j}^{C_k} = \bar{count}_{P_j}^{C_k} \cdot \sqrt{\frac{s_{C_k}}{|C_k|}} \tag{4.1}$$

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 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-attributed graph G in Figure 1 and the partitions $\mathcal{P}$ and clusters C in Figure 19, respectively. Suppose that the data curator desires to define the clusters of each partition according to the 50th percentile. Figure 20 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 4.1. 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.*

Figure 20 – Partition-Cluster Mapping phase.



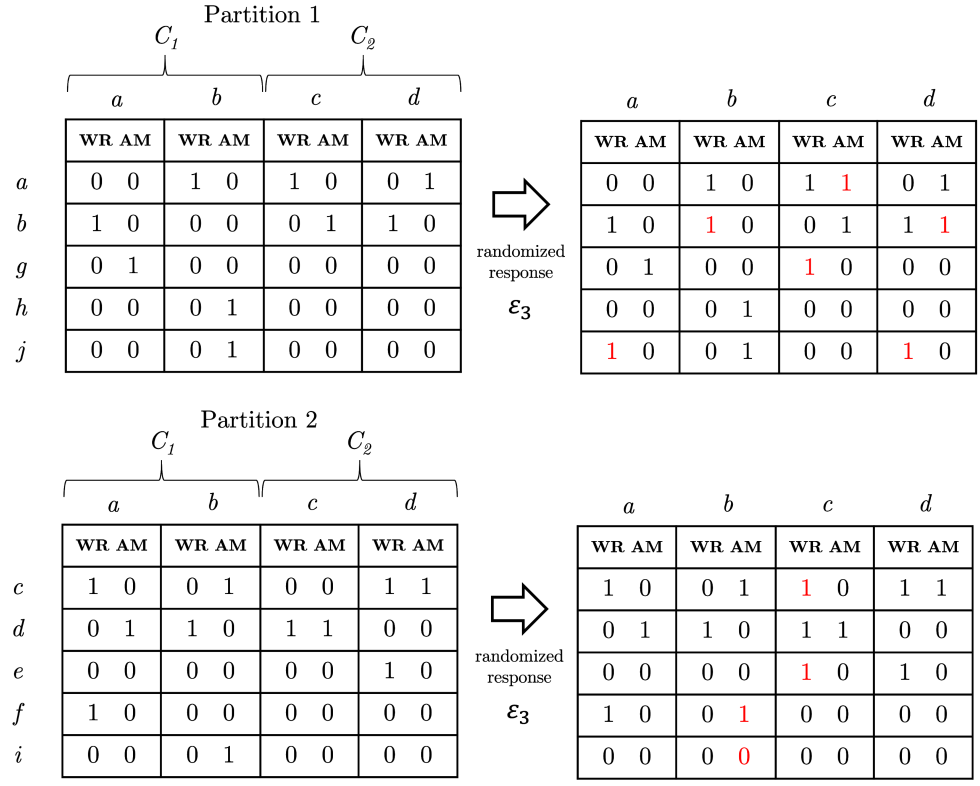
Source: Elaborated by the author.

4.4 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 attributes that form the 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

Figure 21 – RANL Reporting phase.



Source: Elaborated by the author.

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-attributed graph G in Figure 1, the partitions $\mathcal{P}$ and clusters C in Figure 19, and their respective partition-cluster mapping in Figure 20. Figure 21 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 $C_1 \cup C_2$.*

4.5 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 Property Degrees Adjustment*; and (iii) *Disconnected Nodes Rewiring*.

4.5.1 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-attributed 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 and an edge property x_k. The edge $e_{i,j,k}$ is only considered to exist if $RANL_{v_i}[e_{i,j,k}] = RANL_{v_j}[e_{j,i,k}] = 1$. Otherwise, if the edge is present only in one of these RANLs, the edge is removed from the released private graph G'.

This double-check is essential for maintaining the graph's consistency and improving the data utility since we are dealing with undirected graphs. Then, we have to ensure that both related nodes report 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 involved nodes.

4.5.2 Edge Property Degrees Adjustment.

In this stage, we use the noisy edge property degrees obtained in the first phase of PEG (Section 4.2) to adjust the users' edge property degrees according to the noisy information by randomly adding and removing edges as necessary. This adjustment is necessary because the RR protocol tends to add extra edges to the users' RANLs and, consequently, to the perturbed graph G'.

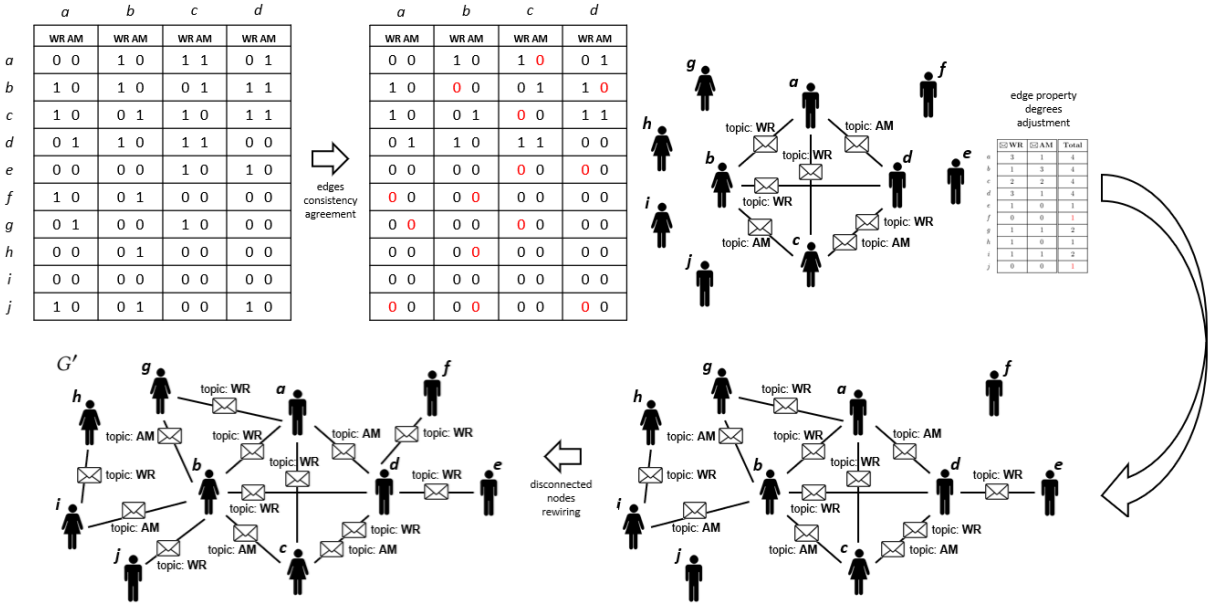
4.5.3 Disconnected Nodes Rewiring.

It is worth mentioning that some users may present all edge property degrees equal to zero after reporting it to the data curator. This is expected to happen especially when facing datasets with long-tailed degree distribution, where an expressive percentage of the users have degrees closer to zero. In these cases, some 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 property 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 property x_k is sampled proportionally to the edge properties present in G'.

Example 4 illustrates this post-processing step.

Figure 22 – Post-Processing phase.



Source: Elaborated by the author.

Example 4. *Consider the users RANLs reported in Figure 21. Figure 22 presents how the graph is created through the RANLs as well as how the post-processing step is applied. We exemplify the application of the post-processing steps from the edges consistency agreement until the disconnected nodes rewiring, when the perturbed edge-attributed graph* $G' = (V, E', X)$ *is released.*

4.6 The PEG Algorithm

The PEG algorithm[1], detailed in Algorithm 1, expects as input an edge-attributed graph $G = (V, E, X)$, the number of partitions p, the number of clusters c, the y-th percentile value p_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 property degrees through the geometric mechanism, with $\alpha = \frac{\varepsilon_1}{2}$. In lines 4-10, the data curator aggregates and estimates the users' noisy edge property 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

[1] The source code and other artifacts are available at https://github.com/andreluiscm/peg-ldp

Algorithm 1: PEG (**P**rivacy for **E**dge-attributed **G**raphs)

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

Output: Perturbed edge-attributed 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 $\bar{d}^{x_k} \leftarrow$ `AggregateEdgePropertyDegrees`$(\tilde{d}_{v_i}^{x_k}, \ldots, \tilde{d}_{v_n}^{x_k})$;
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 $E' \leftarrow$ `AdjustEdgePropertyDegrees`$(E', \bar{d}^{x_1}, \ldots, \bar{d}^{x_t})$;
26 $E' \leftarrow$ `AdjustDisconnectedNodes`$(E', \bar{d}^{x_1}, \ldots, \bar{d}^{x_t})$;
27 **return** $G' = (V, E', X)$;

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-attributed graph $G' = (V, E', X)$ is released in line 27.

4.7 Computational Cost

The computational cost of the Partitioning & Clustering phase (Algorithm 1 – lines 3-10) consists of the time to partition the users V in the original graph G into a set of partitions $\mathcal{P}$, report and estimate the users' degrees, and build the set of clusters C according to their sorted node degrees, which implies in an expected time $O(|V| \cdot log|V|)$, which is the worst time spent in the sorting procedure. 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 attributes. 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 $O(|V| \cdot |C_*| \cdot |X|)$.

4.8 Privacy Analysis

The threat model for PEG considers that the adversary may possess background information about the edges and their attributes and may use this information to infer private details from the released graph G'. PEG aims to ensure that the adversary cannot determine with high confidence whether a specific edge (with its attribute) 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 4) and consequently do not leak information.

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

The remaining steps of PEG require privacy protection since they require that users send their data privately through LDP mechanisms. The clustering step uses ε_1 to locally report the user's edge property degrees through the geometric mechanism. Although each user is supposed to send multiple reports, one for each edge property, these reports are performed in parallel since the degree of one edge property is independent of the degree of the remaining edge properties. Then, by the parallel composition of DP, the clustering step satisfies ε_1-edge-LDP. In the partition-cluster mapping, the users of each partition report to which cluster they

belong through the OUE protocol. Reporting data through the OUE protocol requires a privacy budget ε_2. Since each user sends this information just once, this step consumes only ε_2 and satisfies ε_2-edge-LDP. Finally, the RANL reporting step is performed similarly to the previous one. Each user sends its RANL through the RR protocol, which also requires a privacy budget ε_3. As the RANL of each user is reported just once, this step consumes only ε_3 and satisfies ε_3-edge-LDP. Finally, by the DP sequential composition (Theorem 5), we can state that PEG satisfies ε-edge-LDP, where $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3$.

4.9 Summary

In this chapter, we presented the PEG approach, an innovative method for releasing edge-attributed graphs under LDP guarantees. PEG was structured into four main phases: Partitioning & Clustering, Partition-Cluster Mapping, RANL Reporting, and Graph Post-Processing.

We began the chapter by introducing RANL, a novel data structure for effectively encoding edge-attribute graphs. Following this, we detailed the partitioning & clustering phase, where nodes were grouped into partitions and clusters. In the partition-cluster mapping phase, we assigned partitions to clusters in a differentially private manner. This was followed by the RANL reporting phase, where users privately reported their local graphs using the RANL data structure. The final phase, graph post-processing, ensured consistency and improved the utility of the released graph through several techniques. Finally, we also outlined the PEG algorithm, highlighting its computational cost and privacy guarantees.

5 EXPERIMENTAL EVALUATION

In this chapter, we empirically evaluate the effectiveness of PEG on four real-world edge-attributed graphs. The experiments were carried out in Linux 64-bit, Intel(R) Core(TM) i7-7820X CPU @ 3.60GHz CPU and 128GB RAM. We implemented our approach in Python with the graph-tool (PEIXOTO, 2014) and Gurobi (Gurobi Optimization, LLC, 2024) 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 set according to the dataset structure, such that $p = \lfloor \frac{n}{1{,}000} \rfloor$ and $c = \lfloor \sqrt[3]{n} \rfloor$, where n is the number of nodes in the dataset. It is worth mentioning that there does not exist an immediate solution for determining the optional values for p and c, respectively. For instance, we adopt some known heuristics to aid the choice of these parameters. We set $p = \lfloor \frac{n}{1{,}000} \rfloor$ since we need a reasonable amount of users to perform a good estimation of the reported values (ERLINGSSON *et al.*, 2014). Then, $1,000$ seems to be a great choice. In turn, we used a similar heuristic presented in (JORGENSEN *et al.*, 2016) to set $c = \lfloor \sqrt[3]{n} \rfloor$, which states that in most scenarios, the degrees of a user 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 that has one connection with users of every other cluster.

We varied the privacy budget ε in the experiments according to $\varepsilon \in \{0.1, 0.5, 1.0\}$, 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 (BUREAU, 2021) 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. The allocation of the privacy budget ε for the *clustering* (ε_1), *partition-cluster mapping* (ε_2), and *RANL reporting* (ε_3) phases, and the choice of the y-th percentile, are is explained in Section 5.3.

5.1 Datasets

We conducted experiments over four real-world undirected edge-attribute network datasets from different domains and characteristics. The *DBLP*[1] (PANDHRE *et al.*, 2016; LI *et al.*, 2023) and *Netscience*[2] are co-authorship datasets, while *Yeast Landscape*[2] and *Pierre Auger*[2] are genetic datasets. Table 2 summarizes their characteristics. ‘‘EPP’’ is the abbreviation for

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

edge property proportions, detailed in Section 5.4.2.

Table 2 – Characteristics of the edge-attributed graph datasets.

	DBLP	**Netscience**	**Yeast Landscape**	**Pierre Auger**
# Nodes	41,427	14,065	4,458	514
# Edges	124,214	59,026	8,450,408	7,153
# Edge Properties	4	13	4	16
Degree$_{avg}$	5.99	8.39	3,791.12	27.83
Degree$_{max}$	358	361	5,044	123
St. Deviation$_{EPP}$	0.13	0.07	0.24	0.18
Domain	Co-Authorship	Co-Authorship	Genetic	Genetic

- *DBLP*: This is a sparse network that is widely used in scientific research for analyzing co-authorship networks. It comprises bibliographic information on major computer science journals and proceedings. The nodes in this dataset represent authors, while the edges denote co-authorship relationships between them. Each edge is attributed to the research area of the published work. The research areas are restricted to Data Mining (DM), Databases (DB), Information Retrieval (IR), and Machine Learning (ML). This dataset is particularly valuable for studying collaboration patterns and the structure of academic communities.
- *Netscience*: This dataset consists of a comprehensive network that maps the co-authorship relationships between scientists working in the field of network theory. The network is undirected, unweighted, and mostly sparse, where nodes are individual scientists and edges represent co-authorship of scientific papers. This dataset is valuable for studying collaboration patterns, community detection, and the overall structure and dynamics of scientific collaboration networks.
- *Yeast Landscape*: This is a dense network that provides a comprehensive view of the genetic and protein interactions within the yeast species *Saccharomyces Cerevisiae*. The dataset is structured as a multiplex network, where each layer represents different types of interactions, such as physical association, direct interaction, and genetic interaction. This dataset is essential for studying the network biology of yeast and can be used for various analyses, including network structure, dynamics, and functional module identification.
- *Pierre Auger*: This network occupies a mid-term position between sparse and dense networks. The dataset represents a multiplex network derived from the Pierre Auger Observatory, which is used to study ultra-high-energy cosmic rays. The dataset captures various

types of interactions, such as collaboration networks between scientists and the complex network of scientific contributions linked to the observatory's research. This dataset is particularly valuable for analyzing the collaborative structure within large-scale scientific experiments and the dissemination of knowledge within the astrophysics community.

The purpose of selecting datasets from different domains and sparsities is to evaluate how our proposed approach, PEG, behaves under different scenarios.

5.2 Baselines

To the best of our knowledge, no prior work exists on the DP release of entire edge-attributed 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 (LIU *et al.*, 2020) 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.

5.2.1 RANL-random

In this approach, only the RANL reporting and post-processing phases are considered. The RANL-random approach builds a perturbed graph based only on the reported users' RANLs. In this approach, the whole privacy budget is used to report the RANL of each user, with a length $h = |V| \cdot |X|$. Also, the edges consistency agreement post-processing step is performed randomly, i.e., this approach chooses randomly from which RANL the edge information is true. For example, consider two users $v_i, v_j \in V$ and an edge property $x_k \in X$. The edge $e_{i,j,k}$ is considered to exist if $RANL_{v_i}[e_{i,j,k}] = 1$ or $RANL_{v_j}[e_{j,i,k}] = 1$. Then, in the RANL-random approach, we consider that the true connections information between v_i and v_j has come from any of their RANLs.

5.2.2 RANL-consensus

RANL-consensus and RANL-random are similar approaches. They differ only in the edges consistency agreement post-processing step. Differently from the RANL-random, which chooses randomly from which RANL the edge information is true, the RANL-consensus uses the same idea as PEG. For example, consider two users $v_i, v_j \in V$ and an edge property $x_k \in X$.

The edge $e_{i,j,k}$ is only considered to exist if $RANL_{v_i}[e_{i,j,k}] = RANL_{v_j}[e_{j,i,k}] = 1$. Then, in the RANL-consensus approach, we consider that the true connections information between v_i and v_j is only considered to exist if they are present in both RANLs simultaneously.

5.2.3 PEG-random

The PEG-random approach is quite similar to PEG, but the difference is that it does not consider any degree information. Thus, the clustering is made randomly, the same way as in the partitioning. Also, only the cluster with the highest count (top-1) is chosen to be the cluster of the partition, disregarding the use of the weighting function. Finally, since there are only two private phases in PEG-random, the privacy budget allocation is split equally, 50% of the privacy budget ε for both the partition-cluster mapping and RANL reporting phases. Note that there are no degree adjustments in any of the baselines.

5.3 Privacy Budget Allocation

The use of the total privacy budget ε needs to be carefully allocated in each phase of PEG. Recall that our proposed approach divides the entire budget into three parts: ε_1 to the Partitioning & Clustering phase, ε_2 to the Partition-Cluster Mapping phase, and ε_3 to the RANL Reporting phase, such that $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 = \varepsilon$. However, the PEG approach is not only dependent on the privacy budget allocation. The y-th percentile, which is required to determine the clusters of the partitions in the Partition-Cluster Mapping phase, also impacts the quality of the released graph. For this matter, the y-th percentile needs to be carefully established.

In order to determine which is the best privacy budget allocation and the y-th percentile, this experiment empirically measures how the original graph G and the perturbed one G' are similar for different privacy budget and percentile value combinations, according to the Jaccard Similarity (JS), which is presented in detail in Chapter 5. This analysis is shown in Figures 23, 24, 25, and 26. In this experiment, we set $\varepsilon = 1$.

For the DBLP, Netscience, and Pierre Auger datasets, the similarity values are higher when $\varepsilon_1 \approx 0.5, \varepsilon_2 \approx 0.1$, $\varepsilon_3 \approx 0.4$, and $y \approx 90$. It makes sense to use this combination of parameters for datasets of these characteristics, which are not very dense. The highest amounts of budget are allocated to ε_1 and ε_3, which are responsible for the most sensitive phases. The node degrees used to post-process the perturbed graph are queried with ε_1, while the RANL

is reported with ε_3. Note that the length of the reported RANL impacts the data quality and depends on the cluster selection, which depends on the y-th percentile. As $y \approx 90$, the length of the reported RANL tends to be shortened, improving the data utility in non-dense graphs. In turn, the ε_2 is used for identifying which clusters have more connections with the nodes within a partition, and since this acts as a voting composed by many users, it does not require a high budget.

In counterpart, for the Yeast Landscape dataset, the similarity values are higher when $\varepsilon_1 \approx 0.1, \varepsilon_2 \approx 0.1$, $\varepsilon_8 \approx 0.8$, and $y \approx 10$. It is also expected and makes sense to use this combination of parameters for very dense datasets. Since the Yeast Landscape is a very dense dataset, assigning a low budget to ε_1 to query the node degrees will not harm the degree information since the node degrees tend to be much higher than the noise magnitude. Therefore, as each user has many edges, it is desirable to allocate more budget to ε_3 to maintain as much information as possible across the user's RANL. Additionally, as $y \approx 10$, it means that almost all possible clusters are relevant to represent the partitions. In other words, since the graph is very connected, every cluster is considered relevant for having many connections with the users within a partition.

5.4 Utility Analysis

In this section, we conduct various analyses to evaluate the effectiveness of the graphs released by PEG in terms of utility.

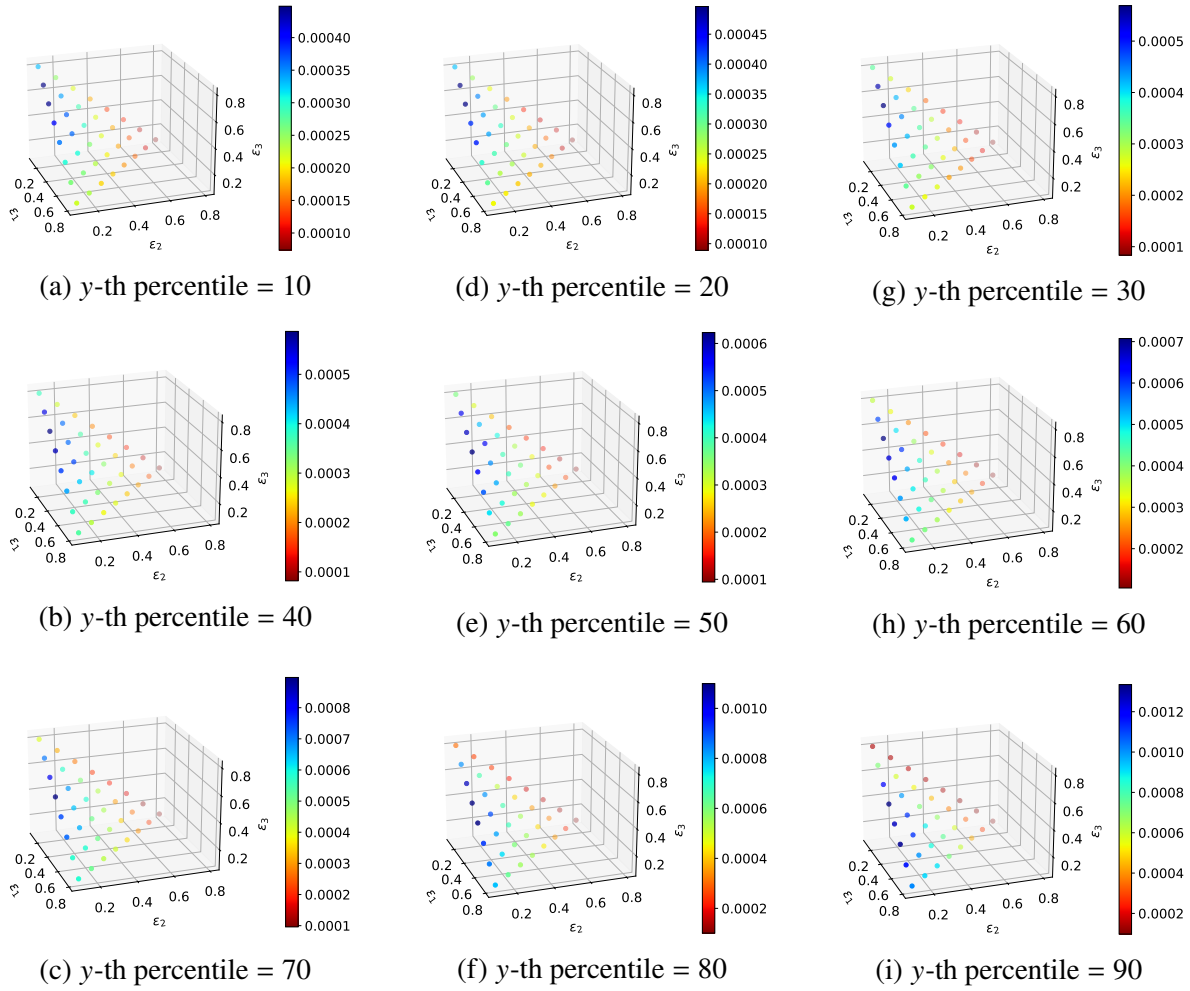
5.4.1 Degree Distribution

To evaluate how well G' (the released graph) captures the degree distribution of the original edge-attributed graph G, we applied the Kolmogorov-Smirnov (KS) statistic, which quantifies the maximum distance between two-degree distributions. Then, 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 $\mathrm{KS}(SD, \tilde{SD})$ can be calculated according to Equation 5.1. The lower the KS statistic, the higher the data utility.

$$\mathrm{KS}(SD, \tilde{SD}) = max_d |Cum_{SD}(d) - Cum_{\tilde{SD}}(d)| \tag{5.1}$$

Figure 27 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

Figure 23 – Jaccard Similarity (JS) between the original and the perturbed graphs, G and G', with different privacy budget allocations and percentile values for the DBLP dataset.



(a) y-th percentile = 10 (d) y-th percentile = 20 (g) y-th percentile = 30

(b) y-th percentile = 40 (e) y-th percentile = 50 (h) y-th percentile = 60

(c) y-th percentile = 70 (f) y-th percentile = 80 (i) y-th percentile = 90

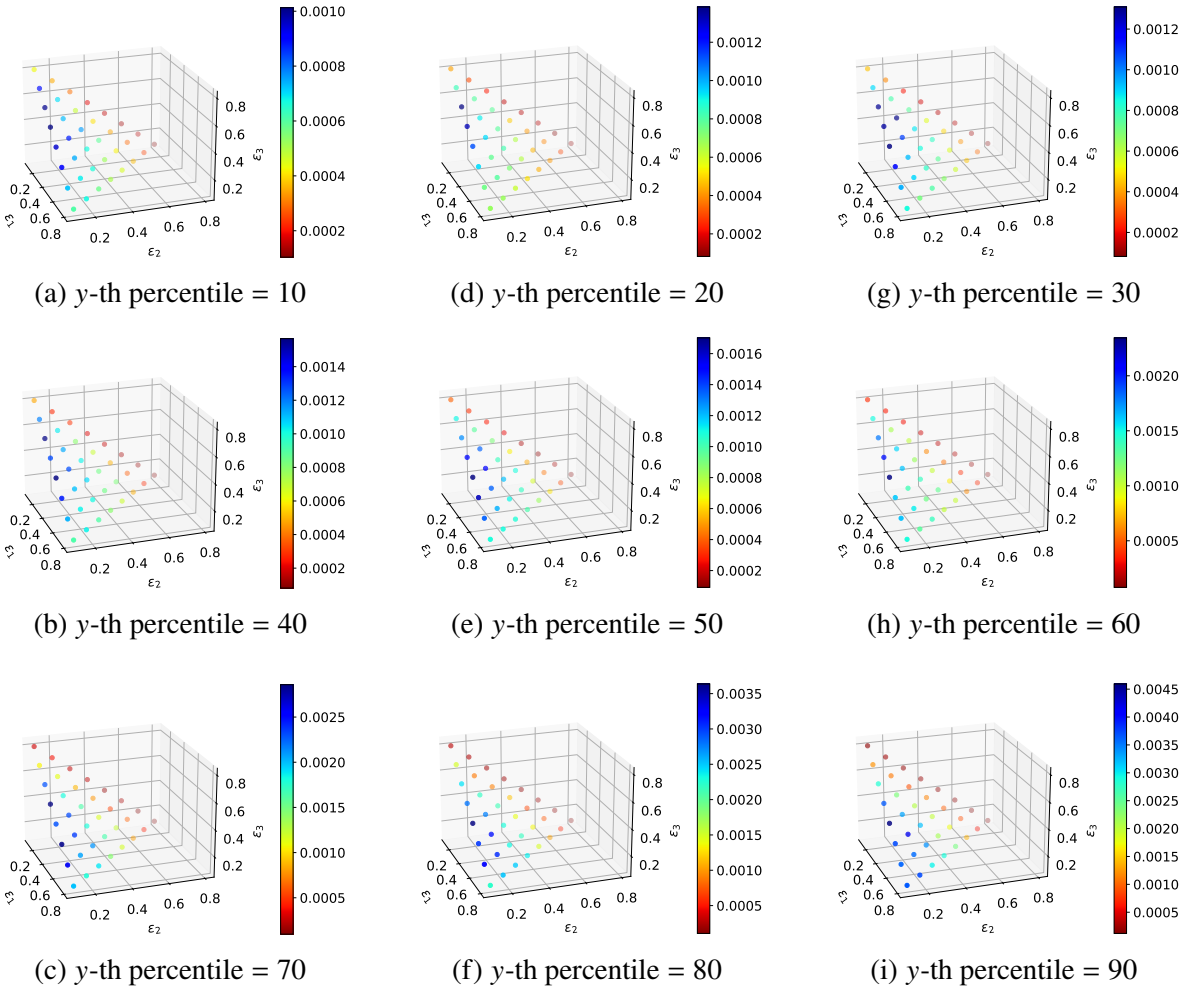
Source: Elaborated by the author.

the only approach with an additional degree correction step. The exception occurs in the DBLP, Netscience, and Pierre Auger datasets for $\varepsilon = 0.1$, where PEG-random slightly surpasses PEG. This occurs because these datasets are more sparse than dense graphs. Thus, many edge property degrees that are originally zero are estimated to different values after being perturbed, harming the graph's degree distribution. Also, many non-zero edge property degrees are too small, which can make them become zero after being perturbed. Although the Pierre Auger dataset also has many edge properties (16 in total), this graph is much denser than DBLP and Netscience, which makes this graph less sensitive to the estimation of the noisy degrees.

5.4.2 Edge Property Proportions

To evaluate how well the edge property proportions (EPP) of G are being maintained in G', we measured the Mean Absolute Error (MAE) of the edge property proportions between G

Figure 24 – Jaccard Similarity (JS) between the original and the perturbed graphs, G and G', with different privacy budget allocations and percentile values for the Netscience dataset.



(a) y-th percentile = 10 (d) y-th percentile = 20 (g) y-th percentile = 30

(b) y-th percentile = 40 (e) y-th percentile = 50 (h) y-th percentile = 60

(c) y-th percentile = 70 (f) y-th percentile = 80 (i) y-th percentile = 90

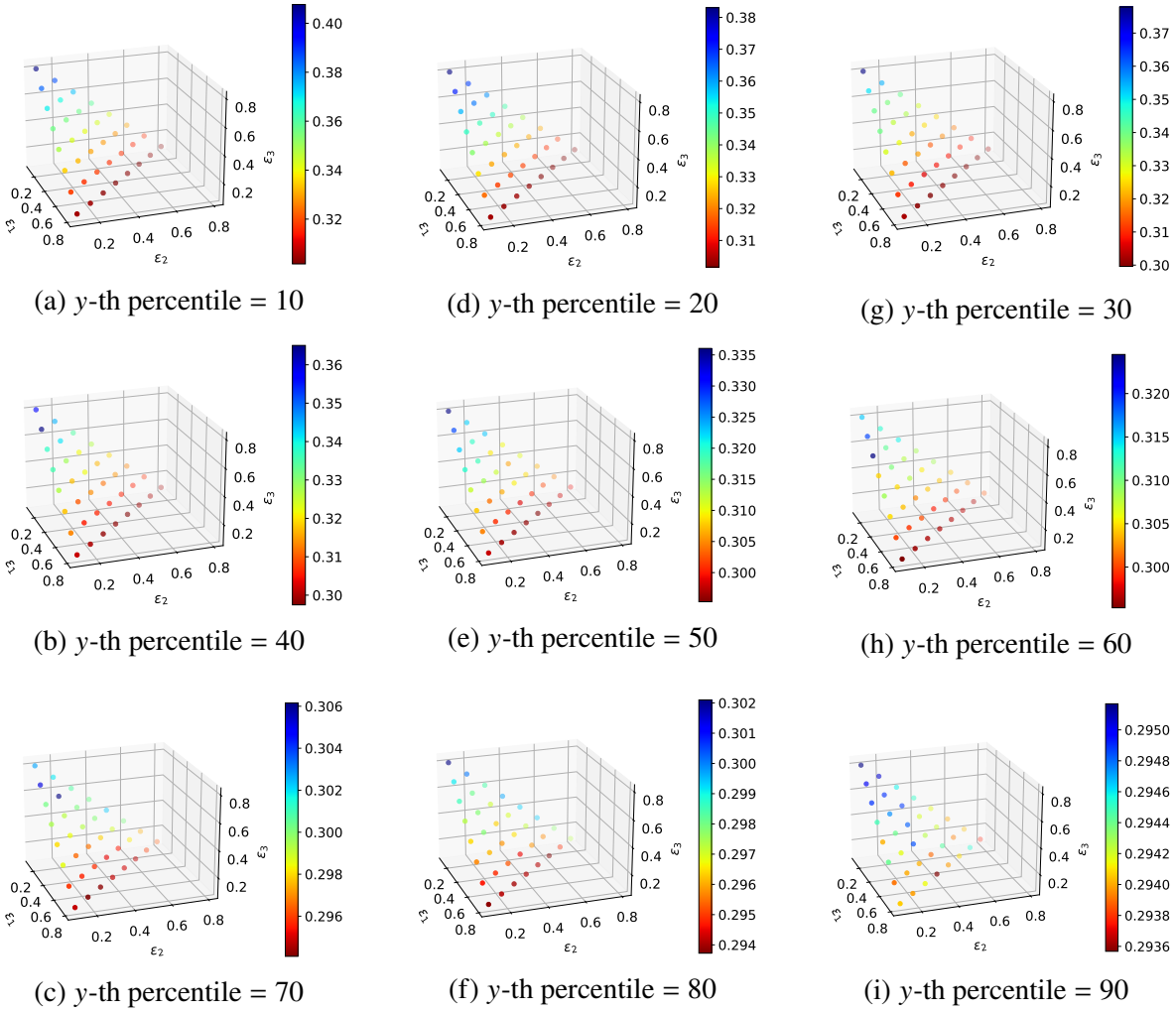
Source: Elaborated by the author.

and G'. Let $prop^G_{v_i} = (prop^{x_1}_{v_i}, \ldots, prop^{x_k}_{v_i})$ be the edge property proportions of the node $v_i \in V$, such that $prop^{x_k}_{v_i}$ denotes the proportion of adjacent edges of v_i associated with the property $x_k \in X$. 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 $\text{EPP}_{MAE}(G, G')$ according to Equation 5.2. The lower the EPP_{MAE}, the higher the data utility.

$$\text{EPP}_{MAE}(G, G') = \frac{\sum_{v_i \in V} \frac{||prop^G_{v_i} - prop^{G'}_{v_i}||_1}{|X|}}{|V|} \tag{5.2}$$

Figure 28 shows the results for the edge property proportions. We observe that PEG outperforms all baselines. Similarly to the degree distribution analysis, the exception occurs in the Netscience dataset for similar reasons. According to Table 2, the standard deviation of the edge property proportions in the Netscience dataset is considerably low. It means that this

Figure 25 – Jaccard Similarity (JS) between the original and the perturbed graphs, G and G', with different privacy budget allocations and percentile values for the Yeast Landscape dataset.



(a) y-th percentile = 10 (d) y-th percentile = 20 (g) y-th percentile = 30

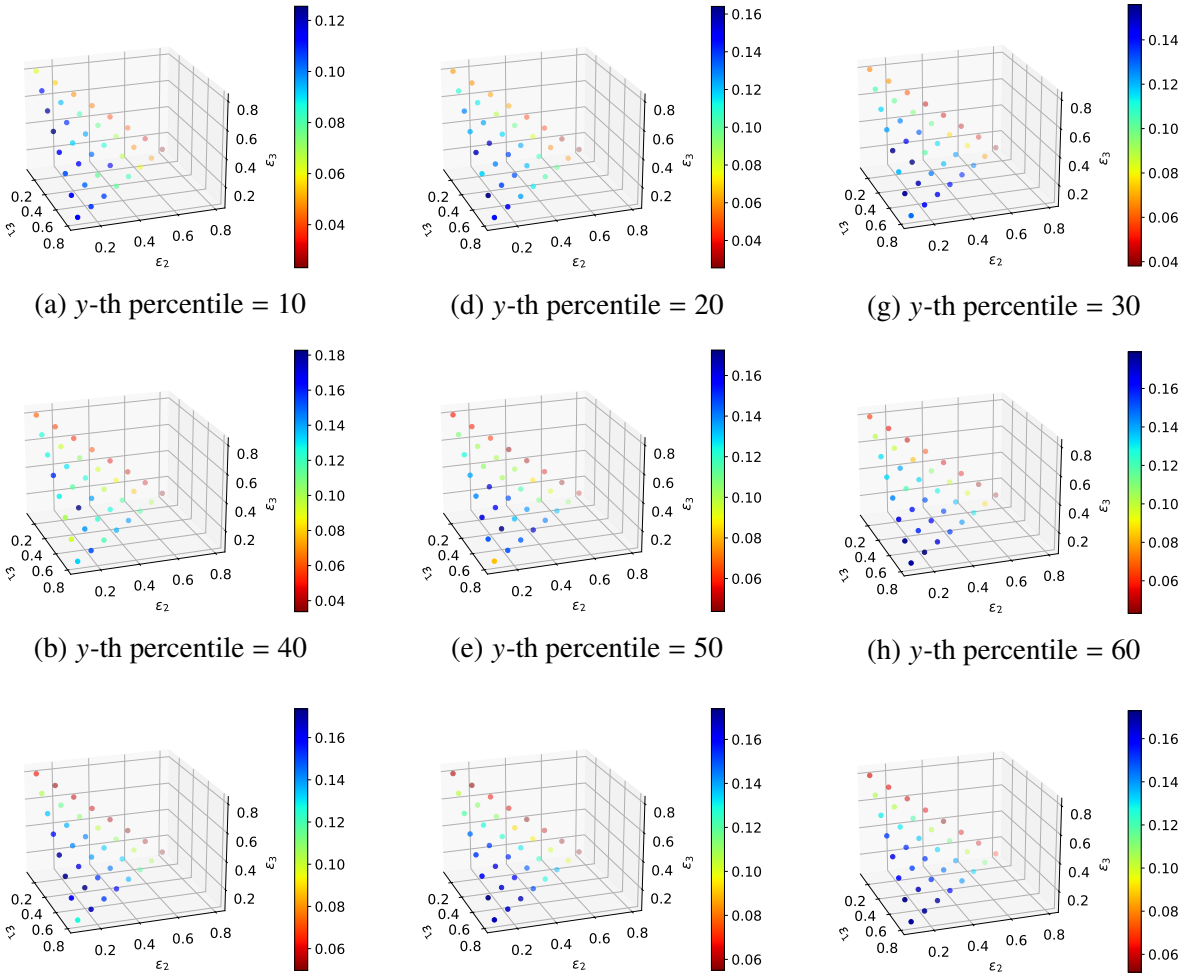
(b) y-th percentile = 40 (e) y-th percentile = 50 (h) y-th percentile = 60

(c) y-th percentile = 70 (f) y-th percentile = 80 (i) y-th percentile = 90

Source: Elaborated by the author.

dataset does not have a dominant edge property, i.e., the edge property proportions within the dataset are too close. Additionally, the graph's sparsity makes preserving the true edge property degrees significantly more challenging. For the remaining datasets, we have much more available information to improve the EPP. Although the DBLP dataset is also very sparse, it only has 4 edge properties, with a dominant edge property and a considerable number of nodes, which makes the estimation more accurate. The Yeast Landscape and Pierre Auger datasets benefit from being denser graphs, having a clear dominant edge property and an average degree expressively higher than the number of edge properties.

Figure 26 – Jaccard Similarity (JS) between the original and the perturbed graphs, G and G', with different privacy budget allocations and percentile values for the Pierre Auger dataset.



(a) y-th percentile = 10 (d) y-th percentile = 20 (g) y-th percentile = 30

(b) y-th percentile = 40 (e) y-th percentile = 50 (h) y-th percentile = 60

(c) y-th percentile = 70 (f) y-th percentile = 80 (i) y-th percentile = 90

Source: Elaborated by the author.

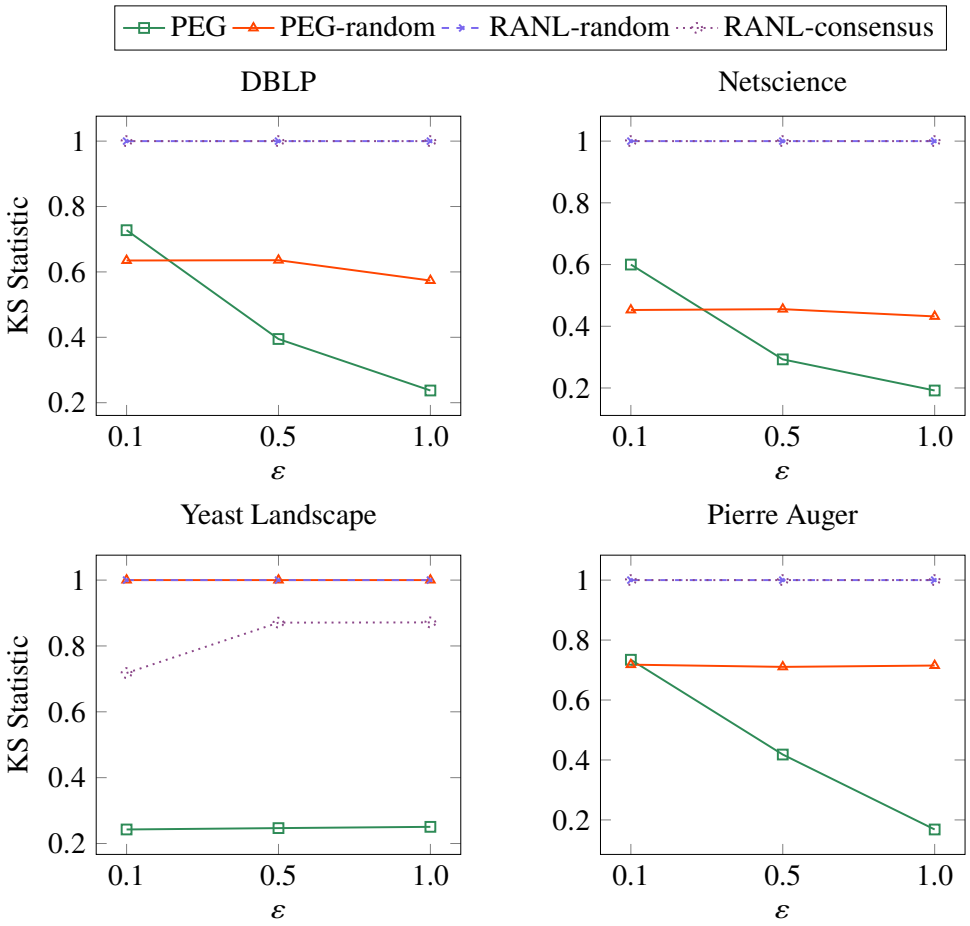
5.4.3 Number of Edges

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'. Then, we can define $\text{NE}_{MRE}(G, G')$ according to Equation 5.3, 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.

$$\text{NE}_{MRE}(G, G') = \frac{||E(G)| - |E(G')||}{|E(G)|} \tag{5.3}$$

Figure 29 shows the results for the number of edges. We can observe that PEG outperforms all baselines for almost all scenarios. The exception occurs only in the Pierre Auger

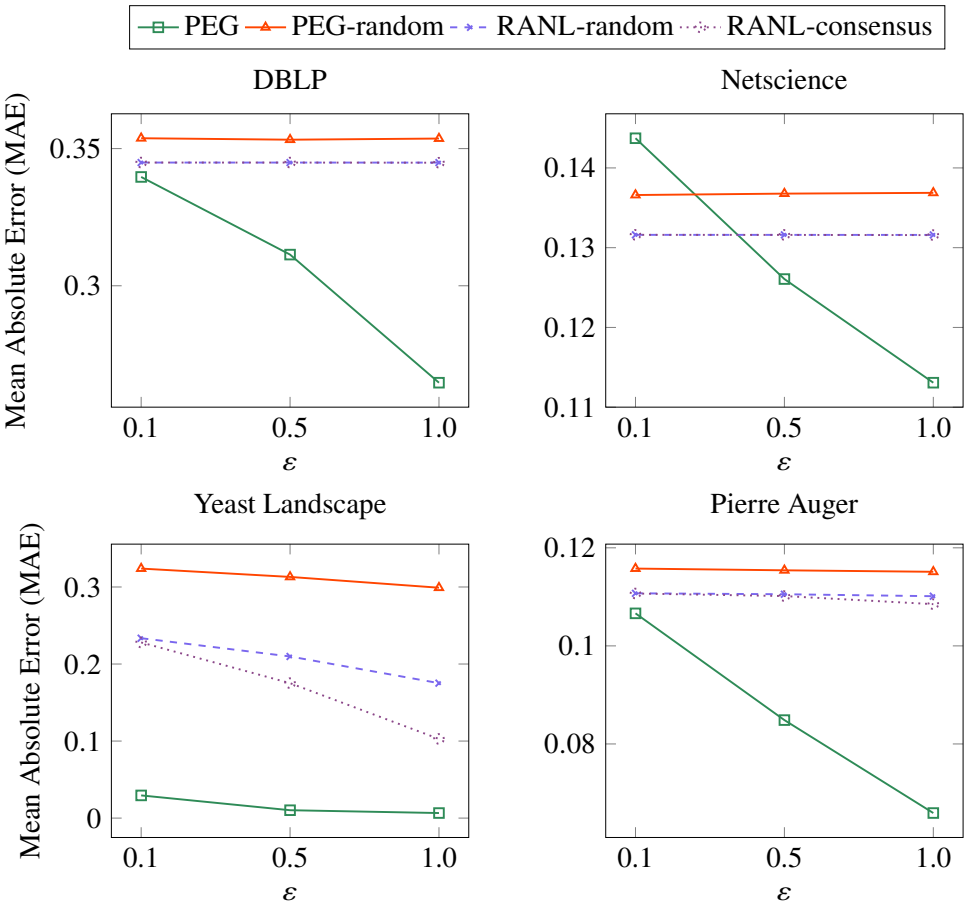
Figure 27 – Comparison of PEG and the baselines for the Degree Distribution analysis. The plots show the average Kolmogorov-Smirnov (KS) statistic after 10 runs, where $\varepsilon \in \{0.1, 0.5, 1.0\}$. The x-axis is the ε, while the y-axis is the KS statistic. Notably, the performance of PEG achieves better results in almost all scenarios, except for sparse networks when ε is too low.



Source: Elaborated by the author.

dataset for $\varepsilon = 1.0$. Note that the noisy node degrees are estimated based on the noisy edge property degrees of each user. Since the Pierre Auger dataset has a high number of edge properties with a dominant one, many users' edge property degrees are expected to be strongly harmed by the noise introduced. Small edge property degrees may suffer from the noise magnitude and can flip to zero or a considerably high value. Then, in these cases, the edge property degrees post-processing step may perform inaccurately. In counterpart, although the PEG-random does not have an edge property degree adjustment step, it outperforms PEG in these datasets with many edge properties due to the benefits of the clustering step. Even though the clustering is performed randomly, the length of the RANL is expressively reduced compared with the RANL-random and RANL-consensus baselines. Then, the perturbed graph tends to have much fewer noisy edges added, which compensates for the reduced length to build a perturbed graph with a number of

Figure 28 – Comparison of PEG and the baselines for the Edge Property Proportions (EPP) analysis. The plots show the Mean Absolute Error (MAE) 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. Notably, the performance of PEG achieves better results in almost all scenarios, except for sparse networks with many edge properties when ε is too low.



Source: Elaborated by the author.

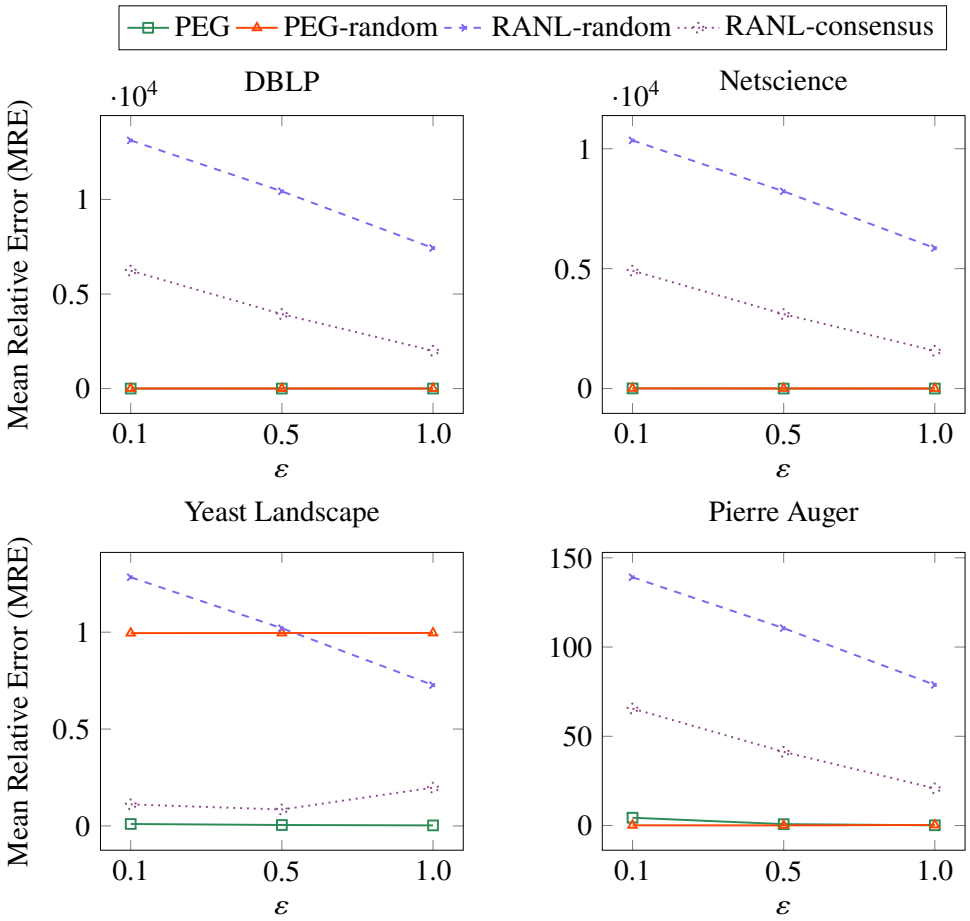
edges more similar to the original one. Finally, RANL-random and RANL-consensus baselines suffer from the length of the $RANL$. In these approaches, the length of the RANL is equal to $|V| \cdot |X|$, which leads to extremely noisy RANLs.

5.4.4 Graph Similarity

This analysis consists of a general metric that compares two graphs with different edges and properties. We applied the Jaccard Similarity (JS), which quantifies how similar two graphs are in terms of their original connections. We define $\mathrm{JS}(G, G')$ according to Equation 5.4, where $E(G)$ and $E(G')$ denote the set of edges in G and G', respectively. The higher the JS, the higher the similarity and, consequently, the data utility.

$$\mathrm{JS}(G, G') = \frac{|E(G) \cap E(G')|}{|E(G) \cup E(G')|} \tag{5.4}$$

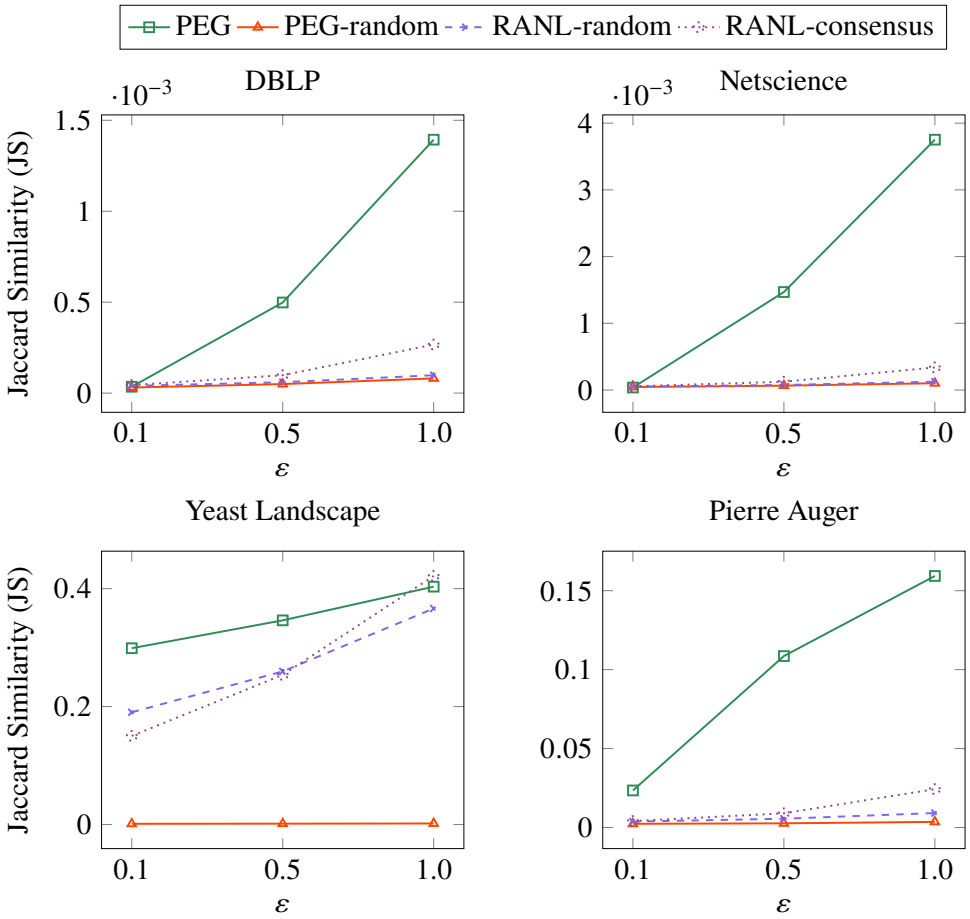
Figure 29 – Comparison of PEG and the baselines for the Number of Edges analysis. The plots show the Mean Relative Error (MRE) after 10 runs, where $\varepsilon \in \{0.1, 0.5, 1.0\}$. The x-axis is the ε, while the y-axis is the MRE. Notably, the performance of PEG achieves better results in almost all scenarios, except for networks with a dominant edge property among a few edge properties when ε is too low.



Source: Elaborated by the author.

Figure 30 shows the results for the graph similarity. Note that PEG outperforms all baselines in almost all datasets. The exception occurs in the Yeast Landscape graph, where PEG starts to lose for the RANL-random and RANL-consensus after reaching an $\varepsilon = 1.0$. This behavior can be attributed to the specific characteristics of the Yeast Landscape dataset, which is very dense and has a high average degree. When applying PEG, the clustering phase takes only the clusters with weighted counts above the y-th percentile to be the clusters of the partition. This clustering criterion may lose many existing edges between the partition's nodes and nodes from other clusters that have not been chosen. For this purpose, the RANL-random and RANL-consensus baselines perform better when ε starts to grow. Since the dataset is very dense and these approaches only use perturbed RANLs to build the private graph, a large portion of the RANLs already consists of true edges, leaving little room for adding false edges. For less dense

Figure 30 – Comparison of PEG and the baselines for the Graph Similarity analysis. The plots show the Jaccard Similarity (JS) 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. Notably, the performance of PEG achieves better results in almost all scenarios, except for dense networks, when ε becomes higher.



Source: Elaborated by the author.

datasets, PEG outperforms by shortening the length of the RANL during the clustering phase. By reducing the length, fewer false edges are introduced into the perturbed RANL, allowing PEG to preserve more original edges compared to the baselines.

5.4.5 Community Similarity

In graph analytics, communities are extremely relevant since they help us understand network complexities. Then, 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 $CM_{v_i}^{G}$ and $CM_{v_i}^{G'}$ denote the label of the community assigned to the user v_i in G and G', respectively, there are no guarantees that v_i remained in the same community, even though the labels $CM_{v_i}^{G} = CM_{v_i}^{G'}$. It may happen

since the communities in G and G' may be labeled differently.

However, community detection algorithms were not formerly designed for edge-attributed multigraphs, i.e., more than one edge connecting the same two nodes but with different attributes. 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 property proportions of G, such that $prop_{x_k}$ denotes the proportion of edges associated with the property $x_k \in X$ 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. Therefore, 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 (ABBE, 2018) to find graph communities. This process of weighting and discovering the communities is repeated for the graph 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.

$$
\begin{aligned}
\textbf{maximize} \quad & Z = \sum_{i=1}^{z} \sum_{j=1}^{\tilde{z}} score_{i,j} \cdot x_{i,j} \\
\textbf{s.t.} \quad & \sum_{i=1}^{z} x_{i,j} = 1 \quad \forall_{j \leq \tilde{z}} \\
& \sum_{j=1}^{\tilde{z}} x_{i,j} = 1 \quad \forall_{i \leq z} \\
& x_{i,j} \geq 0 \quad \forall_{i \leq z}; \forall_{j \leq \tilde{z}}
\end{aligned}
\tag{5.5}
$$

Due to the constraints of our problem, we modeled it as a variant of the assignment problem (KUHN, 1955), where the goal is to find the best worker for each task, such that each worker can only execute one task, and each task is executed by one worker, while the objective function is minimized or maximized. In our problem, we can consider that the communities of G are the workers, while the communities of G' are the tasks. Our goal is to determine which pair of communities of G and G' maximize the objective function Z in Equation 5.5. 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 $score_{i,j} = |CM_{s_i}^{G} \cap CM_{s_j}^{G'}|$.

Figure 31 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 when $\varepsilon = 0.1$ since this dataset is very sparse and also has an average degree smaller than the number of edge properties. This characteristic leads PEG to query the edge property degrees inaccurately, which can harm the edge property degree adjustment step and, consequently, form unexpected communities. Differently, PEG is drastically affected in the Yeast Landscape. However, it is already expected since this dataset is very dense. Although the edge property degrees are queried accurately, the clustering phase may cause the loss of many original connections, which are further randomly rewired according to the degrees, leading to inaccurate communities. For the remaining datasets, whose average degree is higher than the number of edge properties and which are not too dense, PEG outperforms due to the clustering phase's improvements by shortening the length of the RANL. By reducing the length, the number of false edges introduced into the perturbed RANL is significantly smaller. Therefore, PEG can maintain many more original edges when compared to the baselines.

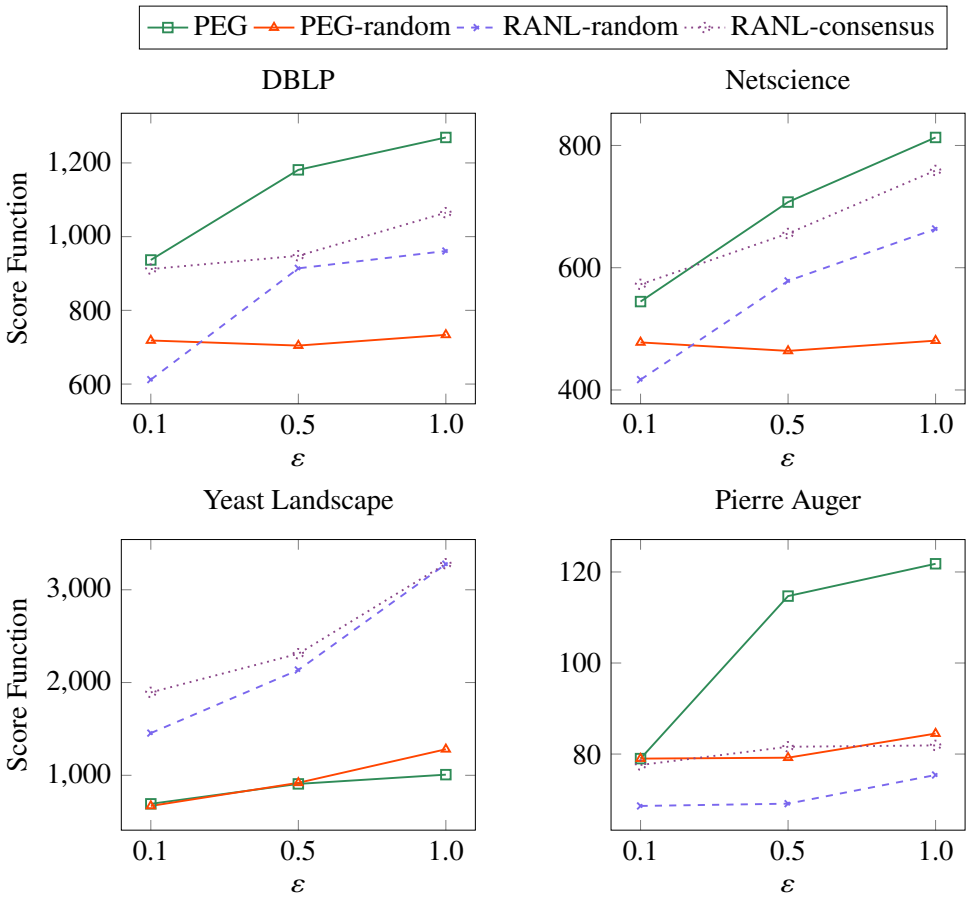
5.5 Summary

In this chapter, we conducted an extensive experimental evaluation of our proposed approach, PEG, using four real-world edge-attributed graph datasets, DBLP, Netscience, Yeast Landscape, and Pierre Auger, of different domains. Our experiments aimed to assess the utility and effectiveness of PEG in maintaining the structural properties of the original graphs while providing strong privacy guarantees under LDP.

Throughout the experiments, we analyzed several aspects, including the degree distribution, edge property proportions, the number of edges, and graph and community similarities. Our findings demonstrated that PEG generally outperforms baseline methods in most scenarios, mainly in non-extreme datasets, i.e., too sparse or too dense, where PEG surpasses the baselines in all scenarios.

However, in highly dense datasets, PEG's performance was slightly compromised due to the inherent complexities and the high average degrees, which affected the accuracy of the discovered communities. Despite this, PEG is still competitive when compared to the baselines

Figure 31 – Comparison of PEG and the baselines for the Community Similarity analysis. The plots show the score according to an optimization function, i.e., the number of nodes that remained in the same communities in the original graphs and in their perturbed versions 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. Notably, the performance of PEG achieves better results in almost all scenarios except for dense networks.



Source: Elaborated by the author.

and outperforms them in the major scenarios.

Overall, the experimental results confirmed that PEG is a robust approach for privately releasing edge-attributed graphs, achieving a good balance between privacy and data utility across various graph analyses. Our evaluation highlights the potential of PEG in real-world applications, providing directions for future enhancements and broader applications.

6 CONCLUSION

In this chapter, we present a summary of the results of this thesis, provide future work directions, and discuss how the performed study can impact other research fields.

6.1 Summary of Results

In this work, we addressed the problem of releasing edge-attributed graphs under LDP guarantees. We have done an extensive review of the literature about releasing graph data under differential privacy, including applying it to social network analysis, which gave rise to the works listed in Section 1.3, including a submitted paper that is actually under revision.

Then, we revisit the thesis hypothesis:

Hypothesis: *''Given an edge-attribute graph, there is an LDP approach that is able to publish a perturbed version of this graph without adding an amount of noise that makes the published data useless. Moreover, there are protocols associated with this DP setting that have low variance, i.e., achieve better data utility levels.''*

To prove the hypothesis, we developed PEG, a novel decentralized dynamic degree-based clustering approach designed for privately releasing edge-attribute graphs under the notion of edge-LDP, which improves the data utility by reducing the dimensionality of the reported data. The novel favors our approach to the proposed Randomized Attribute Neighbor List (RANL) data structure, which complies with edge-LDP, to report the users' local edge-attributed graphs. Additionally, PEG combines optimized LDP protocols which provide better data utility. 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 achieves high utility for various graph analysis metrics on the released graph, including applications in community detection. It outperforms the baselines in almost all presented scenarios, except for some analyses performed within extremely dense datasets.

6.2 Future Work

Immediately, the evaluation of PEG's performance will be done through other LDP protocols. For instance, the Generalized Randomized Response (GRR) and other optimized protocols, like Optimized Local Hashing (OLH) (WANG *et al.*, 2017), could be tested to compare their

performance with the current approach. These evaluations would provide a broader understanding of how different LDP protocols impact the quality and utility of the released graphs.

Additionally, exploring other DP applications for edge-attributed graphs is a valuable next step. The proposed solution has potential applications in fields such as anomaly detection and mobility analysis. For anomaly detection, future research could focus on developing specific algorithms that leverage the private edge-attributed graphs to identify unusual patterns and deviations more accurately. In mobility analysis, researchers could investigate how private edge-attributed graphs can be used to optimize traffic management and urban planning without compromising individual privacy.

Another significant direction for future research is extending the notion of node-LDP to edge-attributed graphs. This task presents an increased challenge since it is a stronger notion of privacy in the graph context. This extension would involve developing new techniques to handle the increased complexity and ensure that the released graphs remain useful for analyses.

Moreover, the scalability of PEG to handle larger datasets is another area for investigation. As data grows in volume and complexity, ensuring that PEG can efficiently process and release large-scale edge-attributed graphs without compromising privacy or utility will be essential. Research could focus on optimizing the algorithm for parallel processing and distributed computing environments.

Lastly, conducting extensive experimental evaluations on diverse real-world datasets beyond those used in the current work would provide deeper insights into the robustness and applicability of PEG.

6.3 Broader Impact

The broader impact of the research presented in this thesis extends beyond data privacy, offering significant contributions across various fields. The innovative approach to release edge-attributed graphs while preserving data utility opens up new possibilities in multiple domains.

In anomaly detection, the solution enhances the accuracy and reliability of systems that identify unusual patterns and deviations within network data, which is crucial for cybersecurity and network management. In mobility analysis, our approach helps urban planners and transportation engineers optimize routes, improve traffic management, and enhance public transportation systems, leading to more efficient urban mobility solutions.

The methodology also benefits social network analysis, marketing, and organizational behavior studies by facilitating the identification and exploration of connected groups or communities within larger networks. This deeper understanding of graph structures and interactions informs marketing, community building, and organizational development strategies.

In genetic and biological research, the approach improves the representation of interactions between genes or proteins, aiding in the discovery of new biological insights and the development of medical treatments. Additionally, analyzing network communication patterns, such as email or social interactions, becomes more accurate and insightful, informing strategies for effective information dissemination and organizational efficiency.

By preserving data utility while ensuring privacy, this solution enhances the applicability and reliability of edge-attributed graph analysis across these diverse domains, advancing both privacy-preserving data analysis and the understanding of complex networks in various scientific and practical applications.

REFERENCES

ABBE, E. Community detection and stochastic block models: Recent developments. **Journal of Machine Learning Research**, v. 18, p. 1–86, 2018.

ACHARYA, J.; SUN, Z.; ZHANG, H. Hadamard response: Estimating distributions privately, efficiently, and with little communication. In: **The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)**. Naha, Okinawa, Japan: PMLR, 2019. v. 89, p. 1120–1129.

ALLARDYCE, C. S. **Fat chemistry: The science behind obesity**. [S.l.]: Royal Society of Chemistry, 2012.

ALSMADI, I.; ALHAMI, I. Clustering and classification of email contents. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, v. 27, n. 1, p. 46–57, 2015.

BASSILY, R.; SMITH, A. D. Local, private, efficient protocols for succinct histograms. In: **Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)**. Portland, OR, USA: ACM, 2015. p. 127–135.

BENDIMERAD, A. A. **Mining Useful Patterns in Attributed Graphs. (Fouille de motifs utiles dans les graphes attribués)**. Phd Thesis (PhD Thesis) — University of Lyon, France, 2019.

BERGE, C. **The theory of graphs**. [S.l.]: Courier Corporation, 2001.

BRITO, F. T. **Differentially private release of count-weighted graphs**. Phd Thesis (PhD Thesis) — Universidade Federal do Ceará, Brazil, 2023.

BRITO, F. T.; FARIAS, V. A. E. de; FLYNN, C. J.; MAJUMDAR, S.; MACHADO, J. C.; SRIVASTAVA, D. Global and local differentially private release of count-weighted graphs. **Proceedings of the ACM on Management of Data**, ACM, v. 1, n. 2, p. 1–25, 2023.

BRITO, F. T.; MACHADO, J. C. Preservação de privacidade de dados: Fundamentos, técnicas e aplicações. **Jornadas de Atualização em Informática**, p. 91–130, 2017.

BRITO, F. T.; MENDONÇA, A. L. C.; MACHADO, J. C. A differentially private guide for graph analytics. In: **Proceedings 27th International Conference on Extending Database Technology (EDBT)**. Paestum, Italy: OpenProceedings.org, 2024. p. 850–853.

BUREAU, U. S. C. **Census Bureau Sets Key Parameters to Protect Privacy in 2020 Census Results**. 2021. Available at: https://www.census.gov/newsroom/press-releases/2021/2020-census-key-parameters.html. Accessed on: 08 May 2024.

CHEN, L.; HAN, K.; XIU, Q.; GAO, D. Graph clustering under weight-differential privacy. In: **24th IEEE 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)**. Hainan, China: IEEE, 2022. p. 1457–1464.

CHEN, R.; FUNG, B. C. M.; YU, P. S.; DESAI, B. C. Correlated network data publication via differential privacy. **The VLDB Journal**, Springer, v. 23, n. 4, p. 653–676, 2014.

CHEN, X.; MAUW, S.; RAMÍREZ-CRUZ, Y. Publishing community-preserving attributed social graphs with a differential privacy guarantee. **Proceedings on Privacy Enhancing Technologies**, v. 2020, n. 4, p. 131–152, 2020.

CLAUSET, A.; MOORE, C.; NEWMAN, M. E. J. Structural inference of hierarchies in networks. In: **Statistical Network Analysis: Models, Issues, and New Directions - ICML Workshop on Statistical Network Analysis**. Pittsburgh, PA, USA: Springer, 2006. v. 4503, p. 1–13.

COOK, D. J.; HOLDER, L. B. **Mining graph data**. [S.l.]: John Wiley & Sons, 2006.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to algorithms**. [S.l.]: MIT Press, 2022.

CORMODE, G.; PROCOPIUC, C. M.; SRIVASTAVA, D.; TRAN, T. T. L. Differentially private summaries for sparse data. In: **15th International Conference on Database Theory (ICDT)**. Berlin, Germany: ACM, 2012. p. 299–311.

DUCHI, J. C.; JORDAN, M. I.; WAINWRIGHT, M. J. Local privacy and statistical minimax rates. In: **54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)**. Berkeley, CA, USA: IEEE, 2013. p. 429–438.

DWORK, C. Differential privacy. In: **33rd International Colloquium on Automata, Languages and Programming (ICALP)**. Venice, Italy: Springer, 2006. v. 4052, p. 1–12.

DWORK, C.; MCSHERRY, F.; NISSIM, K.; SMITH, A. D. Calibrating noise to sensitivity in private data analysis. In: **Theory of Cryptography: Third Theory of Cryptography Conference (TCC)**. New York, NY, USA: Springer, 2006. v. 3876, p. 265–284.

DWORK, C.; ROTH, A. The algorithmic foundations of differential privacy. **Foundations and Trends® in Theoretical Computer Science**, Now Publishers, Inc., v. 9, n. 3–4, p. 211–407, 2014.

ERLINGSSON, Ú.; PIHUR, V.; KOROLOVA, A. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In: **Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security**. Scottsdale, AZ, USA: ACM, 2014. p. 1054–1067.

FAN, C.; LI, P. Distances release with differential privacy in tree and grid graph. In: **IEEE International Symposium on Information Theory (ISIT)**. Espoo, Finland: IEEE, 2022. p. 2190–2195.

FILHO, J. S. da C.; MACHADO, J. C. FELIP: A local differentially private approach to frequency estimation on multidimensional datasets. In: **Proceedings 26th International Conference on Extending Database Technology (EDBT)**. Ioannina, Greece: OpenProceedings.org, 2023. p. 671–683.

GAO, T.; LI, F.; CHEN, Y.; ZOU, X. Local differential privately anonymizing online social networks under hrg-based model. **IEEE Transactions on Computational Social Systems**, IEEE, v. 5, n. 4, p. 1009–1020, 2018.

GHOSH, A.; ROUGHGARDEN, T.; SUNDARARAJAN, M. Universally utility-maximizing privacy mechanisms. In: **Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)**. Bethesda, MD, USA: ACM, 2009. p. 351–360.

GRYGORASH, O.; ZHOU, Y.; JORGENSEN, Z. Minimum spanning tree based clustering algorithms. In: **18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)**. Arlington, VA, USA: IEEE, 2006. p. 73–81.

Gurobi Optimization, LLC. **Gurobi Optimizer Reference Manual**. 2024. Available at: https://www.gurobi.com.

HAY, M.; LI, C.; MIKLAU, G.; JENSEN, D. D. Accurate estimation of the degree distribution of private networks. In: **The Ninth IEEE International Conference on Data Mining (ICDM)**. Miami, Florida, USA: IEEE, 2009. p. 169–178.

HSU, J.; GABOARDI, M.; HAEBERLEN, A.; KHANNA, S.; NARAYAN, A.; PIERCE, B. C.; ROTH, A. Differential privacy: An economic method for choosing epsilon. In: **IEEE 27th Computer Security Foundations Symposium (CSF)**. Vienna, Austria: IEEE, 2014. p. 398–410.

HU, A. L.; CHAN, K. C. Utilizing both topological and attribute information for protein complex identification in ppi networks. **IEEE/ACM Transactions on Computational Biology and Bioinformatics**, IEEE, v. 10, n. 3, p. 780–792, 2013.

HUANG, H.; ZHANG, D.; XIAO, F.; WANG, K.; GU, J.; WANG, R. Privacy-preserving approach PBCN in social network with differential privacy. **IEEE Transactions on Network and Service Management**, IEEE, v. 17, n. 2, p. 931–945, 2020.

IFTIKHAR, M.; WANG, Q.; LIN, Y. dk-microaggregation: Anonymizing graphs with differential privacy guarantees. In: **24th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)**. Singapore: Springer, 2020. v. 12085, p. 191–203.

III, J. J. P.; FOND, T. L.; MORENO, S.; NEVILLE, J. Fast generation of large scale social networks while incorporating transitive closures. In: **International Conference on Privacy, Security, Risk and Trust (PASSAT) and International Confernece on Social Computing (SocialCom)**. Amsterdam, Netherlands: IEEE, 2012. p. 154–165.

III, J. J. P.; MORENO, S.; FOND, T. L.; NEVILLE, J.; GALLAGHER, B. Attributed graph models: modeling network structure with correlated attributes. In: **23rd International World Wide Web Conference (WWW)**. Seoul, Republic of Korea: ACM, 2014. p. 831–842.

JORGENSEN, Z.; YU, T.; CORMODE, G. Publishing attributed social graphs with formal privacy guarantees. In: **Proceedings of the 2016 International Conference on Management of Data (SIGMOD)**. San Francisco, CA, USA: ACM, 2016. p. 107–122.

KARWA, V.; RASKHODNIKOVA, S.; SMITH, A. D.; YAROSLAVTSEV, G. Private analysis of graph structure. **Proceedings of the VLDB Endowment**, v. 4, n. 11, p. 1146–1157, 2011.

KASIVISWANATHAN, S. P.; NISSIM, K.; RASKHODNIKOVA, S.; SMITH, A. D. Analyzing graphs with node differential privacy. In: **Theory of Cryptography - 10th Theory of Cryptography Conference (TCC)**. Tokyo, Japan: Springer, 2013. v. 7785, p. 457–476.

KAYTOUE, M.; PLANTEVIT, M.; ZIMMERMANN, A.; BENDIMERAD, A. A.; ROBARDET, C. Exceptional contextual subgraph mining. **Machine Learning**, Springer, v. 106, n. 8, p. 1171–1211, 2017.

KRAUSE, A.; GOLOVIN, D. Submodular function maximization. In: **Tractability: Practical Approaches to Hard Problems**. [S.l.]: Cambridge University Press, 2014. p. 71–104.

KUHN, H. W. The hungarian method for the assignment problem. **Naval research logistics quarterly**, Wiley Online Library, v. 2, n. 1–2, p. 83–97, 1955.

LI, L.; ZHAO, Y.; LUO, S.; WANG, G.; WANG, Z. Efficient community search in edge-attributed graphs. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 35, n. 10, p. 10790–10806, 2023.

LI, N.; LYU, M.; SU, D.; YANG, W. **Differential Privacy: From Theory to Practice**. [S.l.]: Morgan & Claypool Publishers, 2016.

LI, X.; YANG, J.; SUN, Z.; ZHANG, J. Differential privacy for edge weights in social networks. **Security and Communication Networks**, Wiley Online Library, v. 2017, n. 1, p. 1–10, 2017.

LIANG, F.; LIU, C.; CARROLL, R. **Advanced Markov chain Monte Carlo methods: learning from past samples**. [S.l.]: John Wiley & Sons, 2011.

LIU, Z.; HUANG, L.; XU, H.; YANG, W.; WANG, S. Privag: Analyzing attributed graph data with local differential privacy. In: **IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)**. Hong Kong: IEEE, 2020. p. 422–429.

MCSHERRY, F.; TALWAR, K. Mechanism design via differential privacy. In: **48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)**. Providence, RI, USA: IEEE, 2007. p. 94–103.

MENDONÇA, A. L.; BRITO, F. T.; MACHADO, J. C. Privacy-preserving techniques for social network analysis. In: **Anais Estendidos do XXXVIII Simpósio Brasileiro de Bancos de Dados**. Belo Horizonte, MG, Brazil: SBC, 2023. p. 174–178.

MENDONÇA, A. L.; BRITO, F. T.; MACHADO, J. C. Análise de dados privada em redes sociais. **Jornadas de Atualização em Informática**, 2024.

MENDONÇA, A. L. C.; BRITO, F. T.; MACHADO, J. C. **PEG: Local Differential Privacy for Edge-Labeled Graphs**. 2024. Submitted for publication to the International Conference on Extending Database Technology (EDBT, 2025).

MIRZASOLEIMAN, B.; BADANIDIYURU, A.; KARBASI, A.; VONDRÁK, J.; KRAUSE, A. Lazier than lazy greedy. In: **Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (CAI)**. Austin, Texas, USA: AAAI Press, 2015. v. 29, n. 1, p. 1812–1818.

MORVAN, A.; CHOROMANSKI, K.; GOUY-PAILLER, C.; ATIF, J. Graph sketching-based massive data clustering. **arXiv preprint arXiv:1703.02375**, 2017.

NEAR, J. P.; ABUAH, C. **Programming Differential Privacy**. [s.n.], 2021. Available at: https://programming-dp.com/.

NETO, E. R. D.; MACHADO, J. C.; MENDONÇA, A. L. Privlbs: Preserving privacy in location based services. **Journal of Information and Data Management**, v. 10, n. 2, p. 81–96, 2019.

NETO, E. R. D.; MENDONÇA, A. L. C.; BRITO, F. T.; MACHADO, J. C. Privlbs: uma abordagem para preservação de privacidade de dados em serviços baseados em localização. In: **XXXIII Simpósio Brasileiro de Banco de Dados (SBBD)**. Rio de Janeiro, RJ, Brazil: SBC, 2018. p. 109–120.

NGUYEN, H. H.; IMINE, A.; RUSINOWITCH, M. Differentially private publication of social graphs at linear cost. In: **Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)**. Paris, France: ACM, 2015. p. 596–599.

NISSIM, K.; RASKHODNIKOVA, S.; SMITH, A. D. Smooth sensitivity and sampling in private data analysis. In: **Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)**. San Diego, California, USA: ACM, 2007. p. 75–84.

PANDHRE, S.; GUPTA, M.; BALASUBRAMANIAN, V. N. Community-based outlier detection for edge-attributed graphs. **arXiv preprint arXiv:1612.09435**, 2016.

PEIXOTO, T. P. The graph-tool python library. **figshare**, 2014. Available at: http://-figshare.com/articles/graph_tool/1164194.

PEROZZI, B. **Local Modeling of Attributed Graphs: Algorithms and Applications**. Phd Thesis (PhD Thesis) — State University of New York at Stony Brook, USA, 2016.

PINOT, R.; MORVAN, A.; YGER, F.; GOUY-PAILLER, C.; ATIF, J. Graph-based clustering under differential privacy. In: **Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI) 2018**. Monterey, California, USA: AUAI Press, 2018. p. 329–338.

QIN, Z.; YU, T.; YANG, Y.; KHALIL, I.; XIAO, X.; REN, K. Generating synthetic decentralized social graphs with local differential privacy. In: **Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)**. Dallas, TX, USA: ACM, 2017. p. 425–438.

RIBEIRO, P.; PAREDES, P.; SILVA, M. E. P.; APARÍCIO, D.; SILVA, F. M. A. A survey on subgraph counting: Concepts, algorithms, and applications to network motifs and graphlets. **ACM Computing Surveys (CSUR)**, ACM, v. 54, n. 2, p. 1–36, 2022.

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 Internet Measurement Conference (IMC)**. Berlin, Germany: ACM, 2011. p. 81–98.

SAMOYLENKO, I.; ALEJA, D.; PRIMO, E.; ALFARO-BITTNER, K.; VASILYEVA, E.; KOVALENKO, K.; MUSATOV, D.; RAIGORODSKII, A. M.; CRIADO, R.; ROMANCE, M. *et al.* Why are there six degrees of separation in a social network? **Physical Review X**, APS, v. 13, n. 2, p. 021032, 2023.

SEALFON, A. Shortest paths and distances with differential privacy. In: **Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)**. San Francisco, CA, USA: ACM, 2016. p. 29–41.

SESHADHRI, C.; KOLDA, T. G.; PINAR, A. Community structure and scale-free collections of erdős-rényi graphs. **Physical Review E**, APS, v. 85, n. 5, p. 056109, 2012.

SHAH, N.; BEUTEL, A.; HOOI, B.; AKOGLU, L.; GÜNNEMANN, S.; MAKHIJA, D.; KUMAR, M.; FALOUTSOS, C. Edgecentric: Anomaly detection in edge-attributed networks. In: **IEEE 16th International Conference on Data Mining Workshops (ICDMW)**. Barcelona, Spain: IEEE, 2016. p. 327–334.

SHI, C.; LI, Y.; ZHANG, J.; SUN, Y.; YU, P. S. A survey of heterogeneous information network analysis. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 29, n. 1, p. 17–37, 2017.

VIDAL, I. C.; MENDONÇA, A. L.; ROUSSEAU, F.; MACHADO, J. de C. Protecting: An application of local differential privacy for iot at the edge in smart home scenarios. In: **XXXVIII Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)**. Rio de Janeiro, RJ, Brazil: SBC, 2020. p. 547–560.

WANG, C.; LAI, J.; YU, P. S. Neiwalk: Community discovery in dynamic content-based networks. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 26, n. 7, p. 1734–1748, 2013.

WANG, D.; LONG, S. Boosting the accuracy of differentially private in weighted social networks. **Multimedia tools and applications**, Springer, v. 78, n. 24, p. 34801–34817, 2019.

WANG, T.; BLOCKI, J.; LI, N.; JHA, S. Locally differentially private protocols for frequency estimation. In: **26th USENIX Security Symposium**. Vancouver, BC, Canada: USENIX, 2017. p. 729–745.

WANG, Y.; WU, X. Preserving differential privacy in degree-correlation based graph generation. **Transactions on Data Privacy**, NIH, v. 6, n. 2, p. 127–145, 2013.

WARNER, S. L. Randomized response: A survey technique for eliminating evasive answer bias. **Journal of the American Statistical Association**, Taylor & Francis, v. 60, n. 309, p. 63–69, 1965.

WEI, C.; JI, S.; LIU, C.; CHEN, W.; WANG, T. Asgldp: Collecting and generating decentralized attributed graphs with local differential privacy. **IEEE Transactions on Information Forensics and Security**, IEEE, v. 15, p. 3239–3254, 2020.

XIAO, Q.; CHEN, R.; TAN, K. Differentially private network data release via structural inference. In: **The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)**. New York, NY, USA: ACM, 2014. p. 911–920.

YE, M.; BARG, A. Optimal schemes for discrete distribution estimation under locally differential privacy. **IEEE Transactions on Information Theory**, IEEE, v. 64, n. 8, p. 5662–5676, 2018.

YE, Q.; HU, H.; AU, M. H.; MENG, X.; XIAO, X. LF-GDPR: A framework for estimating graph metrics with local differential privacy. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 34, n. 10, p. 4905–4920, 2022.

ZHANG, J.; CORMODE, G.; PROCOPIUC, C. M.; SRIVASTAVA, D.; XIAO, X. Private release of graph statistics using ladder functions. In: **Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data**. Melbourne, Victoria, Australia: ACM, 2015. p. 731–745.

ZHOU, N.; LONG, S.; LIU, H.; LIU, H. Structure-attribute social network graph data publishing satisfying differential privacy. **Symmetry**, MDPI, v. 14, n. 12, p. 2531, 2022.